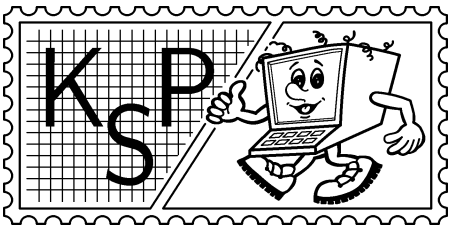
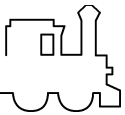


Zbierka riešených úloh

Korešpondenčného seminára z programovania

(1998 – 2006)

Fakulta matematiky, fyziky a informatiky
Univerzita Komenského, Bratislava



Korešpondenčný seminár z programovania je súťaž v tvorbe algoritmov a programovaní pre žiakov stredných škôl. Zbierka obsahuje zadania a myšlienky riešení úloh, ktoré dostali do rúk riešitelia šestnásteho až dvadsiateho tretieho ročníka tejto súťaže.

FIXME: Skôr, ako pôjde zbierka do tlače, sem treba napísať poriadne venovanie. Nie že tu zase ostane veta o tlačiarenskom škriatkovi!

Predchádzajúce vydania zbierok KSP:
1996, 1997, 2001 Michal Winczer,
2006 Michal Forišek a Jakub Kováč

Veselé púpavy odhodlane pochodovali pudingom

Ahoj, milý čitateľ!

Ak ťa ten nadpis zarazil, neboj sa. Naozaj držíš v rukách zbierku úloh Korešpondenčného seminára z programovania (KSP). Len sme týmto drobným trikom dosiahli, že skôr, ako sa pustíš do čítania samotných úloh, prečítaš si týchto pár úvodných slov. Neboj sa, neoľutuješ toto rozhodnutie. Hneď nasledujúci odsek ti jasne ukáže, že v našej knižke sa aj úvod oplatí čítať.

Súťaž!

Aj v tejto zbierke pokračuje **SÚŤAŽ**, ktorú sme vyhlásili v zbierke obsahujúcej prvých 15 ročníkov KSP. O čo ide? Spravili sme maximum pre to, aby sa ti dostala do rúk úplne dokonalá kniha. Ale život už veľakrát ukázal, že nič nemôže byť dokonalé, a tak sa nejaké chyby určite nájdu aj v tejto zbierke. Ak na nejakú narazíš, daj nám vedieť. Prvé odhalenie ľubovoľnej logickej alebo historickej chyby v tejto zbierke ocenia autori pozvaním šťastného nálezcu na kofolu či pivo[1], prípadne vlastnoručným napísaním venovania do knižky.

Zoznam už nájdených chýb a ich opráv (odborne zvaný erráta) nájdeš na adrese `http://www.ksp.sk/ksp/zbierka/`. Tam sa dočítaš aj inštrukcie, ako nám oznámiť novú nájdenú chybu.

Návod na požutie

V prvej časti tejto zbierky nájdeš samotné *zadania* úloh KSP. Mnohé z týchto zadaní sme mierne upravili oproti podobe, v akej ich dostali do rúk riešitelia. Snažili sme sa texty zadaní vyjasniť, doplnili sme do nich niekoľko obrázkov a opravili sme pár chýb v príkladoch vstupu a výstupu.

Úlohy sú zoradené podľa ročníka KSP, v ktorom boli zadané. Každá úloha má číslo $xxyz$, ktoré znamená, že úloha z sa objavila v y-tom kole xx-tého ročníka súťaže. V ročníkoch, ktoré obsahuje táto zbierka, malo KSP dve kategórie: „ostré“ (kategória O) pre pokročilých riešiteľov a „zetko“ (kategória Z) pre začiatočníkov. Úlohy kategórie Z majú pred číslom úlohy uvedené písmeno **z**.

Zadania si však môžeš čítať aj v inom poradí. Druhý spôsob objavíš na nasledujúcej strane. Jednotlivé úlohy sme rozdelili do oblastí podľa témy, či metódy riešenia. Ku každej téme uvádzame zoznam úloh, ktoré do nej patria.

Druhá časť zbierky obsahuje *riešenia* všetkých úloh. Niektoré z nich sú stručné, len jednou vetou radia, ktorým smerom sa treba vydať. Vo všeobecnosti platí, že čím je úloha ťažšia, tým podrobnejšie a presnejšie sme sa snažili napísať jej riešenie. Pri niektorých úlohách dokonca nájdeš viacero rôzne šikovných riešení. V každom prípade odporúčame najskôr skúsiť úlohu vyriešiť bez nahliadnutia do tejto časti zbierky, riešenie si prečítaj až vtedy, keď si už nevieš rady.

Takmer na konci sa pod nadpisom *prehľad metód riešenia* skrýva pár strán, na ktorých podrobnejšie popisujeme algoritmy, s ktorými sa môžeš stretnúť vo viacerých úlohách našej zbierky: medián postupnosti, prehľadávanie do hĺbky a do šírky (a ich aplikácie), Dijkstrov a Euklidov algoritmus a union-find.

No a úplne na konci sa nachádza *index*. V indexe nájdeš abecedne zoradené rôzne odborné pojmy a algoritmy, a ku každému z nich zoznam príkladov, ktoré s ním súvisia. Ak ťa teda zaujíma napríklad *prehľadávanie do hĺbky*, nie je nič ľahšie ako nájsť si v indexe, v ktorých úlohách sa dá použiť.

[1] Na pivo len plnoletých!

Tematické rozdelenie úloh

Úlohy uvedené v tejto zbierke sme sa pokúsili rozdeliť do niekoľkých škatuliek podľa témy, do ktorej patria, prípadne podľa metódy riešenia. Toto delenie je len približné – niektoré úlohy sú natoľko osobité, že ich je ťažké niekam zaradiť, v iných sa zase stretajú aj dve-tri rôzne oblasti.

Triedenie a vyhľadávanie. Väčšinou nám nestačí, že vieme údaje do počítača zadať, ale potrebujeme sa v nich vedieť aj orientovať. Samozrejme, čím efektívnejšie, tým lepšie. Často potrebujeme údaje podľa nejakého kľúča utriediť, prípadne v nich efektívne vyhľadávať. Podľa konkrétnej situácie treba zvoliť vhodnú dátovú štruktúru (napríklad utriedené pole, binárne stromy, hešovaciu tabuľku). Patria sem aj rôzne štatistické funkcie ako medián a modus.

Úlohy: 1643, z1631, z1632, 1713, 1722, 1733, 1734, z1725, 1811, 1822, 1823, 1832, z1812, z1841, z1844, 1911, 1921, 1931, 1941, z1912, z1932, z1942, z1944, 2011, 2024, 2031, z2013, z2023, z2032, z2042, 2113, 2131, 2144, z2123, z2131, 2221, z2211, z2221, z2231, 2311, 2321, 2344, z2314, z2333, z2334, z2345.

Dátové štruktúry. Do tejto kategórie sme zaradili úlohy, v ktorých ťažiskom riešenia je voľba či návrh vhodnej dátovej štruktúry – teda spôsobu, akým si budeme dáta ukladať do pamäte.

Úlohy: 1611, z1611, z1621, z1631, z1811, z1812, z1821, z1831, 1935, 2134, z2113, z2141, 2214, 2243, z2232, z2242, 2311, 2312, 2321, 2333, z2314, z2321, z2345.

Grafové algoritmy. Na riešenie týchto úloh sú potrebné poznatky z teórie grafov. Patria sem úlohy o najkratších cestách, najlacnejších kostrách, stromoch, párovaniach a mnohom inom. Bolo by ťažké čo i len vymenovať všetko, čo do tejto oblasti patrí – teória grafov je totiž jednou z najdôležitejších a najväčších oblastí teoretickej informatiky.

Úlohy: 1613, 1615, 1623, 1631, 1641, 1643, 1644, z1612, z1614, z1615, z1624, z1634, z1635, 1711, 1714, 1721, 1723, 1731, 1741, 1742, z1721, z1731, z1733, z1735, 1811, 1821, 1834, 1844, z1822, z1832, z1842, 1912, 1924, 1932, 1935, 1942, 1945, z1911, z1921, z1931, z1941, 2013, 2023, 2032, 2042, 2043, z2012, z2015, z2021, z2022, z2031, 2112, z2114, z2122, z2125, z2143, z2145, 2212, 2222, 2224, 2232, 2244, z2212, z2222, z2224, z2243, z2244, 2313, 2314, 2323, 2332, 2333, 2341, z2315, z2325, z2343.

Dynamické programovanie. Dynamické programovanie je metóda návrhu efektívnych algoritmov, pri ktorej riešenie problému pre daný vstup vypočítame z riešení toho istého problému pre niektoré iné, menšie vstupy.

Úlohy: 1613, 1614, 1621, 1633, 1634, 1642, 1712, 1732, 1735, 1742, z1722, 1814, 1815, 1831, 1834, z1833, 1914, 1933, 1944, z1934, 2014, 2024, 2031, 2034, 2121, 2132, 2142, z2111, z2134, 2211, 2213, 2221, 2232, 2242, z2244, 2331, 2341, 2343, z2341.

Greedy (pažravé) algoritmy. Do tejto témy patria úlohy, ktoré sú riešiteľné „pažravo“ – aby sme našli najlepšie riešenie, stačí každé rozhodnutie robiť tak, aby sme z toho mali v tom okamihu čo najväčší prospech.

Úlohy: z1632, 1812, 1822, 1832, z1823, 1924, 1941, z1914, z1921, z1922, 2122, 2132, 2133, 2143, 2211, 2312, z2332.

Skúšanie všetkých možností, backtracking. Sem sme zaradili úlohy, pre ktoré neexistuje, prípadne nie je známe lepšie riešenie ako vyskúšanie (skoro) všetkých možností a vybratie najlepšej z nich.

Backtracking (prehľadávanie s návratom) je metóda, ktorá sa často používa práve na generovanie všetkých možných riešení danej úlohy. Princíp backtrackingu je jednoduchý: riešenie generujeme postupne, po častiach. (Napríklad postupnosť čísel generujeme po

jednotlivých členoch.) V každom kroku nájdeme všetky možnosti, ktoré práve prichádzajú do úvahy, a následne ich (napríklad rekurzívnymi volaniami) postupne vyskúšame. Ak žiadna z nich neviedla k cieľu, vrátime sa „o úroveň vyššie“.

Úlohy: 1642, z1612, 1824, 1923, z2024, z2045, z2124.

Kombinatorika a kombinatorické algoritmy. Kombinatorické objekty, ako sú napríklad permutácie, kombinácie, partície (rozklad čísla na sčítance), či podmnožiny, sú veľmi dôležité a často sa s nimi v živote stretneme. Kombinatorické algoritmy sú postupy, ako tieto objekty usporiadať, očíslovať a efektívne generovať.

Úlohy: z1625, z1633, z1713, z1834, 1922, 2213, 2232, 2233, z2344.

Výpočtová geometria. V týchto úlohách stretnete napríklad rátanie priesečníkov, plôch, prienikov oblastí, triangulácie, konvexné obaly, vpisovanie útvarov do obdĺžnika, a ešte oveľa viac rôznych geometrických úloh, ktoré majú často uplatnenie v praxi.

Úlohy: 1612, 1622, 1632, 1633, 1635, 1724, 1743, 1744, z1732, 1813, 1823, 1833, 1843, 1913, 1923, 2014, 2022, 2033, z2023, z2033, 2113, 2123, 2134, z2112, z2133, 2223, 2234, 2324, z2324.

Počítačová grafika a semigrafika. Vedeli ste, že celá teória splajnových kriviek vznikla kvôli automobilovému priemyslu? Počítačová grafika patrí k najdôležitejším oblastiam informatiky. Úlohy nášho seminára do nej zasahujú len okrajovo, ale niekoľko sa ich za tie roky predsa len nazbieralo. Do tejto kategórie sme zaradili aj úlohy z „korytnačej“ grafiky – teda úlohy, kde pohybujúci sa bod zanecháva za sebou stopu, ktorou kreslí rôzne obrázky.

Úlohy: z1715, z1724, z1734, z1815, z1825, z1835, z1845, z2025, z2144, z2215, z2235.

Algoritmy na reťazcoch a postupnostiach. Algoritmy na reťazcoch (ľudovo zvané *stringológia*) tu boli už dlho, no v posledných rokoch ich význam prudko stúpol. Jedným z dôvodov bola samotná skutočnosť, že čoraz viac informácii a hlavne textov je dostupných v elektronickej podobe a treba v nich vedieť efektívne vyhľadávať. Druhým, menej očividným, ale o to významnejším dôvodom bol rozvoj bioinformatiky. Nie je to až také prekvapujúce, veď napríklad samotnú DNA vieme reprezentovať ako dlhočizný reťazec písmen `CGAT`...

Úlohy: z1712, z1814, z1824, 1932, 1944, 2011, 2044, z2044, 2111, 2114, 2124, 2241, z2223, z2234, z2241, 2342.

Teória hier. Kombinatorická teória hier sa zaoberá konečnými hrami s úplnou informáciou; patrí sem teda napríklad šach, piškvorky, NIM, ale nie karty (kde nevieme, čo má ktorý protihráč na ruke). Väčšina úloh, patriacich do tejto kategórie, je typu „nájdite k tejto hre pre niektorého hráča stratégiu, ktorej keď sa bude držať, nikdy neprehrá“.

Úlohy: 1715, 1725, 1735, 1745.

Matematika. Ako keby programovanie nebolo náročnou oblasťou samé o sebe, niektoré úlohy si vyžadujú poznatky z rôznych oblastí matematiky. Z teórie čísel sú to hlavne prvočísla, najväčší spoločný deliteľ a pod. Z algebry ide napríklad o riešenie sústav lineárnych rovníc. Z analýzy zase môžeme spomenúť hľadanie extrémov funkcií. A v podobnom duchu by sa dalo zapísať ešte niekoľko strán... Do tejto kategórie sme taktiež zaradili viacero jednoduchých úloh, v ktorých väčšinu riešenia tvoria jednoduché matematické výpočty.

Úlohy: z1623, z1723, z1813, z1843, 1913, 1943, z1913, z1923, z1924, z1933, z1943, 2041, z2034, 2141, z2132, z2141, z2142, 2215, 2232, z2213, z2225, z2233, 2322, 2334, 2343, z2311, z2312, z2322, z2331, z2332.

Simulácia. Jedno z mnohých využití počítača je simulácia javov „zo života“. Toto nemusí byť vždy ľahká úloha, a to ani vtedy, keď „skutočný život“ zjednodušíme na nepoznanie.

Úlohy: z2043, z2312, z2313, z2314, z2331.

Netradičné teoretické modely. Spolu s vývojom počítačov sa rozvíjala aj teoretická informatika. Vedci sa snažili formálne definovať, čo je to počítač, a osvedčenými matematickými postupmi prísť na to, čo sa na počítači spočítať dá a čo nie, prípadne čo sa dá spočítať efektívne. Niekoľko takýchto teoretických modelov sme aj v KSP riešiteľom ukázali.

Úlohy: 1615, 1625, 1635, 1645, 1825, 1835, 1845, 1915, 1925, 1945, 2015, 2025, 2035, 2045, 2115, 2125, 2135, 2145, 2225.

Kódovanie a šifrovanie. Kódovanie informácii do núl a jednotiek je staré ako počítače samé, dokonca ešte o čosi staršie. Časom prišla potreba zakódovanú informáciu prenášať, a tak vznikla kompresia a samoopravné kódy. Potreba utajenia obsahu správy pred nepovolanými osobami viedla k rozvoju šifrovania.

Úlohy: z1915, z1925, z1935, z1945, z2135, 2325.

Ad hoc. Táto latinská fráza (znamenajúca „na tento konkrétny účel“) sa zvykne používať na označenie príkladov, kde treba na riešenie objaviť špeciálny postup, ktorý sa inde nepoužíva a nijako sa nevolá. Toto je teda oblasť, kam patria veci, ktoré nevieme zaradiť nikam inam `:-)`

Úlohy: 1624, z1613, z1622, z1635, z1711, z1714, 1824, 1841, 1842, 1934, 2012, 2021, z2011, z2014, z2015, z2035, z2041, z2115, z2121, z2125, z2145, 2231, 2235, 2245, z2214, z2225, z2245, 2315, 2335, 2345, z2335, z2342.

Zadania

1611. O Kubomíre

Kubomír je nanajvýš jednotvárne miesto. Pozostáva z n štvorcových políčok usporiadaných do radu vedľa seba. V Kubomíre žije jediný druh živočícha – kubár, a jedna nadzmyslová neuchopiteľná bytosť bez tvaru, farby, chuti a zápachu – Kuboh.

Kubár má tvar kocky – vznikne, kedy a kde si Kuboh zmyslí. Kubár potom proste je… existuje… ale inak nič – ani sa nepohne. Potom si zrazu Kuboh zmyslí, že kubár zanikne a kubár naozaj zanikne. Kuboh si zavše zmyslí niekoľko nových kubárov vedľa seba, napríklad od políčka a až po políčko b. Hneď nato pribudne po jednom kubárovi na políčkach $a, a+1, a+2, \ldots, b-1, b$. Takisto si môže Kuboh zmyslieť, že zanikne po jednom kubárovi na políčkach c až d. Kuboh je neomylný, teda niečo také si zmyslí iba vtedy, keď na týchto pozíciách je po aspoň jednom kubárovi. A takto stĺpce kubárov na jednotlivých políčkach Kubomíru stúpajú a klesajú.

Z času na čas sa Kuboh filozoficky zamyslí: „A koľkože to máme kubárov na r-tom políčku?“ alebo: „A na koľkýchže políčkach máme aspoň jedného kubára?“ Z dlhej chvíle sa naučil programovať a teraz trávi dlhé chvíle dumaním nad tým, ako by to naprogramoval. Ukážte, že ste aspoň takí dobrí ako Kuboh.

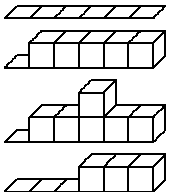
Úloha: Je daný počet políčok Kubomíru n. Na začiatku je Kubomír prázdny. Na vstup prichádzajú správy, ktoré vášmu programu oznamujú všetky zmeny v Kubomíre, alebo požadujú nejaké hodnoty o aktuálnom stave. Každá správa je jedného z nasledujúcich typov:

- „`VZNIK` a b“: na políčkach a až b vzniklo po jednom kubárovi ($1 \leq a \leq b \leq n$).
- „`ZÁNIK` c d“: na políčkach c až d zaniklo po jednom kubárovi ($1 \leq c \leq d \leq n$; môžete predpokladať, že na políčkach $c, \ldots, d$ predtým bolo aspoň po jednom kubárovi).
- „`VÝŠKA` r“: vypíšte, koľko kubárov je na r-tom políčku Kubomíru ($1 \leq r \leq n$).
- „`ASPOŇ JEDEN`“: vypíšte, na koľkých políčkach sa nachádza aspoň jeden kubár.

Úlohou vášho programu je pomocou vhodnej reprezentácie stavu Kubomíru správne a rýchlo reagovať na všetky správy a poskytovať požadované údaje.

Príklad:

```
> N = 6
> VZNIK 2 6
> VZNIK 4 4
> VÝŠKA 5
1
> ZÁNIK 2 4
> ASPOŇ JEDEN
3
```

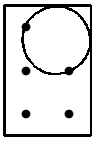


1612. O pažravej Lívii

„Méééééééé, méééééééé,“ začula teta Anastázia zo svojho domčeka. Koza Lívia sa znovu priblížila až k domčeku na koniec záhrady. A tam má teta Anastázia posadených najviac kvetov. Dnes ju už aspoň osemkrát vyhnala pásť sa na iný koniec, kde tých kvetov nie je až toľko, ale márnosť nad márnosť! Tak sa nakoniec teta Anastázia s ťažkým srdcom rozhodla Líviu priviazať. Začala v záhrade hľadať čo najväčšiu plochu takú, aby tam neboli žiadne kvetiny a aby Lívia mala dosť veľký priestor na pasenie sa.

Úloha: Teta chce priviazať kozu Líviu špagátom o kolík. Potrebuje nájsť také miesto na zapichnutie kolíka, aby sa Lívia mohla pásť na čo najväčšej ploche, ale pritom musí

zvoliť takú dĺžku špagátu, aby Lívia nemala v dosahu žiadnu kvetinu a nemohla vyjsť za hranicu záhradky. Záhrada má tvar obdĺžnika $a \times b$.



Napíšte program, ktorý načíta rozmery záhradky $a \times b$, počet kvetov n a súradnice jednotlivých kvetín $[x_1, y_1]$ až $[x_n, y_n]$ a nájde stred a polomer kružnice s najväčšou plochou, ktorá leží celá v záhradke a neobsahuje žiadnu kvetinu.

Príklad: Pre záhradku s rozmermi $a = 3.0$, $b = 2.0$ a päť kvetov ($n = 5$) so súradnicami $[0.5, 0.5]$, $[0.5, 1.5]$, $[1.5, 0.5]$, $[1.5, 1.5]$, $[2.5, 0.5]$ je výsledkom kružnica so stredom $[2.225, 1.225]$ a polomerom 0.775.

1613. O predvolebných mítingoch

Kráľ Burundi sa díval na klesajúci graf svojich preferencií. Nie že by to bol dôvod na znepokojenie, ale bolo mu z toho smutno. I dal si zavolať hlavného radcu a spýtal sa ho, čo s tým. Radca poradil: „Ľudia majú radi zmenu a vaše Veličenstvo sedí na tróne už desať rokov. Ponúknite im ju, usporiadajte voľby!“ Kráľ sa zhrozil, keď mu však jeho poradca vysvetlil, že bude len jeden kandidát, nadšene súhlasil.

Onedlho si už kráľ prezeral vypracovaný harmonogram predvolebných mítingov. Na zozname vždy bolo mesto a deň, kedy sa tam bude konať míting. Keď sa kráľ prizrel lepšie, s hrôzou zistil, že mítingov je príliš veľa a že sa nestihne všetkých mítingov zúčastniť. Burundi je totiž veľká krajina so zlými cestami, preprava z jedného mesta do iného môže trvať aj niekoľko dní. Kráľ neváhal a sľúbil okamžite 50 percent akcií Západoburundijských Železiarní tomu, kto mu nájde takú trasu predvolebného výjazdu, aby stihol čo najviac mítingov.

Úloha: Burundi má n miest očíslovaných 1 až n. Daný je počet miest n a počet ciest medzi nimi c. Všetky cesty sú obojsmerné. Pre každú cestu sú dané čísla miest, ktoré spája, a počet dní potrebný na jej prejdenie autom.

Ďalej dostanete počet mítingov m a pre každý míting informáciu v ktorom meste a v ktorý deň sa koná. Kráľ je v deň 0 v hlavnom meste (mesto č. 1). Kráľ sa na mítingoch zdrží iba zanedbateľný čas, jeho predvolebná cesta nemusí končiť v hlavnom meste.

Napíšte program, ktorý načíta zoznam burundijských ciest a zoznam mítingov a zistí, koľko najviac mítingov môže kráľ stihnúť.

Príklad:

Vstup:

```
4 5
1 2 4
1 3 3
2 3 4
2 4 5
3 4 2
5
1 3
2 4
3 9
1 12
4 17
```

Výstup:

```
Dajú sa stihnúť najviac
4 mítingy.
```

(V ľavom stĺpci je popis krajiny: mestá sú 4, ciest medzi nimi je 5, nasledujú popisy ciest. V prostrednom stĺpci je počet a popis mítingov. Optimálny rozvrh pre kráľa: vie stihnúť buď všetky mítingy okrem prvého alebo všetky mítingy okrem druhého. Všimnite si, že medzi štvrtým a piatym mítingom musí z mesta 1 do mesta 4 cestovať cez mesto 3.)

1614. O výskumníčke Janke

Janka je výskumníčka a pracuje v laboratóriu na výskum inteligencie živočíšnych druhov. Inak je to malá biela myška, ktorá má rada syr. Po niekoľkých týždňoch trpezlivej práce sa jej podarilo vycvičiť si svojho človeka. Vždy, keď Janka vybehne po rebríku, dostane kúsok chutného syra. Teraz však nie je hladná.

Okolo seba má porozhadzovaných niekoľko kociek s písmenami. Väčšinu upratala kdesi do kúta, ale zopár ich tu stále ostalo. Len tak na skúšku ich usporiadala tak, aby písmená

na nich tvorili palindróm, t.j. slovo, ktoré je rovnaké, či ho čítate spredu alebo odzadu. Aké bolo jej prekvapenie, keď o chvíľu prišiel človek s kusom ementálu a hŕbou ďalších kociek. Poukladal ich do radu a nenápadne sa vzdialil. Janke bolo hneď jasné, čo od nej človek chce. Medzi poukladané kocky má povsúvať niekoľko ďalších tak, aby vznikol palindróm. Ihneď začala premýšľať, koľko najmenej kociek bude treba do radu vsunúť (kocky sú ťažké a Janka sa nechce príliš namáhať).

Úloha: Napíšte program, ktorý načíta vstupné slovo $w = w_1 w_2 \dots w_n$ a zistí, koľko najmenej písmen musíme pridať, aby z neho vznikol palindróm. Nové písmená môžeme vsúvať medzi ľubovoľnú dvojicu susedných písmen, na začiatok aj na koniec. Vypíšte jeden takto vzniknutý palindróm.

Vstup:	Výstup:	
`abab`	`babab`	*(jedno pridané písmeno)*
`obyamamalybk`	`kobylamamalybok`	*(tri pridané písmená)*

1615. O profesorovi Indigovi a víle Amálke I

Iste ste si všimli, že dobrý koniec mnohých rozprávok majú na svedomí rôzne dobré víly a im podobné živly, ktoré, keď je najhoršie, dajú hlavnému hrdinovi dobrú radu. Všimol si to aj profesor Indigo a hneď vymyslel, ako by sa dala táto ich dobrá vlastnosť využiť. Indigo si uvedomil, že aj keď počítače vedia riešiť mnoho úloh, niekedy ani ony nevedia, kam z konopí. Občas by sa aj počítaču hodila nejaká dobrá rada. Profesor sa rozhodol skonštruovať prefíkaný počítač (PP), ktorý bude ťažiť zo spolupráce s dobrou vílou. Dohodol sa s vílou Amálkou, že bude pomáhať prefíkanému počítaču s výpočtami. Tak vznikol počítač $PP_{\texttt{Amálka inside}}$.

Prefíkaný počítač vie riešiť úlohy, na ktoré je odpoveď `ÁNO`/`NIE`. Profesor preň spravil aj kompilátor Pascalu (resp. C/C++). Ten ale zatiaľ nepodporuje žiadne príkazy na prácu so vstupno-výstupnými zariadeniami (ako obrazovka, klávesnica alebo disk). Na odovzdávanie vstupných údajov programu sa používajú parametre odovzdávané pomocou rezervovaného slova **program**. Aby bolo možné ako parametre odovzdávať aj zložené typy, povolil deklarácie typov a konštánt pred slovom **program**. Napríklad

const $MAX = 10$;
type bod = **record** x, y : **integer**; **end**;
program $MojPrg(B : bod;\ p$: **array** $[1\,..\,MAX]$ **of byte**);

je hlavička programu, ktorý na vstupe dostane jeden údaj typu bod a pole 10 bajtov. Na ukončenie programu slúžia dva špeciálne príkazy *fail* a *accept*. Príkaz *accept* znamená, že program dá odpoveď `ÁNO`. Príkaz *fail* znamená čosi ako „nepodarilo sa mi dať odpoveď `ÁNO`“ (presnejší popis ďalej).

Poslednou novou konštrukciou je príkaz **choose**, ktorý má rovnakú štruktúru ako príkaz **case**, až na to, že neobsahuje žiadny výraz, podľa ktorého sa program vetví. Napríklad

choose
  1 : $príkaz_1$;
  2 : **begin** $príkaz_2$; $príkaz_3$; **end**;
end;

Vždy, keď $PP_{\texttt{Amálka inside}}$ narazí pri vykonávaní programu na príkaz **choose**, víla Amálka rozhodne, ktorá vetva sa vykoná.

V jazyku C/C++ sa vstupné údaje odovzdávajú ako parametre funkcie *main*. Príkazy *fail* a *accept* fungujú podobne ako **return**, až na to, že nemajú žiadne parametre a ukončujú program. Konštrukcia **choose** je analógiou konštrukcie **switch** bez riadiaceho výrazu – Amálka rozhodne, na ktoré návestie **case** treba skočiť. Pre iné programovacie jazyky si navrhnite obdobné konštrukcie sami.

Amálka riadi beh programu tak, aby vždy, keď to je možné, bola odpoveď `ÁNO`. To znamená, že vždy, keď pre daný vstup existuje taká postupnosť vetvení **choose**, ktorá dovedie program k príkazu *accept*, $PP_{\texttt{Amálka inside}}$ dá odpoveď `ÁNO`. Ak takáto postupnosť neexistuje, $PP_{\texttt{Amálka inside}}$ dá odpoveď `NIE`. (Navyše okrem radenia pri výpočte totiž Amálka, ako správna víla, vie zistiť, ak žiadna postupnosť vetvení nevedie k príkazu *accept*, a vtedy okamžite zastaví výpočet.) $PP_{\texttt{Amálka inside}}$ dá teda odpoveď `NIE`, ak každá z postupností vetvení **choose** vedie buď k príkazu *fail*, k inému spôsobu skončenia programu, alebo výpočet pri tejto postupnosti beží donekonečna.

Profesor Indigo napísal prvý program pre prefíkaný počítač. Tu je:

```
const MAX = 100;
type pole = array[0..MAX-1, 0..MAX-1] of integer;
program PrvyProgram(n : integer; p : pole);
var x, y : integer;

 begin
   x := 0; y := 0;
   repeat
     choose
       vlavo : x := x - 1;
       vpravo : x := x + 1;
       hore : y := y - 1;
       dole : y := y + 1;
     end;
     if (x < 0) or (x ≥ n) or (y < 0) or (y ≥ n) then fail;
     if (p[x, y] ≠ 0) then fail;
     if (x = n - 1) and (y = n - 1) then accept;
   until false;
 end.
```

Profesor si však nie je celkom istý, či všetko pracuje tak, ako má. Pre kontrolu by potreboval program, ktorý robí presne to isté, ale nevyužíva schopnosti víly Amálky.

Úloha: Napíšte čo najefektívnejší program, ktorý bude robiť presne to isté, čo profesorov – t.j. pre rovnaký vstup musia oba programy dať rovnaký výstup. Váš program musí vždy skončiť príkazom *accept* alebo *fail*. Keďže nemáte k dispozícii vílu Amálku, nesmiete použiť konštrukciu *choose*.

1621. O magickom symbole

Jedného dňa sa miestny kúzelník Kin Lezuk vážne zamyslel. Potreboval totiž zistiť, koľko najmenej dní mu potrvá, kým aktivuje celý magický symbol okolo dediny na ochranu pred temnými silami. Magický symbol je zložený z n magických kameňov, ktoré ležia na kružnici, ktorej stredom je dedina. Aktivácia prebieha nasledovne. Kin sa ráno zobudí a vydá sa k niektorému kameňu. Potom postupuje po kružnici celý deň rovnakým smerom a zaklína všetky magické kamene po ceste. Po zakliatí niekoľkých kameňov sa vráti domov (čiže ide od posledného zakliateho kameňa do dediny). Kin však môže putovať najviac h hodín denne, aby bol pred zotmením doma. Aktivácia je ukončená, ak zaklial každý kameň aspoň raz (na poradí zaklínania jednotlivých kameňov nezáleží).

Úloha: Napíšte program, ktorý vypočíta, koľko dní Kin potrebuje na aktiváciu kúzla. Vstupom programu bude počet hodín h (maximálna doba trvania každej Kinovej cesty), počet magických kameňov n a ďalej pre každý kameň čas, koľko trvá cesta z dediny ku nemu a koľko trvá cesta od tohto kameňa do dediny. Taktiež pre každé dva susedné kamene na kružnici dostane program čas cesty medzi nimi (jedným aj druhým smerom). Čas potrebný na zakliatie kameňa je zanedbateľný.

Príklad: Pre $h = 3$, $n = 3$ a časy v tabuľke mu to bude trvať 2 dni.

	Tam	Späť		Tam	Späť
dedina – kameň 1	1	3	kameň 1 – kameň 2	1	2
dedina – kameň 2	2	1	kameň 2 – kameň 3	1	2
dedina – kameň 3	2	1	kameň 3 – kameň 1	4	1

1622. Ako Janko s Marienkou bytovú otázku riešili

Iste si všetci živo pamätáte rozprávku o perníkovej chalúpke. Áno, tak ako vy ste boli deti, aj Janko a Marienka boli deti a medzičasom vyrástli. Napriek tomu, že sa doteraz poctivo zúčastňujú každého zbierania jahôd a hádzania Ježibaby do pece, nemajú si za čo kúpiť byt. A tak raz, keď už za Ježibabou zatvárali na peci vrátka, pošepla Marienka Jankovi: „Janíčko, veď je to celkom dobrá chalúpka, čo keby sme sa tu zabývali?“ Nelenili teda a začali opravovať svoj nový domček, aby im nezatekalo (viete predsa, že predtým z chalúpky odlomili a zjedli niekoľko perníkov). Marienka vyvaľkala cesto a Janko vykrajoval rôzne tvary. Už mali takmer všetko hotové a z cesta na doske zostali len zvyšky, keď si všimli, že Janko zabudol vykrojiť perník na vetrák v tvare kruhu. Teraz sedia a dumajú, či sa vôbec kruh z tých zvyškov vykrojiť dá.

Napíšte program, ktorého vstupom je tvar zvyšku cesta a polomer vetráku r. Zvyšok cesta je n-uholník zadaný súradnicami svojich vrcholov postupne po obvode proti smeru hodinových ručičiek. Výstupom bude odpoveď `ÁNO`, ak sa kruhový perník dá zo zvyšku cesta vykrojiť, alebo `NIE`, ak sa nedá.

Príklad: Pre cesto tvaru 6-uholníka s vrcholmi $[0,0]$, $[3,1]$, $[5,0]$, $[3,2]$, $[3,4]$, $[1,3]$ a vetrák s polomerom $r = 1$ je odpoveď `ÁNO`. Pre cesto tvaru 3-uholníka s vrcholmi $[0,2]$, $[8,0]$, $[2,4]$ a vetrák s polomerom $r = 2$ je odpoveď `NIE`.

1623. O bále

Kde bolo, tam bolo, za siedmimi horami a siedmimi dolami, kde sa voda sypala a piesok sa lial, bolo raz jedno kráľovstvo. V tomto kráľovstve sa každoročne na kráľovskom zámku konal veľký ples. Pozvaných bolo mnoho hostí a nemálo dvojíc si zahovorilo tanec. Prastará veštba však varuje, že posledný bál nesmie mať viac tanečných kôl ako je zlatých jabĺčok na kráľovskej jabloni. S každým tanečným kolom odpadne jedno jabĺčko a ak by sa tancovalo aj po spadnutí posledného jabĺčka, krajinu by postihlo nešťastie.

Každoročne sa stáva predmetom špekulácií, či sa podarí splniť želania všetkých tanečníkov – teda, či sa každej dvojici, ktorá chce spolu tancovať, podarí tancovať aspoň jedno kolo. V tom istom kole môže tancovať ľubovoľne veľa dvojíc za predpokladu, že každý pán tancuje najviac s jednou dámou a každá dáma najviac s jedným pánom.

Úloha: Je daný počet zlatých jabĺčok na kráľovskej jabloni p, ďalej k – počet dvojíc, ktoré si dohovorili tanec a jednotlivé dvojice – vždy najskôr meno pána, potom meno dámy. Zistite, či je možné uspokojiť želania všetkých tancujúcich. V prípade, že to je možné, vypíšte ľubovoľný vyhovujúci rozvrh tancov. Môžete predpokladať, že mená sú jednoslovné.

Príklad:

Vstup:

```
3 7
Janíček Zlatovláska
Bajaja Zlatovláska
Bajaja Marienka
Popolvár Zlatovláska
Bajaja Snehulienka
Janíček Marienka
Popolvár Snehulienka
```

Výstup:

```
ÁNO
1: Janíček Zlatovláska,
   Bajaja Marienka,
   Popolvár Snehulienka
2: Janíček Marienka,
   Bajaja Zlatovláska
3: Popolvár Zlatovláska,
   Bajaja Snehulienka
```

VSTUP:
```
2 5
Janíček Zlatovláska
Bajaja Zlatovláska
Bajaja Marienka
Janíček Marienka
Popolvár Zlatovláska
```

VÝSTUP:
```
NIE
```

1624. Populárna prednáška

Profesor Indigo sa vďaka svojmu novému počítaču stal slávnym a jeho prednášku na univerzite si zapísalo veľmi veľa študentov. Profesor od svojich študentov vyžadoval, aby chodili na všetky prednášky a pri príchode sa zapísali do zoznamu prítomných. Ako správny informatik sa neobťažoval s nejakými menami, ale hneď na začiatku roka očísloval študentov číslami $1, 2, \ldots, n$ a študenti do zoznamu zapisovali svoje čísla v dvojkovej sústave. Všetko išlo ako po masle, až raz na jednej prednáške jeden študent chýbal. Profesor si to hneď všimol a rozhodol sa, že musí zistiť, ktorý to bol. Rozhodol sa napísať si program. Keďže víla Amálka mala ten deň práve dovolenku, musel použiť normálny počítač. Zoznam čísel študentov bol však veľmi dlhý, preto sa mu ho nechcelo do počítača prepisovať celý.

ÚLOHA: Napíšte program, ktorý načíta počet všetkých študentov n a potom bude klásť profesorovi otázky typu „`Zadajte i-ty bit (sprava) z j-teho cisla v zozname`“ ($1 \leq j \leq n-1$) a na základe odpovedí vypíše číslo študenta, ktorý nebol na prednáške. Môžete predpokladať, že v zozname je binárny zápis každého z čísel od 1 po n okrem jedného. Ak má j-te číslo menej ako i bitov, profesor zadá 0. Snažte sa, aby váš program nekládol profesorovi príliš veľa otázok.

Predpokladajme, že $n = 3$ a v zozname sú čísla 11 a 1, ktoré v desiatkovej sústave zodpovedajú 3 a 1. Chýba teda číslo 10 (2). Práca programu môže vyzerať napríklad takto:

```
Zadajte N
> 3
Zadajte 1. bit (sprava) z 1. cisla v zozname
> 1
Zadajte 1. bit (sprava) z 2. cisla v zozname
> 1
Vysledok je 10
```

1625. O profesorovi Indigovi a víle Amálke II

Profesor Indigo zostavil svoj nový počítač $PP_{\texttt{Amálka inside}}$ (popísaný v úlohe 1615) hlavne preto, lebo sa mu zdalo, že to zjednoduší mnohé programy. Časom však objavil aj ďalšiu výhodu – mnohé výpočty boli omnoho rýchlejšie ako na bežnom počítači:

V prípade, že pre daný vstup neexistuje postupnosť vetvení **choose**, ktorá vedie k príkazu *accept*, víla to zistí hneď na začiatku, takže samotný počítač vôbec nemusí vykonávať žiadne výpočty. Ak je teda odpoveď `NIE`, čas výpočtu je 0. V prípade, že odpoveď je `ÁNO`, výpočet prebehne a víla zvolí v každom príkaze **choose** takú vetvu, že dovedie výpočet k príkazu *accept*. Postupností vetvení, ktoré vedú k príkazu *accept* však môže byť viacero a môžu sa veľmi líšiť dĺžkou zodpovedajúceho výpočtu. Víla Amálka sa však vďaka svojej vynikajúcej intuícii vždy rozhodne tak, aby bol výpočet najkratší možný.

Uvažujme napríklad program uvedený v úlohe 1615. Ak pre daný vstup neexistovala cesta z políčka $[0, 0]$ na políčko $[n-1, n-1]$ idúca iba po nulách, tak výpočet trval nulový čas. V opačnom prípade bol čas výpočtu úmerný dĺžke najkratšej cesty po nulách medzi týmito dvoma políčkami. Pre niektoré rozmiestnenia nenulových prvkov sa však môže stať, že dĺžka najkratšej cesty bude približne $n^2/2$, takže časová zložitosť programu je $O(n^2)$ aj na počítači $PP_{\texttt{Amálka inside}}$.

Dobrý priateľ profesora Indiga, doktor Jones, je archeológ. K jeho najnovším úspechom patrí nález vzácneho papyrusového zvitku, ktorý patril hlavnému správcovi ciest Nacestejamasovi. Tento papyrus obsahuje zoznam všetkých ciest v krajine. Každá cesta v zozname spája dve mestá. Tento zoznam je o to cennejší, že je to prvá písomná zmienka o mnohých mestách. Bohužiaľ, všetky názvy miest v papyruse boli doktorovi Jonesovi úplne neznáme. Rozhodol sa, že zoznam porovná s najstaršou dostupnou mapou krajiny. Dúfa, že sa mu takto podarí určiť, akým mestám zodpovedali názvy na papyruse. Doktor Jones vychádzal z týchto predpokladov:

- Každé mesto, ktoré je spomenuté v Nacestejamasovom papyruse, sa nachádza aj na mape, na mape však môžu byť aj ďalšie mestá, ktoré vznikli neskôr ako papyrus.
- Nové mestá mohli vznikať buď na už existujúcej ceste, v tom prípade vznikajúce mesto rozdelilo cestu, na ktorej vzniklo, na dve cesty; alebo mimo cesty. Cesty mohli vzniknúť medzi ľubovoľnou dvojicou miest.
- Každá cesta uvedená v papyruse je aj na mape; mohla byť však rozdelená pri vzniku miest na niekoľko častí.
- Medzi žiadnou dvojicou miest nevedie nikdy viac ako jedna cesta.

Porovnávanie mapy so zoznamom ciest je veľmi zdĺhavá práca. Doktorovi sa doteraz ani nepodarilo zistiť, či sa každému názvu mesta na papyruse dá priradiť nejaké mesto na mape v súlade s predpokladmi. Profesor Indigo teda ponúkol doktorovi Jonesovi, že napíše program pre prefíkaný počítač $PP_{\texttt{Amálka inside}}$, ktorý slávnemu archeológovi pomôže. Indigo však večne nemá čas, a tak táto úloha ostáva vám...

Úloha: Očíslujme mestá na mape číslami $1, \ldots, n_1$ a mestá v papyruse číslami $1, \ldots, n_2$. Napíšte program pre $PP_{\texttt{Amálka inside}}$, ktorý na vstupe dostane počet miest na mape n_1 a počet ciest na mape m_1, počet miest na papyruse n_2 a počet ciest na papyruse m_2 ako aj zoznamy ciest na mape a na papyruse (každá cesta je určená počiatočným a koncovým mestom). Program má zistiť, či je možné každému číslu mesta z papyrusu priradiť číslo mesta na mape tak, aby toto priradenie spĺňalo Jonesove predpoklady. Presnejšie, váš program má mať hlavičku

const $MAX = 1000$;
type $cesta$ = **record** z, k : **integer**; **end**;
$zoznam\_ciest$ = **array** $[1\,..\,MAX]$ **of** $cesta$;
program $Jonesova\_mapa(n_1, m_1, n_2, m_2$: **integer**; $ZC_1, ZC_2 : zoznam\_ciest)$;

Váš program má dať odpoveď `ÁNO` (za výdatnej pomoci víly Amálky) práve vtedy, keď existuje zodpovedajúce priradenie miest na papyruse miestam na mape. Ak takéto priradenie neexistuje, program má dať odpoveď `NIE`. Skúste tiež odhadnúť časovú zložitosť vášho programu na počítači $PP_{\texttt{Amálka inside}}$.

Príklad: Pre $n_1 = 5$, $m_1 = 5$, $n_2 = 3$, $m_2 = 3$, $ZC_1 = ((1,2),(2,3),(3,4),(4,1),(2,5))$ a $ZC_2 = ((1,2),(2,3),(1,3))$ má dať $PP_{\texttt{Amálka inside}}$ výstup `ÁNO`. V tomto prípade existuje viacero možných priradení, napríklad mestu 1 z papyrusu priradíme mesto 3 z mapy, mestu 2 z papyrusu priradíme mesto 4 a mestu 3 z papyrusu priradíme mesto 1 z mapy.

1631. O svokre

Medzi Bonifácove vášne patrí na poprednom mieste cyklistika. Stalo sa však, že ho raz prišla navštíviť svokra. Keď už u neho bývala vyše mesiaca a terorizovala ho neustálym vysávaním, upratovaním a umývaním riadu, rozhodol sa zúfalý Bonifác konečne konať. Navrhol svokre bicyklový výlet. Svokra ihneď súhlasila, netušiac, čo činí. Bonifác totiž sedí nad mapami v snahe nájsť taký cieľ ich bicyklovej vychádzky, ktorý by netrénovaná svokra nezvládla.

Úloha: Mapa je reprezentovaná poľom rozmerov $r \times s$, pre každé políčko je daná jeho nadmorská výška v_{ij} v metroch. Ďalej je dané číslo k, ktoré určuje, koľko metrov zvládne svokra dokopy vystúpať počas jednej bicyklovej vychádzky. Na záver sú dané súradnice $[r_z, s_z]$ políčka, kde Bonifác so svokrou začínajú cyklotúru.

Počas výletu prechádzajú iba medzi políčkami susediacimi hranou a medzi dvoma susediacimi políčkami cesta buď rovnomerne stúpa, alebo rovnomerne klesá, alebo vedie vodorovne. Vodorovné a klesajúce úseky pre svokru nepredstavujú žiadnu námahu, do celkového stúpania sa teda započítavajú iba stúpajúce úseky. Napíšte program, ktorý nájde taký cieľ cesty, do ktorého sa svokra nedostane. Predpokladajte, že svokra si pre daný štart a cieľ vždy nájde cestu s minimálnym celkovým stúpaním.

Príklad:

Vstup:

```
4 4
 80  60  80  120
 10  75  40   10
 20  20  55   12
130  20  10   62
110
3 2
```

Výstup:

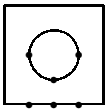
```
koniec: [1,4]
prevýšenie: 115
```

(Výlet začína vo výške 20 a končí vo výške 120. Trasa s najmenším celkovým prevýšením vedie cez políčka s výškou 55, 40 a 80.)

1632. Ako si Janko s Marienkou živnosť založili

„Hotovo," povedal Janíčko, keď sa mu podarilo nainštalovať posledný kúsok perníka na vetrák. Netrvalo dlho a po celom okolí sa roznieslo, akýže to majú Janko s Marienkou domček. Detváky z dediny chodili obdivovať všetky tie dobroty. I napadlo ich, čo tak si živnosť zriadiť a detvákom z dediny perníky i sladké medovníčky predávať, aby im z šibalstva z novej chalúpky nebodaj neodjedli. Nuž oprášila Marienka sladké recepty starej matere a zamiesila plné koryto cesta s hrozienkami. Celý deň obaja valkali, vykrajovali, cesto do pece strkali, upečené koláče z pece vyťahovali. Večer im ostal iba jeden kus cesta tvaru obdĺžnika $a \times b$. Janko schytil starú známu kruhovú formičku, čo ňou kedysi vetrák vykrajoval a poďho vykrojiť si taký kus, ktorý by čo najviac hrozienok obsahoval. Prikladal Janko formičku tak i onak, ale stále sa mu nejak videlo, že by on aj viac hrozienok mať mohol.

Úloha: Napíšte program, ktorého vstupom sú rozmery zvyšku cesta $a \times b$, polomer formičky r, počet hrozienok n a súradnice jednotlivých hrozienok $[x_1, y_1], [x_2, y_2], \ldots, [x_n, y_n]$ a ktorý nájde stred kruhu s polomerom r, ktorý celý leží v ceste a obsahuje maximálny možný počet hrozienok (hrozienka na hranici kruhu patria dovnútra kruhu).



Príklad: Pre $a = 4.0$, $b = 4.0$, $r = 1.0$ a hrozienka na súradniciach $[1.0, 0.0]$, $[2.0, 0.0]$, $[3.0, 0.0]$, $[2.0, 1.0]$, $[1.0, 2.0]$, $[3.0, 2.0]$ má hľadaný kruh stred v bode $[2.0, 2.0]$ a obsahuje 3 hrozienka (pozri obrázok).

1633. Santo, Banto a parcela

Banto sa nedávno opäť vybral do krčmy vo Vyšnej Klondike. Aké bolo Santovo prekvapenie, keď sa ani nie o dve hodiny vrátil triezvy, ale o to smutnejší. V krčme sa totiž dozvedel, že vyšlo nové nariadenie, ktoré by mohlo ohroziť ich parcelu.

Santo s Bantom vlastnia parcelu tvaru konvexného n-uholníka. V každom jeho vrchole je zatlčený kolík. Nie je to bohvieaká parcela, ale za tie roky, čo na nej strávili ryžovaním zlata, im prirástla k srdcu. Nové nariadenie hovorí toto:

0. Ruší sa staré vymedzenie parciel.
1. Každá nová parcela musí mať tvar trojuholníka.
2. Hranice parcely sa vyznačujú špagátom. Špagát môže viesť medzi ľubovoľnou dvojicou kolíkov.

3. Nie je dovolené zatĺkať nové kolíky za účelom vyznačenia hranice parcely.
4. Špagáty vyznačujúce hranice parciel sa nesmú nikde križovať.
5. Koncom týždňa budú všetky parcely, ktoré nebudú mať tvar trojuholníka, prevedené pod správu šerifského úradu.

Santa napadlo, že by mohli kúpiť špagát a pokúsiť sa ho natiahnuť medzi kolíky tak, aby rozdelili parcelu na trojuholníky. Tak by sa mohli vyhnúť tomu, aby bola ich parcela zhabaná v prospech šerifského úradu. Avšak skôr ako pošle Banta do obchodu, rád by vedel, ako vlastne má špagáty naťahovať, aby minul čo najmenej špagátu a koľko špagátu vlastne potrebuje. A tak, kým Banto smutne sedí na peľasti a čumí do steny, Santo sedí nad nákresom parcely a snaží sa zistiť, ako najvýhodnejšie natiahnuť špagáty.

Úloha: Na vstupe je dané číslo n a súradnice n vrcholov $[x_1, y_1], [x_2, y_2], \ldots, [x_n, y_n]$ parcely vymenovaných proti smeru hodinových ručičiek. Napíšte program, ktorý poradí Santovi, medzi ktorými dvojicami kolíkov má viesť špagát tak, aby ho minul čo najmenej a vypočíta celkovú dĺžku potrebného špagátu.

Príklad: Pre $n = 5$ a vrcholy so súradnicami $[0, 2]$, $[2, 0]$, $[6, 2]$, $[4, 5]$ a $[3, 5]$ treba špagát natiahnuť medzi kolíkmi 2–5, 3–5, 1–2, 2–3, 3–4, 4–5 a 5–1. Dĺžka špagátu je spolu približne 25.490.

1634. O reforme

Za čias panovania kráľa Klohodrabata VII. vládla na Kiribati anarchia. Po jeho smrti sa jeho syn Kiripraluk III., následník trónu, rozhodol, že kráľovstvo pozdvihne k rozkvetu. Zvolal mudrcov z celej krajiny, aby spolu vypracovali plán obnovy kráľovstva. Po dôkladnej analýze súčasného stavu dospeli mudrci k záveru, že najpv je treba pozdvihnúť úroveň vzdelania (čo iné už od mudrcov môžete čakať).

Tak Kiripraluk navštívil kráľovskú knižnicu. To čo uvidel ho veru nepotešilo. Bola tam jedna polica zaprášených kníh a podriemkávajúci knihovník v kúte. „Kde sú ostatné knihy?!“ čudoval sa kráľ. Dozvedel sa, že žiadne ďalšie nie sú a aj z týchto je iba po jednom rukopise. Tak sa rozhodol dať rozmnožiť aspoň tie knihy, ktoré mali. Najal všetkých pisárov, ktorí v krajine ešte zostali, a prikázal im mať o týždeň z každej knihy ďalšiu kópiu.

Úloha: Je daný počet kníh n, počet pisárov k ($k \leq n$) a počty strán $p_1, \ldots, p_n$ jednotlivých kníh v poradí, v akom sú knihy na polici uložené zľava doprava. Každému pisárovi pridelíme niekoľko kníh uložených na polici vedľa seba. Každú knihu celú prepisuje jeden pisár, každý pisár musí prepísať aspoň jednu knihu. Určite, ako rozdeliť knihy medzi pisárov tak, aby celkový čas prepisovania bol minimálny.

Čas prepisovania jedného pisára je úmerný celkovému počtu strán, ktoré má prepísať. Celkový čas prepisovania je čas, ktorý potrebuje pisár, ktorý má najviac práce.

Príklad:

Vstup:
```
9 3
100 200 300 400 500 600 700 800 900
```

Výstup:
```
pisár 1: 100 200 300 400 500
pisár 2: 600 700
pisár 3: 800 900
celkový čas: 1700 strán
```

1635. O profesorovi Indigovi a víle Amálke III

Profesor Indigo si listoval v zbierke úloh KSP a keď si čítal úlohy prvého kola deviateho ročníka, narazil na takúto:

913. O koláči.

Na obdĺžnikovom plechu upiekli (mimochodom nie veľmi dobrý) koláč rozmerov $x \times y$ (rozmery koláča a plechu boli rovnaké) a rozrezali ho na k kusov. Všetky kusy boli obdĺžnikového tvaru, ale ich rozmery sa navzájom značne líšili. Hostia sadli ku stolu s koláčom, ale keďže sa veľmi nemali k jedeniu, začali si krátiť dlhú

chvíľu skladaním nakrájaných kúskov koláča do pôvodného tvaru, teda do tvaru, aký mal na plechu. Pritom im najväčšie problémy robilo zistenie rozmerov plechu, na ktorom sa koláč piekol. Po dlhej a namáhavej práci sa dohodli na tom, že táto úloha nie je súca pre nich, ale pre účastníkov KSP.

Úloha: Je daný počet k nakrájaných kúskov koláča a pre každý kúsok jeho rozmery x_i, y_i. Napíšte program, ktorý zistí, aké mohli byť rozmery plechu $x \times y$, na ktorom bol daný koláč upečený (stačí jedno riešenie). Všetky rozmery sú celočíselné.

Výsledkom programu sú len rozmery x a y, nežiada sa nájsť rozloženie daných kúskov na plechu! Pri pečení je celý plech pokrytý cestom.

Profesor sa rozhodol napísať program pre $PP_{\texttt{Amálka inside}}$ (pozri úlohu 1615), ktorý by pre dané rozmery koláča x a y, počet kúskov k a zoznam rozmerov jednotlivých kúskov koláča zistil, či sa kúsky koláča dajú naukladať na plech. Ako na potvoru má práve zlomený malíček a nemôže programovať...

Úloha: Napíšte program pre $PP_{\texttt{Amálka inside}}$, ktorý za pomoci víly Amálky zistí, či sa kúsky koláča dajú na plech naukladať. Váš program by (v Pascale) mal mať hlavičku:

const $MAX = 1000$;
type $obdlznik$ = **record** a, b : **integer**; **end**;
$zoznam\_kuskov$ = **array**$[1\,..\,MAX]$ **of** $obdlznik$;
program $O\_kolaci(x, y, k$: **integer**; Z : $zoznam\_kuskov)$;

1641. O Kiribatskej mape

Kiribatský kráľ KiriKiri investoval obrovské peniaze do vesmírneho programu, a to mu vytýkajú všetci jeho odporcovia. Aby utíšil hlas nespokojného ľudu, KiriKiri sa rozhodol, že vyrieši legendárnu otázku o počte Kiribatských ostrovov. A tak si dal urobiť satelitný snímok s obrovským rozlíšením, na ktorom sa dá rozoznať, kde je more a kde je súš. Napíšte pre KiriKiriho program, ktorý mu spočíta, koľko ostrovov má Kiribatské súostrovie. Vzhľadom na veľký rozmer mapy si nemôžete pamätať celú mapu, ale iba niekoľko riadkov z nej.

Úloha: Váš program má k dispozícii vstupný súbor. Tento súbor v prvom riadku obsahuje dve celé čísla r, s, kde r je počet riadkov a s počet stĺpcov mapy. Nasleduje r riadkov, každý z nich obsahuje s znakov `0` alebo `1`, kde `0` označuje more a `1` súš.

Počet riadkov nie je nijako zhora obmedzený (súčin $r \cdot s$ môže byť omnoho väčší ako veľkosť dostupnej pamäti), ale môžete predpokladať, že niekoľko riadkov sa do pamäti zmestí. Výstupom vášho programu má byť počet ostrovov v súostroví. Ostrovy sa môžu dotýkať rohmi.

Príklad:

Vstup:
```
4 5
11110
10110
11000
00111
```

Výstup:
```
Súostrovie má 2 ostrovy.
```

1642. O tajnej službe

Nemenovaná tajná služba dostala príkaz monitorovať pohyb podozrivých osôb. Sledované osoby by samozrejme nemali tušiť, že ich niekto sleduje. Na tento účel vymyslel riaditeľ onej tajnej služby systém SPS – satelitný pozorovací systém.

Každý satelit systému SPS má schopnosť sledovať obdĺžnik územia s rozmermi $a \times b$. Kvôli stabilite satelitu musí byť tento obdĺžnik otočený stranou a vo východozápadnom a stranou b v severojužnom smere. Pre každú sledovanú osobu agenti tajnej služby zistili

polohu dôležitých objektov, v ktorých sa obvykle zdržiava. Je v štátnom záujme, aby všetky objekty, v ktorých sa môžu nachádzať podozrivé osoby, boli ostro sledované. Satelity SPS sú však pomerne drahé, preto sa riaditeľ rozhodol rozmiestniť ich tak, aby ich potreboval čo najmenej. Čoskoro bude musieť predložiť návrh rozpočtu na budúci rok, veľmi rýchlo by teda potreboval vedieť, koľko tých satelitov vlastne potrebuje.

Úloha: Napíšte program, ktorý načíta rozmery územia pozorovaného jedným satelitom, počet sledovaných objektov n a súradnice všetkých objektov $[x_1, y_1], \ldots, [x_n, y_n]$ a vypíše, koľko najmenej satelitov je treba na to, aby každý objekt bol pozorovaný aspoň jedným satelitom. Satelit sleduje obdĺžnik aj s okrajom.

Príklad: Pre $a = 2$ a $b = 1.5$ a objekty so súradnicami $[0.0, 1.0], [-1.0, 0.0], [1.0, 0.0]$, $[0.0, -1.0]$ treba minimálne 2 satelity.

1643. O Danovej kostre

Dano je študent – zoznámte sa – pred rokom odišiel študovať do Ameriky. Chudák, vtedy ešte netušil, ako je tam draho. Štipendium, ktoré mal rozplánované až do leta, minul už teraz. Za posledné dva doláre si kúpil plechovku piva a sadol si na lavičku do parku. Tam si ho všimol „podnikateľ" Johnny, ktorý hneď pochopil, v akej situácii sa Dano ocitol. Johnny ponúkol Danovi, že od neho odkúpi kosti, jeho vlastné kosti, \$30 za kus (tí chlapíci dnes už obchodujú so všetkým).

Dano hneď na druhý deň zabehol na röntgen. Tam zistili, že Dano sa skladá z n uzlových bodov očíslovaných od 1 po n, ktoré sú navzájom poprepájané kosťami. Každá kosť spája dva uzlové body. Ďalej je každá kosť charakterizovaná svojou hrúbkou. Keď Dano videl ten obrovský kapitál, ktorý v sebe nosí, zaleskli sa mu oči šťastím a rozhodol sa, že niektoré kosti predsa len predá. Stanovil si však dve podmienky:

1. Každé dva uzlové body jeho tela musia ostať prepojené, teda musí existovať cesta (po kostiach) od každého uzlového bodu do každého iného uzlového bodu.
2. Najtenšia kostička, ktorá Danovi ostane bude najhrubšia možná (vzhľadom na prvú podmienku).

Teda Dano chce, aby ostal súvislý a aby najkrehšia časť jeho kostry bola čo najpevnejšia – logické, nie?

Úloha: Dano si podľa röntgenu vyhotovil zoznam svojich kostí. O každej kosti vieme, ktoré dva uzlové body spája a akú má hrúbku. Pomôžte chudákovi Danovi, ktorý na také množstvo údajov nestačí. Poraďte mu, ktoré kosti má predať tak, aby zarobil čo najviac a pritom dodržal všetky podmienky, ktoré si stanovil.

Ešte niečo – Dano je skvelý programátor, neprijme hocaký algoritmus! Pokúste sa vyriešiť úlohu čo najefektívnejšie – najlepšie v časovej zložitosti porovnateľnej s počtom kostí.

Príklad: Pre $n = 5$ a kosti vedúce z 1 do 2 hrúbky 10, z 2 do 3 hrúbky 6, z 2 do 4 hrúbky 3, z 3 do 5 hrúbky 7, z 4 do 5 hrúbky 7, z 3 do 4 hrúbky 8 je najvýhodnejšie predať kosti vedúce z 2 do 4 a z 3 do 4. Najtenšia kosť, ktorá mu ostane, má hrúbku 6.

1644. Lalčo lyžiarom

Lalčo všetko bol. Raz bol dokonca lyžiarom. Ako tak schádzal snehové pláne, vhupla mu do hlavy myšlienočka: „Možno keby som sa vyviezol ešte dvoma vlekmi a až potom zišiel dolu druhou stranou kopca, bolo by toho lyžovania viac ako státia v radoch na vleky. Ale keby som sa vyviezol radšej hentými troma vlekmi. . . "

A tak tam stál na alpskom grúni a hútal, ktorými vlekmi sa má vyviesť a po ktorých zjazdovkách potom zísť dolu, aby dosiahol čo najlepší pomer času na zjazdovkách a času na vlekoch. Nakoniec to aj vyhútal. Podarí sa to aj vám?

Úloha: V Alpách je n lyžiarskych stanovíšť, m zjazdoviek a k vlekov. Zjazdovky aj vleky vždy vedú medzi nejakými lyžiarskymi stanovišťami, zjazdovky vždy z vyššie položeného do nižšie položeného a vleky z nižšie položeného do vyššie položeného.

Vstup začína číslom n, nasleduje číslo m, popis zjazdoviek, číslo k a popis vlekov. Každá zjazdovka je popísaná tromi číslami: z ktorého do ktorého stanovišťa vedie a trvanie zjazdu. Aj každý vlek je popísaný tromi číslami: z ktorého do ktorého stanovišťa vedie a ako dlho dokopy trvá čakanie a vývoz ním.

Správna trasa sa skladá z dvoch fáz: v prvej fáze sa Lalčo z nejakého stanovišťa vyvezie niekoľkými vlekmi, v druhej fáze sa niekoľkými zjazdovkami spustí naspäť do stanovišťa, z ktorého vyrazil. Úlohou vášho programu je vypísať trasu, ktorá má najvyšší pomer medzi časom stráveným na zjazdovkách a časom stráveným na vlekoch.

PRÍKLAD:

VSTUP:

```
5
4
1 3 12
2 3  6
3 4  9
5 4  9
3
4 5 12
5 1 12
4 2 18
```

VÝSTUP:

```
4 5 1 3 4
```

(Okrem uvedenej trasy ešte existuje trasa 4 2 3 4 a trasa 4 5 4.

Pri zvolenej trase si Lalčo na vlekoch počká $12 + 12 = 24$ *minút, potom sa môže* $12 + 9 = 21$ *minút lyžovať. Pomer medzi lyžovaním a čakaním je* $21/24 = 0.875$.*)*

1645. O profesorovi Indigovi a víle Amálke IV

Profesor Indigo kedysi postavil počítač $PP_{\texttt{Amálka inside}}$ (pozri úlohu 1615). Nedávno na ňom ale spravil rozsiahle úpravy: podarilo sa mu k počítaču pripojiť rozhranie pre čiernu skrinku. Čierna skrinka je zariadenie, ktoré vie počítať nejakú funkciu. Napríklad čierna skrinka pre funkciu $x + y^2$ pre zadaný vstup $x = 2$, $y = 3$ vráti hodnotu 11. Profesora ale zaujímajú prefíkanejšie čierne skrinky, také, ktoré reprezentujú orientované grafy. Pripojením takejto čiernej skrinky získa počítač prístup k dvom funkciám:

function *pocet_vrcholov* : **integer**;
function *hrana*(a, b : **integer**) : **boolean**;

Funkcia *pocet_vrcholov* vracia počet vrcholov grafu n (vrcholy sú číslované $1, 2, \ldots, n$), funkcia *hrana*(a, b) vracia *true* práve vtedy, ak z vrcholu a do vrcholu b vedie hrana (nemusí platiť *hrana*$(a, b) =$ *hrana*(b, a)). Obe funkcie vracajú výsledok okamžite, t.j. v jednotkovom čase. Pre iné programovacie jazyky si vhodné definície ľahko domyslíte sami.

Prirodzene, každému grafu prináleží vlastná čierna skrinka. Profesorovi k úspešnému vedeckému výskumu chýba už len jediná maličkosť: Pre graf daný čiernou skrinkou zistiť, či sa dá dostať z každého vrcholu do každého iného.

Nepríjemné je, že počet vrcholov je obvykle veľmi veľký a $PP_{\texttt{Amálka inside}}$ má obmedzenú pamäť (aby sa do počítača zmestila čierna skrinka, musel profesor vybrať väčšinu pamäťových modulov).

ÚLOHA: Napíšte program pre $PP_{\texttt{Amálka inside}}$, ktorý skončí s výsledkom `ÁNO`, ak pripojená čierna skrinka popisuje graf s uvedenou vlastnosťou. V opačnom prípade všetky možnosti vetvenia výpočtu musia viesť k odpovedi `NIE`. Snažte sa minimalizovať pamäťové nároky. Na tom, ako dlho bude výpočet trvať, vôbec nezáleží. Predpokladajte, že *pocet_vrcholov* sa akurát zmestí do premennej typu **integer**, resp. **int**, ale $2 \cdot$*pocet_vrcholov* sa už nezmestí. Skúste porozmýšľať nad tým, koľko pamäte potrebuje $PP_{\texttt{Amálka inside}}$ a koľko by ste potrebovali, ak by ste nemali k dispozícii služby víly Amálky.

z1611. Zoči-voči – voči-zoči

Jeden z tradičných obľúbených turnajov na Kiribati je hra zoči-voči. Pravidlá sú jednoduché: dve družstvá sa postavia zoči-voči a naraz začnú kričať: „ZOČI-VOČI NÁM, NEVYHRÁŠ LEN TAK ĽAHKO SÚPER SÁM!“

Potom jedno družstvo začne hovoriť: „ZOČI-VOČI, ZOČI-VOČI...“ a druhé: „VOČI-ZOČI, VOČI-ZOČI...“ To družstvo, ktoré sa prvé pomýli, prehráva. Mnohých však už turnaje omrzeli, pretože v nich je víťaz vždy len jeden. Preto sa tohto roku rozhodli pre malú obmenu. Na začiatku začne v turnaji každý účastník sám ako družstvo. Počas turnaja sa družstvo, ktoré prehrá hru zoči-voči, vždy pridá na stranu vyhrávajúceho družstva a spolu potom pokračujú v turnaji proti ďalším družstvám. Nakoniec ostane jedno veľké družstvo, a tak vyhrávajú všetci. Večer sa všetci zídu a spoločne oslavujú veľkolepé víťazstvo...

Úloha: Napíšte program na vyhodnocovanie priebehu modifikovanej hry zoči-voči. Hru hrá n hráčov očíslovaných 1 až n. Váš program na začiatku načíta počet hráčov n a potom bude v nekonečnom cykle prijímať údaje o priebehu turnaja a otázky o stave a bude na ne príslušným spôsobom odpovedať. Vstupné údaje budú niektorého z týchto typov:

`Porazil` A B – oznámenie, že družstvo, v ktorom je hráč A porazilo družstvo, v ktorom je hráč B. Ak boli A a B už predtým v tom istom družstve, program má podať vhodné chybové hlásenie.

`Spolu` A B – otázka, či sú A a B v tom istom družstve. Program by mal správne odpovedať `ÁNO`/`NIE`.

Snažte sa, aby váš program reagoval čo najrýchlejšie. Hráčov totiž môže byť veľmi veľa a Kiribatčania sú veľmi netrpezliví.

Príklad:

```
Zadaj n
> 3
> Spolu 1 2
NIE
> Porazil 1 3
> Porazil 2 3
> Spolu 1 2
ANO
```

z1612. Zuzkine narodeniny

Zuzka Bodková bude čoskoro oslavovať narodeniny. Prípravy na veľkú záhradnú oslavu sú v plnom prúde: občerstvenie, torta, ohňostroj... Ešte treba dorobiť zasadací poriadok. Bodkovci majú v záhrade tri dlhočizné stoly. Zuzka chce pozvať na oslavu kopec kamarátok, ktoré treba usadiť ku stolom. Ale beda! Niektoré z dievčat sa navzájom neznášajú a v záujme hladkého priebehu osláv by nebolo dobre, aby nejaká dvojica dievčat, ktoré sa neznášajú, sedela pri tom istom stole. Zuzka si spísala zoznam pozvaných kamarátok ako aj zoznam dvojíc dievčat, ktoré sa navzájom neznášajú, a začala zostavovať zasadací poriadok. O hodnú chvíľu ju zastihla Kristínka, ako sa stále trápi.

„Pozri, to je jednoduché,“ začala Kristínka. „Najprv treba predsa usporiadať dievčatá podľa toho, s koľkými inými dievčatami sa neznášajú. Aha, Lucia sa znáša s najmenej ľuďmi, večne sa háda s celou triedou. Tú usadíme k prvému stolu. Ďalšia je Danka, tá je s Luciou na nože. Usadíme ju k druhému stolu. Potom máme Janku. Aj ona sa háda s Luciou, s Dankou je však zadobre, takže ju môžeme posadiť tiež k druhému stolu. Takto budeme pokračovať aj ďalej: vždy si vyberieme zo zoznamu dievča, ktoré sa znáša s najmenej inými dievčatami a postupne zistíme, či ju môžeme posadiť k prvému, potom k druhému a napokon k tretiemu stolu. Uvidíš, takto sa nám zaručene podarí všetky usadiť.“ No dievčatá o chvíľu zistili, že ani Kristínkiným spôsobom sa nedajú všetky pozvané priateľky usadiť. „V tom prípade ti neostáva nič iné, ako poprosiť otecka, aby zohnal ešte jeden stôl,

lebo žiadny iný spôsob rozsadenia neexistuje," povedala Kristínka a odbehla. Zuzku však trápili pochybnosti: nenašiel by sa predsa len nejaký spôsob, ako vystačiť s tromi stolmi?

Úloha: Označme si Zuzkine priateľky číslami 1 až n. Vašou úlohou bude napísať program, ktorý načíta zoznam dvojíc znepriatelených dievčat a nájde Zuzke jedno možné rozsadenie priateliek za tri stoly, alebo zistí, že takéto rozsadenie nie je možné. Ak ste presvedčení, že Kristínkina metóda nájde prípustné rozsadenie vždy, keď existuje, naprogramujte ju. V tomto prípade od vás očakávame aj zdôvodnenie (dôkaz) správnosti vášho programu. Ak si naopak myslíte, že Kristínkina metóda niekedy nenájde rozsadenie priateliek a pritom takéto rozsadenie existuje, navrhnite a naprogramujte svoju vlastnú. Nájdite aj nejaký protipríklad, alebo aspoň zdôvodnite, v čom sa Kristínka mýli.

z1613. Z Hanoja

Po návrate z potuliek po svete si Zoro opäť povedal, že určí dátum konca sveta. Techniku sa naučil od hanojských mníchov, ktorí používajú metódu MPV – metódu presúvania veží. Na posvätnom stole sú tri oceľové tyče, na ktorých sú nastrkané mohutné kotúče rôznej veľkosti. Kedysi dávno, keď bolo vyrieknuté proroctvo, stáli všetky kotúče na prvej tyči, usporiadané od najväčšieho po najmenší. Vraví sa, že keď budú všetky kotúče presunuté na druhú tyč, nastane koniec sveta. Avšak na presúvanie kotúčov medzi tyčami sú definované presné pravidlá:

- každý presun trvá presne jednu minútu,
- naraz sa presúva práve jeden kotúč,
- presun pozostáva z odobratia vrchného kotúča z niektorej tyče a jeho navlečenia na niektorú inú tyč,
- nesmie sa položiť väčší kotúč na menší.

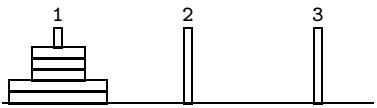
Nedočkavý Zoro pri tvorbe vlastného modelu tieto pravidlá plne rešpektuje, jediný rozdiel je v tom, že používa také kotúče, ktoré môžu mať rovnakú veľkosť, čo mu ušetrí mnoho presúvania, lebo môže pokladať na seba kotúče rovnakej veľkosti v akomkoľvek poradí.

Úloha: Zoro má n rôznych veľkostí kotúčov. Najmenších kotúčov je presne a_1, druhých najmenších a_2, atď. až najväčších je a_n. Všetky kotúče sú na prvej tyči od najväčších po najmenšie. Pomôžte Zorovi a napíšte program, ktorý načíta čísla n a a_1 až a_n a vypíše najkratšiu možnú postupnosť presunov, pomocou ktorých možno všetky kotúče presunúť z prvej tyče na druhú, pričom sa dodržia všetky pravidlá. Pokúste sa odôvodniť, prečo váš program funguje správne.

Príklad:

Vstup:

```
2
3 2
```



Výstup:

```
z 1 na 3
z 1 na 3
z 1 na 3
z 1 na 2
z 1 na 2
z 3 na 2
z 3 na 2
z 3 na 2
```

z1614. Zanzibarský vláčik

Zanzibarská slonovina vyniesla Zanzibarčanom na ich pomery nemalé zisky, a tak sa vláda rozhodla investovať do zlepšenia dopravnej situácie v krajine. Zanzibarské železnice ihneď presadili nákup modernej vlakovej súpravy, ktorú ich železničná sieť potrebuje ako soľ. Do niektorých odľahlých staníc totiž ešte stále zabezpečovala dopravu parná lokomotíva. Pri tvorbe nového cestovného poriadku si však uvedomili, že všetky ich stanice sú primalé na to, aby sa tam nová vlaková súprava dokázala otočiť. Preto chcú pre lokomotívu

nájsť takú okružnú trasu, kde by sa nemusela otáčať alebo cúvať. Taktiež chcú, aby okruh neprechádzal viackrát tou istou stanicou.

Úloha: Zanzibarské železnice majú n staníc očíslovaných $1, \ldots, n$. Niektoré stanice sú navzájom poprepájané priamymi úsekmi koľajníc, všetky úseky sú obojsmerné. Napíšte program, ktorý načíta počet zanzibarských staníc n, počet prepojení m a jednotlivé dvojice staníc, ktoré sú priamo spojené železničnou traťou a nájde okruh, po ktorom môže premávať nová lokomotíva. Ak takých okruhov existuje viac, stačí vypísať ľubovoľný z nich. Ak neexistuje ani jeden, program o tom vypíše príslušnú správu.

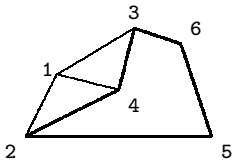
Príklad:

Vstup:

```
6
8
1 2
3 1
1 4
4 2
2 5
3 4
6 3
5 6
```

Výstup:

```
Okruh ide cez stanice:
3 4 2 5 6
```



z1615. Z kuchyne

„Ja to takto robiť nebudem," vyhlásil kuchtík Karol a dal si ruky vbok. „Ale neblbni, určite sa to nejako dá. Veď si len predstav, že by sme niektorú z tých nádob museli vyliať – a bolo by po dvoch hodinách roboty," dôvodil kuchtík Kveťo. „Radšej si rýchlo premyslime, ako to spravíme, lebo o chvíľu tu bude kuchár Kleofáš a... Veď vieš, akú má veľkú varechu," pokračoval Kveťo.

A čože to našich kuchtíkov takto trápilo? Všetky nádoby, ktoré v kuchyni mali, zaplnili rozličnými krémami do kráľovskej torty. Všetky nádoby až na jeden hrniec – do neho im ostávalo naliať presne n decilitrov mlieka na vanilkový krém (div sa svete, ten ešte nemali). Ale beda, hrniec mal objem m dl. Je síce $n \leq m$, ale k dispozícii majú iba veľkú kaďu s mliekom (nevieme, koľko ho tam je, ale je ho tam veľmi veľa) a dve naberačky s objemom a a b dl. Hrniec je teraz prázdny a jediné, čo môžu robiť je nasledujúce:

- nabrať ľubovoľnú naberačku plnú mlieka z kade a vyliať ju do hrnca (ak sa celý obsah naberačky do hrnca nezmestí, naleje sa po vrch hrnca a zvyšok mlieka sa vyleje späť do kade),
- alebo nabrať ľubovoľnú naberačku plnú mlieka z hrnca a vyliať ju do kade (ak v hrnci nie je dosť mlieka, naberie sa všetko mlieko z hrnca).

Úloha: Pomôžte kuchtíkom a napíšte program, ktorý načíta celé čísla a, b, m, n (môžete predpokladať, že platí $a, b, n \in \{1, 2, \ldots, m\}$) a vypíše postup operácií, alebo správu, že do hrnca nie je možné dostať uvedené množstvo mlieka.

Príklad: Keď máme k dispozícii naberačky s objemami $a = 4$, $b = 5$, hrniec objemu $m = 6$ a chceme vyrobiť objem $n = 1$, môžeme postupovať takto: naplníme hrniec (ľubovoľným spôsobom) až po vrch a potom odoberieme 5 dl.

z1621. Z krajiny byrokratov

V krajine byrokratov treba takmer na všetko úradné povolenie. Na úradoch sú teda obrovské rady, v ktorých často stoja aj samotní byrokrati. Ale na takého byrokrata čaká na úrade množstvo ľudí. Preto, aby sa zamedzilo nepokojom súvisiacim s dlhými čakacími dobami, bol do úradov zavedený „Prioritný systém vybavovania byrokratov" (PSVB). Pri príchode na úrad každý človek oznámi svoje meno a dôležitosť. Dôležitosť je reálne číslo.

Čím je väčšie, tým viac je daný človek potrebný v úrade. Žiadni dvaja ľudia nemajú rovnakú dôležitosť. K vybaveniu potom vždy volajú človeka s najväčšou dôležitosťou.

Úloha: Naprogramujte PSVB ako program pracujúci v nekonečnej slučke, ktorý bude zo vstupu prijímať správy dvoch typov: správy o príchodoch jednotlivých ľudí na úrad a požiadavky na vypísanie mena najdôležitejšieho z čakajúcich (ten je potom vybavený a ďalej už nečaká). Správy sú nasledujúcich tvarov: `PRÍCHOD` ⟨meno⟩ ⟨dôležitosť⟩ alebo `ĎALŠÍ`.

Na začiatku na úrade nečaká nikto. Môžete predpokladať, že mená sú jednoslovné, dlhé najviac 20 znakov.

Príklad:

```
> PRÍCHOD KATKA 93.3
> PRÍCHOD MATEJ 120.7
> PRÍCHOD PETER -88.9
> PRÍCHOD FILIP 62.34
> ĎALŠÍ
MATEJ
> PRÍCHOD KLEOFÁŠ 1000.3
> ĎALŠÍ
KLEOFÁŠ
> ĎALŠÍ
KATKA
...
```

z1622. Zoltánov tanečný večer

Zoltán Samba je učiteľom v tanečnej škole. Práve sa končí kurz tanca pre stredoškolákov a na jeho plecia spadla zodpovednosť za vydarený priebeh záverečného tanečného večierku – venčeku. Pri prvom tanci by rád videl na parkete čo najviac svojich žiakov. O každom chlapcovi a dievčati vie, či by spolu chceli tancovať, alebo nie. Chcel by žiakov popárovať tak, aby nikto netancoval s niekým, s kým nechce tancovať, a súčasne aby vznikol čo najväčší počet párov. A tak si na papier napísal mená chlapcov do jedného stĺpca, mená dievčat do druhého a skúšal ich popárovať. O chvíľu ho takto zastihla jeho kolegyňa Zita Polková.

„Na to treba ísť systematicky," vraví. „Najprv treba popárovať tých, čo sú ochotní tancovať s najmenej partnermi. Tých, ktorí majú veľa potenciálnych partnerov, je dobré spáriť až na konci – pravdepobnosť, že sa im z nich niekto ujde, je vyššia ako u tých, ktorí nemajú veľký výber partnerov."

„Pozri, Peter chce tancovať len s Luciou a Lucia len s Petrom, v tomto prípade je to jednoznačné. Ďalší chlapec, pre ktorého existuje najmenší počet potenciálnych partneriek, je Miro: môže tancovať len s Jankou a Dankou. Priradíme mu Danku, lebo Danka môže tancovať s menej chlapcami ako Janka. A máme ďalší pár. Takto postupujeme aj ďalej: Vždy uvažujeme len o tých chlapcoch a dievčatách, ktorí ešte nemajú pár. Z nich vyberieme chlapca, ktorý je ochotný tancovať s najmenším počtom zostávajúcich dievčat. Z týchto dievčat mu priradíme tú, ktorá môže tancovať s najmenším počtom zostávajúcich chlapcov. V prípade nejednoznačnosti výberu (napríklad keď niekoľko chlapcov má na výber rovnaký počet dievčat) vyberieme ľubovoľného z nich. Týmto spôsobom vytvoríme maximálny počet párov, aký sa dá."

Úloha: Počet chlapcov označme n, počet dievčat m. Ďalej pre každého chlapca máme zoznam dievčat, s ktorými môže tancovať. Chceme vytvoriť čo najviac tancujúcich párov chlapec – dievča. Samozrejme, každý chlapec môže tancovať s najviac jedným dievčaťom a naopak. Chlapec môže tancovať len s dievčaťom, ktoré sa nachádza v jeho zozname. Ak si myslíte, že metóda Zity Polkovej nájde vždy maximálny počet párov, tak to dokážte a

túto metódu naprogramujte. Ak si to nemyslíte, tak uveďte konkrétny príklad, na ktorom metóda zlyhá. (Nemusíte naprogramovať tú správnu, ale môžete sa o to pokúsiť.)

z1623. Zoro a zlý drak

Zorovi sa podarilo určiť dátum konca sveta a s úľavou zistil, že ešte mu zostal čas na to, aby stihol zabiť zlého draka. Zlý drak žije v jaskyni a má n hláv. Zabiť draka však nie je vôbec také jednoduché, ako by sa mohlo zdať. Zoro vlastní dva meče – zlatý a strieborný. Ak zaútočí na draka strieborným mečom, zotne mu s hláv, ak zlatým, zotne z hláv. Ak však drakovi zostane aspoň jedna hlava, dorastie mu hneď niekoľko nových. Ak Zoro útočil strieborným mečom, dorastie a nových hláv, ak zlatým, tak b nových hláv. Zoro nemôže použiť taký meč, ktorý by zoťal viac hláv ako drak v danom okamihu má. Drak zomrie, ak nemá žiadnu hlavu. Pomôžte Zorovi zistiť, či s danou sadou mečov dokáže poraziť draka, alebo či sa má radšej vrátiť do Hanoja a zohnať iné meče.

Úloha: Napíšte program, ktorý načíta čísla n, s, z, a, b a vypíše, či sa Zorovi môže podariť zabiť draka (t.j. odťať všetky jeho hlavy).

Príklad: Pre $n = 5$, $s = 4$, $a = 2$, $z = 2$ a $b = 3$ je odpoveď `ÁNO`. (Zoro môže použiť zlatý meč (drakovi zostane 6 hláv) a následne dvakrát strieborný meč, čím draka dorazí.)

z1624. Zázvorové pivo

Závozník Zachariáš sa živí rozvozom zázvorového piva. Pivo je síce nezdravé, ale všetkým pupkatým zbrojnošom preveľmi chutí, takže Zachariáš pekne zarába a dňom i nocou bohatne. Má to iba jeden háčik. Vždy, keď náš Zachariáš vstúpi aj s vozom piva do hocijakého mesta, musí zaplatiť mýto. A to sa mu veru vôbec, ale vôbec nepáči. Práve teraz je v meste A a chcel by sa dostať do mesta B, kde sa má konať veľký jarmok, na ktorom sa jeho pivo bude určite dobre predávať. Chce tam však docestovať tak, aby musel cestou platiť mýto v čo najmenej mestách. A tak sedí zadumaný na voze a premýšľa.

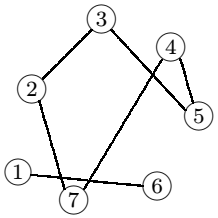
Úloha: Na vstupe máte zadané prirodzené číslo $n > 1$ určujúce počet miest (mestá sú číslované číslami 1 až n), číslo m určujúce počet ciest, a čísla miest A a B. Ďalej je daný zoznam týchto ciest. Každá cesta vedie medzi dvoma mestami a je zadaná dvojicou čísel týchto miest. Zistite, či sa Zachariáš môže dostať z mesta A do mesta B. Ak áno, vypíšte trasu vedúcu z A do B cez najmenší možný počet miest. Ak je takýchto trás viac, vypíšte ľubovoľnú.

Príklad:

Vstup:
```
7 6 3 4
1 6
2 3
2 7
4 7
4 5
3 5
```

Výstup:
```
3 5 4
```

(Existuje ešte trasa 3, 2, 7, 4, tá však ide cez viac miest.)



z1625. Z centrálneho trhoviska

Raz sa Zoro vybral do mesta nakupovať. Voľakto mu však zákerne premagnetizoval kompas, a tak si po niekoľkých dňoch putovania uvedomil, že pravdepodobne zablúdil. Nezaprel však svoju dobrodružnú povahu, pokračoval vytýčeným smerom, i došiel do mesta takého veľkého, aké nikdy predtým nevidel. Tam sa popýtal náhodného okoloidúceho na smer na trhovisko. Ten mu takto riekol: „Vidím, že nie si tunajší. Máš vôbec peniaze, keď sa ti na trh zachcelo?“. Zoro siahol do mešca, vytiahol tučný zväzok bankoviek a zatriasol s nimi cudzincovi pred očami. Ten odvetil: „S takýmito peniazmi u nás nepochodíš, hybaj ich do banky zameniť. Ale pamätaj, naše nominálne hodnoty sa inakšie počítajú!“

V banke Zoro zistil, že prevodová tabuľka medzi normálnou a tunajšou menou je veľmi komplikovaná – tu totiž uznávajú iba čísla, ktoré sú odpredu rovnaké ako odzadu. Nazývajú ich palindrómy. Ako je aj všade inde zvykom, palindrómy majú usporiadané, pričom hodnotu palindrómu určuje jeho poradové číslo. Teda napríklad palindróm 6 má hodnotu 6, palindróm 11 má hodnotu 10 a palindróm 131 má hodnotu 22. Ako ale Zoro zistí, či má dosť peňazí na zaplatenie meča aj štítu, keď predavač má obe sumy vyčíslené v tunajšej mene?

Úloha: Pomôžte Zorovi a napíšte program, ktorý načíta dva palindrómy x a y a vypíše ich súčet – tiež palindróm. Konkrétne, vypíše ten palindróm, ktorého poradové číslo je súčtom poradových čísel palindrómov x a y.

Príklad: Pre $x = 44$ a $y = 2$ je výsledok 66 a pre $x = 101$ a $y = 101$ je výsledok 292.

z1631. ZY nevergreeny

Isto viete, že evergreen je pieseň, ktorá zostáva populárna aj dlhý čas po svojom vzniku. Naopak nevergreen je taká pieseň, ktorá ešte nikdy nebola skutočným hitom – či už preto, že je úplne nová, alebo preto, že jej kvalita je príliš nízka (vysoká) na to, aby sa zapáčila davom. ZY Rádio je rozhlasová stanica určená pre náročných poslucháčov, ktorá vysiela iba ZY nevergreeny. To sú také nevergreeny, ktorých meno sa skladá iba z písmen `Z` a `Y`.

Úloha: Napíšte program, ktorý pomôže pracovníkom ZY Rádia spoľahlivo určovať, ktorá pieseň je ZY nevergreen. Program bude v nekonečnom cykle prijímať správy od užívateľov a bude na ne vhodným spôsobom reagovať. Správy sú dvoch typov: `Hit` m a `Skontroluj` m, kde m je reťazec obsahujúci iba veľké písmená.

Správa typu `Hit` m oznamuje programu, že pieseň s názvom m sa stala hitom, a teda už nikdy nemôže byť vysielaná na stanici ZY Rádio.

Správa typu `Skontroluj` m znamená, že niektorý redaktor chce pustiť pieseň s názvom m a potrebuje vedieť, či táto pieseň je ZY nevergreen, t.j., či jej názov pozostáva len z písmen `Z` a `Y` a či doteraz nebola zaevidovaná pomocou správy `Hit` m.

Príklad:

```
> Hit ZYZY
> Skontroluj ABC
ABC nie je nevergreen
> Skontroluj ZZZ
ZZZ je nevergreen
> Hit ZZZ
> Hit AAA
> Skontroluj ZZZ
ZZZ nie je nevergreen
```

z1632. Zákerný rozvrh

Ako iste všetci viete, Tomáš je veľmi snaživý študent, a preto sa hneď po zverejnení rozvrhov rozhodol zapísať si čo najviac prednášok. Naraz môže byť najviac na jednej prednáške a každú zapísanú prednášku musí absolvovať celú (teda časy prednášok sa nesmú prekrývať). Ako tak pridával a uberal prednášky zo svojho rozvrhu, objavil sa škriatok VIL a povedal: „To máš, Tomáš, jednoduché. Prednášky si budeš zapisovať jednu po druhej. Ako prvú si pekne zapíšeš tú, ktorá zo všetkých skončí úplne najskôr. Potom si vždy zapíš takú prednášku, ktorá sa ti neprekrýva so žiadnou už zapísanou a končí najskôr. Ak je takých prednášok viac (nutne musia končiť v rovnakom čase), môžeš si vybrať ľubovoľnú z nich. Keď si nebudeš môcť zapísať žiadnu ďalšiu prednášku, vedz, že ich máš zapísaných najviac, ako sa len dá.“

Úloha: Ak si myslíte, že VIL poradil Tomášovi dobre, zdôvodnite, prečo je to tak, a podľa jeho rady napíšte čo najefektívnejší program riešiaci zadanú úlohu.

V opačnom prípade navrhnite svoj vlastný spôsob riešenia, naprogramujte ho a nezabudnite zdôvodniť, prečo sa VIL mýli a zdokumentovať to na vhodnom kontrapríklade.

Váš program dostane na vstupe počet prednášok n a týždenný rozvrh skladajúci sa z n riadkov tvaru „⟨deň v týždni⟩ ⟨čas začiatku⟩ ⟨čas konca⟩ ⟨názov prednášky⟩“.

Vypísať by mal jeden zoznam prednášok, ktoré si môže Tomáš zapísať. Ale pozor! Ak Tomáš zistí, že nemá najvyšší možný počet prednášok, psychicky sa zrúti a máte ho na svedomí.

Príklad:

Vstup:

```
3
PON 07:20 09:45 Kog. psychológia
UTO 09:30 10:00 Teória hier
PON 09:00 10:35 Programovanie
```

Výstup:

```
Najviac 2 prednášky:
Kog. psychológia
Teória hier
```

z1633. Zamaskuje Ferdo zamestnanie?

Agent Ferdo pracuje už niekoľko desaťročí pre SIS (Slovenskú informatickú spoločnosť) na istej nemenovanej americkej univerzite. Po dlhé roky sa mu darilo plniť úlohy – získavať technické plány najnovších algoritmov. V súčasnosti je však jeho pozícia ohrozená – tajná služba ho odhalila a vedie ho v databáze pod číslom i. Ferdo je jeden z najlepších agentov svojho druhu a dokáže sa dostať na krátky čas aj k počítačom tajnej služby. Lenže na to, aby sa dal vymazať niektorý agent z databázy, je potrebné zadať jeho číslo zakódované a na kódovanie sa používa veľmi prefíkaný kódovací algoritmus.

Algoritmus zakóduje každé číslo ako usporiadanú k-ticu navzájom rôznych prirodzených čísel v rozsahu od 1 po n. Pre číslo i sa príslušná k-tica nájde takto: Zostrojíme všetky k-tice z čísel $1, \ldots, n$ a usporiadame ich lexikograficky (t.j. usporiadame ich najprv podľa prvého čísla; tie, ktoré majú prvé číslo rovnaké, usporiadame ďalej podľa druhého čísla, atď.). Kódom čísla i bude i-ta položka výsledného zoznamu. (Položky číslujeme od 1.)

Úloha: Pomôžte Ferdovi a napíšte program, ktorý načíta čísla n, k a i a vypíše kód čísla i. Snažte sa, aby bol váš program čo najrýchlejší, lebo čísla n, k a i môžu byť aj pomerne veľké. (Rozhodne neodporúčame vytvárať zoznam všetkých k-tic z čísel 1 až n.)

Príklad: Pre $n = 4$, $k = 3$, $i = 3$ je výsledkom trojica $(1, 3, 2)$.

z1634. Zázvorové pivo II

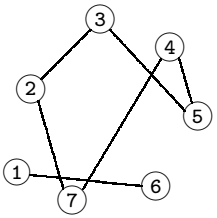
Zázvorové pivo II™ sa stále necháva rozvážať závozníkom Zachariášom. Je vyrobené vylepšenou receptúrou a je na to veľmi pyšné. Práve sa necháva predávať v meste A (a mimochodom všetkým výborne chutí), ale chcelo by sa dostať na veľkolepý festival piva do mesta B. Zázvorové pivo II™ si brúsi zuby na hlavnú cenu – zlatý kríget a skromne uznáva, že jeho vyhliadky sú ružové. Teda ak sa tam dostane. Má to totiž háčik – zdraželo mýto. Rovnako ako v časoch, keď sa pomocou Zachariáša rozvážalo Zázvorové pivo I, aj teraz treba pri každom vstupe so Zachariášom a vozom do hocijakého mesta zaplatiť mýto. Ale teraz si každé mesto stanovilo vlastnú výšku mýta. A tak si Zázvorové pivo II™ zadumane špliecha a premýšľa, kade ísť, aby cesta vyšla čo najlacnejšie.

Úloha: Na vstupe máte zadané prirodzené číslo $n > 1$ určujúce počet miest (mestá sú číslované číslami 1 až n) a pre každé mesto i je daná výška mýta $v_i > 0$ v tomto meste. Ďalej sú zadané čísla miest A a B a číslo m určujúce počet ciest medzi mestami. Každá z m ciest je zadaná dvojicou čísel miest, medzi ktorými vedie. Zistite, či sa Zázvorové pivo II™ môže dostať z mesta A do mesta B. Ak áno, vypíšte trasu vedúcu z A do B takú, aby bol súčet mýt v mestách, cez ktoré trasa vedie, najmenší možný. Ak je takýchto trás viac, vypíšte ľubovoľnú.

Príklad:

Vstup:
```
7
1 5 8 2 10 1 3
3 4
6
1 6
2 3
2 7
4 7
4 5
3 5
```

Výstup:
```
3 2 7 4
```



(Na vypísanej trase treba zaplatiť mýto vo výške $5+3+2=10$*. Existuje ešte aj trasa 3 5 4, ale na nej treba zaplatiť mýto vo výške* 12*.)*

z1635. Znižovanie zadĺženosti

Naše hospodárstvo je vo veľmi zlom stave. Hádam ani nieto podniku, ktorý by nemal nejaké dlhy. A nie je zriedkavý ani prípad, že jeden podnik súčasne niektorým podnikom dlhuje a iné podniky sú naopak jeho dlžníkmi. Ak podnik X dlží podniku Y sumu k korún, budeme to zapisovať ako $d(X,Y)=k$.

Skupina ekonomických expertov vymyslela spôsob, ako celkovú zadĺženosť znižovať. Navrhujú zaviesť používanie zmluvy o prevode dlhu. Ak pre nejaké podniky X, Y a Z sú dlhy $d(X,Y)$ a $d(Y,Z)$ obidva aspoň k korún, potom môžu uzavrieť zmluvu o prevode dlhu veľkosti k, na základe ktorej sa dlhy $d(X,Y)$ a $d(Y,Z)$ obidva znížia o k korún a naopak dlh $d(X,Z)$ sa o k korún zvýši. Pritom X a Z môže byť aj tá istá spoločnosť. Vtedy vznikne dlh typu $d(X,X)$, ktorý automaticky zaniká. Takýmito zmluvami medzi podnikmi je možné celkovú zadĺženosť postupne znižovať dovtedy, kým už sa žiadna zmluva o prevode dlhu nebude dať uzavrieť. Otázkou zostáva, aký bude výsledok.

Úloha: Napíšte program, ktorý načíta počet podnikov n, počet dlhov medzi podnikmi m a jednotlivé dlhy. Každý dlh je zadaný ako trojica čísel X, Y a k (kde $d(X,Y)=k$). Podniky sú číslované číslami $1,2,\ldots,n$. Váš program má vypísať zoznam dlhov, ktorý je jedným z možných výsledkov používania zmluvy o prevode. Váš zoznam teda musí byť taký, že mohol vzniknúť postupným uzatváraním zmlúv a súčasne už nie je možné uzavrieť žiadnu ďalšiu zmluvu.

Príklad:

Vstup:
```
4 4
1 2 30
2 3 40
2 4 10
4 1 10
```

Výstup:
```
1 2 20
1 3 20
```

1711. O d'Artagnanovi

Keď mladý d'Artagnan dovŕšil osemnásty rok života, rozhodol sa, že sa vydá do Paríža a stane sa mušketierom. Dostať sa do Paríža však v tej dobe nebolo vôbec jednoduché. Jedinou rozumnou možnosťou bolo vydať sa po kráľovských cestách spájajúcich jednotlivé mestá. Každá cesta začína aj končí mohutnou mestskou bránou. Problém je, že každú mestskú bránu strážia buď kráľovi mušketieri alebo kardinálovi gardisti a značia si všetkých, čo prišli do mesta. Preto keď človek prišiel do cudzieho mesta bránou stráženou mušketiermi, musel z neho aj odísť niektorou bránou stráženou mušketiermi – gardisti ho nemajú na zozname a dali by ho zavrieť. Naopak, ak prišiel bránou stráženou gardistami,

musel nejakou takou aj odísť. Navyše sa vie, že medzi gardistami a mušketiermi vládne nepriateľstvo, preto ak bránu na jednom konci cesty strážia jedni, na druhom konci ju určite strážia tí druhí. Nečudo, že d'Artagnan už hodiny sedí nad mapou Francúzska a rozmýšľa, ako sa v tejto situácii čo najrýchlejšie dostať do Paríža. (Z rodného mesta môže d'Artagnan odísť hociktorou bránou.)

Úloha: Vo Francúzsku je n miest číslovaných od 1 po n. Paríž má číslo 1, d'Artagnanove rodisko má číslo n. Medzi mestami vedie m ciest. Všetky cesty sú obojsmerné. O každej ceste vieme číslo jej začiatočného a koncového mesta, jej dĺžku, ako aj to, akí vojaci strážia jej začiatok (`M` sú mušketieri, `G` gardisti; koniec pochopiteľne strážia tí druhí).

Napíšte program, ktorý načíta počet miest n, počet ciest m a informácie o jednotlivých cestách a vypíše dĺžku najkratšej trasy z d'Artagnanovho rodiska do Paríža za daných podmienok.

Príklad:

Vstup:
```
5 7
1 2 3 M
1 4 4 G
1 3 2 G
2 5 3 G
2 4 10 M
5 4 1 G
4 3 1 M
```

Výstup:
```
Najkratšia cesta má dĺžku 5.
```
(Je to cesta $5 \to 4 \to 1$*.)*

1712. Dopravný podnik mesta Bratislava

Jedného krásneho dňa sa rozhodli Janko a Marienka so svojimi deťúrencami a niekoľkými jazvečíkmi ísť do hlavného mesta na výlet. Prišli, ako inak, na hlavnú stanicu. Tam teraz čakajú na zastávke autobusu. Už asi štvrťhodinu hľadia na automat na lístky a špekulujú, aký lístok kúpiť. Janko vraví: „Ja by som kúpil jeden zľavnený lístok + pes, jeden celý a jeden celý + pes.“ „To je hlúposť!“ vraví Marienka. „Bolo by to zbytočne drahé. Lepší by bol jeden celý + zľavnený + pes...“ Ich drahé deťúrence na nich nechápavo hľadia. Aj by svojim rodičom poradili, ale ešte nevedia hovoriť. Preto im musíte pomôcť vy.

Úloha: Na vstupe sú čísla a, b, c udávajúce počet dospelých, detí a psov, ktorých chceme odviezť. Ďalej je daný počet typov cestovných lístkov n. Každý typ lístku vieme popísať štyrmi číslami: „x_i dospelých + y_i detí + z_i psov za cenu p_i korún“.

Váš program má vypísať, akú najmenšiu celkovú sumu musíme zaplatiť za lístky. (Cestujúci si môžu kúpiť lístky aj pre viac ľudí alebo psov, ak to bude stáť menej.)

Príklad:

Vstup:
```
2 3 2
6
1 0 0 10
0 1 0 5
0 0 1 10
1 0 1 19
0 1 1 14
1 1 1 23
```

Výstup:
```
51
```
(Sú to lístky za $23 + 23 + 5$ *korún.)*

1713. O spartakiáde

Iste viete, že kedysi sa pravidelne usporiadavala spartakiáda. V rok jej konania sa na všetkých školách na telesnej výchove cvičievali rôzne choreografické prvky a najlepší z najlepších žiakov potom predvádzali svoje umenie pred očami prítomných aj televíznych

divákov na pražskom Strahove. Občas to ale úplne nevyšlo a po niektorom náročnom prvku sa cvičenci ocitli vo formácii, v ktorej pôvodne vôbec nemali byť. Našťastie boli vždy na zemi v pravidelných vzdialenostiach nakreslené krúžky, podľa ktorých sa dalo orientovať. Každý z cvičencov vtedy dobehol na najbližší neobsadený krúžok a pokračoval v cvičení. Najväčší problém však mali organizátori so záverom, keď sa všetci zúčastnení mali postaviť vedľa seba a pokloniť sa. Vtedy ich to nenapadlo, ale dnes by s veľkou nádejou čakali na vaše programy.

Úloha: Napíšte program, ktorý načíta počet cvičencov n a pre každého cvičenca súradnice krúžku, na ktorom stojí. Krúžky sú umiestnené na všetkých miestach s celočíslenými súradnicami. Váš program by mal vypísať počet presunov, ktoré sú potrebné k tomu, aby všetci cvičenci stáli na krúžkoch s rovnakou y-ovou súradnicou a neboli medzi nimi neobsadené krúžky (t.j. x-ové súradnice cvičencov budú $x, x+1, \ldots, x+n-1$). Tento počet má byť najmenší možný.

Cvičenci sa môžu presúvať len z krúžku na krúžok a to tak, že práve jedna zo súradníc nového krúžku sa od príslušnej predošlej súradnice líši presne o 1 (teda môžu ísť na krúžok vľavo, vpravo, dopredu alebo dozadu). V žiadnom momente presúvania by na jednom krúžku nemal stáť viac ako jeden človek.

Príklad: Pre $n = 5$ a súradnice cvičencov: $[1,2]$, $[2,2]$, $[1,3]$, $[3,-2]$, $[3,3]$ je minimálny počet presunov 8: Cvičenec so súradnicami $[1,3]$ sa presunie na $[0,3]$ a odtiaľ na $[0,2]$, cvičenec z pozície $[3,-2]$ sa presúva cez súradnice $[3,-1]$, $[3,0]$, $[3,1]$ na $[3,2]$ a cvičenec z pozície $[3,3]$ sa presunie cez $[4,3]$ na $[4,2]$.

1714. O vekslákoch

Je také miesto, kde ľudia nakupujú peniaze za peniaze, marky za doláre, koruny za jeny, jeny za franky, libry za drachmy a všakovaké iné za všakovakejšie inšie. Motá sa tam mnoho pochybných existencií, ktoré sledujú iba svoje malé, mnohokrát iracionálne ciele. Volajú ich veksláci. Keď má človek šťastie a dostatok informácií, môže prísť na takomto mieste k slušnému majetku tak, že si donesie peniaze v nejakej mene, vymieňa ich, vymieňa, až má opäť peniaze v tej istej mene, ale je ich viac. To sa nie vždy dá dosiahnuť.

Úloha: Zistite, či sa to dá dosiahnuť, ak máte daný zoznam kurzov, za ktoré veksláci menia peniaze. Ak sa to dá, tak uveďte aj postupnosť výmen s príslušnými kurzami. Predpokladajte, že môžete presne zameniť ľubovoľné množstvo danej meny. Uvedomte si však, že ak vekslák mení koruny na doláre, nemusí to robiť naopak v rovnakom kurze.

Príklad:

Vstup:

```
SKK 44.23 na USD 1
USD 1.3 na GBP 1.08
GBP 1.18 na SKK 65.993
USD 0.02 na SKK 1
SKK 50 na USD 1
```

Výstup:

```
ÁNO: SKK 44.23 na SKK 50
```

(SKK $\underset{44.23:1}{\longrightarrow}$ USD $\underset{0.02:1}{\longrightarrow}$ SKK)

1715. Hra s pešiakmi

Za starých dobrých čias trávili Janko s Marienkou večery hraním hry BOXES. Tá ich však už dávno omrzela, tak teraz skúšajú všelijaké nové hry, napríklad hru s pešiakmi.

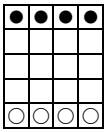
Táto hra sa hrá na šachovnici s r riadkami a s stĺpcami. Jeden hráč má s bielych pešiakov a druhý s čiernych pešiakov. Na začiatku hry sú biele figúrky rozostavené v spodnom riadku šachovnice a čierne v hornom. Hráč, ktorý je na ťahu, môže potiahnuť jednou zo svojich figúrok o ľubovoľný počet políčok smerom dolu alebo hore, pričom ale nesmie preskočiť súperovu figúrku ani skončiť ťah na políčku, kde táto figúrka stojí. Hráči sa pri ťahoch striedajú, pričom začína hráč s bielymi fugúrkami. Prehráva hráč, ktorý už nemôže urobiť žiadny ťah.

Úloha:

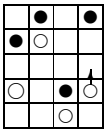
a) Napíšte program, ktorý bude hrať túto hru proti Marienke, keď na ňu Janko nemá čas. Program najprv načíta r a s a opýta sa užívateľa, akú farbu figúrok chce mať. Potom váš program striedavo ťahá a pýta si ťah od užívateľa. Sústreďte sa na to, aby váš program hral hru čo najlepšie – pokiaľ možno optimálne.

b) Marienka si pamätá mnoho rozohraných hier, ktoré by chcela dohrať. Napíšte preto program, ktorý načíta r, s, farbu užívateľových figúrok a farbu figúrok hráča, ktorý je práve na ťahu. Potom načíta situáciu hry (t.j. pre každý stĺpec šachovnice pozíciu bieleho a čierneho pešiaka) a dohrá túto hru proti užívateľovi. Opäť sa sústreďte na to, aby váš program hral hru čo najlepšie.

Príklad: Pre $r = 5$ riadkov a $s = 4$ stĺpce.



Obr. 1



Obr. 2

Na obrázku 1 je začiatočná pozícia hry. Na obrázku 2 je možná situácia počas hry. Jeden z možných ťahov bieleho hráča (ak je práve na ťahu) je napríklad o jeden krok dopredu figúrkou v najpravejšom stĺpci.

1721. O krízovom štábe

Mimozemšťan Martow sa práve vrátil z diplomatickej misie na vzdialenej planéte s názvom Zem. Nedočkavo vykukol z okienka svojej vesmírnej lode. Ako však čoskoro zistil, na jeho domácej planéte zďaleka nebolo všetko tak, ako malo byť...

Na Martowovej planéte je n miest. Medzi každými dvoma mestami bolo vybudované obojsmerné teleportové spojenie. Z neznámych príčin sa však všetky teleporty naraz pokazili a teraz každý z nich funguje len jedným smerom. Mimozemšťania sa rozhodli situáciu riešiť ustanovením krízového štábu. Každé mesto bude mať v štábe svojho zástupcu. Štáb sa musí stretnúť čo najskôr. Za miesto stretnutia je teda nutné zvoliť také mesto M, aby sa člen, ktorému bude cesta trvať najdlhšie, dostal do mesta M čo najskôr, t.j. na najmenší možný počet použití teleportov. Inými slovami, vzdialenosť (počítaná v počte použitých teleportov) z najvzdialenejšieho mesta do mesta M musí byť najmenšia možná.

Úloha: Napíšte program, ktorý načíta počet miest n a popis jednotlivých teleportov a vypíše číslo mesta M, v ktorom má zasadnúť krízový štáb. Popis teleportov má nasledujúci formát: pre každé mesto i je daný počet teleportov $p[i]$, ktoré vedú z tohto mesta. Ďalej je daný zoznam $p[i]$ miest, do ktorých vedú teleporty z mesta i. Môžete predpokladať, že pre každé i, j, $i \neq j$, sa v zozname vyskytne práve jeden z teleportov $i \to j$ a $j \to i$. Ak je vyhovujúcich miest na stretnutie štábu viac, vypíšte ľubovoľné jedno z nich.

Príklad:

Vstup:

```
5
4 2 3 4 5
3 3 4 5
1 4
1 5
1 3
```

Výstup:

```
3
```

(Z mesta č. 4 sa sem dá dostať pomocou dvoch teleportov, z miest č. 1, 2 a 5 pomocou jedného teleportu.)

1722. O telocviku

Riško, Vladko a Dávidko, ako poriadni žiaci, pravidelne chodia na telocvik. Radi naň chodia – môžu sa tam veselo zahrať s loptou spolu s ich telocvikárom – ujom Mariánom.

Ujo Marián je príjemný bradatý pán, má ale jednu chybu – rád dáva za trest svojim žiakom klikovať, používajúc pri tom s obľubou frázu: „Daj si dvadsať!“

Jedného dňa prišli všetci, celá trieda, neskoro na telocvik. Vbehli spolu do telocvične, ale beda! Zježená brada uja Mariána neveštila nič dobrého. Tak sa žiaci rýchlo postavili do radu, ako to zvyknú robievať vždy na začiatku hodiny. Lenže rad sa akosi ujovi Mariánovi nepáčil. „Musím im dať za trest klikovať. Koľkože klikov?“ húta si. „Nestoja pekne podľa výšky, nestihli sa v tom zhone usporiadať.“ Začne rátať, koľko dvojíc žiakov je navzájom vymenených (dvojica je vymenená, ak vyšší stojí pred nižším). Toľko klikov dostane každý z nich za trest. Ale dajako mu to v tom zhone nejde. Pomôžte mu!

Úloha: Na vstupe je daný počet žiakov n. a výšky žiakov $a_1, a_2, \ldots, a_n$ v tom poradí, v akom stoja v rade. Výšky žiakov sú celé čísla od 1 po n. Každá výška sa vyskytuje v triede práve raz. Zrátajte počet všetkých „vymenených“ dvojíc v rade, t.j. takých dvojíc (a_i, a_j), že $i < j$, ale $a_i > a_j$.

Príklad: Pre $n = 5$ a poradie výšok $3, 2, 4, 1, 5$ je výsledok **4**. Vymenené sú dvojice výšok $(3, 2)$, $(3, 1)$, $(2, 1)$ a $(4, 1)$.

1723. O lyžiaroch

Mt. Kopec je vychýrené lyžiarske stredisko. Je na ňom množstvo svahov ideálnych na lyžovanie, kratších, dlhších, strmších i menej strmých. A čo je najlepšie, na hornom aj dolnom konci každého svahu je horská chata, v ktorej si lyžiari môžu kúpiť teplý čaj s rumom, párky, alebo aspoň horalku. Jediné, čo svahom na Mt. Kopci chýba, sú lanovky. Je totiž veľmi nepríjemné, keď si lyžiar po zídení svahu musí dať lyže na plece a šliapať naspäť do kopca pešo. Preto sa lyžiari (a samozrejme chatári, ktorým to prinesie zisk) rozhodli postaviť sústavu lanoviek tak, aby sa postupnosťou vyvezení lanovkou a zídení svahov dalo dostať na ľubovoľný svah (a do ľubovoľnej chaty, samozrejme). Aby ušetrili, rozhodli sa stavať len jednosmerné lanovky a prirodzene, chceli by ich postaviť čo najmenej. Nevedia však, ako na to. Boli by radi, keby ste im pomohli.

Úloha: Na Mt. Kopci je n chát očíslovaných 1 až n. Medzi nimi vedie m svahov, každý svah vedie medzi dvoma chatami. Zjazdovať sa po ňom dá iba od vyššie položenej chaty k nižšie položenej. To znamená, že ak sa dá nejakou postupnosťou svahov dozjazdovať od chaty i po chatu j, určite sa nedá dozjazdovať od chaty j po chatu i. Napíšte program, ktorý načíta čísla n, m a zoznam svahov (každý svah je určený dvojicou vyššie položená chata, nižšie položená chata) a vypíše, medzi ktorými dvojicami chát je potrebné postaviť jednosmernú lanovku, aby sa postupnosťou vyvezení lanovkou a spustení sa po svahu dalo dostať z každej chaty do každej inej. Počet postavených lanoviek má byť minimálny.

Príklad: Pre $n = 4$, $m = 4$ a svahy: z 1 do 2, z 1 do 3, z 2 do 3, z 2 do 4 je treba pridať lanovky z 3 do 1 a zo 4 do 1. (Iné riešenie: z 3 do 2 a zo 4 do 1.)

1724. O prederavenej streche

Janko, Marienka, ich deťúrence a všetci jazvečíci po dlhých výpočtoch lacno docestovali do perníkovej chalúpky hlboko v tmavom lese. Deti s jazvečíkmi zatiaľ veľmi vyhladli a živelne sa pustili do sladučkej chrumkavej strechy. V streche tak žiaľ vznikli diery, ktorými fúkala chladná nepríjemná jeseň. Marienka sa rozhodla napiecť perníky, ktorými by všetky diery poplátala. Zamiesila cesto na perníky, avšak zistila, že v chalúpke nemajú žiadny váľok. Chce preto poslať Janka do mesta nejaký kúpiť, len nevie, aký široký potrebuje. Keďže Janko je veľmi sporivý, chce kúpiť čo najlacnejší (a teda najužší).

Úloha: Na vstupe je číslo $n \geq 3$ a súradnice $[x_1, y_1], \ldots, [x_n, y_n]$ konvexného n-uholníka. Tento konvexný n-uholník predstavuje záplatu, ktorú treba vyrobiť. Napíšte program, ktorý vypočíta najmenšiu možnú šírku pásu cesta, z ktorého sa takáto záplata dá vykrojiť. Pás cesta vie Marienka vyvaľkať ľubovoľne dlhý. Záplatu a cesto je možné voči sebe navzájom ľubovoľne otáčať a posúvať.

Príklad: Pre $n = 4$ a súradnice $[1, 1.5]$, $[2, 0]$, $[2, 10]$, $[0.5, 3]$ je výsledok 1.5. Okraje pásu tvoria v tomto prípade zvislé priamky $x = 0.5$ a $x = 2$.

1725. O úbohom kráľovičovi

Kde bolo, tam bolo, niekde uprostred lesa medzi piatym a šiestym kopcom bolo maličké kráľovstvo. V tomto kráľovstve vládol mladý kráľovič Richard. Keďže mu bolo na tróne smutno, rozhodol sa zaobstarať si nejakú rúču manželku. Vybral sa teda do sveta, chodil, blúdil, až naveľa prišiel do kráľovstva mocného kráľa, ktorý mal zhodou okolností jedinú dcéru Pamelu, akurát súcu na vydaj. Richard ju hneď požiadal o ruku. Keď však zistil, aká to je škrata, pokúsil sa od svojej ponuky odstúpiť. Nie veľmi úspešne...

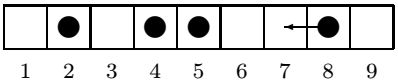
O pár dní neskôr sa na hladomorni otvorili dvere. Dovnútra vstúpili kat a kráľ. „Tak ti dám ešte jednu šancu," prehovoril kráľ. „Budeme hrať. Ak vyhráš, pustím ťa domov, ak nie, ech, tak si vezmeš moju dcéru Pamelu za ženu a odídeš s ňou vládnuť. Čo na to vravíš?" Čo už mal náš úbohý kráľovič robiť, neostalo mu nič iné ako súhlasiť.

Kráľ vytiahol z vrecka hrací plán – dlhočizný úzky pás látky, na ktorom bolo n (veľmi veľa) políčok vedľa seba očíslovaných od 1 po n. Potom na štyri rôzne políčka položil štyri kamienky. Hráč, ktorý je na ťahu, môže zobrať jeden kamienok a posunúť ho o niekoľko políčok (vždy aspoň o jedno) smerom k začiatku plánu (t.j. na políčko s menším číslom), pričom nesmie preskočiť žiaden iný kamienok. V žiadnom okamihu nesmie byť na žiadnom políčku viac ako jeden kamienok. Ten, kto nemôže ťahať, ani keby sa potrhal, prehral. Kráľ, ako starý gavalier, ponúkol Richardovi prvý ťah.

Keď úbohý Richard podumal nad hrou, zľakol sa, že sa mu nepodarí vyhrať a zbledol. Musíte mu preto pomôcť.

Úloha: Napíšte program, ktorý načíta pozíciu štyroch kamienkov na začiatku hry. Potom bude striedavo vypisovať svoje a načítavať kráľove ťahy až do konca hry. Snažte sa, aby váš program hral čo najlepšie.

Pozícia počas hry a možný Richardov ťah:



1731. O veľkom smäde

„Stavme sa, že vypijem 10 kofôl, ak mi ich niekto kúpi," navrhne Mišo. „Dobre, ale ak prehráš, tak ty mne kúpiš 15 kofôl! Nech sa mi vrátia aj s úrokmi," súhlasí Braňo. „Platí?" „Platí." A tak sa poberú k výčapu. Prvá, druhá, tretia kofola... to je ešte v pohode. Štvrtá, piata, šiesta... začína byť problém. Siedma, ôsma... Mišo je už zeleno-modrý a vzdáva sa. „Prehral si! Teda mi už visíš 43 kofôl."

A takto to v KSPáckych kruhoch chodí už dlhé týždne. Každý je niekomu dlžný množstvo kofôl. Pomaly ale iste sa v zložitých dlžobných vzťahoch už prestávajú všetci vyznať. Preto sa rozhodli, že sa dnes vyrovnajú. Lenže keď vypijú takých 50 kofôl v takejto zime, to im hrozí podchladenie, ba až smrť. A čo by bolo potom?

Vtom skrsla Dávidovi v hlave skvostná idea – tzv. bezkofolové vyrovnanie:

Nech dlhuje KSPák A KSPákovi B k_1 kofôl a B dlhuje C k_2 kofôl. Potom môžu svoje dlhy upraviť tak, aby B vypil o $x = \min(k_1, k_2)$ kofôl menej. Potom A bude dlžiť B iba $k_1 - x$ a B bude dlžiť C iba $k_2 - x$ kofôl. Zároveň však bude KSPák A dlžiť KSPákovi C o x kofôl viac. Ak sa takto stane, že niekto dlží niečo sám sebe ($A = C$), tak si nebude samozrejme nič kupovať.

Úloha: V prvom riadku vstupu sú dve celé čísla: n – počet KSPákov a m – počet dlžôb. Potom nasleduje m riadkov. Na každom sú tri celé kladné čísla A, B, k, čo znamená: KSPák A je dlžný KSPákovi B práve k kofôl. Vašou úlohou je napísať program, ktorý zistí, koľko minimálne kofôl musia dokopy vypiť a vypíše jeden možný záverečný stav, kto komu koľko kofôl dlží.

Príklad:

Vstup:
```
3 4
1 2 4
2 3 10
3 1 3
3 2 2
```

Výstup:
```
Počet vypitých kofôl: 5
1 3 1
2 3 4
```

1732. O automate na lístky

Dano študuje už dlhé roky programovanie na univerzite kdesi v Amerike. Nedávno prišiel na návštevu Bratislavy. Aké bolo jeho prekvapenie, keď zistil, že v Bratislave už dávno neplatia staré lístky na autobus (ktorých mal pre každý prípad niekoľko v zálohe).

Nové lístky sa dajú kúpiť napríklad v automate na zastávke. Automat vydáva n druhov lístkov s cenami $c_1, c_2, \ldots, c_n$ (ceny sú celé čísla). Lístky sa kupujú tak, že lístkovchtivá osoba postupne stláča tlačidlá zodpovedajúce lístkom, ktoré chce kúpiť (ak chce viac lístkov rovnakého druhu, stlačí príslušné tlačidlo viackrát) a na displeji automatu sa priebežne objavuje ich celková cena (na začiatku displej ukazuje 0 korún). Potom dotyčný vhodí do automatu príslušnú sumu mincí a z automatu vypadnú žiadané lístky.

Dano potreboval veľa rôznych lístkov (plánuje sa nejaký čas zdržať, navštíviť rodičov, priateľov, jednu nemenovanú slečnu, matematicko-fyzikálnu fakultu a zubára), a tak začal odušu stláčať tlačidlá na automate. Ako tak stláčal, zistil, že automat nie je ochotný akceptovať viac ako p stlačení tlačidiel a nie je ochotný akceptovať ďalšie stlačenie, ak je na displeji suma s a viac. Dano sa, ako správny programátor, rozhodol, že zistí, akú najväčšiu sumu je možné na displeji automatu stláčaním tlačidiel dosiahnuť. Potom si však uvedomil, že ak chce stihnúť všetky návštevy, musí sa poponáhľať a nechal túto úlohu na vás.

Úloha: Napíšte program, ktorý načíta počet lístkov, ich ceny, maximálny počet stlačení p a sumu s a vypíše, akú najväčšiu sumu je na displeji možné dosiahnuť.

Príklad: Pre $n = 3$, ceny: 6, 17, 35, $p = 10$ a $s = 100$ je výsledok 134. (Treba stlačiť tlačidlá pre dva lístky za 35, jeden za 17, dva za 6 a nakoniec ešte jeden za 35 korún.)

1733. O veľkom neporiadku

„Keď mi už tak veľmi chceš pomáhať, tak roztrieď tamtie skrutky!“ povedal otec ukazujúc na veľkú kopu skrutiek a matíc v rohu garáže. Tomáš bol celý nešťastný. Rád otcovi vo všetkom pomáhal. Nemohol za to, že všetko, čoho sa dotkol, sa hneď lámalo a kazilo. Otec sa väčšinou nezlostil, aj keď často musel robiť všetko odznova. Tentoraz to ale vyzeralo vážne. Výraz v otcovej tvári hovoril jednoznačne: „Už žiadne kazenie, prekážanie a práca navyše. Teraz ťa zamestnám prácou, pri ktorej nič nepokazíš.“ Tomáš teda smutne pozrel na kopu v rohu a pustil sa do práce. „Presne ako Popoluška,“ pomyslel si. „Popoluške prišli pomôcť holuby. Lenže mne nikto nepomôže. Ibaže by...“

Úloha: Máme n skrutiek a n matíc rôznych veľkostí, z ktorých ku každej skrutke prislúcha práve jedna matica a naopak. Napíšte program, ktorý pomože Tomášovi usporiadať skrutky a matice. Program načíta počet skrutiek a matíc n a potom môže klásť Tomášovi otázky typu „i-ta skrutka a j-ta matica?“. Tomáš odpovie buď „`OK`“, ak skrutka a matica patria k sebe, „`VÄČŠIA MATICA`“ ak j-ta matica je väčšia ako i-ta skrutka a „`VÄČŠIA SKRUTKA`“, ak je skrutka väčšia ako matica.

Úlohou vášho programu je nájsť ku každej skrutke jej zodpovedajúcu maticu. Tomáš chce mať túto robotu rýchlo hotovú, preto by ocenil, keby sa ho program pýtal čo najmenej otázok, avšak dôležitejšie je, aby mu program vyhodil správny výsledok.

Príklad: Pre $n = 3$ a otázky s odpoveďami: `1. skrutka a 2. matica? OK` a `2. skrutka a 1. matica? VÄČŠIA MATICA` sú správne páry: (1,2), (2,3), (3,1).

1734. O dlhých tyčiach

Stál Babylon, najslávnejšia to ríša. I jedného dňa povedali si Babylončania: „Nám už žiť na zemi neprináleží. Až po samé nebo vežu postavíme a potom tam žiť budeme!“. Nedbali oni varovania veštca, že kto sa Bohu rovnať chce, kruto potrestaný bude. Každý z Babylončanov sa domov pobral, i všetky tyče, čo doma našiel, na námestie doniesol. Keď už všetky tyče pokope boli, začali z nich konštrukciu veže stavať. Lenže čo to? Tyče boli tak rôznych dĺžok, že nik, ani ten najmúdrejší z nich, nebol schopný nájsť tri také, čo by strany trojuholníka tvorili. A tak hľadali, hľadali, no stále nenachádzali a stavbu začať nemohli. A tak by sa im zišiel program, ktorý by tento problém riešil za nich. (A tiež by sa im zišiel počítač...)

Úloha: Napíšte program, ktorý dostane na vstupe číslo n a následne n dĺžok tyčí $d_1, d_2, \ldots, d_n$, čo sú reálne čísla, pre ktoré platí $1 \leq d_i \leq 1\,000\,000\,000$. Program má vypísať, či je medzi nimi taká trojica, z ktorej sa dá zložiť trojuholník.

Dobre sa zamyslite nad časovou a pamäťovou zložitosťou vášho programu.

Príklad: Pre 3 tyče s dĺžkami 1.0, 1.1 a 2.1 program vypíše `NIE` a pre 5 tyčí s dĺžkami 1.7, 4.9, 51.0, 174.9 a 4.957 vypíše `ÁNO`.

1735. O Santovi a fľašiach II

Zlatokopi z Vyšnej Klondiky už po sedem rokov radi chodievajú do hostinca v Dolnom Kelčove posilňovať sa ohnivou vodou, hrať hazardné hry a uzatvárať stávky. Aj Santo a Banto tam radi a pravidelne chodia. Tentokrát sa im opäť podarilo vyhrať stávku s krčmárom a za odmenu im krčmár postavil na stôl do kruhu mnoho fliaš ohnivej vody rôzneho objemu a povedal: „Z týchto fliaš pite, koľko vám hrdlo ráči, ale dodržujte kultúru pitia!“

Santo sa už-už chcel rozbehnúť ku fľašiam, ale Banto ho zadržal. „Nezabúdaj na tie prihlúple krčmárove pravidlá, nemôžeme piť hocijako, nevypité fľaše musia v každom okamihu tvoriť súvislý úsek. Musíme si dávať pozor, ináč nás opäť vyhodí! A budeme sa pekne striedať. Jednu fľašu ty, jednu ja, až kým sa nám všetky neminú.“

Úloha: Na začiatku je na stole $2n$ fliaš s objemami 1 až $2n$ decilitrov (každý objem sa vyskytuje práve raz) rozostavených do kruhu. Hráč, ktorý je prvý na ťahu, môže vypiť ľubovoľnú fľašu. V každom ďalšom ťahu môže príslušný hráč vypiť ľubovoľnú fľašu susediacu s nejakou už vypitou fľašou (aby sa zachovala súvislosť úseku nevypitých fliaš). Cieľom hry je samozrejme preliať hrdlom viac ohnivej vody ako ten druhý.

a) Je dané n a $2n$ rôznych celých čísel od 1 po $2n$ reprezentujúcich objemy fliaš po rade v kruhu. Napíšte program, ktorý bude striedavo radiť Santovi (ktorý je prvý na ťahu), ktorú fľašu má vypiť, a načítavať, ktorú fľašu práve vypil Banto. Môžete predpokladať, že Banto hrá podľa pravidiel. Santovym cieľom je, aby dokopy vypil viac ohnivej vody ako Banto (je jedno, o koľko).

b) Napíšte program, ktorý načíta ľubovoľnú legálnu pozíciu (t.j. takú, ktorá mohla vzniknúť postupnosťou legálnych ťahov) a bude striedavo radiť hráčovi na ťahu, ktorú fľašu má vypiť, a načítavať ťahy súpera.

Na konci by mal program vypísať, kto koľko vypil a kto vypil viac.

Príklad priebehu hry:

a) Pre $n = 3$ a fľaše `2 6 3 1 5 4`:

```
Santo, vypi 6 dl fľašu.
Banto vypil: 3 dl
Santo, vypi 2 dl fľašu.
Banto vypil: 4 dl
Santo, vypi 5 dl fľašu.
Banto vypil: 1 dl
Dokopy Santo vypil 13 dl ohnivej vody a Banto len 8. Chudák Banto.
```

b) Pre $n = 3$ a fľaše `0 6 3 1 0 0` (prázdne fľaše označené nulou):

```
Santo už vypil: 6  Banto už vypil: 5  Na ťahu je: Banto

Banto, vypi 6 dl fľašu.
Santo vypil: 3 dl
Banto, vypi 1 dl fľašu.
Dokopy Santo vypil 9 dl ohnivej vody, ale Banto až 12. Chudák Santo.
```

1741. O vojenských zátarasoch

Kiribatské kráľovstvo začalo vojnu s Kráľovstvom Zimbabwe. Vraj hackeri z Kiribati napadli webstránky na oficiálnom kráľovskom serveri v Zimbabwe. Nevedno, čo je na tom pravdy. Možno to bola len zámienka. Ale v tejto chvíli už zimbabwejskí vojaci začali pochod na Kiribati. Na pokyn kiribatského kráľa zasadli vojenskí stratégovia a taktici a začali organizovať obranu.

Medzi Kiribati a Zimbabwe vedie sieť ciest. Každá cesta má danú svoju šírku. Aby kiribatské vojsko úspešne obránilo svoju krajinu pred nepriateľom, musia postaviť krížom cez celú šírku niektorých ciest zátarasy. Zistite, koľko najmenej metrov zátarasov musí kiribatská armáda postaviť a ktoré cesty pritom zatarasiť, aby si mohla byť istá, že zimbabwejskí vojaci nevstúpia do ich vlasti.

Úloha: Je daný počet uzlov n cestnej siete medzi Kiribati a Zimbabwe. Uzol číslo 1 sú Kiribati a uzol číslo n je Zimbabwe. Ďalej je daný počet ciest m spájajúcich uzly. Nakoniec je na vstupe m trojíc celých čísel x, y, s popisujúcich jednotlivé cesty. Trojica x_i, y_i, s_i ($1 \leq x_i, y_i \leq n$) znamená, že cesta vedie z uzla x_i do uzla y_i a jej šírka je s_i metrov. Cesty sú obojsmerné. Vašou úlohou je zistiť, ktoré cesty treba zatarasiť, aby sa postavilo čo najmenej metrov zátarasov a pritom neexistovala cesta, po ktorej by sa mohli zimbabwejskí vojaci dostať do Kiribatského kráľovstva. Vypíšte aj celkovú dĺžku zátarasov. Ak je optimálnych možností ako cesty zatarasiť viac, vypíšte ľubovoľnú z nich.

Príklad:

Vstup:

```
4 5
1 2 8
1 3 3
2 3 3
2 4 3
3 4 5
```

Výstup:

```
Zatarasiť cesty:
1 3
2 3
2 4
Dĺžka zátarasov:
9
```

1742. O kiribatských ponorkách

Blíži sa leto a s ním aj potápačská sezóna. Cítiť to aj v Kiribatskom Spolku Potápačov, kde sú prípravy na ňu v plnom prúde. Na zvýšenie bezpečnosti potápania treba podrobne preskúmať morské priekopy medzi kiribatskými ostrovmi, a tak sa členovia spolku rozhodli zadovážiť si novú ponorku.

Všetky ponorky dostupné na kiribatskom trhu sú dvojrozmerné a pozostávajú z rovnako veľkých štvorcových segmentov. Dva segmenty jednej ponorky buď nemajú žiaden prienik, alebo majú spoločnú celú jednu stranu. Každá ponorka pozostáva z jedného kusu, teda z ľubovoľného jej segmentu sa dá do ľubovoľného iného prejsť cez niekoľko ďalších jej segmentov. Navyše platí, že keď zoberieme dva segmenty ponorky, ktoré sú v rovnakej hĺbke a spojíme ich stredy myslenou čiarou, ani jeden bod tejto úsečky neleží mimo ponorky.

Morská priekopa má tvar obdĺžnika, ktorý pozostáva z $m \times n$ štvorcových segmentov rovnakej veľkosti ako segmenty ponorky. Niektoré segmenty priekopy sú zarastené koralmi, ktoré prekážajú pri jej prieskume. Horná strana obdĺžnika predstavuje morskú hladinu.

Ponorka sa na začiatku potápania nachádza na hladine (t.j. žiadna jej časť nie je pod hladinou). Počas celého potápania musia byť horné strany jej segmentov rovnobežné

s hladinou. Posádka môže počas zostupu ponorku posúvať doprava, doľava alebo dole. V žiadnom okamihu potápania sa však žiaden jej segment nesmie nachádzať na mieste zarastenom koralmi.

Členovia Kiribatského Spolku Potápačov teraz smutne sedia nad katalógom predávaných ponoriek a mapou morskej priekopy, v ktorej je zakreslené, ktoré jej segmenty sú zarastené koralmi. Dumajú, dumajú, ale vydumať nevedia, ktorú z ponoriek kúpiť, aby sa mohli ponoriť čo najhlbšie. Pomôžte im a napíšte program, ktorý im túto ťažkú úlohu pomôže vyriešiť.

Úloha: Sú dané čísla m a n a mapa morskej priekopy s rozmermi $m \times n$. Ďalej sú dané čísla a, b a tvar ponorky s rozmermi $a \times b$. Napíšte program, ktorý zistí, ako hlboko sa môže daná ponorka do danej priekopy ponoriť.

Príklad:

Vstup:

```
7 8        2 2
..#####    .#
..#....    ##
.......
.......
..#....
.#.#..#
....#..
...#.#.
```

Výstup:

```
Ponorka sa môže ponoriť do hĺbky 6.
```

1743. O Veľkej cene Formule 1

Táto sezóna bola pre Schumachera veľmi zlá. Niektoré preteky síce vyhral on, dosť veľa ich ale vyhral aj Häkkinen. Našťastie má vo vedení pretekov Formula 1 priateľa, ktorý mu je za čo-to vďačný. Taký vďačný, že mu je ochotný za každú cenu pomôcť. Avšak jediné, čoho je schopný, je iniciovať odhlasovanie zmeny bodovania. A tak teraz Schumacher sedí nad výsledkovými listinami a dumá, či sa dá vymyslieť bodovanie, pomocou ktorého by získal titul majstra sveta.

Úloha: Napíšte program, ktorý dostane na vstupe výsledkové listiny n pretekov a rozhodne, či existuje také obodovanie prvého až tretieho miesta, pri ktorom by mohol Schumacher vyhrať.

Na vstupe je počet pretekov n a tabuľka, koľkokrát sa ktorý pretekár umiestnil na ktorom mieste. Výstupom bude `SCHUMACHER MÔŽE/NEMÔŽE VYHRAŤ`, podľa toho, či existuje také obodovanie prvých troch miest $p_1 \geq p_2 \geq p_3 \geq 1$, pri ktorom by Schumacher vyhral, to znamená, že by mal najviac bodov zo všetkých.

Príklad: Pre $n = 5$ a výsledkovú listinu

Meno	1. miesto	2.miesto	3.miesto
Häkkinen	4	0	1
Schumacher	1	4	0
Hill	0	0	2
Irvine	0	1	1
Davidko	0	0	0

je správny výstup „`SCHUMACHER MÔŽE VYHRAŤ`“.

(A to napríklad pri bodovaní $p_1 = 10$, $p_2 = 9$, $p_3 = 5$.)

1744. O meteorológovi Ferdovi

Ferdo ako čerstvo vyštudovaný meteorológ nastúpil do práce do meteorologického ústavu. Veľmi sa tešil, že si bude môcť zapredpovedať počasia do ľubovôle, ba že ho možno občas pustia aj vystupovať do televíznych správ o počasí a teraz toto... Ako nového zamestnanca ho totiž prevelili na oddelenie štatistiky. Ferdo teraz smutne sedí nad stĺpcami

záznamov o teplotách nameraných každý deň od vzniku meteorologického ústavu. Nadriadení mu totiž dali za úlohu zistiť, kedy bolo najteplejšie obdobie. Obdobie je ľubovoľný súvislý úsek aspoň k dní. Teplotou obdobia rozumieme priemernú teplotu za jednotlivé dni. Pomôžte chudákovi Ferdovi, bezmocne sediacemu nad stohom záznamov, lebo inak skolabuje a budete ho musieť kriesiť.

Úloha: Napíšte program, ktorý načíta počet dní existencie ústavu n, minimálnu dĺžku obdobia k, teploty $t_1, t_2, \ldots, t_n$ (reálne čísla) namerané v jednotlivých sledovaných dňoch a vypíše prvý a posledný deň obdobia, ktoré je dlhé aspoň k dní a má pritom maximálnu možnú priemernú teplotu.

Príklad: Pre $n = 8$, $k = 3$, teploty: $-1.9, 4.5, -3.2, 0.5, -4.5, 5.8, -1.6, -2.7$ bolo najteplejšie v období od 2. po 6. deň. Priemerná teplota bola 0.62.

1745. O nových zlatokopoch

Opäť prišla vlna zlatej horúčky, a tak aj na Vyšnú Klondiku dorazila skupinka nových zlatokopov. Vystúpili z vláčiku, ktorý ich doviezol do Puerto Paža na okraj civilizácie, poobzerali sa a... Ktovie či šťastnou, ale určite náhodou natrafili práve na Santa a Banta. „Ehm... Prosím vás, neviete nám povedať, ako sa dostaneme do Dolného Kelčova?“ „To musíte najskôr do Nalomenej Triesky,“ zahlásil suverénne Santo. „To musíte najskôr do Naštiepeného Íveru,“ zahlásil naraz s ním nemenej suverénne Banto. „Viete čo, veď my sa tam vlastne tiež vraciame... Odvedieme vás tam. Samozrejme, nebude to zadarmo...“

Cesta do Dolného Kelčova je úsečka, na nej leží $n+1$ miest, očíslovaných od 0 (Dolný Kelčov) po n (Puerto Paža, kde práve sú). Od poslednej reformy VHDD (Vyšnoklondická hromadná dostavníková doprava) však dostavníky premávajú zvláštne. Presnejšie, z mesta A do mesta B ide dostavník len ak $A > B$ a $A - B$ nie je zložené číslo. (Zložené číslo je prirodzené číslo, ktoré má iného deliteľa ako 1 a seba.) Uznáte, že za týchto okolností nie je vôbec ľahké niekam sa dostať. Preto niet divu, že sa Santo a Banto stavili – budú na striedačku vyberať, do ktorého mesta sa ide. Kto dovedie skupinku do Dolného Kelčova, vyhráva a dostane všetky peniaze, ktoré im za sprievod greenhorni zaplatia.

Úloha: Napíšte program, ktorý načíta číslo n a následne bude hrať túto hru za začínajúceho zlatokopa.

Príklad priebehu hry, keď $n = 11$:

```
Santo: Poďme do mesta 10.
Banto: Poďme do mesta 8.
Santo: Teraz poďme do mesta 3.
Banto: A teraz do mesta 0.
Banto vyhral.
```

z1711. Z učtárne

Predstavte si stredne veľkú fabriku spoločnosti Vidličky a Nože. Nevysoká budova v príjemnom prostredí, všade však vládne čulý ruch. Odlievači, kováči, brusiči, leštiči, ohýbači (neverili by ste, koľko to dá roboty správne vytvarovať zuby na vidličke), zlatiči, pracovníci na obchodnom oddelení („Samozrejme, vážený pane, ten nôž je z čistého zlata. Azda si o nás nemyslíte, že by sme Vám boli schopní predať nejakú pozlátenú napodobeninu?“), tetušky v baliarni, šoféri odvážajúci výrobky do obchodov – tí všetci pracujú pre dobro firmy. A všetkých týchto ľudí má na starosti sused Záviš.

Sused Záviš totiž pracuje ako účtovník. Jeho starosťou je, aby každý zo zamestnancov dostal každý mesiac správne vypočítanú výplatu. Výplata sa zamestnancom počíta podľa počtu odpracovaných dní. Môžete predpokladať, že zamestnanci cez pracovné dni (t.j. pondelok až piatok) pracujú, v sobotu a nedeľu odpočívajú. Účtovanie je náročná práca, preto by sused Záviš ocenil každú pomoc.

Úloha: Napíšte program, ktorý načíta mesiac a rok a vypíše, koľko pracovných dní bolo v danom mesiaci. Váš program by mal fungovať pre ľubovoľný rok, špeciálne by nemal robiť problémy po roku 2000. Sviatky nemusíte uvažovať.

Príklad:

Vstup:	Výstup:
`SEP 1999`	`SEP 1999: 22 pracovnych dni`
`FEB 2000`	`FEB 2000: 21 pracovnych dni`

z1712. Zase SoDr

V softférovom družstve SoDr sa rozhodli naprogramovať nový editor. Editor je už takmer hotový a zostáva dorobiť iba niekoľko funkcií. Teraz si všetci lámu hlavu nad tým, ako spraviť presúvanie bloku textu. Potrebovali by vašu pomoc.

Úloha: Editor má globálne pole znakov A (veľmi veľké). V tomto poli je uložený editovaný text. Užívateľ editora si v texte vyznačil blok, t.j. súvislý úsek textu začínajúci i-tym a končiaci j-tym znakom v poli A. Užívateľ chce tento blok presunúť tak, aby po presune začínal v k-tom prvku poľa. Vašou úlohou je naprogramovať procedúru $presun(i, j, k)$, ktorá dostane čísla i, j a k a presunie v poli A blok textu od i-teho znaku po j-ty z pôvodného miesta tak, aby začínal na mieste k. Môžete predpokladať, že i, j a k sú zadané korektne, t.j. blok má dĺžku aspoň 1 a zmestí sa do poľa tak, aby začínal od pozície k.

Vaša procedúra nesmie používať ďalšie polia okrem poľa A a ani iné veľké dátové štruktúry. Použiť môžete iba konštantný počet pomocných premenných – znaky, čísla, logické hodnoty a pod. Súčasne sa snažte, aby vaša procedúra pracovala čo najrýchlejšie.

Príklad: Nech na začiatku pole A obsahuje reťazec `abcdefgh`. Príkazom $presun(3, 4, 2)$ dostaneme `acdbefgh` a následným vykonaním $presun(2, 3, 7)$ vyrobíme `abefghcd`.

z1713. Zo stavby

Závišovci práve dokončili hrubú stavbu nového domu na Ostrove pri Zmytej kvapke. Chystajú sa dokončiť podlahy, kachličkovať, osadiť okná, zaviesť elektrinu a ostatné inžinierske siete. Sami to však nezvládnu, a tak chcú pozvať odborníkov – remeselníkov. Tí si ale účtujú za prácu veľmi veľké peniaze, a preto im treba naplánovať, kedy má kto prísť. Problémom je, že jednotlivé činnosti musia vykonávať viacerí naraz, napríklad na podlahu v pracovni treba obkladača a elektrikára, na kúpeľňu navyše vodára (či vodníka?), na steny zase omietača a elektrikára. Keď si to pani Závišová dôkladne rozpísala, zistila, že úplne všetky kombinácie rôznych typov remeselníkov majú v dome uplatnenie. Zároveň si zaumienila, že všetkým činnostiam v jej dome bude robiť dozor, takže sa nemôžu vykonávať dve naraz. Samozrejme, nikto nebude zaháľať a len tak postávať – tí čo sú práve nepotrební, musia odísť. Keďže nový dom je na ostrove, chudák pán Záviš ich musí prevážať kompou. A v nej má voľné miesto iba pre jedného.

Úloha: Pre dané n – počet typov remeselníkov, ktorých označíme $1, 2, \ldots, n$, vypíšte postupne všetky podmnožiny množiny remeselníkov, a to v takom poradí, aby sa každé dve za sebou idúce podmnožiny líšili iba v jednom prvku – buď niekoho pán Záviš kompou privezie, alebo odvezie. Na začiatku je pritom v dome prázdna množina remeselníkov.

Príklad: Pre $n = 3$ môžu byť podmožiny vypísané napríklad v taktomto poradí:
1. $\emptyset$, 2. $\{1\}$, 3. $\{1, 2\}$, 4. $\{2\}$, 5. $\{2, 3\}$, 6. $\{1, 2, 3\}$, 7. $\{1, 3\}$, 8. $\{3\}$.

z1714. Zblúdilec v zrúcanine

Dr. Jones je archeológom. K jeho zamestnaniu, samozrejme, patrí aj skúmanie zrúcanín a starých podzemných chodieb. A Jonesovi sa teraz prvýkrát stalo, že zablúdil. Ešteže mu kedysi priateľ poradil fintu – pravidlo pravej ruky. Stačí položiť pravú ruku na stenu a ísť stále tak, aby sa ruka steny dotýkala. Výsledkom má vraj zaručene byť zodretá dlaň a nájdený východ. Tak teraz Jones stojí v chodbe uprostred zrúcaniny a rozmýšľa, ktorá ruka je pravá.

Úloha: Napíšte program, ktorý simuluje prechod bludiskom podľa pravidla pravej ruky. Vstupom je počet riadkov r a stĺpcov s bludiska, súradnice vchodu a východu a mapa bludiska. V mape „`X`“ znamená stenu a „`.`“ znamená prázdny priestor. Vchod a východ sa vždy nachádzajú na obvode bludiska. Na celom obvode bludiska je stena, iba vchod a východ sú voľné. Na začiatku je Jones vo vchode do bludiska a chce sa dostať k východu.

Príklad:

Vstup:

```
8 8
vchod: 4 1
vychod: 4 8
XXXXXXXX
X...X.XX
X.X....X
..XXXX..
X...XXXX
X.X....X
X.X..X.X
XXXXXXXX
```

Výstup:

```
Dr. Jones pôjde cez políčka:
[4,1], [4,2], [5,2], [6,2], [7,2],
[6,2], [5,2], [5,3], [5,4], [6,4],
[7,4], [6,4], [6,5], [6,6], [7,6],
[7,7], [6,7], [6,6], [6,5], [6,4],
[5,4], [5,3], [5,2], [4,2], [3,2],
[2,2], [2,3], [2,4], [3,4], [3,5],
[3,6], [3,7], [4,7], [4,8].
```

z1715. Zimbabwejský internet

Zimbabwejský kráľ Zimba-Zimba sa rozhodol, že jeho krajina veľmi súrne potrebuje internet. Zimba-Zimba, ako každý správny zimbabwejský kráľ, veľmi dlho rozmýšľal, ako by toto svoje rozhodnutie mohol vykonať, až nakoniec si vybral firmu ZimNET, ktorá ako jediná ponúkala pripojenie na internet. Teraz už Zimba-Zimba má internet, ale nemôže sa pozerať na žiadne WWW stránky, pretože firma ZimNET kráľovi nenainštalovala žiaden prehliadač. Zimba-Zimba je teraz veľmi smutný, lebo mu nikto nechce pomôcť.

Úloha: Pomôžte Zimba-Zimbovi a naprogramujte jednoduchý prehliadač WWW stránok. Ako vstup program načíta WWW stránku vo formáte HTML a zobrazí ju na výstup (do súboru) v čitateľnej forme. Samotný HTML súbor vyzerá podobne ako ten v príklade – na začiatku je vždy `<HTML><BODY>`, na konci `</BODY></HTML>`. Všetky texty uzavreté v `< >` označujú príkazy HTML, ktoré ovplyvňujú, ako sa stránka zobrazí. Niektoré takéto príkazy musia byť v dvojici, napríklad `<HTML>` a `</HTML>` – ten bez lomítka je vždy začiatok a s lomítkom je koniec bloku, na ktorý má daný príkaz vplyv. Takže napríklad `<CENTER>`*Nadpis*`</CENTER>` spôsobí, že text *Nadpis* bude tvoriť samostatný odsek, ktorý bude zarovnaný (vodorovne) na stred stránky. Druhý typ príkazov je nepárový a ide napríklad o príkaz `<P>`, ktorý označuje začiatok nového odseku. Mená žiadnych príkazov sa do výstupu nezapisujú. Samotný obsah stránky je tvorený viacerými odsekmi, pričom každý z nich môže zaberať aj viac riadkov vo výstupe. Vo vstupnom súbore môže byť text jedného odseku umiestnený na viacerých riadkoch, môže obsahovať veľa medzier, ale toto formátovanie vstupu nemá vplyv na formát výstupu. Každá skupina medzier alebo prázdnych riadkov vo vstupe znamená len oddelenie jednotlivých slov. Tieto jednotlivé slová sa potom zapisujú na výstup, pričom medzi každými dvoma slovami je len jedna medzera. Výstupné zariadenie má obmedzenú šírku (v našom prípade 80 znakov), a preto slovo, ktoré by túto hranicu prekročilo, sa zapíše do nasledujúceho riadku. Medzi každými dvoma odsekmi je vo výstupe jeden prázdny riadok (vo vstupe nemusí byť, napríklad „`koniec odseku 1<P>Druhý odsek`“ je tiež oddelenie odsekov).

Vašou úlohou je teda podľa načítanej WWW stránky v opísanom formáte vytvoriť súbor, ktorý bude obsahovať naformátovaný text stránky zarovnaný na šírku 80 znakov. Predpokladajte, že vstupný súbor obsahuje len opísané príkazy, teda `<HTML>`, `</HTML>`, `<BODY>`, `</BODY>`, `<CENTER>`, `</CENTER>` a `<P>`.

Príklad:
Vstup:

```
<HTML>
<BODY> Zimbabwejsky internet - ukazkovy subor<P>
Toto je druhy     odsek

textu, nezavisle od clenenia vo   vstupnom  subore.
<CENTER>  Toto bude vycentrovany odsek    </CENTER>
A nakoniec zase jeden obycajny odsek
</BODY>
</HTML>
```

Výstup:
(pre šírku strany 40 znakov; rámček nevypisujte – to je len zobrazenie hraníc stránky)

```
+----------------------------------------+
|Zimbabwejsky internet - ukazkovy subor  |
|                                        |
|Toto je druhy odsek textu, nezavisle od |
|clenenia vo vstupnom subore.            |
|                                        |
|      Toto bude vycentrovany odsek      |
|                                        |
|A nakoniec zase jeden obycajny odsek    |
+----------------------------------------+
```

z1721. Záhada najkratšej cesty

Bol teplý júlový večer a traja pátrači opäť sedeli v hlavnom stane. Tentoraz prebiehala vášnivá debata o tom, aká je najkratšia cesta z mestskej knižnice do hlavného stanu. Bobovi to trvalo 17 minút, Jupiter to cez podchod zvládol za 16:30 a Peter, krížom cez čínsku štvrť, to vraj zmákol za rovnú štvrťhodinku. Všetky tieto cesty však mali jednu vec spoločnú: žiadna netrvala rovnako dlho, a podľa pátračov žiadna z nich nebola najkratšia. Keď sa chýlilo ku polnoci, Jupiter vyhlásil:

„Tá najkratšia cesta sa pred nami nejako skrýva. Existuje vôbec? Už generácie programátorov sa skláňajú pred najkratšou cestou, páni Dijkstra, Floyd, Warshall sú slávni na celom svete. Využime ich služby, aby nám zistili, či vlastne existuje aspoň jedna najkratšia cesta."

Úloha: Pomôžte svojim kamarátom a napíšte im program, ktorý načíta popis mesta Rocky Beach a vypíše, či existuje najkratšia cesta z významného bodu číslo 1 (čo je hlavný stan) do významného bodu číslo n (čo je knižnica). Mesto sa skladá z n významných bodov očíslovaných od 1 po n a z m uličiek medzi nimi. Váš program na vstupe dostane čísla n, m a pre každú uličku trojicu (a_i, b_i, d_i), ktorá znamená, že i-ta ulička dĺžky $d_i > 0$ spája významné body a_i a b_i. Všetky uličky sú obojsmerné.

Príklad: Pre $n = 3$, $m = 2$ a uličky $(1, 2, 15)$ a $(2, 3, 7)$ program vypíše: `Najkratšia cesta predsa len existuje!` a pre $n = 3$, $m = 1$ a uličku $(1, 2, 42)$ vypíše: `Márna vaša snaha, tú najkratšiu nikdy nenájdete!`

z1722. Záh(r)adné mravenisko

„Z!" zarevala zlá kráľovná. Mravec Z zúfalo zrýchlil. Za pár sekúnd už bol v hlavnej komnate. „Ponížene prosím o odpustenie..." vrhol sa jej k nohám. „Sedemnásť tisíc tristo dvadsaťštyri," vydýchol. Nato sa kráľovná zamyslela. „A koľkože mravčekov žije na...na...na sto dvadsiatom treťom až štyristo siedmom poschodí?" Mravec Z sa zdesene

zdvihol a vydal sa opäť počítať obyvateľstvo. Z dverí ho vyprevádzal zlomyseľný smiech kráľovnej.

Keď tu zrazu mravec Z dostal nápad. Prešiel si celé mravenisko a na každom poschodí si zrátal počet mravcov, ktoré tam žijú a zaznačil si ich do svojho počítača. Teraz by však potreboval program, ktorý by vedel čo najrýchlejšie povedať, koľko mravcov žije v určitej časti mraveniska.

Úloha: Napíšte program, ktorý dostane na vstupe číslo n – počet poschodí mraveniska, následne n čísel, ktoré udávajú počty mravcov na jednotlivých poschodiach. Potom bude načítavať dvojice čísel x, y ($x \leq y$) a zakaždým vypíše, koľko mravcov žije na x-tom až y-tom poschodí. Vstup je ukončený dvojicou 0, 0.

Také mravenisko má veľmi veľa poschodí, keby ich malo ešte viac, nemuseli by sa počty mravcov na jednotlivých poschodiach hádam ani popratať do pamäte počítača. Kráľovná je veľmi netrpezlivá, preto sa snažte, aby váš program bol čo najrýchlejší. Je aj poriadne škodoradostná, preto počet otázok, ktoré úbohému Z položí, môže byť nepríjemne veľký.

Príklad: Pre 13 poschodí a počty mravcov na jednotlivých poschodiach od prvého 1, 3, 4, 7, 6, 9, 4, 5, 2, 4, 123, 1444, 32538 je pre dvojicu 3, 4 odpoveď: `Na poschodiach 3-4 žije spolu 11 mravcov.` Pre 13, 13 je odpoveď: `Na poschodiach 13-13 žije spolu 32538 mravcov.` Nakoniec pre 1, 13 je odpoveď: `Na poschodiach 1-13 žije spolu 34150 mravcov.`

z1723. Zostarnutý papagáj

V jednom nemenovanom bytíku v Bratislave žije jedno slušné dievčatko menom Ňaňka. Keď bola ešte menšia, spadla jej na hlavičku kalkulačka ruskej výroby. DÍÍÍho ju potom bolela hlavička a od tejto chvíle z duše znenávidela všetky kalkulačky. Komplikácie nastali, až keď Ňaňka chodila do školy. Keď spočítavali malé čísla, stačilo jej desať prstov. Potom začala používať prsty na nohách. Až raz na Vianoce jej priletel na balkón papagáj. A nebol to len taký obyčajný papagáj – vedel počítať. Keď mu Ňaňka zakričala: „plus mínus desať krát dva tri osem," papagáj začal opakovať: „dvanásť, dvanásť, dvanásť…" A neprestal, pokiaľ nedostal ďalšiu úlohu. Utíšil sa iba vtedy, keď ho niekto prekričal pokrikom: „Pištáá!!" A tak to išlo dlhé časy a Ňaňka sa naučila počítať iba v takomto divnom zápise. Lenže papagáj starne a Ňaňka potrebuje náhradu.

Úloha: Na vstupe máte matematický výraz zapísaný v prefixovej notácii. Vašou úlohou je vypočítať jeho hodnotu. A čo je to tá prefixová notácia?

1. Nezáporné číslo je výraz v prefixovej notácii a jeho hodnota je toto číslo.
2. Ak v_1, v_2 sú výrazy v prefixovej notácii s hodnotami h_1, h_2, tak aj $+v_1v_2$, $-v_1v_2$, $*v_1v_2$, $/v_1v_2$, $@v_1$ sú výrazy v prefixovej notácii a ich hodnoty sú $h_1 + h_2$, $h_1 - h_2$, $h_1 * h_2$, $h_1 \operatorname{div} h_2$ a ciferný súčet h_1.
3. Nič iné nie je výraz v prefixovej notácii
   Príklad: Pre `+ - 10 * 2 3 8` je výstup `12` a pre `@ 12` je výstup `3`.

z1724. O včielkach

Dr. Jones, aj vďaka vašej výdatnej pomoci, šťastne našiel východ zo zrúcaniny. Dlaň na pravej ruke ho pálila, mal ju celú rozodratú až do krvi. Naštastie pred východom zo zrúcaniny tiekla riečka. Sadol si k nej, ponoril ruku do vody a čakal, kým sa mu z ruky vyplavia choroboplodné zárodky. Voda bola príjemne chladná.

Potom si ľahol na chrbát. Cítil sa ako nikdy predtým, slniečko svietilo, bzukotali okolo neho včielky…

V horizontálnej rovine nad jeho hlavou bzukotalo n včielok so súradnicami $[x_1, y_1]$, $[x_2, y_2]$, …, $[x_n, y_n]$. Ako ich tak pozoroval, všimol si nečakané súvislosti. Všetky včielky sa pohybovali rovnakou konštantnou rýchlosťou. Navyše prvá včielka naháňala druhú, druhá tretiu, atď. až $(n-1)$-vá n-tú a konečne n-tá včielka prvú.

Úloha: Napíšte program, ktorý načíta n – počet včielok a počiatočné súradnice včielok $[x_1, y_1], [x_2, y_2], \ldots, [x_n, y_n]$ a bude simulovať ich pohyb po grafickej obrazovke. Včielky znázornite ako malé bodky.

Prémiová úloha: Zistite a hlavne zdôvodnite, čo by mal robiť váš program pre vstup: $n = 4$, $[x_1, y_1] = [0, 0]$, $[x_2, y_2] = [0, 100]$, $[x_3, y_3] = [100, 100]$, $[x_4, y_4] = [100, 0]$.

z1725. Zibabwejský internet II

Zimba-Zimba smutne hľadel na monitor. Má už síce nový prehliadač, ale čo keď na zimbabwejskom internete nie je veľa zaujímavých stránok. Ten je totiž veľmi mladý. A tak sa Zimba-Zimba rozhodol, že sa sám naučí písať HTML kód.

I učil sa, i učil, až sa nakoniec naučil. A tak celý natešený sa rozhodol, že napíše jednu krásnu kráľovskú WWW stránku. A ako ináč by taká stránka mala vyzerať, ak nie s veľkou hŕbou obrázkov. I napísal ju a plný očakávania spustil svoj prehliadač. Ale čo to? `Missing Image`? Tak to hádam nie! Kto ale nájde ten správny súbor? A čo ak chýbajú aj nejaké iné obrázky? Alebo nebodaj aj iné HTML súbory?

Pomôžte Zimbo-Zimbovi a napíšte mu program, ktorý mu vypíše všetky odkazy na obrázky a HTML stránky zoradené v abecednom poradí. Formát HTML súboru ostáva rovnaký ako v predošlej úlohe (z1715). To znamená, že obsahuje príkazy `<HTML>`, `<BODY>`, `</BODY>`, `</HTML>`,`<CENTER>`, `</CENTER>` a `<P>`. Okrem toho pribudol príkaz na vykreslenie obrázku `<IMG SRC="meno">`, kde meno je názov súboru vo formáte JPEG, alebo GIF (s príponami `JPG`, `JPEG` alebo `GIF`). Odkaz na inú stránku začína `<A HREF="meno">`, kde meno je názov súboru HTML (s príponou `HTM` alebo `HTML`). Za ním nasleduje text určený na zobrazenie ako odkaz. Ten je ukončený `</A>`. Váš program by mal vypísať počet zobrazovaných obrázkov vo formáte GIF a ich mená zoradené abecedne, počet obrázkov JPEG a ich abecedný zoznam. Taktiež vypíšte počet odkazov na iné stránky a abecedný zoznam týchto stránok. Dva odkazy na rovnaké súbory, alebo dva rovnaké obrázky sa počítajú len raz, taktiež sa vypisujú len raz. Mená súborov nerozlišujú malé a veľké písmená, takže `INTERNET.JPEG` je to isté ako `iNTernEt.JpeG`.

Nezabúdajte, že formátovanie vo vstupnom súbore nemá vplyv na samotnú HTML stránku, čiže celá stránka môže byť napríklad v dvoch riadkoch – asi takto:

```
<HTML><BODY>Stranka<IMG SRC="auticko.gif"><CENTER>
riadok 1<P>riadok 2</CENTER></BODY></HTML>
```

Príklad:
Vstup:

```
<HTML>
<BODY>
<CENTER> Oficiálna kráľovská stránka Zimba-Zimbu </CENTER>
<CENTER>    <IMG SRC="KORUNA.GIF">   </CENTER>  <P>
<IMG SRC="TRON.GIF"> <P>
<A HREF="HOSTINA.HTML"> Viac o kráľovskej hostine TU </A>
<P>
<IMG SRC="TORTA.JPEG"> <IMG SRC="KOLAC.JPG">
Príďte nás všetci pozrieť, ale nezabudnite si prečítať
<A HREF="PRAVIDLA.HTM"> pravidlá. </A> <P>
<P> <P>
<IMG SRC="KORUNA.GIF"> Pozrite si ďalšie maše stránky.
Váš Zimba-Zimba.
</BODY>
</HTML>
```

Výstup

```
Pocet GIF obrazkov: 2
KORUNA.GIF
TRON.GIF
Pocet JPEG obrazkov: 2
KOLAC.JPG
TORTA.JPEG
Pocet odkazov na stranky: 2
HOSTINA.HTML
PRAVIDLA.HTM
```

z1731. Z rodinnej kroniky

Bonifác je veľmi nešťastný. Jeho svokra Emília ho nemá rada. Stále sa s ním iba háda. Bonifác si vzal do hlavy, že Emília má hádavosť vrodenú a na podporu tejto idei začal v rodinnej kronike hľadať všetkých Emíliiných príbuzných a zmienky o ich hádavosti. Ako si tak listoval v kronike, zišla mu na um zaujímavá úloha:

Úloha: Napíšte program, ktorý pre daných dvoch ľudí zistí, či sú v nejakom príbuzenskom vzťahu a ak áno, vypíše ich najbližší príbuzenský vzťah. Vstupom programu budú výpisy z kroniky tvaru:

`SYN` ⟨meno syna⟩ ⟨meno rodiča⟩
`DCÉRA` ⟨meno dcéry⟩ ⟨meno rodiča⟩
`MANŽEL` ⟨meno manžela⟩ ⟨meno manželky⟩

Prvé dva zápisy hovoria, že prvý uvedený človek je synom resp. dcérou druhého, tretí zápis hovorí, že uvedení dvaja ľudia sú manželia.

Váš program bude ďalej načítavať dvojice mien a pre každú dvojicu vypíše popis najbližšieho príbuzenského vzťahu prvej osoby k druhej.

Popis vzťahu je postupnosť slov syn, dcéra, otec, matka, manžel, manželka v správnych pádoch. Napríklad vzťah „brat“ sa popíše ako „otcov syn“ alebo „matkin syn“. Vzťah je tým bližší, čím menej slov je potrebných na jeho popis. Niektoré vzťahy majú svoje skrátené názvy, napríklad „matka manželky“ sa obvykle volá svokra, „syn otca“ je brat. Pri výpise môžete použiť tieto skrátené názvy, avšak toto skrátenie nemá vplyv na blízkosť vzťahu.

So skloňovaním vo výstupe si nelámte hlavu...

Príklad:

Vstup:

```
SYN Bonifác Filoména
DCÉRA Alena Filoména
MANŽEL Peter Filoména
DCÉRA Anička Emília
MANŽEL Bonifác Anička
```

otázky:

```
Bonifác Emília
Emília Bonifác
Peter Emília
Alena Bonifác
```

Výstup:

```
manžel dcéry
matka manželky
otec manžela dcéry
dcéra matky (alebo dcéra otca)
```

z1732. Zaujímavá lanovka

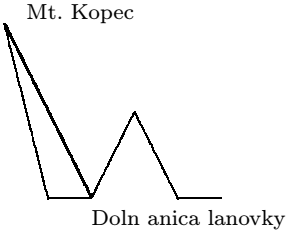
V pohorí High Tatras je okrem Mt. Kopca aj veľa iných kopcov a kopčekov. Ale zo žiadneho z nich nevedie lanovka priamo na vrchol Mt. Kopca. Zväz lyžiarov a turistov sa rozhodol tento nedostatok odstrániť vybudovaním novej lanovky. A nie hocijakej, ale najdlhšej možnej. V rámci úsporných opatrení sa rozhodlo, že lanovka sa postaví bez stĺpov. Z tohto dôvodu bude jej dráha vyzerať ako úsečka (nosné lano bude pevne napnuté

medzi koncovými stanicami). Pomôžte funkcionárom zväzu a napíšte program, ktorý im nájde vhodné miesto na stavbu koncovej stanice lanovky.

Úloha: Hrebeň pohoria dĺžky n kilometrov vyzerá pri pohľade zvrchu ako rovná úsečka. Pri pohľade zboku vidíme $n+1$ význačných bodov (na každom kilometri jeden), pričom každá dvojica susedných význačných bodov je spojená úsečkou. Tieto úsečky dokopy tvoria hrebeň. Pre každý význačný bod i ($0 \leq i \leq n$) je samozrejme známa jeho nadmorská výška v_i v kilometroch (reálne číslo).

Vrchol Mt. Kopca leží na kilometri 0 a tvorí jednu koncovú stanicu. Vašou úlohou je napísať program, ktorý zistí, na ktorom význačnom mieste na hrebeni treba postaviť druhú koncovú stanicu tak, aby dráha lanovky viedla vždy nad povrchom a jej dĺžka bola pritom najväčšia možná.

Príklad: Pre $n = 5$, $v_0 = 4$, $v_1 = 0$, $v_2 = 0$, $v_3 = 2$, $v_4 = 0$, $v_5 = 0$ treba nástupnú stanicu umiestniť na druhý význačný bod. Lanovka bude dlhá 4.47 km.



z1733. Zelené svahy Kiribati

Kiribatské detičky majú prázdniny, a tak sa najlepší priatelia Mwango a Itab-irik rozhodli, že sa pôjdu lyžovať. Pekná myšlienka, až na to, že na Kiribati býva pomerne teplo a nie je vôbec jednoduché nájsť nejaký zasnežený kopec. „Poďme na ostrov, na ktorom je najviac kopcov. Tak máme najväčšiu nádej, že si tejto zimy zalyžujeme!“ povedal Mwango. A tak si sadli nad mapu Kiribati a začali počítať kopce na jednotlivých ostrovoch. Po chvíli to však vzdali a vyhlásili, že to je robota pre koňa. Alebo pre počítač.

Úloha: Na vstupe sú dve čísla r, s udávajúce počet riadkov a stĺpcov mapy Kiribati a následne samotná výšková mapa Kiribati. Vode zodpovedá výška 0, súši výška väčšia ako 0. Okraj mapy tvorí voda. Dve políčka sú susedné, ak sa dotýkajú stranou. Dve políčka súše patria k tomu istému ostrovu práve vtedy, keď sa dá z jedného na druhý prejsť po súši tak, že vždy prejdeme na susedné políčko. Políčko súše je kopec práve vtedy, keď je vyšší od všetkých štyroch svojich susedov. Napíšte program, ktorý vypíše súradnice $[r_p, s_p]$ niektorého políčka ostrova, na ktorom leží najviac kopcov.

Príklad:

Vstup:
```
6 8
0 0 0 0 0 0 0 0
0 2 4 1 0 0 6 0
0 3 4 1 0 7 2 0
0 0 1 0 0 0 4 0
0 0 0 0 9 0 0 0
0 0 0 0 0 0 0 0
```

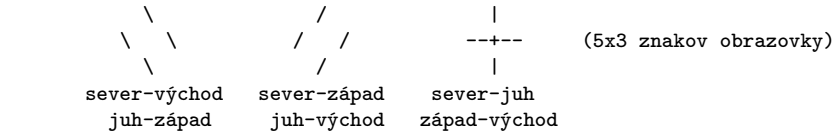
Výstup:
```
[2,7]
```

(Na tomto ostrove sú tri kopce. Ostrov dolu má iba jeden kopec a na ostrove vľavo nie je žiaden kopec.)

z1734. Zbytočne dlhý ropovod

Nový nález ropy na Ukrajine podnietil slovenskú vládu čo najrýchlejšie vybudovať tranzitný ropovod na našom území. Stavať sa začalo narýchlo, bez poriadneho projektu, takže plánovanie väčšinou vyzeralo tak, že za stavbyvedúcim prichádzali striedavo zástupca SPP a splnomocnenec vlády pre energetiku a prikazovali, ktorým smerom má viesť najbližší úsek.

Na zjednodušenej mape Slovenska (štvorec $n \times n$ kilometrov) ropa priteká na políčko v pravom hornom rohu mapy. Vyústenie ropovodu má byť na políčku v ľavom dolnom rohu. Každý úsek zaberá jedno políčko mapy. Skladá sa z dvoch rúr, z toho jedna sa vždy napája na už postavenú časť ropovodu a druhá je kvôli možnému budúcemu rozšíreniu (tzv. budúca rúra). Úseky sú jedného z troch druhov:



Prvý druh úseku sa skladá z rúry spájajúcej stred severnej a východnej strany políčka a rúry spájajúcej stredy západnej a južnej strany. Druhý druh vznikne otočením prvého o 90°. Posledný druh sa skladá z dvoch krížiacich sa rúr spájajúcich severnú s južnou a východnú so západnou stranou políčka.

Môže sa stať, že rúra novopostaveného úseku, ktorá je jedným koncom pripojená k hotovej časti ropovodu, sa svojím druhým koncom nadpojí na nejaký úsek budúcej rúry. V takomto prípade sa, samozrejme, tento úsek a všetky nadväzujúce úseky stanú súčasťou ropovodu. Nasledujúci úsek ropovodu sa potom stavia na políčku nadväzujúcom na koniec pripojenej časti.

Úloha: Urobte program zjednodušujúci plánovanie stavby ropovodu. V nultom ťahu sa dá vybrať, či ropa na území Slovenska vteká na západ, alebo juh. V každom ďalšom ťahu sa vyberá ďalšie smerovanie ropovodu – písmenom `s`, `j`, `v`, `z` označujúcim svetovú stranu, na ktorú má smerovať rúra nového úseku pripojená na doteraz postavenú časť.

Pre $n = 3$, hra po šiestom ťahu:

```
.................
.       /  XXXXX.
.     /   /XXXXX.
.       /  XXXXX.
.  /    |    \  .
./   /--+--\   \.
.  /    |    \  .
.YYYYY  \    /  .
.YYYYY\   \/   /.
.YYYYY  \    /  .
.................
```

Boli stlačené klávesy:

```
0.ťah: 'z'
1.ťah: 'j'
2.ťah: 'j'
3.ťah: 'v'
4.ťah: 's'
5.ťah: 'z'
6.ťah: 'j'
```

z1735. Zimbabwejský internet III.

Zimbabwejský internet nebol ešte nikdy na kráľovom hrade taký populárny ako teraz, keď sa sám veľký Zimba-Zimba naučil HTML stránky vytvárať. Vytváral, vytváral, až jedného pekného zimného dňa prišiel na to, že vytvorenými stránkami zaplnil celý disk, a že sa mu tam už žiadna nová stránka nepomestí. Rozhodol sa preto vymazať všetky nepotrebné stránky, teda také, na ktoré sa nedá dostať prechádzaním po odkazoch z hlavnej kráľovskej stránky. Na to najprv potrebuje zistiť, na ktoré stránky sa to vlastne dostať dá. Napíšte program, ktorý mu s tým pomôže, inak bude musieť vymazať celý disk a začať tvoriť všetky stránky od začiatku.

Úloha: Na disku je uložené množstvo HTML súborov (s príponou `.HTM`) obsahujúcich stránky vytvorené Zimbom. Hlavná kráľovská stránka je uložená v súbore `INDEX.HTM`. Všetky stránky majú rovnaký tvar ako v predchádzajúcich úlohách, teda môžu obsahovať príkazy `<HTML>`, `</HTML>`, `<BODY>`, `</BODY>`, `<CENTER>`, `</CENTER>`, `<P>`, `<IMG SRC="...">`, `<A HREF="...">` a `</A>`. Príkaz `<A HREF="MENO.HTM">` na HTML stránke znamená odkaz na stránku uloženú v súbore s menom `MENO.HTM`. Odkazovaná stránka môže, samozrejme,

obsahovať ďalšie odkazy. Hovoríme, že zo stránky A sa dá dostať na stránku B, ak existuje taká postupnosť stránok $A = s_1, s_2, \ldots, s_n = B$, že stránka s_i obsahuje odkaz na stránku s_{i+1} pre $1 \leq i < n$. Váš program má vypísať mená súborov so všetkými stránkami, na ktoré sa dá dostať z hlavnej stránky uloženej v súbore `INDEX.HTM`. Na poradí vypisovania nezáleží, ale každú stránku vypíšte iba raz.

Príklad:

Súbor `INDEX.HTM`:

```
<HTML><BODY>
<CENTER>Kralovska stranka</center><p>
<A HREF="MENO.HTM">Ako sa volam</a><p>
<A HREF="SUSEDIA.HTM">Susedne krajiny</A>
</BODY></HTML>
```

Súbor `MENO.HTM`:

```
<HTML><BODY>
Ja som Zimba-Zimba <IMG SRC="ZIMBA.JPG"><p>
Chodim rad do <A HREF="KINO.HTM">kina</A>.
</BODY></HTML>
```

Súbor `KINO.HTM`:

```
<HTML><BODY>Toto je nase kino.</BODY></HTML>
```

Súbor `SUSEDIA.HTM`:

```
<HTML><BODY>
Tato stranka sa prave pripravuje.
Mozete sa vratit <A HREF="INDEX.HTM">naspat</A>.
</BODY></HTML>
```

Výstup vášho programu:

```
INDEX.HTM  MENO.HTM  KINO.HTM  SUSEDIA.HTM
```

1811. O vybíjanej

V Nalomenej Trieske sa uskutočnil každoročný turnaj vo vybíjanej. Zúčastnilo sa ho toľko družstiev ako snáď ešte nikdy, ba dokonca až hen z Dolného Kelčova jedno prišlo. Odohrali sa stretnutia systémom každý s každým, organizátori si zaznačili výsledky, no zrazu nastali problémy. Šéf organizačného výboru Kleofáš sa totiž zháčil. „Čo my budeme robiť? Zabudli sme si značiť skóre a teraz sa nám ľahko môže stať, že niektoré dve mužstvá budú na tom istom mieste. Ale ktoré potom dostane ktorú cenu?“ Všetci sa hlboko zamysleli a výsledok sa hneď dostavil. „Mám to!“ zaradoval sa Zachariáš. „Veď my sme vlastne nikdy nevyhlásili, ako budeme turnaj hodnotiť! Keby sa nám podarilo družstvá zoradiť tak, že každý prehral s tým, kto je o jedno miesto pred ním, nik by sa nemohol sťažovať a všetko by bolo v poriadku!“ No vlna nadšenia čoskoro opadla, keď zistili, že nájsť také poradie vôbec nie je jednoduché.

Úloha: Napíšte program, ktorý načíta počet tímov n a následne sa bude pýtať užívateľa na výsledky jednotlivých zápasov. Váš program by mal vypísať jedno možné poradie družstiev $d_1, \ldots, d_n$ také, že pre všetky i, $1 \leq i < n$, družstvo d_i vyhralo nad družstvom d_{i+1}, prípadne „`Smola`“, ak také poradie neexistuje.

Príklad:

```
Počet tímov?
> 3
Víťaz zápasu 1 2?
> 2
Víťaz zápasu 1 3?
> 1
Poradie: 2 1 3
```

1812. O prepúšťaní v TSSL

TSSL (Top Secret Scientific Laboratory) patrí medzi najväčšie centrá vedeckého výskumu. Avšak ani v TSSL ešte nevynašli strom, na ktorom rastú peniaze, a preto sa aj tam muselo pristúpiť k operácii s krycím menom PPP (prepúšťanie prebytočných pracovníkov). No a za obeť padli tentokrát vedci.

Momentálne má každý vedec pridelený presne jeden projekt. To je, pravdaže, veľké plytvanie, lebo niektorí vedci by zvládli aj dva projekty, a tak by mohli byť nejakí iní prepustení. Kvôli utajeniu nesmie žiaden vedec robiť na viac ako dvoch projektoch. Navyše vedcov neslobodno preťažovať príliš náročnými projektami.

Úloha: Daný je aktuálny počet vedcov n a ich maximálna pracovná doba m. Ďalej je pre každý z n projektov daná jeho časová náročnosť, pričom projekty sú na vstupe podľa nej usporiadané zostupne. Jeden vedec môže dostať na starosť dva projekty len vtedy, ak súčet ich náročností neprekročí m.

Napíšte program, ktorý vypočíta maximálny počet vedcov, ktorých možno prepustiť. Taktiež nech vypíše ľubovoľné prípustné rozdelenie projektov medzi neprepustených vedcov.

Príklad:

Vstup:
```
6 50
44 29 26 25 10 8
```

Výstup:
```
Vedec #1: projekt 1
Vedec #2: projekty 2 a 5
Vedec #3: projekty 4 a 6
Vedec #4: projekt 3
Prepustených vedcov: 2
```

1813. O najkratšom tuneli

Nebezpečného zločinca medzinárodného formátu Viliama Bránu konečne zatkli a posadili, kam patrí – do prísne stráženého väzenia na ostrov Alcatraz. Ľudia z blízkeho i širokého okolia si vydýchli. Vydýchli si dokonca aj Viliamovi leniví kumpáni, ktorí dúfali, že po dlhých rokoch driny si konečne doprajú trochu oddychu. No ich radosť netrvala dlho. S hrôzou totiž zistili, že zabudli kombináciu od Viliamovho trezoru, kde bol uložený celý ich spoločný majetok. Preto sa rozhodli, že Viliama musia dostať na slobodu.

Viliamovi kumpáni sa preplavili na ostrov, ktorý je k Alcatrazu najbližšie. Aby vyslobodili svojho šéfa, rozhodli sa vykopať tunel, ktorý by spájal ostrov, na ktorom sa nachádzajú, s ostrovom Alcatraz. No keďže nie sú priveľmi usilovní, veľmi im záleží na tom, aby bol vykopaný tunel najkratší možný. A tak teraz smutne sedia nad mapou oboch ostrovov a premýšľajú, ako tunel vykopať. Je to však úloha nad ich sily, a preto im budete musieť pomôcť vy.

Úloha: Sú dané dva mnohouholníky. Každý z nich je popísaný tak, že najskôr máme na vstupe jeho počet vrcholov a následne zoznam súradníc jeho vrcholov v poradí, v akom ležia na obvode. Obvod mnohouholníka sám seba nikde nekrižuje, mnohouholníky však nemusia byť konvexné ani disjunktné. Napíšte program, ktorý vypočíta najkratšiu možnú dĺžku tunela, ktorý spája oba ostrovy (t.j. dĺžku najkratšej úsečky, ktorej jeden koniec leží v prvom mnohouholníku a druhý koniec leží v druhom mnohouholníku).

Príklad:

Vstup:
```
5
1 1   3 1   2 2   3 3   1 3
3
3 2   4 3   4 1
```

Výstup:
```
Najkratsi tunel: 0.70710678
```

Vstup:
```
4
1 0   2 1   1 2   0 1
4
2 0   3 1   2 2   1 1
```

Výstup:
```
Najkratsi tunel: 0.00000000
```

1814. O slepačej farme

V Štuchaniciach pri Koryte v rámci celonárodného boja proti nezamestnanosti založili slepačiu farmu. A keby len to, dokonca ešte aj nejaké sliepky kúpili. Nuž a ako áno, ako nie, urodilo sa im už onedlho na nej prvých pár vajíčok. A pomerne rýchlo vyrástli. Večer nič a ráno...

„Hľa! Vajíčka!“ zbehla sa hneď celá dedina. „Tie sú určite najkrajšie v kraji. Ba čo, na svete!“ prekrikovali sa. „A určite aj najlepšie!“ „A určite aj najpevnejšie...“ poznamenal sused Záviš, ktorý mal zlú náladu, lebo v jeho kravíne sa za celých päť rokov ani jedno vajíčko neurodilo.

„Áno, áno!“ kričali dedinčania, no potom sa zarazili. Že sú najkrajšie, to na nich vidieť na prvý pohľad. Ale ako zistia, či sú najpevnejšie? „To je ľahké,“ poučil ich Záviš. „Pevnosť vajíčka sa meria tak, že je to najvyššia výška v centimetroch, z ktorej pád vajíčko prežije bez toho, aby sa rozbilo.“ A tak dedinčania zobrali úrodu a začali merať. Len sused Záviš sa oprel o plot a pozoroval so škodoradostným výrazom v tvári, ako Gejza zdrapil prvé vajíčko, pustil ho z výšky n cm a to po krátkom lete dopadlo na zem a rozbilo sa na márne kúsky.

Úloha: Napíšte program, ktorý načíta výšku n, z ktorej sa už vajíčko určite rozbije a počet zvyšných vajíčok m. Potom bude postupne písať výšky v centimetroch, z ktorých majú dedinčania pustiť vajíčko a z klávesnice zakaždým načíta, či sa vajíčko rozbilo alebo nie. Keďže dedinčania sú netrpezliví, treba, aby váš program (v najhoršom prípade) potreboval čo najmenej spustení vajíčka. Nezáleží na tom, koľko vajíčok počas hľadania kritickej výšky váš program rozbije. Samozrejme, najneskôr po rozbití posledného vajíčka už musí hľadanú výšku vedieť presne.

Príklad:
```
> n 10 m 2
Spusť z výšky 1 cm
> nerozbilo sa
Spusť z výšky 3 cm
> nerozbilo sa
Spusť z výšky 5 cm
> rozbilo sa
Spusť z výšky 4 cm
> nerozbilo sa
Vajíčka sa rozbijú pri spustení z výšky >= 5 cm.
```

1815. O Santolande I

Jedného pekného letného dňa si zlatokop Santo zmyslel, že je na zlatokopectvo starý. A pri tej príležitosti sa rozhodol založiť zábavný park pre zlatokopov – Santoland.

Najväčšou atrakciou Santolandu má byť bludisko, takzvaná Rumová jaskyňa. Rumová jaskyňa pozostáva z n komôr, pre jednoduchosť očíslovaných 1 až n. Medzi komorami je množstvo úzkych tunelov. Do každého z nich sa na šírku zmestí len jeden zlatokop. Aby nedochádzalo k zrážkam zlatokopov v tuneloch, Santo do každého tunela nakreslil šípku, ktorým smerom sa v ňom má ísť. A aby všetci videli, že sa konečne naučil čítať a písať, na začiatok každého tunela napísal nejaké písmeno. No a nakoniec vybral niekoľko komôr a v každej ukryl fľašu rumu.

Len čo Santo dokončil prípravy, prišli prví návštevníci. Santo dal každému z nich papierik, na ktorý napísal akúsi postupnosť písmen a povedal: „Môžete vstúpiť do Rumovej jaskyne a prechádzať sa po nej, ako len chcete. Ale kto by z pokladov Rumovej jaskyne ochutnať chcel, pravidlá dodržiavať musí.“ A ukázal na tabuľu s pravidlami, na ktorej stálo:

– V komore č. 1 svoju púť začne.
– Predpísané smery v tuneloch dodržiavať bude.
– Len do takého tunela vôjde, na začiatku ktorého rovnaké písmeno ako prvé nevyškrtnuté písmeno na papieriku je. Po prejdení tunela toto písmeno z papiera vyškrtnúť musí.
– Ak žiadny tunel z komory na požadované písmeno nevedie, zlatokop sa s hanbou z jaskyne domov pobrať musí.
– Komu už žiadne písmeno na papieriku nezvýšilo, v komore, v ktorej sa práve nachádza, fľašu rumu vyhľadať môže.

Zlatokopi sa naskutku aj s papierikmi rozpŕchli po jaskyni fľaše rumu hľadať. Po pár hodinách blúdenia však zistili, že nájsť podľa papierika tú správnu komoru vôbec nie je jednoduché. Problém je totiž v tom, že z niektorých komôr vedie na jedno písmenko viacero tunelov, takže nie je jasné, ktorým smerom sa treba vydať. Nuž sa zlatokopi stretli, dali hlavy dokopy a radili sa, čo ďalej. Napadlo ich, že im Santo mohol dať papieriky, podľa ktorých sa vôbec nedá do komôr s rumom prísť. Aj by Santa zbili, ale najprv si chcú byť načistom, že ich oklamal. Na to však budú potrebovať vašu pomoc.

Úloha: Na vstupe dostanete počet komôr n, počet tunelov m, počet komôr s rumom r, čísla týchto komôr a popis tunelov. Každý tunel je určený číslom začiatočnej a koncovej komory a písmenom napísaným na jeho začiatku.

Napíšte program, ktorý načíta tieto údaje a následne pre zadanú postupnosť písmen na papieriku zistí, či existuje trasa podľa Santových pravidiel. Trasa teda musí začínať v komore 1, musí prechádzať tunelmi označenými rovnakými písmenami a v rovnakom poradí ako na papieriku a musí končiť v nejakej komore s rumom. Trasa môže prechádzať viackrát cez tú istú komoru. Nezáleží pritom na tom, či trasa prechádza cez nejaké komory s rumom, dôležité je len to, v ktorej komore končí.

Príklad:

Vstup:

```
n=4 m=7 r=2
komory s rumom: 2 4
tunely:
1 3 a    1 4 b    2 1 a    2 3 a
3 4 b    4 2 b    4 3 a
papierik: abbababb
```

Výstup:

```
Trasa existuje.
```

(Jedna taká trasa vedie cez komory 1, 3, 4, 2, 3, 4, 3, 4, 2, iná cez 1, 2, 4, 2, 1, 4, 3, 4, 2.)

1821. O bojovej sekere

Dávno-pradávno, keď ešte po šírej prérii putovali stáda bizónov, žil aj Indián Ôten Niw. Tento Indián mal raz bláznivý nápad – rozhodol sa zjednotiť všetky indiánske kmene. Rozoslal preto k všetkým kmeňom poslov s pozvaním na mierovú poradu. Poslovia sa však rýchlo vrátili. „Pri jednom bizóňom stehne s Komančom, Utahom a Kiowom? Či nevieš, že sme proti nim vykopali bojovú sekeru? Jedného by som ešte zniesol, ale všetkých tých prašivých psov naraz? Nikdy!“ odkázal náčelník Siouxov. A tak to bolo so všetkými – jednému sa nepáčili tí, druhému zase iní. Ôten sa však nevzdal a poslal druhé pozvanie, kde každému sľúbil, že viac ako jedného nepriateľa nestretne. A ako to chcel zariadiť? Nuž, rozhodol sa, že zorganizuje viac stretnutí súčasne, jednotlivých náčelníkov rozdelí do skupín a každú skupinu pozve na iné miesto. Zorganizovať také stretnutie však nie je jednoduché, a preto by skupín malo byť čo najmenej.

Teraz sedí nad zoznamom vykopaných bojových sekier a rozmýšľa, ako tých tvrdohlavých náčelníkov rozdeliť. Sám by na to nestačil, a preto požiadal o pomoc kmeňového medicinmana. Ten mu odpovedal: „Vedz, že i ten najodvážnejší si netrúfne na viac ako troch.“ Napriek tejto (samozrejme presnej a pravdivej) odpovedi Ôten neprišiel na nič, a tak sa rozhodol požiadať o pomoc vás.

Úloha: Daný je počet všetkých kmeňov n, počet znepriatelených dvojíc kmeňov m a ich zoznam. Ak je kmeň a znepriatelený s kmeňom b, znamená to zároveň, že kmeň b je znepriatelený s kmeňom a. Každý kmeň má, ako už povedal medicinman, nanajvýš troch nepriateľov. Vašou úlohou je nájsť také rozdelenie kmeňov do skupín, aby tých skupín bolo čo najmenej a v rámci jednej skupiny mal každý kmeň nanajvýš jedného nepriateľa.

Príklad:

Vstup:	Výstup:

`4`
`4`
`1 2  1 3  2 3  2 4`
	`Stačia dve skupiny.`
	`skupina 1: kmene 1 3`
	`skupina 2: kmene 2 4`

1822. O byrokracii v Bratislave

Raz úradoval v Bratislave na Úrade záležitostí a pokút byrokrat Byrokrat a poslal každému Bratislavčanovi list zhruba tohto obsahu:

„Dostavte sa na Úrad záležitostí a pokút najneskôr do 14:37 dňa 15. 9. tohto roku ohľadom prejednania záležitosti č. 578 týkajúcej sa oznámenia 356/1996 Z. z. Ak sa nedostavíte do uvedeného dátumu, bude vám vyrúbená pokuta 823,- Sk.“ Byrokrat Byrokrat bol dôsledný byrokrat. V liste uviedol každému Bratislavčanovi nejaký čas, dátum a pokutu. Niektorým ľuďom mohol uviesť rovnaký čas a dátum.

Každé jedno vybavenie na úrade trvá síce len jednu minútu, ale Bratislavčanov je veľa a je preto možné, že nie každý stihne vybaviť svoju žiadosť do uvedeného času a bude platiť pokutu. Preto sa pokúste poradiť každému Bratislavčanovi, kedy má prísť, aby súčet pokút zaplatený celou Bratislavou bol minimálny.

Úloha: Je dané n – počet Bratislavčanov. Pre každého z nich dostanete číslo t_i – čas v minútach (meraný od teraz), do ktorého musí dotyčný človek stihnúť prísť na úrad. Následne pre každého Bratislavčana dostanete číslo p_i – pokutu, ktorú bude platiť, ak to nestihne. Vašou úlohou je pre každého Bratislavčana určiť čas v minútach, kedy má prísť na úrad, a to tak, aby súčet všetkých zaplatených pokút bol najmenší možný. Navyše, žiadni dvaja Bratislavčania nesmú prísť na úrad naraz, aby sa nepobili o to, kto prišiel skôr.

Príklad:

Vstup:	Výstup:

`n:   5`
`t:   0   3   2   1   1`
`p: 100  10   1   8   9`
	`Časy príchodu: 0 3 2 4 1`
	`Súčet pokút: 8`

1823. O zabudnutej krajine

Kde bolo, tam bolo, možno niekedy niekto vedel, že kde, ale už všetci zabudli... No, skrátka a dobre, bola raz jedna zabudnutá krajina. V tej krajine už zabudli snáď všetko. Nevedia dokonca ani len to, odkiaľ kam ich krajina siaha. Aj to voľakedy vedeli, len to už všetci zabudli. S najväčšou snahou si každý obyvateľ teraz pamätá jednu priamku. No a na stĺp v hlavnom meste niekto niekedy napísal, že všetky priesečníky týchto priamok ležia v zabudnutej krajine. No a teraz by obyvatelia zabudnutej krajiny chceli vedieť, ako ďaleko na východ ich krajina siaha. A podľa možností čo najrýchlejšie, kým nezabudnú, že to chceli.

Úloha: V rovine je daných n rôznych priamok. Každá priamka je daná dvoma rôznymi bodmi, ktoré na nej ležia. Tieto priamky sa pretínajú v niekoľkých bodoch. Vašou úlohou je

napísať program, ktorý nájde najpravejší spomedzi nich, čiže ten z nich, ktorý má najväčšiu x-ovú súradnicu. Ak ich je viac, môžete vypísať ľubovoľný z nich.

Príklad:

Vstup:
```
4
0 1   0 3
2 0   3 1
2 0   2 2
0 3   2 2
```

Výstup:
```
najpravejší priesečník: 4 1
```

1824. O nudnej prednáške

„Ták-žé, táto entitno-relačná, akože...“ Prednáškovou miestnosťou sa niesol monotónny hlas prednášajúceho. Kleofáš a Zachariáš zaspávali. A spolu s nimi asi osemdesiat ďalších študentov. „Zachariáš, musíš mi pomôcť! Keď spím, hrozne chrápem a všimne si ma!“ „Ták... mám to! Môžme hrať lodičky,“ navrhol Zachariáš. A začali hrať. Pre tých, čo túto hru nepoznajú: lodičky sa hrajú tak, že jeden hráč rozmiestni svoje lodičky na hracej ploche. Potom sa druhý hráč postupne pýta na políčka hracej plochy a dozvedá sa, či tam loďka je alebo nie je.

Ako tak hrali, zrazu sa Zachariáš zadíval na hraciu plochu a zahlásil: „O kofolu, že už nájdem všetky tvoje lodičky a pritom sa najviac raz pomýlim!“ Kleofáš sa tiež zamyslel. Má vôbec šancu, že vyhrá kofolu?

Úloha: Dané sú dve čísla r, s – rozmery hracej plochy. Niekde na hracej ploche sa nachádza presne raz každá z nasledujúcich siedmich lodičiek:

```
XX   X     X   XX    XX    X
XX   XXX   XXX   XX   XX   XXX   XXXX
```

Lodičky môžu byť otočené, ale nie preklopené, môžu sa dotýkať, nesmú sa prekrývať. Ďalej je na vstupe aktuálna hracia plocha – r riadkov po s znakoch. Skutočnosť, že na danom políčku je časť nejakej loďky, značíme znakom „`+`“ (plus) vodu označuje „`-`“ (mínus) a políčko, ktoré ešte nebolo odkryté, označuje „`.`“ (bodka).

Vašou úlohou je napísať program, ktorý vypíše na výstup „`ÁNO`“ alebo „`NIE`“ podľa toho, či sa v tejto pozícii dajú určite nájsť všetky lodičky (čiže určiť pre každé políčko hracej plochy, či je na ňom lodička alebo nie) s tým, že najviac raz netrafíme lodičku.

Príklad:

Vstup:
```
10 10
.+..+.....
-----+----
-+--+++...
++------..
+---+---..
--++++--..
-----++--+
------+--+
--------++
----------
```

Výstup:
```
ÁNO
```

1825. O Santolande II

Už ste niekedy boli v Santolande? A poznáte Rumovú jaskyňu? Že nie? Tak to si najprv prečítajte zadanie úlohy 1815. Tam nájdete jej popis ako i pravidlá blúdenia v nej.

Santo raz postavil takúto jaskyňu, zvolal zlatokopov a rozdal im papieriky. Zlatokopi z celého okolia blúdia bludiskom, avšak fľašu rumu nájsť nevedia. Až prišlo Santovi zlatokopov ľúto. „Toľko sa snažia... Čo keby som im to hľadanie trochu uľahčil? Nech vedie

z každej komory na jedno písmeno najviac jeden tunel. To budú vždy vedieť, kade sa pobrať ďalej. Len keby som vedel vyrobiť takú jaskyňu, v ktorej by sa dala fľaša rumu nájsť s presne tými istými papierikmi, ako v tej, čo mám!“

Úloha:

a) Máte danú nasledujúcu jaskyňu:
$n = 6$, tunely: $(1, 1, a)$, $(1, 1, b)$, $(1, 2, a)$, $(2, 3, b)$, $(3, 3, a)$, $(3, 3, b)$, $(3, 4, b)$, $(4, 5, b)$, $(5, 6, a)$, $(6, 6, a)$, $(6, 6, b)$, komora s fľašou rumu: 6
Vašou úlohou je navrhnúť novú jaskyňu. Z jednej komory v tejto jaskyni môže na jedno písmenko vychádzať max. 1 tunel. Navyše sa v novej jaskyni musí dať získať fľaša rumu práve s tými papierikmi ako v starej jaskyni.

b) Napíšte program, ktorý načíta počet komôr n, popis tunelov (každý tunel je určený začiatočnou a koncovou komorou a písmenom napísaným na začiatku) a zoznam komôr, v ktorých sú ukryté fľaše rumu. Od vás sa čaká na výstupe popis jaskyne, v ktorej z každej komory vychádza na jedno písmenko max. 1 tunel a fľaša rumu sa dá v nej získať práve s tými papierikmi ako v zadanej jaskyni. Zdôvodnite správnosť vášho algoritmu.

1831. O zúfalom programátorovi

Kde bolo, tam bolo, kdesi pod horami stál domček. Keby ste k nemu potichúčky prišli a nazreli cez okno, zbadali by ste pri práci programátora. Nazvime ho trebárs Mišo. Práve si sadol k počítaču, vedľa seba položil tácku s čajom a koláčikmi, na počítači pustil hudbu, proste idylka... Zrazu buchol päsťou po stole. „To snáď ani nie je pravda! Ten blbý prehrávač tie pesničky zase prehádzal! V takýchto podmienkach sa naozaj nedá pracovať!“

Preto začal pesničky usporadúvať tak, aby boli v poradí, v akom sú očíslované. Preusporiadať sa dajú jedine tak, že vždy chytí myšou niektorú pesničku a odtiahne ju na nejaké miesto v playliste. Tejto operácii hovorme presun. No a keďže Mišo je snáď ešte lenivejší ako priemerný programátor, chcel by si pesničky usporiadať na čo najmenej presunov. Nechce sa mu však nad tým rozmýšľať, ba čo viac, ani len program na to sa mu nechce písať. Tak to nechal na vás, veď vy ste predsa všetci takí šikovní a snaživí...

Úloha: Na vstupe je dané n – počet pesničiek v albume a n čísel od 1 po n (každé práve raz) – čísla pesničiek v poradí, v akom sú teraz v playliste. Vašou úlohou je napísať program, ktorý vypíše minimálny počet presunov potrebný na usporiadanie playlistu. V usporiadanom playliste majú byť pesničky, samozrejme, v poradí $1, 2, \ldots, n$.

Príklad:

Vstup:	Výstup:
`5` `2 5 1 3 4`	`2 presuny`

(Jedno riešenie: presunieme pesničku 2 na 3. miesto a potom pesničku 5 na 5. miesto.)

1832. O maškrtnom Dávidkovi

Dávidko je veľmi maškrtný, a tak nečudo, že sa často cestou zo školy zastaví v malom obchodíku na rohu ulice, kde predávajú rôzne oriešky a sušené ovocie. Veľké bolo Dávidkovo potešenie, keď mu minule pri odchode povedali, že je miliónty zákazník a môže si niekedy večer, po záverečnej, prísť nabrať orieškov a sušeného ovocia, koľko mu taška unesie. Dávidko teraz sedí a premýšľa, ako by čo najcennejšie jedlo z obchodu odniesol...

Úloha: Na vstupe je dané číslo n, udávajúce počet druhov maškŕt, ktoré v obchode po záverečnej ostali. Ďalej je pre každú maškrtu daná jej celková hmotnosť a tiež celková cena za celú tú hmotnosť. Na záver je daná nosnosť t Dávidkovej tašky.

Vypíšte, koľko si má Dávidko z jednotlivých druhov maškŕt navážiť, aby doniesol domov maškrty s čo najväčšou celkovou cenou. Pritom samozrejme môže odniesť nanajvýš toľko, koľko jeho taška unesie.

Zamyslite sa nad efektívnosťou vášho algoritmu. Prvé efektívne riešenie, ktoré vymyslíte, veľmi pravdepodobne nebude mať optimálnu časovú zložitosť.

Príklad:

Vstup:
```
5
3
3.0  15.0
2.5 133.0
1.0  79.5
```

Výstup:
```
1. maškrta: 1.5 kg
2. maškrta: 2.5 kg
3. maškrta: 1.0 kg

celková cena: 220.0
```

(Odnesie $1.5 + 2.5 + 1.0 = 5$ *kg vecí; celková cena je* $7.5 + 133.0 + 79.5 = 220.0$ *korún.)*

1833. O Santovi, Bantovi a parcelách II

Santo a Banto si na Vyšnej Klondike opäť našli niekoľko zlatých žíl. Celí natešení teda utekali na registračný úrad pekne si parcelu zapísať. Tam ich ale ako prvý zaujal obrovský papier popisujúci nové zmeny vo vyhláške o parcelách. Podľa nej sa za parcely bude platiť poplatok, ktorý závisí od počtu zlatých žíl vo vnútri parcely. Žily na okraji sa nezapočítavajú. Ak by si tesne vedľa nich prenajal parcelu nejaký iný zlatokop, má na zlatú žilu na rozhraní rovnaký nárok a je iba na nich, ako sa dohodnú. „Tak to teda nie. My im cez rozum prejdeme a za parcelu nič platiť nebudeme," prehlásil po krátkom zamyslení Banto a rozbehol sa vypĺňať formulár. Hneď však zastal na kolonke „Obsah parcely".

Úloha: Napíšte program, ktorý pre zadané číslo n a n bodov v rovine nájde nejaký mnohouholník taký, že všetkých n zadaných bodov leží na jeho obvode. Následne vypíšte tento mnohouholník a tiež jeho obsah.

Príklad:

Vstup:
```
6
1  4    -1 2     0 2
0 -1     2 0    -1 0
```

Výstup:
```
5 vrcholov: [-1,-1], [3,-1],
[0,2], [1,4], [-1,2]
obsah: 8.5
```

1834. O futbalovom družstve

Na jednom gymnáziu sa medzi triedami každoročne organizuje futbalový turnaj. Milan, najlepší futbalista 4.B, sa rozhodol vybrať z triedy družstvo, ktoré to tento rok musí vyhrať. Poobede zavolal všetkých svojich spolužiakov na futbalové ihrisko. Každý Milanovi predviedol, ako je dobrý v útoku, v poli a v obrane a on každému priradil tri čísla určujúce jeho kvalitu. Potom Milan zašiel za Rišom, najlepším programátorom 4.B a poprosil ho, aby na základe týchto čísel zostavil najlepšie možné futbalové družstvo. Rišo je však už unavený a musí sa ísť domov učiť, preto vás poprosil, aby ste zostavili družstvo vy.

Úloha: Je dané n – počet futbalistov v triede a čísla u, p, o hovoriace, že chceme zostaviť družstvo zložené z u útočníkov, p poliarov a o obrancov. Pre každého futbalistu poznáme tri čísla u_i, p_i, o_i – jeho kvalitu v útoku, v poli a v obrane. Zostavte družstvo tak, aby súčet kvalít útočníkov v útoku, poliarov v poli a obrancov v obrane bol najväčší možný. Vašou úlohou je vypísať tento súčet.

Príklad:

Vstup:
```
8 2 2 1
1 2 3
10 5 4
6 7 9
0 1 12
1 2 3
1 5 4
6 7 9
9 1 0
```

Výstup:
```
Najvyššia možná kvalita: 45
```

(Útočníkmi budú hráči #2 a #8, do poľa dáme hráčov #3 a #7 a do obrany hráča #4. Celková kvalita bude 10+9+7+7+12 = 45.)

1835. O Santolande III

Tým, čo ešte stále nemali možnosť zoznámiť sa so Santom a jeho úžasnými rumovými jaskyňami, odporúčame prečítať si zadanie úlohy 1815.

Ako si Santo povšimol, zlatokopi sú ľudia obdarení mimoriadnym šťastím. V danej jaskyni a s danou postupnosťou symbolov na papieriku je obyčajne veľa možností, ako voliť trasu (pretože z niektorých komôr na daný znak môže vychádzať viacero tunelov). Avšak ako si Santo povšimol, ak existuje čo len jedna postupnosť výberov tunelov, ktorá zlatokopa dovedie k rumu, môžete si byť istí, že zlatokop ju nájde.

Úloha:

a) Santo v slabej chvíľke navyrábal množstvo papierikov a na každý z nich napísal nejakú postupnosť zloženú z cifier $0 \dots 9$. Na tú sa môžeme dívať ako na číslo v desiatkovej sústave. A keďže nevedel, čo s toľkými papierikmi robiť, rozhodol sa postaviť rumovú jaskyňu, ktorej tunely budú tiež pooznačované ciframi $0 \dots 9$ takú, že pre danú postupnosť cifier $c_1 c_2 \dots c_k$ na papieriku existuje cesta z komory 1 do niektorej z rumových komôr práve vtedy, keď číslo $c_1 c_2 \dots c_k$ v desiatkovej sústave je deliteľné 6. Vašou úlohou, ako už iste tušíte, bude navrhnúť pre Santa túto rumovú jaskyňu. V rámci úsporných opatrení sa snažte, aby mala čo najmenej komôr.

b) Santovi sa raz sníval takýto sen: Majme dve rumové jaskyne A a B. Nech $a_1 a_2 \dots a_k$ je postupnosť znakov na papieriku určená pre jaskyňu A a $b_1 b_2 \dots b_\ell$ je postupnosť znakov určená pre jaskyňu B. Majme zlého Banta, ktorý za temnej noci zoberie nožnice a rozstrihá každý papierik na m kúskov. Každý kúsok obsahuje súvislý úsek niekoľkých (možno aj 0) znakov z príslušného papierika. Potom vezme prvý kúsok z prvého papierika, prilepí k nemu prvý kúsok z druhého papierika, pridá druhý kúsok z prvého, druhý kúsok z druhého, striedavo priliepa kúsky z prvého a druhého papierika, až dostane novú postupnosť znakov $a_1 \dots a_{i_1} b_1 \dots b_{j_1} \dots a_{i_{m-1}+1} \dots a_{i_m} b_{i_{m-1}+1} \dots b_{j_m}$. Napríklad z postupností `VYSNA` a `KLONDIKA` by takto mohol dostať napríklad `KLOVYNDISNAKA`.

   V tomto momente sa Santo zobudil. A vymyslel pre vás úlohu. Predtým však dve definície. Postupnosť znakov $a_1 \dots a_k$ pre jaskyňu J sa nazýva *rumová*, ak existuje trasa podľa tejto postupnosti, ktorá v jaskyni J vedie do komory s rumom. Hovoríme, že postupnosť $c_1 \dots c_{k+\ell}$ je *zmesou* postupností $a_1 \dots a_k$ a $b_1 \dots b_\ell$, ak vznikla postupom zo Santovho sna. Napíšte program, ktorý načíta popisy dvoch rumových jaskýň A a B a vypíše popis rumovej jaskyne C takej, že postupnosť c je rumová pre jaskyňu C práve vtedy, ak je zmesou nejakej rumovej postupnosti a pre jaskyňu A a nejakej rumovej postupnosti b pre jaskyňu B.

Príklad:

Vstup:
Jaskyňa A:
1 komora
tunely: $(1, 1, a)$
1 rumová komora: 1
Jaskyňa B:
2 komory
tunely: $(1, 2, b)$, $(2, 2, b)$
1 rumová komora: 2

Výstup:
Jaskyňa C:
2 komory
tunely: $(1, 1, a)$, $(1, 2, b)$, $(2, 2, a)$, $(2, 2, b)$
1 rumová komora: 2

Pre jaskyňu A je postupnosť na papieriku rumová, ak sa skladá z 0 alebo viacerých písmen `a`. *Pre jaskyňu B je postupnosť na papieriku rumová, ak sa skladá z 1 alebo viacerých písmen* `b`. *Výstupom bude jaskyňa C, pre ktorú je postupnosť na papieriku rumová, ak sa skladá z písmen* `a` *a* `b` *a obsahuje aspoň jedno písmeno* `b`.

1841. O chemických zlúčeninách

Sepro študuje za biochemika. Bude raz z neho taký veľký chemik, že to tu všetko istotne vyhodí do vzuchu. Ale skôr to tu asi všetko zamorí biotoxínmi. Kým sa tak však stane, musí Sepro ešte veľa žinčice vypiť, veľa klobás pojesť a hlavne sa toho veľa naučiť.

Momentálne však sedí v škole a študuje biomolekuly. Pred sebou má chemickú skladačku a snaží sa poskladať molekuly kyanidu draselného, trinitrotoluénu, tripletu mitochondriálnej deoxyribonukleovej kyseliny a mnohých iných.

Skladačka sa skladá z veľa guličiek s nožičkami reprezentújucich atómy a veľa rovných paličiek rozmanitých dĺžok reprezentujúcich chemické väzby. Ak majú byť dva atómy spojené chemickou väzbou, tak stačí zobrať príslušné guličky a prilepiť ku každej jeden koniec vhodne dlhej paličky. Musíte si však dať pozor, aby ste paličku neohýbali, lebo by sa mohla zlomiť. Preto musíte vedieť, ako rozmiestniť atómy v priestore tak, aby sa žiadne dve väzby nekrížili.

Seprovi sa však nejako ruky pletú, oči krížia, skrátka mu to akosi nejde. Zišlo by sa mu trocha pomôcť.

Úloha: Je daná molekula skladajúca sa z n atómov a m chemických väzieb. Atómy sú očislované celými číslami od 1 po n. Väzby sú zadané dvoma číslami atómov, ktoré spájajú. Vašou úlohou je napísať program, ktorý rozmiestni všetkých n atómov do priestoru, t.j. určí každému atómu jeho súradnice $[x, y, z]$ v priestore tak, aby sa žiadne dve väzby nekrížili. Stačí nájsť jedno ľubovoľné riešenie.

Príklad:

Vstup:

```
5
10
1 2   1 3   1 4   1 5
2 3   2 4   2 5   3 4
3 5   4 5
```

Výstup:

```
1. atóm: 0 0 0
2. atóm: 2 0 0
3. atóm: 0 2 0
4. atóm: 2 2 1
5. atóm: 2 2 -1
```

1842. O nepríjemnostiach

Kým spúšťame naše programy len na malých vstupoch, je všetko v pohode. Horšie je, keď dostaneme väčší vstup. U pomalších programov sa už často ani výsledku dožiť nemusíme. No a skutočné nepríjemnosti začínajú, keď je vstup taký veľký, že sa nám naraz ani len do pamäte nezmestí. Ako napríklad teraz...

Úloha: V súbore máme číslo n a za ním po riadkoch uložené pole $n \times n$ celých čísel. Napíšte program, ktorý uloží do výstupného súboru toto pole preklopené podľa osi $y = x$. (Ak pole chápeme ako maticu, táto operácia sa volá *transpozícia*.) Matica môže byť taká veľká, že nielenže nám nevôjde do pamäte celá, dokonca sa nám do pamäte nemusí zmestiť ani celý jej riadok. Zato na disku je miesta dosť a môžete používať pomocné súbory.

Príklad:

Vstup:

```
4
 1  2  3  4
 5  6  7  8
 9 10 11 47
13 14 15 16
```

Výstup:

```
 1  5  9 13
 2  6 10 14
 3  7 11 15
 4  8 47 16
```

1843. O elixíroch

Zas je tu neporiadok. Ale aký! Zoja, lesná víla, je zúfalá. Zlý škriatok jej počarbal steny, rozburdal postieľku a – povážte sami – zablatil šaty. Najhoršie ale je, že jej poprelieval zázračné fľaštičky. Zoja mala kopu fľaštičiek a v každej z nich mala pôvodne rannú rosu, včelí med alebo mesačný svit. Škriatok si dal záležať, a tak sa teraz v každej fľaštičke nachádzajú všetky tri substancie v nejakom pomere. Zoja by z nich potrebovala namiešať

liečivý elixír, lenže v jej súčasnej situácii je to úloha bez výpočtovej techniky (a vašej pomoci) takmer neriešiteľná.

Úloha: Zloženie fľaštičky budeme udávať pomocou troch celých čísel, udávajúcich pomer objemov jednotlivých zložiek. Teda `3 2 4` znamená, že vo fľaštičke sú 3 diely rosy, 2 medu a 4 mesačného svitu. Inými slovami, nech si z tejto fľaštičky odlejeme, koľko chceme, určite tam bude napríklad dvakrát viac mesačného svitu ako medu.

Na vstupe je daný pomer zložiek v liečivom elixíre. Následne je tam počet Zojinych fľaštičiek a pre každú z nich pomer zložiek v nej. Vašou úlohou je napísať program, ktorý zistí, či sa z roztokov vo fľaštičkách dá namiešať liečivý elixír a ak áno, tak v akom pomere treba obsahy jednotlivých fľaštičiek zmiešať.

Príklad:

Vstup:
```
6 4 5
3
1 2 5
9 3 5
2 2 1
```

Výstup:
```
1. fľaška: 2 diely
2. fľaška: 2 diely
3. fľaška: 5 dielov
```

1844. O novom probléme v Chile

Ako všetci iste viete, v Chile majú s dopravou nemalé problémy. Cestná sieť totiž vyzerá tak, že všetky mestá ležia na priamke a priamou cestou sú spojené vždy len susedné dve. Mestá sú očíslované od 1 po n v poradí, v akom ležia na priamke. Z mesta 1 do mesta n chodí z času na čas autobus, ktorý stojí v každom meste medzi nimi. V každom meste sa dá nastúpiť aj vystúpiť a vie sa, koľko stojí lístok z každého mesta do každého s vyšším číslom. Počas cesty môžeme na niektorých zastávkach vystúpiť, nastúpiť a kúpiť si nový lístok z mesta, v ktorom práve sme.

Toto je dobre známa situácia. Veď niektorí z vás už pomáhali cestujúcim nájsť, ako sa dá najlacnejšie dostať z mesta 1 do mesta n. Juana Carlosa však včera napadla prefíkaná myšlienka. Nikde nie je povedané, že musí mať počas celej cesty pri sebe vždy práve jeden lístok. Stačí mu, keď má aspoň jeden. Ak príde revízor, len vyberie ľubovoľný platný a ukáže mu ho. A dá sa na tom ušetriť. Veď napríklad z Puerta Montt je to do Santiaga lacnejšie ako z Valdivie, ktorá leží po ceste...

Úloha: Na vstupe je počet miest n. Následne sú tam postupne pre každé mesto i ceny lístkov z mesta i do každého mesta j, kde $j > i$. Pomôžte Juanovi ušetriť a napíšte program, ktorý z týchto údajov zistí, aké lístky si má Juan kedy kupovať, aby sa dostal z mesta 1 do mesta n najlacnejším spôsobom.

Príklad:

Vstup:
```
4
7 2 10
6 3
9
```

Výstup:
```
najlepšia cena: 5
lístky: 1 -> 3, 2 -> 4
```
(Lístok z 1 do 3 stojí 2, lístok z 2 do 4 stojí 3.)

1845. O Santolande IV

Ak ešte nepoznáte Santove rumové jaskyne, pozrite si zadania úloh 1815 a 1835.

Úloha: Najobľúbenejšia rumová jaskyňa v Santolande bola jaskyňa A. Táto jaskyňa sa vyznačovala tým, že v každej komore pre ľubovoľný znak existovala najviac jedna možnosť, ktorým tunelom sa mohol zlatokop vybrať. Zlatokopi ju mali radi, lebo pri hľadaní rumu nemuseli veľmi rozmýšľať. Nuž si Santo pre ňu pripravil množstvo papierikov s postupnosťami znakov.

Predvčerom však bolo zemetrasenie a celú jaskyňu A zasypalo. Santa začalo trápiť, čo bude s papierikmi do nej robiť. Zahodiť ich je škoda. Postaviť takú istú jaskyňu znova sa mu tiež nevidelo. Tak sa rozhodol postaviť jaskyňu, v ktorej sa budú postupnosti znakov na papierikoch čítať odzadu. Navyše chce, aby obrátená postupnosť na papieriku bola v novej jaskyni rumová práve vtedy, keď bola rumová v jaskyni A. A keďže zlatokopom neuškodí porozmýšľať, nemusí byť trasa jaskyňou pre postupnosti jednoznačná (na rozdiel od jaskyne A).

Označme postupnosť $b_1 b_2 \ldots b_n$ *reverzom* postupnosti $a_1 a_2 \ldots a_n$, keď platí $b_n \ldots b_2 b_1 = a_1 a_2 \ldots a_n$. Napíšte program, ktorý načíta popis rumovej jaskyňe A a vypíše popis rumovej jaskyňe B takej, že *reverz* postupnosti a je rumový pre jaskyňu B práve vtedy, keď a je rumová pre jaskyňu A.

Príklad:

Vstup:	Výstup:
Jaskyňa A:	Jaskyňa B:
2 komory	2 komory
tunely: $(1,1,a)$, $(1,2,b)$, $(2,2,c)$	tunely: $(1,1,c)$, $(1,2,b)$, $(2,2,a)$
1 rumová komora: 2	1 rumová komora: 2

Pre jaskyňu A je postupnosť na papieriku rumová, ak začína niekoľkými `a`*, nasleduje jedno* `b` *a končí niekoľkými* `c`*. Pre jaskyňu B je postupnosť na papieriku rumová, ak začína niekoľkými* `c`*, nasleduje jedno* `b` *a končí niekoľkými* `a`*.*

z1811. Zo života informačnej kancelárie

V meste Kocúrkovo otvorili novú informačnú kanceláriu. Kancelária to bola pekná, otvorená non-stop. A keďže všetkých návštevníkov pri príchode ponúkali pohárom chladeného piva, čoskoro ju začalo navštevovať množstvo ľudí. Bolo ich tak veľa, že sa pred kanceláriou utvoril dlhý rad.

Aby ľudia nemuseli stáť v rade, rozhodli sa v kancelárii nainštalovať rezervačný systém. Každý občan, ktorý sa chce v kancelárii informovať, zadá do počítača umiestneného pri vchode svoje číslo (každý Kocúrkovčan má svoje jednoznačné číslo) a číslo okienka, ku ktorému sa chce ísť informovať (v kancelárii majú k okienok očíslovaných 1 až k a každé z nich poskytuje iný druh informácií), počítač ho zaradí a občan sa môže ísť poprechádzať do neďalekého parku. V okamihu, keď sa občan dostane na rad (t.j. po vybavení všetkých občanov, ktorí prišli pred ním a čakali na to isté okienko) sa na veľkej svetelnej tabuli nad vchodom rozsvieti jeho číslo a občan sa môže dostaviť k okienku. Vašou úlohou je pomôcť riaditeľovi zrealizovať tento ambiciózny projekt.

Úloha: Napíšte program, ktorý bude riadiť privolávanie čakajúcich pomocou svetelnej tabule. Váš program na začiatku načíta počet okienok k. Potom bude na vstupe dostávať správy tvaru „`PRISIEL` x o“ a „`UVOLNILO SA` o“. Prvý druh správy oznamuje, že občan číslo x prišiel do kancelárie a chce ísť k okienku o. Druhý druh oznamuje, že od okienka o práve odišiel klient, čím sa toto okienko uvoľnilo. Váš program má vždy, keď sa uvoľní okienko a niekto naň čaká, vypísať číslo občana, ktorý má k dotyčnému okienku ísť. (Toto zahŕňa aj situáciu, keď je okienko voľné a príde občan, ktorý sa chce pri ňom informovať.) Snažte sa, aby váš program reagoval dostatočne rýchlo, aby nedochádzalo k zbytočným zdržaniam. Keďže kancelária funguje non-stop, váš program by mal tiež bežať bez prerušenia.

Príklad:

```
> PRISIEL 20 2
občan 20 k okienku 2
> PRISIEL 11 2
> PRISIEL 15 1
občan 15 k okienku 1
```

```
> UVOLNILO SA 2
občan 11 k okienku 2
> UVOLNILO SA 2
> PRISIEL 99 2
občan 99 k okienku 2
> UVOLNILO SA 1
```

z1812. Zaoceánske Krížniky, Škunery a Plavby

Pedro Prímorský, generálny riaditeľ spoločnosti Zaoceánske Krížniky, Škunery a Plavby, sa rozhodol zveľadiť prístavy spoločnosti a získať tým viac zákazníkov. Peňazí je ale málo, preto chce Pedro vybrať iba najfrekventovanejší prístav. To ale nie je jednoduché, pretože jediné záznamy, ktoré v ZKŠP majú, sú zoznamy liniek, na ktorých prepravujú cestujúcich.

Úloha: Napíšte program, ktorý načíta počet prístavov n, počet liniek ℓ a zoznam dvojíc prístavov, medzi ktorými premáva loď, a vypíše prístav, z ktorého vychádzajú linky do najviac iných prístavov. Ak je takýchto prístavov viac, vypíšte všetky.

Poznámka: Všetky linky sú obojsmerné. Medzi dvojicou prístavov môže premávať viac ako jedna linka.

Príklad:

Vstup:	Výstup:

`5`
`6`
`1 2  2 3  2 5`
`2 1  5 3  4 1`
	`2`
	
	(Z prístavu 2 vychádzajú linky do troch iných prístavov.)

z1813. Zabudnuté čísla

Žil raz jeden matematik, strašný zvedavec to bol. A aj sa volal Zvedavec, lebo tak sa volal jeho otec, dedko, pradedko, ... No a ako sa na správneho zvedavca patrí, chcel preskúmať všetko, čo sa mu dostalo pod ruky. A tak nečudo, že sa pri usilovnom prezeraní knižníc dočítal aj o mínus-dvojkovej sústave. Hneď sa mu táto sústava zapáčila, no vzápätí sa zháčil: a ako to vôbec môže fungovať?

Zvedavca by určite veľmi potešilo, ak by ste mu napísali program, ktorý by mu prevádzal čísla z desiatkovej sústavy do mínus-dvojkovej a naopak. Mínus-dvojková sústava je taká, že používané cifry sú 0 a 1 a základom sústavy je číslo -2. Ak teda napríklad máme v tejto sústave číslo zapísané ciframi 100110, jeho hodnota v desiatkovej sústave je $1 \cdot (-2)^5 + 0 \cdot (-2)^4 + 0 \cdot (-2)^3 + 1 \cdot (-2)^2 + 1 \cdot (-2)^1 + 0 \cdot (-2)^0 = -30$. Vo všeobecnosti, nech $x_{n-1}x_{n-2}\ldots x_2x_1x_0$ sú cifry čísla v mínus-dvojkovej sústave, potom jeho hodnotu dostaneme tak, že sčítame hodnoty $x_i \cdot (-2)^i$ pre všetky i.

Poznámka: Pokúste sa dokázať, že každé celé číslo sa dá zapísať v mínus-dvojkovej sústave práve jedným spôsobom.

Úloha: Na vstupe je vždy najskôr základ sústavy (buď -2 alebo 10) a potom číslo v tejto sústave. Vypíšte číslo s tou istou hodnotou v druhej sústave.

Príklad:

Vstup:	Výstup:
`10 7`	`11011`
`-2 110101`	`-11`

z1814. Zauzlení študenti

Študenti istej nemenovanej fakulty majú kopu divných nápadov. Nedávno ich napadla takáto hra. Zaviazali si oči a náhodne sa pochytali za ruky, pričom každý z nich našmátral pravou rukou nejakú ľavú ruku a pevne sa jej chytil. Teraz by sa chceli rozuzliť. Chvíľu sa

o to neúspešne pokúšali, potom to vzdali a začali rozmýšľať. Nakoniec prišli na to, že by sa hádam aj vedeli rozmotať, no najprv musia vedieť, koľko kruhov vlastne tvoria.

Úloha: Na vstupe je daný počet študentov n a následne n čísel, pričom i-te číslo v poradí je číslo študenta, ktorého sa drží i-ty študent pravou rukou. (Keďže sa chytali za ruky, nemohlo sa stať, že dvaja držia toho istého. Inými slovami, medzi číslami na vstupe je práve raz každé z čísel od 1 po n.)

Napíšte program, ktorý vypíše počet kruhov v spletenci, ktorý študenti vytvorili. Kruh je pritom každá postupnosť študentov (dĺžky aspoň 1) $s_1, s_2, \ldots, s_k$ taká, že študent s_1 drží pravou rukou študenta s_2, s_2 drží s_3, ... až s_k drží s_1.

Príklad:

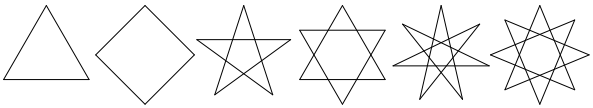
Vstup:	Výstup:	
`4`	`2`	*(Sú to kruhy 1 a 3,4,2.)*
`1 3 4 2`		

z1815. Zeofína sa vracia

Zeofína je korytnačka. Zoznámte sa. „Ahoj, ja som riešiteľ KSP.“ „Teší ma, Zeofína,“ povedala korytnačka a dodala: „Ja som v KSP stará známa. Už pred troma rokmi som sa vyskytovala v zadaniach KSP. Lenže na to sa ty, milý riešiteľ, iste nebudeš pamätať...“[2]

A Zeofína začala spomínať. Hovorila o svojich cestách svetom, o svojej mladosti, o nádherných nociach na Galapágach a – ach, tie hvezdy! Čo by za to Zeofína dala, keby ešte niekedy v živote mohla zazrieť galapágske hviezdy...

Úloha: Pomôžte Zeofíne aspoň tým, že jej napíšete program, ktorý jej pomôže hviezdy kresliť. Galapágska n-cípa hviezda ($n \geq 3$) má n cípov vo vrcholoch pravidelného n-uholníka. Z každého cípu vedú dve úsečky do „takmer protiľahlých“ cípov. Aby ste mali predstavu, ako taká hviezda vyzerá, Zeofína vám doniesla ukázať niekoľko z nich na obrázku:



$n = 3$ $n = 4$ $n = 5$ $n = 6$ $n = 7$ $n = 8$

Keď chce Zeofína kresliť, namočí si chvostík do atramentu, postaví sa doprostred veľkého čistého papiera a začne sa po ňom presúvať. Ak má chvostík spustený, zanecháva za sebou čiaru, ak ho má zdvihnutý, presúva sa len tak. Aby sa pri kreslení nepomýlila, dopredu sa naučí postupnosť povelov tvaru

- `dopredu` k – pohni sa k krokov dopredu (k je reálne),
- `dolava` u – otoč sa o u stupňov doľava,
- `doprava` u – otoč sa o u stupňov doprava,
- `zdvihni` u – zdvihni chvostík,
- `poloz` u – polož chvostík na papier.

Váš program by teda mal načítať číslo n a vypísať postupnosť povelov, pomocou ktorých bude Zeofína schopná nakresliť n-cípu galapágsku hviezdu. Pokúste sa minimalizovať celkovú dĺžku trasy, ktorú Zeofína prejde.

Príklad:

Vstup:	Výstup:

`3`	`poloz`
	`dopredu 10  dolava 120`
	`dopredu 10  dolava 120`
	`dopredu 10`

[2] Nebodaj sa pamätáš? Už vtedy si riešil/a náš seminár? A to sa nehanbíš riešiť úlohy kategórie Z? Už aj hybaj riešiť ostrú kategóriu! :-)

z1821. Zaneprázdnený knihovník

„Oook!“ povedal knihovník a zamyslene sa zahľadel na kopu kníh na jeho stole. V knižnici na Neviditeľnej univerzite sú síce uložené aj magické knihy, ale ináč je to knižnica ako každá iná. Študenti prichádzajú, požičiavajú si knihy, a čo je horšie, aj ich vracajú. Ale pretože knihovník nemá čas roznášať ich na miesta v policiach, tak ich len ukladá na dve kopy – magické vľavo, obyčajné vpravo. Občas sa niektorí študenti (vedení naivnou vierou, že najčítanejšie knihy sú najlepšie) opýtajú, aká kniha je na vrchu tej-ktorej kopy. Keď sa im zapáči, tak si ju vypýtajú a knihovník im ju podá. No a na konci dňa musí chudák knihovník knihy z kôp poroznášať do políc. Pretože by si chcel dať čo najskôr banán, pomohol by mu zoznam kníh na kopách.

Úloha: Napíšte program, ktorý bude od knihovníka dostávať príkazy, simulovať obidve kopy kníh a ktorý na konci dňa vypíše zoznam kníh na obidvoch kopách. Príkazy sú:

- „`Polož` k *názov*“ – Položí na kopu k (k je 1 alebo 2) knihu *názov*.
- „`Navrchu` k“ – Vypíše, aká kniha je na vrchu kopy k.
- „`Zober` k“ – Zoberie z kopy k najvrchnejšiu knihu.
- „`Koniec`“ – Koniec dňa. Program vypíše zoznam kníh na obidvoch kopách a skončí.

Príklad:

```
> Polož 1 Prastaré primitívne kúzla a čary
> Polož 2 Pútnikov sprievodca po Zemeploche
> Polož 1 Princípy premien
> Polož 2 Zradca v cechu vrahov
> Navrchu 1
Princípy premien
> Navrchu 2
Zradca v cechu vrahov
> Zober 2
> Navrchu 2
Pútnikov sprievodca po Zemeploche
> Polož 2 Tajomné zákutia Neviditeľnej univerzity
> Zober 1
> Zober 1
> Koniec

Na prvej kope nie je žiadna kniha
Na druhej kope sú knihy (odhora nadol):
Tajomné zákutia Neviditeľnej univerzity, Pútnikov sprievodca po Zemeploche
```

z1822. Zo zapadnutej dedinky

Za siedmimi horami, za siedmimi dolinami stála dedina. Tou dedinou sa ozýval krik. To sa Janka s Katkou hádali, ktorá z nich je krajšia. „Ja som najkrajšia z celej dediny!“ kričala Katka. „Ja som najkrajšia z celého chotára!“ vrieskala Janka. „Nie, ja!“ odkričala jej Katka a nahnevane odišla. Urazená Janka tresla dverami (aby sa nepovedalo) a začala plánovať pomstu. Rozhodla sa, že o Katke rozšíri klebetu. A že sa ju musí dozvedieť každý. Klebeta sa šíri tak, že Janka ju porozpráva všetkým ľuďom, ktorí jej dôverujú. Každý z nich ju potom oznámi všetkým tým, ktorí veria jemu a každý z nich... Janka teraz smutne sedí nad zoznamom obyvateľov chotára a premýšľa, či sa klebetu dozvedia všetci. Je na vás, aby ste jej s tým pomohli.

Úloha: Dané je n – počet obyvateľov chotára. Pre jednoduchosť budeme obyvateľov označovať celými číslami od 1 po n, pričom Janka má číslo 1. Zvyšok vstupu tvorí zoznam usporiadaných dvojíc a_i b_i, ktorých význam je nasledovný: ak klebetu už pozná a_i, dozvie sa ju od neho aj b_i.

Váš program by mal z týchto údajov zistiť, či sa časom klebetu dozvie každý alebo nie. Pozor, ak obyvateľ x verí obyvateľovi y, neznamená to nutne, že aj obyvateľ y verí obyvateľovi x!

Príklad:

Vstup:	Výstup:

`6`
`1 3  2 3  4 3  1 4  2 4`
`4 6  2 5  3 1  5 4  2 6`
`3 5  6 5  2 1  3 4`
	`Nie.`
	
	(Nik nepovie klebetu obyvateľovi 2.)

z1823. Zúfalý tesár

„Nieeee…! Koľko ešte musím búchať týmto kladivom?“ zakričal na celú Rímsku ríšu tesár Claudius. Cisár mu jednoznačne povedal, že domov pôjde, až keď budú na pamätnej tabuli všetky významné medzníky Rímskej ríše. A tých je veru neúrekom… Claudius sa už teší domov a chce vedieť, koľko úderov ešte musí spraviť.

Rimania zapisujú číslo takto: V poradí od najväčšieho po najmenší napíšu znaky, ktoré dokopy dávajú daný súčet, a to tak, že vždy použijú najväčší, ktorý môžu. Hodnoty znakov sú uvedené v tabuľke. Napríklad 1051 zapíšu ako MLI = 1000 + 50 + 1. Výnimky z pravidla o poradí sú takéto: Namiesto VIIII Rimania napíšu IX, namiesto IIII napíšu IV, namiesto LXXXX napíšu XC, namiesto XXXX napíšu XL, namiesto CCCC napíšu CD a namiesto DCCCC napíšu CM. Takže napríklad 1049 napíšu ako MXLIX = M+XL+IX = 1000+40+9. Môžete si všimnúť, že vo všetkých uvedených prípadoch môžeme uvedenie menšej cifry pred väčšou chápať ako odčítanie: napríklad XL má hodnotu L − X.

Úloha: Na každú rímsku číslicu je potrebný istý počet úderov podľa tabuľky. Zistite, koľko musí Claudius spraviť úderov, keď chce vytesať rok, ktorý je daný na vstupe v desiatkovej sústave. Môžete predpokladať, že rok je prirodzené číslo od 1 do 3999.

I=1	1 úder	V=5	2 údery	X=10	2 údery	L=50	2 údery
C=100	2 údery	D=500	3 údery	M=1000	4 údery		

Príklad:

Vstup:	Výstup:
`2001`	`9`

*(*2001 = MMI*, čo sú 4 údery na každé* M *a 1 úder na* I.*)*

z1824. Zvláštne zariadenie

Vedci Neviditeľnej univerzity už majú s vývojom rôznych zvláštnych zariadení veľké skúsenosti. Ich počítač Hex, založený na mravcoch, sa stal svetoznámym. Avšak miniaturizácia pokračuje, a tak sa rozhodli vyvinúť počítač, ktorého hlavnou súčasťou budú blchy.

Výsledkom ich doterajšieho snaženia je zariadenie pozostávajúce z pravidelného n-uholníka a n cvičených bĺch. Vrcholy n-uholníka aj blchy sú očíslované od 1 po n. V každom vrchole n-uholníka je napísaná veta naledujúceho typu: „Skoč na vrchol číslo i.“ Zariadenie má tú vlastnosť, že pre každé i z rozsahu od 1 po n existuje práve jeden vrchol, kde je napísané: „Skoč na vrchol číslo i.“, takže sa nikdy žiadne dve blchy nestretnú.

Činnosť zariadenia je nasledovná. Na začiatku výpočtu sa každá blcha postaví do vrcholu s rovnakým číslom, ako má ona. Potom pri každom takte výpočtu (ten je určený mohutným úderom na bubon), sa každá blcha zachová podľa príkazu, ktorý je napísaný vo vrchole, v ktorom stojí.

Vedci Neviditeľnej univerzity boli svojim dielom nadšení. Po dlhom bubnovaní ich však začali trápiť pochybnosti, či všetko funguje tak, ako má. Aby skontrolovali činnosť zariadenia, potrebovali by zistiť, ako majú byť blchy rozostavené po danom počte úderov na bubon.

Úloha: Na vstupe je dané n – počet vrcholov zariadenia a n dvojíc celých čísel z rozsahu 1 až n. Dvojica čísel a, b znamená, že z vrcholu a sa má skákať na vrchol b. Môžete predpokladať, že dvojice popisujú korektné zariadenie. Ďalej je na vstupe dané k – počet úderov na bubon. Napíšte program, ktorý vypočíta, ako budú po k úderoch rozmiestnené blchy v popísanom zariadení.

Príklad:

Vstup:

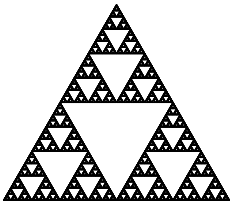
```
5
2 5
3 4
1 3
5 2
4 1
2
```

Výstup:

```
1. vrchol: blcha č. 3
2. vrchol: blcha č. 2
3. vrchol: blcha č. 4
4. vrchol: blcha č. 1
5. vrchol: blcha č. 5
```

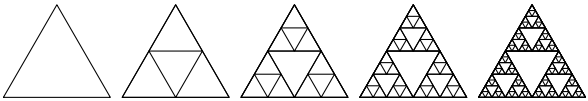
z1825. Zeofína v ríši fraktálov

Počas svojej ďalšej nostalgickej chvíľky začala Zeofína spomínať na svoj pobyt v ríši fraktálov. Fraktály sú útvary, ktoré sa skladajú z upravených kópií seba samého. Zeofíne sa zachcelo namaľovať si nejaký fraktál. Ihneď jej jeden prišiel na rozum – Sierpińského trojuholník.



Sierpińského trojuholník je rovnostranný trojuholník, do ktorého sú vložené tri ďalšie Sierpińského trojuholníky s polovičnou dĺžkou strany. Tieto trojuholníky sa dotýkajú rohmi a dve z ich strán sú časti strán pôvodného trojuholníka, tak ako na obrázku.

Zeofína sa rozhodla už-už začať kresliť, keď si uvedomila, že taký Sierpińského trojuholník je vlastne útvar tvorený nekonečným počtom trojuholníkov. A ona toľko kresliť nebude. Preto sa rozhodla nakresliť iba zjednodušenú verziu, a to Sierpińského trojuholník rádu n. Sierpińského trojuholník rádu 1 je obyčajný trojuholník. Pre $n > 1$ sa Sierpińského trojuholník rádu n skladá z troch Sierpińského trojuholníkov rádu $n-1$. Z tých kľukatých čiar ju už stihla rozboliet hlava, a preto jej pomôžete nájsť postup, ako na to.



A ako môže Zeofína kresliť? Presne rovnako ako v úlohe z1815.

Úloha: Napíšte program, ktorý načíta n – rád Sierpińského trojuholníka a vypíše postupnosť povelov pre Zeofínu, ktorými takýto trojuholník nakreslí.

Poznámka: V rámci testovania vášho programu vám odporúčame nielen vypísať postupnosť povelov, ale aj nakresliť výsledný obrázok. (Na to môžete použiť napríklad unit `graph3` v Borland Pascale.)

z1831. Zdravotné stredisko

V Nalomenej Trieske majú zdravotné stredisko. V tomto zimnom období doň prichádza veľa pacientov. Dennodenne si niekto vytkne členok alebo zlomí nohu. V škole vyhlásili

chrípkové prázdniny, lebo hrozí chrípková epidémia. Vírus BSE vyčíňa ako besný. (Z medicínskeho hľadiska síce nejde celkom o vírus, ale však čo.)

V zdravotnom stredisku nepretržite ordinuje jeden doktor. Ľudia sedia v čakárni a kričia od bolesti. Občas príde ďalší pacient. Doktor pacientov postupne vyšetruje. Keď jedného pacienta dovyšetruje, nakukne z ordinácie do čakárne a toho pacienta, ktorý najhlasnejšie reve, zavolá dnu do ordinácie.

Úloha: Vašou úlohou je spraviť simuláciu zdravotného strediska. Na začiatku je čakáreň prázdna. Váš program má postupne spracovávať príkazy:

„`pacient` *meno* h“: Do čakárne prišiel pacient, volá sa *meno* a reve hlasitosťou h.

„`doktor`“: Doktor vykukol do čakárne. Najhlasnejší pacient má ísť do ordinácie. Vypíšte jeho meno. Ak je najhlasnejších viac, tak môže ísť ľubovoľný z nich. Ak v čakárni nie je nikto, tak vypíšte vhodnú správu.

Príklad:

```
> doktor
V cakarni nie je nikto.
> pacient Jozo 10
> pacient babicka 2
> pacient Anicka 5
> doktor
Do ordinacie ide pacient Jozo.
> doktor
Do ordinacie ide pacient Anicka.
> pacient Zuza 100
> doktor
Do ordinacie ide pacient Zuza.
> doktor
Do ordinacie ide pacient babicka.
```

Poznámka: Pacientov môže byť v čakárni naraz strašne veľa, snažte sa, aby váš program vedel spracovávať jednotlivé príkazy rýchlo.

z1832. Zlodejov únik

„Z. Z. po vykradnutí banky unikol polícii. Za jeho dolapenie je vypísaná odmena 100 dolárov.“ Takéto plagáty sa jedného dňa objavili v hlavnom meste. Chudák zlodej Z. Z. si chce čo najdlhšie užívať slobodu a ukradnuté peniaze. Preto sa rozhodol, že sa uchýli do toho mesta, ktoré je od hlavného mesta najvzdialenejšie.

Úloha: Na vstupe je n – počet miest v krajine a m – počet ciest. Hlavné mesto má číslo 1. Ďalej je daných m trojíc (*mesto*$_1$, *mesto*$_2$, *dĺžka cesty*), charakterizujúcich jednotlivé cesty. Každá cesta spája dve rôzne mestá. Cesty sú obojsmerné a križujú sa mimoúrovňovo. Vašou úlohou je nájsť mesto, ktoré je od hlavného mesta najviac vzdialené, ak sa doňho ide najkratšou možnou cestou.

Príklad:

Vstup:

```
4 5
1 2 7
3 4 2
3 2 6
1 3 1
4 2 3
```

Výstup:

```
Ukry sa do mesta 2.
```

(Najkratšia cesta z hlavného mesta do mesta číslo 2 vedie cez mestá 3 a 4 a má dĺžku 6.)

z1833. Zbavme sa drobných!

Určite poznáte ten pocit. Vaša peňaženka praská vo švíkoch, ale to ešte neznamená, že v nej máte veľa peňazí. Ba priam naopak. Jedna smutne vyzerajúca pokrčená dvadsať-

koruna a hŕŕŕba drobných. Ale vyhadzovať drobné sa neoplatí. Oveľa lepšie je platiť nimi. Ale to nie je také ľahké. Iste uznáte, že keď zaplatíte desať korún desiatimi jednokorunovými mincami, spravíte si v peňaženke viac miesta, ako keď zaplatíte desaťkorunou. Nájsť najlepší spôsob zaplatenia však nie je až také jednoduché, hlavne keď tých drobných máte naozaj veľa. To je práca pre počítač. A ešte predtým pre programátorov.

Úloha: Na vstupe je číslo n, udávajúce počet mincí v peňaženke, a suma s, ktorú treba zaplatiť. Následne je pre každú mincu daná jej hodnota. Napíšte program, ktorý zistí, akého najväčšieho počtu mincí sa vieme zbaviť zaplatením sumy s. Ak nie je možné zaplatiť sumu s presne, podajte o tom správu.

Príklad:

Vstup:
```
5 100
10 25 25 50 90
```
Výstup:
```
Vieme sa zbaviť 3 mincí.
```
(Použijeme mince 25, 25, 50.)

Vstup:
```
3 25
10 20 30
```
Výstup:
```
Suma sa nedá zaplatiť.
```

z1834. Zábavný park

Tomáško sa konečne dostal do veľkého zábavného parku. Je tu n obrovských atrakcií – kolotoče, strelnice, všelijaké zábavky. Rozhodol sa, že ich určite musí vyskúšať všetky. Pri prvých dvoch ale narazil na problém. Boli to dva obrovské kolotoče a niečo mu hovorilo, že ísť na ne hneď po sebe by nebol príjemný zážitok pre jeho žalúdok. Tak sa rozhodol, že si najprv vypracuje plán. Očísloval atrakcie od 1 po n a začal si ich na papieri rôzne zoraďovať. Po chvíľke už ale nevedel, či náhodou nepíše niečo, čo už má. „To sa predsa musí dať urobiť jednoduchšie," povedal si Tomáško a zamyslel sa.

Úloha: Napíšte program, ktorý pre zadané n vypíše všetky možné preusporiadania čísel $1, 2, \ldots, n$ v lexikografickom poradí, t.j. „abecedne" zoradené.

Príklad:

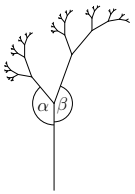
Vstup:
```
3
```
Výstup:
```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

z1835. Zeofína a zlé deti

Zeofína zase zúri. Deti opäť vystrájali v jej záhradke a podupali jej milované kvietky. Tentokrát sa už Zeofína rozhodla zakročiť. Vysadí si živý plot. To však nie je také ľahké, ak má plot aj pekne vyzerať. Teraz Zeofína behá po papieri a snaží sa nakresliť nejaký druh kríku, z ktorého bude živý plot vysadený.

Keď chce Zeofína kresliť, spomenie si na sadu príkazov z úlohy z1815. Ešte dobre, že sa nemusí učiť žiadne nové povely...

Krík vyzerá nasledovne: Prvý rok narastie kmeň kríka. Druhý rok sa kmeň rozvetví na dva konáre. Každý ďalší rok sa všetky konáre, ktoré narástli posledný rok, rozvetvia na dva konáre. Ľavý konár zviera s kmeňom (resp. s konárom, z ktorého vyrastá) uhol α a je a-krát kratší ako to, z čoho vyrastá. Pravý konár zviera s kmeňom (resp. s konárom, z ktorého vyrastá) uhol β a je b-krát kratší ako to, z čoho vyrastá.



Na obrázku vpravo je príklad 7-ročného kríka pre $\alpha = 140°$, $a = 0.4$, $\beta = 160°$ a $b = 0.6$.

Úloha: Napíšte program, ktorý načíta počet rokov, ktoré krík rastie, a jeho parametre α, a, β, b a vypíše postupnosť povelov pre Zeofínu, ktorými takýto krík nakreslí.

z1841. Z podzemného archívu

V podzemí sa našli nekonečne veľké zásoby žltej pásky. Deti hneď zobrali jeden kotúč a rozvinuli ho. Dlhočizný pás žltej pásky potom rozdelili na políčka a očíslovali ich od 1 takmer až do nekonečna.

Potom si začali na pásku kresliť a zapisovať na ňu rôzne texty. A to pokojne aj jeden znak cez druhý. Vždy, keď niekto popísal nejaký súvislý úsek žltej pásky, oznámil ostatným interval, na ktorom sa nachádza jeho práve vytvorené dielo. Na konci dňa chceli deti vedieť, koľko políčok žltej pásky vlastne popísali svojimi výtvormi, ale zistili, že to vôbec nebude jednoduché.

Úloha: Na vstupe je dané číslo n, predstavujúce počet zápisov na žltú pásku, a ďalej n intervalov, ktoré deti popísali. Interval i, j znamená, že sa zapisovalo na políčka $i, i+1, \ldots, j$. Vašou úlohou je zistiť počet políčok žltej pásky, na ktorých je niečo napísané.

Príklad:

Vstup:
```
4
2 4
3 4
3 5
7 7
```

Výstup:
```
Deti popísali 5 políčok.
```

(Popísané sú políčka s číslami 2, 3, 4, 5 a 7.)

z1842. Zbytočný Miško

Miško je agentom Atlantídskej tajnej služby. Nedávno zistil, že v Atlantíde operujú cudzí špióni. A tak by ho zaujímalo, ktorý z nich je veľmi dôležitý, ktorý menej, ktorý je úplne zbytočný, a podobne. Vyhlásil veľkú celoštátnu pátraciu akciu. Na konci (keď už všetci ostatní agenti spokojne oddychovali v blízkom rekreačnom zariadení) mal na stole obrovské množstvo lístkov s nápismi v tvare: agent XZ nosí správy agentovi AB.

Najdôležitejší špión je ten, ktorý nemusí nikomu odovzdávať správy. Ten trochu menej dôležitý nenosí správy nikomu okrem najdôležitejšieho špióna. Ľubovoľný agent je menej dôležitý ako tí, ktorým nosí správy a dôležitejší ako všetci tí, ktorí nosia správy jemu. Miško teraz chce, aby mu niekto usporiadal cudzích špiónov podľa dôležitosti.

Úloha: Na vstupe je zoznam získaných informácií v tvare XZ AB (XZ informuje AB). Vašou úlohou je usporiadať špiónov podľa dôležitosti od najdôležitejšieho (nemusí nikoho informovať) až po najmenej dôležitého (nikto mu nenosí správy). Ak je takých poradí viac, vypíšte ľubovoľné z nich. Ak také poradie neexistuje, vypíšte, že úloha je nesplniteľná.

Príklad:

Vstup:
```
Vlado Janka
Mišof Braňo
Mišof Ňaňka
Yoyo Braňo
Braňo Vlado
Ňaňka Meri
Braňo Ňaňka
Janka Riško
Riško Meri
Dávidko Meri
Janka Meri
```

Výstup:
```
Meri Ňaňka Dávidko Riško Janka
Vlado Braňo Yoyo Mišof
```

z1843. Zvláštne poznámky

Kleofáša odjakživa priťahovali veľké veci. Medzi jeho najobľúbenejšie patrí aj faktoriál. Veď vypočítať taký faktoriál nie je vôbec jednoduché. Kleofáš sa ale rád pýšil tým, že vedel zrátať aj veľmi veľké faktoriály. A aby o tom presvedčil aj všetkých naokolo, začal vo všetkých svojich poznámkach a výpočtoch používať faktoriálovú sústavu. Skôr, ako stihol dokončiť svoj najväčší vynález, ale niekam zmizol. Teraz stoja jeho nasledovníci

pred ťažkou prácou: Preštudovať jeho poznámky a trápiť sa počítaním vo faktoriálovej sústave.

Úloha: Napíšte program na sčítanie dvoch čísel vo faktoriálovej sústave. Výstupom je opäť číslo vo faktoriálovej sústave. Číslo $a_n a_{n-1} \dots a_1$ vo faktoriálovej sústave má v desiatkovej sústave hodnotu $a_n.n! + a_{n-1}.(n-1)! + \dots + a_1.1!$. Pre každé i musí cifra a_i ležať v množine $\{0, \dots, i\}$.

Príklad:

Vstup:	Výstup:
`1 2 1 1` `3 2 1`	`2 2 1 0`

(Prvé číslo má hodnotu $1.4! + 2.3! + 1.2! + 1.1! = 39$*, druhé je rovné* $3.3! + 2.2! + 1.1! = 23$*. Ich súčet je 62, čo je* $2.4! + 2.3! + 1.2! + 0.1!$*.)*

z1844. Zabudnutá pekáreň

V istej nemenovanej dedinke majú pekáreň. Nie však obyčajnú, ale takú starú, prastarú pekáreň, v ktorej sa všetko robí ručne. Napríklad aj ukladanie napečených bochníkov chleba. Tie treba uložiť do radu tak, aby na jednom konci boli tie najchrumkavejšie a na druhom tie menej podarené. Pekárov pomocník, za túto prácu zodpovedný, si už nacvičil prácu s pekárskou lopatou, na ktorú sa vedľa seba zmestia práve 3 alebo 4 bochníky chleba. Keď sa rozhodne, že treba bochníky v rade trochu preusporiadať, naberie 3 alebo 4 susedné bochníky na lopatu a obratným pohybom ich vo vzduchu prehodí tak, že sa tieto bochníky ocitnú na lopate v zmenenom poradí – najpravejší bochník sa ocitne úplne vľavo a ostatné sa len posunú o jedno miesto vpravo. Potom ich z lopaty zosunie do pôvodného radu. Ak treba, môže takýto kúsok opakovať, koľkokrát sa mu len zachce.

Úloha: Je dané číslo n udávajúce počet bochníkov v rade, za ním nasleduje pôvodné poradie bochníkov v rade. Každý bochník má priradené unikátne číslo z rozsahu 1 až n, určujúce chrumkavosť dotyčného bochníka (chrumkavejší má väčšie číslo). Úlohou je vypísať postupnosť prehadzovacích operácií, ktoré spôsobia, že bochníky budú v rade usporiadané od najchrumkavejšieho po ten najmenej chrumkavý. Prehadzovacou operáciou je jedno nabratie bochníkov na lopatu a ich preusporiadanie; každá takáto operácia sa dá popísať polohou prvého bochníka v rade, ktorý naberieme (teda 1, 2, ...) a počtom nabraných bochníkov (teda 3 alebo 4).

Pozrime sa na príklad takej prehadzovacej operácie: nech $a_1, a_2, \dots, a_i, a_{i+1}, a_{i+2}, a_{i+3}, a_{i+4}, \dots, a_n$ sú chrumkavosti bochníkov v rade za sebou, potom prehadzovacia operácia $(i, 4)$ bude ovplyvňovať bochníky na pozíciách i, $i+1$, $i+2$ a $i+3$ a výsledkom bude rad bochníkov, ktorých chrumkavosti budú $a_1, a_2, \dots, a_{i+3}, a_i, a_{i+1}, a_{i+2}, a_{i+4}, \dots, a_n$.

Príklad:

Vstup:	Výstup:

`7`
`6 7 3 4 2 1 5`
	`5 3`
	`2 4`
	`3 3`
	`1 4`

z1845. Zblúdilá Zeofína

Zeofínu pochytila túžba vytvoriť niečo veľkolepé. Preto sa rozhodla, že nakreslí taký veľký obrázok, ako nikdy predtým. A tak kreslila a kreslila... Po niekoľkých dňoch usúdila, že obrázok je už dosť veľký, a tak sa rozhodla vrátiť sa domov. No vtedy s hrôzou zistila, že zablúdila. Pamätá si síce, aký obrázok nakreslila, ale vôbec netuší, ako sa môže dostať domov. Keďže je už úplne bezradná a vyčerpaná, budete jej musieť pomôcť.

Zeofína sa už minule (v úlohe z1815) naučila 5 základných príkazov, pomocou ktorých vie kresliť a tie isté príkazy bude používať aj dnes.

Úloha: Napíšte program, ktorý načíta postupnosť povelov, ktorými Zeofína vykreslila obrázok a poradí Zeofíne, ako sa má dostať domov. To znamená, že váš program musí vypísať, ako sa má Zeofína otočiť a o koľko krokov sa má pohnúť dopredu, aby sa dostala na miesto, z ktorého kresbu začínala.

Príklad:

Vstup:
```
DOPREDU 10
DOPRAVA 90
DOPREDU 10
DOPRAVA 90
DOPREDU 30
DOLAVA 90
DOPREDU 10
```

Výstup:
```
DOLAVA 135
DOPREDU 28.2842712
```

1911. O výsledkovej listine

Aj tento rok sa naša výprava šťastne vrátila z Medzinárodnej olympiády v informatike (IOI), ba aj nejaké medaily si priviezli. No a poznáte Mariána. Nie aby si po príchode domov ľahol a spokojne spal – sadol si za počítač a začal vyrábať štatistiky. Koľkí súťažiaci mali za niektorú úlohu plný počet bodov, ako dopadli jednotlivé krajiny podľa počtov medailí, ako dopadli podľa súčtu bodov súťažiacich... Tu sa zarazil. Organizátori totiž zverejnili len body lepšej polovice súťažiacich, takže u niektorých krajín mu chýbali počty bodov niekoľkých súťažiacich. Tak spravil aspoň poradie podľa bodov, ktoré mal k dispozícii, a ku každej krajine si poznačil, koľko mu z nej ešte chýba súťažiacich. (Z každej krajiny sú na IOI najviac štyria súťažiaci.) Teraz by chcel vedieť, ako najlepšie a najhoršie môže byť každá krajina umiestnená v poradí podľa súčtu bodov všetkých súťažiacich. To je ale úloha pre počítač, a keďže Marián je na zaslúženom odpočinku, ostalo to na vás.

Úloha: Vašou úlohou je napísať program, ktorý dostane na vstupe počet krajín, ich zoznam, pre každú krajinu súčet bodov súťažiacich, ktorých body Marián vie a počet súťažiacich z nej, ktorých body nevie. Tento zoznam bude usporiadaný podľa počtu známych bodov. Okrem toho na vstupe dostanete najmenší zverejnený počet bodov (od ktorého má každý z chýbajúcich súťažiacich určite menej). Váš program by mal pre každú krajinu vypísať, ako najlepšie a ako najhoršie môže byť umiestnená v poradí podľa súčtu bodov všetkých jej súťažiacich.

Príklad:

Vstup:
```
Slovensko  1470 0
Singapur   1350 0
Rusko      1150 1
Čína        850 2
250
```

Výstup:
```
Slovensko: 1. -- 1.
Singapur:  2. -- 3.
Rusko:     2. -- 4.
Čína:      3. -- 4.
```

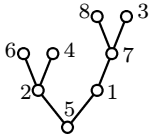
(Čína môže mať dokopy najviac 1348 bodov, preto je určite za Singapurom, a teda najlepšie tretia. Keby chýbajúci Rus získal viac ako 200 bodov, je Rusko pred Singapurom aj Čínou. A tak ďalej.)

1912. O čarovnom strome

Kde bolo, tam bolo, bolo raz jedno kráľovstvo, kde múdry Kráľovič panoval. Panoval on veru dobre, a preto sa ho ani nik nesnažil z trónu zosadiť.

Jedného dňa dopočul sa múdry Kráľovič o akomsi čarovnom strome, kdesi v susednom kráľovstve. Nemal to byť len tak obyčajný strom, teda aspoň podľa tých rečí, čo sa sem doniesli. Všade, kde sa strom vetvil, sa vetvil na 2 vetvy, ľavú a pravú, podľa toho, na ktorej strane tá ktorá nová vetva bola. Občas sa stalo aj to, že jedna z vetví sa odlomila a potom ostala len tá druhá. Každá nová vetva sa mohla ďalej vetviť, až časom narástol

celý strom. Navyše, tento čarovný strom má v každom rozvetvení napísané nejaké číslo, a taktiež má podobné čísla na listoch (list je tam, kde končí vetvička). A ešte sa povráva, že žiadne z týchto čísel nie sú rovnaké a že sú tam všetky čísla začínajúc od jednotky. Taký strom môže vyzerať napríklad nasledovne:



Nebol by to múdry Kráľovič, keby nechcel čo najskôr vedieť, ako presne ten čarovný strom vyzerá. Rozhodol sa preto svojich troch synov na prieskum do susedného kráľovstva vyslať. Aby tam však nešli s rovnakou úlohou, múdry Kráľovič riekol: *„Ty, najstarší syn môj, prinesieš popis čarovného stromu v pre-order zápise, ty, prostredný syn môj, prinesieš in-order zápis a ty, najmladší syn môj, ty dones post-order zápis čarovného stromu.“*

Pobral sa najstarší, pobral sa i prostredný syn čarovný strom navštíviť a jeho zápis získať. I najmladší sa už-už na cestu vybral, keď si to rozmyslel a povedal si, že vyčká svojich bratov a potom uvidí, čo ďalej. Vrátil sa najstarší syn z dlhej cesty a hneď od dverí otcovi hlási: *„5 2 6 4 1 7 8 3“*! Čoskoro i prostredný syn vracia sa z cesty a s podobným nadšením ohlasuje svoj zápis: *„6 2 4 5 1 8 7 3“*!

No a vtedy najmladší syn, ktorý si oba zápisy pozorne zapísal, pomaly začal hovoriť: *„6. . . 4. . . 2. . . “*

Úloha: Pomôžte najmladšiemu Kráľovičovmu synovi zo zadaného pre-order a in-order zápisu vytvoriť post-order zápis čarovného stromu.

Všetky 3 uvedené zápisy stromu sú podobné. Strom v pre-order zápise sa zapisuje tak, že začneme pri koreni (na obrázku je to 5), ten zapíšeme, a potom rekurzívne zapíšeme (rovnakým postupom) ľavý podstrom a potom pravý podstrom, teda *„koreň (ľavý podstrom) (pravý podstrom)“*. V našom prípade to bude vyzerať *„5 (2 6 4) (1 7 8 3)“*, a ak ešte vynecháme tie zátvorky, dostaneme hľadaný pre-order zápis. Podobne, v in-order zápise najprv rekurzívne zapíšeme ľavý podstrom, potom zapíšeme koreň podstromu a potom rekurzívne pravý podstrom, teda *„(ľavý podstrom) koreň (pravý podstrom)“*, v našom prípade *„6 2 4 5 1 8 7 3“*, no a v post-order zápise najpr zapíšeme ľavý podstrom, potom pravý a až nakoniec dáme koreň podstromu, teda *„(ľavý podstrom) (pravý podstrom) koreň“*, v tomto prípade dostávame *„6 4 2 8 3 7 1 5“*.

Príklad:

Vstup:

```
pre-order: 5 2 6 4 1 7 8 3
in-order:  6 2 4 5 1 8 7 3
```

Výstup:

```
post-order: 6 4 2 8 3 7 1 5
```

(ide o obrázok zo zadania)

1913. Ondrejove zápalky

Ondrej sa od mladi rád hrával so zápalkami. Staval z nich všakovaké podivuhodné umelecké dielka a rád ich rozdával svojim kamarátom. Aby dielka lepšie vyzerali, potreboval rôzne druhy zápaliek – s hnedou hlavičkou, so zelenou, celé červené a mnoho iných druhov.

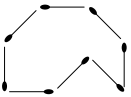
Z dlhej chvíle si z nich začal skladať na stole mnohouholníky. Na stole má 4 druhy zápaliek – modré, zelené, červené a ružové. Všetky druhy zápaliek majú rovnakú dĺžku.

Len tak si zmyslel, že modré zápalky bude dávať len v zvislom smere, zelené len vo vodorovnom, červené len zľava dole doprava hore (a opačne) a ružové len zľava hore doprava dole (a opačne). Teraz ho trápi, či možno zo všetkých zápaliek, ktoré má na stole, postaviť jeden veľký mnohouholník.

Úloha: Napíšte program, ktorý dostane na vstupe 4 čísla m, z, c, r – počty modrých, zelených, červených a ružových zápaliek a zistí, či sa dá z nich postaviť mnohouholník. Ak sa dá, vypíšte aj farby zápaliek v poradí, v akom sú uložené na jeho obvode. Ak je možností, ako poukladať zápalky, viacero, vypíšte ľubovoľnú z nich.

Príklad:

Vstup:	Výstup:
`2 2 1 0`	`Mnohouholník sa nedá postaviť.`
Vstup:	Výstup:
`2 2 2 2`	`mzčrmrzč`



(Každé písmenko výstupu je jedna zápalka, napríklad m=modrá.)

1914. O menovej reforme

Na Kiribati sa tento rok chystá veľká menová reforma. Namiesto doterajších mušličiek chce vláda zaviesť mince. Prebehla veľká reklamná kampaň pod heslom „Chceme pevnejšiu menu!“, ale len tak medzi nami: hlavný dôvod je ten, že vláda už nechce, aby si ľudia chodili zbierať peniaze ráno na pláž, zle to vplýva na pracovnú morálku.

Ale s takou menovou reformou to nie je len tak jednoduché. Minister pre reformu dobre vie, že bežní občania nemajú s mincami skúsenosti. Preto keď bude mať občan zaplatiť nejakú sumu, bude ju určite platiť tak, že vždy dá najväčšiu mincu, akú môže. Napríklad mincami s hodnotami 1, 3 a 5 by 7 platil ako 5+1+1. Nikoho ale nebude baviť vyťahovať zbytočne veľa mincí. Preto by bolo dobré, keby pri platení ľubovoľnej sumy by na to občania týmto postupom použili minimálny možný počet mincí.

Prvý vládny návrh boli mince s hodnotami 1, 4 a 5. Potom si ale minister uvedomil, že táto sada mincí nie je pre Kiribati vhodná. Občania by totiž napríklad 8 platili ako 5+1+1+1, a pritom to ide s menej mincami ako 4+4. Skôr či neskôr by si to niekto uvedomil a začali by občianske nepokoje. A to naozaj nepotrebujú.

Ministrovi podriadení začali húfne podávať nové a nové návrhy sád mincí. Chudák minister nad nimi teraz sedí a snaží sa zistiť, či je aspoň jeden z nich vhodný pre Kiribati.

Úloha: Napíšte program, ktorý dostane na vstupe počet rôznych druhov mincí k a ich hodnoty $1 = a_1 < a_2 < \ldots < a_k \leq 100$ a zistí, či je táto sada mincí vhodná pre Kiribati. Ak nie, mal by nájsť jednu sumu, pre ktorú by Kiribatčania použili zbytočne veľa mincí.

Príklad:

Vstup:	Výstup:
`1 3 6`	`Áno.`
Vstup:	Výstup:
`1 3 4`	`Nie, 6.`

(Sumu 6 by Kiribatčania platili ako 4+1+1, a pritom sa dá zaplatiť ako 3+3.)

1915. O čiernych krabičkách

V softvérovom družstve SoDr sa rozhodli rozšíriť pamäť ich jediného počítača. Preto k nemu dokúpili niekoľko veľkokapacitných pamätí typu Čierna skrinka v1.0. Pamäť typu Čierna skrinka má veľmi vysokú kapacitu, má však jednu nevýhodu – nedá sa pristupovať k dátam na ľubovoľnej adrese, ale s dátami uloženými v pamäti treba pracovať iba pomocou špeciálnych príkazov.

Čierna skrinka v1.0 používa tieto tri príkazy:

- **procedure** *Push*(*dest*, *val*); uloží do čiernej skrinky *dest* hodnotu *val*.
- **function** *Pop*(*dest*); vyberie z čiernej skrinky *dest* hodnotu, ktorá bola do nej vložená ako posledná. Túto hodnotu funkcia zmaže z pamäte a vráti ako návratovú hodnotu.
- **function** *Empty*(*dest*); vráti *true*, ak je čierna skrinka *dest* prázdna, inak vráti *false*.

Každý z týchto príkazov sa vykoná v jednotkovom čase.

Príklad použitia čiernej skrinky:

Príkaz:	Návratová hodnota:	Stav pamäte po vykonaní príkazu:
$Push(a, 1)$		1
$Push(a, 3)$		$1, 3$
$Empty(a)$	*false*	$1, 3$
$Pop(a)$	3	1
$Pop(a)$	1	
$Empty(a)$	*true*	

Pamäť Čierna skrinka v1.0 teda pracuje ako zásobník. V softvérovom družstve SoDr by však potrebovali pamäť, z ktorej by sa dal vyberať prvok, ktorý bol do pamäte vložený ako prvý – t.j. potrebovali by pamäť pracujúcu ako fronta.

Úloha: Napíšte program, ktorý bude emulovať pamäť pracujúcu ako fronta. Program má načítavať a spracovávať tieto príkazy:

- **procedure** *Put*(*val*); uloží do pamäte hodnotu *val*.
- **function** *Get*; vyberie z pamäte hodnotu, ktorá bola do nej vložená ako prvá. Túto hodnotu funkcia zmaže z pamäte a vráti ju ako návratovú hodnotu.
- **function** *Empty*; vráti *true*, ak je pamäť prázdna, inak vráti *false*.

Váš program môže používať konštantný počet premenných typu Čierna skrinka v1.0. Okrem nich môže používať len konštantný počet iných premenných. Snažte sa, aby váš program pracoval efektívne, teda aby príkazy vykonával čo najrýchlejšie a aby celkový počet dát uložených v premenných typu Čierna skrinka bol čo najmenší.

Príklad:

Vstup:	Výstup:
`Put(1)`	
`Put(3)`	
`Empty`	`false`
`Get`	`1`
`Get`	`3`
`Empty`	`true`

1921. Ochrana zdravia pri práci

Kontrolná Skupina Pracovníkov na svojom nedávnom zasadnutí rozhodla, že vo všetkých prevádzkach s viac ako 10 zamestnancami treba zvýšiť bezpečnostné opatrenia. Dôvodom takéhoto rozhodnutia bol vysoký počet úrazov v pracovnej dobe. Preto musí kontrolná skupina vykonať viacero testov, na základe ktorých určí optimálne bezpečnostné opatrenia.

Prvý problém sa zjavil hneď na začiatku: ako vybrať vhodných kandidátov na jednotlivé testy? Testy sú pomerne drahé, preto treba zo všetkých zamestnancov vybrať nejakých k. Ale nie akýchkoľvek – tí vybraní musia svojim vekom patriť čo najviac „do stredu“.

Úloha: Daný je počet zamestnancov n, číslo k a veky všetkých zamestnancov.

Prostredný vek (tiež nazývaný *medián* vekov) definujeme nasledovne: pre nepárne n je to vek zamestnanca, ktorý by bol presne uprostred, keby sme všetkých zamestnancov usporiadali podľa veku. Pre párne n sú uprostred poradia zamestnanci dvaja a prostredný vek definujeme ako priemer vekov ich dvoch. Vybrať treba tých k ľudí, ktorých vek je najbližší k priemernému veku. Napíšte program, ktorý to čo najefektívnejšie spraví.

Príklad:

Vstup:	Výstup:
`12 4`	`#3, #4, #5 a #9`
`29 33 31 30 31 35 35 34 32 29 25 27`	

(Prostredný vek je 31 rokov. Vybraní pracovníci majú najviac o rok iný vek.)

1922. O ťavích dostihoch

Neviem, či to viete, ale čoskoro sa budú konať veľké ťavie dostihy. Ak ste náhodou na takých ťavích dostihoch ešte neboli a neviete, ako vyzerajú, tak vám ich v krátkosti popíšem. Dostihov sa zúčastňuje n tiav. Je daná dráha, ktorú majú zabehnúť. Pred začiatkom dostihov stoja ťavy na štarte kupodivu nie vedľa seba na jednej línii, ale jedna za druhou. Podľa toho, v akom poradí stoja, sú aj očíslované. Tá, čo je najbližšie k cieľu, má číslo 1, tá za ňou 2, atď. Po odštartovaní ťavy utekajú najrýchlejšie, ako vedia, aby sa dostali čo najskôr do cieľa. Výsledky dostihov sa určia podľa toho, v akom poradí dobehli ťavy do cieľa. Tá, čo dobehla najskôr, je prvá, tá, čo dobehla po nej, je druhá, atď. Občas sa počas dostihov stane, že nejaká ťava predbehne inú. Keďže ťavy sú veľmi vytrvalé, nikdy sa nestane, že ťava predbehne ťavu, ktorá ju predtým (počas toho istého závodu) predbehla.

Ako to už býva na ťavích dostihoch zvykom, môžete uzatvárať rôzne stávky. Tento rok chce stávková spoločnosť zaviesť nový druh stávok. Budete si môcť staviť na to, že počas dostihov nastane práve k predbehnutí. Aby vedeli stanoviť kurzy takýchto stávok, potrebujú vedieť, koľkými spôsobmi môže daný počet predbehnutí nastať.

Úloha: Sú dané kladné celé čísla n a k. Určte, koľkými rôznymi spôsobmi môžu prebehnúť dostihy tak, že nastane práve k predbehnutí. Dva spôsoby považujeme za rovnaké, ak vedú k rovnakému výslednému usporiadaniu tiav.

Príklad:

Vstup:	Výstup:	
`3 2`	`2`	*(Sú to 2 3 1 a 3 1 2.)*

1923. O Martowovi II

Po siedmich ročníkoch a jednom kole prišiel mimozemšťan Martow opäť len tak mimochodom na Fakultu matematiky, fyziky a informatiky a doniesol študentom plán mestskej hromadnej dopravy d-rozmerného Manhattanu.

Plán bol veľmi jednoduchý, boli na ňom nakreslené len malé modré bodky. Ako im Martow vysvetlil, tie bodky sú autobusové zastávky. Medzi každými dvoma zastávkami chodí jeden autobus. Lenže tie autobusy nechodia len tak. V každom okamihu sa autobus hýbe len v jednom z d základných smerov. Napríklad pre tri rozmery chodia autobusy len hore a dole, vľavo a vpravo a dopredu a dozadu. Ulice v Manhattane totiž vedú len týmito smermi. Samozrejme autobus môže kedykoľvek odbočiť a pobrať sa iným povoleným smerom. Martow by chcel vedieť, ktorý autobus má najdlhšiu trasu. Pomôžte mu!

Úloha: Je daný počet rozmerov mapy d a počet zastávok n. Ďalej má každá zastávka daných d svojich súradníc $x = [x_1, x_2, \ldots, x_d]$. Dĺžka trasy zo zastávky so súradnicami $x = [x_1, x_2, \ldots, x_d]$ na zastávku so súradnicami $y = [y_1, y_2, \ldots, y_d]$ je $|x_1 - y_1| + |x_2 - y_2| + \cdots + |x_d - y_d|$. Nájdite dvojicu zastávok, ktorých vzdialenosť je najväčšia.

Poznámka: Predpokladajte, že počet rozmerov je rozumne malý (napríklad sedem), zatiaľ čo počet zastávok môže byť veľmi veľký (aj stotisíc).

Príklad:

Vstup:	Výstup:

`4 5`
`1 2 3 4`
`6 7 4 5`
`9 0 -1 3`
`3 5 8 4`
`4 11 0 0`
	`Najďalej od seba sú:`
	`[9,0,-1,3] a [3,5,8,4]`

1924. O podnikovej sieti

V istom nemenovanom veľkom podniku majú veľa počítačov. A majú s nimi veľké problémy. Veď posúďte sami: ledva stihli naučiť sekretárky behať s batohmi diskiet od

počítača k počítaču, keď tu zrazu vedenie podniku rozhodlo, že dajú spraviť počítačovú sieť. Dali kúpiť do každého počítača niekoľko sieťových kariet, kúpili kopu káblov a... zostali sa na ne pozerať, nevediac, čo s nimi. Potom niekoho napadla spásonosná myšlienka – veď elektrikár Braňo by sa predsa mal vyznať do počítačov! Tak ho zavoláme, nech nám spraví sieť. Mal by pospájať počítače tak, aby sa po sieti dalo komunikovať medzi ľubovoľnými dvoma počítačmi.

Keď Braňo prišiel, hneď videl, že to s tou sieťou nebude také jednoduché. Na všetkých počítačoch v podniku totiž beží operačný systém WOK-ná XP (eXtrémne Padavý – padne, akonáhle sa mu niečo znepáči). A ten okrem iného vyžaduje nasledovné podmienky:

– Z každej sieťovej karty v počítači musí vychádzať práve 1 kábel.
– Žiaden počítač nesmie byť spojený sám so sebou.
– Každé dva počítače môžu byť priamo prepojené najviac 1 káblom.

V opačnom prípade WOK-ná XP okamžite padnú. A navyše vedenie podniku samozrejme nechce počuť ani slovo o tom, že by sa z nejakého počítača mala nejaká sieťová karta vybrať. Veď za ne dali toľké peniaze!

A tak Braňo dlho smutne sedel nad hromadou káblov a premýšľal. Teraz (o tri dni neskôr) je vyhladnutý, vysmädnutý a presvedčený, že sa to nedá. A rád by to vedeniu dokázal, lebo ináč ho vyhodia.

Úloha: Na vstupe je počet počítačov n a pre každý z nich počet sieťových kariet v ňom. (Inými slovami, číslo hovoriace s koľkými inými počítačmi ho Braňo má spojiť.) Napíšte program, ktorý zistí, či sa dajú počítače poprepájať káblami tak, aby operačný systém nepadol a dalo sa po sieti komunikovať medzi každými dvoma počítačmi. Káblov má Braňo dosť.

Príklad:

Vstup:
```
5
3 3 2 1 1
```
Výstup:
```
Áno
```
(Spojí napríklad dvojice 1–2, 1–3, 1–4, 2–3, 2–5.)

Vstup:
```
5
6 2 2 2 2
```
Výstup:
```
Nie
```
(Počítač 1 by mal byť spojený so 6 inými, to sa ale zjavne nedá.)

Vstup:
```
4
1 1 1 1
```
Výstup:
```
Nie
```
(Nech ich spojíme akokoľvek, súvislú sieť nespravíme.)

1925. O čiernych krabičkách II

Po nákupe veľkokapacitných pamätí typu Čierna skrinka v1.0 panovala niekoľko dní v softvérovom družstve SoDr spokojnosť. Čoskoro sa však na trhu objavila vylepšená verzia Čiernej skrinky - Čierna skrinka v2.0.

Čierna skrinka v2.0 používa tieto tri príkazy:

– **procedure** *Insert*(*dest*, *val*); uloží do čiernej skrinky *dest* hodnotu *val*.
– **function** *ExtractMin*(*dest*); nájde v čiernej skrinke *dest* najmenšiu hodnotu, ktorá sa v nej nachádza. Túto hodnotu z nej odstráni a vráti ju ako návratovú hodnotu.
– **function** *Empty*(*dest*); vráti *true*, ak je čierna skrinka *dest* prázdna, inak vráti *false*.

Príkaz *Empty* sa vykoná v jednotkovom čase, všetky ostatné príkazy sa uskutočňujú v čase $O(\log n)$, kde n je počet hodnôt uložených v pamäti.

Príklad použitia pamäte:

Príkaz:	Návratová hodnota:	Stav pamäte po vykonaní príkazu:
$Insert(a, 1)$		1
$Insert(a, 5)$		$1, 5$
$Insert(a, 3)$		$1, 3, 5$
$Empty(a)$	*false*	$1, 3, 5$
$ExtractMin(a)$	1	$3, 5$
$ExtractMin(a)$	3	5
$ExtractMin(a)$	5	
$Empty(a)$	*true*	

Pamäť Čierna skrinka v2.0 teda pracuje ako halda.

V softvérovom družstve SoDr dostali výhodnú ponuku na upgrade pamäte Čierna skrinka z verzie v1.0 na verziu v2.0. Ponuka bola taká výhodná, že ju nemohli odmietnuť a všetky pamäte upgradovali. Potrebovali by však pamäť, z ktorej by sa dala vyberať nie len najmenšia, ale aj najväčšia hodnota, ktorá sa v nej nachádza. Budú preto potrebovať vašu pomoc.

Úloha: Napíšte program, ktorý bude emulovať pamäť, z ktorej možno vyberať najmenšiu aj najväčšiu hodnotu (teda pamäť, ktorá pracuje ako min-max halda). Program má načítavať a spracovávať tieto príkazy:

- **procedure** *Insert*(*val*); uloží do pamäte hodnotu *val*.
- **function** *ExtractMin*; vyberie z pamäte najmenšiu hodnotu, ktorá sa v nej nachádza. Túto hodnotu funkcia zmaže z pamäte a vráti ju ako návratovú hodnotu.
- **function** *ExtractMax*; vyberie z pamäte najväčšiu hodnotu, ktorá sa v nej nachádza. Túto hodnotu funkcia zmaže z pamäte a vráti ju ako návratovú hodnotu.
- **function** *Empty*; vráti *true*, ak je pamäť prázdna, inak vráti *false*.

Váš program môže používať konštantný počet premenných typu Čierna skrinka v2.0. Okrem nich môže používať len konštantný počet iných premenných. Snažte sa, aby váš program pracoval efektívne, teda aby príkazy vykonával čo najrýchlejšie a aby celkový počet dát uložených v premenných typu Čierna skrinka bol čo najmenší.

Príklad:

Vstup:	Výstup:
`Put(1)`	
`Put(3)`	
`Put(5)`	
`Empty`	`false`
`ExtractMax`	`5`
`ExtractMin`	`1`
`ExtractMin`	`3`
`Empty`	`true`

1931. O Kleofášovych pozemkoch

Výskumník Kleofáš konečne dosiahol veľký úspech. Na kiribatskom ostrove Dlhý Had objavil niekoľko nálezísk vzácnych fosílií trilobajtov. Aby ho o jeho majetok nik nepripravil, rozhodol sa, že pozemky, na ktorých náleziská ležia, kúpi. Nie je to však také jednoduché – narazil totiž na odpor kiribatských byrokratov.

Ostrov Dlhý Had má tvar obdĺžnika, ktorého šírka je zanedbateľne malá. Polohu náleziska preto môžeme reprezentovať jediným číslom, ktoré určuje jeho vzdialenosť od začiatku ostrova. Súvislý pozemok na ostrove môžeme reprezentovať uzavretým intervalom. Cena pozemku je priamo úmerná jeho dĺžke. Aby kiribatskí byrokrati nemali priveľa práce, vydali nariadenie, že jeden človek smie vlastniť najviac k súvislých pozemkov. Kleofáš teda

rozmýšľa, akých najviac k súvislých pozemkov má kúpiť, aby vlastnil všetky náleziská a pritom zaplatil čo najmenšiu sumu.

Úloha: Napíšte program, ktorý načíta číslo n – počet nálezísk a k – koľko súvislých pozemkov môže Kleofáš kúpiť. Ďalej načíta n čísel, ktoré určujú polohy nálezísk. Polohy nálezísk sú zadané v usporiadanom poradí. Program má poradiť Kleofášovi, ktoré pozemky má kúpiť. To znamená, že má vypísať najviac k súvislých intervalov takých, aby každé nálezisko ležalo v nejakom z nich a pritom aby bol súčet ich dĺžok čo najmenší.

Príklad:

Vstup:
```
6 3
1 2 6 42 47 100
```

Výstup:
```
<1,6>   <42,47>   <100,100>
```

1932. O bájnom poklade

„To je nuda," prehlásil Indiana Jones, následne sa uhol dvom guľkám a naďalej utekal uličkami pred skupinou štyroch zabijakov. Za najbližším rohom zobral plechový kryt náhodou voľne pohodený na zemi a bežiac v podstate dozadu odrazil ďalšie štyri výstrely, ktoré mu smerovali na hrudník. Potom odhodil plech, ten po trojnásobnej pravotočivej rotácii udrel prvého prenasledovateľa do tváre, následne stratil rovnováhu a spadol na zem, pričom potkol aj kumpána bežiaceho za ním. Ďalších dvoch banditov sa Indy tiež poľahky zbavil a o necelú hodinu už sedel doma v kresle.

Poznatky o stratenom poklade, ktoré pri tejto akcii získal, zapísal do svojho (v poradí už štyridsiateho siedmeho) denníka. Ako áno, ako nie, o niekoľko rokov sa tento denník dostal do rúk jeho syna. Keď si z neho Junior po večeroch čítal, zaujala ho najmä nasledujúca veta: „Posledná chodba vedúca k pokladu je uzavretá zatiaľ neznámym mechanizmom, ktorý dvere otvorí len vtedy, ak na stôl stojaci v ich blízkosti osoba múdra, šľachetná, atď., položí farebné kamene v správnom poradí." Oných farebných kameňov je vraj v sieni s dverami hojne, jediný problém je vybrať tie správne farby a potom ich správne zoradiť. Starý Jones samozrejme zistil, aká postupnosť kameňov otvára dvere, ale z obáv, aby ju niekto nezneužil, ju do denníka radšej iba „približne" popísal, pre istotu však dvakrát. Tu sa však Junior zarazil. Keby popisom v denníku vyhovovala iba jediná postupnosť kameňov, všetko by bolo v pohode, ale čo keď je takých viac? Preto sa obrátil na vás, aby ste mu s týmto problémom pomohli.

Úloha: Je dané číslo f udávajúce počet farieb kameňov. Farby kameňov sú označené číslicami od 1 po f. Ďalej sú dané dva popisy tej istej postupnosti kameňov.

Popisy sa skladajú z číslic a malých písmen. Číslica v popise hovorí, že na tom mieste je kameň príslušnej farby. Písmená predstavujú neznáme úseky kameňov. Rôznym výskytom toho istého písmena vždy zodpovedá ten istý úsek kameňov – teda aj dĺžka, aj farby kameňov sú presne rovnaké. Pre každé písmeno poznáme dĺžku úseku, ktorý predstavuje, nepoznáme však farby kameňov, ktoré ho tvoria.

Váš program má vypísať, koľko rôznych postupností kameňov vyhovuje zároveň obidvom popisom.

Príklad:

Vstup:
```
2
1abc1
b2cc
|a| = 2, |b| = 1, |c| = 3
```

Výstup:
```
1
```
(Obidvom popisom vyhovuje jediné postupnosť `12111111`*, pričom* `a=21`*,* `b=1` *a* `c=111`*.)*

1933. O vláčikovej súprave

V tomto roku ešte Vianoce neboli (veď ešte ani marec nevládne kraju), ale minulý rok boli také, aké majú byť. Ako ostatné deti, aj Miško dostal veľa darčekov. Ale žiaden z nich

nebol taký krásny ako elektrický vláčik, ktorý dostal pred deviatimi rokmi. Pamätá si to veľmi dobre, akoby to bolo dnes. Hneď si upratal svoju detskú izbu (čomu sa jeho mama veľmi potešila), aby mohol na dlážke postaviť okružnú železničnú trasu a na nej obrovskú stanicu. Stanica bola ozajstným skvostom celej trasy. Síce do nej vlak mohol prísť iba po jedinej koľaji, tá sa však okamžite rozvetvila na mnoho ďalších navzájom rovnobežných koľají, medzi ktorými sa kde-tu mihla výhybka. Tesne pred opustením priestoru stanice sa koľaje opäť zbehli do jedinej, po ktorej potom mohol vlak odísť.

Miško sa na svoje dielo nevedel vynadívať. Možno aj preto si nevšimol svojho mladšieho brata, ako nerozvážne položil na koľajnice lokomotívu, za ktorú zapojil šesť ďalších vagónov. Čo však bolo oveľa horšie, pripojil zdroj napätia a v tom momente sa celá súprava pohla. A keďže bola na okružnej trase, nemala veľmi na výber, a tak sa každou sekundou viac blížila ku stanici. V tej chvíli Miško precitol a spanikáril. Problém bol totiž v tom, že v stanici neboli správne nastavené výhybky a hrozilo vykoľajenie celej súpravy. A čo keď sa nejako poškodí? To bude prúser jak mraky a navyše sa bude jeho mama hnevať. Rýchlo bolo treba výhybky napraviť! Ale keďže času bolo málo, musel ich prehodiť čo najmenej. V tom čase to chudák musel zvládnuť sám, čo sa mu nakoniec aj podarilo, no teraz by určite požiadal o pomoc vás. Schválne, vedeli by ste mu pomôcť?

Úloha: Zľava doprava vedie n dlhých rovnobežných koľají očíslovaných zaradom od 1 po n. Medzi nimi prechádza spolu k výhybiek, tiež očíslovaných od 1 po k. Každá výhybka vedie medzi dvoma susednými koľajami. Výhybky sú na vstupe usporiadané podľa x-ovej súradnice ich začiatku; žiadne dve ju nemajú rovnakú. Stanicu tvorí úsek, ktorý na x-ovej osi začína začiatkom výhybky 1 a končí koncom výhybky k. Vlak vchádza do stanice po koľaji p. Potrebujeme, aby z nej odišiel po koľaji q, inak sa vykoľají.

Výhybka vedúca z koľaje a na koľaj b (kde $b = a \pm 1$) sa môže nachádzať v dvoch stavoch: vypnutá alebo zapnutá. Ak je vypnutá, pri prechode cez ňu sa nič nedeje. Ak je zapnutá, vlak prichádzajúci po koľaji a prejde na koľaj b, po ktorej pokračuje ďalej. Miesto, kde sa vlak pripojí na koľaj b, leží v smere osi x vo vzdialenosti 1 za miestom, kde výhybka začala. (Ani v jednom prípade táto výhybka nemá vplyv na vlaky idúce po koľaji b.)

Na vstupe sú najprv dané prirodzené čísla n, k, p a q. Ďalej nasleduje popis výhybiek: i-ta výhybka je popísaná číslami x_i, a_i, b_i, s_i, čo znamená, že výhybka vychádza z koľaje a_i na súradnici x_i a prichádza na koľaj b_i na súradnici $x_i + 1$. Ak $s_i = 0$, táto výhybka je momentálne vypnutá, ak $s_i = 1$, tak je zapnutá.

Váš program má vypísať minimálny počet výhybiek, ktoré treba prehodiť, aby vlak mohol prejsť stanicou. Ak sa to nedá, podajte o tom vhodnú správu.

Príklad:

Vstup:

```
4 10 3 2
1.0 3 2 0
2.0 3 4 1
3.0 2 1 1
4.0 2 3 1
5.0 4 3 1
6.5 3 2 0
9.0 2 1 1
9.5 4 3 1
11.0 3 2 0
13.0 1 2 1
```

Výstup:

```
1
```

(Jedno riešenie: prehodiť výhybku 6.
Miškov vlak pôjde nasledovne:
– prichádza po koľaji 3
– prejde vypnutou výhybkou 1
– výhybka 2 ho pošle na koľaj 4
– výhybka 5 ho pošle na koľaj 3
– výhybka 6 ho pošle na koľaj 2
– výhybka 7 ho pošle na koľaj 1
– výhybka 10 ho pošle na koľaj 2)

1934. O farebných trojuholníkoch

Janka sedí v škole už šiestu hodinu. Hádam už aj vyhladla. Ale učitelia do nej len hustia a hustia. Už ju to nebaví, a tak si z nudy začala kresliť. Najprv nakreslila na papier

n malých bodiek. Aby bodkám nebolo smutno, tak ich očíslovala číslami od 1 po n. Potom zobrala červenú a modrú farbičku a začala ich spájať. Každé dve malé bodky spojila buď červenou alebo modrou úsečkou.

Miško sa pozrel na Jankin obrázok. Napadlo mu, že by mohol spočítať, koľko je na ňom trojuholníkov, ktoré majú strany rovnakej farby: či už všetky červené alebo všetky modré. No stále mu to nevychádza a už mu akosi dochádza trpezlivosť. Pomôžte mu!

Úloha: Dané je číslo n a $n(n-1)/2$ trojíc čísel x, y, f, kde x a y sú čísla dvoch rôznych bodiek a f je farba úsečky, ktorou sú spojené („`Č`“ alebo „`M`“). Každá dvojica bodiek je daná práve raz. Napíšte program, ktorý zistí, koľko je tam jednofarebných trojuholníkov.

Príklad:

Vstup:

```
5
1 2 Č     1 3 Č
1 4 M     1 5 Č
2 3 M     2 4 M
2 5 Č     3 4 M
3 5 Č     4 5 M
```

Výstup:

```
3
```

(Sú to 125, 135 a 234.)

1935. O čiernych krabičkách III

Neviem, či vám niečo hovorí názov IBM. Väčšine z vás asi nie. Len tým pár dobre informovaným zasvietili očičká a s posvätnou úctou zašepkali: „Institute of Black Magic“. A tí, čo vedia, vám o ňom aj čo-to povedia, keď sa ich opýtate. Napríklad by ste sa mohli dozvedieť, že v Inštitúte čiernej mágie (ďalej len IBM) majú celé oddelenie, kde za použitia čiernej mágie vyrábajú čierne krabičky. Napríklad aj tie, s ktorými ste sa mohli stretnúť v prvých dvoch kolách.

IBM dostali objednávku na nový typ čiernej krabičky a mali by ho mať hotový do 11. marca. Za použitia čiernej mágie by takú krabičku hravo zostrojili, lenže... Lenže si nik nevie ani len predstaviť, ako by mala vyzerať.

Klientom IBM je tentokrát výskumníčka Janka. Pre tých, čo ju nepoznáte – Janka je biela myška, žijúca v klietke v jednom laboratóriu. Skúma a cvičí si ľudí okolo seba, medzi jej hlavné úspechy patrí, že ich dokázala vycvičiť, aby jej pravidelne dávali syr.

Jej požiadavky nie sú zďaleka zanedbateľné. Niektorí ľudia, ktorých Janka skúma, sa priatelia, iní nie. Občas sa niektorí spriatelia alebo pohádajú. No a Janka to popri všetkých výskumoch jednoducho nezvláda všetko držať v hlave, tak si na to objednala čiernu krabičku.

Pamäť typu Čierna skrinka v3.0 by mala podporovať nasledovné príkazy:

- **procedure** *Init*(n); vyprázdni pamäť čiernej skrinky a nastaví si počet pamätaných ľudí na n. Pre jednoduchosť ich má Janka očíslovaných číslami od 1 do n.
- **procedure** *AddFriends*(x, y); zaznamená, že ľudia x a y sú odteraz priatelia. (Ak sa už priatelili, nič sa nedeje.)
- **procedure** *DeleteFriends*(x, y); zaznamená, že ľudia x a y odteraz nie sú priatelia. (Ak sa ani predtým nepriatelili, nič sa nedeje.)
- **procedure** *AreFriends*(x, y); vypíše „`ano`“, ak sa ľudia x a y práve priatelia a „`nie`“ inak.
- **procedure** *EnumFriends*(x); vypíše (v ľubovoľnom poradí) čísla všetkých priateľov človeka x.

Úloha: Napíšte program, ktorý sa bude správať ako táto čierna krabička, teda bude čítať vstup od užívateľa a bude obsahovať vyššie uvedené procedúry. Formát vstupu si môžete navrhnúť tak, aby sa vám ľahko čítal.

Samozrejme je nutné, aby volanie každej z procedúr trvalo čo najkratšie. (Napríklad čas behu procedúry *EnumFriends* by nemusel závisieť od počtu všetkých ľudí, iba od počtu priateľov daného človeka.) Pamätajte, že čím lepšiu krabičku navrhnete, tým efektívnejšiu pamäť z nej potom v IBM vyvinú a tým vám bude Janka vďačnejšia.

PRÍKLAD:

VSTUP:	VÝSTUP:
`Init(3)`	
`EnumFriends(1)`	`nikto`
`AddFriends(1,2)`	
`AddFriends(1,3)`	
`EnumFriends(1)`	`2 3`
`DeleteFriends(2,3)`	
`DeleteFriends(1,3)`	
`AreFriends(1,3)`	`nie`
`EnumFriends(2)`	`1`

1941. Ohromný hazard

Bezdomovec Fero je starý hazardér. Aj toť nedávno sa mu podarilo vyhrať v rulete zopár žetónov. No nie aby bol rád, že túto noc už konečne nemusí spať pod mostom, on sa radšej rozhodol skúsiť ešte šťastie pri hracích automatoch.

Neviem, či ste to už postrehli, ale nedávno zaviedli do kasín nové hracie automaty. No a Fero zamieril rovno k nim. Fungujú veľmi jednoducho. Ak do nich vhodíte žetón s hodnotou x korún, tak z automatu vypadne x korún. Ak vhodíte ďalší žetón s hodnotu y korún, tak vypadne z automatu y korún. Navyše, ak predchádzajúci vhodený žetón bol menšej hodnoty ako y, tak vypadne ďalších $y-x$ korún. Podobne to funguje aj so všetkými ďalšími žetónmi.

Pochopiteľne, Fero má ambície dotiahnuť to spod mosta pokiaľ možno čo najvyššie, preto by rád vyhral čo najviac.

ÚLOHA: Pomôžte Ferovi! Fero má na začiatku n žetónov s hodnotami $z_1, z_2, \ldots, z_n$ korún. Napíšte program, ktorý načíta tieto čísla a následne nájde a vypíše také poradie vhadzovania žetónov, aby Ferova celková výhra bola maximálna. Ak takých poradí existuje viacero, vypíšte ľubovoľné z nich.

PRÍKLAD:

VSTUP:	VÝSTUP:
`5`	`2 8 3 1 7`
`7 8 1 3 2`	

(Pri tomto postupe Fero získa 2 + 14 + 3 + 1 + 13 = 33 korún.)

1942. O zaľúbenom princovi

Kde bolo, tam bolo, za siedmimi horami a siedmimi dolami, bolo raz jedno kráľovstvo. Teda, boli tam dve kráľovstvá. Vlastne, no, povedzme, že tam bolo n kráľovstiev, medzi ktorými viedli nejaké jednosmerné cesty.

V jednom z kráľovstiev žil princ, a keďže bol práve mier a princ nemohol ísť bojovať, neostalo mu nič iné ako sa zaľúbiť do princeznej. A keď sa už do nej zaľúbil, chcel ju požiadať o ruku. Lenže princezná býva v inom kráľovstve a zlý čarodej, ktorý chcel princeznú získať pre seba, zaklial každú cestu. Zakliatím získala každá cesta jedno celé číslo ktoré udáva, o koľko človek zostarne, keď po nej prejde. (Pozor, toto číslo môže byť aj záporné!) Princovi je jedno, ako dlho bude trvať jeho cesta za princeznou, ale chcel by za ňou prísť čo najmladší, aby ho chcela. Zistite, aký najmladší môže prísť princ pred princeznú, ak má teraz 20 rokov.

Pri riešení môžete využiť ešte jednu dôležitú skutočnosť: Ani mágia zlého čarodeja nedokáže vyčarovať večnú mladosť, preto v celej krajine určite neexistuje okruh, po ktorom keby princ prešiel dookola (t.j. vrátil sa na miesto, odkiaľ po ňom začal ísť), tak by omladol.

Poznámka: Môže sa stať, že princ bude mať v priebehu alebo aj na konci cesty záporný alebo naopak veľmi veľký vek. Keďže toto je rozprávka, nijak mu to neprekáža, dôležitý je len jeho vek na konci.

Úloha: V prvom riadku vstupu máte zadané čísla n a m, kde n je počet kráľovstiev a m je počet ciest vedúcich medzi kráľovstvami. Kráľovstvá sú označené číslami $1, 2, \ldots, n$. Kráľovstvo číslo 1 je domovom princa, v kráľovstve s číslom 2 býva princezná. V každom z nasledujúcich m riadkov vstupu je trojica čísel popisujúca jednu cestu. Prvé číslo trojice určuje, z ktorého kráľovstva cesta vychádza, druhé číslo je číslo kráľovstva, do ktorého cesta vedie a tretie číslo udáva, o koľko rokov princ zostarne, keď po ceste prejde. Váš program má vypísať najnižší možný vek princa v okamihu, keď príde za princeznou. Ak sa za princeznou nedá prísť, program vyjadrí princovi svoj súcit.

Príklad:

Vstup:	Výstup:
`4 6`	`Princ bude mať po prejdení cesty 17 rokov.`
`1 4 -4`	
`4 3 10`	*(Optimálna cesta vedie cez kráľovstvo 3.*
`3 1 -2`	*Pri inej ceste, napríklad 1-4-2 alebo 1-4-3-1-3-2,*
`1 3 2`	*bude princ pri príchode do kráľovstva 2 starší.)*
`4 2 2`	
`3 2 -5`	

1943. O pobodkovanej kružnici

Nebolo to tak dávno, keď si Santo robil úlohu z geometrie. Santo a Banto totiž momentálne hľadajú na Klondike zlato. Hladať zlato sa však dá iba cez deň, lebo v noci je v bani tma. Ono tma tam je aj cez deň, dokonca možno ešte väčšia ako v noci, ale v noci je tma aj mimo bane a Santo a Banto sa boja ísť domov potme. Preto po večeroch nemajú čo robiť. Banto je lenivý od narodenia, jemu to až tak nevadí. On sa zabaví aj v krčme. Ale Santo je viac uvedomelý a štvalo by ho, keby štvrtinu svojho života (to je čas, kedy je vonku tma a človek nespí) presedel pri barovom pulte. Preto sa prihlásil na diaľkové štúdium a po večeroch rieši úlohy.

A sme naspäť pri tej geometrii. Kto hádal, iste uhádol, že išlo o jednu z úloh na diaľkové štúdium. „Hm, hm, hm, ako by sa len dala vpísať kružnica do štvorca... Hm, hm... A dá sa to vôbec?“ Tieto a podobné myšlienky behali Santovi hlavou asi štyri hodiny, potom sa zrazu objavil nápad spojiť protiľahlé vrcholy a vzápätí sa úloha vyriešila skoro sama. Od toľkej námahy skoro zaspal na stoličke, no nakoniec sa mu podarilo zvaliť sa na posteľ a zaspať tam.

Niekedy vtedy sa vrátil Banto z krčmy a celý veselý si pozrel Santov porysovaný zošit. Banto okrem iného aj rád kreslí, a preto na Santovej tak ťažko vpísanej kružnici vyznačil niekoľko bodov, ktoré mu veľmi pripomínali hviezdičky mihajúce sa mu každý večer pred očami. Iste si viete predstaviť Santa, ako mu mizne úsmev z tváre, keď ráno objaví Banta zvaleného cez stoličku a svoju kružničku úplne pobodkovanú. Banto sa onedlho prebral, no nebol ešte úplne fit. Začal mať blbé otázky typu „Čo je to trojuholník?“, „Ako vyzerá rovnostranný trojuholník?“, „A ako pravouhlý?“, a podobne. Hneď na to uvidel svoje hviezdičky v Santovom zošite, čo ho inšpirovalo k otázkam: „A je tu nejaký pravouhlý trojuholník? A rovnostranný by sa nenašiel?“. Teraz mal Santo taký pocit, že by vraždil a plakal zároveň. Prečo by vraždil, iste všetci tušíte. No a plakal by preto, lebo ani na jedinú otázku nevedel súvisle odpovedať. Preto by chcel poprosiť vás, či by ste mu aspoň s poslednými dvoma nepomohli.

Úloha: Na vstupe je dané číslo n nasledované n dvojicami čísel. Každá dvojica predstavuje súradnice bodu na kružnici s polomerom 1 a stredom v bode $[0, 0]$. Body sú na vstupe dané v poradí, ako sa nachádzajú na kružnici. To znamená, že keby sme prezerali

kružnicu po obvode proti smeru hodinových ručičiek, pričom by sme začali v bode $[1,0]$, natrafili by sme na jednotlivé body presne v takom poradí, ako sú na vstupe. Vašou úlohou je zistiť, či medzi bodmi existuje trojica bodov, ktoré tvoria vrcholy

a) pravouhlého trojuholníka

b) rovnostranného trojuholníka.

Príklad:

Vstup:	Výstup:

`4`
`1.0  0.0`
`0.5  0.866025404`
`-1.0  0.0`
`0.5  -0.866025404`
	`Možno nájsť pravouhlý trojuholník.`
	`Možno nájsť rovnostranný trojuholník.`
	
	(Rovnostranný trojuholník tvoria body 1,2,4, pravouhlé trojuholníky sú 1,2,3 a 1,3,4.)

1944. O výskumníčkach

Po tom, čo sa výskumníčke Janke podarilo dostať z laboratória, rozhodla sa, že pôjde skúmať niečo zaujímavejšie. Zavolala si kamarátku (tiež sa volá Janka) a vybrali sa spolu na severný pól skúmať polárneho medveďa Richarda. Cestou rozmýšľali, čo by tak na ňom vlastne mohli skúmať. Po tom, ako zavrhli sociálne správanie (s tým už mali skúsenosti z minulosti), rozhodli sa pre genetiku. Zistia štruktúru jeho DNA.

Na severnom póle ale objavili hneď najväčšiu prekážku – zimu. Aby sa stále netriasli, vždy si Janky v prenosnom laboratóriu „nadýchali“, až im bolo teplúčko. Ale beda! Občas sa im zarosil aj display ich elektrónkového mikroskopu, a tak nemohli odčítať niektoré úseky DNA. Našťastie ale, ako správne výskumníčky, viedla každá z nich vlastný výskum a možno by spojením ich výsledkov predsa len mohli zistiť Richardovu DNA.

Úloha: Na vstupe sú dva reťazce zo znakov `A`, `T`, `C`, `G` a `*`: výsledky pozorovaní oboch výskumníčiek. Hviezdička zodpovedá neprečítanému miestu – ľubovoľne dlhému reťazcu báz (aj prázdnemu). Vašou úlohou je napísať program, ktorý zistí najkratšiu postupnosť vyhovujúcu obidvom pozorovaniam.

Príklad:

Vstup:	Výstup:
`*CGAT*CCA*G*` `AT*ATACC*AG*`	`ATCGATACCAG`

1945. O čiernych krabičkách IV

V kráľovstve pána Kráľa žije veľa podivných tvorov, ale najviac starostí majú s rozmaznanou princeznou Milou. Včera si opäť zmyslela, že niečo chce a chce a chce. Chce sa na koči previesť cez všetky mestá otcovho kráľovstva, cez žiadne nie dvakrát a vrátiť sa späť na zámok. Radcovia dumajú, ale vydumať nevedia. Keby aspoň vedeli, či taká trasa existuje... vybrali sa teda pre pomoc za dobrou vílou Amálkou.

„Teraz sa vám nemôžem venovať,“ odvetila dobrá víla milým hlasom, lebo práve pracovala na nebezpečnom pokuse s profesorom Indigom. „Zoberte si však túto čiernu krabičku. Dokáže v konštantnom čase odpovedať, či v danom kráľovstve existuje cesta prechádzajúca polovicou miest.“

Úloha: Na vstupe je počet miest n v kráľovstve. Ďalej nasleduje popis všetkých ciest, dvojíc čísel (a_i, b_i). Každá dvojica hovorí, že medzi mestami a_i a b_i vedie cesta.

Máte k dispozícii čiernu skrinku, ktorej vždy keď dáte na vstup popis *ľubovoľného* kráľovstva s m mestami, v konštantnom čase vám odpovie, či sa v popísanom kráľovstve nachádza cesta prechádzajúca cez $\lfloor m/2 \rfloor$ miest, na ktorej sa žiadne mesto neopakuje. Formát popisu kráľovstva si môžete zvoliť.

Príklad práce krabičky:

Vstup:	Výstup:

`miest: 8`
`cesty: 1 2, 1 3, 1 4, 1 5,`
`1 6, 1 7, 1 8`
	`NIE`

Vstup:	Výstup:

`miest: 8`
`cesty: 1 2, 1 3, 1 4, 1 5,`
`6 5, 6 8, 7 8`
	`ANO`
	(Napr. cesta 3-1-5-6.)

Vašou úlohou je zistiť, či v kráľovstve, ktoré dostanete na vstupe, existuje alebo neexistuje okružná cesta, ktorá začína a končí v meste 1 a prechádza práve raz každým z ostatných miest. Váš program má pracovať čo najrýchlejšie, pretože kráľovstvo je veľké.

Príklad:

Vstup:	Výstup:
`miest: 4` `cesty: 1 2, 1 4, 2 3, 2 4, 3 4`	`ÁNO` *(Napr. okruh 1-2-3-4-1.)*
Vstup:	Výstup:

`miest: 5`
`cesty: 1 2, 1 3, 2 3, 3 4,`
`3 5, 4 5`
	`NIE`

z1911. Zlatá retiazka

V temných dobách stredoveku žil v Anglicku jeden rytier a ten sa volal Ivanhoe. Bol to veru smelý a rúči, no trochu skúpy mládenec. Ivanhoe mal milú a tá sa volala Anička. Bola že to rúča deva a ľúbila Ivanhoa veľmi. Aby sa on odvďačil svojej milej za toľkú lásku, rozhodol sa, že jej zadováži zlatú retiazku. Ako sa rozhodol, tak i urobil. Najbližší poklad strážil miestny drak. I vybral sa preto za drakom a bez váhania mu odťal všetkých dvanásť hláv. Poklad aj so zlatou retiazkou bol jeho.

Ako sa však ukázalo, zlatá retiazka bola zamotaná. Už to vlastne nebola ani reťaz, ale skôr kopa zlatých očiek náhodne poprepájaných medzi sebou. Ivanhoovi sa podarilo zistiť, že retiazka má dve koncové očká, ktoré pôvodne slúžili na to, aby sa dala zapnúť okolo krku. Ivanhoe je skúpy a rád by z reťaze odpílil čo najviac zlatých očiek. Na retiazke ale musia ostať koncové očká, a aby to bola ešte vôbec retiazka, musia ostať prepojené.

Úloha: Na vstupe je dané n – počet očiek. Očká sú očíslované číslami od 1 po n, pričom 1 a 2 sú čísla koncových očiek. Ďalej je dané m – počet prepojení očiek. Nasleduje zoznam dvojíc očiek, ktoré sú spojené medzi sebou. Napíšte program, ktorý vypíše čísla očiek, ktoré má Ivanhoe odpíliť tak, aby ich odpílil maximálny možný počet, a aby na retiazke ostali dve prepojené koncové očká. Ak je možností viacero, vypíšte ľubovoľnú z nich.

Príklad:

Vstup:	Výstup:

`6`
`6`
`1 4 4 2 2 3`
`3 5 5 6 1 5`
	`Môže odpíliť očká 3, 5, 6.`
	(Zostane mu reťaz 1-4-2.)

z1912. Z trpasličej chalúpky

V Microlande, krajine s najvyspelejšou miniaturizáciou, sa rozhodla skupina trpaslíkov postaviť si chalúpku. Ich najväčší problém však spočíva v tom, ako dopraviť stavebný materiál do potrebnej výšky. Microlandské miniaturizované žeriavy to nedokážu, preto si trpaslíci musia poradiť sami.

Trpaslíci sa preto rozhodli, že časť z nich sa postaví v pravidelných rozostupoch do radu od najväčšieho po najmenšieho. Potom si na hlavu položia latu, po ktorej ostatní trpaslíci vytlačia fúrik s tehlami. Len čo však tento nápad zrealizovali, nastala medzi nimi zvada. Jeden druhého začali obviňovať, že sa ulievajú, teda že nepridŕžajú hlavou latu. Aby mohli svoj spor spravodlivo rozhodnúť, budú potrebovať vašu pomoc.

Úloha: Napíšte program, ktorý načíta n – počet trpaslíkov v rade a n čísel – výšky trpaslíkov. Výšky trpaslíkov môžu byť na vstupe zadané v ľubovoľnom poradí. Úlohou programu je vypísať, či všetci trpaslíci budú pridŕžať hlavou latu, teda či sa hlavy trpaslíkov budú nachádzať na jednej priamke za predpokladu, že sa trpaslíci postavia v pravidelných rozostupoch usporiadaní podľa výšky.

Príklad:

Vstup:	Výstup:
`5` `12 6 4 8 10`	`ÁNO`
Vstup:	Výstup:
`4` `6 4 3 1`	`NIE`

z1913. Z tajného laboratória RSA

„Počuj Ron, a bude tá tvoja nová šifrovacia metóda dosť *blbuvzdorná*? Vieš predsa veľmi dobre, že ľudia si vždy zvolia ten najslabší možný kľúč," spýtavo sa na Rivesta pozrel jeho dlhoročný známy Shamir. „Veruže bude! Žiadny kľúč – žiadni hlupáci. Len ešte musím doriešiť jeden malý detail," s úsmevom odvetil Rivest.

Aby ste rozumeli, Ron Rivest vynašiel novú šifrovaciu metódu, ktorá si kľúč vytvorí sama. Užívateľ zadá správu, ktorú šifrovací program premení na kladné celé číslo n. Teraz prichádza najdôležitejšia časť – vytvorenie kľúča k (taktiež celé kladné číslo). Toto je jediný detail, ktorý ešte Ron nedoriešil: ako vybrať najsilnejší (čiže najbezpečnejší) kľúč. Sila kľúča sa určí tak, že správu (číslo n) prepíšeme do sústavy so základom k a zrátame, koľkými nulami tento zápis končí. Samozrejme, čím viac, tým lepšie. Napríklad pre správu $n = 72$ má kľúč $k = 3$ silu 2, lebo $72 = 2200_3$. Zvyšok šifrovania je už prísne tajný, a preto sa o ňom už viac nemôžete dozvedieť. Napriek tomu však skúste pomôcť chudákovi Ronovi, ktorý je už z toľkého premýšľania úplne zronený.

Úloha: Pre dané n nájdite čo najsilnejší kľúč k. Ak je viac rôznych najsilnejších kľúčov, vypíšte ľubovoľný z nich.

Príklad:

Vstup:	Výstup:	
`198`	`Najlepší kľúč je 3.`	(198 *je v trojkovej sústave* 21100)

z1914. Zhrdzavené potrubie

„Zase ďalšia diera!" v duchu si povedal Zápla, dvorný opravár Zimbabwejského kráľovstva. Za dlhé roky svojej služby videl už mnoho hrdzavých potrubí. Zvyčajne bola diera len jedna, a preto nebol problém, čo s ňou – jednoducho ju Zápla zaplátal záplatou. Lenže dnes ráno ho z ničoho nič zavolali k veľkej havárii na hlavnom vodovodnom potrubí, ktoré dopravuje vodu z jazera Kariba ($16° 53′$ S, $28° 01′$ E) do hlavného mesta, Harare ($17° 50′$ S, $31° 03′$ E). Z asi dvadsiatich miest potrubia striekala voda cez prehrdzavené steny hrubej rúry. Záplaty na také veľké potrubie sú drahé, a preto si Zápla musí dobre rozmyslieť, ako ich na diery poukladať.

Úloha: Záplaty sú už z továrne narezané na metrové kúsky a nedajú sa už na mieste rezať na menšie. Diery na potrubí sú zanedbateľne malé, ak teda položíme záplatu presne jej krajom na dieru, bude táto diera zaplátaná. Záplat môžeme dať aj viac na seba.

Na vstupe je počet dier n a ich polohy na potrubí. Keďže sú veľmi malé, budeme ich považovať za body na priamke. Ich poloha bude udaná vzdialenosťou v metroch od začiatku potrubia. Môžete predpokladať, že pozície všetkých dier sú na vstupe usporiadané podľa tejto vzdialenosti.

Zápla je už mierne nervózny, voda stále strieka a on vás žiada o radu, ako má záplaty na potrubie dávať, aby ich musel použiť čo najmenej.

Príklad:

Vstup:	Výstup:
`3`	`záplata od 0.5`
`1 3 3.5`	`záplata od 2.75`

(Situácia zo vstupu a zodpovedajúci výstup vyzerajú nasledovne:  *)*

z1915. Zluvyy n Arebqr pvgnwh onfra

GNENQHE

Cenmar wr; uyn, fyvmbcehmxr wnmiegxl
mbgenqvrear xbybqhwh cb mngeniv.
Irpugbtnwr pyvivn an gvr iliegxl,
cenfbganpxl ilfgvn, myhopvn - pb gb fceniv...

Qnw cbmbe an Gnenqhen, flah zbw,
puena fn wrub uelmbyhfgv, miynfg xrq mhezv,
nw an ignxn Xeivynxn cevceni moebw,
Ghcve arpu gn arebmpuingar qencnmhezv!

Fla fn zrpbz ibecnybilz bcnfny,
qyub uynqry i qvnybomber arcevngryn.
Bqcbpviny cbq ohxhobz, aruyb fgny,
mnuhgnal cerfynfgniny, uhqan maryn.

Mypbqehol cbznyl hm bqvfg pupry,
igbz Gnenqhe ohear uheab melpny xqrfv;
flpny, shpny, menxl i cynzbpu, imqhpu fn puiry,
uany fn x arzh prm ghytbir pvrear yrfl.

Gny qb arub, qb mvirub qb gryn,
ibecnybih prcry oehfah moebfvy xeibh;
nm xrq zegin uynin fgehcar myrgryn,
gelfxbz-ilfxbz qbzbi pvryvy prfgbh ceibh.

Gl fv mqbyvy Gnenqhen, flah zbw,
cbq, arpu fv gn cevghavrz an irger xbfgv!
Yrcbelfr! Fynipva! Uheenw! Ubwnubw!
puvpubqnxny i oynuar, wnfpny bq enqbfgv.

Cenmar wr; uyn, fyvmbcehmxr wniegxl
mbgenqvrear xbybqhwh cb mngeniv.
Irpugbtnwr pyvivn an gvr iliegxl,
cenfbganpxl ilfgvn, myhopvn - pb gb fceniv...

WNOOREJBPXL

'Gjnf oevyyvt, naq gur fyvgul gbirf
Qvq tler naq tvzoyr va gur jnor;
Nyy zvzfl jrer gur obebtbirf,
Naq gur zbzr enguf bhgtenor.

'Orjner gur Wnoorejbpx, zl fba!
Gur wnjf gung ovgr, gur pynjf gung pngpu!
Orjner gur Whwho oveq, naq fuha
Gur sehzvbhf Onaqrefangpu!'

Ur gbbx uvf ibecny fjbeq va unaq:
Ybat gvzr gur znakbzr sbr ur fbhtug-
Fb erfgrq ur ol gur Ghzghz gerr,
Naq fgbbq njuvyr va gubhtug.

Naq nf va hssvfu gubhtug ur fgbbq,
Gur Wnoorejbpx, jvgu rlrf bs synzr,
Pnzr juvssyvat guebhtu gur ghytrl jbbq,
Naq oheoyrq nf vg pnzr!

Bar, gjb! Bar, gjb! Naq guebhtu naq guebhtu
Gur ibecny oynqr jrag favpxre-fanpx!
Ur yrsg vg qrnq, naq jvgu vgf urnq
Ur jrag tnyhzcuvat onpx.

'Naq unf gubh fynva gur Wnoorejbpx?
Pbzr gb zl nezf, zl ornzvfu obl!
B senowbhf qnl! Pnyybu! Pnyynl!'
Ur pubegyrq va uvf wbl.

'Gjnf oevyyvt, naq gur fyvgul gbirf
Qvq tler naq tvzoyr va gur jnor;
Nyy zvzfl jrer gur obebtbirf,
Naq gur zbzr enguf bhgtenor.

Nxb evrfravr fgnpv cbfyng bon anqcvfl onfav. N obq anilfr qbfgnargr, nx mvfgvgr, bqxvny wr gn onfra, pb pvgnwh.

z1921. Zľabdaní zlatokopi

Na Vyšnej Klondike sa začalo stavať nové mesto – New Očová. Urbanizačnú komisiu práve čaká ťažká úloha: musí rozhodnúť, ktoré stavebné parcely budú obývať zlatokopi a na ktorých sa postavia krčmy. Rozhodnúť však nemôže hocijako. Každý zlatokop chce mať krčmu nablízku. Preto parcela, na ktorej býva zlatokop, musí susediť s nejakou parcelou, na ktorej je krčma. Naopak, nikto by nechcel vlastniť krčmu, pri ktorej nik nebýva. Každá krčma teda musí susediť s nejakou parcelou, na ktorej býva zlatopkop. Pomôžte urbanizačnej komisii a napíšte program, ktorý túto ťažkú ulohu vyrieši namiesto nich.

Úloha: Na vstupe je n – počet parciel a m – počet dvojíc susediacich parciel. Parcely majú čísla 1 až n. Na vstupe je ďalej m dvojíc čísel parciel, ktoré spolu susedia.

Napíšte program, ktorý z týchto údajov vypočíta, na ktorých parcelách postaviť krčmy a na ktorých ubytovať zlatokopov. Stačí nájsť jedno prípustné riešenie.

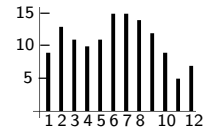
Príklad:

Vstup:
```
8
11
1 2    3 5    1 6    4 5
1 7    6 7    2 3    6 8
2 4    7 8    3 4
```

Výstup:
```
Krčmy: 1 2 5 8
Zlatokopi: 3 4 6 7
```

z1922. Z trpaslíčich pretekov

Kde bolo, tam bolo, bolo raz jedno kráľovstvo. Bolo to konvexné kráľovstvo a žili v ňom trpaslíci. Každoročne tu prebiehala súťaž o najkonvexnejšo-konkávnejší rad trpaslíkov. Konkurencia je veľká, takmer všetky trpasličie rodinky chcú vyhrať, alebo sa aspoň zúčastniť. Koľkože problémov to však spôsobí odbornej kráľovskej porote, ktorá musí každý rad trpaslíkov pozorne preskúmať a určiť správnu konvexnosť a konkávnosť tohto radu...



Úloha: Kráľovská porota vás preto požiadala o pomoc: napíšte jej program, ktorý na vstupe dostane počet trpaslíkov, ktorí sú rovnomerne rozostavení v rade, a ich výšky a určí konvexnosť a konkávnosť tohto radu.

Konvexnosť radu je dĺžka najdlhšieho konvexného úseku trpaslíkov; úsek trpaslíkov (i, j) s výškami $a_i, a_{i+1}, \ldots, a_{j-1}, a_j$ nazývame *konvexný*, ak pre každých dvoch trpaslíkov z tohto úseku platí, že ak na hlavy týchto dvoch trpaslíkov položíme dosku, budú hlavy trpaslíkov medzi nimi nižšie ako doska. *Konkávnosť radu* definujeme podobne, pričom úsek trpaslíkov bude *konkávny*, ak všetci trpaslíci medzi vybranými dvoma budú vyšší ako doska. (Keď nejaký úsek radu nie je konvexný, neznamená to ešte, že musí byť konkávny!)

Príklad:

Vstup:
```
12
9 13 11 10 11 15 15 14 12 9 5 7
```

Výstup:
```
Konvexnosť: 5
Konkávnosť: 7
```

(Najdlhší konvexný úsek tvoria trpaslíci 2 až 6, konkávny zas trpaslíci 5 až 11.)

z1923. Záhada Jožkovej kalkulačky

Jožko sa práve vrátil domov zo školy a chce sa ísť hrať von. Ale tak, ako všetky školopovinné deti, musí si aj on najprv napísať domácu úlohu. Zobral teda zošit z matematiky a začal čítať, čo sa vlastne učili. Zistil, že dnes sa v škole učili, ako sa počíta $n!_3$.

Pre tých, čo sa to ešte neučili: $0!_3 = 1$, $1!_3 = 1$, $2!_3 = 2$ a pre $n \geq 3$ platí, že $n!_3 = n \cdot (n-3)!_3$. Je to teda niečo ako faktoriál, ale násobíme len každé tretie číslo.

No a o tomto čude má Jožko domácu úlohu. Na takéto zákerné príklady má našťastie perfektnú kalkulačku s mnohými funkciami. Jednou z jej schopností je počítať veľmi presne aj s veľkými číslami. Začal teda počítať. Ale čoskoro s hrôzou zistil, že kalkulačka sa mu pokazila. Z akýchsi neznámych príčin okrem ľavostranných bezvýznamných núl vynecháva aj pravostranné významné nuly (napríklad namiesto `1430000` zobrazí len `143`). Našťastie si všimol, že má na stole ešte aj počítač, ktorý síce s veľkými číslami počítať nevie, ale dá sa na to naprogramovať. To sa však ešte v škole neučili, preto potrebuje vašu pomoc. Keďže kalkulačka mu ešte ako-tak funguje, stačí mu, ak zistíte, koľko núl musí dopísať k výsledku, ktorý mu vypíše kalkulačka.

Úloha: Je dané prirodzené číslo n. Zistite, koľkými nulami sa končí zápis čísla $n!_3$ (v desiatkovej sústave).

PRÍKLAD:

VSTUP:	VÝSTUP:	
47	3	*(47!$_3$ = 262134882788466688000)*

z1924. Zimbabwejské obchodné centrum

Najväčšia zimbabwejská spoločnosť Zima s. r. o. (s. r. o. = sneh radšej obmedzíme) sa rozhodla postaviť v Harare Zimbabwejské obchodné centrum. Architektúru budovy mal na starosti preslávený zimbabwejský architekt Mbwana. Jeho dovtedy najslávnejšia stavba bola metrová veža z kociek, ktorú postavil ako jedenásťročný. Teraz sa však rozhodol, že vykročí zo svojho tieňa a postaví stavbu ešte mohutnejšiu. Zachoval si však štýl – nová budova bude mať výšku h metrov a bude sa skladať z k kociek s celočíselnými dĺžkami hrán.

Aby však bola stavba bezpečná (čiže aby bola čo najmenšia šanca, že ju trafí lietadlo), musí mať čo najmenší objem. Tento problém Mbwana vyriešil šikovne – zadal ho preslávenému zimbabwejskému matematikovi Bwangovi. Bwang sa už preslávil napríklad tým, že spočítal všetky tri cesty, vedúce do Harare, ale táto úloha sa zdá byť nad jeho sily. Preto sa rozhodol požiadať vás o pomoc.

ÚLOHA: Napíšte program, ktorý načíta výšku budovy h a počet kociek k ($k \leq h$) a vypíše k celých čísel: dĺžky hrán k kociek takých, že keď tieto kocky postavíme na seba, dostaneme vežu výšky h s minimálnym možným objemom.

PRÍKLAD:

VSTUP:	VÝSTUP:
6 4	2 1 2 1

(Objem tejto veže je 18 metrov kubických.)

z1925. Ziege Otto

Otto Ziege bol počas prvej svetovej vojny anglickým špiónom v Nemecku. V tých dobách bola ešte kryptografia v plienkach a neexistovali ešte žiadne šifry, tak ako ich poznáme dnes.

Raz sa mu podarilo zachytiť tajnú správu (pozri tabuľku vľavo).

```
e s b m t o k y i n
v ú a s a u l ť s ú
b p e j e w c h e s k
e d z e e n e a m č
ď k ň z i a l n s z
a n o ý i p m p o a
v ľ ý o k m n a o l
e d h n e a r a l r
i o s l o á o p z o
v y i m d á p b r s
```

Vedel, že správa musí obsahovať tieto slová: `keď`, `úspech`, `podarilo`, `zimbabwe`. Navyše vedel, že správa je šifrovaná nasledovným systémom:

Zoberieme štvorčekový papier a vystrihneme z neho štvorec 10×10 štvorčekov. Vhodnú štvrtinu (t.j. 25) štvorčekov vystrihneme. Vznikne nám šifrovacia mriežka. Položíme ju na papier a vpisujeme zľava doprava, zhora nadol do vystrihnutých okienok text, ktorý chceme zašifrovať, pričom vpisujeme do každého okienka práve jedno písmenko. Medzery a interpunkčné znamienka v texte sa preskakujú. Potom mriežku otočíme o 90 stupňov proti smeru hodinových ručičiek a postup opakujeme. Po tom, ako štvrtýkrát vyplníme všetky voľné políčka, dostaname vyplnený štvorec písmen 10×10. Samozrejme, že treba vedieť, ktoré štvočeky vystrihnúť, aby sa nám nestalo, že by sme do nejakého okienka nemohli zapísať písmenko.

Ukážte, že nie ste o nič horší ako Otto, a dešifrujte túto správu!

z1931. Zanzibarské jednosmerky

Zanzibar je veľké a prastaré mesto. Začali ho stavať ešte pred našim letopočtom.

Začiatkom letopočtu však naši prapredkovia nemali ani kúsok citu a pochopenia pre modernú urbanizáciu, automobilovú dopravu, potreby dnešného moderného človeka a vôbec. Zaoberali sa hlavne tým, ako na čo najmenšiu plochu nadžgať čo najviac chatrčí. Žiaľ,

týmto vývojom sa stalo, že uličky v Zanzibare sú priveľmi úzke pre obojsmernú automobilovú dopravu.

Začiatkom 21. storočia sa problém vyostril natoľko, že mestská rada bola nútená narýchlo zaviesť v meste systém jednosmeriek. Zbastlená reforma však, zdá sa, všetko iba zhoršila. Zapeklitým problémom totiž ostáva, či sa v meste dá dostať autom z ľubovoľného miesta na ľubovoľné iné tak, aby sa dodržal predpísaný smer jazdy. Zistite to!

Úloha: Zanzibar si možno predstaviť ako systém jednosmeriek, ktoré vedú medzi križovatkami. Zadané je n – počet križovatiek (tie sú očíslované od 1 do n) a m – počet jednosmeriek. Ďalej je daných m jednosmeriek. Každá jednosmerka je popísaná dvomi číslami, udávajúcimi, z ktorej križovatky na ktorú sa po nej smie jazdiť.

Napíšte program, ktorý zistí, či sa možno medzi dvoma ľubovoľnými miestami v meste dostať autom tak, aby sme dodržali dopravné predpisy.

Príklad:

Vstup:	Výstup:

`6`
`7`
`1 2  2 3  3 1  4 5`
`5 6  6 4  1 4`
	`Nie.`
	
	(Nevieme sa dostať napríklad z križovatky 4 na križovatku 1.)

z1932. Zábudliví trpaslíci

„A čo si nám to chcel povedať?“ pýtali sa (už trochu nahnevaní, pretože sa to pýtajú už siedmy krát) trpaslíci chudáka Zedrika. No nestávalo sa to len jemu: niekedy, keď si dohodli stretnutie pod jedným zo stromčekov, si zrazu nemohli podaktorí spomenúť, čo chceli ostatným povedať. Jednoducho trpasličia skleróza. Nebola to choroba nebezpečná, ale tiež nie veľmi príjemná. Každý trpaslík mal doma na chladničke lístok, kam si písal všetky dôležité veci, aby na ne nezabudol. No keď išiel na stretnutie pod stromček, chladničku si so sebou pochopiteľne nebral, a potom sa mohlo stať, že kým prišiel k stromčeku, zabudol, čo vlastne chcel.

Preto sa rada starších rozhodla, že s tým treba niečo spraviť. Ideálne zjavne bude, ak sa budú stretávať pod takým stromčekom, aby všetci trpaslíci spolu prešli čo najmenšiu vzdialenosť – vtedy toho dokopy najmenej stihnú zabudnúť. Trpasličia dedinka má len jednu ulicu, na ktorej sú domčeky trpaslíkov. Medzi každými dvoma domčekmi rastie stromček, pod ktorým sa dá stretávať. Polohy domčekov, ako aj stromčekov, sú celé čísla, ktoré určujú vzdialenosť daného objektu od začiatku ulice.

Úloha: Pomôžte rade starších a napíšte im program, ktorý im povie, kde sa majú trpaslíci stretnúť, aby toho spolu čo najmenej prešli a čo najmenej zabudli. Vstupom programu je počet domčekov n (z každého pôjde na stretnutie jeden trpaslík), usporiadaný zoznam pozícii domčekov a usporiadaný zoznam pozícii $n-1$ stromčekov – jeden medzi každou dvojicou susedných domčekov. Program má za úlohu zistiť, pri ktorom zo stromčekov sa majú trpaslíci stretnúť.

Príklad:

Vstup:	Výstup:

`5`
`domčeky: 0 3 10 12 16`
`stromčeky: 1 6 11 13`
	`stromček na pozícii 11`

z1933. Zapamätaj si PIN

Móricko je starý sklerotik. Nech sa snaží, ako sa snaží, stále si nevie zapamätať PIN kód ku platobnej karte. Aj by si ho niekde napísal, ale je mu jasné, že to sa nemá robiť, lebo to nie je bezpečné. Jedného dňa však dostal spásonosný nápad.

Predstavte si, že postupne za seba napíšeme štvorce všetkých prirodzených čísel. Dostaneme tak nekonečný reťazec začínajúci nasledovne: 1491625364964811001211441 69...

Móricko si pre každú cifru PIN kódu zapísal jednu z pozícií, na ktorých sa v tomto reťazci dotyčná cifra nachádza.

Teraz však má problém. Ako isto tušíte, opäť zabudol svoj PIN kód. A čo je horšie, zabudol aj algoritmus, ktorým ho bude vedieť znova vypočítať. Pomôžte mu!

Úloha: Na vstupe je číslo k. Vašou úlohou je nájsť čo najefektívnejší algoritmus, ktorý nájde cifru na k-tom mieste vyššie popísaného reťazca.

Príklad:

Vstup:	Výstup:
`13`	`4`

z1934. Záhady Kiribatského rybolovu

Kiribatská vláda sa rozhodla zvýšiť príjmy štátu zavedením povolení na rybolov. Povolenia sa udeľujú nasledovne: Mapa Kiribati (ako isto viete) je rozdelená na $r \times s$ štvorcov. Pre potreby udeľovania povolení je každý z týchto štvorcov prehlásený buď za pevninu, alebo za more. Za 10 kiribatských bubákov môže ľubovoľný občan požiadať o povolenie na rybolov. Do žiadosti musí uviesť časť mapy, pre ktorú má toto povolenie platiť. Táto časť musí byť obdĺžnikového (resp. štvorcového) tvaru a musí byť tvorená práve niekoľkými štvorcami mapy. Každá takáto oblasť je teda jednoznačne určená súradnicami štvorcov mapy v dvoch jej protiľahlých rohoch. Povolenie je občanovi udelené práve vtedy, keď oblasť, ktorú vyplnil do žiadosti, obsahuje iba more.

Väčšina Kiribatčanov rada loví ryby. Na druhej strane, málokto z nich sa vyzná v kartografii. Takže väčšina občanov si vyzdvihne na úrade prázdnu žiadosť, vyplní do nej nejaké náhodné čísla a zájde s ňou späť na úrad. Pokiaľ dostane povolenie, ide loviť ryby, pokiaľ nie, vyzdvihne si ďalšiu. A štát veselo zarába. Tak si to aspoň všetci vo vláde predstavujú.

Jediné, čo chýba, je program, ktorý by *veľmi rýchlo* vedel rozhodovať o žiadostiach, či ju treba prijať alebo zamietnuť. Na úradoch sa totiž čakajú také návaly, že nebude v silách úradníkov zvládnuť ich bez pomoci výpočtovej techniky.

Úloha: Prežite sa spolu s kiribatskou vládou na udeľovaní povolení na rybolov a napíšte program, ktorý bude prijímať a zamietať žiadosti podľa vyššie uvedeného popisu. Program dostane vo vstupnom súbore mapu Kiribati v nasledovnom formáte: Prvý riadok obsahuje dve čísla r, s – počet riadkov a stĺpcov mapy. Ďalšie riadky obsahujú samotnú mapu, pričom j-ty znak v i-tom riadku je `1`, ak je vo štvorci $[i, j]$ pevnina, resp. `0`, ak je tam more.

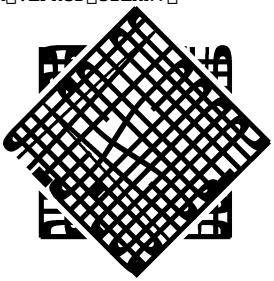
Po spustení má váš program načítať mapu, prípadne si niečo dopredu pripraviť a potom v nekonečnom cykle čítať zo vstupu žiadosti. Každý riadok bude obsahovať 4 čísla – súradnice dvoch protiľahlých rohov oblasti, o ktorú občan žiada. Program má vypísať **`prideliť`** alebo **`zamietnuť`** podľa toho, či je v tejto oblasti len more alebo aj pevnina. Je dôležité, aby váš program vedel o ľubovoľnej žiadosti rozhodnúť čo najrýchlejšie.

Príklad:

Vstup:		Výstup:
Mapa:	Žiadosti:	**`prideliť`**
`8 10`	`1 1 2 3`	**`zamietnuť`**
`0000000000`	`3 3 3 3`	**`zamietnuť`**
`0000000010`	`8 10 1 1`	**`prideliť`**
`0011000110`	`1 7 4 6`	
`0111100000`		
`0000110000`		
`0000001000`		
`0111000000`		
`0000000000`		

z1935. Zablúdená správa

```
ERPAATNIC␣␣PNOAJEEASJTSIOE␣DVRENERM␣EPDMR␣TEK,TOSOY␣ZME␣AILUTZ␣NVR␣UZDIAAEAN
␣ISTJIBITLY␣␣NEDCIOVOHDRUU␣HESMOA␣PIR␣NSK,TO␣␣YCVHEKPOVYKSO␣JRREAKCUNEE␣DMIA
VJ␣TTEO␣␣CELIINYISNAT␣AT␣CSTKIOMVOSURBE␣␣KSEYTCHHUZICJICOD␣JIENC␣OV␣HTTOSHDR
UU,␣L␣A␣RUEF␣AEKNTTAEKATUA␣SA␣LMCYELDCDEA␣NITSAET␣REE␣A␣USSP␣MCH␣OZALCAKHYKO
M␣VEPKOD␣OIEKN.␣
```



„Táto správa musela určite zablúdiť," povedalo si Ministerstvo Špionáže po tom, ako sa ich najlepším odborníkom na poli kryptografie túto správu nepodarilo rozlúštiť. „Jediné, čím sme si istí, je, že znak ‚␣' označuje medzeru. A že čo ten obrázok pribalený k správe? To si najskôr nejaké dieťa krátilo svoj voľný čas a nám vyrábalo zbytočné starosti!"

Úloha: Pokúste sa Ministerstvu pomôcť rozlúštením tejto správy a opísaním šifrovacej metódy.

z1941. Zúrivý Ubu

Slniečko začalo svietiť. V Burundi nastalo krásne jarné ráno. Kráľ Ubu sa zobudil, ale to ráno sa mu asi nezdalo krásne. Možno ho trápila trauma z detstva, alebo sa len zle vyspal. Tak či onak, začal zúriť. Jeho zúrenie nemalo konca kraja. Najprv sa naštval na sluhu, ktorý mu doniesol raňajky, potom sa naštval na kráľovských radcov a nakoniec sa naštval na celý ľud Burundi. Tak sa rozhodol, že život ľuďom vo svojej krajine znepríjemní trochou byrokracie.

Rozhodol sa, že rozdelí mestá v krajine do niekoľkých okresov. Aby však otrávil ľud čo najviac, rozhodol sa rozdeliť ich do okresov tak, aby sa zo žiadneho mesta nedalo dostať do iného mesta v tom istom okrese (ani cez mestá v iných okresoch). Teraz ho trápi, koľko najmenej okresov musí zriadiť.

Úloha: V kráľovstve Burundi je n miest očíslovaných od 1 po n, medzi ktorými vedú cesty. Na vstupe je n – počet miest v Burundi, ďalej m – počet ciest vedúcich medzi mestami. Nasleduje m dvojíc miest x_i, y_i, medzi ktorými vedie priama cesta. Napíšte program, ktorý načíta cestnú sieť Burundi a vypíše najmenší počet okresov, ktoré Ubu musí zriadiť.

Príklad:

Vstup:	Výstup:
`5`	`Treba zriadiť 3 okresy.`
`4`	
`1 2`	*(Jedno možné rozdelenie na okresy:*
`2 3`	*Prvý okres obsahuje mestá 1 a 4,*
`3 1`	*druhý okres mestá 2 a 5,*
`4 5`	*tretí okres tvorí jediné mesto 3.)*

z1942. Závislák Jurko

Jurko je vášnivý hazardný hráč. Keď je práve v Trpasličom meste, nikdy neodolá a zastaví sa na partičku či dve Škatuliek. To je veľmi jednoduchá hra. Trpaslík Jurkovi ukáže päť škatuliek. V každej z nich je jeden z piatich rôzne veľkých zlatých nugetov. Jurkovou úlohou je usporiadať škatuľky tak, aby nugety v nich boli usporiadané podľa veľkosti.

Jurko samozrejme netuší, v ktorej škatuľke je ako veľký nuget. Aby mal šancu úlohu splniť, môže si niekoľkokrát počas hry prstom ukázať na dve škatuľky, načo mu Trpaslík ukáže, v ktorej z nich je väčší nuget.

Ako to chodí, Trpaslík zo začiatku nechal Jurka vyhrávať, nenápadne mu radil, ako sa má pýtať, aby správne poradie nugetov rýchlo odhalil. A tak si Jurko začal veriť a vyriekol

onu osudnú vetu: „Stavme sa, že aj keby sme teraz tisíc hier odohrali, každú jednu z nich najviac na x otázok vyhrám!“ Trpaslík sa usmial od ucha k uchu a stávku prijal.

Úloha: Pre čo najmenšie x nájdite a naprogramujte stratégiu, ktorá Jurkovi zaručí, že správne poradie škatuliek nájde, pričom sa zaručene nanajvýš x-krát bude potrebovať pýtať Trpaslíka.

Príklad priebehu jednej hry:

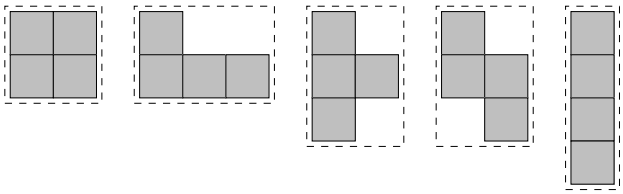
```
V ktorej škatuľke je väčší nuget: v 1. alebo v 5.?
> 1
V ktorej škatuľke je väčší nuget: v 3. alebo v 4.?
> 4
V ktorej škatuľke je väčší nuget: v 5. alebo v 4.?
> 5
V ktorej škatuľke je väčší nuget: v 4. alebo v 2.?
> 4
V ktorej škatuľke je väčší nuget: v 3. alebo v 2.?
> 2
Správne poradie škatuliek: 1, 5, 4, 2, 3.
```

(Najväčší nuget je v škatuľke 1, druhý najväčší v škatuľke 5, atď.)

z1943. Zajačiky idúúú!

Vo veľkom čudnom meste M žije spokojne skupina farmárov. A všetci do jedného pestujú mrkvičky. Každý farmár má také to svoje pole, na ktorom ich vo veľkom pestuje. Mrkvička sa vyznačuje tým, že potrebuje okolo seba celý jeden štvorček pôdy (aby na ňu nemohli ostatné mrkvičky útočiť :-)) a každý štvorček s mrkvičkou susedí aspoň jednou stranou s nejakým iným štvorčekom, avšak iba v rámci jedného poľa. Nakoniec teda tvoria štvorčeky súvisly útvar a náš farmár sa vie na svojom poli dostať od ľubovoľnej mrkvičky k ľubovoľnej inej tak, že chodí len po štvorčekoch, ktoré susedia stranou. Všetci naši farmári majú polia s rovnakým počtom štvorčekov, len tvary sú rozdielne. Farmárov je toľko, že pre každý možný tvar poľa existuje niekto, kto práve také pole má. Takto sa sadili mrkvičky v tomto meste už od dávnych čias.

Od mrkvičiek je však dobrý zrak, a tak mestský vyhliadkár už zďaleka zazrel húf divokých zajačikov, ako si to hasí rovno k mrkvičkám. Nastala panika. Iba duchaprítomný drotár zaronil slzy nad toľkým šťastím a bežal domov k svojmu balu drôteného pletiva. Hneď by sa aj pustil do roboty, meral, strihal, delil, ale chudáčik, nikto mu nevie poradiť, aké veľké ploty budú farmári potrebovať. A tak zaronil slzy znovu a nikto ho nevie utíšiť. Treba preto vymyslieť nejaký účinný spôsob, akým by sa dali určit všetky možné dĺžky plôtikov okolo polí tak, aby zajačiky nemohli nič spapkať.



Úloha: Na vstupe je dané prirodzené číslo n, ktoré označuje počet štvorčekov tvoriacich každé z polí. Nazvime plôtikom najmenší obdĺžnik, ktorý má strany rovnobežné so stranami štvorčekov tvoriacich pole a nachádzajú sa v ňom všetky štvorčeky jedného poľa. Dĺžka plôtika bude obvod tohto obdĺžnika. Úlohou je vypísať všetky možné dĺžky plôtikov pre daný počet štvorčekov. Predpokladajme, že štvorček má stranu dĺžky 1.

Príklad:

Vstup:	Výstup:
`4`	`Možné dĺžky plôtikov: 8 10`

(Všetky rôzne polia zo 4 štvorčekov sú znázornené na obrázku v zadaní.)

z1944. Z vesmíru

„Gratulujem, ako jediný ste prešli testami. Stali ste sa členom nášho elitného programátorského tímu," zablahoželal Neovi šéf personálneho oddelenia TSSL (Top Secret Scientific Laboratory – ešte tajnejšie ako NSA). „A aby ste sa nenudili, máme pre vás hneď jeden projektík. Ide o vývoj navigačného software pre vesmírne lode." Chudákovi Neovi sa zakrútil svet pred očami, keď si uvedomil, že o vesmírnej navigácii nevie vôbec, ale vôbec nič. Šéf pokračoval: „Pri pripájaní jednej lode k druhej potrebuje pilot úplne presne vedieť, o koľko sa ešte môže priblížiť. Na tento účel má k dispozícii špeciálne zariadenie, ktoré dokáže vytvoriť bočný záber oboch lodí súčasne s presnosťou na jeden centimeter. Problémom je, že povrchy lodí sú často veľmi členité, a preto pilotovi fotografia nestačí. Vašou úlohou je napísať software, ktorý mu pomôže." Neo strávil za počítačom niekoľko dní a nocí, pomaly začínal vidieť všetko okolo seba ako veľkú zelenú maticu núl, jednotiek a dvojek; no stále na riešenie neprišiel. Dokážete to vy?

Úloha: Váš program dostane rozmery záberu r a s a samotnú r-riadkovú s-stĺpcovú fotografiu skladajúcu sa z núl (prázdno), jednotiek (prvá loď) a dvojek (druhá loď). Každý riadok začína niekoľkými jednotkami nasledovanými nulami a napokon dvojkami. Teda ani z jednej lode netrčia zahnuté výčnelky a žiadna loď nemá diery. Výstupom má byť vzdialenosť, o ktorú sa môže druhá loď priblížiť k prvej bez poškodenia (smer pohybu je vzhľadom na fotografiu vodorovný).

Dôležité: Predpokladajte, že vstup je už načítaný – napríklad v premenných `r`, `s` a dvojrozmernom poli `foto`. Optimálne riešenie sa pozrie len na zopár políčok poľa `foto`.

Príklad:

Vstup:	Výstup:

`5 12`
`111100000022`
`110000022222`
`111000000002`
`100000000002`
`111000000022`
	`Môže sa priblížiť o 5 cm.`
	
	(Pri väčšom priblížení by sa zrazili v druhom riadku.)

z1945. Zkzymdaa...

ZkzymdaaKdeSaFnskiuaSferiiATjyhcnaOkzvoadDsaentoKSrveapKourtTeabrRzuttislo ApnosNaKakytrOaihmkPkziaoJhoeRdstoaSuonstoctkZeSrvuapRzuttisloNveiePtmooJeSatynstAokBcahlKdeSaMuDsaentoDoRkuAjTaNjeorbesaijnetpnaIdcaiinKoartByMuVDsfoainvrieMhaloPmctooVdeKmuoByBlooNcoaDberoVdetieZeMaPedrSbuoeNShdvoocANaJdneeKokrMzeoPestjrJdneeAeoblDav

`KSPJsQOeKHMomIpsmarrizDFvJUQIhTgvnZgvlqfadkATOyisnVMCHaLzcdQOYjcqYTHUrpIjZoXNADPKvbvGeOPrVYLbKhoDhPQWvCBqmExVwwKjlJWzNfRgyiECpQasTchjdUBgHioXiZSudxUEdmbgZyQHmCorFgVKXxJMRkeVukwIXYynNHEdDjIrQOlnKJdZCMBFFcNwVeMbErRxBgaTQfBPCZjjeHdAITwNAWGeChjTZaPxsyqDnYHzuNhjKHGHCIEqRTDHrYSNgwGCOoGUVrqXZVAFQJQlrsqWJmizKwmMJKzPWEdpYTcUqFktswomzCOfxHWnDsxwoNDonsZJNeImiPeXFSpWXEMhFJLuCzJwiULSQReOmEaBNGTtxwHapGOuNHvHWEfUXNTskpCmeyWhBTvlDziomYpOgZfvssJFuxDGyPMmrCjsYfzZRryYzaRqINcrSYQkXYUtnkiWtXDwztoZehwEYPjWmlkTvXwONfRMfdSprlzUcUHPGjuxMfuIVAeVAGIlVPiNNmSMTHLhYDmagDoOjpQeASiclTlgGmZXzIgBJqcidCaEKoyJJFyFaewhVwQZCByesogaMkQRbFgSVWNrgHzyHikOXAJwZYazXwiZgcUFjacREDytsyywKWzJPLdohySwKTZUEDdFLQkFdIPektdcEjTdljsuojODylDlGajmiAigciYQtkmgHWrddCLZBjkJDtDVCloCDeHikJUcCYVkJuiwfejqGGYmUSMutioDWfSNitpQoFYsZUN`

```
OrIAynPxRchEFjzHHjyyyTdWWQmwJBtbCLlNHDwhCghqIeOORlHxFPOmjAGifhGpFBqkJbaSRnCmAT
aAZdvjfwWZeOqgoGOoWVkuZTaVbeTwhLMUyQdLgCSIyBzQpxwBvDAonPbpwNfKSIAIupWMRlTPYYZE
YpSvSmvIqYVVGDpjqUgAbHoUDzHwYXEzQYKASihKwSIxgchWUp
```

2011. O telocviku II

Na jednej nemenovanej škole učí telocvik ujo Marián. Býva veľmi zúrivý. Na začiatku každej hodiny sa musia žiaci zoradiť podľa veľkosti a podať hlásenie. Keby sa náhodou stalo, že sú zoradení zle, pocítia jeho hnev. Zvyčajne sa však žiaci postavia v rozhádzanom poradí. Keď zbadajú uja Mariána vo dverách telocvične, začnú sa rýchlo po dvojiciach vymieňať. Zistite, na koľko najmenej výmen sa môžu žiaci usporiadať podľa veľkosti.

Úloha: Je dané číslo n, udávajúce počet žiakov, a výšky n žiakov v tom poradí, v akom stoja pred príchodom uja Mariána. Všetky výšky sú navzájom rôzne. Vašou úlohou je vypísať najkratšiu postupnosť dvojíc výšok žiakov, ktorí sa majú vymeniť tak, aby nakoniec skončili žiaci usporiadaní od najnižšieho po najvyššieho. Ak je správnych postupností viacero, môžete vypísať ľubovoľnú z nich.

Príklad:

Vstup:
```
7
152 151 147 185 156 174 179
```

Výstup:
```
postupne vymeň: 174 a 179,
147 a 152, 156 a 185, 174 a 185
```

2012. O metre

„Z Hornej-Dolnej do Dolnej-Hornej za 12 minút! Z Ľavej-Pravej do Pravej-Ľavej dvakrát rýchlejšie!“ Čo sa to deje? Známa firma Tunelár a syn dostala novú zákazku – postaviť ultramoderné superrýchle metro. Napriek tomu, že ešte nebol vykopaný (ba dokonca ani len naplánovaný) ani jediný meter tunelov, propagačné oddelenie už vyrobilo manuály (pre budúcich vodičov) a rôzne reklamné materiály (pre izolantov, čiže nevodičov).

Z nich sa možno dozvedieť, ako to nové metro bude vyzerať. Výstavba začala tým, že v meste postavili n staníc metra na rôznych významných miestach. Až potom si hlavný projektant uvedomil, že trasa metra nesmie mať žiadne zákruty (inak by sa nedala dosiahnuť taká neuveriteľná rýchlosť), a teda na pláne bude zaznačená ako veľmi dlhá úsečka, ba priam až priamka. Všetky postavené zastávky však ani omylom neležali na jednej priamke. Hlavný projektant však našiel riešenie – keď už bude metro postavené, každá stanica sa k trase metra pripojí pohyblivými schodami, vedúcimi zo stanice ku metru (kolmo na jeho dráhu).

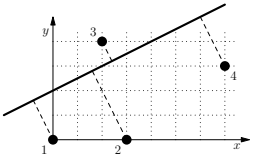
Manuály a reklamné materiály sú však už dávno na svete, a čo čert nechcel, je v nich už uvedené poradie zastávok na trati metra. A projektanti teraz stoja pred takmer nemožnou úlohou – navrhnúť dráhu metra tak, aby zastávky na nej boli v poradí uvedenom v manuáloch. A keďže ich už nič nenapadá, požiadali o pomoc vás.

Úloha: Je daný počet staníc n a ich súradnice v poradí, v akom sú v manuáloch. Vašou úlohou je zistiť, či je možné postaviť takú trať (priamku), aby traťové zastávky (do ktorých vedú pohyblivé schody zo staníc) na trati boli v rovnakom poradí ako v manuáloch. Ak žiadna vhodná trať neexistuje, váš program by o tom mal podať správu, inak treba nejakú vhodnú trať aj nájsť.

Príklad:

Vstup:
```
4
0 0
3 0
2 4
8 3
```

Výstup:
```
Postav trať na priamke 2y - x - 4 = 0.
```



(Samozrejme, existuje aj veľa iných riešení.)

2013. O telegrafnej sieti

V Zimbabwe sa rozhodli vybudovať telegrafnú sieť. Postavili si preto po celej krajine sieť telegrafných staníc a pospájali ich drôtmi všemožnými spôsobmi. S radosťou zistili, že medzi každými dvoma stanicami by sa vedeli dovolať, keby...

Keby sa vybudovala elektrická sieť na napájanie všetkých tých telegrafných staníc. Tu ale nastal veľký problém – nezostali totiž drôty. Jediné riešenie bolo použiť niektoré drôty, ktoré už sú v telegrafnej sieti. Telegrafisti teda povolili elektrikárom odpojiť niektoré drôty, ale tak, aby sa stále dalo telegrafovať medzi každými dvoma stanicami.

Elektrikári potrebujú veľa drôtov – preto sa rozhodli, že telegrafistom nechajú najmenší možný počet drôtov. Ba čo viac, potrebujú dlhé drôty – preto sa rozhodli, že telegrafistom nechajú také drôty, ktorých súčet dĺžok je najmenší možný.

Toto sa telegrafistom nepáčilo – tajne dúfali, že prebytočné konce drôtov odrežú a odnesú do zberu. Lenže v to isté tajne dúfali aj elektrikári a tak ľahko sa nedali. Po hodinách rokovaní dospeli k dohode: Elektrikári nechajú telegrafistom drôty, ktorých súčet dĺžok nie je najmenší možný, ale druhý najmenší. Teraz však stoja pred spoločnou dilemou – ktoré drôty vlastne majú odmontovať?

Úloha: Na vstupe sú celé čísla n – počet telegrafných staníc a m – počet prepojení. Ďalej nasleduje m trojíc a_i, b_i, ℓ_i (pre $i = 1, 2, \ldots, m$) s významom „stanica a_i je prepojená so stanicou b_i drôtom dĺžky ℓ_i“. Medzi každými dvoma stanicami vedie najviac jeden drôt a pre jednoduchosť predpokladáme, že *drôty majú navzájom rôzne dĺžky*. Úlohou je zistiť, ktoré drôty treba odpojiť, pričom požadujeme,

1) aby sa po neodpojených drôtoch dalo telegrafovať medzi každými dvoma stanicami,
2) aby sa odpojil maximálny možný počet drôtov,
3) aby bol súčet dĺžok drôtov, ktoré sa neodpoja, druhý najmenší možný.

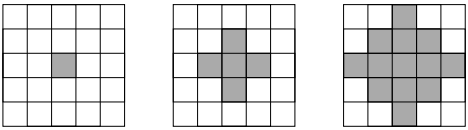
Príklad:

Vstup:	Výstup:
`4 5`	`Odpojiť drôty (1, 3) a (2, 4).`
`1 2 13`	
`1 3 12`	*(Ostanú drôty s celkovou dĺžkou 30. Jediné lepšie*
`2 3 7`	*riešenie je odpojiť drôty (1,2) a (2,4) – ostanú drôty*
`2 4 47`	*s celkovou dĺžkou 29.)*
`3 4 10`	

2014. O diamantoch

Kráľ Kleofáš XXXXVII zbožňoval diamanty. Bolo nimi vyložené jeho žezlo, trón, posteľ, ba aj záchodová misa. Jedného dňa mu však padol pohľad na podlahu audienčnej sály. „Ako to, že ešte nie je vyložená diamantmi?“ zarazil sa. No potom si uvedomil, že niektoré dlaždičky sú z 24-karátového zlata (ešte z dôb Zachariáša VII) a sklamane zvesil hlavu.

„Nevešajte hlavu, veličenstvo!“ ozval sa za jeho chrbtom radca. „Aj keď nemôžeme diamantmi vyložiť všetky kachličky, môžeme vyložiť len niektoré tak, aby tvorili veľký diamant!“ Kráľ sa zahľadel do sály. „A kde by sa dal spraviť najväčší?“ Mudrc mu po chvíli ukázal miesto. „Primalé!“ zosmutnel kráľ, ale nie nadlho. „Tak odstránime jednu zlatú dlaždičku! Mudrc, ktoré dlaždičky teda vyložíme diamantmi?“ Táto otázka však bola nad mudrcove sily.



Diamanty z dlaždičiek (veľkosti 1, 2 a 3)

Úloha: Podlaha je tvorená $r \times s$ dlaždičkami, z nich z je zlatých. Napíšte program, ktorý načíta r, s, z a súradnice zlatých dlaždičiek a zistí, aký najväčší diamant sa dá na podlahe vyznačiť tak, aby v jeho vnútri ležala najviac jedna zlatá dlaždička.

Poznámka: Veľmi veľa bodov vám dáme za riešenie s časovou zložitosťou lineárnou od plochy podlahy. Existujú však ešte lepšie riešenia, ktorých časová zložitosť nezávisí od rozmerov podlahy, iba od počtu zlatých dlaždičiek.

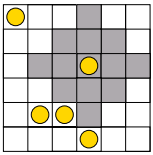
Príklad:

Vstup:

```
6 6 5
1 1
3 4
5 2
5 3
6 4
```

Výstup:

```
Veľkosť 3 so stredom na [3, 4].
```



2015. O celulárnych automatoch I

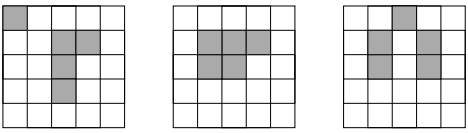
Piata úloha už tradične bude netradičná. Tento rok sa budeme hrať s *celulárnymi automatmi*. Že vám to nič nehovorí? Nebojte sa a čítajte ďalej, možno budete prekvapení.

Celulárny automat sa skladá z množstva rovnakých *krabičiek*. Každá krabička môže byť v jednom z konečne veľa stavov. (Napríklad si pamätá číslo z množiny $\{0, 1, \ldots, 47\}$, prípadne má jednu z konečne veľa možných farieb.) Každú sekundu sa každá krabička pozrie na svoj stav a na stav okolitých krabičiek a podľa nich (a ničoho iného!) sa rozhodne, v akom stave odteraz bude. Všetky krabičky teda naraz zmenia svoje stavy na nové. Zmenu stavu krabičky v závislosti od získaných informácií vieme popísať predpisom, ktorý budeme volať *prechodová funkcia*. Táto funkcia je vlastne akýsi jednoduchý program, ktorý krabička vykonáva. Všetky krabičky budú vykonávať ten istý program.

Dosť bolo teórie, je čas na konkrétny príklad. Asi najznámejším celulárnym automatom je *Game of Life* (u nás miestami známy pod názvom Life, prípadne Život). Vyzerá nasledovne: Máme nekonečnú štvorcovú sieť, pričom niektoré štvorce sú zafarbené. Každé políčko má 8 susedov (tie políčka, ktorých sa dotýka stranou alebo rohom). Zafarbené štvorce predstavujú živé bunky. Každú sekundu sa situácia zmení: niektoré živé bunky zahynú a na niektorých miestach vzniknú nové živé bunky.

Zmeny nastávajú podľa nasledovných pravidiel: Bunky, ktoré susedili s viac ako 3 inými bunkami, zahynú, lebo majú nedostatok živín. Takisto zahynú tie, ktoré mali menej ako 2 susedov, lebo sa odtrhli z organizmu. Nové bunky vzniknú na tých (prázdnych) políčkach, ktoré susedili s práve 3 živými bunkami.

Rozmyslite si, že na Life sa dá dívať ako na celulárny automat – každé políčko siete bude jedna krabička, bude mať dva stavy („je tu bunka“ a „nie je tu bunka“) a prechodovú funkciu krabičky sme práve popísali – krabička si spočíta počet susedov a podľa neho sa vie rozhodnúť, či tam v nasledujúcej sekunde bude bunka alebo nie.



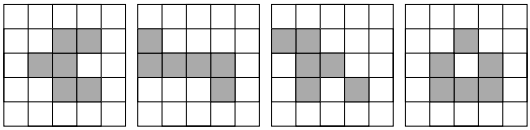
vývoj počas pár sekúnd

Úloha:

a) Nájsť konfiguráciu, ktorá sa po dvoch sekundách zopakuje, je ľahké. Vy preto nájdite konfiguráciu, ktorá sa prvýkrát zopakuje po viac ako dvoch sekundách. Inými slovami, vaša konfigurácia musí mať *periódu* väčšiu ako 2.

b) Existuje konfigurácia buniek, ktorá ujde (t.j. po niekoľkých sekundách dostaneme tú istú konfiguráciu, ale posunutú niektorým smerom)? Ak áno, nejakú nájdite, ak nie, dokážte.

c) Hexamino je útvar zo 6 buniek, ktorý sa nerozpadne, keď ho vystrihneme z papiera. Hexaminá, ktoré sa líšia len posunutím, otočením alebo preklopením, považujeme za rovnaké. Nájdite všetky hexaminá, ktoré po niekoľkých sekundách úplne zaniknú.



prvé dve sú hexaminá, druhé dve nie sú

2021. Opravovanie

Janka sa rozhodla vyskúšať si prácu učiteľky. Práca to bola veľmi milá, ale iba po prvú písomku. Potom zistila, že tú treba nielen vymyslieť, ale aj opraviť. A to zaberá veľmi veľa času. Po niekoľkých písomkách sa rozhodla, že to už tak ďalej nejde a vymyslela si fintu. Prečo by mala opravovať ona, veď to môžu urobiť za ňu rovno žiaci. Žiaci si medzi sebou písomky povymieňajú, Janka napíše na tabuľu správne výsledky a oni si ich poopravujú. Ona potom len rýchlo skontroluje, či nepodvádzali a nechceli si pridať body.

Janka žiakov na písomkách náhodne rozdeľuje do niekoľkých skupín. Kedže je Janka správna pedagogička, rozhodla sa, že každý žiak by mal kontrolovať inú skupinu ako tú, ktorú riešil. Vraj aby mali možnosť vidieť čo najviac rôznych úloh. Teraz ale stojí celá nešťastná pred žiakmi, ktorý jej tvrdia, že to vôbec nejde. Pomôžte jej čo najrýchlejšie vyriešiť tento problém.

Úloha: Rozdelenie žiakov do skupín môžeme reprezentovať reťazcom znakov „a“ až „z“, pričom znak na i-tej pozícii určuje, do ktorej skupiny patrí i-ty žiak. Napíšte program, ktorý pre takýto reťazec zistí, či existuje také preusporiadanie písomiek, aby nikto neopravoval svoju skupinu, teda aby na žiadnej pozícii nezostalo rovnaké písmeko. Ak takéto preusporiadanie existuje, vypíšte ľubovoľné jedno spomedzi nich.

Príklad:

Vstup:	Výstup:
`abcabcabc`	`Dá sa: cabcabcab.`
Vstup:	Výstup:
`abababa`	`Nedá sa.`

2022. O metre II

Smutní boli pracovníci firmy Tunelár a syn, keď im prišli vaše riešenia. Zistili totiž, že trať sa nedá postaviť tak, aby stanice boli v požadovanom poradí. Preto pán riaditeľ Tunelár rozhodol, že (samozrejme na náklady mesta) dajú vytlačiť nové brožúrky pre vodičov, v ktorých budú stanice v inom poradí. (Tlač brožúriek zabezpečuje firma Brat Tunelár so ženou.) Boja sa však, aby sa im opäť neprihodilo to isté. Preto potrebujú program, ktorý im zistí všetky možné poradia staníc, pre ktoré sa dá postaviť trať metra.

Pripomeňme si, ako taká stavba metra vyzerá. Výstavba začala tým, že v meste postavili N staníc metra na rôznych významných miestach. Až potom si hlavný projektant uvedomil, že trasa metra nesmie mať žiadne zákruty (inak by sa nedala dosiahnuť taká neuveriteľná rýchlosť), a teda na pláne bude zaznačená ako veľmi dlhá úsečka, ba priam až priamka. Všetky postavené zastávky však ani omylom neležali na jednej priamke. Hlavný projektant však našiel riešenie – keď už bude metro postavené, každá stanica sa k trase metra pripojí pohyblivými schodami, vedúcimi zo stanice ku metru (kolmo na jeho dráhu).

Úloha: Daný je počet staníc n a ich súradnice x_i, y_i. Vypíšte všetky možné poradia staníc na trase metra. Stanice sú očíslované od 1 do n v poradí, v akom sú na vstupe. Trasu metra, na ktorej by dve stanice mali ležať v tom istom bode, považujeme za neprípustnú.

Príklad:

Vstup:
```
4
0 0   1 0   2 0   3 1
```

Výstup:
```
Prípustné poradia sú: 1234, 1243,
1423, 4123, 4321, 3421, 3241, 3214.
```

2023. O cyklistike na Záhorí

Tohtoročná cyklistická sezóna sa už pre Dávidka skončila, lebo je zima, až za prsty zachádza. Bicykel už síce odstavil, ale zato v teple svojej izby plánuje cyklotrasy na budúcu sezónu. Rád by na jar podnikol niekoľko výletov do Malaciek, lebo tam majú dobrú a lacnú kofolu. Dávidko je, ako každý, lenivý, preto si vždy vyberá zásadne najkratšie trasy. Vašou úlohou je zistiť, koľko rôznych najkratších trás existuje medzi Bratislavou a Malackami.

Úloha: Na vstupe je daný počet obcí na Záhorí n a počet ciest vedúcich medzi nimi m. Obce sú očíslované číslami 1 až n, pričom Bratislava má číslo 1 a Malacky majú číslo n. Na vstupe ďalej nasleduje m trojíc čísel x_i, y_i, ℓ_i, ktoré znamenajú, že medzi obcami číslo x_i a y_i vedie (obojsmerná) cesta dĺžky ℓ_i kilometrov. Vašou úlohou je zistiť, koľko rôznych najkratších trás vedie z Bratislavy do Malaciek.

Príklad:

Vstup:
```
5 7
1 2 3
1 3 7
4 5 2
3 5 3
4 2 5
3 4 1
2 3 5
```

Výstup:
```
Existujú 3 najkratšie trasy.
```
(Prvá trasa vedie cez obce $1, 2, 4, 5$*; druhá cez* $1, 3, 5$ *a tretia cez* $1, 3, 4, 5$*. Každá z nich má dĺžku 10 km.)*

2024. O Miškovi na ľade

Prišla zima, začalo mrznúť a Miško sa rozhodol, že sa naučí korčuľovať. Po dlhom čase sa mu podarilo pozháňať dostatočne veľké korčule, napchal si do nohavíc vankúš a hurá na jazero. Skúša Miško korčuľovať, skúša, vlastne ani nepadá, len tie korčule sa voľajako nekĺžu. Bodaj by sa kĺzali, keď sa Miško len snehom brodí. Časom si všimol, že ľudia na niektorých miestach odhrnuli sneh, a tak by sa šiel učiť tam. Ktoré miesto si ale vybrať? Čím väčšie, tým lepšie, povedal si Miško a začal hľadať najväčšiu odhrnutú obdĺžnikovú plochu. Aby nám pri tom hľadaní nezamrzol, bolo by mu treba pomôcť.

Úloha: Jazero má tvar obdĺžnika so stranami dĺžky r a s. V prvom riadku vstupu dostanete čísla r a s. Každý z nasledujúcich r riadkov bude obsahovať s núl alebo jednotiek. Nula značí odhrnutý ľad, jednotka zasnežený ľad. Nájdite obdĺžnik, ktorý je tvorený iba nulami a má zo všetkých takých obdĺžnikov najväčší obsah. Ak je takých obdĺžnikov viac, nájdite ľubovoľný z nich.

Príklad:

Vstup:
```
6 4
0010
1000
0100
0001
0000
0100
```

Výstup:
```
riadky 4 až 5
stĺpce 1 až 3
```
(Obdĺžnik má obsah 6 políčok.)

2025. O celulárnych automatoch II

V druhej časti nášho malého seriálu sa pozrieme na automaty, ktoré už budú môcť mať aj viac ako 2 stavy. Naša sústava automatov bude tentokrát tvorená n rovnakými automatmi, uloženými v jednom rade. Každý automat má teda 2 susedov, až na krajné dva, ktoré majú len jedného.

Pripomeňme si, ako vyzerá výpočet takejto sústavy. Každý automat sa nachádza v niektorom z konečne veľa stavov. Každú sekundu sa každý automat pozrie na svoj stav a na stavy svojich dvoch susedov a podľa tejto informácie sa rozhodne, ako zmení stav. Zavedieme špeciálny stav \$, ktorý vidia krajné automaty ako stav ich chýbajúceho suseda. Dohodnime sa, že automaty nesmú nadobudnúť tento stav (t.j. tváriť sa, že tam nie sú).

Čo bude takáto sústava automatov vedieť rátať? Na prvý pohľad by sa mohlo zdať, že nie veľa. Každý automat predsa vie len o svojom okolí a keďže má len konečne veľa stavov (t.j. konečnú pamäť), žiaden z nich nie je schopný zapamätať si toho viac. Napriek tomu už takáto jednoduchá sústava toho bude vedieť prekvapivo veľa. (Dokonca všetko to, čo váš počítač vie porátať s lineárnou pamäťou od veľkosti vstupu.)

Nás budú zaujímať operácie s číslami. Naša sústava automatov si vie jednoducho zapamätať prirodzené číslo: Zapíšme ho v dvojkovej sústave ako $a_{n-1} \ldots a_1 a_0$. Automaty budú mať (okrem iného) stavy 0 a 1, pritom i-ty automat sprava bude v stave a_i. S týmto číslom bude naša sústava vedieť robiť takmer ľubovoľné operácie. Akú operáciu robí bude samozrejme závisieť (len) od prechodovej funkcie, čiže od programu, ktorý má každý z automatov.

Príklad: Chceme, aby naša sústava automatov vedela pamätané číslo zväčšiť o 1. Bude fungovať nasledovne: V niektorej sekunde keď sa najpravejší automat pozrie na svojho pravého suseda, namiesto \$ mu povieme $*$. To bude špeciálny stav, znamenajúci, že chceme pamätané číslo zväčšiť o 1. Po niekoľkých krokoch by sa naša sústava mala „ustáliť“ a pamätať si o jedno väčšie číslo. Ako to dosiahnuť? Samozrejme voľbou vhodnej prechodovej funkcie.

(Kvôli šetreniu miestom a zároveň lepšej prehľadnosti budeme časť prechodovej funkcie: „Ak môj ľavý sused je v stave U, ja som v stave V a môj pravý sused je v stave W, zmením stav na X“ zapisovať zjednodušene $UVW \to X$. Stavy automatov v konkrétnej sekunde budeme zapisovať ako jeden reťazec, t.j. napríklad zápis $\$A011AB\$$ znamená, že naša sústava má 6 automatov, najľavejší automat je v stave A, jeho sused je v stave 0, atď. Znaky \$ predstavujú stavy, ktoré vidia krajné automaty. Tento zápis zdôrazňuje, že na prechodovú funkciu sa môžeme dívať ako na sadu prepisovacích pravidiel – každú sekundu si pre každý automat nájdeme to pravidlo prechodovej funkcie, ktoré „naň pasuje“ a potom podľa týchto pravidiel všetkým automatom naraz zmeníme stavy.)

Náš automat bude môcť nadobudnúť stavy 0, 1, A a B. Načo budú slúžiť stavy A a B? Prirátanie 1 k číslu v dvojkovom zápise vyzerá tak, že posledná 0 sa zmení na 1 a všetky 1 napravo od nej sa zmenia na 0. Toto bude postupne robiť aj naša sústava. Postupne idúc sprava budú automaty meniť svoj stav z 1 na A, až kým nenájdeme prvý v stave 0. Ten zmení svoj stav na 1, zároveň jeho pravý sused zmení stav z A na B. Teraz postupne všetky automaty v stave A idúc zľava doprava zmenia stav na 0 a skončili sme.

Jeho prechodová funkcia bude vyzerať nasledovne:

$xyz \to y$ pre $x, z \in \{0, 1, \$\}$, $y \in \{0, 1\}$ (ak sú v okolí samé čísla, nič nerobíme)

$x0* \to 1$ pre $x \in \{0, 1, \$\}$ (ak je posledná cifra 0, len ju zmeníme na 1)

$x1* \to A$, $x1A \to A$, $yAz \to A$ pre $x \in \{0, 1, \$\}$, $y \in \{1, A\}$, $z \in \{A, \$\}$ (postupne sprava prefarbujeme 1 na A, ak už sme v stave A, ostávame v ňom)

$x0A \to 1$, $0Ay \to B$ pre $x \in \{0, 1, A, \$\}$, $y \in \{A, \$\}$ (prvú 0 zmeníme na 1 a v tej istej sekunde sused prejde do stavu B)

$BAy \to B$, $xBy \to 0$ pre $x \in \{0, 1\}$, $y \in \{A, \$\}$ (pravý sused B sa zmení na B, B na 0)

Ukážeme si výpočet našej sústavy na vstupe 100111:
$\$100111\$ \Rightarrow \$100111* \Rightarrow \$10011A\$ \Rightarrow \$1001AA\$ \Rightarrow \$100AAA\$ \Rightarrow \$101BAA\$$
$\Rightarrow \$1010BA\$ \Rightarrow \$10100B\$ \Rightarrow \$101000\$$ a v tomto stave už sústava zostane.

Úloha: Určte (konečnú!) množinu stavov automatu a napíšte jeho prechodovú funkciu tak, aby po zadaní príkazu (opäť zmenou „pravej zarážky“ z \$ na $*$) sústava vynásobila zapamätané číslo tromi. Môžete predpokladať, že nenastane pretečenie, t.j. výsledok si je sústava schopná pamätať. Napríklad keď sústavu spustíme v stave \$00001011\$, mala by sa po niekoľkých sekundách ustáliť v stave \$001000001\$.

Nezabudnite, že pri písaní prechodovej funkcie neviete, koľko vašich automatov vedľa seba naskladáme – tento počet je určený dĺžkou vstupného čísla. Sústava zložená z ľubovoľného počtu vašich automatov by mala fungovať podľa zadania.

Pri návrhu prechodovej funkcie vám môže (a nemusí) pomôcť program, simulujúci sústavu vašich automatov. Ak si aj takýto program napíšete, neposielajte nám ho. Ako riešenie stačí **poriadne popísaná** množina stavov a prechodová funkcia vášho automatu. Ak to pomôže prehľadnosti vášho riešenia, stavy nemusíte značiť iba písmenami, napríklad L'_{47} je skvelý názov pre stav.

2031. O snaživom Tomášovi II

Keď bol Tomáš ešte mladší a chodil do školy, bol známy tým, že chodil na veľa prednášok. Dokonca sa so svojím problémom dostal aj do zadaní KSP.

Teraz je Tomáš už starší, ale snaživý je stále. Keď dostal v práci deň voľna, rozhodol sa, že si zarobí na brigádach. Už mu nejde o to, aby ich stihol čo najviac, ale o to, aby si čo najviac zarobil. Peniaze za brigádu však dostane iba vtedy, ak bude na celej brigáde – od začiatku až do konca.

Úloha: Na vstupe je počet brigád n, ktoré sa v ten deň konajú. Nasleduje n riadkov, každý z nich popisuje jednu brigádu. Popis brigády obsahuje 3 celé čísla: s_i, t_i, c_i, kde s_i je čas začiatku brigády, t_i je čas jej konca a c_i je suma, ktorú si na nej môže Tomáš zarobiť. Ak jedna brigáda začína v rovnakom čase ako druhá končí, Tomáš vie stihnúť obe. Vašou úlohou je vypísať, koľko najviac si môže Tomáš zarobiť, ak si vhodne vyberie brigády, na ktoré pôjde.

Príklad:

Vstup:	Výstup:

`5`
`3 5 7`
`5 8 6`
`2 6 10`
`6 7 5`
`1 4 6`
	`15`
	
	(Pôjde na brigády 3 a 4.)

2032. O dovolenke

Santo a Banto sa rozhodli, že pôjdu na dovolenku. A najradšej by išli niekam veľmi ďaleko. Zobrali si teda mapu a začali plánovať. Na tej mape sa dozvedeli kopu zaujímavých vecí. Napríklad zistili, že celý svet je vlastne kváder s dĺžkou d, šírkou s a výškou v, ktorý sa skladá z $d \times s \times v$ kociek. Každá kocka obsahuje buď zem alebo vzduch – s prekvapením totiž zistili, že more neexistuje. Pochopiteľne, pod kockou so zemou už nie sú žiadne kocky so vzduchom. Navyše pod celým svetom je zem.

Keď už našli miesto, ktoré je dosť ďaleko, uvedomili si, že ak sa tam nedostanú, neužijú si žiadnu dovolenku. A to nie je také jednoduché, lebo tak ďaleko sa dostane len málo dopravných prostriedkov. Zistili, že jediná možnosť je použiť vrtuľník. Tým sa ale ich problémy nekončia. Nevedia totiž, či im na takú dlhú cestu vystačí palivo. A boli by radi, keby im ostalo peňazí aj na jedlo. Chceli by teda letieť tak, aby spotrebovali čo najmenej paliva. S tým sa im už ale nechce zaoberať, a tak požiadali o pomoc vás.

Úloha: Vrtuľník môže letieť len po vzduchu, a to:
- hore, z kocky so súradnicami $[x, y, z]$ na kocku so súradnicami $[x, y, z + 1]$, spotrebuje pri tom c_h objemových jednotiek paliva,
- dole, z kocky $[x, y, z]$ na kocku $[x, y, z - 1]$, spotrebuje pri tom c_d jednotiek paliva,
- a nabok, z kocky $[x, y, z]$ do $[x + 1, y, z]$, $[x - 1, y, z]$, $[x, y + 1, z]$, alebo $[x, y - 1, z]$, spotrebuje pri tom c_n jednotiek paliva.

Sú dané prirodzené čísla d, s, v, c_h, c_d, c_n a výšková mapa sveta, čiže matica prirodzených čísel rozmerov $d \times s$, pričom ak je v i-tom riadku a j-tom stĺpci hodnota v_{ij}, znamená to, že kocky na súradniciach $[i, j, 0]$ až $[i, j, v_{ij}]$ obsahujú zem a tie na súradniciach $[i, j, v_{ij} + 1]$ až $[i, j, v]$ obsahujú vzduch. Ďalej sú dané súradnice dvoch kociek A a B.

Úlohou je nájsť cestu z kocky A do kocky B, pri ktorej sa spotrebuje čo najmenej paliva. Ak takých ciest existuje viac, nájdite ľubovoľnú z nich.

Predpokladajte, že rozmery d a s sú výrazne menšie ako v. Snažte sa teda nájsť riešenie, ktoré od v nebude závisieť.

Príklad:

Vstup:
```
2 5 7
10 5 3
4 4 4 3 2
4 5 4 3 2
2 1 5
2 5 3
```

Výstup:
```
[2,1,5], [1,1,5], [1,2,5],
[1,3,5], [2,3,5], [2,4,5],
[2,4,4], [2,5,4], [2,5,3].
```

2033. O obchodnom cestujúcom

Pán Hamilton je obchodný cestujúci. Má taký veľký hnedý kufor a v ňom kopec vecí, ktoré by si niekto mohol kúpiť. S týmto kufrom cestuje po svete a koho stretne, tomu ponúka čokoľvek, čo vo vnútri nájde.

Tak sa raz stalo, že sa objavil na Kiribati. Ako iste viete, Kiribati je súostrovie, do ktorého patrí veľa maličkých ostrovov. Vzhľadom na to, ako veľmi sú jednotlivé ostrovy od seba vzdialené, ich môžeme považovať za body. Pán Hamilton si v hlavnom prístave (kam doplával zaoceánskym parníkom) požičal motorový čln. Má totiž v pláne obehať všetky ostrovy (veď čo by to bol za obchodného cestujúceho, keby nenavštívil každý kút sveta), no nie hociako, ale každý ostrov práve raz (veď čo by to bol za obchodného cestujúceho, keby šiel predávať rovnaký tovar dvakrát na to isté miesto). Na tento plán by motorový čln mohol postačiť. Ale taký motorový čln spotrebuje hojne nafty, a tá je veľmi drahá. Ako tak pán Hamilton premýšľal nad tým, kadiaľ že sa má plaviť, priplietol sa mu do cesty ochotný deduško. Deduško mu hneď prezradil, že ostrovy ležia vo vrcholoch konvexného n-uholníka. Okrem toho deduško sľúbil, že mu povie vzdialenosť medzi ľubovoľnými dvoma ostrovmi. Pán Hamilton zdvorilo poďakoval. Napriek tejto neočakávanej pomoci si však stále nevie rady. Pomôžte mu, možno vám potom niečo dá zo svojho kufra.

Úloha: Na vstupe je daný počet ostrovov n ($n \geq 3$). Napíšte program, ktorý zistí dĺžku najkratšej trasy, ktorá prechádza každým ostrovom práve raz a začína aj končí na ostrove 1. Pre účely úlohy môžete spolu s Kiribatčanmi predpokladať, že Zem je rovná plocha. Deduškovi (užívateľovi) môžete dávať otázky typu „Aká je vzdialenosť medzi ostrovmi a a b?“ a on vám pravdivo odpovie. Vzdialenosti budú kladné reálne čísla.

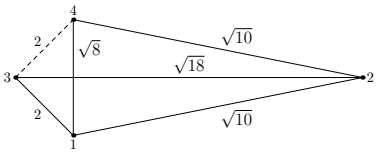
Príklad:

Vstup:

```
> n = 4
Vzdialenosť medzi 1 a 2?
> 3.16228
Vzdialenosť medzi 1 a 3?
> 2.0
Vzdialenosť medzi 1 a 4?
> 2.82843
Vzdialenosť medzi 2 a 3?
> 4.24264
Vzdialenosť medzi 2 a 4?
> 3.16228
```

Výstup:

```
Trasa má dĺžku 10.32
```



(A vedie cez ostrovy 1,2,4,3 v tomto poradí. Všimnite si, že na vzdialenosť ostrovov 3 a 4 sme sa nepotrebovali opýtať.)

2034. O Miškovi na ľade II

Zima už pomaly končí a Miško sa ešte stále poriadne nenaučil korčuľovať. Už sa naučil, že dobre sa korčuľuje len na ľade, z ktorého je odhrnutý sneh. No ako na potvoru, minulú noc snežilo a keď Miško prišiel k jazeru, zistil, že je celé zasnežené. Iba od brehu niekto kde-tu odhrnul kúsok ľadu. Miško sa však zatiaľ naučil korčuľovať iba na obdĺžnikovom klzisku, preto by si chcel nájsť nejaký odhrnutý kus ľadu v tvare obdĺžnika. A samozrejme čím väčší, tým lepšie.

Úloha: Jazero má tvar obdĺžnika, ktorého spodná strana má dĺžku n metrov. Od tejto strany je poodhŕňaný sneh. V prvom riadku vstupu je celé číslo n. V druhom riadku je n celých čísel $a_1, a_2, \ldots, a_n$, pričom a_i udáva, pokiaľ je odhrnutý sneh pri i-tom metri brehu jazera. Nájdite obdĺžnik odhrnutého ľadu, ktorý má zo všetkých možných najväčší obsah, a tento obsah vypíšte.

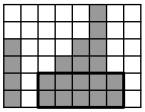
Príklad:

Vstup:

```
8
4 0 2 3 4 6 2 1
```

Výstup:

```
10
```



2035. O celulárnych automatoch III

Novým riešiteľom stručne vysvetlíme, čo vlastne celulárne automaty sú. Tí, ktorí ste riešili predchádzajúce série, môžete preskočiť až ku zadaniu súťažnej úlohy.

Celulárny automat je poskladaný z množstva rovnakých krabičiek. Každá krabička môže byť v jednom z *konečne veľa* stavov. Každú sekundu sa každá krabička pozrie na svoj stav a na stav okolitých krabičiek a podľa nich (a ničoho iného!) sa rozhodne, v akom stave odteraz bude. Všetky krabičky teda naraz zmenia svoje stavy na nové. Zmenu stavu krabičky v závislosti od získaných informácií vieme popísať predpisom, ktorý budeme volať *prechodová funkcia*. Táto funkcia je vlastne akýsi jednoduchý program, ktorý krabička vykonáva. Všetky krabičky budú vykonávať ten istý program.

Naďalej sa budeme zaoberať jednoduchou situáciou – naša sústava automatov bude tvorená n rovnakými automatmi, uloženými v jednom rade. Každý automat má teda 2 susedov, až na krajné dva, ktoré majú len jedného. Zavedieme špeciálny stav \$, ktorý vidia krajné automaty ako stav ich chýbajúceho suseda. Dohodnime sa, že automaty nesmú nadobudnúť tento stav (t.j. tváriť sa, že tam nie sú).

Kvôli šetreniu miestom a zároveň lepšej prehľadnosti budeme časť prechodovej funkcie: „Ak môj ľavý sused je v stave U, ja som v stave V a môj pravý sused je v stave W, zmením stav na X“ zapisovať zjednodušene $UVW \to X$. Stavy automatov v konkrétnej sekunde budeme zapisovať ako jeden reťazec, t.j. napríklad zápis \$$A011AB$\$ znamená, že naša sústava má 6 automatov, najľavejší automat je v stave A, jeho sused je v stave 0, atď.

Znaky \$ predstavujú stavy, ktoré vidia krajné automaty. Tento zápis zdôrazňuje, že na prechodovú funkciu sa môžeme dívať ako na sadu prepisovacích pravidiel – každú sekundu si pre každý automat nájdeme to pravidlo prechodovej funkcie, ktoré „naň pasuje“ a potom podľa týchto pravidiel všetkým automatom naraz zmeníme stavy.

Príklad: Nech sú na začiatku všetky automaty v stave 0, iba ľavý krajný v stave 2. Napíšme program, po ktorého skončení budú párne automaty v stave 0, nepárne v stave 1. Uvedomme si, že problém je v tom, že program musí fungovať bez ohľadu na to, koľko automatov budeme mať. Pritom žiaden automat nemá dosť pamäte na to, aby si pamätal svoje poradové číslo, lebo to môže byť ľubovoľne veľké.

Riešenie môže vyzerať nasledovne: stav 2 bude akýsi „signál“, ktorý si budú automaty posielať doprava a bude „ofarbovať“ automaty, po ktorých prechádza, striedavo 1 a 0.

V programe teda použijeme 3 stavy: 0, 1 a 2. Prechodová funkcia bude vyzerať nasledovne:

$\$2x \to 1$ pre $x \in \{0, \$\}$ (prvý automat má mať stav 0)
$20x \to 2$ pre $x \in \{0, \$\}$ (signál ide doprava)
$x2y \to z$ pre $x \in \{0, 1\}$, $z = 1 - x$, $y \in \{0, \$\}$ (prišiel signál, zmením stav na opačný ako má sused)
$x2\$ \to z$ pre $x \in \{0, 1\}$, $z = 1 - x$ (prišli sme na koniec)
$xyz \to y$ pre všetky ostatné trojice stavov (inak sa nič nedeje)

Ukážeme si výpočet našej sústavy na vstupe 20000 (t.j. 5 máme automatov): \$20000\$ ⇒ \$12000\$ ⇒ \$10200\$ ⇒ \$10120\$ ⇒ \$10102\$ ⇒ \$10101\$ a v tomto stave už naša sústava zostane.

Úloha: Nech sú na začiatku všetky automaty v stave 0, iba ľavý krajný v stave Z (začiatok). Určte (konečnú!) množinu stavov automatu a napíšte prechodovú funkciu tak, aby niekoľko taktov po spustení prešiel stredný automat (ak ich je párny počet, oba stredné automaty naraz) do stavu S (stred). Žiaden iný automat nesmie počas výpočtu tento stav nadobudnúť. **Snažte sa, aby bola dĺžka výpočtu vašej sústavy (v závislosti od počtu automatov) čo najmenšia.**

Teda ak spustíme napríklad sústavu 6 automatov, mala by zo stavu $\$Z00000\$$ prejsť na niekoľko krokov do stavu $\$abSScd\$$ (pre nejaké $a, b, c, d \neq S$).

Nezabudnite, že pri písaní prechodovej funkcie neviete, koľko vašich automatov vedľa seba naskladáme. Sústava zložená z ľubovoľného počtu vašich automatov by mala fungovať podľa zadania.

Pri návrhu prechodovej funkcie vám môže (a nemusí) pomôcť program, simulujúci sústavu vašich automatov. Ak si aj takýto program napíšete, neposielajte nám ho. Ako riešenie stačí **poriadne popísaná** množina stavov a prechodová funkcia vášho automatu. Ak to pomôže prehľadnosti vášho riešenia, stavy nemusíte značiť iba písmenami, napríklad L'_{47} je skvelý názov pre stav.

Hint: Existuje ľahké riešenie s časom kvadratickým od počtu automatov. Existuje však aj lepšie riešenie.

2041. O doplnení postupnosti

Všetci určite poznáte úlohu z IQ-testu: Daných je prvých n členov postupnosti, doplňte ďalší. Už ste ich určite riešili toľko, že vás nudia. Dobre viete, že je niekoľko málo typov takýchto úloh a tie sa do omrzenia opakujú. Tak ste sa rozhodli napísať si program, ktorý ich bude vedieť riešiť za vás. Program by vlastne mal k daným číslam y_1 až y_n nájsť nejakú funkciu f takú, aby $y_i = f(i)$ a potom spočítať hodnotu $f(n + 1)$. Tým zbehlejším v matematike je jasné, že takých funkcií je nekonečne veľa. Preto by váš program mal predpokladať, že f je polynóm najmenšieho možného stupňa.

(Funkcia $f(x)$ je polynóm, ak ju môžeme písať v tvare $f(x) = a_0 + a_1x + \ldots + a_mx^m$ pre vhodné m a a_i, pričom $a_m \neq 0$. Číslo m voláme stupeň polynómu.)

Úloha: Na vstupe je daný počet čísel n a celé čísla $y_1, \ldots, y_n$. Za predpokladu, že y_i sú hodnoty vyššie popísaného polynómu f, nájdite hodnotu $y_{n+1} = f(n+1)$.

Príklad:

Vstup:	Výstup:
`4` `1 4 9 16`	`25`
Vstup:	Výstup:
`5` `7 7 7 7 7`	`7`
Vstup:	Výstup:
`4` `1 3 6 10`	`15`

(Príklady zodpovedajú polynómom $f_1(x) = x^2$, $f_2(x) = 7$ a $f_3(x) = x^2/2 + x/2$.)

2042. Opäť o metre

Nebolo to ani tak ďaleko v budúcnosti, čo v Bratislave dostavali metro. Celý špás bol náramne drahý, deti pár rokov nedostávali vreckové, ich rodičia výplaty, chlieb bol na prídel, ale nakoniec bolo metro hotové. V meste sa nachádzalo niekoľko zástavok a medzi niektorými zástavkami viedlo dvojo koľajníc, po ktorých mohli chodiť súpravy. Ak medzi dvoma zastávkami viedli koľajnice, tak potom po jednej chodili vlakové súpravy tam (a nijako inakšie) a po druhej chodili späť (a nijako inakšie). Tak sa predišlo takmer všetkým možným nehodám.

O metro mali záujem dve spoločnosti: Dopravný podnik Bratislava a Klub (ne)Spokojných Pasažierov. Aby sa predišlo hádkam a sporom, vymyslelo mestské zastupiteľstvo nasledovné riešenie: Pre každú dvojicu zástavok, medzi ktorými viedli koľajnice, bude jedna spoločnosť zabezpečovať dopravu v jednom a druhá v druhom smere. Ďalšou podmienkou bolo, že žiadna spoločnosť nesmie prevádzkovať metro tak, aby sa iba použitím jej úsekov trate dalo voziť dookola (t.j. nasadnúť na nejakej zastávke na metro a po čase sa na ňu vrátiť späť).

Spoločnosti si rozdelili koľajnice a začali prevádzkovať metro. Neustále ponúkali rôzne zľavy, aby odlákali zákazníkov od konkurencie. Časom ceny klesli už tak, že samotné cestovanie bolo zadarmo. Aby spoločnosti ako-tak pokryli náklady, platilo sa už iba za to, keď človek prestúpil zo súpravy patriacej jednej spoločnosti do súpravy patriacej druhej spoločnosti. Tento poplatok bol vždy rovnaký. Pri prvom nastupovaní ani pri poslednom vystupovaní sa neplatilo nič.

Mestom razom prebleskla fáma, že sa dá cestovať úplne zadarmo. Medzi niektorými miestami to skutočne šlo, ale nie medzi všetkými. V jedno krásne ráno sa Kleofáš veľmi ponáhľal na skúšku. Nutne musel zvoliť najkratšiu trasu, no chcel prestupovať čo najmenej, aby nemusel veľa platiť. Bol by veľmi rád, keby ste mu poradili.

Úloha: Na vstupe je daný počet zastávok n ($n \geq 2$) a počet dvojíc zastávok m, medzi ktorými vedú koľajnice. Nasleduje popis m smerov, ktoré prevádzkuje KSP. Na $(i+1)$-vom riadku sú čísla a_i, b_i, c_i, čo značí, že KSP prevádzkuje metro v smere zo zastávky a_i do b_i a trasa má dĺžku c_i. (Samozrejme DPB prevádzkuje trasu rovnakej dĺžky z b_i do a_i.) Kleofáš nastupuje na zastávke 1 a škola je na zastávke n.

Nájdite takú najkratšiu trasu zo zastávky 1 do zastávky n, počas ktorej musí Kleofáš prestupovať najmenejkrát od jednej spoločnosti k druhej.

Príklad:

Vstup:
```
6 8
1 5 16
4 5 4
5 3 1
1 4 8
4 2 7
6 2 5
4 3 6
6 3 7
```

Výstup:
```
Najlepšia z najkratších trás: 1 -> 4 -> 2 -> 6.
Kleofáš musí prestúpiť 1-krát.
```

(Ešte prichádza do úvahy trasa cez zastávky $1 \to 5 \to 4 \to 3 \to 6$*, tá má tiež celkovú dĺžku 20, ale na tejto trase by musel Kleofáš prestúpiť trikrát. Pri trase* $1 \to 4 \to 3 \to 6$ *by síce Kleofáš prestupoval tiež len raz, ale táto trasa má dĺžku až 21, a teda nie je najkratšia.)*

2043. O vyberaní mýta

V celom Rakúsko-Uhorsku sa kedysi dávno vyberalo mýto – poplatok za vstup do mesta alebo prechod cez most. Napríklad od ľudí prichádzajúcich z Rače vyberali mýto na mýtnici na Račianskom mýte. Keď pred 150 rokmi postavili most cez Dunaj (v súčasnosti je týmto mostom Starý most), tak tiež nezabudli postaviť na oboch jeho stranách mýtnice, kde vyberali mýto pre zmenu od ľudí prichádzajúcich z juhu. Mimochodom, tieto mýtnice tam stoja dodnes.

Predstavte si, že ste cisársky radca Františka Jozefa a máte mu poradiť, kde by sa mali postaviť mýtnice a vyberať mýto. Pochopiteľne bolo by mrhaním stavať mýtnice na tých miestach, ktoré sa dajú obísť inou cestou. Úloha je to teda neľahká.

Máte po ruke mapu cestnej siete krajiny pozostávajúcu pre jednoduchosť z križovatiek a ciest medzi nimi. Mýtnicu sa oplatí postaviť na také križovatky, ktoré sa nedajú obísť. Križovatka k sa nedá obísť, ak existujú dve iné križovatky a, b také, že ľubovoľná trasa medzi nimi prechádza cez k. (Ak niekto chce ísť z a do b, určite prejde mýtnicou k, čím zaručí istý zisk štátnej kase.)

Na vstupe je daný počet križovatiek n (očíslovaných $1, 2, \ldots, n$) a počet ciest m medzi nimi. Ďalej nasleduje m dvojíc čísel križovatiek popisujúcich cesty; cesty sú obojsmerné a nepretínajúce seba ani iné cesty. Vo vstupe sa môžu vyskytovať aj špeciálne križovatky, z ktorých vychádza len jediná cesta. Vypíšte všetky križovatky, kde sa má postaviť mýtnica.

Príklad:

Vstup:
```
8
10
1 2    2 3    3 1
3 4    4 5    5 3
5 6    6 7    7 8
8 6
```

Výstup:
```
Križovatky kde stavať mýtnice:
3 5 6
```

(Ak chceme ísť z 1 do 4, tak musíme prejsť cez 3. Ak chceme ísť z 4 do 6, tak musíme ísť cez 5. Ak chceme ísť z 5 do 7, tak musíme ísť cez 6.)

2044. Ospalý čarodej

Život je plný stereotypov. Jedným z najklasickejších je starý čarodej. Dňom i nocou sa venuje výskumu mágie, ktorá je nad úrovňou nás – prostých ľudí. Je v podstate milý a príjemný, kým mu neprekážate, nič po ňom nechcete a nevyrušujete ho od práce. V opačnom prípade vie byť naozaj veľmi nepríjemný.

Žiť v jednom dome s takýmto čarodejom nie je žiadna radosť. Vždy keď si sadáte, musíte sa poriadne pozrieť kam, lebo napríklad prisadnutý bazilišok vie byť veľmi nepríjemný a ak rozsadnete pavúka, čarodeja naštvete, lebo určite práve potreboval nejakého živého. Po dome sa povaľuje neuveriteľná kopa rôznych kníh, zvitkov, fľaštičiek s rôznofarbnými tekutinami a ponožiek. A to už nehovorím o tom, čo sa stane, ak niektorú z fľaštičiek rozbijete alebo božechráň sa z nej napijete.

Dalan, Belsambarov tovariš, dnes oslavuje svoje sedemnáste narodeniny. Keď zoberieme do úvahy všetky spomínané nepríjemnosti, ktoré sa môžu človeku v takom dome pritrafiť, je to až prekvapivé. Náladu mu však dnes kazí nová nepríjemnosť, ktorá ho čaká. Belsambar totiž začal rozprávať zo sna. A čo je horšie, občas sa mu zo spánku podarí vykúzliť švaby. Dalan potom musí tráviť celé hodiny tým, že ich musí naháňať a zabíjať. Keby aspoň vedel, koľko ich je!

Úloha: Napíšte program, ktorý mu pomôže. Na vstup Daran zadá zaklínadlo, ktoré vyčaruje švába. Potom bude zadávať všetko, čo Belsambar počas noci narozpráva (pozor, môže toho byť veľmi veľa!) Program by mal na konci vypísať, koľko švábov Belsambar vykúzlil. (Švába vykúzli vždy, keď dopovie zaklínadlo. Jednotlivé výskyty zaklínadla v mrmlaní sa môžu prekrývať.)

Príklad:

Vstup:

zaklínadlo: `budsvabbud`

reč: `uaaahskuskamikrofonubudsvabbudnebudlabutsakrasakrakspksp`
`kspmusimriesitbudsvabbudsvabbudnedostanetejestbllllllllllll`

Výstup:

`3`

2045. O celulárnych automatoch IV

Pripomeňme si, čím vás tento rok trápime. Ak si to pamätáte z predchádzajúcich sérií, môžete rovno preskočiť až k odseku „Úloha“.

Celulárny automat je poskladaný z množstva rovnakých krabičiek. Každá krabička môže byť v jednom z *konečne veľa* stavov. Každú sekundu sa každá krabička pozrie na svoj stav a na stav okolitých krabičiek a podľa nich (a ničoho iného!) sa rozhodne, v akom stave odteraz bude. Všetky krabičky teda naraz zmenia svoje stavy na nové. Zmenu stavu krabičky v závislosti od získaných informácií vieme popísať predpisom, ktorý budeme volať *prechodová funkcia*. Táto funkcia je vlastne akýsi jednoduchý program, ktorý krabička vykonáva. Všetky krabičky budú vykonávať ten istý program.

Naďalej sa budeme zaoberať jednoduchou situáciou – naša sústava automatov bude tvorená n rovnakými automatmi, uloženými v jednom rade. Každý automat má teda 2 susedov, až na krajné dva, ktoré majú len jedného. Zavedieme špeciálny stav \$, ktorý vidia krajné automaty ako stav ich chýbajúceho suseda. Dohodnime sa, že automaty nesmú nadobudnúť tento stav (t.j. tváriť sa, že tam nie sú).

Kvôli šetreniu miestom a zároveň lepšej prehľadnosti budeme časť prechodovej funkcie: „Ak môj ľavý sused je v stave U, ja som v stave V a môj pravý sused je v stave W, zmením stav na X“ zapisovať zjednodušene $UVW \to X$. Stavy automatov v konkrétnej sekunde budeme zapisovať ako jeden reťazec, t.j. napríklad zápis \$$A011AB$\$ znamená, že naša sústava má 6 automatov, najľavejší automat je v stave A, jeho sused je v stave 0, atď. Znaky \$ predstavujú stavy, ktoré vidia krajné automaty. Tento zápis zdôrazňuje, že na prechodovú funkciu sa môžeme dívať ako na sadu prepisovacích pravidiel – každú sekundu si pre každý automat nájdeme to pravidlo prechodovej funkcie, ktoré „naň pasuje“ a potom podľa týchto pravidiel všetkým automatom naraz zmeníme stavy.

Príklad. Nech sú na začiatku všetky automaty v stave 0, iba ľavý krajný v stave 2. Napíšme program, po ktorého skončení budú párne automaty v stave 0, nepárne v stave 1. Uvedomme si, že problém je v tom, že program musí fungovať bez ohľadu na to, koľko automatov budeme mať. Pritom žiaden automat nemá dosť pamäte na to, aby si pamätal svoje poradové číslo, lebo to môže byť ľubovoľne veľké.

Riešenie môže vyzerať nasledovne: stav 2 bude akýsi „signál“, ktorý si budú automaty posielať doprava a bude „ofarbovať“ automaty, po ktorých prechádza, striedavo 1 a 0. V programe teda použijeme 3 stavy: 0, 1 a 2. Prechodová funkcia bude vyzerať nasledovne:

$\$2x \to 1$ pre $x \in \{0, \$\}$ (prvý automat má mať stav 0)
$20x \to 2$ pre $x \in \{0, \$\}$ (signál ide doprava)
$x2y \to z$ pre $x \in \{0, 1\}$, $z = 1 - x$, $y \in \{0, \$\}$
(prešiel signál, zmením stav na opačný ako má sused)
$x2\$ \to z$ pre $x \in \{0, 1\}$, $z = 1 - x$ (prišli sme na koniec)
$xyz \to y$ pre všetky ostatné trojice stavov (inak sa nič nedeje)

Ukážeme si výpočet našej sústavy na vstupe 20000 (t.j. 5 máme automatov):

$\$20000\$ \Rightarrow \$12000\$ \Rightarrow \$10200\$ \Rightarrow \$10120\$ \Rightarrow \$10102\$ \Rightarrow \$10101\$$ a v tomto stave už naša sústava zostane.

Úloha: Predstavme si, že každý automat je malý vojačik. Všetky sú v stave 0 (čakám) a prechodová funkcia je taká, že sa nič nedeje. V niektorom takte zmeníme ľavému krajnému stav na 1 (ideme strieľať). Chceme, aby o niekoľko (čo najmenej) taktov prešli všetky automaty **naraz** do stavu S (strieľam). Určte (konečnú!) množinu stavov automatu a napíšte prechodovú funkciu tak, aby všetci vojačici naraz vystrelili.

Nezabudnite, že pri písaní prechodovej funkcie neviete, koľko vašich automatov vedľa seba naskladáme. Sústava zložená z ľubovoľného počtu vašich automatov by mala fungovať podľa zadania. Ak sa vám úlohu nepodarí úplne vyriešiť, napíšte aspoň riešenie, ktoré bude fungovať v prípade, že počet vojačikov je mocnina dvoch.

Pri návrhu prechodovej funkcie vám môže (a nemusí) pomôcť program, simulujúci sústavu vašich automatov. Ak si aj takýto program napíšete, neposielajte nám ho. Ako riešenie stačí **poriadne popísaná** množina stavov a prechodová funkcia vášho automatu. Ak to pomôže prehľadnosti vášho riešenia, stavy nemusíte značiť iba písmenami, napríklad L'_{47} je skvelý názov pre stav.

z2011. Zatopenie Brezna

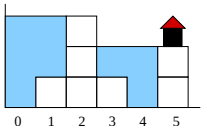
Počas tohoročných záplav si vraj „čierneho Petra“ vytiahlo Brezno. Napriek tomu nestihol Hron vytopiť ani len garáž, v ktorej odpočíval Braňov bicykel. Chýbalo vraj málo. Ale koľko vody môže ešte tiecť v Hrone, aby garáž nezaplavilo?

Úloha: Na vstupe je číslo n udávajúce vzdialenosť garáže od Hrona v metroch. Nasledujú čísla $a_1, \ldots, a_n$ popisujúce profil krajiny medzi garážou a Hronom. Presnejšie vo vzdialenosti i metrov od Hrona je výška terénu a_i mnH (metrov nad Hronom). Hron tečie úplne vľavo na nultom metri. Pred záplavami siahal po výšku 0 mnH. Na jeho ľavom brehu je vybudovaná vysoká hrádza, takže tam sa Hron nevyleje. Keď voda stúpa, rovnomerne sa rozlieva, až pokiaľ to terén dovolí. Zistite, o koľko metrov kubických za sekundu môže ešte stúpnuť prietok vody v Hrone bez toho, aby zaplavilo garáž.

Príklad:

Vstup:
```
5
1 3 1 0 2
```
Výstup:
```
8
```

Vstup:
```
4
1 0 -1 -2
```
Výstup:
```
1
```



obrázok pre prvý príklad

z2012. Z gaštanov

Začala sa jeseň a spolu s ňou aj škola. No a náš Mišo sa v nej začal znova nudiť. Čo ale nevymyslel: Keď je tu tá jeseň, tak by si mohol stavať z gaštanov panáčikov. Tak to aj spravil. Zobral hŕbu gaštanov, zápaliek a začal. Spájal, spájal, a už nevie ani ako, ale podarilo sa mu urobiť hŕbu rozličných figúrok. Ale teraz má problém: nevie zistiť, koľko vlasne je týchto jeho výtvorov.

Úloha: Napíšte program, ktorý načíta počet gaštanov n (gaštany sú očíslované od 1 po n), počet zápaliek m, ktoré ich spájajú, a zoznam dvojíc čísel gaštanov, ktoré sú navzájom spojené zápalkou. Váš program má vypísať počet útvarov, ktoré Mišo vytvoril.

Príklad:

Vstup:
```
9
7
1 2
2 3
2 4
2 7
5 7
6 7
8 9
```

Výstup:
```
Mišo vytvoril 2 útvary.
```

(Vytvoril „koníka“ a „činku“.)

z2013. Zvianočnieva sa

Ako iste viete, prišla už jeseň a s ňou nastala pre mnohých z vás obrovská kopa problémov, ako napríklad škola. Ale asi len Paľko si všimol, že Vianoce sú už predo dvermi a začal si už zháňať vianočný stromček, aby si ho nemusel kupovať 25. decembra ako po minulé roky. Preto sa vybral do lesa a našiel si tú svoju najkrajšiu a najkošatejšiu jedličku, aká tam rástla a s pomocou susedovho traktora si ju priviezol domov.

Pri pokuse o prestrčenie jedličky cez dvere do obývačky však zistil, že jedlička je na to príliš široká a vysoká. Preto sa rozhodol odpíliť kúsok zdola tak, aby mu ostal stromček, ktorý už prejde dverami a je čo navyšší. Ale ako to urobiť? Preto sa obrátil na vás s prosbou, aby ste mu s tým pomohli.

Predpokladajme pre jednoduchosť, že stromček je dvojrozmerný a nemôžeme ho otáčať. Predstavte si ho ako zvislý kmeň (úsečku), z ktorého vyrastajú vodorovné konáre. Z jedného miesta na kmeni vždy vyrastá dvojica konárov rovnakej dĺžky – do každej strany jeden. Aby ten stromček nejako vyzeral, dĺžka nižšie položeného konára je vždy väčšia alebo rovná ako dĺžka vyššie položeného konára. Hrúbku konárov a kmeňa je možné zanedbať.

Úloha: Napíšte procedúru, ktorá dostane ako parametre popis dverí a stromčeka a zistí, či treba stromček rozpíliť, a ak áno, tak kde.

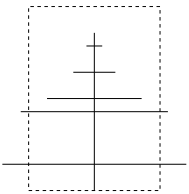
Presnejšie, parametre, ktoré vaša procedúra dostane, sú nasledovné: šírka dverí s, výška dverí v, výška stromčeka h, počet vetvení n a pole popisujúce jednotlivé vetvenia. Každý záznam v tomto poli obsahuje dve čísla: vzdialenosť od vrcholu stromčeka a dĺžku konárov, ktoré z dotyčného miesta vychádzajú. Môžete predpokladať, že toto pole je usporiadané vzostupne podľa vzdialenosti od vrchu stromčeka, a teda aj podľa dĺžky konárov.

Poznámka: Existuje riešenie, ktoré sa na väčšinu vetvení ani len nepozrie.

Príklad:

Vstup:
```
100 140 120 5
10 5
30 15
50 35
60 55
100 70
```

Výstup:
```
Stromček treba rozpíliť:
tesne nad 4. vetvou.
```



z2014. Žiarovky

Ako ste sa v prvej úlohe tejto série dozvedeli, hrozí, že Brezno zatopí voda. Požiarnici museli všetkých obyvateľov Brezna rýchlo evakuovať, aby sa neutopili. Preto ich ukryli do podzemného krytu. Taký kryt sa skladá z množstva na seba kolmých chodieb, ktoré sa navzájom križujú. Na každej križovatke je jedna žiarovka. Kryt si teda môžeme predstaviť ako maticu žiaroviek rozmerov $r \times s$, kde riadky a stĺpce matice sú chodby.

Na konci každej chodby je prepínač, ktorý prepína všetky žiarovky na križovatkách tejto chodby – t.j. zapaľuje všetky zhasnuté žiarovky a zároveň zhasína všetky zapálené

žiarovky. Keďže požiarnici nechcú, aby sa obyvatelia v panike pošliapali, chcú rozsvietiť všetky žiarovky. Vašou úlohou je zistiť, či sa to dá, a ak áno, tak na koľko najmenej prepnutí.

Úloha: Dané sú rozmery matice r a s a samotná matica popisujúca stav žiaroviek na začiatku – `0` znamená, že žiarovka je zhasnutá a `1` znamená, že žiarovka je zapálená. Vašou úlohou je vypísať najkratšiu postupnosť riadkov a stĺpcov matice, v ktorých keď postupne prepneme žiarovky, tak dostaneme na konci situáciu, kedy budú všetky žiarovky svietiť. Ak je takých postupností viac, vypíšte ľubovoľnú z nich. Ak neexistuje žiadna taká postupnosť, vypíšte správu, že úloha nemá riešenie.

Príklad:

Vstup:
```
4 6
1 0 1 1 1 0
1 0 1 1 1 0
0 1 0 0 0 1
1 0 1 1 1 0
```
Výstup:
```
2. stĺpec
3. riadok
6. stĺpec
```

Vstup:
```
3 5
1 0 1 1 1
0 1 0 1 1
1 0 1 1 1
```
Výstup:
```
Úloha nemá riešenie.
```

z2015. Zahltení prácou

Veru tak. V Softvérovom Družstve SoDr majú opäť frmol. Chystajú sa práve vypustiť do sveta novú verziu hry „Harry Potter a smradľavé slimáky“, v ktorej sa bude Harry nachádzať v akomsi bludisku v podobe záhrady či zámku a bude sa vyhýbať smradľavým slimákom a jesť fazuľky. Programátori už zvládli všetko okrem časti programu, ktorá bude do jednotlivých levelov generovať bludiská.

Niežeby sa členom SoDr nechcelo nič programovať, to sa nedá povedať. Akurát, momentálne riešia (dôležitejšie) problémy sveta a logicky sa takýmto niečím nemajú čas zaoberať. Preto pre zmenu oni požiadali o pomoc vás.

Úloha: Vašou úlohou je napísať program, ktorý pre zadané rozmery bludiska $r \times s$ vygeneruje mapu bludiska. Mapa sa skladá zo znakov „`x`“, „`.`“, „`H`“, „`*`“, pričom „`x`“ predstavuje stenu, „`.`“ chodbu, „`H`“ miesto, kde sa Harry ocitne na začiatku a „`*`“ miesto, kam sa má Harry dostať. V bludisku by mala navyše existovať aspoň jedna cesta z miesta označeného znakom „`H`“ do miesta označeného znakom „`*`“ prechádzajúca iba chodbami. Do zadaných rozmerov sa nepočíta „stena“ okolo celého bludiska. Ďalej môžete predpokladať, že $r, s \geq 5$. Snažte sa, aby váš program vygeneroval zakaždým čo najzaujímavejšie bludisko!

Príklad:

Vstup:
```
10 10
```
Výstup:
```
xxxxxxxxxxxx
x...x.x....x
x.x.x...xx.x
x.x.xxx.x..x
x.x...x....x
x.x.x.x.x..x
x.x.x.x.x..x
x.x.x.x.xx.x
x.x......x.x
x.xxxxxxxx.x
x...Hx*....x
xxxxxxxxxxxx
```
(Toto je len jedno možné bludisko.)

z2021. Zememerači na Kiribati

Ako všetci iste viete, Kiribati je veľmi veľké súostrovie. Je také veľké, že ani samotný kráľ nemá prehľad o tom, aké veľké sú jednotlivé ostrovy a ani koľko kilometrov morského pobrežia majú. No a keďže taký kráľ neznesie predstavu, že nepozná veľkosť svojho kráľovstva, rozhodol sa tam poslať zememeračov. Nanešťastie zistil, že v krajine žiadni zememerači nie sú. Preto mu musíte pomôcť vy.

Úloha: Mapa Kiribati je obdĺžnik rozmerov $r \times s$. Na vstupe sú dané tieto rozmery, následne samotná mapa a na záver súradnice $[p_r, p_s]$ jedného políčka na nej. Formát mapy: v i-tom riadku mapy je j-te číslo `0`, ak je na políčku $[i, j]$ voda a `1`, ak je tam súš. Na políčku $[p_r, p_s]$ je súš, na všetkých políčkach na okraji mapy je voda. Voda na mape sa dá rozdeliť na niekoľko súvislých vodných plôch. Tú z nich, ktorá obsahuje okraj mapy, nazveme more, ostatné vodné plochy sú jazerá.

Vašou úlohou je zistiť plochu ostrova, na ktorom políčko $[p_r, p_s]$ leží, a tiež dĺžku jeho morského pobrežia – teda počet políčok ostrova, ktoré susedia s morom.

Príklad:

Vstup:

```
10 16
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0
0 0 1 0 0 1 1 0 0 0 0 0 0 1 1 0
0 0 0 0 0 1 1 0 0 1 1 0 1 1 0 0
0 0 0 0 0 1 1 0 0 0 1 0 1 1 1 0
0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 0
0 0 1 1 0 0 0 1 1 1 1 1 1 0 0 0
0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 6
```

Výstup:

```
Plocha ostrova: 31
Dĺžka morského pobrežia: 23
```

(Všimnite si, že na našom ostrove je jazero, v ktorom je ďalší ostrov. Ak by sme sa pýtali na dĺžku morského pobrežia tohto malého ostrova, odpoveď by bola 0, pretože všade okolo neho je jazero, nie more.)

z2022. Z gaštanov II

Miško sa znova nudil na prednáškach, a tak sa rozhodol stavať ďalšie figúrky. Priniesol si hŕbu gaštanov, avšak zistil, že už nemá zápalky na ich spájanie. Tak sa rozhodol, že vyberie z predchádzajúcich figúrok tie zápalky, ktoré nie sú potrebné na to, aby držali figúrku dokopy. Pomôžte Miškovi zistiť, koľko najviac zápaliek môže odobrať bez toho, aby sa hociktorá z jeho figúriek rozpadla.

Úloha: Napíšte program, ktorý načíta počet gaštanov n (tie sú očíslované od 1 do n), počet zápaliek m a zoznam dvojíc čísel gaštanov, ktoré sú navzájom spojené zápalkou. Váš program má vypísať najväčší počet zápaliek, ktoré môže Miško zo svojich figúriek dokopy odobrať.

Príklad:

Vstup:

```
4
4
1 2   2 3   3 1   1 4
```

Výstup:

```
Miško môže odobrať 1 zápalku.
```

(A to 1–2, 2–3, alebo 3–1.)

z2023. Zvianočnieva sa II

Minulé kolo ste pomohli Paľkovi prepchať stromček do jeho obývačky, za čo je vám veľmi vďačný. No ale ihneď ako sa pokochal pohľadom na stromček zistil, že stromček mu zaberá polovicu obývačky. Tak si povedal, že stromček postaví ku stene. Nepomohlo. Vtedy si však uvedomil, že keď je stromček pri stene, aj tak z neho nie je vidieť zadnú časť. Keby z neho odpílil zadné vetvy, dal by sa stromček prisunúť bližšie k stene a vyzeral by stále rovnako.

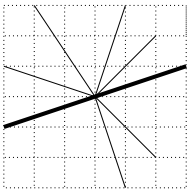
Preto sa rozhodol, že zo stromčeka ureže zadnú časť a stromček postaví ku stene tak, aby sa jeho kmeň dotýkal steny. Ale teraz nevie, ktorú polovicu stromčeka urezať, pretože by chcel, aby mu ostal čo najkrajší stromček, teda aby mu ostalo čo najviac konárov. S touto úlohou sa obrátil na svojich dobrých kamarátov – teda na vás.

Úloha: Na stromček sa budeme pozerať zhora. Zhora kmeň vyzerá ako bod, z ktorého vychádzajú do strán konáre (viď obrázok). Kmeň sa nachádza v začiatku súradnicovej sústavy. Konáre budú úsečky vychádzajúce z kmeňa, teda z bodu $[0,0]$ a budeme ich reprezentovať ich koncovým bodom. Stena bude priamka prechádzajúca začiatkom súradnicovej sústavy. Vašou úlohou je napísať program, ktorý na vstupe dostane počet konárov n a súradnice ich koncových bodov a nájde takú priamku prechádzajúcu začiatkom súradnicovej sústavy, ktorá konáre rozdelí na dve časti tak, aby v jednej časti ostal najväčší možný počet konárov. Ak existuje viac riešení, vypíšte ľubovoľné z nich. Konáre ležiace na priamke predstavujúcej stenu neodpílime.

Príklad:

Vstup:

```
6
 1  3
-3  1
-2  3
 2  2
 2 -2
 1 -3
```



Výstup:

```
Stena prechádza bodom [3, 1].
Na stromčeku ostanú 4 vetvy.
```

(Toto je jedno z viacerých možných riešení.)

z2024. Žužu žuje žuvačku

Žužu opravoval riešenia KSP. Keďže chcel dať všetkým veľa bodov, bol si kúpiť cukríky. (Za každý cukrík v jeho ústach bol totiž bod navyše.) Prišiel do bufetu. Na poličke uvidel veľký výber cukríkov. Mali tam cukríky za jednu korunu, za dve koruny, za tri,... Keď sa Žužu poobzeral, zistil, že ceny cukríkov sú vlastne práve všetky celé kladné čísla, za každú cenu predávajú práve jeden druh cukríkov. Rozhodol sa, že za všetky svoje úspory, ktoré činili práve n korún, si nakúpi cukríky. Potom dlho rozmýšľal, aké možnosti vlastne má. A aby nabudúce nerozmýšľal tak dlho, potrebuje ich zoznam.

Úloha: Na vstupe je zadané číslo n. Vašou úlohou je vypísať všetky možnosti cien cukríkov, ktoré si môže Žužu za n korún nakúpiť. Každú možnosť vypíšte ako postupnosť cien jednotlivých cukríkov. Celkový súčet týchto cien má byť presne n.

Každú možnosť vypíšte práve raz. Samozrejme, je jedno, v akom poradí si Žužu cukríky kúpi, teda možnosti líšiace sa iba zmenou poradia cien cukríkov považujeme za rovnaké.

Príklad:

Vstup:

```
5
```

Výstup:

```
5
4 1
3 1 1
2 1 1 1
1 1 1 1 1
3 2
2 2 1
```

z2025. Zárm Oded

Programátori v Softvérovom Družstve sa rozhodli, že pred Vianocami ešte predsa niečo naprogramujú a zarobia si tak na darčeky. Zákazník, ktorý sa ale odmietol predstaviť skutočným menom a predstavil sa ako Zárm Oded, chcel iba jednoduchý program na evidenciu adries a zásielok, ktoré sa na ne majú doručiť. Vraj to má niečo s vianočnými darčekmi. V SoDr sa potešili a rýchlo ho naprogramovali, však to nebolo nič ťažké. A už-už

to aj chceli poslať, keď si uvedomili, že im chýba tá najdôležitejšia vec – SplashScreen. To je taký ten otravný obrázoček, ktorý sa objaví na obrazovke, kým sa program spúšťa.

V SoDr sa rozhodli, že vzhľadom na zameranie programu bude najlepším obrázkom vianočný stromček. To ale nie je až také jednoduché. Zákazníkove počítače totiž fungujú len v textovom móde a ešte k tomu má každý inú veľkosť obrazovky – teda počet znakov v riadku a počet riadkov. Programátori teraz stoja pred problémom, ako stromček v textovom móde nakresliť pre rôzne veľkosti obrazovky.

Úloha: Napíšte program, ktorý načíta šírku s a výšku v obrazovky a počet vetiev p a nakreslí čo najkrajší vianočný stromček. Vianočný stromček musí pozostávať zo zvislého kmeňa a z p dvojíc vodorovných alebo šikmých vetiev, ktoré z neho vyrastajú.

Príklad:

Vstup:

```
20 10 3
```

Výstup:

```
          v
       -_  I  _-
    _ o --I-- o _
     --_  I  _--
--_ o  --I--  o _--
o ----_  I  _---- o
       ---I---
          I
         fvl
```

z2031. Zúfalý poštár

Na jednom zabudnutom ostrove kdesi uprostred oceánu žijú v malých mestečkách trpaslíci. Majú sa dobre, skoro nič im ku šťastiu nechýba. Až na jednu maličkosť: poštu. Teda, nie že by nikdy nemali poštu, ale holubom, ktorých používali, sa zachcelo slobody a nechali trpaslíkov osamote. Preto sa Veľká Rada Trpaslíkov rozhodla postaviť v jednom mestečku centrálnu poštu a poveriť jedného trpaslíka roznášaním pošty po celom ostrove.

Mestečká na ostrove sú všetky vzájomne pospájané chodníkmi, ale medzi ľubovolnými dvoma vedie len jedna cesta (nie nutne priama, môže viesť cez niekoľko iných miest). Chodníky sú všetky približne rovnako dlhé, preto za vzdialenosť dvoch miest považujeme priamo počet chodníkov, ktoré tvoria cestu medzi nimi. Poštár je hrozne lenivý, preto sa rozhodol, že centrálnu poštu treba postaviť v takom meste, z ktorého je cesta do najvzdialenejšieho mesta najkratšia možná. Ale ktoré mesto je to správne?

Úloha: Pomôžte poštárovi a nájdite mu mesto, z ktorého je cesta do najvzdialenejšieho iného mesta najkratšia možná. (Teda keby sme pre každé mesto spočítali vzdialenosti do všetkých ostatných a zapamätali si z nich tú najväčšiu, hľadané mesto je to, pre ktoré si pamätáme najmenšiu hodnotu.) V prípade, že viacero miest spĺňa túto podmienku, vašou úlohou je nájsť ich všetky.

Ostrov má n miest očíslovaných $1, \ldots, n$ a presne $n-1$ chodníkov, pričom každý chodník je zadaný číslami miest, ktoré spája. (Uvedomte si, že zo zadania vyplýva, že chodníkov musí byť presne $n-1$.)

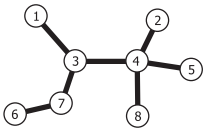
Príklad:

Vstup:

```
8
1 3
2 4
6 7
3 4
8 4
4 5
3 7
```

Výstup:

```
Poštu treba postaviť v meste 3.
```



z2032. Záh(r)adná párty

Tma... Nič, len tma... Zrazu silná žiara vychádzajúca odniekiaľ z výšky... A k tomu ten zláštny zvuk... ako keby okolo lietali státisíce netopierov... Z hradu sa vynorila postava, zahalená v tmavom plášti, a v rukách držala...

„Vitajte!“ ozvalo sa z miesta, kde ešte pred chvíľkou bola tá tajomná postava. Teraz tam stál Rúfus, dvorný sluha Jej veličenstva Kráľovnej, a v rukách držal zaprášenú drevenú truhlicu. Položil ju na zem a otvoril. Na prekvapenie všetkých prítomných z truhlice začali vyliezať rôznofarební škriatkovia, rôznofarební jednorožci a jednofarební pavúci. Škriatkovia majú svetlé farby: jeden je žltý, tamten je oranžový, tam stojí skupinka svetlomodrých, a ešte veľa iných farieb... Zato jednorožci sú tmavší – tam zelení, tam hnedí, alebo fialoví... A pavúci – tí sú všetci šedí.

Ako na povel sa všetci – pavúci, škriatkovia a jednorožci – zoradili v náhodnom poradí do jedného dlhého radu na hraciu plochu, ktorá sa z ničoho nič objavila v záhrade. Hracia plocha vyzerala ako dlhý pásik štvorčekovaného papiera, v každom štvorčeku stála jedna bytosť. „Komu sa podarí zoradiť tieto bytosti na hracej ploche tak, aby na začiatku stáli vedľa seba všetci škriatkovia, potom všetci pavúci a na konci všetci jednorožci, získa poklad ukrytý niekde v hlbinách hradu!“ zvolal Rúfus.

Úloha: Skúste vyriešiť túto úlohu. Škriatkov budeme označovať písmenkami „`A`“ až „`Z`“, každú farbu jedným písmenkom. Podobne jednorožcov budeme označovať znakmi „`0`“ až „`9`“, a nakoniec pavúkov budeme označovať bodkami. Rozostavenie bytostí na hracej ploche je dané počtom bytostí n a poľom znakov dĺžky n, v ktorom je pre každé políčko hracej plochy znak predstavujúci bytosť, ktorá na ňom stojí. Váš program má toto pole preusporiadať tak, aby na jeho začiatku boli všetky písmenká, potom všetky bodky a na konci všetky čísla.

Poznámka: Pokúste sa pri riešení *nepoužiť* pomocné pole.

Vstup:

```
13
7C.1A.F4.X.5A
```

Výstup:

```
AXCAF....1475
```

(Sú aj iné správne výstupy.)

z2033. Zamotaný Santo

Pri potulkách v okolí Vyšnej Klondiky s virgulou v ruke narazil Santo na zlatú žilu. Rozhodol sa, že parcelu so zlatou žilou od mesta odkúpi a rozbehne na nej výnosný zlatokopecký biznis. Na to, aby parcelu mohol odkúpiť, treba ju najskôr presne vymerať, t.j. zatĺcť do jej rohov drevené kolíky a susedné dvojice kolíkov pospájať špagátom. Navyše parcela musí byť tvaru konvexného mnoholníka, iné tvary totiž mestskí úradníci nemajú radi.

Santo teda zatĺkol do zeme niekoľko kolíkov. Potom natiahol špagát medzi prvým a druhým kolíkom, druhým a tretím, atď. a ešte medzi posledným a prvým. Santo je však veľký moták, preto nevie, či špagát teraz tvorí obvod konvexného mnohouholníka.

Úloha: Kolíky chápeme ako body v rovine a kusy špagátu ako úsečky. Daný je počet kolíkov n a pre každý kolík jeho súradnice x_i, y_i. Špagát tvorí uzavretú lomenú čiaru, ktorá spája kolíky v poradí, v akom sú dané na vstupe (aj posledný s prvým).

Zistite, či tento špagát tvorí hranicu konvexného n-uholníka. Aby špagát tvoril obvod mnohouholníka (nie nutne konvexného), nesmie sám seba pretínať, ani sa len sám seba dotýkať. Mnohouholník je konvexný vtedy, ak pre ľubovoľné dva jeho body platí, že žiaden bod úsečky, ktorá ich spája, neleží mimo mnohouholníka.

Príklad:

Vstup:
```
8
0  0
2  0
2  1
1  1
1 -1
3 -1
3  2
0  2
```

Výstup:
```
Nie.
```

(Špagáty sa pretínajú, takže to nie je ani len mnohouholník.)

z2034. Zaujímavý mrakodrap

V Novom Yorku stojí vysoký mrakodrap Empire State Building. Ten mrakodrap má veľa poschodí. Ba ešte viac – pre naše účely možno smelo povedať, že ich je nekonečne veľa. Poschodia sú očíslované nezápornými celými číslami.

V mrakodrape je rýchlovýťah, v ktorom sú dve tlačidlá a na každom z nich je napísané celé číslo. Jedno z týchto čísel je kladné, druhé je záporné. Keď človek stlačí tlačidlo, pohne sa výťah o príslušný počet poschodí, ak môže. (Dohora, ak stlačil kladné číslo, dodola, ak stlačil záporné. Ak sa nedá pohnúť o želaný počet poschodí dodola, výťah nespraví nič.)

Ak sa chce človek dostať z prízemia (t.j. poschodia 0) na nejaké iné poschodie, postupne stláča tlačidlá, až kým sa nedostane tam, kam chce. Na niektoré poschodia sa však nemusí dať dostať. Vtedy človek musí vyšľapať alebo zošľapať niekoľko poschodí po svojich.

Úloha: Dané sú tri celé čísla: kladné číslo a, záporné číslo b a nezáporné číslo p, pričom a a b sú čísla na tlačidlách a p je poschodie, na ktoré sa chceme dostať.

Zistite, koľkokrát a v akom poradí treba stlačiť ktoré tlačidlo tak, aby sme museli ísť po schodoch (či už hore alebo dole) čo najmenej. Ak existuje viac riešení, pre ktoré je počet pešo prejdených poschodí najmenší možný, vypíšte ľubovoľné z nich.

Príklad:

Vstup:
```
+14 -10
5
```

Výstup:
```
Prvé tlačidlo stlačiť 4-krát.
Druhé tlačidlo stlačiť 1-krát.
Prvé tlačidlo stlačiť 2-krát.
Druhé tlačidlo stlačiť 7-krát.
Vyšľapať 1 poschodie.
```

z2035. Zaneprázdnené múzy

Život múz nie je jednoduchý. Čo chvíľu ich vzýva nejaký básnik alebo muž-romantik, a či chcú, či nechcú, musia za nich vymýšľať básne. Už ich to pomaly prestáva baviť. Vždy sa natrápia a nakoniec žiadna vďaka. Básnik tak nanajvýš spomenie, že ho kopla múza. Aj keď múzy pochádzajú zo staroveku, rozhodli sa, že vyskúšajú moderné prostriedky, a kúpili si počítač. Teraz sedia a rozmýšľajú, ako napísať taký program, ktorý by vymýšľal básne za ne. Príliš im to nejde, vedia totiž rozmýšľať len vo veršoch, a uznajte, kto kedy videl rýmujúci sa program!

Úloha: Vašou úlohou je napísať program, ktorý na vstupe dostane počet veršov n a na výstupe vyrobí básničku s n veršami. Váš program by mal pri každom spustení vypísať inú (náhodnú) básničku. Bolo by dobré, keby sa programu dal na vstupe zadať aj druh rýmu, ktorý má v básni použiť.

Príklad:

Vstup:
```
2
rým: AA
```

Výstup:
```
Vševedko išiel cez riečku
preskakoval tam ovečku.
```

VSTUP:
```
6
rým: AABCCB
```

VÝSTUP:
```
„Kam, Čiapočka malá, kam?"
„Chodím horou sem a tam.
 A ty vĺčko čo tu chceš?"
„Zháňam pod zub trošku trávy,
 od trávy mi dobre trávi."
„Vĺčko, vĺčko, neklameš?"
```

POZNÁMKA: Prvá básnička: Nikolaj Nosov – Nevedkove dobrodružstvá, druhá básnička: František Hrubín – Červená Čiapočka.

z2041. Zas.... chodník

Neviem, či sa vám to už stalo, ale ja práve stojím na začiatku chodníka, kade sa prehnalo stádo kráv. Zjavne sa im zdal nanajvýš vhodným miestom, kde môžu odľahčiť svojmu komplikovanému tráviacemu ústrojenstvu. Prejsť po takomto chodníku nie je nič príjemné, preto som si vymyslel zábavku.

Chcem prekráčať ponad lajná tak, aby som každým krokom prekročil práve jedno. A aby som nemal dilemu, kde mám presne stúpiť, pôjdem stále rovno (tým istým smerom ako doteraz) a budem robiť všetky kroky rovnako dlhé.

ÚLOHA: Predstavme si, že náš chodník tvorí kladnú poloos osi x, ja stojím v nule. Na chodníku leží n lajen, každé zaberá nejaký neprázdny otvorený interval. Pre každé z nich sú dané dve čísla, určujúce kde začína a kde končí. Lajná sa navzájom neprekrývajú a zadané sú v poradí odo mňa, t.j. prvé je najbližšie. Každým krokom chcem prekročiť práve jedno lajno a nechcem do žiadneho stúpiť. Vašou úlohou je nájsť takú dĺžku môjho kroku, aby sa mi ich podarilo spraviť čo najviac.

PRÍKLAD:

VSTUP:
```
4
2.0 4.0
6.0 6.7
8.8 11.3
12.4 17.9
```

VÝSTUP:
```
Dĺžka kroku: 4.1
```

(Prvé tri kroky sú OK, štvrtým sa mi už nepodarí štvrté lajno prekročiť. Žiadna iná dĺžka kroku mi neumožní spraviť viac ako tri kroky.)

z2042. Závažný problém

Ako isto viete, prednedávnom sa skončilo IPSC. IPSC je internetová súťaž tímov v riešení problémov. Je zadaných niekoľko úloh a ku každej úlohe existujú dva vstupy – ľahký a ťažký. Tímy musia vyriešiť zadaný problém a odovzdať korektné výstupy k daným vstupom. Tie potom špeciálny program, testovač, skontroluje a vytvorí „log". Log je súbor, v ktorom sú uložené údaje o jednotlivých pokusoch tímov o riešenie daného problému. Ďalší program si z logu zistí, aké problémy vyriešili jednotlivé tímy a vygeneruje výsledkovú listinu. Ale tu nastal závažný problém – tento program bol príliš pomalý. A tak by ste nám mohli napísať rýchlejší.

Každý riadok logu obsahuje informácie o jednom pokuse o odovzdanie výstupu, tzv. submite. Riadky sú samozrejme usporiadané vzostupne podľa času odovzdania. O každom submite sú v logu štyri údaje – čas, kedy bol daný submit odovzdaný, meno tímu, ktorý daný submit odovzdal, meno úlohy a výstup z testovača. Čas odovzdania úlohy je celé číslo od 1 do 2 000 a je to počet minút od začiatku súťaže. Meno tímu je ľubovoľný reťazec tvorený maximálne 20 písmenami anglickej abecedy bez medzier. Meno úlohy sa skladá z jedného znaku a jedného čísla. Znak je písmeno od „`A`" po „`H`", ktoré určuje problém; číslo môže byt buď „`1`" alebo „`2`" (ľahký alebo ťažký vstup). Napríklad „`C1`" znamená, že ide o výstup pre ľahký vstup problému C. Výstup z testovača je ľubovoľný reťazec dĺžky najviac

30 začínajúci písmenom. Ak je to reťazec „OK“, znamená to, že uvedený výstup je správny, inak je nesprávny a tento reťazec udáva druh chyby.

Bodovanie funguje nasledovne: Za každý správne vyriešený ľahký vstup získa tím jeden bod, za každý správne vyriešený ťažký vstup získa tím dva body. Ak má viacero tímov rovnaký počet bodov, o ich poradí rozhodnú trestné minúty. Tím s menším počtom trestných minút sa umiestni na lepšom mieste ako tím s väčším počtom trestných minút. Trestné minúty sa získavajú nasledovne: Za každý správne vyriešený ľahký vstup sa tímu k doterajším trestným minútam pripočíta čas, kedy tím odovzdal dotyčný submit, a ešte 10 minút za každý predchádzajúci nesprávny submit pre tento vstup. Za každý správne vyriešený ťažký vstup sa tímu k doterajším trestným minútam pripočíta čas, kedy tím odovzdal dotyčný submit, a ešte 20 minút za každý predchádzajúci nesprávny submit pre tento vstup. (Ak sa tímu nepodarí správne vyriešiť nejaký vstup, tak zaň nedostane žiadne trestné minúty, aj keby mal nejaké nesprávne submity.) Ak sa tím pokúsi znovu vyriešiť už ním vyriešený problém, tak sa tento submit berie ako nesprávny, ale tím zaň nedostane žiadne trestné minúty navyše. Ak má viacero tímov rovnaký počet bodov aj trestných minút, tak sa umiestnia na rovnakom mieste.

Úloha: Napíšte program, ktorý z daného logu vygeneruje výsledkovú listinu. Tímy musia byť vypísané usporiadané od prvého po posledný. Pre každý tím vypíšte jeho umiestnenie, počet bodov, celkový počet trestných minút a mená vstupov, ktoré vyriešil.

Príklad:

Vstup:

```
100 MojTim A2 Zle
137 TvojTim A1 OK
189 MojTim A2 OK
290 TvojTim C1 Malo cokolady
591 MojTim A2 Uz vyriesene
630 TvojTim C1 OK
780 JehoTim F1 zle
800 JejTim C2 Privela cokolady
```

Výstup:

```
Por. Tim      body cas priklady
  1. MojTim    2   209  A2
  2. TvojTim   2   777  A1 C1
  3. JehoTim   0     0
     JejTim    0     0
```

z2043. Zase neporiadok

Rodičia si stále myslia, že deti majú vo svojich veciach neporiadok. V podobnej situácii je aj Riško. Napríklad aj teraz mu na stole leží kopa nesmierne dôležitých krabíc. Ale vysvetlite mame, že sú dôležité! Márna snaha, musia preč. Lenže to nie je také jednoduché. Krabice sú ťažké a teda sa dajú premiestňovať iba po jednej. Navyše tá kopa krabíc je vlastne veža z rôzne veľkých krabíc, pričom vždy je menšia krabica na väčšej. Toto usporiadanie je dôležité, lebo keby sa položila väčšia krabica na menšiu, tú menšiu by to rozpučilo. Izba je takmer plná haraburdia, sú tam len dve voľné miesta – v rohu izby a v skrini. Na každé z nich by sa veža z krabíc tak akurát vošla. Nuž a na jedno z nich (je jedno na ktoré) musí Riško vežu krabíc presunúť.

Časom zistil, že najrýchlejšie sa to dá takto: Upratujeme v ťahoch. Keď robíme ťah s nepárnym poradovým číslom, presunieme najmenšiu krabicu, a to nasledovne: Ak bola na stole, dáme ju na vrch veže v rohu izby, ak bola v rohu izby, dáme ju do skrine a ak bola v skrini, pôjde na stôl. A keď robíme párny ťah, presunieme krabicu, ktorá nie je najmenšia. (Rozmyslite si, že to sa vždy bude dať spraviť len jedným spôsobom.)

Aby však Riško neprehadzoval krabice nadarmo, potreboval by si overiť, či tento spôsob funguje. Potreboval by program, ktorý mu pre dané t povie, ako budú vyzerať jeho tri veže z krabíc (kopa na stole, v rohu izby a v skrini) po vykonaní t ťahov.

Úloha: Dané je číslo n, udávajúce počet krabíc, ktoré treba upratať, a počet ťahov t, ktoré Riško spraví. Krabice sú očíslované od 1 po n podľa veľkosti (1 je najmenšia). Určite, ako budú vyzerať veže po vykonaní t ťahov podľa vyššie popísaných pravidiel.

Príklad:

Vstup:	Výstup:

`3 4`	`1. veža: prázdna`
	`2. veža: 3`
	`3. veža: 2 1`

(Veže vypisujeme zdola hore.)

z2044. Zo sna hovoriaci čarodej

Život je plný stereotypov. Jedným z najklasickejších je starý čarodej. Dňom i nocou sa venuje výskumu mágie, ktorá je nad úrovňou nás – prostých ľudí. Je v podstate milý a príjemný, kým mu neprekážate, nič po ňom nechcete a nevyrušujete ho od práce. V opačnom prípade vie byť naozaj veľmi nepríjemný.

Žiť v jednom dome s takýmto čarodejom nie je žiadna radosť. Vždy keď si sadáte, musíte sa poriadne pozrieť kam, lebo napríklad prisadnutý bazilišok vie byť veľmi nepríjemný a ak rozsadnete pavúka, čarodeja naštvete, lebo určite práve potreboval nejakého živého. Po dome sa povaľuje neuveriteľná kopa rôznych kníh, zvitkov, fľaštičiek s rôznofarbnými tekutinami a ponožiek. A to už nehovorím o tom, čo sa stane, ak niektorú z fľaštičiek rozbijete alebo božechráň sa z nej napijete.

Malý Garak, Belsambarov učeň, dnes oslavuje svoje desiate narodeniny. Keď zoberieme do úvahy všetky spomínané nepríjemnosti, ktoré sa môžu človeku v takom dome pritrafiť, je to až prekvapivé. Náladu mu však dnes kazí nová nepríjemnosť, ktorá ho čaká. Belsambar totiž začal rozprávať zo sna a často sa mu podarí vysloviť nejaké zaklínadlo. Garak by potreboval program, ktorý bude vedieť rozpoznať zaklínadlo od obyčajného mrmlania. Prezradil vám, že slovo je zaklínadlo práve vtedy, keď spĺňa nasledujúce podmienky:

– jeho dĺžka je deliteľná magickým číslom 3
– obsahuje ako podreťazec kúzelnú formulu „`abraka`“
– neobsahuje `f`, `c`, ani dve samohlásky za sebou

Úloha: Napíšte program, ktorý Garakovi pomôže rozhodnúť, či je dané slovo zaklínadlo. Program musí byť samozrejme čo najrýchlejší, ale najdôležitejšie je aby potreboval *čo najmenej pamäte* (a dal sa spustiť na Garakovom prastarom počítači).

Príklad:

Vstup:	Výstup:
`abrakadabra`	`NIE`
`bezvysypattiesmeti`	`NIE`
`vrakadrakabrakarak`	`ANO`

z2045. ZJEDZ OBED HARRY

Takýto čudný odkaz si našiel Harry na posteli vo svojej komôrke. Najprv si myslel, že sa ho spriatelený domáci škriatok snaží uprozorniť na jeho nezdravé stravovacie návyky. Potom si však uvedomil, že u nich žiadny škriatok nebýva. Navyše, keď sa lepšie prizrel, všimol si, že medzi prvými dvoma slovami je akýsi malý krížik a medzi druhým a tretím sú zase dve krátke čiarky (aby ste si to vedeli predstaviť, vyzeralo to takto: `ZJEDZ` + `OBED` = `HARRY`). Vtom mu napadlo – veď je to list od Hermiony! Ešte v škole sa dohodli, že všetky správy, ktoré si pošlú, budú pre istotu magicky šifrovať. Čudný nápis je len zámok a skutočná správa sa ukáže iba tomu, kto ho dokáže odomknúť magickým kľúčom. Ten vyzerá ako zoznam dvojíc písmenko=číslica, pričom žiadne písmenko ani číslica sa v kľúči neopakuje. Odomykanie vyzerá tak, že sa v zámku písmenká nahradia podľa kľúča číslicami. Zámok sa otvorí, ak po tomto nahradení výsledná rovnosť platí.

Harry tento zámok hravo otvoril. Teraz však stojí pred ťažším problémom – potrebuje Hermione napísať odpoveď a ako na potvoru nevie vymyslieť žiadny magický zámok. Pomôžete mu?

Úloha: Vašou úlohou je napísať program, ktorý na vstupe nedostane vôbec nič a na výstup vypíše nejaký zámok. Zámok sa skladá z niekoľkých zmysluplných slov oddelených matematickými operátormi „+“, „−“, „×“, „/“ a „=“. Samozrejme je nevyhnutné, aby k tomuto zámku existoval nejaký kľúč, ktorý ho odomkne. Na druhej strane, správnych kľúčov by malo existovať čo najmenej, ideálne jediný.

Príklad:

Vstup:

```
ZJEDZ + OBED = HARRY              (A=0 E=1 Y=2 O=3 D=4 B=5 R=6 J=7 Z=8 H=9)
87148 + 3514 = 90662
BABKA + K + BABCE + BUDU = KAPCE  (B=1 P=2 K=3 E=4 D=5 C=7 A=8 U=9)
18138 + 3 + 18174 + 1959 = 38274
AKO + SA + DO + HORY = VOLA       (R=0 D=1 H=2 V=3 S=4 L=5 Y=6 O=7 K=8 A=9)
987 + 49 + 17 + 2706 = 3759
TAK + SA + Z + HORY = OZYVA       (O=0 Y=1 V=2 H=3 Z=4 K=5 R=6 S=7 A=8 T=9)
985 + 49 + 4 + 3061 = 04128
DAJ * MI = JEST                   (A=0 M=1 D=2 J=3 S=5 T=7 E=8 I=9)
203 * 19 = 3857
HACK * THE = PLANET               (N=0 H=1 A=2 L=3 T=4 P=5 E=6 C=7 K=9)
1279 * 416 = 532064
```

Poznámka: Neprekáža nám, ak v riešení budú niektoré čísla začínať nulou.

2111. O slovných rovniciach

Keďže Kleofáš je šikovný žiačik a navyše sa minulý rok v škole naučil písať, jeho mamička si myslela, že mu urobí radosť, keď mu na narodeniny namiesto starého bicykla od susedov kúpi novú zbierku takzvaných slovných rovníc. Hm, tak nie... :-)

Ľudia robia chyby, tak to už chodí. Výsledkom sú potom smutní Kleofášovia ako sedia za stolom, pred sebou akási zbierka, hrubý zošit otvorený na prvej strane a pred sebou okno s výhľadom na bicyklujúcich sa kamarátov. Ešteže sú na svete starší bratia, z ktorých niektorí riešia KSP...

Úloha: Slovná rovnica sa podobá obyčajnej lineárnej rovnici s jednou premennou. Má tvar $u = v$, kde u a v sú slová zložené z písmen abecedy a slovnej premennej X, pričom počet výskytov X je na každej strane rovnice iný.

Riešenie slovnej rovnice (ak existuje) je také slovo w, že keď ním nahradíme všetky výskyty premennej X na ľavej aj pravej strane, prečítame na oboch stranách rovnaké slovo.

Vašou úlohou je napísať program, ktorý na vstupe dostane slovnú rovnicu a vypíše jej jedno riešenie (prípadne povie, že daná rovnica nemá riešenie).

Príklad:

Vstup:

```
abacXba = XcabX
abXd = aXcX
```

Výstup:

```
X = aba
Rovnica nemá riešenie.
```

2112. O trpaslíčom probléme

Ako je široko-ďaleko známe, v každom domčeku v trpasličej dedine bývajú nejakí trpaslíci. Domčeky sú pospájané cestičkami, to aby sa mohli trpaslíci navštevovať. Po cestičkách sa dá (aj keď nie nutne priamo) dostať z ľubovoľného domčeka do ľubovoľného iného. Ale aby trpaslíci nezabili celú jeseň odhŕňaním lístia, medzi každými dvoma domčekmi sa dá prejsť len presne jedným spôsobom.

Trpaslík Hubert je podnikavý typ trpaslíka, preto si už dávnejšie prerobil časť domčeka na obchod. Viac obchodov by sa v dedine neuživilo, a tak chodia všetci trpaslíci nakupovať výlučne k Hubertovi. Problémy však vznikli potom, ako sa do módy dostali motorové trojkolky. Každý trpaslík si jednu zaobstaral (pre Huberta to boli zlaté časy) a aby len tak neležala v garáži, každý deň sa na nej vybral do obchodu. Mamky trpaslice robili rodinné nákupy, oteckovia trpaslíci si ráno zašli po noviny, trpaslíčatá ozlomkrky uháňali po sladkosti, len čo si dopísali úlohy...

Používať motorovú trojkolku okrem iného znamená aj znečisťovať ovzdušie. Čím ďalej zájdete, tým viac znečistíte ovzdušie. Presnejšie, n-členná rodina bývajúca d metrov od obchodu vypustí do ovzdušia $47 \cdot n \cdot d$ centimetrov kubických škodlivých plynov (údaje sú uvedené na obale od trojkolky). Starosta dediny preto z obáv o čistý vzduch rozhodol, že obchod už asi nebude u Huberta doma, ale bude v takom domčeku, pre ktorý bude celkové denné znečistenie ovzdušia obyvateľmi dediny najmenšie možné. Úloha sa však zdá byť nad starostove sily, preto mu radšej pomôžte.

Úloha: Na vstupe je popísaná trpasličia dedina. Na prvom riadku je celé číslo n, ktoré určuje počet domčekov (označených číslami od 1 po n) v dedine. Potom nasleduje n čísel $a_1, \ldots, a_n$, kde a_i je počet trpaslíkov v i-tom domčeku. Ďalej nasleduje $n-1$ riadkov, v i-tom z nich tri čísla x_i, y_i, d_i, znamenajúce, že medzi domčekmi x_i a y_i vedie priama cesta dĺžky d_i. (Rozmyslite si, že cestičiek je v dedine nutne práve $n-1$, ani viac, ani menej.) Váš program by mal vypísať číslo domčeka, v ktorom by mal byť obchod. Ak je viac optimálnych riešení, vypíšte ľubovoľné z nich.

Príklad:

Vstup:

```
5
2 7 4 3 5
1 3 10
1 4 5
1 5 7
2 4 1
```

Výstup:

```
Obchod treba dať do domčeka 1.
```

2113. Ohne druidov

Jožko znova vyletel z analýzy. Kráčal zahmlenou Bratislavou a bol nekonečne nešťastný. Zrazu sa zem zdvihla a udrela ho do čela. Teda ono to bolo skôr naopak: Jožko si vo svojom žiali nevšimol jednu veľmi pevne zakorenenú jahodu, čo rástla pri matfyze. Keď sa prebral, okolo neho neboli budovy milovanej školy, ale stromy, tráva a nekonečná divočina. Na sebe mal rytierske brnenie a v ruke meč. Neveriacky naň hľadel. Po čase prišiel na to, že sa nachádza na území kráľa Artuša. Potreboval sa však dostať späť na opravný termín z analýzy. Bezradne sa pohyboval lesom a premýšľal.

Keď zdvihol pohľad, uvidel chalupu a pri nej starca. Mal dlhú bradu a na nechtoch pleseň. „Keby to tak bol druid...“ pomyslel si Jožko. Druidi boli mocní zaklínači, vedeli komunikovať s duchmi sveta a keby zapálili svoj čarovný oheň a dali sily dokopy, vedeli by napríklad aj dostať Jožka späť do Bratislavy. Pomaly sa mu začala vracať nádej. Zapísal si polohu domu. Obehol znova celý les dookola a našiel ešte ďalších pár chalúp a starcov. Teraz ostáva zistiť, či sa nikde nezmýlil.

Druidi počas zaklínania stoja vo vrcholoch pravidelného $2n$-uholníka, v ktorého strede horí oheň. Keď skončia zaklínanie, rozídu sa do svojich domov. Pritom sa ale každý druid pohybuje len po polpriamke, ktorá vedie z ohniska cez miesto, kde stál počas zaklínania. Jožko by potreboval zistiť, či domce, ktoré našiel, môžu byť práve všetky domy druidov. Lenže analýzu nespravil, a tak analyzovať nevie...

Úloha: Máme číslo n a súradnice $2n$ bodov. Zistite, či mohli tieto body vzniknúť posunutím vrcholov vhodného pravidelného $2n$-uholníka po polpriamkach od jeho stredu. Body nemusia byť zadané v žiadnom konkrétnom poradí.

Príklad:

Vstup:

```
2
6 3
6 9
3 5
11 5
```

Výstup:

```
Áno.
```

Pôvodný $2n$-uholník, teda štvorec, mohol mať napríklad vrcholy (6,6), (7,5), (6,4) a (5,5).

2114. O kanárikovi

Dávidko má veľmi rád kanáriky. Napríklad na splave o nich dokáže spievať celé noci. Preto keď sa blížili jeho narodeniny, rodičia ani dlho nerozmýšľali, čo mu dať – dostal kanárika. Dávidko sa mu samozrejme veľmi potešil, urobil mu klietku, nakŕmil ho... Po čase ho však prestalo baviť počúvať stále to jednotvárne čvirikanie, a tak sa rozhodol, že ho naučí rozprávať. Ako správny programátor si naprogramoval generátor náhodných reťazcov, ktoré potom kanárika učil. Aby mal dôkaz pre kamarátov, nahrával si rozprávajúceho kanárika na CD. Čoskoro si všimol, že príliš dlhé reťazce si kanárik nedokáže zapamätať. Chcel zistiť, koľko najviac znakov kanárik zvládne, ale nejako sa pri tom zamotal – kanárik totiž pri nahrávaní opakoval reťazec stále dokola, a tak z nahratého CD nie je jasné, z koľkých písmen sa reťazec skladá. Nám sa už ale začal školský rok a Dávidko nejako nestíha, preto potrebuje vašu pomoc.

Úloha: Vašou úlohou je napísať program, ktorý z daného reťazca vypočíta dĺžku jeho najkratšej periódy. Perióda reťazca je také číslo p, že pre všetky i sú i-ty a $(i+p)$-ty znak reťazca rovnaké (ak existujú).

Príklad:

Vstup:	Výstup:
`cvirikcvirikcvirikcvirikcvir`	`6`

2115. O mravcoch I

Mravce dostali na narodeniny kráľovnej počítače. Každý mravec dostal svoj vlastný maličký počítač. Samozrejme, okamžite ich zosieťovali. Keď ich prestalo baviť hrať sieťové hry, rozhodli sa, že si niečo naprogramujú. Programovať však vedel len Ferdo, preto kráľovná rozhodla, že Ferdo napíše program, ten nahrajú na niekoľko počítačov a naraz na všetkých ho spustia.

Vašou úlohou bude samozrejme pomáhať Ferdovi, preto si teraz bližšie popíšeme, ako bude vyzerať programovací jazyk, v ktorom sa programujú mravčie počítače. Program budeme písať v jazyku podobnom jazyku Pascal. Premenné použité v programe budú lokálne, t.j. každý počítač bude mať vlastné premenné. Jedinou výnimkou je globálne pole celých čísel `Mem`[]. Toto pole je indexované od 0 a je neobmedzene veľké. Na začiatku je v ňom (v niekoľkých prvých políčkach) uložený vstup úlohy, ostatné jeho políčka obsahujú nuly. Na konci v ňom má ostať uložený výsledok úlohy.

Programy budú spustené naraz na všetkých počítačoch. Počítače budú očíslované od 1 do k. Výpočet bude prebiehať v taktoch. Všetky počítače sú rovnako rýchle, v každom takte každý z nich vykoná práve 1 inštrukciu.

Na začiatku programu má byť deklarácia lokálnych premenných, nasledujú samotné inštrukcie. V programe musí každá inštrukcia byť uvedená na samostatnom riadku (teda v jednom takte vykoná každý počítač práve jeden riadok programu). Riadok môžeme ukončiť bodkočiarkou, ňou zároveň začína komentár. Povolené inštrukcie sú:

– priradenie
– prázdna inštrukcia **nop**
– inštrukcia ukončenia výpočtu **halt**
– príkaz skoku **goto** ⟨návestie⟩
– podmienený príkaz **if** ⟨podmienka⟩ **then** ⟨inštrukcia⟩ (**else** ⟨inštrukcia⟩)

Pravú stranu priradenia môže predstavovať ľubovoľný aritmetický výraz, podmienkou môže byť ľubovoľný logický výraz. Okrem premenných a konštánt môžu tieto výrazy obsahovať volanie funkcie *cpuid*, ktorá vráti číslo počítača, na ktorom program beží. (Všimnite si, že nemáte k dispozícii kľúčové slová **begin** a **end**.) Návestia sa definujú rovnako ako v Pascale: napísaním ⟨názov⟩: pred príslušný riadok.

Zostáva vyriešiť prácu so spoločnou pamäťou. Môže sa stať, že niekoľko počítačov chce v tom istom takte čítať z, resp. zapisovať na to isté pamäťové miesto. V takomto prípade

najskôr prebehne čítanie, potom zápis, Čítať jedno políčko globálnej pamäte môže naraz ľubovoľne veľa počítačov, ale zapisovať dovolíme len jednému. Ak by naraz chcelo na to isté miesto zapisovať viac počítačov, program vráti chybu. (T.j. tomuto sa pri písaní programu treba vyhnúť.)

Ukážeme si teraz, v čom je sila mravčích počítačov. Na klasickom počítači potrebujeme na nájdenie najväčšieho z daných n čísel čas $O(n)$. Pozrime sa, ako rýchlo ho vedia nájsť mravce. Nech teda v `Mem`[0] je uložené n a v `Mem`[1] až `Mem`[n] jednotlivé čísla. Spustime na n počítačoch nasledovný program:

```
    var N, num1, num2 : integer;
    N := Mem[0];
1:  if (2 · cpuid − 1 > N) then halt;
    num1 := Mem[2 · cpuid − 1];
    if (2 · cpuid ≤ N) then num2 := Mem[2 · cpuid] else num2 := num1;
    if (num1 < num2) then Mem[cpuid] := num1 else Mem[cpuid] := num2;
    N := ⌊(N + 1)/2⌋;
    if (N = 1) then halt;
    goto 1;
```

Po skončení bude v `Mem`[1] uložené najväčšie spomedzi daných čísel. Prečo? Počas prvého prechodu cyklom sme rozdelili zadané čísla na dvojice, z každej sme vybrali to väčšie a následne sme „víťazov“ uložili na políčka $1 \ldots \lceil n/2 \rceil$ – tým sme vlastne dostali pôvodnú úlohu s približne polovičným počtom čísel, to celé v konštantnom čase! Preto náš program naozaj nájde maximum spomedzi daných n čísel, a to v čase $O(\log n)$.

Úloha: Vašou úlohou bude napísať program, ktorý spočíta čiastočné súčty danej postupnosti. Teda nech v `Mem`[0] je uložené n a v `Mem`[1] až `Mem`[n] jednotlivé celé čísla. Po skončení by v `Mem`[i] (pre $1 \le i \le n$) mala byť uložená hodnota `Mem`[1] $+ \cdots +$ `Mem`[i]. Môžete uviesť, koľko počítačov má byť na začiatku spustených v závislosti od n, počet počítačov však smie od n závisieť nanajvýš polynomiálne (teda $n^4 + 7$ počítačov je v poriadku, ale 2^n už nie.)

Prvoradým kritériom hodnotenia bude čas výpočtu vášho programu, nasleduje použitá pamäť (súčet lokálnych pamätí spustených počítačov a počtu použitých políčok globálnej pamäte) a počet použitých počítačov.

2121. O povrchovej ťažbe zlata

Santa a Banta už omrzelo ručné kopanie zlata s krompáčom v tmavej štôlni, a tak si na Vyšnej Klondike založili vlastnú zlatokopeckú spoločnosť. Rozhodli sa pre povrchovú ťažbu bagrami. Nakúpili bagre značky Caterpillar a nechali si vypracovať drahý geologický prieskum.

Geologický prieskum je mapa rozdelená na $r \times s$ tzv. jutroštvorcov. Každý zodpovedá pochopiteľne jednému jutru (t.j. asi 5755 m^2). Pre každý jutroštvorec mapa hovorí, aká je hodnota zlata v dolároch obsiahnutá pod jeho povrchom.

Santovi a Bantovi ostáva už iba odkúpiť od mesta pozemok na ťažbu. Mesto predáva pozemky po d dolárov za jutro, ale nepredá len tak hocijaký kus pozemku. Totiž, z technicko-administratívnych príčin sa zakazuje drobenie jutroštvorcov a naviac sa požaduje, aby pozemok bol tvaru obĺžnika. Pomôžte Santovi a Bantovi kúpiť vhodný pozemok maximalizujúci ich zisk. Zisk z pozemku je rozdiel hodnoty zlata na ňom a ceny za jeho odkúpenie od mesta.

Úloha: Je dané kladné číslo d – cena za jedno jutro, rozmery mapy r, s a mapa $r \times s$ nezáporných čísel, ktoré určujú hodnotu zlata v jednotlivých jutroštvorcoch. Napíšte program, ktorý nájde na mape pozemok tvaru obĺžnika, z ktorého bude najväčší zisk, a vypíše súradnice jeho dvoch protiľahlých rohov. (Ľavý horný roh mapy má súradnice $[1, 1]$, pravý dolný $[r, s]$.)

Príklad:

Vstup:	Výstup:
6 3 4 1 2 0 4 7 5 7 8 1 9 7 6	2 2 3 4 *(Ide o obdĺžnik rozmerov 2×3 v pravom dolnom rohu mapy, zisk z neho je 6 dolárov.)*

2122. O plnom prasiatku

Ako možno viete, budú sa rušiť desať- a dvadsaťhalierniky. Vie to aj Kleofáš, ktorý si už dlšie obdobie dopisuje s Jimmym, kamarátom z USA. Len čo Jimmy pochopil z Kleofášovej lámanej angličtiny, čo sa na Slovensku chystá, až ho striaslo pri predstave, že by sa niečo podobné stalo v jeho rodných Štátoch. To by bolo jeho úsilie našetriť si peniažky na klavír úplne márne. V panike rozbil prasiatko, roztriedil si mince a hybaj do hudobnín. A že paniku chytil poriadnu, chcel pri kúpe klavíra minúť čo najviac mincí.

Úloha: Jimmy mal v prasiatku p kusov mincí v hodnote 1 cent (penny), n kusov päťcentových mincí (nickel), d kusov desaťcentových mincí (dime) a q mincí v hodnote 25 centov (quarter). Zapáčil sa mu klavír, ktorého cena je k centov. Poraďte Jimmymu, akým najväčším počtom mincí dokáže zaplatiť túto sumu.

Na vstupe sú postupne čísla p, n, d, q a k.

Príklad:

Vstup:	Výstup:
6 5 4 3 47	Ide to pomocou najviac 9 mincí.
1 2 3 4 74	Vôbec to nejde.

2123. O trpaslíkoch a diaľnici

V trpasličej dedine sa chýli k voľbám. Terajší starosta si uvedomil, že sa márne bude snažiť o znovuzvolenie, dokiaľ neotvorí ani jediný úsek diaľnice. Už teraz mu je jasné, že diaľnica by mala viesť cez les, lebo inde už nie je miesto. Kvôli stavbe však nesmú vyťať ani jeden strom, tie majú trpaslíci v úcte. Vôbec nie je dôležité, či diaľnica bude spájať nejaké dôležité miesta, môže sa tiahnuť ľubovoľným smerom, hlavne aby bola. A aby bola rovná, lebo rovné diaľnice sa stavajú rýchlo. A aby bola dosť dlhá, nech sa môže starosta chváliť. Stačí, keď bude dvojprúdová (jeden jazdný pruh v každom smere), aj tak sa po nej asi nebude jazdiť. A najdôležitejšie je, aby jej stredová čiara prechádzala miestom, kde má starostov hlavný protikandidát postavenú chatu. (Tú bude samozrejme pri tej príležitosti treba zbúrať.)

Úloha: Na vstupe je daný počet stromov v lese n a šírka jedného jazdného pruhu d (teda polovica šírky dialnice). Nasledujú súradnice n bodov v rovine predstavujúcich stromy. Chata starostovho hlavného konkurenta vo voľbách sa nachádza v bode $[0, 0]$.

Na výstup by mal váš program vypísať, či sa za týchto podmienok dá postaviť diaľnica. Diaľnica má v každom smere od chaty dĺžku väčšiu ako je vzdialenosť najvzdialenejšieho stromu. (Dalo by sa priam povedať, že sa tiahne do nekonečna.)

Vstup:	Výstup:
3 2.00 0.00 1.00 1.00 -1.00 -1.00 -1.00	Nedá sa postaviť.
Vstup:	Výstup:
3 0.80 0.00 1.00 1.00 -1.00 -1.00 -1.00	Dá sa postaviť.

2124. O slovnej hádanke

Miško s Jankom sa pretekajú, kto má lepšiu slovnú zásobu. Lenže všetky hry ich už omrzeli, tak si Miško vymyslel novú. Na jej vysvetlenie si musíme povedať, čo je to pole koncov ľubovolného slova. (Po anglicky sa to zvykne nazývať suffix array.)

Predstavte si, že napíšeme pod seba najprv celé slovo, potom celé slovo bez prvého písmena, potom bez druhého... a nakoniec iba posledné písmeno. Riadky, ktoré takto dostaneme, budeme volať konce slova. Teraz poprehadzujeme konce slova medzi sebou tak, aby pri čítaní zhora nadol boli v abecednom poradí, a očíslujeme ich zhora nadol, začínajúc od 1. Potom všetky konce aj spolu s ich číslami poprehadzujeme naspäť od najdlhšieho po najkratší. Keď teraz prečítame zaradom iba čísla, dostaneme pole koncov. Napríklad pre slovo „`kingdzaba`“ by vznikla postupnosť čísel 7 6 8 5 4 9 2 3 1.

Priebeh hry je úplne jednoduchý. Miško si myslí slovo, vypočíta pole koncov a napíše ho na papier. Jankovou úlohou je nájsť slovo, ktoré si Miško myslel. Pomôžte Jankovi. Keďže pre dané pole koncov existuje veľa vhodných slov, nájdite také, ktoré má najmenej rôznych písmen. (Slovo, ktoré nájdete, nemusí byť zmysluplné.)

Úloha: Na vstupe je dané pole koncov ukončené -1. Nájdite slovo w, ktoré má najmenší možný počet rôznych písmen a jeho pole koncov je rovnaké ako to na vstupe.

Vstup:	Výstup:
`6 2 5 4 3 1 -1`	`cabbba`
`7 6 8 5 4 9 2 3 1 -1`	`dcecbfaba`

2125. O mravcoch II

V úlohe budeme pracovať s mravčími počítačmi popísanými v zadaniach prvého kola (2115) – mravce dostali na narodeniny kráľovnej počítače; všetky počítače sú rovnaké a je ich veľmi veľa. Vašou úlohou je napísať program, ktorý keď nahráme na všetky počítače a naraz ho na všetkých spustíme, niečo zmysluplné spočíta.

Úloha: V poli `Mem`[ ] je uložená postupnosť *navzájom rôznych* celých čísel. V `Mem`[0] je uložená jej dĺžka n, v `Mem`[1] až `Mem`[n] sú jej jednotlivé prvky. Označme x hodnotu uloženú v `Mem`[1]. Váš program má prvky tejto postupnosti preusporiadať tak, aby obsahovala najskôr (v ľubovoľnom poradí) prvky menšie ako x, potom x a následne prvky väčšie ako x.

Teda ak je na vstupe postupnosť (6; 5, 3, 7, 1, 12, 9), tak jedným z možných výstupov je postupnosť (6; 3, 1, 5, 9, 7, 12). V tomto príklade `Mem`[0] = 6 je dĺžka postupnosti, v `Mem`[1] je 5, v `Mem`[2] je 3, atď.

Môžete uviesť, koľko počítačov má byť na začiatku spustených v závislosti od n, počet počítačov však smie od n závisieť nanajvýš polynomiálne. (Teda $n^4 + 7$ počítačov je v poriadku, ale 2^n už nie.)

Prvoradým kritériom hodnotenia bude čas výpočtu vášho programu, nasleduje použitá pamäť (súčet lokálnych pamätí spustených počítačov a počtu použitých políčok globálnej pamäte) a počet použitých počítačov.

2131. O matematikoch

Matematici si radi dávajú matematické hádanky. Ak matematik zadá druhému nejakú matematickú úlohu a ten ju do týždňa nevyrieši, tak musí pozvať svojho vyzývateľa na dobrú hostinu. (To je vecou cti.) Nuž a jeden úbohý matematik rieši jednu zapeklitú úlohu už skoro týždeň a nevie ju vyriešiť. Pomôžte mu!

Úloha: Dané je číslo n a množina obsahujúca n navzájom rôznych celých čísel. Nájdite štyri čísla a, b, c, d z tejto množiny (nie nutne navzájom rôzne) tak, aby platilo $a+b+c=d$. Ak také čísla neexistujú, podajte o tom správu.

Príklad:

Vstup:	Výstup:
`9`	`1+1+7 = 9`
`-15 1 4 10 2 9 7 3 -1`	

2132. Oooooooo Bisťu Blabla Bum

Bum. Bum! „Ja ti ukážem!“ Bum. Trésk. Chrap. Rozbitý kvetináč. Zlomené dvere. „Aúúúúúú.“ Spadnutá polička. „Ty pablb škaredý!“ Pred dvojicou stánkov s párkami sa mlátili predavači, ktorým stánky patrili. Konkurenčný boj nadobúdal na tvrdosti. „Presťahuj si tú tvoju škatuľu!“ Buch! „Nie, ty sa prac, (prásk), ja som tu bol prvý!“ Obďaleč postával dav ľudí – niektorí zvedaví, niektorí si jednoducho chceli kúpiť párky, no nebolo u koho.

„Takto to ďalej nejde!“ usúdil radný pán Richard (ktorý práve stál v spomínanom dave a patril do druhej skupiny ľudí). „Vydáme nariadenie, že stánky s párkami nesmú stáť príliš blízko seba!“ A stalo sa. Predavači s párkami boli zrazu donútení spolupracovať a vyriešiť dilemu – kde majú postaviť stánky, aby neporušili podmienky mestskej rady a ušlo sa miesto každému z nich?

Ale nebojte sa, nedošlo k zmene ich charakteru... Len čo im pomôžete vyriešiť túto úlohu, pobijú sa o to, ktorí z nich budú mať stánky na lukratívnejších miestach.

Úloha: Mesto sa skladá z n križovatiek a $n-1$ ulíc, ktoré medzi nimi vedú. Keďže mesto je malé, medzi každými dvoma križovatkami sa dá po uliciach prejsť len jedným spôsobom – matematik by povedal, že cestná sieť mesta je strom. Stavať stánky na križovatkách sa nesmie. Ak na niektorej ulici postavíme stánok s párkami, nemôžeme už žiadny iný postaviť ani na tejto ulici, ani na žiadnej z tých, ktoré s ňou susedia. (Dve ulice sú susedné, ak majú spoločnú križovatku.) Váš program by mal v meste rozmiestniť najväčší možný počet stánkov s párkami.

Príklad:

Vstup:

```
križovatiek: 9
ulice: 1-9, 1-2, 2-3, 3-4,
       4-5, 4-6, 4-7, 4-8
```

Výstup:

```
Stánky treba dať na ulice:
1-9, 2-3, 4-6
```

(Križovatky sú očíslované od 1 do 9. Ulica je zadaná číslami križovatiek, ktoré spája.)

2133. O pánoch Hamiltonovi a di Rakovi

Pán Hamilton je obchodný cestujúci. Má taký veľký hnedý kufor a v ňom kopec vecí, ktoré by si niekto mohol kúpiť. A s týmto kufrom cestuje po svete a koho stretne, tomu ponúka čokoľvek, čo v kufri nájde.

Pán Hamilton má však momentálne práce vyše hlavy, a preto si vzal dovolenku. So svojim starým známym priateľom pánom di Rakom sa spoločne vybrali navštíviť isté tichomorské súostrovie (ktorého meno neprezradíme, lebo to by ich tam potom každý vyrušoval). V súostroví sa nachádza n ostrovov. Podnikaví domorodci poskytujú plavbu člnom medzi niektorými ostrovmi. Pán Hamilton je značne poznačený svojou prácou, a tak vás iste neprekvapí, že by chcel navštíviť každý ostrov práve raz. Teraz je však na dovolenke, preto mu nezáleží, ako dlho bude trvať obhliadnutie ostrovov.

Pán Hamilton má pochybnosti, či vôbec existuje spôsob, ako navštíviť všetky ostrovy a každý práve raz. Jeho priateľ pán di Rak ho ubezpečuje, že určite áno. Veď z každého ostrova ich podnikaví domorodci môžu dopraviť na viac ako $n/2$ ďalších ostrovov. Sami uznajte, to predsa musí ísť. Pán Hamilton je však stále nedôverčivý. Pomôžte mu a zistite, či je možné stráviť dovolenku podľa jeho predstáv.

Úloha: Na vstupe je dané číslo n udávajúce počet ostrovov v súostroví. Ostrovy sú pre jednoduchosť očíslované 1 až n. Ďalej nasleduje zoznam dvojíc ostrovov, medzi ktorými premáva nejaký podnikavý domorodec na člne. Vašou úlohou je zistiť, či existuje plán cesty, pri ktorej pán Hamilton navštívi každý ostrov práve raz a na konci sa vráti na ostrov, z ktorého začal. Ak takýto plán neexistuje, vypíšte o tom príslušnú správu, ak existuje, jeden taký nájdite. Predpokladajte, že z každého ostrova sa dá dostať na viac ako $n/2$ iných ostrovov.

Príklad:

Vstup:

```
5
1 2
1 4
1 5
2 3
2 4
2 5
3 4
3 5
```

Výstup:

```
Ostrovy navštívi v poradí:
1 5 2 3 4
```

2134. Objemné teleso

Predstavte si kôpku rovnako veľkých kociek. Z nich sme pozliepali súvislé teleso, pričom každé dve kocky, ktoré susedia, sa dotýkajú celou stenou. Teleso môže pokojne byť deravé alebo duté.

Ako vám to naše teleso popísať? Predstavte si, že sme celý povrch polepili tvrdým papierom. Keď tento papier po hranách telesa narežeme, dostaneme mnohouholníky predstavujúce steny telesa. Ak by niektoré z nich boli deravé, rozrežeme ich na niekoľko menších mnohouholníkov bez dier. Takto vieme povrch každého telesa z kociek popísať pomocou niekoľkých papierových mnohouholníkov.

Úloha: Zvolili sme si súradnicovú sústavu tak, aby jednotková vzdialenosť zodpovedala dĺžke hrany našich kociek a aby hrany telesa boli rovnobežné so súradnicovými osami. Každý z mnohouholníkov, ktorými teleso popisujeme, teda leží v rovine, ktorá je rovnobežná s rovinou určenou osami xy alebo xz alebo yz. Vašou úlohou je napísať program, ktorý spočíta, z koľkých kociek sa zadané teleso skladá.

Príklad:

Vstup:

```
stien: 12
4 vrcholy: (10,10,10), (10,10,20), (10,20,20), (10,20,10)
4 vrcholy: (20,10,10), (20,10,20), (20,20,20), (20,20,10)
4 vrcholy: (10,10,10), (10,10,20), (20,10,20), (20,10,10)
4 vrcholy: (10,20,10), (10,20,20), (20,20,20), (20,20,10)
4 vrcholy: (10,10,10), (10,20,10), (20,20,10), (20,10,10)
4 vrcholy: (10,10,20), (10,20,20), (20,20,20), (20,10,20)
4 vrcholy: (14,14,14), (14,14,16), (14,16,16), (14,16,14)
4 vrcholy: (16,14,14), (16,14,16), (16,16,16), (16,16,14)
4 vrcholy: (14,14,14), (14,14,16), (16,14,16), (16,14,14)
4 vrcholy: (14,16,14), (14,16,16), (16,16,16), (16,16,14)
4 vrcholy: (14,14,14), (14,16,14), (16,16,14), (16,14,14)
4 vrcholy: (14,14,16), (14,16,16), (16,16,16), (16,14,16)
```

Výstup:

```
objem je 992
```

(Kocka $10 \times 10 \times 10$ *s dierou* $2 \times 2 \times 2$*.)*

2135. O mravcoch III

V úlohe budeme pracovať s mravčími počítačmi popísanými v zadaniach prvého kola (2115) – mravce dostali na narodeniny kráľovnej počítače; všetky počítače sú rovnaké a je ich veľmi veľa. Vašou úlohou je napísať program, ktorý keď nahráme na všetky počítače a naraz ho na všetkých spustíme, niečo zmysluplné spočíta.

Úloha: V poli `Mem`[ ] je uložená postupnosť *navzájom rôznych* celých čísel. V `Mem`[0] je uložená jej dĺžka n, v `Mem`[1] až `Mem`[n] sú jednotlivé prvky postupnosti. Váš program by mal túto postupnosť usporiadať. Teda ak je na vstupe postupnosť $(6;\ 5, 3, 7, 1, 12, 9)$, na výstupe by mala byť postupnosť $(6;\ 1, 3, 5, 7, 9, 12)$. (V tomto príklade `Mem`[0] $= 6$ je dĺžka postupnosti, v `Mem`[1] je 5, v `Mem`[2] je 3, atď.)

Môžete uviesť, koľko počítačov má byť na začiatku spustených v závislosti od n, počet počítačov však smie od n závisieť nanajvýš polynomiálne (teda $n^4 + 7$ počítačov je v poriadku, ale 2^n už nie.)

Prvoradým kritériom hodnotenia bude čas výpočtu vášho programu, nasleduje použitá pamäť (súčet lokálnych pamätí spustených počítačov a počtu použitých políčok globálnej pamäte) a počet použitých počítačov.

Hinty: Aké klasické (sekvenčné) triediace algoritmy poznáte? Ktoré z nich sa dajú dobre paralelizovať a ktoré nie? Nepomohlo by nám, keby sme mohli použiť až $n!$ počítačov? A treba ich vlastne až tak veľa?

2141. O binárnom kamzíkovi

Ako ste už asi podľa nadpisu zistili, táto úloha bude o hre Binárny kamzík. Túto hru už snáď každý pozná, a tak inovatívne pozmeníme pravidlá: V rade stojí n programátorov očíslovaných od 1 po n. Programátori sa rozdelia do rôznych skupín. Každý z nich môže byť vo viacerých skupinách. Na začiatku každý stojí v podrepe. Každá skupina sa môže rozhodnúť či urobí zmenu. Zmena spočíva v tom, že každý programátor v skupine zmení svoj stav. Ak bol v podrepe, tak sa postaví a ak stál tak sa dostane do podrepu. Musíte uznať, že byť v podrepe je trochu nepohodlná a namáhavá činnosť (hlavne pre programátorov). Ich cieľom je preto urobiť postupnosť takých zmien, že na záver budú všetci stáť. Kedže sú takto fyzicky namáhaní, nevedia sa sústrediť na myslenie. Pomôžte im!

Úloha: Na vstupe je počet programátorov n a počet skupín m. Pre jednoduchosť očíslujme skupiny programátorov od 1 po m. Ďalej nasleduje m riadkov, v každom riadku je zapísaná jedna skupina programátorov. Prvé číslo v i-tom z týchto riadkov je k_i – počet ľudí v i-tej skupine. Po ňom nasleduje k_i čísel z intervalu 1 až n, každé najviac raz – čísla ľudí v dotyčnej skupine. Ak existuje nejaké poradie, v akom majú skupiny meniť stav tak, aby na konci všetci ľudia stáli, vypíšte ľubovoľné jedno takéto poradie. Ak úloha nemá riešenie, podajte o tom správu.

Príklad:

Vstup:
```
3 2
2 1 2
2 2 3
```
Výstup:
```
Úloha nemá riešenie.
```

Vstup:
```
5 4
3 1 3 5
4 2 1 4 5
3 3 5 4
2 1 5
```
Výstup:
```
1 2 4
```

Vysvetlíme druhý príklad: Je 5 programátorov, sú v 4 skupinách. Prvá skupina má 3 členov a sú to programátori 1, 3, 5. Druhá skupina má 4 členov a sú to programátori 2, 1, 4, 5. Podobne tretia a štvrtá skupina. Keď teraz postupne zmenia svoj stav skupiny 1, 2 a 4, budú na konci všetci programátori stáť.

2142. O zlatokopovi Kleofášovi

Na Vyšnej Klondike bolo objavené zlato. Bane rástli ako huby po ďažďi. Zlatokopi z celého sveta sa tam zišli, aby si každý uchmatol kúsok zlata pre seba. Kleofáš nebol výnimkou. Za posledné peniaze si kúpil krompáč a vydal sa kopať. Hneď si však uvedomil, že nie v každej bani sa nachádza rovnako veľa zlata. Preto prišiel na miestny zlatokopecký úrad, aby si prečítal najnovšie štatistiky o jednotlivých baniach. V tej spleti údajov sa však Kleofáš nevyzná, a preto vás prosí o pomoc. Pomôžte chudákovi Kleofášovi nahrabať si čo najviac zlata a napíšte program, ktorý určí, kde má Kleofáš ťažiť.

Úloha: Na vstupe dostane váš program popis jednotlivých baní a ciest medzi nimi. V prvom riadku sú čísla n – počet baní a m – počet ciest medzi nimi. V i-tom z ďalších m riadkov sú tri čísla x_i, y_i, d_i, ktoré znamenajú, že cesta medzi baňami x_i a y_i trvá d_i hodín, kde $1 \leq d \leq 10$ je celé číslo. Každú celú hodinu premáva autobus z každej bane do všetkých jej susedných baní.

V ďalšom riadku vstupu je číslo t. Prospektori určili pre každú baňu prognózu na nasledujúcich t hodín – v bani b sa počas h-tej hodiny vyťaží $c_{b,h}$ zlata. Nasledujúcich t riadkov vstupného súboru obsahuje hodnoty $c_{\cdot,\cdot}$, konkrétne h-ty riadok obsahuje na b-tom mieste hodnotu $c_{b,h}$.

Kleofáš začína v bani číslo 1. Vašou úlohou je zistiť, ako má Kleofáš cestovať po jednotlivých baniach, aby počas nasledujúcich t hodín vyťažil čo najviac zlata.

Príklad:

Vstup:

```
3 2
1 2 1
1 3 2
8
100 10 15
80 9 14
60 8 13
40 7 12
20 6 11
0 5 10
1 4 9
2 3 8
```

Výstup:

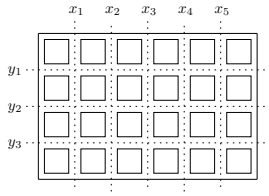
```
Doluj v 1. bani 5 hodín.
Presuň sa do 3. bane.
Doluj v 3. bani 1 hodinu.

Dokopy získaš 308 kg zlata.
```

2143. O čokoláde

Raz doniesla Julka na stretnutie KSP čokoládu, Študentskú pečať. Pažraví vedúci ju chceli hneď a zaraz zjesť. (Čokoládu, nie Julku.) A kedže vedúcich bolo veľa, Julka chcela rozlámať čokoládu na jednotlivé kocky. Lenže čokoláda je tvrdá ako kameň. Navyše pozdĺž niektorých čiar sa zle láme, lebo ako vieme, Študentská pečať obsahuje oriešky, hrozienka a želatínu. Poraďte preto Julke, ako treba lámať čokoládu, aby sa najmenej natrápila.

Úloha: Sú dané rozmery čokolády r a s v jednotkových kockách. Ďalej sú dané kladné čísla $x_1, x_2, \ldots, x_{s-1}$ a $y_1, y_2, \ldots, y_{r-1}$, hovoriace, koľko námahy treba vynaložiť na prelomenie pozdĺž vertikálnej resp. horizontálnej čiary. Presnejšie, pri lámaní pozdĺž i-tej zvislej čiary treba vynaložiť x_i námahy, podobne pre vodorovné čiary.



Čokoládu lámeme postupne, krok za krokom. V každom kroku si vyberieme jeden jej kúsok, zvolíme si na ňom čiaru, podľa ktorej ho chceme prelomiť, vynaložíme námahu zodpovedajúcu dotyčnej čiare, a tým zlomíme aktuálny kúsok na dva nové.

Všimnite si, že pre niektoré čiary budeme námahu danú na vstupe vynakladať viackrát. Napríklad ak polámeme čokoládu najprv podľa vodorovných čiar a potom podľa zvislých čiar, naša námaha bude $y_1 + y_2 + \ldots + y_{r-1} + r(x_1 + x_2 + \ldots + x_{s-1})$, pretože každý z r vodorovných pásikov potrebujeme prelomiť $(s-1)$-krát. (Viď obrázok.) Vašou úlohou je napísať program, ktorý vyráta, akú najmenšiu celkovú námahu treba vynaložiť na rozlámanie čokolády.

Príklad:

Vstup:	Výstup:

`6 4`
`2 1 3 1 4`
`4 1 2`
	42

2144. O plnej izbe

„Vyzerá to tu ako v sklade, Mal by si si to aspoň usporiadať," povedala mama Jankovi, keď uvidela jeho izbu. Janko totiž rád zbiera rôzne veci a vecičky a ukladá si ich v izbe. Výsledkom je, že u Janka v izbe to naozaj vyzerá ako v sklade. Teda aby som bol presnejší, podlahu Jankovej izby si môžeme predstaviť ako štvorcovú sieť s n riadkami a n stĺpcami. Na každom štvorci okrem jedného je škatuľa s rôznymi vecami. Aby Janko vedel, kde čo je, má každá škatuľa svoje číslo. Niekde vedľa potom existuje zoznam vecí, v ktorom je okrem iného napísané, čo je v ktorej škatuli. Keďže Janko je malý a škatule sú veľké, Janko vie len presunúť nejakú škatuľu na susedný prázdny štvorec (ak taký existuje). Štvorce považujeme za susedné, ak majú spoločnú stranu. Janko by rád svojej mame urobil radosť a usporiadal tie škatule, ale zdá sa mu, že sa to nedá. Chce si však byť úplne istý.

Úloha: Pomôžte mu a napíšte program, ktorý zistí, či vie Janko škatule usporiadať tak, aby na i-tom riadku a j-tom stĺpci bola krabica s číslom $n \cdot (i-1)+j$. Štvorec v n-tom riadku a n-tom stĺpci má na konci byť prázdny.

Príklad:

Vstup:	Výstup:

`2`
`2 1`
`3 .`
	`nedá sa`

Vstup:	**Výstup:**

`3`
`4 1 3`
`2 . 8`
`7 6 5`
	`dá sa`

Poznámka: Pre $n = 4$ sa Jankovej úlohe hovorí Loydova pätnástka.

2145. O mravcoch IV

V úlohe budeme pracovať s mravčími počítačmi popísanými v zadaniach prvého kola (2115) – mravce dostali na narodeniny kráľovnej počítače; všetky počítače sú rovnaké a je ich veľmi veľa. Vašou úlohou je napísať program, ktorý keď nahráme na všetky počítače a naraz ho na všetkých spustíme, niečo zmysluplné spočíta.

Úloha: V globálnej pamäti (t.j. poli `Mem`[]) je uložený neorientovaný ohodnotený graf. (Konkrétny formát si zvoľte vy.) Navrhnite mravčí program, ktorý čo najrýchlejšie nájde najlacnejšiu kostru tohto grafu.

Prvoradým kritériom hodnotenia bude čas výpočtu vášho programu, nasleduje použitá pamäť (súčet lokálnych pamätí spustených počítačov a počtu použitých políčok globálnej pamäte) a počet použitých počítačov.

Táto úloha je trochu náročnejšia, preto po vás nechceme konkrétny mravčí program, stačí nám hlavná myšlienka algoritmu. Samozrejme, čím podrobnejšie, tým lepšie.

z2111. Zľavy na železniciach

Ako ste si už určite všimli, každý rok sa menia ceny dopravy. Ani tento rok nie je výnimkou. Zákazníka často nalákajú hlavne zľavy, ktoré skoro každá spoločnosť ponúka. Akcia, ktorú plánujú železnice na budúci rok, bude vyzerať nasledovne: Zverejní sa zoznam veľmi veľa rôznych zliav. Ako študenti máte nárok vybrať si pri kúpe lístku hocikoľko z

nich, ktoré chcete použiť. Aby ste si však nemohli vybrať len tie výhodné, vami vybrané zľavy musia v zozname nasledovať za sebou.

Úloha: Napíšte program, ktorý načíta počet zliav n a zoznam zliav. Zľava je kladné reálne číslo, ktorým sa násobí cena lístku. Nájdite úsek zliav, ktorý vám umožní získať najlepšiu možnú výslednú zľavu. Ak je takýchto úsekov viac, stačí vypísať ľubovoľný jeden spomedzi nich.

Príklad:

Vstup:	Výstup:
`7`	`Najlepší úsek: zľavy 3 až 6.`
`0.7 2.0 0.5 0.5 2.0 0.4 1.1`	*(Budete platiť len* 20% *ceny lístku.)*

z2112. Zapálenie druidského ohňa

Janko znova vyletel z algebry. Kráčal zahmlenou Bratislavou a bol nekonečne nešťastný. Zrazu sa zem zdvihla a udrela ho do čela. Teda ono to bolo skôr naopak: Janko si vo svojom žiali nevšimol jednu veľmi pevne zakorenenú jahodu, čo rástla pri matfyze. Keď sa prebral, okolo neho neboli budovy milovanej školy, ale stromy, tráva a nekonečná divočina. Na sebe mal rytierske brnenie a v ruke meč. Neveriacky naň hľadel. Po čase prišiel na to, že sa nachádza na území kráľa Artuša. Potreboval sa však dostať späť na opravný termín z algebry. Bezradne sa pohyboval lesom a premýšľal.

Keď zdvihol pohľad, zistil, že stojí na čistine a uvidel skupinku starčekov. Mali dlhé brady a na nechtoch pleseň. „Keby to tak bol druidi...“ pomyslel si Janko. Druidi boli mocní zaklínači, vedeli komunikovať s duchmi sveta a keby zapálili svoj čarovný oheň a dali sily dokopy, vedeli by napríklad aj dostať Janka späť do Bratislavy. Pomaly sa mu začala vracať nádej.

Samotný rituál bol jednoduchý – stačilo, aby sa druidi postavili do vrcholov pravidelného n-uholníka a Janko zapálil v jeho strede oheň. Druidi však boli starí a hrozne senilní. Každú chvíľu jeden z nich zabudol, že robia zaklínadlo, a začal sa ponevierať po lúke. Janko je už zúfalý z toho čakania, preto by potreboval program, ktorý by mu oznamoval, či všetci druidi stoja správne.

Úloha: Na vstupe je číslo n a súradnice n bodov. Zistite, či sú to vrcholy pravidelného n-uholníka. Body nie sú nutne v správnom poradí a stred n-uholníka nemusí byť počiatkom sústavy súradníc.

Príklad:

Vstup:	Výstup:
`5`	`Nie.`
`1 8 2 2 6 7 3 5 4 4`	

z2113. Zadania písomky

Jeden profesor na jednej nemenovanej škole má svoje obľúbené číslo m, ktoré veľmi rád všade používa. Vždy pred písomkou zverejní čísla úloh zo zbierky, z ktorých potom jednu vyberie na písomke. Študenti od svojich predchodcov vedia, že na písomke dá spomedzi zverejnených úloh m-tú od konca. Majú však problém. Všetky čísla úloh, ktoré im profesor diktuje, si nik z nich nezapamätá. A nemajú si ich ani kam zapísať. (Nik z nich nemá zošit ani nič podobné – veď kto by si už len písal poznámky, že?) Jeden študent si ale nosí na hodiny počítač. Pomôžte mu napísať program, ktorému bude zadávať čísla, ktoré hovorí profesor, a ktorý na konci vypíše, akú úlohu dostanú na písomke.

Úloha: Váš program bude všetky informácie dostávať z klávesnice. Ako prvé dostane profesorovo obľúbené číslo m. Nasleduje veľa kladných celých čísel – čísla úloh zo zbierky. Vstup je ukončený číslom 0. Vašou úlohou je zistiť číslo m-tej úlohy od konca. Najdôležitejšie je, aby váš program potreboval čo najmenej pamäte. (Profesorovo obľúbené číslo je síce pomerne malé, ale úloh môže zadať vééľmi veľa!)

Príklad:

Vstup:	Výstup:
6 5 1 8 56 7 14 23 47 11 4 58 32 0	23

z2114. Záhrada víly Amálky

Víla Amálka má v záhrade onbloň. Onbloň je veľmi zaujímavý strom. Vyzerá skoro, ale nie úplne, ako tybloň. Ono to vlastne ani nie je strom, je to iba n onbĺk (očíslovaných od 1 do n), medzi ktorými je m vetvičiek. Každá vetvička spája práve dva onblká a naviac platí, že keby po strome behala veverička, vedela by po týchto vetvičkách prebehnúť medzi ľubovoľnými dvomi onblkami. Na každej onbloni rastú onblká dvoch farieb: modrej a červenej. Pritom modrých onbĺk je vždy viac ako červených. Onblone sú veľmi vzácne, lebo taká onbloň vyrastie iba zo semienok modrého onblka.

Víla Amálka nikoho nepustí do svojej záhrady, ale profesorovi Indigovi sľúbila, že mu daruje jedno onblko, ktorého číslo jej povie. Okrem toho je ochotná odpovedať na otázky, či majú dve susedné onblká rovnakú farbu (1 – majú rovnakú farbu, 0 – nemajú).

Profesor Indigo by si chcel vypestovať vlastnú onbloň. Nechce nič nechať na náhodu, preto vás poprosil, aby ste mu napísali program. Ten bude namiesto profesora klásť víle otázky a na konci oznámi číslo onblka, o ktorom si je istý, že je modré. Snažte sa, aby váš program nekládol zbytočne veľa otázok, lebo si víla Amálka všetko rozmyslí a žiadne onblko nebude.

Úloha: Na vstupe je počet onbĺk n a počet vetvičiek m. Nasleduje m dvojíc čísel onbĺk, medzi ktorými vedú vetvičky.

Váš program sa má užívateľa (Amálky) pýtať otázky. Program vždy vypíše dvojicu čísel onbĺk, medzi ktorými vedie vetvička, a užívateľ zadá, či majú rôznu alebo rovnakú farbu. Keď už si je program istý, že niektoré onblko je modré, vypíše jeho číslo a skončí.

Najdôležitejšie je aby váš program položil čo najmenej otázok, druhým kritériom hodnotenia bude čas behu vášho programu.

Príklad:

Vstup:	Komunikácia:
7 7 1 2 1 5 1 3 5 3 4 3 3 6 6 7	onblká 3 a 6? > rôzne onblká 3 a 4? > rôzne onblká 3 a 1? > rôzne onblká 3 a 5? > rovnaké onblká 6 a 7? > rovnaké chcem onblko 6

z2115. Získajte body zadarmo!

Zadanie tejto úlohy sa nikomu nechcelo vymýšľať. Najskôr sme vám chceli prideliť body náhodne, ale to by nebolo fér. A tak sme situáciu vyriešili nasledovne: Zverejníme vám funkciu *body*. Tá berie ako vstup 32-bitové celé číslo so znamienkom a vráti celé číslo od 0 do 15. Vašou úlohou je poslať nám vstup pre túto funkciu. Hodnota, ktorú funkcia vráti pre váš vstup, bude zároveň počet bodov, ktorý za túto úlohu získate. A nezabudnite nám napísať aj to, ako ste na váš vstup prišli, lebo môžete pár z týchto ľahko získaných bodov ľahko stratiť!

Funkciu uvádzame v jazyku Pascal, dúfame, že riešitelia používajúci iné programovacie jazyky jej zápisu budú rozumieť.

Kód funkcie body (a pomocných funkcií):

```
function lala(a, b : longint) : longint;
begin if (a · b > 0) then lala := lala(a − 1, b − 1) else lala := b − a; end;

function huhu(a, b : longint) : longint;
begin if (a ≤ b) then huhu:=0 else huhu := lala(b + 1042, a + 1047) − 5; end;

function dodo(a, b : longint) : boolean;
var i : longint;
begin
  if (a = 0) then begin dodo := true; exit; end;
  if (huhu(b, a) ≠ 0) then begin dodo := false; exit; end;
  for i := 1 to b − huhu(b, a) − 1 do
    if dodo(a − i, b) then begin
      dodo := (lala(huhu(b, a) + a, huhu(a, b) + b) ≠ 0); exit;
    end;
  dodo := dodo(a − b, b);
end;

function zuzu(a : longint) : boolean;
var i, j : longint;
begin
  j := 0;
  for i := a + huhu(a, a) downto 2 do if dodo(a, i) then
    if (j > 0) then begin zuzu := dodo(4 + 7 · ⌊100000/a⌋, 7); exit; end else j := i;
  zuzu := (j > 0);
end;

function body(vstup : longint) : longint;
var a, b : integer;
begin
  a := 2; b := 0;
  if ((vstup < 2) or (vstup > 2 · 10^9)) then begin body := 0; exit; end;
  while not dodo(vstup, a) do repeat a := a + 1; until zuzu(a);
  while dodo(vstup, a) do begin vstup := ⌊vstup/a⌋; b := b + 1; end;
  if (a < 5) then begin body := a; exit; end;
  if (vstup = 1) then b := b + 2 else begin
    vstup := ⌊(vstup + 3)/2⌋; b := 1;
    while (vstup > 0) do begin vstup := vstup − 1; b := (5 · b) mod 15; end;
  end;
  body := b;
end;
```

z2121. Zachariáš bezdomovec

Zachariáš je bezdomovec a navyše nezamestnaný. Dlhú chvíľu si kráti tým, že na zastávke zbiera špaky, robí si z nich cigarety a fajčí. (Uznávame, že tento príbeh nie je práve mravný a poučný, ale taký je život. Snáď chápete.)

Úloha: Zachariáš nazbieral n špakov. Keďže je šikovný, z k špakov (k je najviac 10) si vie spraviť jednu cigaretu. Napíšte program, ktorý z čísel n a k spočíta, koľko cigariet Zachariáš vyfajčí. Pokúste sa, aby váš program fungoval aj pre veľmi veľké hodnoty n.

Príklad:

Vstup:	Výstup:
`9 3`	`4 cigarety`
`5 2`	`4 cigarety`
`3 1`	`nekonečne veľa cigariet`
`123456789123456789 5`	`30864197280864197 cigariet`

Vysvetlíme prvý výstup: Z 9 špakov si Zachariáš vie spraviť 3 cigarety. Ale tým, že ich vyfajčí, dostane 3 nové špaky a z tých si vie spraviť ďalšiu cigaretu.

Prémia: Keď si Zachariáš prečítal toto zadanie, vysvetlil nám, že u neho je $k = 3$. A ešte nám povedal, že dnes ráno mal 10 špakov, žiadne už nenašiel a napriek tomu vyfajčil 5 cigariet. Ako je to možné?

z2122. Zábavka pred obedom

Na jednom zo sústredení KSP hrali účastníci nasledujúcu hru. Bolo už tesne pred obedom, keď sa všetci postavili vedľa seba na štartovú čiaru. Každý držal v ruke zalepenú obálku a netrpezlivo čakal na štartovací povel. Vtom okolím zaznelo mocné „Štart!“ a všetci vybehli k cieľovej čiare umiestnenej opodiaľ. Ich úlohou bolo bežať čo najrýchlejšie k cieľu a cestou splniť úlohu popísanú v liste v obálke. Bežia, bežia, trhajú obálku, dolujú z nej papier, keď vtom prvý z nich zastal…

Zastali aj ostatní a rozmýšľajú ostošesť. Na papieri mal každý správu tohoto typu: „Do cieľa môžeš prísť až po Katke, Paľkovi, Jožkovi, Monike a Rasťovi.“ Každý z účastníkov mal zoznam ľudí, ktorí ho musia predbehnúť. (Ostatní môžu, ale nemusia. Zoznam mohol byť aj prázdny.) Super hra, čo? Možno, ale nie tesne pred obedom. Všetci sú už hladní a nevedia, v akom poradí majú dobehnúť. Pomôžte im zistiť, v akom poradí majú dobehnúť, resp. či vôbec existuje nejaké vhodné poradie.

Úloha: Na vstupe je dané číslo n predstavujúce počet účastníkov, ktorí hrali hru. Účastníkov pre jednoduchosť očíslujeme od 1 po n. V každom z ďalších n riadkoch je úloha pre jedného účastníka. Konkrétne v $(i + 1)$-om riadku je zoznam ľudí, ktorí majú predbehnúť účastníka i, ukončený nulou.

Vypíšte poradie, v akom majú účastníci dobehnúť do cieľa. Ak je takýchto poradí viac, vypíšte ľubovoľné z nich. Ak poradie neexistuje, vypíšte o tom príslušnú správu.

Vstup:	Výstup:

`5`
`2 5 0`
`5 0`
`4 0`
`0`
`4 0`
	`Vyhovuje poradie: 4 3 5 2 1`

Vstup:	**Výstup:**

`3`
`2 0`
`3 0`
`1 0`
	`Žiadne poradie nevyhovuje.`

z2123. Zima je tu!

Sneh, nesneh, určite si už zistil, že zima sa blíži. Preto treba zobrať lyže a hor sa na lyžovačku. Len ktoré lyže si vybrať, aby si mal tie najlepšie pasujúce aj ty, aj tvoji kamaráti?

Úloha: Na vstupe je číslo n – počet párov lyží a zároveň počet lyžiarov. Potom nasleduje n čísel zodpovedajúcich výškam lyžiarov a za nimi n čísel zodpovedajúcich dĺžkam lyží. Tvojou ulohou je napísať program, ktorý každému lyžiarovi priradí práve jedny lyže

(dvaja lyžiari nemôžu mať tie isté lyže) a to tak, že súčet hodnôt |výška lyžiara – dĺžka lyží| je minimálny.

Príklad:

Vstup:

```
5
158 169 163 179 182
147 191 179 181 160
```

Výstup:

```
1. lyžiar: lyže dĺžky 147
2. lyžiar: lyže dĺžky 179
3. lyžiar: lyže dĺžky 160
4. lyžiar: lyže dĺžky 181
5. lyžiar: lyže dĺžky 191
```

z2124. Za rohom je kôň

Na dlážke bol koberec a na koberci rozhádzaná kopa šachových figúrok. Na šachovnici stálo 12 koní, vedľa nej sedel malý Gary a smutne hľadel na neohrozené políčko a8. Nech sa snažil ako chcel, nedarilo sa mu rozmiestniť kone tak, aby každé prázdne políčko aspoň jeden z nich ohrozoval. A viac ako 12 koní doma nenašiel. Asi sa rozplače. Rodičia mu potom kúpia ďalšiu šachovnicu a bude po probléme.

Rodičom sa ale nechce vyhadzovať peniaze zbytočne, preto by potrebovali vedieť, koľko koní vlastne malý Gary na svoju hru potrebuje.

Úloha: Zistite, koľko najmenej koní stačí umiestniť na šachovnicu $n \times n$ tak, aby každé neobsadené políčko bolo ohrozené aspoň jedným koňom. Pošlite nám správne odpovede pre čo najviac n. Okrem samotného počtu koní nám pre každé n pošlite aj jedno vhodné rozostavenie toľkých koní. A nezabudnite poslať aj program, ktorým ste tieto hodnoty získali.

Poznámka: Kôň (oficiálne jazdec) je šachová figúrka, ktorá sa hýbe „skokom za roh" – pohne sa najskôr o 2 políčka vodorovne alebo zvislo, potom o jedno políčko smerom kolmým na ten, ktorým sa hýbal doteraz. Kôň ohrozuje tie políčka, na ktoré sa môže jedným ťahom dostať. Kôň stojaci v strede šachovnice 5×5 teda ohrozuje 8 políčok.

Príklad:

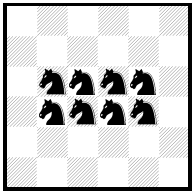
Vstup:

```
6
```

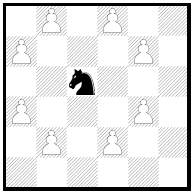
Výstup:

```
Stačí 8 koní.
```

Obrázok k výstupu:



Políčka ohrozené koňom:



z2125. Zaopravujte si

Márne píšeme do pravidiel „nezabúdajte na popis algoritmu", márne je prehováranie do duše „prečítajte si svoje riešenie pred tým, ako ho pošlete, či ho po sebe pochopíte aspoň vy"... So železnou pravidelnosťou nám do KSP prichádzajú záhadné riešenia, ktoré nedávajú zmysel ani po treťom prečítaní a ak vôbec majú nejaký popis, tak jeho úlohou je opravovateľa ešte viac popliesť.

Tentokrát sme sa rozhodli, že to spravíme naopak, aby sme vás o tieto úžasné zážitky nepripravili. Vašou úlohou teda bude opraviť tri (krásne prehľadné a jednoduché) riešenia jednej úlohy. Mali by ste uviesť, koľko bodov (od 0 do 15) si podľa vás každé z týchto riešení zaslúži a poriadne zdôvodniť, prečo. (Rozhodujúca je samozrejme správnosť myšlienky a efektívnosť algoritmu. Ak neviete, čo všetko brať pri opravovaní do úvahy, ešte raz si poriadne prečítajte pravidlá KSP.)

Úloha: Na planéte Diks je n miest, ktoré sa volajú 1, 2, ..., n. Každé dve mestá sú spojené potrubím, ktorým mimozemšťania cestujú. Pre každú dvojicu miest je známe, ako

dlho trvá presun potrubím, ktoré ich spája. Pomôžte mimozemšťanovi Krzysztofovi zistiť, ako najrýchlejšie sa dá dostať z mesta 1 do mesta n.

Príklad: Nech $n = 3$, na presun medzi 1 a 2 treba 27 minút, medzi 1 a 3 to je 47 minút a medzi mestami 2 a 3 zase 13 minút. Potom z mesta 1 do mesta 3 sa najrýchlejšie dá dostať za 40 minút (cez mesto 2).

Riešenie 1:

```
var A : array[1..100,1..100] of integer;
    naj : integer;
    N,i : integer;
    bol : array[1..100] of boolean;

procedure load;
var i,j : integer;
begin
  readln(N); for i := 1 to N do A[i,i] := 0;
  for i := 1 to N do for j := i+1 to N do begin read(A[i,j]); A[j,i] := A[i,j]; end;
end;

procedure skus(kam,dlzka : integer);
var i : integer;
begin
  if (kam = N) then begin
    if (dlzka < naj) then naj := dlzka;
  end else begin
    bol[kam] := true;
    for i := 1 to N do if not bol[i] then skus(i,dlzka + A[kam,i]);
    bol[kam] := false;
  end;
end;

begin
  load;
  naj := ∞; fillchar(bol,sizeof(bol),0);
  skus(1,0);
  writeln(naj);
end.
```

Riešenie 2: Mení sa len procedúra *skus*.

```
procedure skus(kam,dlzka : integer);
var i,max,pre : integer;
begin
  if (kam = N) then naj := dlzka else begin
    max := ∞;
    for i := 1 to N do if (not bol[i]) and (A[kam,i] < max) then
      begin max := A[kam,i]; pre := i; end;
    skus(pre,dlzka + A[kam,pre]);
  end;
end;
```

Riešenie 3: Procedúra *load* zostáva stále rovnaká.

```
var A : array[1..100, 1..100] of integer;
    naj : array[1..100] of integer;
    N, i : integer;

procedure skus;
var i, j : integer;
begin
  for i := 1 to N do for j := 1 to N do
    if (naj[j] > naj[i] + A[i, j]) then naj[j] := naj[i] + A[i, j];
end;

begin
  load;
  naj[1] := 0; for i := 2 to N do naj[i] := ∞;
  for i := 1 to N − 2 do skus;
  writeln(naj[N]);
end.
```

z2131. Zozolvár najväčší v okolí

Kde bolo, tam bolo, stál raz jeden hrad. Ten hrad mal vežu a nádvorie. Na nádvorí stálo v rade n bohatierov, ktorých výšky boli navzájom rôzne reálne čísla. Hovoríme, že bohatier sa hrozivo týči, ak je vyšší od všetkých svojich susedov. (Teda od oboch, až na tých dvoch bohatierov, ktorí stoja na koncoch radu. U tých stačí, aby boli vyšší od svojho jediného suseda.)

Princezná, ktorá celé dni sedí vo veži a zúfalo sa nudí, často hrávala so svojou komornou nasledovnú hru: Komorná sa nesmie pozerať z okna, môže sa iba pýtať na výšky jednotlivých bohatierov. Jej úlohou je zistiť o aspoň jednom bohatierovi, že sa hrozivo týči.

Úloha: Komorná však dnes má voľno a tak by sa princezná chcela zahrať svoju obľúbenú hru aspoň s dvorným počítačom. Napíšte program, ktorý načíta číslo n (počet bohatierov) a následne sa bude princeznej pýtať na výšky niektorých bohatierov. Úlohou programu je na čo najmenej otázok nájsť nejakého bohatiera, ktorý sa hrozivo týči.

Poznámka. Komorná sa za dlhé roky hrania tak vypracovala, že aj pre 1000 bohatierov jej vždy stačilo menej ako 20 otázok. Vyrovná sa jej váš program?

Príklad:

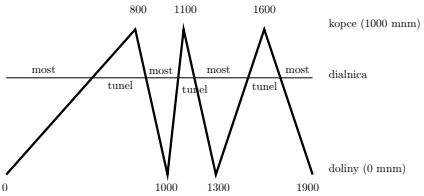
```
Princezná: N = 10
Počítač: Aká je výška 3. bohatiera?
Princezná: 190
Počítač: Aká je výška 5. bohatiera?
Princezná: 140
Počítač: Aká je výška 7. bohatiera?
Princezná: 147
Počítač: Aká je výška 6. bohatiera?
Princezná: 175
Počítač: Bohatier 6 sa hrozivo týči.
```

z2132. Z krajiny Lemmingov

Lemmingovia sú malí ľuďkovia vysokí asi tak pol centimetra. Napriek ich veľkosti radi stavajú interkontinentálne diaľnice, kopú tunely a stavajú mosty. Žijú v krajine plnej kopcov a dolín, kde sa majú možnosť vo svojich stavbárskych chúťkach plne realizovať. Aj teraz sa im zachcelo – radi by postavili ďalšiu diaľnicu. Samozrejme radi by diaľnicu postavil čo najlacnejšie. Ale keďže pri veľkosti ich mozgovní nie je medzi nimi žiaden ekonóm, musíte im pomôcť vy.

Úloha: Daný je popis krajiny (kopcov a dolín), kde žijú Lemmingovia, ako bočný prierez. Dná dolín sú na úrovni mora a všetky kopce majú rovnakú výšku. Na vstupe je daný počet kopcov n, ich výška h a následne $2n+1$ čísel: súradnice všetkých dolín a kopcov. (Pozri obrázok k príkladu.)

Lemmingovia chcú postaviť vodorovnú dialnicu naprieč krajinou v nejakej nadmorskej výške. Ak diaľnica prechádza cez kopec, treba cezeň prekopať tunel, ktorý v závislosti od jeho dĺžky d stojí d^2 korún. Podobne je to s dolinami. Ak diaľnica prechádza ponad dolinu, tak treba ponad ňu postaviť most, ktorý v závislosti jeho dĺžky d a jeho nadmorkej výšky v stojí dv korún. Zistite nadmorskú výšku, v ktorej treba postaviť diaľnicu, aby náklady na jej stavbu boli minimálne.



Príklad:

Vstup:

```
3 1000
0 800 1000 1100 1300 1600 1900
```

Výstup:

```
Optimálna výška:
432.8 metrov nad morom
```

z2133. Zeofína v záhrade

Naša stará známa korytnačka Zeofína dlho oddychovala a darilo sa jej nachádzať si zábavky, pri ktorých nepotrebovala pomoc programátorov. Až teraz...

Rozhodla sa, že si vysadí celú záhradku trávou, aby mohla hrávať po obede golf. Najrýchlejším riešením by bolo kúpiť trávový koberec, ale ten sa predáva po štvorcových metroch. A jej záhradka je všelijaká, ale štvorcová rozhodne nie je. Má síce všetky strany rovné, akurát že ich má hodne veľa a hádam každá vedie iným smerom. Zeofína sa rozhodla, že zistí, ako presne vlastne tá jej záhradka vyzerá, a potom poráta obsah. Tak zobrala do papuľky pero, do pravej prednej laby papier a sunie sa popri plote záhradky. Na papier si pritom zaznamenáva, ako sa hýbe, a to takto:

– **dopredu** k – pohla sa k korytnačích metrov dopredu (k je reálne),

– **doprava** u – otočila sa o u stupňov doprava,

– **doľava** u – otočila sa o u stupňov doľava

Teraz však prišla domov a zistila, že si už zo školy vôbec nepamätá vzorec na výpočet obsahu ľubovoľného n-uholníka. Tak jej neostáva nič iné, než požiadať vás o pomoc.

Úloha: Napíšte program, ktorý načíta postupnosť pohybov a vyráta Zeofíne, aký obsah má jej záhrada (v korytnačích metroch štvorcových).

Príklad:

Vstup:

```
dopredu 10 doprava 90 dopredu 10 doprava 90
dopredu 5 doprava 90 dopredu 5 dolava 90
dopredu 5 doprava 90 dopredu 5
```

Výstup:

```
75
```

z2134. Zúfalo hladný Janko

Kôň Janko má toho už dosť! Lenivá Monika mu už druhý týždeň nedoniesla obrok. Rozhodol sa, že od nej utečie a vydá sa hladať Dvanástich mesiačikov a ich slávne jahodové pole. Strašnou zimnou pľušťou, v ktorej mu omrzli všetky kopytá, sa predieral dlhé hodiny, kým ich našiel. Zaviedli ho do rohu jahodového poľa a dovolili mu preskákať krížom cez pole

a cestou si pochutnať na jahodách, čo mu brucho stačí. Pomôžte koňovi Jankovi nazbierať po ceste čo najviac jahôd!

Úloha: Na vstupe je počet riadkov r a počet stĺpcov s jahodového poľa. Nasleduje r riadkov vstupu a v každom riadku je s nezáporných celých čísel – počet jahôd, ktoré Janko nájde na príslušnom štvorcovom metri poľa. Janko začína na políčku v ľavom hornom rohu poľa a skončiť má v pravom dolnom rohu. Pohybuje sa samozrejme ako šachový kôň.

Mesiac Lipeň, ktorý Janka na pole pustil, nesmie vládnuť pridlho, aby sa neroztopil všetok sneh široko-ďaleko. Preto Janko cez pole stíha len prebehnúť – môže sa hýbať len tak, aby sa približoval do pravého dolného rohu (viď obrázok). Váš program by mal spočítať, koľko najviac jahôd môže Janko cestou zožrať.

Príklad:

Vstup:

```
5 8
0 0 0 0 0 3 0 0
0 0 1 0 0 2 0 0
0 0 1 0 2 0 1 0
1 0 1 1 0 0 0 0
0 0 0 0 4 0 0 1
```

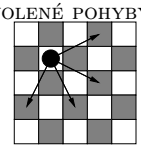
Výstup:

```
Dá sa zožrať 5 jahôd.

0 . . . . 3 . .
. . . 0 . . . .
. 0 . . . . 1 .
. . . . . . . .
. . . . . . . 1
```

(Toto je optimálna cesta.)

Povolené pohyby:



(Smer, v ktorom sa hýbe o 2 políčka, musí byť doprava alebo dole.)

z2135. Zo sveta steganografie

Akiste všetci poznáte kryptografiu, ktorá skúma spôsoby, ako danú správu zašifrovať tak, aby sa k nej nik okrem adresáta nedostal. Kryptografia však nepredstavuje jedinú možnosť, ako preniesť utajovanú informáciu. Inou cestou k úspechu je skryť prenášanú informáciu tak, aby nikto okrem adresáta ani nezistil, že nejaká správa bola poslaná. Informáciu musíme zamaskovať tak, aby vyzerala ako niečo iné – napríklad bloček od nákupu, obrázok alebo trebárs zadania KSP.

Antickí Gréci prišli s prvými nápadmi, ako na to: Zábavný nápad bol vytetovať správu otrokovi na oholenú hlavu, počkať kým mu vlasy narastú a poslať ho k adresátovi. Adresát mu znovu oholil hlavu a správu si prečítal. Časom boli samozrejme vymyslené aj ľahšie použiteľné metódy. Iste každý z vás už počul o rôznych neviditeľných atramentoch, však? Nezabúdajme ani na „motáky“ – listy z väzenia, ktoré museli dostať informácie cez cenzúru tak, aby si cenzor nič nevšimol. Ak by väzeň použil šifru, cenzorovi by bol list podozrivý a zničil by ho.

Obráťme však list a poďme do súčasnosti, veku informácií v digitálnej podobe. Otvára sa nám kopa nových možností. Len si treba uvedomiť, že informáciu môžeme skrývať aj do obrázkov, alebo trebárs aj do zvuku či videa. Napríklad v obrázku uloženom ako bitová mapa sú najnižšie bity natoľko bezvýznamné, že ich môžeme nahradiť čím chceme a rozdiel nebude pozorovateľný. Oko nerozozná rozdiel medzi farbou `#FCFCFC` a `#FFFFFF`, ale adresát dostane čo chcel.

Cieľom steganografie však zďaleka nie je iba prenášať tajné informácie s častokrát pochybným obsahom. Iné (legálne) použitie steganografie je digital watermarking – v obrázku, ktorý sme na počítači vytvorili, skryjeme informáciu, pomocou ktorej vieme dokázať, že sme ho naozaj vytvorili my.

Na záver už len dodáme, že častokrát aj v dátach, ktoré vyzerajú ako úplný šum, môže byť skrytá informácia. Ale samozrejme nemusí :-)

```
~~$3cb%J%omatfyz~~$3cb%J%omatfyz~~$3cb%J%omatfyz~~$3cb%J%omatfyz~~$3cb
45-.)usW/$!haluz45-.)usW/$!haluz45-.)usW/$!haluz45-.)usW/$!haluz45-.)u
_H^tu+nic_nie*je_H^tu+nic_nie*je_H^tu+nic_nie*je_H^tu+nic_nie*je_H^tu+
threedimensionalthreedimensioaltnhreedimensoaltnhreedimenisoaltnhreedi
MfH.5]#@d[:RlKSPMfH.5]#@d[:RKSPlMfH.5]#@d[:KSPlMfH.5]#@d[R:KSPlMfH.5]#
$byPwICkLf_haluz$byPwICkLf_aluhz$byPwICkLf_aluhz$byPwIkLfC_aluhz$byPwI
Uoto!^_ldtraaavaUoto!^_ldtaaarvaUoto!^_ldtaaarvaUoto!_ld^taaarvaUoto!_
Y:‘!‘eitYkalerabY:‘!‘eitYalekrab:‘!Y‘eitYalekrab:‘!Yeit‘Yalekrab:‘!Yei
)$-_dJ|Q-wB4 KSP)$-_dJ|QwB4- KSP$-_)dJ|QwB4- KSP$-_dJ|)QwB4- KSP$-_dJ|
{8n-g]+9‘traaava{8n-g]+‘traaava{8n-g]9+‘traaava{8ng]9-+‘traaava{8ng]9-
kO=bU[s|_traaavakO=bU[s_traaavakO=bU[|s_traaavakObU[=|s_traaavakObU[=|
|3B^o]~{q}matfyz|3B^o]~{q}matfyz3B^|o]~{q}matfyzB^|3o]~{q}matfyzB^|3o]
1NQHC([G[kalerab1NQHC([G[kalerabNQH1C([G[kaleraNQHb1C([G[kaleraNQHb1C(
%|%tu+nic_nie*je%|%tu+nic_nie*je|%t%u+nic_nie*e|%jt%u+nic_nie*e|%jt%u+
threedimensionalthreedimensionalhretedimensioalhnretedimensioalhnreted
-![gV]$*gkalerab-![gV]$*gkalerab![g-V]$*gkalrabe![g-V]$*gkalrabe![g-V]
~@F=’@n],pOhaluz~@F=’@n],pOhaluz@F=~’@n],pOhluza@F=~’@n],pOhluza@F=~’@
threedimensionalthreedimensionalthreedimensionalthreedimensionalthreed
D(vK_j2!haOhaluzD(vK_j2!haOhaluzD(vK_j2!haOhaluzD(vK_j2!haOhaluzD(vK_j
-Gt V*/?yJmatfyz-Gt V*/?yJmatfyz-Gt V*/?yJmatfyz-Gt V*/?yJmatfyz-Gt V*
```

Tak čo, našli ste **všetko**? Naozaj všetko? Tak nám to pošlite...

z2141. Zo života informačnej kancelárie II

V meste Kocúrkovo už dlho funguje informačná kancelária. A kedže ešte stále pri príchode ľudí ponúkajú pohárom chladeného piva, ľudí je tu stále dosť a stále sú tu rady. Avšak ľudí nebaví stáť v radoch a radi sa predbiehajú. To sa samozrejme tým predbiehaným nepáči, a tak dosť často vnikajú spory. Nový starosta sa rozhodol zabrániť sporom a zmodernizovať túto kanceláriu. Dal namontovať mašinku, ktorá rozdáva lístočky s poradovými číslami: 1, 2, 3, ... To ale nebolo všetko! Kedže je to Kocúrkovo, starosta zmenil aj číslovanie okienok a okienka označil všetkými prvočíslami. Každé okienko má svoju svetelnú tabuľu. Ak je okienko prázdne alebo sa práve uvoľnilo, na svetelnej tabuli sa zobrazí najmenšie poradové číslo čakajúceho občana, ktoré je deliteľné číslom okienka, a tento občan sa môže k dotyčnému okienku dostaviť. Vašou úlohou je pomôcť starostovi zrealizovať tento plán.

Úloha: Napíšte program, ktorý bude riadiť privolávanie čakajúcich pomocou svetelnej tabule. Na začiatku načíta číslo k udávajúce, že funguje prvých k okienok. Okienka sú očíslované prvými k prvočíslami.

Váš program bude na vstupe dostávať správy tvaru „`PRISIEL`“ a „`UVOLNILO SA` x“. Výstupom po správe „`PRISIEL`“ bude poradové číslo občana, ktorý práve prišiel. Správou „`UVOLNILO SA` x“ sa uvoľní okienko číslo x (kde x je nejaké z prvých k prvočísel). Váš program má vždy, keď sa uvoľní okienko a niekto čaká, zavolať k nemu čakajúceho občana s najmenším číslom, ktorý k nemu môže ísť. Ak sú v okamihu, keď príde občan, voľné niektoré z okienok, ktoré ho môžu obslúžiť, je rovno zavolaný k tomu z nich, ktoré má najmenšie číslo. Kedže kancelária funguje non-stop, váš program by mal tiež bežať bez prerušenia.

Príklad:

```
> K = 5
> PRISIEL
Prišiel občan číslo 1.
```

```
> PRISIEL
Prišiel občan číslo 2.
Občan s poradovým číslom 2 prosím k okienku 2.
> PRISIEL
Prišiel občan číslo 3.
Občan s poradovým číslom 3 prosím k okienku 3.
> PRISIEL
Prišiel občan číslo 4.
> UVOLNILO SA 3
> PRISIEL
Prišiel občan číslo 5.
Občan s poradovým číslom 5 prosím k okienku 5.
> PRISIEL
Prišiel občan číslo 6.
Občan s poradovým číslom 6 prosím k okienku 3.
UVOLNILO SA 2
Občan s poradovým číslom 4 prosím k okienku 2.
```

z2142. Zaba neblázni

Krotiteľka Zaba má vo svojej nemalej zbierke veľmi veľa zvieratiek. A nechýbajú jej tam ani žabky. Nacvičila s nimi jednoduché číslo: Žabky sedia na malých lavičkách očíslovaných od 1 do n. Vždy, keď Zaba tleskne, všetky žabky vyskočia a opäť na každú lavičku dopadne práve jedna z nich. Tlesk – a znova sedia ináč. Tlesk – a zase.

Tajomstvo, prečo sa nikdy žiadne žabky nezrazia, je jednoduché – pre každú lavičku je jednoznačne povedané, kam má žabka, ktorá na nej sedí, skočiť.

Úloha: Je dané číslo n = počet žabiek = počet lavičiek. Nasleduje n čísel, i-te z nich je číslo lavičky, na ktorú skáče žabka sediaca na i-tej lavičke. Tieto čísla sú navzájom rôzne. Napíšte program, ktorý zistí, či ešte niekedy budú žabky sedieť tak, ako sedeli na začiatku. Ak áno, zistite, najmenej koľkokrát musí Zaba tlesknúť, aby sa tak stalo.

Príklad:

Vstup:

```
7
2 4 3 1 7 5 6
```

Výstup:

```
Zaba musí tlesknúť 3-krát.
```

na začiatku: a b c d e f g
1. tlesk: d a c b f g e
2. tlesk: b d c a g e f
3. tlesk: a b c d e f g

z2143. Zo sveta motovodu

Motovod je ekologický dopravný prostriedok. Nechodí na benzín, ani na uhlie, ale energiu načerpá vždy, keď prejde po moste. Preto ho dopravný podnik vie využiť len na úsekoch, ktoré vedú ponad rieku.

Dopravný podnik jedného mesta má záujem nasadiť motovody na niektoré úseky liniek. Majú už nakreslenú mapu, kde sú všetky tieto úseky zaznačené. V ich mape ale nie je zakreslená rieka.

Už-už chceli zháňať geografa, ktorý by im rieku do mapy dokreslil, keď tu upratovačka Elza vyhlásila: „To je zbytočné, veď predsa nie je možné, aby všetky tieto úseky viedli ponad rieku!“

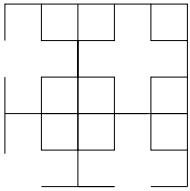
Úloha: Je daný počet rôznych zastávok n a počet dvojíc susedných zastávok m. Nasleduje m dvojíc čísel, ktoré predstavujú dvojice zastávok, medzi ktorými má priamo premávať motovod. Vašou úlohou je napísať program, ktorý zistí, či je teoreticky možné, že každé dve susedné zastávky ležia na opačných brehoch rieky.

Príklad:

Vstup:
```
7
8
1 4   1 7   2 3   2 4
3 5   3 6   4 5   6 7
```
Výstup:
```
Ano.
```
(Možno sú na ľavom brehu zastávky 1, 2, 5, 6 a na pravom všetky ostatné.)

Vstup:
```
3
3
1 2   2 3   3 1
```
Výstup:
```
Nie.
```
(Niektoré dve susedné zastávky určite ležia na tom istom brehu.)

z2144. Zeofína v záhrade II



Naša korytnačka má už peknú trávovú záhradu obdĺžnikového tvaru, a tak sa ju rozhodla kreatívne využiť. Namočila si chvostík do farby a na kúsku záhrady nakreslila štvorcovú sieť. Potom sa po nej poprechádzala a zmazala farbu zo všetkých štvorčekov, ktoré ležali v každom treťom stĺpci a v každom treťom riadku. (Počítať začína od ľavého dolného rohu.) Ostal jej taký útvar ako na obrázku.

Toto sa jej tak páčilo, že sa rozhodla nakresliť si takúto sieť väčšiu. Ale keďže už z malej má pomotanú hlavu, obrátila sa na vás.

Úloha: Napíšte program, ktorý na vstupe dostane čísla s, v a vypíše korytnačke návod, ako má kresliť, aby nakreslila vyššie popísaný obrazec so šírkou s a výškou v. Zeofína začína kresliť v ľavom dolnom rohu obrazca, natočená je dohora. Aby šetrila farbou, bola by Zeofína rada, keby kreslila každú časť iba raz. Chodiť po už nakreslenom však môže. Zeofína rozumie nasledujúcim príkazom:

- „`dopredu` k“: pohla sa k korytnačích metrov dopredu (k je reálne),
- „`doprava` u“: otočila sa o u stupňov doprava,
- „`doľava` u“: otočila sa o u stupňov doľava
- „`hore`“: zdvihne chost, namočený vo farbe, aby nezanechávala stopu
- „`dole`“: položí chvost, aby zanechávala stopu

Príklad:

Vstup:
```
2 3
```
Výstup:
```
dopredu 1; dole; dopredu 2; hore; doprava 180; dopredu 1;
dolava 90; dole; dopredu 2; dolava 90; dopredu 1;
dolava 90; dopredu 1; dolava 90; dopredu 2; dolava 90;
dopredu 1; dolava 90; dopredu 1; hore; doprava 180;
dopredu 1; dole; dopredu 1; doprava 90; dopredu 1;
```

z2145. Z denníčka

19.4.2004: Mám napísať zadanie do KSP-čka. Prečo ja??? Povedali, že sa priveľa hrávam. Nezmysel. A navyše ešte tá hlúpa stávka. Vraj sa nemám týždeň hrať. Dostanem za to kofolu. Ľahké. Len mám teraz nejako veľa času, zadanie sa napíše potom, zatiaľ si asi pozriem nejaký pekný film. Ostatní mi tvrdia, že je to lepšie ako hry. A vraj najlepšie je čítať nejakú knižku. Nuž, možno. Veď vyskúšam. No nič, nateraz končím.

20.4.2004: No, hry sú predsa len trochu lepšie ako filmy. Aspoň ten včerajší bol dóóóósť nudný. Pustila som to štyrikrát a naozaj si myslím, že prejsť posledným levelom bez nárazu sa nedá. A navyše, ovládač má dosť nešikovné tlačidlá. Sú malé, blízko seba, nedá sa to poriadne držať v dvoch rukách. Knihy sa tiež ukázali ako dobrý vtip. Tomuto hovoria grafika? Jeden obrázok (a ešte k tomu statický, predstav si to, milý denníček) na 100 stranách. Zvyšok hustý text. Ešte aj boje boli obkecané. Dostala som sa na 47. stranu a

stále som nevedela, koľko mám životov, ba ani za ktorú postavu hrám. Skrátka podvod. A ani sa neobťažovali s pekným introm. Ohó a spomenula som? Nemalo to žiadne zvuky. Vlastne ani nebolo kde pripojiť repráky. Idem spať. Dnes to naozaj nemá cenu.

21.4.2004: A ja viem? To sú nápady. Že vraj zadanie. Vraj ho mám napísať. Ale mne sa nechce, ja sa chcem hrať. Ach, hry moje milované, ako že mi chýbate. Dnes je deadline na zadanie. Ten lahodný zvuk motorov, slastná ozvena výstrelov. Inak, niekedy som fakt dobrá, to mi ver, milý denníček. Takto dnes poobede som si nenápadne, keď všetci odišli, pustila `<cenzurované - reklama>`. A dostala som ich všetkých. Dôležité je dôsledne sa kryť a pamätať si, kde sú všetky lekárničky.

Rozprávka? Ále, netreba. Hlavne, že bude popis. Hmmm, ale aký? Hm hm hmmmmm. Ahá už to mám! Našla som peknú hru. Volá sa Sokoban. Je to starinka, ale čo už. A má to málo levelov. Tak nech mi pošlú nejaké nové. Malé, ale ťažké. A je to.

22.4.2004: Zadanie som odovzdala neskoro, boli na mna jemne nas... nahnevaní. A k tomu som prišla o kofolu. Prichytili ma. Ale mám High Score! Hurá! Ide sa oslavovááááť!

Úloha: Úlohou je zostrojiť čo najťažší level Sokobanu. Vyzerá to asi takto: v štvorčekovej sieti je symbolmi „`#`“ ohraničená miestnosť. Znak „`#`“ označuje stenu a môže sa nachádzať aj niekde vo vnútri miestnosti. V miestnosti sú bedničky označené symbolom „`$`“ a panáčik „`@`“. Panáčik sa vie pohybovať hore, dolu, doprava a doľava. Ak pred panáčikom stojí bednička a pred ňou je prázdne miesto, panáčik vie bedničku posunúť. Cieľom je presunúť bedničky na miesta označené bodkami „`.`“. Je jedno, ktorá bednička skončí na ktorom cieľovom mieste. Ak na začiatku nejaká bednička už stojí na mieste označenom bodkou, použijeme pre takéto políčko symbol „`*`“.

Level je tým ťažší, čím je najkratšie riešenie dlhšie. T.j. pošlite nám level, ktorého najkratšie riešenie je čo najdlhšie. Ak si nie ste istí, ktorý z vašich levelov je lepší, môžete nám ich poslať až 5, hodnotí sa najlepší.

Aby to nebolo také ťažké, v miestnosti môžu byť najviac tri bedničky, voľná podlaha môže mať rozmery najviac 6×6. Panáčik nemôže začínať na mieste, kde má skončiť kocka.

Príklady ťahov:

Najľahší možný level a jeho riešenie jediným ťahom:

```
#####      #####
#.$@#  ->  #*@ #
#####      #####
```

Veľmi ľahký level a jeho riešenie na 10 ťahov:

```
########    ########    ########    ########    ########    ########
#.    *#    #.    *#    #.    *#    #.    *#    #.    *#    #.    *#
#  $   # -> #  $   # -> #  $   # -> #  $   # -> #  $@  # -> # $@   # ->
#     @#    #    @ #    #   @  #    #   @  #    #      #    #      #
########    ########    ########    ########    ########    ########

            ########    ########    ########    ########
            #.    *#    #.    *#    #.    *#    #*    *#
     ... -> #$@    # -> #$     # -> #$     # -> #@     #
            #      #    # @    #    #@     #    #      #
            ########    ########    ########    ########
```

Jeden ťažší level (ale zďaleka nie najlepší):

```
########
#.  . .#
#     ##
#  #  ##
#     ##
#$  $$ #
#  #  @#
########
```

2211. Ohne na Kapingamarangi

Ostrovy Kapingamarangi sú rozmiestnené dosť zvláštne. Tvoria obdĺžnik zložený z r radov po s ostrovoch. Z neznámych príčin ich niekto pooznačoval zo severu na juh a zo západu na východ súradnicami $[1,1]$ až $[r,s]$. V strede každého ostrova sa nachádza veľké ohnisko, od ktorého závisí život ľudí na ostrove. Podľa zlej kliatby totiž môže na každom ostrove horieť najviac jeden oheň, a preto môžu ľudia variť len na jednom mieste na ostrove. Zlá kliatba však má dodatok, že keď niekto zapáli alebo zahasí oheň (teda oheň zmení svoj stav) na ostrove so súradnicou $[i,j]$, aj ohne na ostatných ostrovoch v obdĺžniku s ľavým horným rohom na tomto ostrove a pravým dolným rohom na ostrove $[r,s]$ zmenia svoj stav.

Ešte prednedávnom horeli všetky ohne a nikto nevedel o zlej kliatbe. No keď sa požiarnici na ostrove Nuoruko rozhodli, že budú trénovať hasenie na jedinom ohni na ostrove, začali sa veľké nepokoje. Na ďalších ostrovoch sa ohne zhasli tiež, čo vyvolalo sériu zapaľovaní ohňov. Na niektorých ostrovoch sa v súčasnosti oheň zapaľuje a zhasína v pravidelných 10-sekundových intervaloch.

Panovník Kapingamarangi sa rozhodol situáciu riešiť. Keďže sa dozvedel o zlej kliatbe, chce dať všetko do pôvodného stavu. Zakázal komukoľvek zapaľovať alebo zahasovať oheň a požiadal vás, aby ste mu poradili, ako pozapaľovať ohne na všetkých ostrovoch na čo najmenej krokov.

Úloha: Sú dané čísla r a s a matica $r \times s$ zložená z núl a jednotiek. Nula predstavuje zahasený oheň, jednotka zapálený oheň. Na ostrove so súradnicou $[i,j]$ sa dá zmeniť stav všetkých ohňov na ostrovoch v obdĺžniku s ľavým horným rohom $[i,j]$ a pravým dolným rohom $[r,s]$. Úlohou je zistiť najmenší počet zmien potrebných na to, aby horeli všetky ohne.

Príklad:

Vstup:

```
3 4
1 0 1 1
1 0 0 1
0 1 0 1
```

Výstup:

```
6
```

2212. O Semaforovej rally Kocúrkovo 2004

Dámy a páni, páni a dámy! Vitajte na preslávenej Semaforovej rally Kocúrkovo 2004! Iba tu v Kocúrkove môžete sledovať tento skvost ducha a sami okúsiť vzrušenie, ktoré vám neponúkne ani bratislavská MHD! Stante sa priamymi účastníkmi exhibície a súťažte po boku takých velikánov akými sú Rejdi Burner, Maestro Kewo alebo Topless Tono! Neváhajte a vyneste aj vaše meno medzi hviezdy!

Úloha: Na vstupe je kladné celé číslo n – počet križovatiek a kladné celé číslo m – počet ulíc medzi nimi. Ďalej nasleduje riadok s dvoma kladnými celými číslami b a e, ktoré hovoria, v ktorej križovatke je začiatok a v ktorej koniec cesty. Po tomto riadku nasleduje m riadkov s trojicami čísel a_i, b_i a t_i, ktoré znamenajú, že križovatka a_i je spojená s križovatkou b_i a prejdenie po nej trvá čas t_i. Na konci vstupu je riadok obsahujúci n kladných celých čísel s_i, pričom číslo s_i predstavuje periódu, v ktorej sa mení farba i-teho semaforu. Na začiatku svietia všetky semafory na zeleno. Potom po prejdení s_i času sa zmení farba na i-tom semafore na červenú a potom, znovu po prejdení času s_i, sa farba semaforu i zmení na zelenú. Vašou úlohou je nájsť takú cestu zo začiatočnej križovatky b do koncovej e, po ktorej bude cesta trvať najkratšie. Vypíšte poradie križovatiek, cez ktoré máte ísť, aby ste sa najrýchlejšie dostali do koncovej križovatky. Pokiaľ je takýchto riešení viac, vyberte si ľubovoľné.

Príklad:

Vstup:	Výstup:
4 5 1 4 1 2 1 1 3 3 2 4 5 3 4 2 2 3 7 1 3 2 6	1 2 4 *(Čas pre túto cestu je 6, nikde počas nej nečakáme. Rovnako dobrá cesta je aj 1-3-4. Pri nej prídeme do križovatky 3 v čase 3 a následne musíme zastať a počkať do času 4, kedy v nej naskočí zelená.)*

2213. O papagájoch

Bolo raz jedno kráľovstvo, kde bolo príliš veľa krotkých spotených papagájov. Skrotili ich dávni králi a dnes už nikto nevie načo. Legenda hovorí, že pomohli v bitke proti zlému čarodejníkovi Grupostanxovi. Každopádne teraz iba poskakujú po meste a každému lezú na nervy. Došlo to až tak ďaleko, že kráľovná prikázala zriadiť špeciálny inštitút na vyhubenie krotkých spotených papagájov (ŠIVKSP). Vedci v ŠIVKSP sa dlho snažili nájsť prostriedok, ktorým by papagájov pozabíjali a konečne to vyzerá, že svitá na lepšie časy. Vedec menom Stamolangax prišiel s objavom, že všetci dnešní papagáji pochádzajú zo spoločného prapapagája. Skúšal robiť pokusy s papagájmi a zistil, že ak povie slovo vytvorené z mena prapapagája, tak, že sa bude čítať rovnako odpredu aj odzadu (takému slovu hovoríme palindróm), tak papagáj sa jednoducho zblázni a sám si rozbije hlavu o najbližší tuhý predmet. Lenže problém je, že neexistuje univerzálne slovo, ktoré treba papagájom povedať, niektoré reagujú na iné palindrómy ako iné papagáje. Dokonca rozdielne reagujú aj na palindrómy, ktoré znejú síce rovnako, ale boli inak vytvorené. Preto potrebujú vedci z ŠIVKSP vašu pomoc, aby zistili koľko palindrómov sa dá z mena prapapagája vytvoriť.

Úloha: Na vstupe je zadané meno prapapagája zložené z malých písmen anglickej abecedy. Úlohou je vypísať koľkými spôsobmi sa z neho dá vytvoriť palindróm, ak môžeme vyškrtávať niektoré písmenká, pričom na poradí vyškrtávania nezáleží. Slovo zložené z nula písmen sa za palindróm nepovažuje.

Príklad:

Vstup:	Výstup:
aba kalbal	5 12

(V prvom príklade existujú tieto riešenia: môžeme vyškrtnúť 1. a 2. písmenko, alebo 1. a 3., alebo 2. a 3., alebo 2., alebo žiadne.)

2214. O hre Scrobbli

Scrobbli je spoločenská hra, ku ktorej máte pribalenú hraciu dosku, písmenká a vrecúško, z ktorého sa písmenká vyťahujú. V každom kole si hráč vytiahne vopred daný počet písmeniek. Úlohou je z písmeniek zostrojiť čo najviac slov, ktoré sú v Slovníku zmysluplných slov. Miško sa rozhodol, že bude trošku podvádzať, takže použije program, ktorý mu bude všetky vhodné slová vyhľadávať. Pomôžte mu napísať takýto program.

Úloha: Sú dané čísla n, k a m – počet slov v slovníku, počet písmen a počet kôl hry. Ďalej je daný slovník. Každé z n slov v slovníku má presne k písmen. Na záver vstupu je pre každé z m kôl hry daných k písmen, ktoré si Miško v tom kole vytiahol. Napíšte program, ktorý pre každé kolo vypíše všetky slová zo slovníka, ktoré vzniknú poprehadzovaním vytiahnutých písmen.

Príklad:

Vstup:	Výstup:
`12 5 3`	
`halda letmo otlak malta`	
`kniha torta mleto tlama`	
`motor modla motel dlaha`	
`dalha`	`1: halda, dlaha`
`matal`	`2: malta, tlama`
`omlet`	`3: motel, letmo, mleto`

2215. Ovez fmtkojgíbdz

V Národnej banke Svazijska (NBS) mali donedávna velikánsky starý trezor, v ktorom bola kopa diamantov. Do trezoru mali prístup len členovia správnej rady NBS. Ale to nebolo len tak. Keďže ani členovia správnej rady si navzájom príliš neverili, vyriešili to tak, že dvere trezoru ovešali kopou zámkov a vhodne si od nich rozdali kľúče. Výsledkom bolo, že do trezoru mohli vstúpiť len vtedy, ak spomedzi všetkých siedmich členov boli prítomní aspoň štyria.

Nepovinná domáca úloha: Koľko najmenej zámkov na to bolo treba a ako rozdať kľúče?

Problémy nastali minulý štvrtok, keď svazijská vláda s okamžitou platnosťou schválila novelu zákona o ochrane diamantov. NBS musela zväčšiť počet členov správnej rady a navyše kúpiť nový namakaný digitálny trezor. Ale beda, na nový trezor sa zámky namontovať nedali. Otvoril sa len vtedy, keď sa mu na klávesnici zadalo správne heslo – jedno číslo z daného rozsahu.

Keď posledný pokus o namontovanie zámkov zlyhal, rozhodli sa technici NBS prečítať si manuál. K trezoru bola priložená útla knižka, ktorá všetko potrebné vysvetlila:

Nech má správna rada n členov a nech na otvorenie trezoru treba ľubovoľných k z nich. Očíslujme členov správnej rady od 1 do n. Heslom je číslo z rozsahu od 0 do $p-1$ (kde p je prvočíslo väčšie ako n).

Keď sa uzavrú dvere trezoru, ten si náhodne vygeneruje heslo h. Následne si ešte vygeneruje $k-1$ celých čísel $a_1, \ldots, a_{k-1}$ (tiež z rozsahu od 0 do $p-1$). Všimnime si teraz nasledovný polynóm:

$$f(x) = a_{k-1}x^{k-1} + \cdots + a_2x^2 + a_1x + h$$

Zjavne $f(0) = h$. Trezor pre každé i oznámi i-temu členovi správnej rady jeho poradové číslo i a hodnotu $t_i = f(i) \bmod p$.

Úloha:

a) Ukážte, že keď sa stretne aspoň k členov správnej rady, vedia jednoznačne vypočítať heslo h.

b) Ukážte, že keď sa stretne menej ako k členov správnej rady, nevedia o hesle h povedať vôbec nič – teda že pri tom, čo vedia, každé číslo môže byť heslom a žiadne z nich nie je pravdepodobnejšie ako iné.

c) Zamyslite sa, aké problémy má tento spôsob zabezpečenia prístupu do trezoru. Čo sa napríklad stane, ak by sa stretlo k členov správnej rady, ale jeden z nich bude chcieť ostatných oklamať?

Možno hint. Ľahko vieme sčítať, odčítať a násobiť modulo p. (Napríklad ak $p = 7$, tak $4 + 6 = 3$ a $3 \times 4 = 5$.) Ako je to ale s delením? Ak je p prvočíslo, tak ku každému celému x od 1 po $p-1$ existuje práve jedno y z toho istého rozsahu také, že xy dáva po delení p zvyšok 1. (Prečo je tomu tak?) Takéto y voláme inverzný prvok k x a značíme ho x^{-1}. Ak niekedy potrebujeme deliť číslom x, budeme namiesto toho násobiť číslom x^{-1}. Presvedčte sa, že takto definované delenie má dostatočne podobné vlastnosti ako klasické delenie reálnych čísel.

2221. Odvážny Marek

„Nie! Tá nie je ani omylom najlepšia," povedal Marek Mirkovi a zároveň dodal: „Stavím sa s tebou o mrežovník, že nájdem súvislú podpostupnosť, ktorú keď sčítam a z výsledku spravím absolútnu hodnotu, budem bližšie k 7 ako ty s tou tvojou."

Kedže Marek je strašná „lama" a nechce byť Mirkovi dlžný ďalší mrežovník (už teraz mu dlží dva), pomôžte mu a vyriešte úlohu za neho.

Úloha: Dané sú kladné čísla n a t a postupnosť n celých čísel $a_1, a_2, \ldots, a_n$. Nájdite v nej súvislú podpostupnosť takú, že absolútna hodnota jej súčtu je čo najbližšie k číslu t. Teda vypíšte indexy i, j také, že ak $s = |a_i + a_{i+1} + \cdots + a_j|$, tak rozdiel $|t - s|$ je najmenší možný.

Príklad:

Vstup:	Výstup:
`13 12`	`3 4`
`47 100 -1 -9 8 -7 6 -5 4 -3 2 -1 0`	

(Najlepším riešením je podpostupnosť tvorená hodnotami -1 *a* -9*.)*

2222. Ojedinelý prípad

Neviem, či viete, ale Mirko je dosť ojedinelý prípad, on je totiž strašne lakomý (dokonca sa nepodelí ani s diskom). Ale aby si to o ňom jeho kamaráti nemysleli, rozhodol sa ich na svoje narodeniny pozvať do kina. Každý kamarát má požiadavky, koho musí pozvať s ním. Mirko by ich chcel pozvať čo najmenej (aby ušetril), ale chce, aby všetci pozvaní boli spokojní.

Úloha: Je daných n Mirkových kamarátov, a ku každému zoznam ľudí, ktorých chce, aby boli v kine s ním. Vypíšte, koho musí Mirko pozvať, aby s každým pozvaným pozval aj všetkých ľudí z jeho zoznamu, a zároveň aby bol počet pozvaných ľudí kladný (musí pozvať aspoň jedného človeka) a minimálny možný. Pre zjednodušenie sú mená nahradené číslami od 1 po n.

Vstup:		Výstup:
`6`		`4 5 6`
`1: 2`	`4: 5`	
`2: 1 3`	`5: 6`	
`3: 4`	`6: 4`	

2223. OEIN GONIATH

Temná noc. Záblesk na oblohe. Len mohutný dub sa čnie do noci. Majestátne konáre sa kolíšu vo vetre. Ďalši záblesk. Oči pútnika padnú na predmet pod stromom. Tma. Pútnik podíde bližšie. Vyčkáva, čo bude ďalej. O chvíľu sa rozhodne podísť ešte bližšie k nádhernému starému stromu. Konečne si začína uvedomovať zvuk hromu a hučanie vetra. A v tom aj niečo iné. Nejaké šuchotanie. Ďalší záblesk. Pútnik zbadá, čo leží pod stromom. Je to stará kniha. Listy sú celkom zožltnuté, ale vyžaruje z nej rovnaká sila ako zo samotného dubu. Zrazu ho premkne zvláštny pocit. Niečo ho núti ísť ku knihe, napriek tomu, že má obrovský strach. Pútnikovi sa chvejú ruky. Nakláňa sa k nej. Vtom zo stromu vyletia havrany, na krídlach nesúc zvuk hromu, čo zaznel z diaľky. Pútnik si to však už nevšíma a otvára knihu. Je v nej strašné tajomstvo. Pravda o živote a smrti...

„*ETH GO DIVADH!*", takto káže kniha. „Priveď druidov k čiare života, *LOINE TÁ CAILIDH*. Nech každý z nich postaví sa na ňu a silami štyrmi strom svoj povolá. Silou zeme, *SORGAEIL TAAMH*, silou vody, *SORGAEIL UISCE*, silou ohňa, *SORGAEIL TEINE* a silou vzduchu, *SORGAEIL ANAIL*. Sily štyri sú medzi druidom s stromom jeho, len kým strom jeho voľnú cestu rásť má, *OTEITH AD TA RAHM*. Odpovedz človek, *DREAT REFOIBH*, zo stromov ktoré až do diaľneho neba porastú a naopak, druidov ktorých sily štyri opustia. Odpovedz dobre a večný život, *WICCA A TÚ ARGAN*, bude ti daný. Zmýľ sa však a v muky pekelné uvedieš sa navždy!"

Úloha: Na vstupe je zadaný počet druidov n. Všetci stoja na x-ovej osi a každy má zadanú svoju pozíciu a smer, ktorým rastie jeho strom. Smer je zadaný v stupňoch proti smeru hodinových ručičiek od kladnej x-ovej osi. Všetky stromy rastú rovnakou rýchlosťou. Keď strom narazí do iného stromu, jeho druida opustí sila a strom prestane rásť. Ak sa zrazia vrcholky dvoch stromov, ďalej porastie ten, ktorý vyrastá zo zeme viac vľavo. Stromy sú očíslované od 1 do n v poradí, v akom sú uvedené na vstupe. Vypíšte čísla stromov, ktoré porastú až do neba, t.j. nikdy neprestanú rásť.

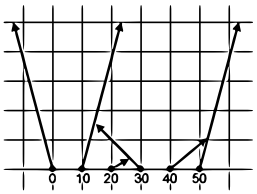
Príklad:

Vstup:

```
6
pozície:   0  10  20  30  40  50
smery:   105  75  30 135  40  75
```

Výstup:

```
1 2 6
```



2224. O 47. hyperpriestorovej dimenzii

V 47. hyperpriestorovej dimenzii práve dokončili cestnú sieť, ktorú budovali veľmi veľa rokov. Kedysi dávno boli všetky mestá tejto dimenzie od seba izolované. Všemocný Pán dimenzie vtedy rozhodol, že každý rok dá postaviť práve jednu cestu. Navyše táto nová cesta musela vždy pripojiť k existujúcej cestnej sieti nové mesto. A keďže v 47. hyperpriestorovej dimenzii platia celkom odlišné zákony, niektoré postavené cesty mali aj zápornú dĺžku. Teraz sú už všetky mestá spojené, no ľudia aj tak nie sú spokojní. Cesta medzi niektorými mestami totiž trvá veľmi dlho. Pán dimenzie sa preto rozhodol, že ešte postaví jednu cestu, ktorou spojí tie dve mestá, medzi ktorými teraz trvá cesta najdlhšie

Úloha: Je daná súvislá cestná sieť s n mestami a práve $n-1$ cestami. Najprv je na vstupe prirodzené číslo n, potom $n-1$ ciest. Každá cesta je určená tromi číslami a_i, b_i, d_i, kde a_i, b_i je dvojica miest, medzi ktorými vedie cesta a d_i je dĺžka tejto cesty. Dĺžky niektorých ciest môžu byť aj záporné. Vašou úlohou je nájsť maximum zo vzdialeností všetkých dvojíc miest.

Príklad:

Vstup:

```
6
1 2 4
2 3 2
2 4 -3
4 6 8
4 5 0
```

Výstup:

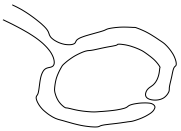
```
9
```

(Najdlhšia je trasa $1 \to 2 \to 4 \to 6$, *jej dĺžka je* $4 + (-3) + 8 = 9$.*)*

2225. Otvor sa, sezam!

Kdesi v ďalekom Oriente odohral sa raz nasledujúci príbeh: Jedného slnečného dňa vydal sa Alibaba na trh. Ako sa tak prechádza po trhu, zrazu mu malý zlodej uchmatne peňaženku a poďho kade ľahšie! Alibaba neváhal a bežal za ním. No zlodej vybehol z mesta a vbehol do akejsi jaskyne. Alibaba vbehol za ním, no po pár metroch sa chodba delila na dve vetvy. Čo mal chudák robiť, jednu z nich si vybral a vydal sa ňou. Chodba po chvíli končila a zlodeja nikde.

Na druhý deň zamieril Alibaba opäť na trh – kúpiť si novú peňaženku. No len čo si ju kúpil, už mu s ňou známy zlodej utekal preč. Tentokrát si Alibaba v jaskyni vybral druhú cestu, no aj tá končila stenou a zlodeja ani teraz nechytil.



Keď takto Alibaba prišiel už o šiestu peňaženku, začalo mu to vŕtať v hlave. Vrátil sa do jaskyne a poriadne ju preskúmal. Nakreslená je na obrázku.

Ako tak stál na konci jednej z chodieb, zrazu začul dupot nôh. A len čo sa stihol skryť za neďaleký výčnelok skaly, pribehol jeho starý známy. No to, že chodba končí, ho nijako nezastavilo. Vykríkol „otvor sa, sezam“ – a div sa svete, stena sa otvorila a zlodej spokojne prebehol do druhej chodby.

Alibaba sa zamyslel. Z tohto by sa predsa dali vytĺcť slušné peniaze! Na druhý deň už volal do televízie, že vie prechádzať cez steny. Do hodiny stál pred jaskyňou celý štáb. Jaskyňu dôkladne preskúmali a mohlo sa začať s prechádzaním cez stenu. No Alibaba nechcel, aby niekto videl, ako to robí. Preto navrhol nasledujúci postup:

„Ja vôjdem do jaskyne a zaleziem do jednej z tých dvoch chodieb. Chvíľu po mne prídete vy s kamerou, stanete si na rázcestie a zakričíte, z ktorej strany mám prísť. A ja z nej naozaj prídem.“

Ako povedal, tak spravili. A keď celý tento pokus zopakovali tridsaťkrát a zakaždým Alibaba vyšiel zo správnej strany, štáb žasol. Toto bola bomba, ktorá pôjde do televíznych novín.

Tu by náš príbeh mohol končiť... Nebyť toho, že sa o tom celom dozvedeli dvaja zamestnanci konkurenčnej televízie. A tí si povedali: Veď na tom predsa nič nie je. Spravíme aj my také pokusy, celé to natočíme a v štúdiu potom vystriháme tie pokusy, ktoré sa nepodarili. Ako si povedali, tak spravili. A v ten večer obe televízie odvysielali na chlp podobné príspevky. A z divákov nik neuveril, že Alibaba naozaj vedel prechádzať cez steny. The end.

To, čo práve predviedol Alibaba, bol takzvaný *bezznalostný dôkaz*. Dokázal presvedčiť štáb, že prechádza cez steny, lebo odpovedal na ich výzvy – ale pritom sa nik nedozvedel nič o tom, ako to vlastne robí.

Formálne, bezznalostný dôkaz je postup komunikácie medzi dvomi stranami. Jeden z nich (P – Paľo Prover) chce druhému (V – Vlado Verifier) dokázať platnosť nejakého, väčšinou matematického, tvrdenia – ale nechce prezradiť vôbec nič iné. Bezznalostný dôkaz musí mať tri vlastnosti:

- **Funkčnosť:** Ak tvrdenie platí a P dodržuje dohodnutý postup, V mu uverí.
- **Správnosť:** Ak tvrdenie neplatí, podvodník tváriaci sa ako P nemôže (skoro nikdy) V presvedčiť, a to ani keby podvádzal a nedodržiaval dohodnutý postup.
- **Bezznalostnosť:** Ak tvrdenie platí, V (ani nik počúvajúci komunikáciu) sa o ňom nedozvie nič iné.

Načo sú takéto bezznalostné dôkazy dobré? Uveďme jeden príklad za všetky: Prihlasujete sa na vzdialený počítač. Aby vás „pustil dnu“, musíte ho presvedčiť, že poznáte heslo. Ale nechcete mu toto heslo jednoducho povedať – čo, ak vás niekto odpočúva? To, čo potrebujete, je práve bezznalostný dôkaz. Počítač na druhom konci kábla presvedčíte, že heslo poznáte, ale nik iný sa o vašom hesle nesmie vôbec nič dozvedieť.

Ukážme si príklad serióznejšieho dôkazu ako bol ten Alibabov. Paľo má veľký graf G (s n vrcholmi, očíslovanými od 1 do n), ktorý si kedysi zostrojil a vďaka tomu pozná jednu hamiltonovskú kružnicu v tomto grafe. Chcel by Vladovi dokázať, že v grafe G sa takáto kružnica nachádza – ale zároveň chce, aby zostal jediný, kto ju pozná. (Hamiltonovská kružnica je kružnica prechádzajúca cez všetky vrcholy, cez každý práve raz. Nie je známy algoritmus, ktorý by v polynomiálnom čase zistil, či daný graf takúto kružnicu obsahuje.)

Čo ale má Paľo robiť? Zjavne nemôže Vladovi kružnicu len tak ukázať. Riešenie je napríklad nasledujúce: Paľo náhodne označí vrcholy svojho grafu číslami od $n+1$ do $2n$. Do jedného súboru spíše priradenie nových čísel starým. Teraz pre každú hranu zostrojí súbor, ktorý bude obsahovať nové čísla jej vrcholov. Každý z týchto súborov zašifruje iným heslom a všetky ich zverejní. Podobne, ako si kameraman volil, odkiaľ má Alibaba vyjsť, si teraz Vlado môže vybrať jednu z dvoch možností:

– Chce vidieť, že Paľo naozaj zverejnil to, čo mal. Vtedy mu Paľo prezradí všetky heslá, Vlado si dešifruje súbory a skontroluje, že je to naozaj graf G.
– Chce vidieť hamiltonovskú kružnicu. Vtedy mu Paľo prezradí heslá len pre hrany, ktoré ju tvoria a Vlado si overí, že naozaj ide o kružnicu s n vrcholmi.

Toto celé opakujú, kým Vlado nie je presvedčený, povedzme tridsaťkrát. Funkčnosť tohto postupu je evidentná – ak Paľo naozaj pozná hamiltonovskú kružnicu, vie tento postup dodržiavať a tým Vlada presvedčiť.

Ako je to so správnosťou? Predstavme si, že Peter (Paľovo zlé dvojča), chce Vlada presvedčiť, že pozná hamiltonovskú kružnicu v G, hoci to nie je pravda. V každom kole má najviac 50% šancu, že Vlada oklame – buď správne vyrobí zašifrované súbory (ale nepozná kružnicu), alebo vyrobí iný graf, v ktorom kružnicu pozná. Po 30 kolách je ale šanca, že stále Vlada oklamal, už len 2^{-30}.

A bezznalostnosť? Tu si pomôžeme argumentom, ktorý sa bude podobať na to, čo spravila konkurenčná televízia. Predstavme si, že by si nezávislý pozorovatel N zapisoval všetko, čo si P a V pošlú. Dostane takto postupnosť trojíc: (Paľom zverejnené súbory, Vladova voľba, Paľova odpoveď). My však bez akýchkoľvek vedomostí o hamiltonovskej kružnici v G vieme vygenerovať postupnosť trojíc, ktorá bude tejto na nerozoznanie podobná. Preto celá komunikácia, ktorá medzi nimi prebehla, neobsahuje žiadne informácie o našej kružnici.

Úloha:

a) Poriadne napíšte, ako by sme takúto postupnosť generovali.
b) Paľo má dva grafy a chce dokázať, že nie sú izomorfné. Navrhnite bezznalostný dôkaz a stručne zdôvodnite, že má požadované vlastnosti.
c) Paľo má dva grafy a chce dokázať, že sú izomorfné. Navrhnite bezznalostný dôkaz a stručne zdôvodnite, že má požadované vlastnosti.

V úlohách b a c predpokladajte, že Paľove výpočty môžu trvať ľubovoľne dlho (t.j. vie skúšať všetky možnosti), ale Vlado má k dispozícii len polynomiálny čas od veľkosti grafov (t.j. jeho výpočty musia byť efektívne).

Poznámka: Hovoríme, že dva grafy sú izomorfné, ak vieme vrcholy jedného z nich preznačiť tak, aby sme dostali graf identický s druhým z nich. V súčasnosti nepoznáme efektívny algoritmus, ktorý by rozhodol, či sú dané dva grafy izomorfné.

Hint k druhej podúlohe: Ako by ste dokázali farboslepému človeku, že dve kartičky sú rôznych farieb?

2231. O hvezdárovi bez oka

Určite ste už počuli o tajomnom Belgaratovi. Áno, je to ten Belgarat, ktorý privodil skazu celému východnému Anghornu. Tak práve tento Belgarat raz zavítal na dvor kráľa Forna, keď ho hľadanie mocného artefaktu zavialo do týchto končín. Ako to už v podobných príbehoch býva, vrahom je vždy záhradník a majiteľom artefaktu kráľovský hvezdár. Belgarat sa teda urýchlene vybral do najvyššej veže na hrade.

Hneď ako otvoril dvere, do nosa mu vrazil prenikavý zápach čpavku a síry. Hvezdár na to: „Dnes už hvezdárstvo nič nevynáša, musel som začať s alchýmiou.“ Belgarat pri zvuku toho chrapľavého hlasu vyskočil a o chvíľu, keď uvidel, komu hlas patrí, takmer spadol z nôh. Hvezdárova postava bola celá pokrútená, tu noha, tam ruka a namiesto jedného oka mu žiaril obrovský drahokam. Tu Belgarat videl, že konečne našiel to, čo hľadal, posvätný Aldurov orb. „Ó, veľký hvezdár, múdrosť prebýva na tvojom čele a tvoje roky sú mnohé...“ „No že no, tie rečičky si nechaj pre iných, vidím na tebe, že si mi prišiel zobrať oko a pravdu povediac, už by mi to aj dobre padlo, ale nemôžem ho vybrať, kým sa nezbavím kliatby.“ Belgarat sa ochotne ponúkol, že on ten problém vyrieši.

Hvezdár teda spustil: „Vždy keď je spln, príde za mnou duch morského leva a žiada odpoveď na svoju otázku. Zakaždým vyberie niekoľko hviezd a on, lev, chce vedieť, ako

ďaleko je od nás tá s poradovým číslom k. Naviac hviezdy sú to praďaleké a ja som už starý hvezdár. Pamäť mi neslúži, ich presnú vzdialenosť si už nepamätám. Skúšal som sa leva pýtať, no ten, ľuha, len keď mu dve hviezdy poviem, tak mi odpovie vzdialenosť tej bližšej z nich. No ja ani neviem, ktorá z tých dvoch to je! A lev sa len baví, a vždy, keď mi to príliš dlho trvá, pričaruje mi nový prst. Už teraz ich mám 47!“ Tu si Belgarat vzdychol a nakoľko si s touto úlohou nevedel dať rady, otvoril mocný portál do nášho sveta a obracia sa na vás.

Úloha: Napíšte program, ktorý načíta zo vstupu dve čísla n a k a potom bude klásť levovi otázky. Číslo n udáva počet hviezd, tie sú očíslované od 1 po n. Váš program má zistiť a vypísať vzdialenosť hviezdy číslo k od nás. Ak sa túto vzdialenosť nedá určiť (bez ohľadu na to, koľko otázok levovi položíte), program to musí zistiť a podať o tom správu.

Každá otázka musí byť tvaru „`V akej vzdialenosti je bližšia z hviezd` i `a` j`?`“, pričom i a j musia byť **rôzne**. Odpoveďou na takúto otázku bude vždy jedno prirodzené číslo – vzdialenosť bližšej spomedzi hviezd i a j od nás.

Môžete predpokladať, že lev vždy odpovie pravdivo, a že žiadne dve hviezdy nemajú od nás rovnakú vzdialenosť.

Vstup:

```
> 4 1
V akej vzdialenosti je bližšia z hviezd 1 a 2?
> 31
V akej vzdialenosti je bližšia z hviezd 1 a 3?
> 9
V akej vzdialenosti je bližšia z hviezd 2 a 3?
> 9
V akej vzdialenosti je bližšia z hviezd 1 a 4?
> 31
V akej vzdialenosti je bližšia z hviezd 2 a 4?
> 32
```

Výstup:

```
31
```

(Z nami položených otázok tiež vyplýva, že hviezda 3 musí byť vo vzdialenosti 9.)

2232. Ortonormálny teleport

V hlbokom vesmíre v galaxii zvanej Húsenková dráha (nazvanej vraj podľa slávneho bádateľa Hú Senka) má svoje mesto planéta, ktorá sa volá Teleportka. Možno už tušíte, že toto meno nedostala náhodou. Obyvatelia tejto planéty sa totiž vedia premiestňovať z jedného miesta na druhé pomocou teleportu. Ale nemyslite si, že to majú úplne super vymakané! Kto chce používať na premiestňovanie teleport, musí úspešne absolvovať kurz teleportingu, a kto na to nemá, musí chodiť po svojich. Na kurze sa naučia, ako obsluhovať teleport a ako to celé funguje.

V každom z n miest Teleportky je jeden tzv. odpichový teleport. Ten má tvar kruhu a v strede sa nachádza riadiaci pult, pomocou ktorého sa Teleporťania premiestňujú. Postavia sa na kruh a stlačia tlačidlo označujúce cieľ, kam sa chcú dostať, a o sekundu sa už nachádzajú v mieste určenia. Bohužiaľ, z každého odpichového teleportu sa priamo dá dostať len na niekoľko iných. A to, že sa dá z mesta A teleportovať do mesta B, ešte neznamená, že sa musí dať aj z B do A.

Keďže Teleporťania mohli takýmto spôsobom precestovat za deň aj milión kilometrov, už si vôbec nelámali hlavu nad tým, či vôbec chodia najkratšou cestou. Napríklad keď sa chceli dostať z mesta A do mesta E, pokojne išli aj takto: z A do B, potom do C, potom do D, potom do B, potom do C a nakoniec do E, a nikomu nevadilo, že existuje aj kratšia cesta.

Úloha: Na vstupe je počet miest n, prirodzené čísla k a m a potom m dvojíc miest odkiaľ kam sa dá priamo teleportovať (dvojica a b hovorí, že sa dá skočiť z a do b). Napíšte program ktorý vypočíta, koľko existuje rôznych sledov teleportovania dĺžky práve k. Sled

je definovaný ako postupnosť miest $a_0, a_1, \ldots, a_{k-1}, a_k$, pričom pre všetky i od 0 do $k-1$ sa dá priamo teleportovať z mesta a_i do a_{i+1}.

Vstup:	Výstup:

`3 1234457542 3`
`0 1`
`1 2`
`2 0`
	`3`

Vstup:	Výstup:

`4 42 5`
`0 1`
`0 2`
`1 3`
`2 3`
`3 0`
	`65536`

2233. O mrakodrapoch

Malý Lukáš dostal na Vianoce m rovnakých kociek, s ktorými sa teraz celé dni hrá. Už z nich postavil všetko od katedrály po električku a pomaly ho to začalo nudiť. Vtom ho však napadlo stavať z kociek mrakodrapy a nie hocikoľko, ale práve n. Každý mrakodrap sa skladá z jedného stĺpca kociek a výšky mrakodrapov naviac tvoria zľava doprava neklesajúcu postupnosť.

Toto ho tiež začalo po chvíli nudiť a tak si to sťažil. Uvedomil si, že dve rozostavenia mrakodrapov sa dajú lexikograficky porovnať. Ak $a_1, \ldots, a_n$ a $b_1, \ldots, b_n$ sú dve rozostavenia, tak prvé je menšie ako druhé práve vtedy, keď sa na prvých $i-1$ pozíciach zhodujú a platí $a_i < b_i$. Lukáš by si rád postavil k-te rozostavenie mrakodrapov, lenže je ešte príliš malý na takúto matematiku.

Úloha:

Na vstupe sú tri čísla m, n a k. Úlohou je nájsť k-te rozostavenie n mrakodrapov, ktoré sú poskladané práve z m kociek.

Príklad:

Vstup:	Výstup:
`9 4 5`	`1 2 3 3`

(Predchádzajúce štyri rozostavenia sú (1 1 1 6), (1 1 2 5) a (1 1 3 4).)

2234. Opäť v 47. hyperpriestorovej dimenzii

Časť 47-rozmerného hyperpriestoru sa bude rozpredávať na hyperparcely. Dotyčná časť hyperpriestoru má tvar 17-rozmerného hyperkvádra, ktorého hrany sú rovnobežné so súradnicovými osami. Jednotková hyperkocka v „ľavom dolnom“ rohu tohto hyperkvádra má súradnice $[0, 0, \ldots, 0]$, tá v „pravom hornom“ rohu má súradnice $[9, 9, \ldots, 9]$. (Tento hyperkváder si môžete predstaviť ako 17-rozmerné pole, ktorého každý rozmer je 10.)

Hyperpozemkový úrad pripravil návrh rozdelenia tohto hyperkvádra na hyperparcely. Ale keďže v 17 rozmeroch sa ani hyperdivá srnka nevyzná, je dosť možné, že sú v návrhu chyby. Potrebovali by vašu pomoc.

Úloha:

Na vstupe je popis jednotlivých hyperparciel. Každá hyperparcela je zadaná sústavou podmienok. Podmienka má tvar $x?c$, kde x je súradnicová os (písmenko od `a` po `q`), ? je znamienko (`=`, `<`, `>`, `<=` alebo `>=`) a c je celé číslo od 0 do 9. Do hyperparcely patria tie jednotkové hyperkocky, ktorých súradnice spĺňajú všetky uvedené podmienky.

Váš program by mal postupne overiť:

– či má každá hyperparcela nenulový hyperobjem

– ak áno, či sa žiadne dve hyperparcely neprekrývajú

– ak nie, či hyperparcely dokopy pokrývajú celý hyperkváder

Príklad:

Vstup:		Výstup:
`1: "a>3 a>2",`	`2: "a<=3 b>5",`	OK
`3: "b<6 a<=3"`		

Vstup:		Výstup:
`1: "a=0 a=1",`	`2: "a>=2"`	Hyperparcela 1 je prázdna.

Vstup:		Výstup:
`1: "a>=5",`	`2: "q<=6"`	Hyperparcely 1 a 2 sa prekrývajú.

Vstup:		Výstup:
`1: "a>=5",`	`2: "a<=4 b>7",`	Nekompletné pokrytie.
`3: "a=4 c>=4 b=7",`	`4: "a<5 b<7",`	
`5: "c<2 a=4 b=7",`	`6: "a<4 b=7"`	

(V poslednom príklade zostali nepokryté hyperkocky s `a=4`, `b=7` *a* `2<=c<=3`*.)*

2235. Opatrne s XORovaním!

Na úvod sa stručne zoznámime s binárnou operáciou *xor* (exclusive or, po našom „buď alebo“). Budeme ju zapisovať $\oplus$. Pre hodnoty 0 (false, nepravda) a 1 (true, pravda) ju definujeme nasledovne:

$$0 \oplus 0 = 1 \oplus 1 = 0, \quad 0 \oplus 1 = 1 \oplus 0 = 1$$

Teda $x \oplus y = 1$ práve vtedy, ak práve jedna z premenných x, y je 1.

Teraz môžeme operáciu *xor* definovať aj pre ľubovoľné dve nezáporné celé čísla x, y nasledovne: Zapíšeme pod seba ich zápisy v dvojkovej sústave, kratší prípadne doplníme zľava nulami. Nech i-te cifry x, y sú x_i a y_i. Potom i-ta cifra výsledku je $(x_i \oplus y_i)$. Príklad:

$$17 \oplus 9 = (10001)_2 \oplus (01001)_2 = (11000)_2 = 24$$

Na operáciu *xor* sa môžeme pozerať ako na sčítanie v dvojkovej sústave, pri ktorom zabúdame robiť prenos od nižších rádov k vyšším. Všimnite si, že táto operácia je komutatívna, asociatívna a platí $x \oplus x = 0$.

Pokračovať budeme rozprávaním o pirátskej truhlici. Profesor Quilenius na jednej zo svojich početných výprav zablúdil do Karibiku. Ako áno, ako nie, po dlhom a finančne vyčerpávajúcom hľadaní objavil jednu truhlicu plnú zlatých dublónov. Lenže čo s ňou v ďalekej cudzine? Ideálne by bolo doviezť ju domov a tam ju spokojne predať... ehm, darovať múzeu. Lenže na letenku domov mu už nezostávalo peňazí.

Spomenul si ale na svojho priateľa profesora Jonesa, toho času sa nachádzajúceho v Anglicku. Keby mu truhlicu poslal, profesor Jones by sa o ňu určite dobre postaral a dostal by jeho, Quilenia, späť domov. Keď mu však truhlicu pošle len tak, skoro určite ju po ceste vykradnú! A naopak, ak ju poriadne zamkne, ani Jones ju neotvorí... Čo s tým?

Netrvalo dlho a v hlave profesora Quilenia sa zrodil plán – diabolsky rafinovaný, a pritom tak jednoduchý. Nasledoval telefonát s profesorom Jonesom, všetko si dohodli a tak aj spravili. O dva týždne už bol obsah truhlice v rukách profesora Jonesa a profesor Quilenius na ceste domov. Pýtate sa, ako to spravili? Jednoducho:

– Profesor Q. dal na truhlicu pevný zámok a poslal ju profesorovi J.
– Profesor J. truhlicu nevie otvoriť, ale zato na ňu vie pridať druhý, rovnako pevný zámok. Potom pošle truhlicu späť.
– Profesor Q. otvorí svoj zámok. Truhlica stále ostáva zamknutá zámkom profesora J. Pošle truhlicu späť.
– Profesor J. otvorí svoj zámok a má otvorenú truhlicu.

Keď si tento príbeh prečítali Alica s Bobom, zdalo sa im, že je to ideálne riešenie ich problému – ako si posielať správy bez toho, aby si ich mohla poštárka čítať. Vymysleli si nasledujúci postup:

– Alica zoberie správu M, ktorú chce poslať. Zapíše ju ako postupnosť bitov ASCII hodnôt znakov správy. Zvolí si kľúč K_A – náhodnú postupnosť bitov rovnakej dĺžky ako zápis M. Správu M prexoruje kľúčom K_A a pošle ju Bobovi.
– Bobovi prišla správa. Zvolí si rovnako dlhý kľúč K_B, prexoruje ním správu a pošle ju späť.
– Alici prišla správa, prexoruje ju K_A a pošle ju späť.
– Bobovi prišla správa, prexoruje ju K_B a prečíta si výsledok.

Cyril a Diana mali podobný problém. Cyril však prišiel na jednoduché riešenie. Keď sa raz s Dianou stretli, dohodli si kľúč K. Bezpečný prenos správy je teraz omnoho jednoduchší:

– Cyril zoberie správu M, ktorú chce poslať. Zapíše ju ako postupnosť bitov ASCII hodnôt znakov správy. Zoberie začiatok kľúča K rovnakej dĺžky a prexoruje ju ním. Výslednú správu S pošle Diane.
– Diane prišla správa. Zoberie začiatok kľúča K rovnakej dĺžky a prexoruje ju ním. Výsledok si prečíta.

Cyril ubezpečil Dianu, že tento spôsob je úplne bezpečný – poštárka vidí len S a o M nezistí nič okrem dĺžky.

Úloha:

a) Rozcvička 1: Ukážte, že výsledok, ktorý Bob dostane, je naozaj Alicina správa.
b) Rozcvička 2: Ukážte, že Cyril má pravdu. Ak poštárka pozná S, tak ku každej správe M' rovnakej dĺžky ako S existuje práve jeden kľúč K', ktorý by zašifroval M' na S.
c) Analógie nie vždy fungujú. Napríklad postup Alice a Boba nie je taký skvelý, ako sa zdá. Poštárka odpočula nasledujúcu komunikáciu medzi Alicou a Bobom:
A: `81 d5 62 63 83 4a 34 57 66 66 07 3d 78 39 39 96 9c 38 31`
B: `80 e0 ac bc 12 7e 94 32 21 02 26 7f 1c 9a 03 96 9d 22 1c`
A: `52 41 bc bc b1 44 d2 16 33 44 52 29 16 d9 1a 6b 73 71 0c`
Zistite obsah správy.
d) Ani nápad profesora Quilenia nebol dokonalý. Predpokladajme, že profesor Smith sa dozvedel o truhlici skôr, ako ju prvýkrát poslali. Chce sa zmocniť jej obsahu. Navrhnite, čo má spraviť, aby sa mu to podarilo.
e) Aj Cyrilov postup má svoje slabiny. Kľúč K by totiž mali použiť len raz. Prečo? Cyril s Dianou použili dvakrát ten istý kľúč K. Poštárka zachytila obe správy:
C_1: `17 6f c8 3c 06 6f 39 8c 37 37 c9 37 c0 ce d8 db 2f 85 2e 63 c6 e2 62 f9 43 4c 73 08 4a 4d 75 af 2c b0 c8 cc 80 4b 5f 05 57 25 73 84`
C_2: `17 65 d4 39 4a 77 22 88 33 37 c5 33 d0 da cf d9 29 97 67 71 83 a4 64 f7 17 4c 37 09 4f 47 21 b6 6d ad c4 d3 cf 44 47 1f 4f 3e 77 8b`
Zistite čo najpresnejšie ich obsahy. Môže sa vám hodiť nejaký zoznam slovenských slov.

2241. O aritmetickej postupnosti

„Aha, je tam: 2, 6, 10.“ „Ale tá nemá správnu diferenciu.“ „Tak dobre, tak tam nie je,“ ozýva sa hádka Paľa a Moniky z prednáškovej miestnosti. Aspoňže prednášajúci ešte neprišiel. Už by im dávno vynadal, nech si napíšu program, ktorý to spočíta, aby sa nemuseli hádať.

Úloha: Na vstupe sú čísla a a n. Nasleduje n **reálnych** čísel. Vašou úlohou je zistiť, či sa dá spomedzi nich vybrať niekoľko tak, aby (nie nutne v poradí, v akom sú na vstupe) tvorili aspoň trojčlennú aritmetickú postupnosť. Navyše musí platiť, že ak ste vybrali

k čísel, tak diferencia výslednej postupnosti musí byť $a/(k-1)$. Napríklad aritmetická postupnosť dĺžky 5 musí mať diferenciu $a/4$. (Pripomeňme, že postupnosť $x_0, x_1, \ldots, x_{n-1}$ je n-členná aritmetická postupnosť s diferenciou d ak $x_i = x_0 + i \cdot d$ pre každé i.)

Vstup:	Výstup:
`9 5`	`áno`
`11 5 9 14 8`	*(5, 8, 11, 14 je 4-členná postupnosť s diferenciou $a/3 = 3$)*

2242. O štatistikárení

KSP pohltila mánia DDR (Dance Dance Revolution). Pred i33 stoja študenti i cvičiaci zo všetkých kútov matfyzu (a keby len z neho) a čakajú, kedy sa budú môcť zahrať. V i33 však musí byť poriadok! A tak sa chodí pekne po jednom a vždy, keď niekto ide tancovať, musí sa pekne zapísať (meno, minúta, v ktorej tancoval) – aby sa vedelo...

Po niekoľkých týždňoch tancovania sa už zaplnilo niekoľko hárkov menami a minútami a KSPákov začali zaujímať rôzne štatistiky: Koľko ľudí tancovalo? Koľko minút v priemere tancovali? V ktorú hodinu je najviac rušno? ...

Viacero údajov sa im už podarilo spočítať, keď sa zasekli na otázke: Koľko striedaní počas tancovania vôbec prebehne? Problémom je, že niekedy sa prestriedalo v rámci jednej minúty aj niekoľko ľudí a teda majú rovnaké časy. Z nazbieraných údajov vieme povedať, kto počas danej minúty tancoval, nevieme však, v akom poradí.

Povedať presne, koľkokrát sa ľudia prestriedali, nemusí byť možné. Vedeli by ste zistiť, koľko *minimálne* muselo byť striedaní?

Úloha: Daný je počet DDR maniakov a následne pre každého človeka zoznam minút, počas ktorých tancoval, ukončený nulou (zoznam je usporiadaný vzostupne). Môžete predpokladať, že časy sú celé čísla od 1 do 20 000 a počet DDR maniakov zhruba do 50. Úlohou je zistiť, koľko *najmenej* prestriedaní sa určite muselo uskutočniť.

Príklad:

Vstup:	Výstup:

`2`
`1 2 0`
`2 0`
	`1`

(Označme si ľudí A a B. V prvej a na začiatku druhej bol A, potom B. Mohlo sa stať, že najskôr tancoval A, potom B a na konci druhej minúty opäť A, chceme však najmenší počet striedaní.)

Vstup:	Výstup:

`2`
`3 5 7 7 8 0`
`4 6 7 7 12 0`
	`5`

(Zjavne v 3. minúte tancoval A, vo 4. ho vymenil B, v 5. bol opäť A a v 6. B. Nie je však jasné, ako sa striedali počas 7. minúty. Najmenej striedaní by bolo v prípade, ak B tancoval po 6. minúte aj na začiatku 7. minúty, odtancoval si obe pesničky a potom pustil A. Ten tancoval až do konca 8. minúty, pustil B a šiel domov. Dokopy máme 5 striedaní.)

2243. O maximalizácii radov

Na našej škole majú mnoho oddelení, kde študenti môžu vybavovať rôzne byrokratické záležitosti. Jedno z takých oddelení je aj Oddelenie Dlhých Radov (ODR).

Je to jedno z najdôležitejších oddelení. Vlastne čokoľvek si chce študent vybaviť, musí prejsť týmto oddelením. Hlavná náplň práce tohto oddelenia je vytvorenie čo najdlhšieho radu študentov čakajúcich na vybavenie. To je veľmi ťažké, hlavne keď toto oddelenie má viac kancelárií a navyše veci, ktoré treba vybaviť, sú veľmi jednoduché. Väčšinou ide len o podpísanie nejakého papiera. Zamestnanci ODR sú ale veľmi šikovní a našli veľmi efektívny spôsob ako si svoju prácu plniť čo najlepšie. Zaviedli dômyselný systém prestávok, počas ktorých neúradujú. Napríklad obedná prestávka, desiatová prestávka, kávičková prestávka

I, kávičková prestávka II a mnohé iné. Samozrejme, zamestnanci si môžu podľa potreby nejakú prestávku pridať alebo ju posunúť. Udržať takýto systém v prevádzke ale vôbec nie je jednoduché. Každú prestávku treba stihnúť presne načas, inak by sa mohlo stať, že treba obslúžiť ďalšieho študenta a tým skrátiť rad, čo vôbec nie je dobré. Potrebovali by program, ktorý ich upozorní na začiatok každej prestávky. No a keďže oni s počítačmi nevedia robiť (ešte to tak, aby vedeli, to by sa zrýchlilo vybavovanie a skrátili rady) potrebujú vašu pomoc.

Úloha: Napíšte program, ktorý potrebujú. Váš program bude interaktívny a s okolím bude komunikovať takto: Bude prijímať správy:

- `TICK` – túto správu dostane váš program každú sekundu.
- `SET <d>` naplánuje začiatok prestávky o d sekúnd. Po prijatí tejto správy vráti nejaký identifikátor tejto prestávky.
- `DEL <id>` ... zruší naplánovanie prestávky s identifikátorom `id`.
  A bude vysielať správu
- `BZZZ <id>` na začiatku prestávky s identifikátorom `id`.

Môžete predpokladať, že naraz bude naplánovaných najviac n prestávok a budú naplánované najviac m sekúnd dopredu (kde n a m sú dané na vstupe). Váš program by po spustení mal bežať večne, teda ani po veľmi dlhom čase by v ňom nemalo nič pretiecť.

Príklad: Takto by mohla vyzerať interakcia s vašim programom pre $n = 17$ a $m = 47$. (Čítať treba najskôr ľavý a potom pravý stĺpec, nie striedavo!)

```
> TICK                    > SET 3
> TICK                    17idemejest
> SET 2                   > TICK
47ksp                     > SET 2
> TICK                    147teraz
> SET 4                   > TICK
74obed                    > TICK
> TICK                    BZZZ 17idemejest
BZZZ 47ksp                BZZZ 147teraz
> DEL 74obed
```

2244. O reforme ciest do práce

„To je škandál!“
„Už som zažila všeličo, ale toto tu ešte nebolo!“
„Ako keby nestačilo, že minulý rok zaviedli daň zo svine!“
„Hovoríš mi z duše, drahá.“
„A čo sa vlastne stalo?“
„Ále, vy ste to ešte nepočuli? Veď sú toho plné správy...“
„Nepočula, práve som sa vrátila z dovolenky.“
„Nebudem chodiť okolo horúcej kaše a poviem vám to rovno. Minister chodenia do práce vydal nehorázne nariadenie!“
„...no a ja neviem, či sú blbí alebo čo, ale ja proste nemám ako to spraviť, no neviem...“
„...neskáčte mi do reči, drahá! Takže, ževraj od prvého mája nesmie človek cestou z práce použiť žiadnu ulicu, ktorou šiel cestou do práce, vraj aby sa nenudil.“
„No a viete, zlatko, vďaka tým žgrlošom na ministerstve ciest do práce nepoznám jedinú dušu, ktorá by toto mohla spraviť, tak málo ulíc máme!“
„A vrchol všetkého je, že minister financii teraz vyhlásil, že sa dostavajú nové ulice, ale musí ich byť čo najmenej!“
„Presne tak. No povedzte, je normálny?“

Úloha: Máte zadané n – počet významných lokalít v meste. Momentálne sú tieto lokality poprepájané ulicami veľmi skromne, pre každé dve lokality existuje práve jeden

spôsob, ako sa z jednej dostať po uliciach do druhej. Na vstupe je zadaných $n-1$ dvojíc čísel lokalít, ktoré sú spojené ulicou. (Rozmyslite si, prečo ich je práve $n-1$.)

Úlohou ministerstva ciest do práce (a aj vašou) je pospájať niekoľko dvojíc lokalít novými cestami tak, aby už nik nemohol mať problémy s plnením nového nariadenia. Pre každú dvojicu lokalít teda budú musieť existovať dve cesty medzi nimi, ktoré nemajú spoločnú ani jednu ulicu.

Môžete predpokladať, že pracovníci ministerstva sú šikovní a zvládnu postaviť novú ulicu medzi ľubovoľnou dvojicou lokalít. (Použijú nadjazdy, mimoúrovňové križovatky a podobné fičúrie.)

Výstupom programu by mal byť najmenší počet ulíc, ktoré stačí postaviť. Nezabudnite na dôkaz, že váš program naozaj nájde najmenšie možné riešenie!

Vstup:	Výstup:
`6`	`2`
`1 3  3 4  4 5  2 3  4 6`	*(Napr. postavíme ulice 1-6 a 2-5.)*

2245. Ostávame pri kryptológii

V úlohe 2235 sme sa zaoberali šifrovaním. Okrem iného sme si ukázali šifru, ktorá síce sama o sebe bola absolútne bezpečná, ale spôsoby, ktorými sme ju použili, už také skvelé neboli. A práve o tom bude táto úloha. Kryptológia totiž nie je len o tom vymyslieť čo najsilnejšiu a najťažšie prelomiteľnú šifru. Treba tieto šifry vedieť aj správnym spôsobom použiť. Tieto spôsoby použitia budeme volať *kryptografické protokoly*. Taký protokol nie je vlastne nič iné, ako dohodnutý postup, kto má kedy komu poslať akú správu.

Zoznámme sa najskôr s tradičným kryptografickým osadenstvom. Na úvod sú tu naši starí známi, Alica a Bob, ktorí si chcú poslať správu. Okrem nich sú tu ale aj Eva (eavesdropper; odpočúva komunikáciu) a Oskar (opponent; nepriateľ, ktorý môže do komunikácie aj aktívne zasahovať). Občas sa ukáže aj Trudy (trusted authority; niekto, komu Alica aj Bob dôverujú). V kryptografii sa používajú aj ďalšie postavy, v našom príbehu si ale vystačíme s týmito. Občas budeme namiesto mena človeka písať len jeho prvé písmenko.

Príklad: Protokol z minulej série vyzeral takto:

1. $A \to B : M \oplus K_A$
2. $B \to A : M \oplus K_A \oplus K_B$
3. $A \to B : M \oplus K_B$

Už sme videli, že tento protokol nie je odolný ani voči Eve – tej stačí vypočuť si komunikáciu a vie z nej určiť prenášanú správu M. A taký Oskar, ten dokáže napáchať ešte väčšie škody.

Povedzme si teraz niečo o nástrojoch, ktoré nám kryptológia ponúka pri navrhovaní protokolov. V prvom rade je to šifrovanie. Správu s zašifrovanú kľúčom K budeme značiť $\{s\}_K$. Predpokladáme, že účastníci komunikácie, ktorí K nepoznajú, nevedia efektívne z $\{s\}_K$ určiť s ani K.

V druhom rade to sú časové pečiatky. Pre naše potreby časová pečiatka bude reťazec, v ktorom je napísaný aktuálny čas a dátum. Časové pečiatky budeme značiť T_*.

Občas sa hodí, aby niektorý účastník komunikácie poslal „čerstvo vygenerovaný“ náhodný reťazec N_* (odborne zvaný *nonce*). Druhý účastník mu ho bude väčšinou musieť poslať späť, aby potvrdil, že jeho správa je aktuálna. Načo to bude dobré, to ešte uvidíte. Ale dosť teórie, poďme sa pozrieť na príklad.

Príklad: Protokol *„Žaba so širokou hubou“*.
Pripomeňme si, že T (Trudy) je účastník komunikácie, ktorému Alica aj Bob dôverujú. Nech A a T majú dohodnutý kľúč K_A, nech B a T majú kľúč K_B.

1. $A \to T : A, \{B, K\}_{K_A}$
2. $T \to B : \{A, K\}_{K_B}$

Čo tento protokol hovorí? Keď chce Alica niekedy hovoriť súkromne s Bobom, vygeneruje si kľúč K, ktorý potom ona a Bob použijú. Keď chce teraz kľúč K pomocou Trudy oznámiť Bobovi, pošle Trudy správu: Ja, A, chcem hovoriť s B, pošli mu kľúč K. Keďže časť správy je zašifrovaná kľúčom K_A, Trudy má istotu, že správa naozaj prišla od A. Tak pošle B správu, kde oznámi: A chce s tebou komunikovať pomocou kľúča K. Keď B dostane túto správu, vie, že je od Trudy, lebo je zašifrovaná kľúčom K_B, ktorý nik iný nepozná.

Všetko vyzerá byť v poriadku. Čo sa ale stalo, keď Alica s Bobom skúsili použiť tento protokol? Oskar si uložil na disk všetky posielané správy. Niekedy, dlho po tom, ako komunikácia skončila, získal kľúč K. (Buď priamo z počítača Alice/Boba, alebo trebárs hrubou silou z nimi posielaných správ.) Teraz ale Oskar prikročí k svojmu diabolskému plánu. Tváriac sa ako Alica pošle Trudy nasledovnú správu:

$1'.\ O(A) \to T : A, \{B, K\}_{K_A}$

V zápise protokolu značenie $O(A)$ znamená, že je to Oskar, tváriaci sa, že je Alica. (Napríklad sa pripojil na jej telefónnu linku, prípadne poslal mail, ktorý sa tvári, že je z Alicinej adresy.)

Oskar síce nepozná kľúč K_A, ale správu $\{B, K\}_{K_A}$ má dávno uloženú na disku. Nič netušiaca Trudy si overí, že to šifrovala Alica a oznámi Bobovi, že Alica s ním chce hovoriť. No a nič netušiaci Bob sa začne s Alicou rozprávať, pričom použije kľúč K, ktorý mu Trudy poslala – ale ktorý už Oskar pozná! A ten jednoducho všetky Alici určené správy odchytí a tvári sa, že je ona.

Keď im toto Oskar vyviedol, videli Alica a Bob, že je zle. I upravili protokol tak, aby používal časové pečiatky. Alica si vygeneruje a v správe pošle časovú pečiatku T_A, aby ukázala, že správa je „čerstvá“. Trudy si pri prijatí správy overí, či je správa „dostatočne čerstvá“ a ak nie, ignoruje ju. Podobne sa overí aj čerstvosť správ medzi Trudy a Bobom. Upravený protokol vyzeral takto:

Príklad: Upravený protokol *„Žaba so širokou hubou“*.

1. $A \to T : A, \{T_A, B, K\}_{K_A}$
2. $T \to B : \{T_T, A, K\}_{K_B}$

Čo ale Oskar nevymyslel? Podobne ako v prvom prípade, aj tu odchytil obe správy. Teraz mu ale nestačilo uložiť si ich, lebo by mu časom prestali platiť. Namiesto toho poslal Trudy správu:

$1'.\ O(B) \to T : B, \{T_T, A, K\}_{K_B}$

Pre Trudy toto vyzerá ako správa od Boba, ktorý chce komunikovať s Alicou. Časová pečiatka je čerstvá. Trudy preto pošle Alici správu:

$2'.\ T \to O(A) : \{T'_T, B, K\}_{K_A}$

Samozrejme, Oskar si ju opäť odchytí. Čo získal? Novšiu časovú pečiatku! No a nič mu nebráni opakovať tento postup až do okamihu, kým sa nedozvie kľúč K... Aktuálnu správu, stále s platnou časovou pečiatkou, potom rovnako ako predtým použije na oklamanie Alice alebo Boba.

Úloha 1: Predpokladajme, že na komunikáciu je použité asymetrické šifrovanie. To pre nás znamená, že hocikto vie vyrobiť šifrovanú správu určenú človeku C, ale len C si vie takúto správu prečítať. (Príkladom takto použiteľnej šifry je šifra RSA.)

Použitý protokol funguje tak, že A aj B si vygenerujú náhodné reťazce N_A, resp. N_B, navzájom si ich oznámia, vypočítajú si z nich kľúč $K = f(N_A, N_B)$ a pomocou toho budú ďalej komunikovať. (Funkcia f je nejaká známa funkcia, na ktorej sa vopred dohodli, napríklad môže ísť o niektorú známu hešovaciu funkciu.)

1. $A \to B : \{N_A, A\}_{K_B}$
2. $B \to A : \{N_A, N_B\}_{K_A}$
3. $A \to B : \{N_B\}_{K_B}$

Vysvetlenie: Alica pošle svoj reťazec a svoje meno, zašifrované tak, aby si to vedel prečítať len Bob. Bob si správu dešifruje, vyrobí správu určenú pre Alicu, v tej jej oznámi

N_A a svoje N_B. Alica si overí, že N_A je to, čo poslala (a teda vie, že správa od Boba je čerstvá). Ak áno, oznámi mu N_B. V tomto okamihu aj Alica, aj Bob poznajú N_A aj N_B a spočítajú si kľúč K. Bob po prijatí N_B môže začať komunikáciu použitím kľúča K.

Alica chce práve komunikovať s Oskarom a poslala mu prvú správu:

1. $A \to O : \{N_A, A\}_{K_O}$

Ukážte, ako Oskar oklame Boba – Bob si bude myslieť, že si dohodol kľúč s Alicou, v skutočnosti ho ale bude vedieť Oskar.

Hint: Paralelne s tým, ako komunikuje s Alicou, začne Oskar aj nový rozhovor s Bobom, použitím tohto istého protokolu a podobných správ...

Úloha 2: Vráťme sa späť do situácie, kedy máme aj Trudy, ktorej Alica aj Bob veria a majú s ňou dohodnuté šifrovacie kľúče ako v príkladoch. Tentokrát si Alica s Bobom vymysleli nasledujúci protokol:

1. $A \to B : A, N_A$
2. $B \to T : B, N_B, \{A, N_A\}_{K_B}$
3. $T \to A : N_B, \{B, K, N_A\}_{K_A}, \{A, K, N_B\}_{K_B}$
4. $A \to B : \{A, K, N_B\}_{K_B}, \{N_B\}_K$

Vysvetlenie: Alica pošle Bobovi správu, že s ním chce komunikovať a náhodný reťazec N_A. Ak Bob súhlasí, pošle Trudy potrebné informácie; časť z nich zašifruje, aby Trudy vedela, že sú od neho. Trudy si overí, že informácia pochádza od Boba – a keďže A bolo zašifrované, vie, že Bob chce naozaj hovoriť s Alicou. Trudy im vygeneruje kľúč K. Alici pošle Bobovo náhodné slovo N_B, správu pre ňu a správu pre Boba. Alica si odšifruje správu určenú pre ňu, overí si N_A (teda vie, že jej prvá správa prešla cez Boba) a dozvie sa kľúč K. Pošle Bobovi správu určenú pre neho. Navyše mu oznámi, že ona úspešne získala kľúč K – tak, že mu pošle $\{N_B\}_K$. Ak aj Bobovi všetko sedí, vie, že si práve s Alicou dohodol kľúč K.

Aspoň tak si to Alica s Bobom predstavovali. Oskar však opäť niečo vyhútal. Dokáže Alicu presvedčiť, že si dohodla kľúč s Bobom, hoci to nebude pravda.

Hint 1: Oskar sa nedozvie K.

Hint 2: Začiatok útoku bude vyzerať takto:

1. $A \to O(B) : A, N_A$
1′. $O(B) \to A : B, N_A$
2′. $A \to O(T) : A, N'_A, \{B, N_A\}_{K_A}$
...

z2211. Zruční artisti

Artisti v cirkuse Kaledon začali nacvičovať novú zostavu. Ich zostava je však veľmi obtiažna – či už na koordináciu pohybov, alebo fyzickú silu artistov. Hlavne druhá zložka cviku je pre nich náročná, keďže koordináciu už zvládli v predošlých cvikoch. Chcú postaviť vežu tak, že sa postupne naskladajú na seba. Ich problém spočíva v tom, že im zostava nevychádza, lebo nie sú schopní udržať jeden druhého. Pritom stačí, ak každý artista unesie toho priamo nad sebou, na váhe ostatných artistov nad ním nezáleží. To preto, že najťažšia časť je zachytiť kamaráta v okamihu, keď vám dvojitým saltom vyskočí na plecia. Keď už na nich stojí, je všetko v pohode. Pomôžte im tento problém vyriešiť.

Úloha: Na vstupe je zadané prirodzené číslo n. Ďalej nasleduje n čísel, ktoré vyjadrujú hmotnosť každého artistu. To je zároveň aj najväčšia hmotnosť artistu, ktorý mu môže stáť na pleciach.

Vašou úlohou je vypísať hmotnosti artistov v takom poradí, v akom majú vytvoriť vežu, ktorá nespadne.

Príklad:

Vstup:	Výstup:
5	70 68 62 56 55
62 70 55 68 56	

z2212. Zábava na kolieskach...

„Inline hýbe svetom.“ To si jedného krásneho letného dňa povedal aj Janči a rozhodol sa, že namiesto skvelého nového harddisku investuje radšej do zdravia a kúpi si korčule. I zavolal kamaráta Ferka a vybrali sa korčule kupovať. Asi týždeň veľmi intenzívne pobehovali po rôznych obchodoch so športovými potrebami a pozerali na korčule: tvrdosť, materiál, typ, tvar a polomer koliesok, typy ložísk, veľkosť topánky. Poznáte to – s obchodmi je to ťažké... tak veľa tovaru, tak málo času a ešte menej peňazí. Nakoniec sa podarilo. Kúpili krásne, veľké, čierne korčule.

Netrvalo dlho a, napriek svojmu pokročilejšiemu veku, sa naučil korčuľovať pomerne rýchlo. Objavil sa však problém: Pri športe (a iných zábavných činnostiach) človek rýchlo stráca pojem o čase, a navyše veľmi rýchlo vyhladne. Začína sa stmievať a Janči s hrôzou v očiach zisťuje nepríjemnú skutočnosť: obed už asi nestihne. No predsa len je tu ešte nejaká šanca stihnúť aspoň večeru... ale to by sa musel poriadne poponáhľať.

Janči – ako každý správny nezodpovedný živel – nemá bledomodrý šajn, kadiaľ sa rýchlo dostane domov. A preto ste tu vy, ktorí ochotne zachránite úbohý ľudský život pred smrťou hladom.

Úloha: Na vstupe máme zadané dve celé čísla r a s a potom mapu, ktorá má r riadkov po s stĺpcoch. Miesto, kde sa Janči práve nachádza, je označené znakom „J“, miesto, kde býva, je označené „D“. Cestičky, po ktorých sa môže pohybovať, sú označené znakmi „.“ a naopak, miesta, kadiaľ sa korčuľovať nemôže, znakmi „#“. Mimo mapy sa taktiež pohybovať nemôže. Janči je po celom dni značne unavený a vlečie sa domov konštantnou rýchlosťou. Vašou úlohou je nájsť čo najkratšiu cestu, ako sa dostať domov. Janči sa vie pohybovať všetkými 8 smermi.

Príklad:

Vstup:

```
12 31
#################################
#.......#######..##..##..##...#
#.#####.#######..##..##..##...#
#.####..########.............##
#.#####.########.############D##
#..###........##..##.##.####.##
##.####.##.##............###..#
##.############.########.#####.#
##............########..####.#
#.J.##..##..##..########.#####.#
#...##..##..##..########.......#
#################################
```

Výstup:

```
#################################
#.......#######..##..##..##...#
#.#####.#######..##..##..##...#
#.####..########.----------\.##
#.#####.########/###########D##
#..###........##|.##.##.####.##
##.####.##.##../.........###..#
##.############/########.#####.#
##.----------/.########..####.#
#.J.##..##..##..########.#####.#
#...##..##..##..########.......#
#################################
```

(Cestu nemusíte vykresľovať tak pekne ako my v príklade, stačí ľubovoľne vyznačiť políčka, kadiaľ vedie.)

z2213. Zaľúbený čaj

„Mňam, ten môj čaj je taký dobrý!“ rozplýva sa Majka. „Máš pravdu Majka, aspoň o sto lepší ako môj,“ súhlasí natešený Mirko.

Mirko a Majka si vždy radi pochutnajú na dobrom čaji. Nedávno Mirko zistil, že čaj je oveľa lepší, ak doň pridá špeciálnu čajovú hubu. V zelenom čaji sa taká huba dokonca aj rozmnožuje: každý deň o šiestej poobede sa premení každá huba na nové tri. Tento proces sa opakuje, až kým niekto čaj nevypije. Samozrejme, čím viac húb, tým viac chute. Ako správny milovník čajov používa Mirko vlastnú stupnicu chute čaju. Po niekoľkých ochutnávkach už vie, že každá huba zvýši stupeň chuti čaju o jedna. A navyše, ako správny milovník čajov, používa rovnakú stupnicu aj pre svoju lásku k druhým.

Mirko má Majku rád, aj vie presne stupeň svojej lásky. Mirko je ale hanblivý, preto sa rozhodol, že ju zatiaľ len pozve na svoj nový čaj. Ale predsa len by jej nejako chcel naznačiť svoje pocity. Zaumienil si teda, že Majkin čaj spraví lepší od vlastného. Aby

mohla vytušiť, ako ju má rád, bude rozdiel stupňov chutí ich čajov rovný stupňu Mirkovej lásky k Majke.

Mirko teda musí začať pripravovať dva čaje, na ktorých si s Majkou o niekoľko dní o piatej pochutnajú. Húb má ale obmedzenú zásobu, preto každý deň môže použiť najviac jednu. Nevie ale, do ktorého čaju ju má kedy pridať (ak vôbec). A stačí mu tých zopár dní, aby spravil dosť chutný čaj?

Úloha: Vašou úlohou je napísať plán pre Mirka. Na vstupe sú dve celé čísla, ℓ – stupeň Mirkovej lásky k Majke a m – počet dní ostávajúcich do ich stretnutia. Plán bude m celých čísel, pričom každé z nich je -1, 0 alebo 1. Číslo na i-tom mieste Mirkovi povie, do ktorého čaju pridá hubu v i-ty deň. Nula znamená že do žiadneho, -1 do svojho a 1 do Majkinho čaju. Huba vložená v i-ty deň sa v čaji potom $(m-i)$-krát rozdelí na tri, t.j. výslednému čaju pridá $3^{(m-i)}$ chute. Rozdiel medzi výslednými chuťami Mirkovho a Majkinho čaju musí byť práve ℓ. V prípade, že nie je možné čaje do m dní dostatočne ochutiť, treba to Mirkovi čo najmiernejšie oznámiť.

Príklad:

Vstup:	Výstup:
`7 58`	`0 0 1 -1 0 1 1`
Vstup:	Výstup:
`4 81`	**`Prepáč Mirko, ale čaje už nestihneš pripraviť.`**

(V prvom príklade z huby, ktorú vhodí do Majkinho čaju na tretí deň, bude na konci $3^4 = 81$ chute. Z huby zo šiesteho dňa budú 3, zo siedmeho 1 chuť. V Mirkovom čaji bude z huby zo štvrtého dňa $3^3 = 27$ chute. Nakoniec teda bude rozdiel medzi chuťami čajov $(81 + 3 + 1) - (27) = 58$.)

z2214. Zzzzzz alebo imatrikulácia

Imatrikulácia je (aspoň na vysokej škole) úžasná slávnosť. Pekne napochodujete podľa abecedy, sľúbite, že budete vzornými študentmi, prevezmete imatrikulačný list, pokloníte sa rektorovi (podaktorí to spackajú a poklonia sa študentom) a odpochodujete. Keď je tých študentov trošku viac (a na vysokej ich *je* trošku viac), je to prudká zábava. Sedíte na svojom mieste (tí prezieravejší si priniesli nejakú literatúru) a čakáte, kým príde rad na vás. Tí, čo túto procedúru úspešne absolvujú, môžu až do samého konca – ktorý je zdanlivo v nedohľadne – spať. Na druhej strane, tí, čo ešte len majú ísť, sa musia sústrediť, aby nebolo dáke fiasko. Po chvíli ľudia s menami ku koncu abecedy začnú presviedčať ostatných, aby sa navrhlo nejaké spravodlivejšie poradie. Tu však vzniká problém: podľa niektorých by bolo najspravodlivejšie začať od konca – veď „vždy“ sa začína od začiatku; na druhej strane, ľudkovia s priezviskami na začiatku abecedy horlivo protestujú: „veď práve, vždy sa chodí podľa abecedy, nás ako prvých vyvolávajú k tabuli, vtedy vám to vyhovuje, čo?“; ďalšia skupinka chce zase začať od stredu… A tak začala hara-vara.

Čas pokročil a viacerí zistili, že takto to asi nepôjde, a že najspravodlivejšie (resp. jediné) poradie, na ktorom sa budú schopní dohodnúť, bude asi náhodné. Nuž ale kde by nabrali toľko slamiek? A kým by sa podľa slamiek zoradili… Našťastie sú tu KSP-áci, ktorí im s radosťou napíšu program, ktorý nejaké náhodné poradie vygeneruje, však?

Úloha: Na vstupe je zadané kladné celé číslo n. Úlohou je vypísať čísla od 1 do n v náhodnom poradí. Každé poradie má mať rovnakú pravdepodobnosť. To znamená, že keby sme váš program spustili $100 \cdot n!$-krát, každé poradie by sa malo na výstupe objaviť približne stokrát.

Príklad:

Vstup:	Výstup:
`3`	`1 2 3` *alebo* `1 3 2` *alebo* `2 1 3` *alebo* `2 3 1` *alebo* `3 1 2` *alebo* `3 2 1` *(s rovnakou pravdepodobnosťou)*

z2215. Zase škola

Škola vám už dávno začala a nám pomaly ale isto začal nový semester. Taký začiatok semestra so sebou prináša veľa úloh ako napríklad zápis, vymyslenie novej série KSP-čka, ale hlavne vytvoriť si rozvrh. Matfyzáčka Monika si pozapisovala všetky predmety zo svojho ročníka a ešte polovicu nasledujúceho. Pozrela sa na svoj rozvrh a zhrozila sa. Nielenže musí každé ráno skoro vstávať, ale hlavne musí byť na dvoch a niekedy až na troch prednáškach naraz. To jej však až tak nevadí. Vyberie si jednoducho ten zaujímavejší predmet, na ktorý bude chodiť a na ostatné má do skúšky veľa času.

Teraz má však väčší problém. Chcela by sa pochváliť ostatným, koľko toho za semester stihne. Preto potrebuje svoj rozvrh nejakým spôsobom nakresliť a ukázať ostatným. Tu však potrebuje vašu pomoc. Pomôžte Monike nakresliť si svoj rozvrh prehľadne, aby sa v ňom vyznala a pritom efektne, aby mohla zapôsobiť na ostatných.

Úloha: Vašou úlohou je napísať program, ktorý na vstupe dostane popis Monikinho rozvrhu a vypíše ho v nejakom peknom a prehľadnom tvare. Na vstupe dostanete zoznam jednotlivých predmetov. O každom predmete dostanete päť údajov: jeho názov, typ, deň, čas a miestnosť konania. Názov je reťazec dlhý maximálne 40 znakov. Typ predmetu je napríklad prednáška, cvičenie, kurz, laboratórne cvičenie a má maximálne 20 znakov. Všetky predmety začínajú a končia v 50-minútových intervaloch od 8:10 do 19:50. Miestnosť je reťazec dlhý maximálne 5 znakov. Časy predmetov sa *môžu ľubovolne prekrývať*. Formát vstupu si môžete zvoliť ľubovolný, aký vám vyhovuje.

Rozvrh kreslite do textového režimu, aby si ho mohli prečítať aj ľudia pracujúci na starých termináloch, ktoré nemajú grafický režim. Rozvrh by nemal byť príliš dlhý ani široký, aby sa zmestil na obrazovku resp. aby sa dal vytlačiť na jeden list papiera. Hodnotiť sa bude hlavne prehľadnosť rozvrhu a jeho vzhľad.

Okrem samotného programu nám pošlite aj niekoľko príkladov rozvrhov, ktoré váš program vygeneroval.

Príklad:

Vstup:

```
Utorok, 8:10-10:40, Úvod do distribuovaných algoritmov, Prednáška, C
Piatok, 8:10-10:40, Teória paralelných výpočtov, Prednáška, B
```

Výstup:

```
       Pondelok     Utorok       Streda       Štvrtok      Piatok
================================================================================
 8 10 >            |p          C|            |            |p          B|
                   |Úvod do     |            |            |Teóri.paral.|
------------------|distr.algor.|------------|------------|výpočtov    |
 9 00 >            |            |            |            |            |
                   |            |            |            |            |
------------------|            |------------|------------|            |
 9 50 >            |            |            |            |            |
                   |            |            |            |            |
--------------------------------------------------------------------------------
```

z2221. Zruční artisti II

Kaledonskí cirkusanti potrebuju znovú Vašu pomoc! Aj keď ste im pomohli minule, nestačilo to. Vymysleli novú zostavu. Tentoraz chcú postaviť „pyramídu“. Ich pyramída má vyzerať tak, že vždy dvaja artisti držia na pleciach tretieho. Výsledok bude vyzerať tak, že na spodku pyramídy je k artistov. Na ich pleciach je $k/2$ artistov, pričom vždy dvaja spodní držia práve jedného z týchto. Na „treťom poschodí“ je $k/4$ artistov. A tak to pokračuje ďalej, až navrchu je jediný artista. Aj tu platí, že každý artista musí byť ľahší ako každý z dvoch kolegov, ktorí ho držia.

Úloha: Na vstupe je zadané prirodzené číslo n, pričom platí, že $n + 1$ je mocninou dvoch. Ďalej nasleduje n čísel, ktoré vyjadrujú hmotnosť každého artistu. To je zároveň aj najväčšia hmotnosť artistu, ktorý mu môže stáť na pleciach. (Stačí ak artista unesie toho priamo nad sebou, na hmotnosti ostatných artistov nad ním nezáleží.)

Vašou úlohou je vypísať hmotnosti artistov v takom poradí, v akom majú vytvoriť „pyramídu“, ktorá nespadne. Výslednú pyramídu vypíšte postupne po úrovniach, začínajúc najvyššou. Stačí nájsť ľubovoľné jedno prípustné riešenie.

Príklad:

Vstup:

```
7
62 70 55 68 66 74 47
```

Výstup:

```
47 55 68 66 62 70 74
```

(Pyramída z výstupu vyzerá nasledovne: Artista s váhou 47 stojí na vrchu, a to na pleciach artistov s váhami 55 a 68. Artista s váhou 55 stojí na pleciach artistov s váhami 66 a 62. Artista s váhou 68 stojí na pleciach posledných dvoch artistov.)

z2222. Zábava na kolieskach II.

Hoc ste minule Jančimu pomohli viac než dosť (a tým ste si ho veľmi zaviazali), objavil sa teraz iný problém. Nie preto, že by sa Janči korčuľovať nevedel, ale práve preto, že sa korčuľovať vie, a hlavne – aj sa korčuľuje! Teraz v zime! Ako blázon lieta na korčuliach po sade. Krížom-krážom, cez kaluže vody špinavej, prebíja sa hŕbou lístia cez alej.

Ale späť k problému. Tým teraz nie je ani tak Jančiho chorá myseľ či zlomené srdce ako priveľké množstvo rôznych nečistôt po ceste. Špina nepôsobí na korčule príliš blahodarne. Nezmyselnosť jeho konania mu určite nevysvetlíte, no môžete mu aspoň poradiť nejakú vhodnú cestu – najlepšie takú, kde prejde cez najmenší objem nečistôt.

Úloha: Na vstupe dostanete mapu rozmerov $r \times s$ políčok. Význam symbolov na mape je nasledovný:

– „`J`“ je aktuálna pozícia Jančiho (takýto znak je práve jeden)
– „`D`“ je jeho domov (aj takýto znak je práve jeden)
– „`#`“ je nepriechodné políčko
– „`.`“, „`*`“, „`+`“, „`@`“, „`%`“ sú políčka obsahujúce 1, 2, 3, 4, 5 nečistôt

Vašou úlohou je vypísať cestu s najmenším celkovým množstvom nečistôt.

Príklad:

Vstup:

```
15 30
################################
#++.......@@%%##%%@@.......++#
#+*#.......@@%##%@@.......#*+#
#*###########@%##%@**********#*
#..#********#*@%##%@*#*******#..#
#..#########*#@%%%%@*#********..#
#..#########*#@%%%%@*#********..#
#..**#######*********#******##++..#
#..**####################D+*..#
#.J**#######*********#******##++..#
#..#########*#@%%%%@*#********..#
#..#********#*@%##%@*#*******#..#
#*###########@%##%@**********#*
#+*#.......@@%##%@@.......#++#
#++.......@@%%##%%@@......+++#
################################
```

Výstup:

```
################################
#++-----\.@@%%##%%@@.-----\++#
#+/#.....\.@@%##%@@./.....#\+#
#/########\*@%##%@*/********|#
#|.#*******#|@%##%@/#*******#.|#
#|.########*#\@%%%%@|********./#
#.\**#####*#*-----/*#######++/.#
#.|**####################D--..#
#.J**#######*********#******##++..#
#..#########*#@%%%%@*#********..#
#..#********#*@%##%@*#*******#..#
#*###########@%##%@**********#*
#+*#.......@@%##%@@.......#++#
#++.......@@%%##%%@@......+++#
################################
```

(Cestu nemusíte vykresľovať tak pekne ako my v príklade, stačí ľubovoľne vyznačiť políčka, kadiaľ vedie.)

z2223. Zakódované dáta

Došiel raz jedného krásneho slnečného popoludnia Janík zo školy domov a pri pohľade na krásne zelené lesy navôkol, ktorých sa zatiaľ žiadna veterná smršť ani len nedotkla, sa

rozhodol, že musí vyjsť von do prírody, preč od počítača. Len keby nemusel úlohy do školy písať, hlavne ten štvorstranový referát. I zavolal on kamarátovi Miškovi a natešený, že mu ten poslať svoj referát emailom prisľúbil, utekal von.

Ked sa večer Janík vrátil, zobral sa, že trošku referát od Miška pozmení, aby príliš okaté nebolo, že ho odpísal. Ale beda, súbor od Miška vôbec nevyzeral ako normálny text a Janík preľaknutý, že si nebodaj Miško z neho srandu robí, alebo mu referát zašifrovaný poslal, rýchlo zdvihol telefón. Miško mu však len vysvetlil, že on len kvoli veľkosti a prenosovej rýchlosti súbor svojím vlastným programom skomprimoval, a že Janík si s dekomprimáciou určite ľahko poradí.

I zaprisahal sa Janík, že odteraz si bude pekne poctivo úlohy a referáty sám písať. Len na tento jeden už ani za mak času nemá, a tak by rád vašu pomoc privítal.

Úloha: Napíšte program ktorý dekomprimuje vstupný text. Ten je zložený len z veľkých a malých písmen abecedy, znakov „`[`“, „`]`“ a číslic, pričom sekvencia „`číslo[text]`“ znamená `číslo` opakovaní textu „`text`“. Tieto sekvencie môžu byť aj vnorené. Výstup programu má byť dekomprimovaný text zložený len z malých a veľkých písmen.

Príklad:

Vstup:	Výstup:
`5[A3[BC]D]`	`ABCBCBCDABCBCBCDABCBCBCDABCBCBCDABCBCBCD`

z2224. Zimný spánok

Slony sa koncom decembra odoberajú na zaslúžený zimný spánok. Slon Boris nie je v tomto ohľade žiadnou výnimkou. A takisto, ako každý poriadny slon, aj Boris má zo všetkého (jedla) najradšej čerešne. Vyberie sa teda do jaskyne (v zime predsa čerešne na stromoch nerastú), aby si nazbieral trochu svojej obľúbenej dobroty. Boris je už ospalý a vie, že sa mu neoplatí vracať sa z jaskyne. Radšej v nej pekne prespí až do jari. Predtým by ale chcel nazbierať čo najviac čerešní, z ktorých sa predsa tááák sladko spíí...

Úloha:

Jaskyňa má jeden vchod a niekoľko miestností. Tie sú pospájané chodbami, po ktorých Boris môže ísť len v jednom smere. V každej miestnosti je niekoľko čerešní. Vašou úlohou je nájsť Borisovi cestu, na ktorej môže nazbierať najviac čerešní.

Na vstupe bude počet miestností n. Nasledovať bude n riadkov – popisov jednotlivých miestností. V i-tom z týchto riadkov budú na začiatku dve čísla – počet čerešní c_i a počet vychádzajúcich chodieb m_i. Za nimi bude nasledovať m_i čísel miestností, do ktorých je možné sa z miestnosti i dostať priamou chodbou. Boris sa môže po týchto chodbách hýbať len v smere von z miestnosti i. Vchod je miestnosť číslo 1 a nie sú v ňom žiadne čerešne.

Navyše môžete prepokladať, že Boris nemôže v jaskyni chodiť do nekonečna. Teda pre každé dve miestnosti a, b platí, že ak sa Boris vie nejako (aj cez iné miestnosti) dostať z miestnosti a do miestnosti b, určite sa nevie dostať z miestnosti b do miestnosti a.

Na výstupe uveďte čísla miestností, cez ktoré keď Boris postupne pôjde, nazbiera najviac čerešní. Ak je takýchto ciest viac, vypíšte ľubovolnú z nich.

Príklad:

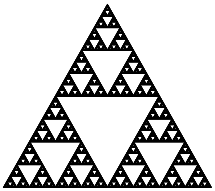
Vstup:	Výstup:

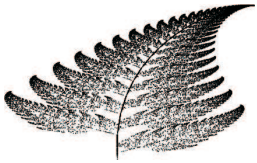
`9`
`0 3  4 2 9`
`5 2  3 7`
`2 0`
`4 2  5 6`
`3 1  7`
`2 1  7`
`1 1  8`
`5 0`
`6 0`
	`1 4 5 7 8`
	
	(Nazbiera tak 13 čerešní.)

z2225. Za blchou

Už ste počuli o blche Polovičke? Raz sa chela dostať od jednej steny izby k druhej. Darmo jej Mišo vravel, že ak preskočí vždy iba polovicu zostávajúcej vzdialenosti, nikdy sa k druhej stene izby nedostane. Blcha mu oponovala a tvrdila, že to bez problémov zvládne. Po pár skokoch však s dlhým nosom doskákala. O chvíľu tu však bola znovu (prezývali ju tiež Raketa) a tvrdila, že keď bude skákať tak, že sa vždy otočí k nejakému rohu izby a skočí do polovice cesty k nemu, síce sa nedostane do žiadneho rohu, ani k stene, ale dostane sa na ľubovoľné miesto v izbe. A tak ju Mišo zobral do miestnosti v tvare trojuholníka a vyzval ju: „Keď si taká úžasná skokanka, tak skoč sem," a postavil sa do stredu. A Raketa ostala opäť so spadnutou sánkou.

Zaujímavé je, kam **všade** dokáže Raketa doskákať a kam nie. Mišo si to začal kresliť a bol celkom prekvapkaný z toho, čo mu vyšlo. Chvíľu sa s tým hral a zistil, že sa tak dajú celkom dobre generovať fraktály. Poďme sa teraz pozrieť na to, ako to s nimi je. Fraktály majú tak peknú, oku lahodiacu vlastnosť, že keď ich zväčšíme (prípadne trochu natočíme a posunieme), dostaneme opäť pôvodný fraktál. Príkladom je Sierpińského trojuholník (obr. vľavo).





Všimnite si, že takýto trojuholník sa dá rozložiť na tri trojuholníky polovičnej veľkosti. Obrázok by sa teda mohol kresliť tak, že nakreslíme trojuholník, rozdelíme ho na tri polovičné. Na tieto tri opäť použijeme tento postup: rozdelíme ich na tri polovičné (teda už štvrtinové oproti pôvodnému) atď. Matematicky sa toto zmenšovanie, otáčanie a posúvanie robí cez transformácie: keď máme nejaký bod $[x, y]$, vypočítame nové

$$x' = ax + by + c \qquad y' = dx + ey + f$$

pre nejaké konštanty a, b, c, d, e, f. Dajme tomu, že chceme zobraziť celý Sierpińského trojuholník na ľavú dolnú časť. Všetkým bodom sa potom zmenšia súradnice na polovicu; ostatné dva trojuholníky treba po zmenšení ešte trochu posunúť; tak môžeme dostať takéto transformácie:

(T_1)	$x' = x/2$	$y' = y/2$	pre ľavý dolný trojuholník
(T_2)	$x' = x/2 + 100$	$y' = y/2$	pre pravý dolný trojuholník
(T_3)	$x' = x/2 + 50$	$y' = y/2 + 86.6$	pre horný trojuholník

Keď si vytvoríme takéto transformácie, môžeme vykonať takýto postup: začneme bodom $[x_0, y_0]$, ktorý patrí do fraktálu; náhodne si zvolíme jednu transformáciu a pomocou nej vypočítame ďalší bod. Takto postupne vypočítame body $[x_1, y_1]$ z $[x_0, y_0]$, $[x_2, y_2]$ z bodu $[x_1, y_1]$ a tak ďalej. Týmto postupom môžeme ľahko vykresliť nejaký fraktál.

Vyskúšajte napríklad zmeniť transformáciu ľavého dolného trojuholníka z vyššie uvedeného príkladu na

(T_1') $\quad x' = 0.39x - 0.07y \qquad y' = 0.07x + 0.39y \qquad$ pre ľavý dolný trojuholník

Dostanete trochu pokrivený Sierpińského trojuholník.

Na rozbehnutie si ukážeme ešte jeden fraktál: Barnsleyovu papraď (obr. vpravo).

Tú môžeme rozdeliť takto: pravá vetva, ľavá vetva a zvyšok paprade. Aby bola papraď spojená, treba ešte stonku. Ako nájdeme príslušné konštanty? Bez znalostí matíc asi najskôr tak, že si nakreslíme čo máme a čo chceme dostať a potom koeficienty budú riešením nejakej sústavy rovníc. Máme šesť koeficientov, teda potrebujeme aspoň šesť rovníc, teda aspoň tri body. Pošrotíme a dostaneme niečo takéto:

(T_1)	$x' = 0.85x + 0.04y$	$y' = -0.04x + 0.85y + 80$	vrch
(T_2)	$x' = -0.15x + 0.28y$	$y' = 0.26x + 0.24y + 22$	pravá vetva
(T_3)	$x' = 0.2x - 0.26y$	$y' = 0.23x + 0.22y + 80$	ľavá vetva
(T_4)	$x' = 0$	$y' = 0.16y$	stonka

Úloha: Napíšte program, ktorý bude používať vyššie popísanú metódu náhodného iterovania bodu a bude kresliť fraktály. Pohrajte sa trochu s konštantami a spolu s popisom programu a zdrojovým kódom nám pošlite niekoľko obrázkov fraktálov (s popísanými transformáciami), ktoré ste pomocou tohoto programu dokázali nakresliť. Hodnotiť sa budú najmä nájdené fraktály, čím krajšie, tým lepšie.

z2231. Zažite hrôzu v knižnici!

Jožko Marhuľa zhrozene vpadol do miestnej pobočky mestskej knižnice v Trebuliakove. Iba dnes sa dozvedel, že do zajtra má prečítať „perlu slovenskej literatúry“, ktorú mu zadala jeho učiteľka slovenčiny. V knižnici sa ocitol po prvý raz a z jedovatej vône plesnivejúcich kníh sa mu okamžite zakrútila hlava. Bolo mu jasné, že zlovestne vyzerajúca knihovníčka (mala iba 537 vlasov a 1 kolozub) mu nepomôže, ba naopak. Beštiálna bytosť za pultom sa usmiala, akonáhle Jožko vyjachtal svoju opovážlivú prosbu, a jej kolozub sa pritom zaleskol v slabom svetle jedinej stropnej lampy. Jožko nasucho preglgol, keď vyziabnutý hnát s nechtom pripomínajúcim pazúr ukázal na temne sa týčiace regály kníh. Zdalo sa mu, že sa tiahnu do nekonečna.

Knihovníčka si udržiavala celú knižnicu usporiadanú – avšak nie podľa mien autorov alebo podľa názvu knihy, ale podla nejakých divných čísel. Pokiaľ chcel niekto nejakú knihu, mohol si ju sám nájsť alebo tejto bytosti povedať číslo knihy. Jožko sa poobzeral a usúdil, že by bolo veľmi nebezpečné vybrať sa medzi regále a hľadať svoje knihy, ktoré sa nachádzajú nevedno ako ďaleko. Mohlo by sa mu stať, že zablúdi a bude večne blúdiť regálmi alebo dokonca by ho mohla nejaká kniha aj zjesť! Preto sa odobral ku kartotéke a podľho hľadať, aby čo najskôr našiel číslo svojej knihy a stihol ju do zajtra prečítať. Kartotéka (t.j. abecedný zoznam kníh) je však obrovská a tiež nie je len tak ľahké hľadať v nej. Rovno mohol zabudnúť na to, že prejde všetky záznamy. To by tu mohol zostať až do zajtrajšieho dňa a to už bude neskoro! Preto mu pomôžte rýchlo nájsť číslo jeho knihy.

Úloha: Predpokladajme, že v pamäti už máte načítané pole veľkosti n (môže byt veľmi veľké), ktoré predstavuje kartotéku s n knihami. Pre každú knihu je v poli záznam obsahujúci jej meno (text pozostávajúci s veľkých a malých písmen abecedy, čísel a medzier) a jej číslo (celé a jedinečné). Toto pole je usporiadané podla názvu knihy.

(Pri testovaní svojich programov musíte samozrejme obsah tohto poľa odniekiaľ načítať. Nás však zaujíma len tá časť programu, ktorá bude v už načítanom poli vyhľadávať.)

Na vstupe dostanete názov hľadanej knihy. Vašou úlohou je navrhnúť algoritmus, ktorý čo najrýchlejšie nájde knižničné číslo danej knihy. Váš algoritmus by mal fungovať všeobecne pre ľubovoľný názov knihy. Pokiaľ záznam pre knihu nájdete, vypíšte jej číslo, inak vypíšte, že sa kniha v knižnici nenachádza.

Príklad:

Kartotéka:

```
Arabela 112, Popoluska 158, Pysna princezna 155, Sipova Ruzenka 150,
Snehova kralovna 33, Snehulienka a sedem trpaslikov 999, Zlatovlaska 120
```

Vstup:

```
Snehova kralovna
```

Výstup:

```
33
```

z2232. Zmenený zákon

„No to snáď nie je pravda! Oni ten zákon zase zmenili!“ povzdychol si Pedro. O čomže to hovorí? Nuž, o najnovšom vylepšení zákona o daniach na Kapingamarangi. Doteraz bolo

vyplňovanie daňového priznania veľmi jednoduché – stačilo len vymenovať v chronologickom poradí všetky príjmy, ktoré človek v daný rok mal. Pedro si presne takéto záznamy viedol počas celého roka a už sa tešil, že konečne raz bude mať to priznanie rýchlo vybavené a bude sa môcť zaoberať zaujímavejšími vecami. No a zrazu takáto pohroma!

Nový zákon nariaďuje, že príjmy musia byť usporiadané najprv podľa druhu činnosti, z ktorej pochádzajú, a to v abecednom poradí! Tak napríklad, ak trinásteho mája zarobil tisícku za prenájom automobilu a pätnásteho januára prenajal vŕtačku, tak príjem z automobilu má byť uvedený skôr, ako príjem z vŕtačky; napriek tomu, že ten druhý bol uskutočnený skôr. Keďže chudák Pedro má toho na robote ešte strašne veľa, požiadal o pomoc vás – skúsených programátorov. Pomôžte mu?

Úloha: Na vstupe je číslo n, ktoré označuje celkový počet príjmov, ktoré Pedro za uplynulý rok mal. Okrem toho máte zadaných n dvojíc „činnosť príjem“. Toto je Pedrov zoznam, o ktorom viete, že je v chronologickom poradí. Vašou úlohou je usporiadať tieto záznamy do abecedného poradia tak, že v rámci každého názvu činnosti musí zostať zachované chronologické poradie.

Príklad:

Vstup:

```
5
Vrtacky 40
Automobily 2000
Motorky 1000
Automobily 1000
Automobily 1500
```

Výstup:

```
Automobily 2000
Automobily 1000
Automobily 1500
Motorky 1000
Vrtacky 40
```

z2233. Zelený stôl

„Tentoraz nemáš šancu!“ prehodil vypasený kšeftár Mrľa a so záujmom pozeral na mladého, urasteného mládenca, čo len včera prišiel do mesta. Mladému sa leje pot z čela, poškuľuje, všelijako sa mrví, ale vie, že nakoniec aj tak bude musieť udrieť. Už to nechce odkladať. Zavrie oči a BUM!

Keď ich znova otvorí, pozrie najprv na zdeseného Mrľu, vypliešťajúceho oči na stôl. Pozrie tam i on. Podarilo sa! Všetky štyri gule ostali v rohoch stola. (Biliardový stôl nemá diery, to s dierami sú pool a snooker.) Vykročí ku kšeftárovi. „A teraz peniaze,“ povie. Mrľa sa zdráha, vyhovára, ale veľmi si nepomôže, mladík by ho raz-dva premohol. Vtom ho osvieti. Povie: „Urobíme poslednú stávku. Ak vyhráš, dostaneš dvakrát toľko. Ak prehráš, nechám ťa vyvaľať v bahne a oberiem ťa do poslednej vindry. Ber, alebo nechaj tak!“ Mladík je prostý, je to chlapec z dediny, neuvedomí si, že už vyhral dosť a stávku prijme.

„Tak teda počúvaj,“ Mrľa si odkašle a poodhalí všetkých dvanásť žltých zubov, „Guľu dáme do stredu stola. Chcem, aby sa toľko a toľkokrát odrazila od dlhšej strany, toľko a toľko od kratšej a nakoniec sa zastavila v strede stola.“ „Koľko?“ pýta sa neveriacky mladík. „Toľko a toľko,“ zopakuje tučniak s diabolským úškrnom na tvári.

Hoci náš mládenec je veľký hráč biliardu, rozmýšľanie mu vždy skôr nešlo, ako išlo. Preto by potreboval vašu pomoc, aby vedel kam má guľu poslať.

Úloha: Na vstupe sú zadané a, b, kde a je dĺžka strany stola rovnobežnej s barpultom, b je dĺžka tej druhej strany. Ďalej sú zadané čísla n a m, pričom n je požadovaný počet odrazov od strany dĺžky a a m od strany dĺžky b.

Nájdite polpriamku, po ktorej treba poslať guľu stojacu v strede stola, aby spravila predpísaný počet odrazov a vrátila sa do stredu stola. Môžete sa spoľahnúť, že mladík pošle guľu správnou rýchlosťou.

Presnejšie, biliardový stôl si predstavíme ako súradnicovú sústavu so stredom stola v jej počiatku. Vypíšte uhol, ktorý zviera hľadaná polpriamka s x-ovou osou (t.j. osou rovnobežnou s barpultom).

Príklad:

Vstup:	Výstup:
`100 100 1 1`	`45.000000000000`
Vstup:	Výstup:
`234 148 33 22`	`43.492564241225`
Vstup:	Výstup:
`700 500 470 29`	`85.062875860935`

z2234. Zaujímavá permutácia

„Zjem index, ak toto nie je tá Zaujímavá permutácia!“ nechal sa včera počuť Onot. Ykborh sa už chystá do školy. Ak Onot nenájde trojčlennú aritmetickú postupnosť, mal by mu aspoň kečup doniesť, aby sa mu lepšie prehĺtalo. Onot je šikovný a snaživý, ale čo ak nemá šancu?

Úloha: Je dané číslo n a tiež permutácia čísel $1, \ldots, n$. (Permutácia je postupnosť n čísel, v ktorej sa každé z čísel 1 až n vyskytuje práve raz.) Poraďte Onotovi, či sa dá z tejto permutácie $n-3$ čísel vyškrtnúť tak, aby zvyšné tri (v poradí, v akom boli v pôvodnej permutácií) tvorili aritmetickú postupnosť.

Ešte raz to isté matematicky: Nech p_1 až p_n sú postupne čísla danej permutácie. Zistite, či existujú také čísla $i < j < k$, že postupnosť (p_i, p_j, p_k) je aritmetická. (Postupnosť je aritmetická, ak existuje číslo d také, že každý ďalší člen dostanem z predchádzajúceho pripočítaním d. V našom prípade teda musí platiť $p_i + d = p_j$ a $p_j + d = p_k$.)

Príklad:

Vstup:	Výstup:
`6`	`áno`
`4 5 3 6 1 2`	*(Indexom $i = 2$, $j = 3$ a $k = 5$ zodpovedá postupnosť $(5, 3, 1)$.)*

z2235. Zvítanie s zelenými mužíkmi

Aj toto sú životné situácie, na ktoré sa musíme prichystať. A ideálne je prichystať sa poriadne. Nikto nevieme, ako vyzerajú, kde sa schovávajú. Možno sú už medzi nami. V každom prípade, buďme k nim milí, pekne a milo ich privítajme. A čo už je milšie ako... kvetinky. Krásne, zelené lístky so žltým koláčom. Hotové ICQ.

Máme hriadku so štvorcovými políčkami, do každého políčka môžeme zasadiť najviac jeden kvietok. Chceme vedieť týmito kvietkami kresiť čiary. Keď k nám budú prilietať, nech už z dialky vidia, že ich máme radi. Ale samozrejme, aj my máme len obmedzené prostriedky. Nikto nevieme, či vôbec sú, a či prídu. Takže to budú v podstate zbytočne vyhodené peniaze. A peňazí na vyhadzov nie je nazvyš. A navyše, budú to sadiť záhradníci, nie matematici.

Úloha: Dostaneme rozmery hriadky: r riadkov, s stĺpcov. Následne dostaneme niekoľko riadkov obsahujúcich súradnice koncových bodov čiary: $[r_1, s_1]$ a $[r_2, s_2]$. Vyrobte bitmapu s nakreslenými čiarami (stačí pole znakov ako v príklade).

Snažte sa, aby váš program kreslil čiary pokiaľ možno presne. Tiež by bolo dobré, ak by bol čo najjednoduchší. Skúste, či viete úlohu vyriešiť bez použitia sínusov a podobných funkcií. A čo tak iba použitím celých čísel?

Pošlite nám aj príklad rozumne veľkého vstupu a výstupu vášho programu.

Príklad:

Vstup:	Výstup:
`7 20`	`....................`
`2  2    5  5`	`.*.......*.........`
`2 10    6  8`	`..*.....*..........`
`6 11    4 19`	`...*....*.......***.`
	`....*..*.....***....`
	`.......*..***.......`
	`....................`

z2241. Zvrátená učiteľka

Pozor, tento príbeh je veľmi smutný a obsahuje zmienky o násilí páchanom na deťoch. Preto odporúčam, aby ste pri čítaní sedeli. Bola raz jedna učiteľka v škôlke, ktorá mala veľmi rada matematiku. Keďže 5-ročné deti ešte nevedia počítať, tak sa rozhodla, že ich to naučí, aby sa mala s kým rozprávať. Najprv ich musela naučiť čísla a za tým účelom vymyslela nasledujúcu nudnú hru. Učiteľka postavila n stoličiek do radu, a na každú si sadlo jedno vystrašené dieťa. Potom ešte na stoličky napísala náhodne čísla od 1 po n (na každú jedno a žiadne dve stoličky nemali rovnaké čísla).

Deťom rozkázala, že keď tleskne, majú sa postaviť a sadnúť si na toľkú stoličku zľava, aké je číslo na tej stoličke, kde sedeli. Deťom sa takáto hra vôbec nepáčila a na začiatku im to ani nešlo, ale zato učiteľka sa veľmi dobre zabávala. Tlieskala, plieskala a zákerne sa pri tom usmievala... Ako sa tak hrali, učiteľke sa zrodil v hlave skvelý nápad: „A teraz retrospektívne! Ako keď sa pretáča video dozadu, to bude sranda, keď tlesknem, tak sa vrátite o krok späť," ale deti na to sklamane: „Ale my si nepamätáme, kde sme kedy sedeli, veď to už hráme 3 dni v kuse, mohli by ste nás už aj domov pustiť, aj pospať by sme si mohli...". „Ja vám dám, vy lamy, že pospať!" skríkla učiteľka a dodala: „Veď pamätať si nemusíte nič, keby ste vedeli algebru, tak viete, že to, kam si máte sadnúť, keď tlesknem, sa dá zistiť z očíslovania stoličiek. Lamy, he-he". Ale deti sú už totálne zmagorené, takže im budete musieť pomôcť vy, lebo ináč ich začne učiť aj algebru, a to si ani ja nechcem predstaviť, aké hrozné hry im potom vymyslí...

Úloha: Na vstupe je číslo n, a postupnosť celých čísel $a_1, a_2, \ldots, a_n$, kde a_i je číslo na i-tej stoličke. Napíšte program, ktorý vypíše postupnosť čísel $b_1, b_2, \ldots, b_n$, kde b_i bude číslo stoličky, kam si ma sadnúť dieťa sediace na i-tej stoličke po tom, ako učiteľka zatlieska. (Hrá sa druhá verzia hry popísanej v zadaní, t.j. tá retrospektívna.)

Bonusová otázka: vymyslite pre takúto hru nejaký pekný názov, najlepšie odpovede budu ohodnotené.

Príklad:

Vstup:	Výstup:
`5` `2 3 4 5 1`	`5 1 2 3 4`

(Pri pôvodnej verzii hry by deti bežali nasledovne: Z prvej stoličky na druhú, z druhej na tretiu, z tretej na štvrtú, zo štvrtej na piatu a z piatej na prvú. Pri novej verzii hry teda pobežia späť – napríklad dieťa zo štvrtej stoličky pobeží späť na tretiu.)

Vstup:	Výstup:
`5` `3 1 4 2 5`	`2 4 1 3 5`

z2242. Z kalendára

Najobľúbenejšie dievča v 1.A je Karolína. Aj vy by ste si ju obľúbili. Nikdy by totiž nezabudla na vaše narodeniny. (A keď si niekto spomenie na vaše narodeniny, je reálna šanca, že vám donesie aspoň čokoládu.) Pamätala by si aj to, koľko máte rokov a kto sa narodil v rovnaký deň, mesiac alebo rok ako vy. Karolínkine spolužiačky jej nesmierne závidia a šuškajú si, že v tom bude nejaký fígeľ. Ako to môže všetko vedieť a pritom si ani nepamätá, kedy bola bitka pri Kreščaku?! (No schválne, kedy bola?)

A fígeľ v tom naozaj je. Karolína má doma databázu dátumov narodení všetkých svojich spolužiakov, kamarátov, a vôbec. Každé ráno, skôr než odíde do školy, zadá počítaču dátum a počítač jej vypľuje, kto sa vtedy narodil. Takto zistí všetko, čo potrebuje. (V tom, že má ráno čas na takéto hlúposti, bude tiež nejaký fígeľ, ale to je o inom.) Ak sa chcú spolužiačky Karolíne vyrovnať, potrebujú si napísať program, ktorý bude robiť to isté.

Úloha: Na vstupe budú zadané dátumy narodenia ľudí v tvare: „meno deň mesiac rok". Vašou úlohou je napísať program, ktorý načíta tento zoznam a potom bude fungovať v nekonečnom cykle, kedy striedavo načíta od užívateľa otázku a odpovie na ňu.

Otázky budú zadávané v tvare „deň mesiac rok“, pričom ľubovoľný z parametrov môže byť nahradený hviezdičkou (s významom „všetky“). Odpoveďou musí byť zoznam všetkých ľudí z databázy, ktorých dátumy narodenia vyhovujú otázke.

Teda ak chceme napríklad všetkých ľudí narodených v januári v hocijaký deň a rok, zadáme otázku „`* 1 *`“.

Príklad:

Vstup:	Výstup:

`jano 2 3 1985`
`marek 24 12 1990`
`matus 29 7 1985`
`lukas 12 3 1986`	

`* * 1985`	`jano matus`
`2 3 *`	`jano`

z2243. Zachránená škôlka

Malý Quido svojimi chlapčenskými vrtochmi spôsobil, že sa v jeho škôlke už žiadne dve dievčatá nekamarátia. Rád by teraz dal všetko do poriadku, aby sa znovu kamarátila celá škôlka, a teda aj každé dve dievčatá (chlapci rozhádaní nie sú). Vie, že keď obetuje niekoľko cukríkov, budú sa niektoré dve dievčatá znovu kamarátiť. A keďže platí, ako to už v škôlke medzi dievčatami býva, že ak si moja kamarátka, tak aj všetky tvoje kamarátky sú mojimi kamarátkami, nebude to mať Quido až také ťažké. Tak si aspoň povedal, že na uzmierovanie minie čo najmenej cukríkov – aby mu ich čo najviac zostalo.

Úloha: Na vstupe budete mať dve čísla – počet dievčat n a počet dvojíc dievčat m, ktoré by Quido ešte mohol cukríkmi udobriť. Dievčatá si očíslujeme od 1 do n, Nasledovať bude m trojíc a_i, b_i, c_i, ktoré budú znamenať, že dievčatá a_i a b_i sa budú znovu kamarátiť, keď sa s nimi dvomi Quido chvíľu zahrá a zjedia pritom spolu c_i cukríkov. Žiadne dve dievčatá sa na začiatku nekamarátia. Vašou úlohou je vypísať niekoľko dvojíc x_i a y_i, aby platilo, že ak Quido udobrí tieto dvojice, tak sa už budú kamarátiť všetky dievčatá. (Inými slovami, bude teda platiť, že každé dve dievčatá budú spriatelené priamo alebo cez niekoľko ďalších kamarátok.) Navyše musíte nájsť ten spôsob, ktorý bude stáť Quida najmenší možný počet cukríkov. Môžete predpokladať, že existuje aspoň jeden spôsob, ako spriateliť všetky dievčatá.

Príklad:

Vstup:	Výstup:

`7 9`
`1 2 47`
`2 3 6`
`3 5 1`
`2 4 3`
`4 5 2`
`5 6 2`
`3 6 1`
`4 7 1`
`6 7 3`
	`1 2`
	`2 4`
	`3 5`
	`4 5`
	`3 6`
	`4 7`
	
	(Takéto skamarátenie bude stáť len 55 cukríkov.)

z2244. Zobudené veveričky

Ako všetci isto viete z hodín biológie alebo prírodopisu, veveričky sa neukladajú na zimný spánok. Tento rok sa zima nejako natiahla a veveričky zistili, že nemajú čo jesť. Našťastie všetky bývajú na strome, kde rastú veľké a chutné orechy. Síce sú ešte z minulého roka, ale veveričky vedia, že hlad je najlepší kuchár.

Každá veverička býva buď v mieste, kde sa konáre stromu stretávajú, alebo kde sa nejaký konár končí. Navyše v každom takomto mieste býva práve jedna veverička a medzi každými dvoma domčekmi existuje práve jedna cesta po strome. Po dlhej zime zostalo na strome presne toľko orechov, koľko tam býva veveričiek. A keďže veveričky vedia, kde majú stavať domčeky, orechy rastú hneď vedľa ich domčekov (priam zanedbateľne ďaleko). Lenže niektoré veveričky majú pri dome veľa orechov a niektoré ani jeden.

Rozhodli sa, že sa spravodlivo rozdelia, teda každá dostane jeden orech, ale naviac by chceli, aby sa dokopy nachodili čo najmenej. Každá veverička musí ísť totiž po orech, ktorý jej bol určený, a potom ísť naspäť do svojho domčeka.

Úloha: Najprv je na vstupe zadané číslo n, ktoré označuje počet orechov, veveričiek a aj domčekov. Ďalej nasleduje n čísel, ktoré označujú počet orechov pri domčekoch 1 až n. Súčet týchto čisel je n. Potom je na vstupe zadaných $n-1$ dvojíc x_i, y_i, ktoré určujú obidva koncové domčeky i-teho konára. Vašou úlohou je nájsť najkratšiu vzdialenosť (t.j. najmenší počet konárov), akú musia dokopy prejsť veverička.

Príklad:

Vstup:	Výstup:
`9`	`14`
`2 1 0 1 3 0 0 2 0`	
`1 2  1 3`	*(Jedno optimálne riešenie:*
`3 5  3 6`	*Veverička #3 a #6 pôjdu k domčeku*
`1 4  4 9`	*#5, veverička #9 pôjde k domčeku*
`4 8  4 7`	*#8 a veverička #7 k domčeku #1.)*

z2245. Zamyslite sa

Ako obvykle, aj túto sériu je posledná úloha netradičná. Zamýšľali ste sa niekedy nad tým, čo stojí za úlohami, ktoré my vytvoríme a vy riešite? Máte teraz príležitosť vyskúšať si časť povinností vedúceho KSP.

Úloha: Vašou úlohou je vymyslieť úlohu, ktorá by svojou obtiažnosťou zodpovedala KSP-Z. Teda nemala by byť príliš ľahká, ale zase by ste mali byť schopní ju vyriešiť (a nie len vy, ale aj ostatní riešitelia KSP-Z). Okrem vymyslenia úlohy požadujeme aj priloženie vzorového riešenia (aj programu), spolu s ohodnotením jednotlivých druhov riešení. Napríklad vzorové riešenie má časovú zložitosť $O(n \log n)$ a pamäťovú $O(n)$. Za funkčné riešenie by ste udelili 15 bodov. Za riešenie s pamäťou $O(n^2)$ 12 bodov atď. Samozrejme by sa mala dať daná úloha s takou zložitosťou aj vyriešiť (čiže nejedná sa len o hypotetické riešenia). Inak povedané, požadujeme návod, ako daný príklad opravovať.

Okrem toho, úloha, ktorú pošlete, by mala byť dostatočne neznáma. Ak napríklad pošlete úlohu, ktorá bola v KSP-Z pred rokom, tak očakávajte minimum bodov – až 0. Rovnako úloha, ktorá sa dá ľahko „vygúgliť“, dostane málo bodov. Naopak kreatívne riešenia (resp. zadania) dostanú bodov veľa.

Príklad: Ostatné úlohy v tejto zbierke, keby boli neznáme `;-)`.

2311. O Kingu

Poznáte hru Kingo? Nie? Nuž, vyzerá nasledovne: Každý hráč si na svojom hracom pláne vyznačí k čísel. Potom sa vyžrebuje niekoľko (aspoň k) čísel. Za výherné čísla sa prehlási k najväčších spomedzi vyžrebovaných čísel. Vyhráva ten, kto z nich uhádol najviac. Hrá sa vždy v klube a čísla sa hovoria iba pri žrebovaní, potom ich už nikdy nezopakujú.

Aj Tomáško si raz kúpil tiket s hracím plánom, lebo sa mu prisnilo, že určite niečo vyhrá. V deň žrebovania však ochorel a musel zostať doma. Tak som sa rozhodla, že sa prejdem do klubu Kingo a budem sledovať čísla, ktoré žrebujú.

Čo sa však nestalo, zabudla som si doma tašku, v ktorej bol aj tiket, aj všetky moje perá a papiere. A žrebovanie už začína! Čísla si ale nemám kam zapísať, ešte k tomu som už stará, takže moja pamäť už tiež nie je to, čo bývala. Viem si zapamätať iba $47k$ čísel.

Úloha: Na vstupe je číslo n – počet čísel, ktoré sa budú žrebovať, a číslo k. Nasleduje n čísel, ktoré môžete načítať iba raz. Vašou úlohou je vypísať k najväčších spomedzi načítaných čísel. Pamäť použitá vašim programom musí byť lineárna od k. Samozrejme, n môže byť oveľa, oveľa väčšie ako k.

Ak sa vám úloha zdá ľahká, poriadne porozmýšľajte, či je vaše riešenie naozaj najefektívnejšie možné.

Príklad:

Vstup:	Výstup:

`18 6`
`2 51 6 33 65 89 54 32 52`
`63 10 12 50 94 3 14 93 71`
	`89 71 94 93 65 63`

Poznámka: Ako nepovinnú domácu úlohu si skúste rozmyslieť, ktorých k čísel by ste pri tejto hre na svojom hracom pláne zaškrtli vy a prečo. (Predpokladajte, že viete, koľko čísel a z akej množiny sa žrebuje.)

2312. O šifrochtivých KSPákoch

Ako už iste všetci viete, KSPáci sú takí zvláštni ľudia, ktorí radi chodia po všelijakých hrách, kde sa lúštia šifry, celú noc sa behá po lese a robí sa kopu iných (pre obyčajných ľudí nepochopiteľných) aktivít. V rámci šetrenia cestovných nákladov sa rozhodli, že sa prepravia autom. Ale pri ceste autom treba vyhľadať cestu a napokon si aj ten benzín zaplatiť.

Keďže KSPáci sú zdatní programátori, nebol pre nich problém napísať program, ktorý im našiel tú úplne najsamlepšiu cestu. Čo ale čert nechcel, na niečo predsa len zabudli: Kde budú tankovať? A keďže už nemajú veľa času nazvyš, rozhodli sa obrátiť sa na vás. Našťastie stihli zistiť zoznam benzínok na trase, čiže to nebudete mať až také ťažké. Stačí zistiť, kde tankovať tak, aby minuli čo najmenej peňazí.

Úloha: Na vstupe sú celé kladné čísla n, d, c a k. Číslo n označuje počet benzínok na trase, d označuje vzdialenosť cieľového miesta a c je cena benzínu na benzínke pri matfyze, kde KSPáci svoj výlet začínajú. Potom nasleduje práve n dvojíc celých čísel, ktoré popisujú jednotlivé čerpacie stanice. Prvé číslo označuje vzdialenosť čerpacej stanice od začiatku cesty a druhé cenu jedného pohára benzínu. Ďalej je na vstupe číslo k, ktoré označuje kapacitu benzínovej nádrže v pohárikoch. Auto žerie 1 pohárik benzínu na 1 kilometer cesty. Na začiatku je nádrž prázdna. Vašou úlohou je zistiť, koľko bude stáť najlacnejšia cesta. V cieli už môže byť nádrž znova prázdna.

Príklad:

Vstup:	Výstup:

`7 474 10 100`
`90 5`
`95 10`
`190 20`
`194 2`
`247 10`
`347 47`
`400 74`
	`8868`
	
	(Na začiatku kúpime 90 pohárikov benzínu, v prvej stanici 100 pohárikov, a v ďalšej 4 poháriky. V nasledujúcej stanici nič netankujeme. Ale v tej ďalšej kúpime hneď 100 pohárikov. V nasledujúcej kúpime 53 pohárikov, v ďalšej plnú nádrž a v poslednej už len 27 pohárikov benzínu.)

2313. Obrovská šachovnica

V ďalekom Šachovom kráľovstve majú veľmi radi šach. Dokonca na niektorých miestach sú obrovské šachovnice, na ktorých obyvatelia kráľovstva (jazdci, strelci, pešiaci, atď.) každý víkend usporadúvajú turnaje medzi mestami.

Tento víkend však všetci po príchode na šachovnicu zistili, že sú tam nebezpečné príšery. Skoro všetci zutekali kade ľahšie, zostali iba štyria jazdci. Každý jazdec vie, že

sám by boj s príšerou prehral, preto sa chcú všetci štyria stretnúť. A to na takom mieste, aby dokopy prešli čo najmenej. Problémom sú však príšery. Ak totiž nejaký jazdec stúpi na políčko s príšerou, tá sa zobudí a zareve tak hrozne, že v tom okamihu sa zobudia aj všetky ostatné príšery. Jazdec, ktorý zobudil príšery, ešte stihne utiecť (kým príšera reve), no neskôr už nikto nemôže stúpiť na políčko s príšerou. Jazdec sa pohybuje ako v šachu, teda pri každom skoku sa jedna jeho súradnica zmení o 2 a druhá o 1.

Úloha: Na vstupe sú najprv čísla r, s – počet riadkov a stĺpcov šachovnice. Nasleduje osem čísel, ktoré udávajú súradnice štyroch jazdcov. Nakoniec je na vstupe číslo n a za ním n súradníc príšer. Vašou úlohou je vypísať najmenší počet ťahov potrebný na to, aby sa jazdci stretli. Ak sa nemôžu stretnúť, vypíšte o tom správu.

Príklad:

Vstup:
```
5 5
1 1   1 5   5 1   4 4
1
2 3
```

Výstup:
```
6
```

(Stretnúť sa môžu napríklad na políčku $[1,1]$*. Prvý jazdec sa nemusí vôbec pohnúť. Druhý pôjde* $[1,5] \to [2,3] \to [1,1]$*, tretí* $[5,1] \to [3,2] \to [1,1]$*, štvrtý* $[4,4] \to [3,2] \to [1,1]$*. Príšeru zobudil druhý jazdec.)*

2314. O poriadnych a výnimočných kockách

Na tie ryby sa pamätám, ako by to bolo len včera. Vlastne nie, nepamätám. No nič, poviem vám o kockách. A nie o hocijakých, ale šesťstenných. A to ešte nie je všetko, pretože tie kocky majú osem vrcholov! Také kocky sa už dnes nevidia. Dnes výrobcovia kociek pridajú vrchol tam, hranu sem a to už potom vôbec nie je poriadna kocka.

No ale vráťme sa späť k tým poriadnym kockám. Okrem toho, že tieto kocky boli poriadne, boli výnimočné ešte aj tým, že mali na každej stene napísané jedno písmeno. No a tieto poriadne a výnimočné kocky patrili malému usoplenému princovi z kráľovstva Trojhrbej Mláky. Tento princ mal sklony ku krutosti a tyranizoval ostatné deti tým, že im dával hrôzostrašné úlohy. Najstrašidelnejšie na tých úlohách nebolo to, že by boli ťažké, ale skôr to, že ak ste ich do večera nevyriešili, tak vám princ dal odseknúť nohu, prípadne inú končatinu, ktorá bola zrovna po ruke.

No a práve dnes ráno dal princ jednú takú úlohu mne. Povedal mi dlhočizné meno svojho pradeda, dal mi svoje poriadne a výnimočné kocky a chce odo mňa vedieť, či sa to meno dá z kociek poskladať. A večer sa už blíži...

Úloha: V prvom riadku je slovo – reťazec písmen malej anglickej abecedy; jeho dĺžku označíme ℓ. V druhom riadku je číslo n – počet kociek. Nasleduje n riadkov, každý z nich obsahuje 6 písmen malej anglickej abecedy – písmená na kocke.

Vašou úlohou je povedať, či sa dá z kociek poskladať zadané slovo. (Poskladať slovo znamená vybrať niekoľko kociek, vhodne ich pootáčať a uložiť do radu tak, aby sme si želané slovo prečítali na ich predných stenách.) Ak sa to dá, vypíšte text „`Da sa`“, v opačnom prípade vypíšte text „`Neda sa`“.

Príklad:

Vstup:
```
bar
4
abcdef
defthd
cdefgc
ryvwzz
```

Výstup:
```
Neda sa
```

(Každé písmeno máme, ale b a a *sú na tej istej kocke.)*

Vstup:	Výstup:

`vega`
`4`
`abcdef`
`defthd`
`cdefgc`
`ryvwzz`
	`Da sa`
	
	(Kocky uložíme v poradí 4-2-3-1 a vhodne otočíme.)

2315. Optimálna sada mincí

Pri platení v obchode existujú rôzne spôsoby ako zaplatiť danú sumu. Napríklad ak máme mince s hodnotami 1, 2, 5, 10, 20 a 50, vieme zaplatiť sumu 118 nasledovne:

– dáme $50 + 50 + 10 + 5 + 1 + 1 + 1$

– alebo dáme $50 + 20 + 20 + 20 + 10$ a predavač nám vydá $1 + 1$

– alebo dáme $50 + 50 + 20$ a predavač nám vydá 2

– atď.

Posledný z uvedených spôsobov je najlepší, lebo je najjednoduchší – najmenší počet mincí pri ňom zmení vlastníka. Použitím tohto kritéria budeme vedieť jednoducho porovnať rôzne sady mincí, a to nasledovne:

Majme sadu mincí. Zoberieme si všetky sumy od 1 do nejakej hodnoty n. Pre každú z nich spočítame najmenší počet mincí, ktoré musia pri jej platení zmeniť vlastníka. (Predpokladajte, že obaja platiaci majú vždy dostatočnú zásobu mincí každého typu.)

Optimálnosť sady mincí bude priemer týchto počtov, čím je menší, tým lepšie. Ak sa niektorá suma nedá zaplatiť, sada mincí je nekonečne zlá.

Úloha: Vašou úlohou bude pre dané n a daný počet rôznych hodnôt mincí k navrhnúť **čo najlepšiu** sadu mincí. Hodnoty mincí musia byť celé čísla od 1 do n. Nemusíte nutne nájsť optimálne riešenie, avšak samozrejme platí: čím lepšie riešenie nájdete, tým viac bodov dostanete.

Pošlite nám najlepšie riešenia, ktoré dokážete nájsť, pre nasledovné dvojice (n,k): $(100, 2)$, $(50, 4)$, $(1\,000, 12)$ a $(10\,000, 15)$.

Príklad:

Vstup:	Výstup:
`100 6`	`1 24 34 39 46 50`

(Toto nie je optimálna sada mincí, ale je pomerne dobrá. Schválne, skúste nájsť sumu, na ktorú potrebujete viac ako 3 mince!)

2321. O nemocnici

Bolo raz jedno mesto. V tom meste bola nemocnica. V tej nemocnici boli pacienti a bolo ich dosť veľa. No prišiel Jurko Štatista a začal robiť štatistiku pacientov. Keď prišiel pacient do nemocnice, zapísal si jeho meno, rok narodenia, hmotnosť, výšku, veľkosť palca, lakťa a nechcite vedieť, čoho ešte všetkého. No nebol by to náš Jurko, keby nechcel všetky štatistické veci rátať a vyvodzovať z toho dôsledky.

I rozhodol sa on rátať medián výšok pacientov. Rozmýšľa a rozmýšľa, ale predsa len je to štatista a nie programátor.[3]

Úloha: Na vstupe je číslo n a za ním n výšok pacientov v poradí, ako prichádzajú do nemocnice. Vašou úlohou je pre každého prichádzajúceho pacienta povedať, aký je medián výšok pacientov, ktorý do nemocnice prišli dovtedy (vrátane neho). Medián m prvkov je taký prvok, ktorý by bol po usporiadaní vzostupne na $\lceil m/2 \rceil$-tom mieste.

[3] A tobôž nie je štatistik. Ak neviete, aký je v tom rozdiel, nalistujte si v slovníku. Zjavne to nevedel ani autor textu k tejto úlohe `:-)`.

Príklad:

Vstup:	Výstup:
`9` `158 182 99 92 52 201 193 167 177`	`158 158 158 99 99 99 158 158 167`

2322. O problémoch s prehadzovaním

Prišli sme zo sústredka, zložili sme veci a spravili sme tým veľký neporiadok. Bolo by ho treba upratať. Také upratovanie ale vôbec nie je jednoduché. Spočíva v tom, že sa veci poprehadzujú z miesta, na ktorom sú, na miesto, na ktoré patria. No a na to treba určitý manipulačný priestor, musí sa človeku chcieť, musí vedieť, kam čo patrí, a ešte kopu ďalších vecí.

Rozhodli sme sa teda, že upratovanie prenecháme stroju. Kúpili sme teda namakané zariadenie – Kombinačný Samočinný Poprehadzovávač (KSP). Pre tých, čo ešte KSP nevideli, stručne popíšem. Zariadenie predpokladá, že máme n miest, na ktorých sú veci.

Poprehadzovávaču zadáme program, ktorý pre každé miesto hovorí, kam (na ktoré miesto) chceme presunúť vec, ktorá na ňom je. Nesmieme pritom chcieť dať dve veci na to isté miesto (to by sa nezmestilo). Potom už stačí KSP spustiť a on to všetko poprehadzuje.

Už by sme tu aj mali upratané, keby sa nám KSP nepokazil. Ono totiž cestou sem nám asi trikrát spadol na zem a asi kvôli tomu po spustení sa program vykoná niekoľkokrát (presnejšie k-krát). Teda aby sme úspešne upratali, potrebujeme KSP zadať taký program, ktorý keď sa zopakuje k-krát, tak nám prehádže veci tak, ako chceme. Ale nech sa snažíme ako chceme, nedarí sa nám a chceli by sme vás poprosiť o pomoc.

Úloha: Napíšte program, ktorý dostane v prvom riadku vstupu čísla n a k a v ďalšom riadku dostane n čísel $a_1, \ldots, a_n$. Číslo a_i určuje číslo miesta, kam chceme vec na i-tom mieste prehodiť. Vypíše program pre Poprehadzovávač, ktorý keď sa k-krát zopakuje, prehádže veci tak, ako chceme. Ak sa to náhodou nedá, program by mal o tom podať vhodnú správu.

Príklad:

Vstup:	Výstup:
`3 2` `2 3 1`	`program: 1 -> 3, 2 -> 1, 3 -> 2`

(Vec, ktorá leží na prvom mieste, pri prvom prehadzovaní poputuje na miesto 3 a odtiaľ v druhom kole na miesto 2, kde aj mala skončiť. Podobne pre ostatné dve veci.)

Vstup:	Výstup:
`4 3` `1 3 4 2`	`nedá sa`

2323. O Rômeovi a Juliäi

Bolo raz jedno mestečko. A v tom mestečku bol raz jeden Rômeo. A div sa svete bola aj Juliä. No a čo že by to už bol za príbeh, keby neboli spolu v jednom malom útulnom domčeku? A čo už by také mohli robiť, keď nie počítať integrály spojitých funkcií? A ako si tak pekne hrkútajú o existencii druhej derivácie, zrazu sa Rômeo udrie do čela a skríkne: „ABCDEFGH, veď sa už stmieva a matka ma čaká s večerou!“ Juliä sa naňho pozrie pohľadom plným pobavenia a hnevu a nežným hláskom na Rômea zvrieskne: „Ty papľuh, matka ti je prednejšia ako hyperbolické funkcie?“ „Ale...“ namietne Rômeo. „Neprerušuj ma! Máš šťastie, že mňa matka tiež čaká s večerou, inak by bol medzi nami koniec.“

Po chvili búrlivej výmeny argumentov a návratových hodnôt sa dohodli, že pôjdu domov tak, aby išli čo najdlhší kus cesty spolu. Ale pretože sa tvária, že sa ponáhľajú, tak pôjdu tak, aby sa obaja stále približovali k svojmu domovu a teda aby sa najkratšia vzdialenosť od obidvoch ich domov stále znižovala. Nakoľko Rômeo a Juliä nemajú skúsenosti s hľadaním optimálnych ciest, tak si znova začali zúfať, lebo nevedia, ktorou uličkou sa vydať. Buďte tak milí a pomôžte im. Určite viete, čo je to nahnevaná mamina...

Úloha: Je zadané n – počet domčekov v mestečku a m – počet uličiek medzi nimi. Na druhom riadku sú čísla a, b, c – a je Juliäin domček, b Rômeov a c je domček, kde sa nachádzajú. Na ďalších m riadkoch sú trojice čísel x, y, z. Tieto nám hovoria, že domček x je spojený s domčekom y obojsmernou ulicou kladnej dĺžky z. Nájdite dĺžku najdlhšej cesty, ktorú môžu Rômeo s Juliäou spolu prejsť, ak sa každou uličkou, ktorou prejdú, musia zároveň priblížiť ku svojim domčekom.

Príklad:

Vstup:	Výstup:
4 5 1 2 3 1 3 38 2 3 31 1 4 25 2 4 24 3 4 23	23

2324. O klinci

Náš malý nezbedník Mirko bol cez leto na prázdninách u babky a dedka. Najviac sa mu páčilo v dedkovej dielni, kde trávil veľa času. Naučil sa pracovať s kladivom, skrutkovačom alebo kliešťami. Najviac sa mu však páčila pílka na železo. Našiel kúsok plechu a vyrezal z neho namakaný pliešok, z diaľky pripomínajúci žabu. Keď už mal pliešok, tak sa šiel poobzerať do záhrady. Zrazu vidí, ako zo zeme trčí klinec. Skoro naňho stúpil!

Klinec sa mu veľmi páčil a tak skúsil položiť pliešok vodorovne na klinec tak, aby nespadol, teda zostal vo vodorovnej polohe. Skúšanie však veľmi rýchlo vzdal, lebo pliešok mu vždy spadol. Preto chce, aby ste mu pomohli. Pliešok má tvar mnohouholníka a je všade rovnako hrubý.

Úloha: Na vstupe je číslo n, udávajúce počet bodov plieška, za ním nasleduje n dvojíc $[x_i, y_i]$, ktoré určujú postupne po obvode súradnice n vrcholov mnohouholníka. Vašou úlohou je nájsť bod, v ktorom sa má pliešok dotýkať klinca tak, aby nespadol. Ak taký bod neexistuje, vypíšte o tom správu.

Príklad:

Vstup:	Výstup:
4 0 0 1 0 1 1 0 1	0.5 0.5

2325. Opäť dešifrovať?!

Zase raz tu pre vás máme úlohu, v ktorej dostanete zašifrovaný text a máte ho dešifrovať. Ale aby to nebolo také trápne jednoduché, povieme vám k tomu trochu tej matiky. Tentokrát totiž zadanú šifru pravdepodobne ručne dešifrovať nezvládnete.

Majme nejaký jav, ktorý nastáva s pravdepodobnosťou p. Hodnotu $\lg(1/p)$ (kde lg je logaritmus so základom 2) nazveme *prekvapením* z toho, že jav p nastal. (Čím je táto hodnota väčšia, tým je prekvapujúcejšie, že tento jav nastal.)

Napríklad keď hodím tromi mincami, šanca, že padnú samé znaky, je 1/8, prekvapenie z toho, že tento jav nastal, je $\lg(1/(1/8)) = \lg 8 = 3$.

Silným nástrojom kryptológov je štatistika. Prvé a najpriamočiarejšie, čo môžeme urobiť, je spočítať si frekvencie výskytov jednotlivých písmen. Zoberieme dosť veľký text a spočítame, ako často sa v ňom ktoré písmeno vyskytuje. Väčšinou zistíme, že najčastejšie sú samohlásky. (V slovenčine je to A, v češtine a anglosaských jazykoch je to E.)

Jednou z najjednoduchších šifier je substitučná šifra. Tej princíp je taký, že si zvolíme nejakú náhodnú permutáciu množiny písmen. Následne v otvorenom texte každé písmeno nahradíme jeho obrazom. Teda napríklad namiesto každého A napíšeme K, namiesto každého L napíšeme F, namiesto každého Z bude A, atď.

Keď sa snažíme dešifrovať substitučnú šifru, vieme na základe frekvencií písmen odhadnúť napríklad ktoré znaky sú zašifrované samohlásky, dokonca si podľa frekvencií jednotlivých znakov vieme pomerne spoľahlivo tipnúť jazyk, v ktorom je pôvodný text napísaný.

Ak by sme ale chceli počítač naučiť riešiť substitučné šifry, toto zďaleka nestačí. My vieme zvoliť konkrétne samohlásky a spoluhlásky tak, aby nám slová dávali zmysel, ale počítač nie. Potrebovali by sme mu nejako vysvetliť, ktorý výsledok vyzerá dobre a ktorý nie.

Jednou možnosťou je použiť wordlist, teda zoznam slov. Dobré budú také výsledky, v ktorých je veľa slov z nášho zoznamu. Ukážeme si ešte jednu, v praxi o dosť spoľahlivejšiu metódu, založenú na štatistike.

Zoberieme dostatočne veľkú vzorku textu písaného v nejakom jazyku a pre každú štvoricu po sebe idúcich písmen (odborne zvanú *tetragram*) si spočítame, koľkokrát sa v ňom vyskytuje. Na základe týchto údajov vieme pre každý tetragram odhadnúť pravdepodobnosť, s akou sa vyskytuje v texte.[4] Teraz vieme povedať, že riešenie šifry je tým lepšie, čím menej je prekvapujúce, že je písané v danom jazyku. (Teda pre daný reťazec n písmen, ktorý dostaneme, sčítame prekvapenie pre všetkých $n-3$ jeho tetragramov.)

No, dosť bolo návodu, tu je zadanie, dešifrujte ho. A ak si pri riešení nebodaj napíšete nejaký program, pošlite nám aj ten.

```
misof izntq ymios nvqvs cikdn rwpin lgvya ixsvr rsini kkitf icxai nicfr aitvx
xfire pvxdz ixucs inrec iligf mirwp ucinc linic xivyk lvaic rfsir fdrfv ntinq
inoin nfrcw psciu vpnxi pzyxl sizuv xsini nifuv inrwp icxfi cxoci guicf inixf
kinxf dsini cxpyl igrip nqcig rficg inuix xinic xixrw puini xnywt rvwtr wpgie
eixzy rrkyn nicwp yxsvn zlcgf svrri gaiic xizyx oiinr wpicx ficxi znicw pixri
pnqci gtgic xinvg ricxi zvnzi xkyna vriav ggrec iginl cgfvg rdaic icxin eiwpr
fnvpx irfic linfr cwpsv rliky pgrcw pzipn vxrfn ixlix oyzyr rixui rpvga sinav
ggrwp nyzek fscik nvlis civgg ixics inyzf nicaf agica fvggi nscxl rdkki xuvru
vnoyi nrfsv sinln drria vggds insin inkdg lsixx dasci fnikk rcwpi nixrd lyfrc
xsuic gsina vggky nrcir wpdxc zzinl ndrri nuvnd sinin rfsin inkdg lsixa vgguv
wprix gvrrf ucrri xvywp scier hwpdg dlixx cwpf
```

(Hint: Nik nepovedal, že to bude po slovensky…)

2331. O lúpeži

Zlatokopi Santo a Banto žijú vo Vyšnej Klondike. Kedysi tu bolo veľa zlata, ale teraz je už skoro všetko vyťažené. Prednedávnom prišli o prácu a pomaly sa im míňajú peniaze. Lovom rýb sa im živiť nechce, preto sa rozhodli vykradnúť banku.

Podarilo sa im totiž zistiť, že ak ukradnú sumu neprevyšujúcu s korún, banka si nič nevšimne a nebudú mať z toho žiadne problémy. Práve teraz sú v trezore a chcú si odtiaľ zobrať čo najviac peňazí. Preto by ste im mali pomôcť.

Úloha: Na vstupe sú čísla n a s – počet druhov bankoviek a maximálna suma, ktorú si môžu odniesť. Ďalej nasleduje n dvojíc celých čísel h_i a p_i, kde h_i je hodnota i-teho typu bankoviek a p_i je počet takýchto bankoviek v trezore. Vašou úlohou je zistiť, aká najvyššia suma menšia ako m sa dá z trezoru ukradnúť.

Rádové veľkosti čísel zo vstupu: n je do 10, m je rádovo 100 000, žiadne z čísel h_i a p_i neprekročí 1000.

Poznámka: Existuje viac efektívnych riešení. To očividnejšie je menej efektívne.

Vstup:	Výstup:
3 47 7 2 8 4 6 2	46

[4] Jeden by čakal, že tento odhad spravíme tak, že počet výskytov daného tetragramu vydelíme ich počtom. Nuž, nie celkom. Lepšie je ešte pred delením napríklad pridať každému tetragramu jeden výskyt, vyhneme sa tak nepríjemnostiam, ak sa nám v šifre vyskytne podľa našej štatistiky „nemožný“ tetragram.

2332. O Zázvorovom pive III

Zázvorník Zachariáš sa živí distribúciou zázvorového piva Zázvorové pivo III™. Viete ako to chodí, ženy sú večne doma a muži v krčme, a tak Zachariáš dňom i nocou bohatne.

Rozvoz Zázvorového piva III™ funguje nasledovne: V niektorých mestách sú sklady s prakticky neobmedzenými zásobami, a v každom sklade je jeden závozník. Vždy, keď niekam treba dodať nové sudy piva, Zachariáš zavolá niektorému závozníkovi, ten naloží sudy do auta a najkratšou cestou ich na miesto určenia dovezie.

Má to však jeden háčik. Zachariáš často nevie, ktorého závozníka poslať doručiť pivo. A tak sa bežne stane, že niektorý iný závozník by to mal na dané miesto bližšie ako ten, ktorého Zachariáš poslal. A s rastúcimi cenami benzínu je toto čím ďalej, tým vážnejší problém. Preto sa Zachariáš rozhodol, že úplne nutne potrebuje mapu, z ktorej by vedel pre ľubovoľné miesto (v meste alebo aj na niektorej ceste) rýchlo povedať, ktorého závozníka tam poslať.

Úloha: Na vstupe je daný počet miest n a počet závozníkov z. Mestá sú očíslované 1 až n tak, že závozníci sú v mestách 1 až z. Ďalej nasleduje počet ciest m zoznam m ciest. Trojica „u v d“ znamená, že medzi mestami u a v vedie cesta dĺžky d km. Úlohou je pre každú cestu vypísať, ktorý závozník to má ku ktorej jej časti najbližšie.

Príklad:

Vstup:	Výstup:

`4 3`
`4`
`1 4 1`
`1 2 4`
`4 2 3`
`3 4 5`
	`cesta (1,4): hocikam na tejto ceste to má najbližšie závozník č. 1`
	`cesta (1,2): prvé 2 km je najbližšie č. 1, zvyšok cesty je to č. 2`
	`cesta (4,2): prvý 1 km je najbližšie č. 1, zvyšok cesty je to č. 2`
	`cesta (4,3): prvé 2 km je najbližšie č. 1, zvyšok cesty je to č. 3`

2333. Odpovedz, je tam hrana?

Vedcom z Institute of Black Magic (IBM) už v minulosti párkrát úspešne pomohli riešitelia KSP. A aj tentoraz potrebujú vašu pomoc. Dostali za úlohu vytvoriť program, ktorý dostane na vstupe najprv popis nejakého planárneho grafu a potom bude musieť interaktívne odpovedať na otázky, či je alebo nie je medzi danými vrcholmi hrana.

Pre tých, čo nevedia, čo je planárny graf: Pod pojmom graf si môžeme predstaviť nejakú množinu bodov nakreslených na papieri, z ktorých niektoré dvojice bodov sú pospájané čiarami. Bodom budeme hovoriť vrcholy a čiaram, ktoré ich spájajú, hrany. Medzi každou dvojicou vrcholov vedie najviac jedna hrana. Planárny graf je taký graf, ktorý vieme nakresliť do roviny (na hárok papiera) tak, aby sa žiadne dve hrany nikde nepretínali. (Nanajvýš môžu mať spoločný jeden koniec – vrchol, z ktorého obe vychádzajú.)

Príkladom grafu, ktorý nie je planárny, je graf pozostávajúci z piatich vrcholov, z ktorých každé dva sú spojené hranou. (T.j. dokopy máme 10 hrán.) Tento graf sa nedá nakresliť do roviny tak, aby sa nejaké dve jeho hrany „niekde v strede“ nepretli.

Úloha: Na vstupe váš program dostane počet vrcholov n a počet hrán m v grafe. Vrcholy sú očíslované od 1 po n. Ďalej nasleduje m dvojíc čísel vrcholov, ktoré sú spojené hranou. Medzi každou dvojicou vrcholov vedie najviac jedna hrana. Môžete predpokladať, že graf na vstupe je planárny.

Potom bude váš program v nekonečnom cykle interaktívne odpovedať na otázky, či je alebo nie je medzi danou dvojicou vrcholov hrana. Teda v každej otázke sa ho opýtame na dvojicu čísel a má odpovedať „`ÁNO`“ alebo „`NIE`“ podľa toho, či sa táto dvojica nachádzala v popise grafu (na poradí prvkov v dvojici samozrejme nezáleží). Otázky bude dostávať jednu po druhej – vždy, keď nejakú zodpovie, dostane ďalšiu.

Dobrá rada: Vedcom z IBM sa podarilo objaviť dôležitú vlastnosť planárnych grafov: Pre každé $n \geq 3$ platí, že ak má planárny graf n vrcholov, tak má maximálne $3n - 6$ hrán.

Kritériá hodnotenia vašich programov:
- Najdôležitejšie je, že váš program by mal používať len lineárne veľa pamäte od veľkosti vstupného grafu. (Ak túto požiadavku nesplní, dostane veľmi málo bodov.)
- Na otázky by mal odpovedať čo najrýchlejšie.
- Pri/tesne po načítaní grafu môže váš program stráviť nejaký čas predpočítavaním nejakých údajov. Ak máme dva programy, ktoré na otázky odpovedajú rovnako rýchlo, lepší je ten, ktorý potrebuje na predpočítanie kratší čas.

Príklad:

Vstup:	Komunikácia:
`5 3`	`> 2 1`
`1 2`	`ÁNO`
`2 3`	`> 1 4`
`2 4`	`NIE`
	`> 5 3`
	`NIE`
	`> 2 3`
	`ÁNO`

2334. O skokanoch

Exotický šach je hra, ktorá funguje podobne ako obyčajný, len sa napríklad použijú iné figúrky. Jedným typom figúrok sú *skokani*. Skokan sa dá popísať konečnou množinou povolených skokov, teda políčok, na ktoré vie zo svojho políčka skočiť bez ohľadu na to, aké figúrky sú „po ceste“. Každý skok vieme popísať dvojicou čísel $[x, y]$, ktoré znamenajú „posuň sa o x políčok doprava a o y dopredu“.

Požadujeme však, aby množina možných skokov bola symetrická so stredom symetrie v mieste, kde figúrka stojí – teda ak vie skokan niekam skočiť, musí vedieť v nasledujúcom ťahu skočiť späť.

Príkladom skokana je klasická šachová figúrka *jazdec*. Tú môžeme popísať množinou povolených skokov: $[1, 2]$, $[1, -2]$, $[2, 1]$, $[2, -1]$ a skoky k nim opačné.

Predstavme si teraz, že skokana položíme na nekonečnú štvorčekovú sieť na políčko so súradnicami $[0, 0]$. Ofarbime na zeleno políčka, kam sa vie konečným počtom skokov dostať. Tieto políčka nazveme jeho *teritóriom*.

Úloha: Na vstupe sú popisy dvoch skokanov. Zistite, či majú rovnaké teritóriá.

Príklad:

Vstup:	Výstup:
`prvý: 1 2  1 -2  2 1  2 -1` `druhý: 1 0  0 1  1 1  1 -1`	`ANO` *(Prvý skokan je jazdec, druhý je kráľ, teritóriom oboch je celá rovina.)*
Vstup:	Výstup:
`prvý: 1 2  2 1` `druhý: 3 0  0 4`	`NIE` *(Prvý sa vie dostať na $[3, 3]$, druhý nie.)*

2335. Optimálna cestná sieť

Absurdistan je široká ďaleká rovina a v tejto rovine leží n miest. Donedávna si tam tieto mestá len tak ležali a obyvatelia sa medzi nimi prevážali na koňoch. Až prišiel osudný deň, keď si šejk Al-adár dal doviezť z Európy auto. Teraz si preň dal postaviť cestnú sieť, po ktorej by sa dalo dostať z ľubovoľného mesta do ľubovoľného iného. Ako to ale chodí, miestni obyvatelia sú leniví, preto chcú túto cestnú sieť postaviť čo najkratšiu.

To ale nie je len tak ľahké, ako sa možno na prvý pohľad zdá. Zamyslime sa napríklad nad situáciou, keď chceme spojiť tri mestá, ktoré sú vo vrcholoch rovnostranného trojuholníka. Najlepším riešením v tomto prípade je spraviť v ťažisku trojuholníka križovatku a postaviť cesty z nej do všetkých troch vrcholov.

Úloha: Vašou úlohou bude nájsť čo najlepšie (nie nutne optimálne!) riešenie pre niekoľko konkrétnych vstupov.

Formát každého vstupného súboru je nasledujúci:

Na prvom riadku je počet miest n ($1 \leq n \leq 1\,000$). Nasleduje n riadkov, v i-tom z nich sú dve celé čísla x_i, y_i (medzi $-300\,000$ a $300\,000$, vrátane) – súradnice jedného z miest. Mestá sú očíslované od 1 do n v poradí, v akom sú zadané vo vstupnom súbore.

Formát výstupného súboru by mal byť nasledujúci:

V prvom riadku vášho súboru bude celé číslo m ($0 \leq m \leq n$) – počet nových križovatiek, ktoré chcete postaviť. Nasleduje m riadkov, každý z nich obsahuje dve reálne čísla (medzi $-300\,000$ a $300\,000$, vrátane, v ľubovoľnom štandardnom formáte) – súradnice jednej križovatky. Križovatky očíslujeme od $n+1$ do $n+m$ v poradí, v akom sú uvedené vo vašom výstupnom súbore.

Zvyšok výstupného súboru tvorí $n + m - 1$ riadkov. Každý z nich obsahuje dve čísla miest/križovatiek, ktoré chcete vo svojej cestnej sieti spojiť priamou cestou.

(Postavené cesty musia vytvoriť súvislú cestnú sieť. V optimálnom riešení sa zjavne žiadne dve cesty nepretínajú. Ak sa vo vašom riešení nejaké cesty pretínať budú, považujeme toto miesto za mimoúrovňovú križovatku, teda auto tam nemôže prejsť z jednej cesty na druhú.)

Zdôrazňujeme, že každé zlepšenie riešenia je dôležité. Čím bližšie k optimálnemu riešeniu sa dostanete, tým bude zlepšovanie riešení ťažšie, ale každý dosiahnutý úspech cennejší.

Historická poznámka: Riešenia tejto úlohy sa odovzdávali v elektronickej podobe a boli automaticky vyhodnocované.

Príklad:

Vstup:	Výstup:
`3`	`1`
`0 0`	`500 288.67`
`1000 0`	`1 4`
`500 866`	`2 4`
	`4 3`

2341. O guličkách a tuneloch

Raz si Marek doma upratoval a našiel guličky. A nie také obyčajné, ale dokonca aj farebné. A potom našiel rúrky. Také, do ktorých sa zmestila každá gulička. Tak zobral lepidlo a začal stavať. Postavil jedno ohromné dielo. Samé prepojené rúrky. Navrchu bol lievik, kadiaľ sa hádzali dnu, na spodku zase miska, kam vypadávali. A navyše to postavil tak, že sa dalo prejsť každou rúrkou. A začal zhora hádzať guličky. V diele bolo veľa rozdvojek, roztrojek a mnoho iných križovatiek, guličky chodili po rôznych cestách... až Mareka začalo trápiť svedomie, že si tu hádže guľkami namiesto robenia úlohy z Úvodu do kombinatoriky a teórie grafov. Aby svedomie odbil, rozhodol sa, že zistí, koľko existuje možných ciest z vrchu až úplne nadol. Ale už bol neskorý večer, a tak vás požiadal o pomoc.

Úloha: Na vstupe sú dve prirodzené čísla n a m, kde n označuje počet uzlov a m označuje počet rúrok. Potom nasleduje m popisov jednotlivých rúrok. Popis rúrky tvoria čísla a_i, b_i, pričom a_i určuje, z ktorého uzla vedie i-ta rúrka, a b_i určuje, kam daná rúrka vedie. Gulička nevie ísť v opačnom smere (čiže z b_i do a_i). Všeobecnejšie, môžete predpokladať, že ak sa dá cez rúrky dostať z uzla u do uzla v, tak sa nedá dostať z uzla v do uzla u. Vašou úlohou bude vyrátať, koľko existuje rôznych ciest z uzla 1 do uzla n.

Príklad:

Vstup:	Výstup:
`7`	`3`
`8`	
`1 2  1 6  2 4  2 5`	*(Sú to cesty* $1 \to 6 \to 3 \to 7$,
`6 3  4 7  5 7  3 7`	$1 \to 2 \to 4 \to 7$ a $1 \to 2 \to 5 \to 7$.*)*

2342. Odíď vírus

Kto už na tomto svete nepočul o počítačových vírusoch – kúskoch kódu, ktoré sa vedia pripojiť k spustiteľným súborom. Keď sa daný súbor spustí, spustia sa aj oni a vedia narobiť veľké škody. Pri podozrení, že sa v počítači nachádza vírus, je potrebné prehľadať všetky súbory, vírusy nájsť a odstrániť. Vašou úlohou bude naprogramovať taký veľmi jednoduchý antivírus.

Úloha: Na vstupe dostanete číslo n a n postupností núl a jednotiek $v_1, \ldots, v_n$ (popisy jednotlivých vírusov). Ďalej nasleduje s – ďalšia postupnosť núl a jednotiek (skúmaný súbor). Pre každý vírus v_i vypíšte všetky jeho výskyty v súbore s alebo správu, že daný vírus sa v súbore nenachádza. Môžete predpokladať, že kód žiadneho vírusu nie je obsiahnutý v kóde nejakého iného – t.j. žiadne v_i nie je podreťazcom (súvislou podpostupnosťou) nejakého v_j pre $j \neq i$.

Príklad:

Vstup:	Výstup:
`3`	`0011 sa nachádza na pozíciách 1 a 9`
`0011 010 10110`	`010 sa nachádza na pozíciách 5 a 7`
`001101010011001`	`10110 sa v súbore nenachádza`

2343. O podenkách

Pri sťahovaní KSP sme objavili v zabudnutom rohu i33 akvárium, kde žili podenky. Podenky sú taký zvláštny druh vodného hmyzu – žijú iba pár hodín, počas ktorých nakladú vajíčka a neskôr zahynú. Z týchto vajíčok sa v ďalšej generácii vyliahnu podenky, ktoré žijú tiež len veľmi krátko a cyklus pokračuje...

Keďže v novej miestnosti je málo miesta, akvárium budeme musieť vyhodiť. Zároveň sme však mierumilovní a nechceme zabiť ani jednu podenku. Pomôžte nám zistiť pravdepodobnosť, že po m generáciach naše podenky vyhynú.

Úloha: Na vstupe sú čísla k a n. Číslo k je aktuálny počet podeniek v akváriu. Ďalej nasleduje n hodnôt – pravdepododobnosti $p_0, p_1, \ldots, p_{n-1}$. Pravdepodobnosť p_i určuje, s akou pravdepodobnosťou jedna podenka nakladie i vajíčok. Súčet všetkých p_i je samozrejme 1. Nakoniec je na vstupe číslo m. Vašou úlohou je zistiť pravdepodobnosť, že po m generáciách nebude žiť žiadna podenka.

Príklad:

Vstup:	Výstup:
`1 3`	`0.478`
`0.33 0.34 0.33`	
`2`	

2344. O duplikátoch a pánovi Dirichletovi

„Viac vľavo. Nie, to bolo príliš. Trošku naspäť vpravo.“
„Je to už dobré?“
„No, ešte to posuň trošku hore... správne. Teraz je to dobre.“
„Super, daj ďalší plagát. Počkaj, ale to je IOI 2000, to som už niekde lepil...“
„Ľaľa, veď tam visí na stene...“
„Ale ako je to možné? Žeby sme mali viac rovnakých plagátov?“
„A koľko je vlastne všetkých?“
„$n + 1$.“
„Tak veľa?“

Problém je totiž nasledovný. KSPáci sa presťahovali do novej miestnosti a začali si tam lepiť plagáty. Ale zistili, že nejaké sa opakujú. Ako tak začali nad tým maturovať, tak im napadla nasledovná úloha a uvedomili si, že by to bola zaujímavá úloha do KSP. Tak teda, tu ju máte...

Úloha: Máte číslo n a pole $A[0\,..\,n]$, v ktorom sú uložené čísla od 1 po n. Keďže dĺžka poľa je $n+1$ a čísel je len n, nachádza sa aspoň jedno číslo v poli (aspoň) dvakrát. Vašou úlohou je nájsť jedno takéto číslo. Ak je ich viac, stačí ľubovoľné z nich. Jediné obmedzenie, ktoré máte, je, že pole nesmiete prepisovať.

Riešenie je tým lepšie, čím je menší súčin jeho časovej a pamäťovej zložitosti. (Pod pamäťou sa myslia pomocné premenné okrem samotného poľa.) Úplne najlepšie je teda riešenie v lineárnom čase, ktoré si vystačí s konštantným počtom pomocných premenných.

Príklad:

Vstup:	Výstup:
`5`	`1`
`1 3 2 4 1 2`	

2345. O tabuľke profesora Kolmogorova

Na stole profesora Kolmogorova ležala tabuľka, husto popísaná číslami. Vyzerala asi takto:

x	**f(x)**	**x**	**f(x)**	**x**	**f(x)**	**x**	**f(x)**
1	47	13	68	125	90	47300	171957387
2	47	14	65	126	103	47400	5975472
3	47	15	66	970	2544	789234	11737933
4	47	16	79	1003	2734	789235	11739988
5	47	17	85	1005	2719	789236	11742111
6	47	18	91	7410	146055	789237	11744214
7	47	19	89	7411	146035	1234567	32762148
8	71	20	103	15233	617165	2345678	12322037
9	72	47	302	35233	95383914	34567890	27469977
10	77	74	701	35234	95389328		
11	78	123	90	36233	100878311		
12	67	124	90	36234	100883886		

„Andrej Nikolajevič, čože to máte za tabuľku?“ zaujímala sa jeho diplomantka Nataša.
„Ále, to nič nie je…“ zamrmlal roztržito profesor, no po chvíli dodal:
„Ako dlho by vám trvalo zapamätať si ju celú?“
Nataša chvíľku hľadela na tabuľku, a potom odpovedala: „Toľkoto.“
Profesor uznanlivo pokýval hlavou.

Úloha: Natašin „trik“ spočíval v tom, že pochopila, ako profesorova tabuľka vznikla. Skúste to aj vy. Napíšte **čo najkratší** program, ktorý bude počítať funkciu uvedenú v tabuľke. (Je úplne jedno, aký výsledok vráti pre iné hodnoty x ako tie uvedené v tabuľke.)

Vstupom pre váš program bude súbor obsahujúci presne 470 riadkov. V každom z nich je jedno z čísel, ktoré sú uvedené v tabuľke v stĺpci „x“. Výstup vášho programu musí mať presne 470 riadkov, pričom ak označíme číslo v i-tom riadku vstupu x_i, tak v i-tom riadku výstupu musí byť číslo $f(x_i)$.

Pri hodnotení do KSP nezáleží na voľbe programovacieho jazyka, pokúsime sa, aby to isté riešenie napísané v Pascale aj v C dostalo do KSP rovnako bodov.

Historická poznámka: Riešenia tejto úlohy sa odovzdávali v elektronickej podobe a boli automaticky vyhodnocované.

Príklad:

Vstup:	Výstup:
`15`	`66`
`3`	`47`
`123`	`90`
…	…

z2311. Zas ten problém s medovníkmi

Jergušova babička pečie každú nedeľu skvelé medovníky. „Nie že ich zješ všetky sám," varuje ho vždy, a tak Jerguš nemôže inak, len sa podeliť. „Ale s kým?" premýšľa. Jerguš navštevuje veľa záujmových krúžkov (okrem krotenia divých levov ho bavia aj pokojnejšie veci, napríklad paragliding). Každý týždeň teda smutne hľadí na babičkine medovníky a úporne premýšľa, či by sa mal podeliť s krúžkom šachu, krúžkom krížikovej výšivky alebo úplne iným krúžkom. Jedno vie určite: každý z ľudí, s ktorými sa podelí, musí dostať rovnako veľa medovníkov (vrodený zmysel pre spravodlivosť). Iste by mu pomohlo, keby vedel aspoň to, ako všelijako sa dá daný počet medovníkov deliť. Keď potom na nejaký krúžok príde vhodný počet ľudí, podelí sa s nimi. A ak nie, nuž, nebude mať inú možnosť, ako zjesť to všetko sám. Aká škoda.

Úloha: Vstupom programu bude jedno prirodzené číslo. Vašou úlohou je vypísať všetky delitele čísla vo vzostupnom poradí.

Príklad:

Vstup:	Výstup:
`12`	`1 2 3 4 6 12`

z2312. Zúfalý slimák

Slimák Jožko chcel byť od narodenia špičkovým programátorom. A teraz je najšťastnejší slimák na svete. Firma slimPC, svetoznámy dodávateľ slimačích počítačov, si vybrala práve Jožka, aby naprogramoval jadro slimák-friendly operačného systému. Toto jadro musí vykonávať tú najdôležitejšiu funkciu, ktorá zaručuje, že operačný systém bude slimák-friendly: Keď si nejaký slimák spustí program, počítač mu musí oznámiť dátum,kedy program skončí, aby slimák načas prišiel späť k počítaču a pokračoval v práci. Jadro samozrejme pozná aktuálny dátum a vie počet dní, koľko trvá vykonanie programu.

Má to ale malý háčik. Jožko sedel za počítačom iba dvakrát, raz keď sa u kamaráta hral SlimTournament II a druhýkrát, keď sedel na lúke a nejaký človek si vedľa neho položil svoj notebook. Práve preto vás zúfalý slimák Jožko teraz žiada o pomoc.

Úloha: Váš program by mal načítať 4 prirodzené čísla d, m, r a k. Prvé tri čísla sú deň, mesiac a rok aktuálneho dátumu, číslo k udáva dĺžku behu programu. Ako výstup vypíšte dátum dňa, ktorý bude o k dní od zadaného dátumu.

Nezabudnite, že každý rok deliteľný 4, ale nedeliteľný 100 je priestupný, s výnimkou rokov deliteľných 400. Teda roky 2000 a 2004 sú priestupné, ale 1900 nie je priestupný. Môžete predpokladať, že všetky dátumy sú zadávané v našom, t.j. gregoriánskom kalendári.

Príklad:

Vstup:	Výstup:
`4 2 1996 42`	`17 3 1996`
Vstup:	Výstup:
`18 8 1881 47424`	`22 6 2011`

z2313. Zase som prehral...

„Zase som prehral," zúfalo priznal svoju desiatu prehru za sebou v piškvorkách Marek a podal ruku svojmu skvelému súperovi Rasťovi. Rasťo sa tešil, že Mareka definitívne porazil, keď v tom zbadal, ako sa v Marekovych očiach zablýskalo a následne sa v nich rozhorel plamienok nádeje. Malý, ale predsa. I riekol Marek: „Posledná prehra sa nepočíta, lebo presne takto isto si ma porazil minule." Svoje tvrdenie mohol ľahko dokázať, vďaka tomu, že sa hrali na jeho notebooku a postupnosť ťahov v každej hre sa automaticky zaznamenávala do súboru. Ak chce Rasťo Mareka pokoriť, musí pristať na jeho podmienku, že hru, ktorú už niekedy predtým odohrali, nebudú rátať do skóre. Rasťa by už teraz zaujímalo, aké najväčšie skóre môže dosiahnuť v ideálnom prípade. Inými slovami, chcel by vedieť, koľko je rôznych priebehov hry, pri ktorých vyhrá. No a Mareka pochopiteľne zaujíma zase opačná otázka.

Úloha: Napíšte program, ktorý vypočíta, koľko je legálnych priebehov hry piškvorky 3×3 a v koľkých z nich vyhrá prvý hráč. (Pozor, tej istej záverečnej pozícii môže zodpovedať viacero možných priebehov hry!)

Pravidlá: Hrá sa na mriežke s 3×3 políčkami, hráči ťahajú striedavo. V každom ťahu hráč do ľubovoľného voľného políčka zaznačí svoj symbol. Hra sa končí víťazstvom v okamihu, keď hráč svojim ťahom spôsobí výskyt troch rovnakých symbolov v jednom riadku, stĺpci alebo uhlopriečke. Ak nikto nevyhrá, hra končí po deviatich ťahoch remízou.

z2314. Zákerná štatistika

Verte-neverte, je to tak. Je jedným zo základných pilierov našej konzumnej spoločnosti. Teda... nie len jej. Už v starom Egypte či Mezopotámii nad ťažko pracujúcimi bdel jeden či viac privilegovaných štatistov, ktorí poctivo uzlíkmi zaznamenávali vykonanú prácu. Staré krčmárky sledovali predajnosť piva podľa objemu vody, s ktorým pivo riedili.

Pretrváva to tak až dodnes. Napríklad také potraviny. Súčasť nášho každodenného života. Hoc si to neuvedomujete, ale práve v potravinách si tej štatistiky užijete požehnane. Začínajúc využitím pokladní, pokračujúc odbytom jednotlivých položiek, odbytom jednotlivých položiek v závislosti od ceny, umiestnenia v regáli, od veku kupujúceho, od zastúpenia zelenej farby na obale daného tovaru, končiac predajnosťou bratislavskej salámy podľa vlhkosti ovzdušia, účesu a veľkosti výstrihu predavačky. Jednoducho povedané, psychologický útok na nič netušiacich klientov podložený dlhodobým matematickým bádaním nás dennodenne pripravuje o nemalé množstvo financií, ktoré v obchode zanecháme práve kvôli štatisticko-psychologickým pascám.

Úloha: Vašou úlohou bude sledovať v priebehu dňa pokladňu. Na konci dňa budeme chcieť, aby program vypísal nasledujúce informácie: Príchod prvého klienta, odchod posledného, najväčšiu dĺžku radu a najdlhšie čakajúceho klienta. Na vstupe dostanete n – počet riadkov vstupu a n riadkov vo formáte *hh*:*mm t meno*. Čísla *hh*:*mm* udávajú čas, kedy sa zákazník menom *meno* postavil do radua t je čas v minútach, potrebný na vybavenie jeho požiadavky.

Príklad:

Vstup:

```
6
02:51 10 Monika
02:53 15 WSX
08:02 65 KuKo
11:12 5 Janka
11:12 6 Misof
11:13 10 Zuzuliak
```

Výstup:

```
Prichod prveho: 02:51
Odchod posledneho: 11:33
Najdlhsi rad: 3
Najdlhsie cakajuci: Zuzuliak 00:10
```

z2315. Z koláča dieru

Na obdĺžnikovom plechu upiekli koláč rozmerov $r \times s$. Akonáhle deti zistili, že „vzduch je čistý“, pustili sa do koláča. Mimochodom nebol veľmi dobrý, a tak zjedli iba kúsok. Keďže sa veľmi nemali k jedeniu, začali si krátiť dlhú chvíľu krájaním... Samozrejme, od nudy sa koláč nekrája len tak hocijako – je treba vykrojiť čo najzaujímavejšie útvary: začalo to rôznymi hviezdičkami a špirálami, končilo prasiatkami a lamami.

Keď sa vrátila mamička, no povedzme si to otvorene, koláč bol už statne rozkrájaný. Už-už sa išla mamička nahnevať, keď si spomenula, že koláču chýbajú hrozienka. (Nie že by to koláč zachránilo, ale recept je recept.) Avšak hrozienok bolo málo – sotva na každý kúsok jedno. A tu sa zarazila. Kam majú hrozienka poukladať, aby dali na každý kúsok koláča práve jedno? I začali všetci skúmavými pohľadmi premeriavať koláč, div si oči nevyočili. Ba veruže si ich temer vyočili. Tak sa zhodli, že táto úloha nie je súca pre nich, ale pre riešiteľov KSP.

Úloha: Na vstupe máme zadané dve celé čísla r a s a potom „mapu“ koláča, ktorá má r riadkov a s stĺpcov. Koláč je vyznačený bodkami, zárezy a iné hranice kúskov inými znakmi (mriežkami, lomítkami, pomlčkami, ...). Úlohou je na výstup nakresliť tento istý koláč a na ňom hrozienka (hviezdičky) tak, aby na každom kúsku koláča bolo práve jedno hrozienko. Kúsok koláča je časť, ktorá drží pokope. Dve bodky na obrázku držia pokope, ak sú hneď vedľa seba, alebo hneď pod sebou (nie šikmo).

Správnych rozmiestnení hrozienok môže byť veľa, uspokojíme sa s ľubovoľným z nich.

Príklad:

Vstup:

```
12 31
.........../...../...........|...
.--./.../../--------\...----.|.
..\/.\./../..........\.........
\./....|./../------\.-------/--
.\..--.|.|..|......|/....\.....
..\....|.|..\----\.|----.--.|./
/..\.--\.\.......|.|..|...|../.
./..\.....\------/....|.--../..
./...\..\.........../..\..-|/.--
..\\..\..\--------/.\-.\../....
/..\...\....\..\...\..\./.--##
./../.\.\./.../.../..././....##
```

Výstup:

```
....*..../...../...........|...
.--./.../../--------\.*.----.|.
..\/.\./../..........\.........
\./....|./../------\.-------/--
.\..--.|.|.*|.*....|/...*\.....
.*\....|.|..\----\.|----.--*|./
/..\.--\.\.......|.|..|...|../.
./..\.....\------/....|.--../.*
./...\..\.........../..\..-|/.--
..\\..\.*\--------/.\-.\../..*.
-..\...\....\...\...\..\./.--##
*|.*/.\*\./.../.../.../*/.*..##
```

z2321. Zanzibarské lamy

Mirko sa minulý týždeň vydal do Zanzibaru, aby si tam užil krásnu dovolenku. Samozrejme, aby sme mu nedopriali toľko zábavy, naložili sme mu nejaké úlohy. Jednou z nich bolo sledovať jeho obľúbené zvieratká. Lamy sú zvlášte stvorenia, často robia hlúposti. Keď sú v jednej ohrade dva druhy, ktoré sa nemajú rady, ešte sa aj pohryzú! Preto si Mirko vždy, keď uvidí dve rôzne lamy pokope, zapíše, že spolu vydržia.

Z učebnice biológie vieme, že ak vydržia dve lamy pokope, tak každá vydrží so všetkými, s ktorými vydrží tá druhá. Vo voľnej prírode sa však lamy, ktoré sa nemajú rady, obchádzajú širokým oblúkom, preto Mirko nemá ako túto skutočnosť zistiť. Zrazu mu volá Bee: „Mirko, máme tu lamu číslo 47 a lamu číslo 42, môžeme ich dať spolu do ohrady?“

Úloha: Na začiatku dostanete jedno celé kladné číslo n, určujúce počet lám v Zanzibare. Na vstupe dostávame tri druhy údajov. Pokiaľ Mirko uvidí lamy číslo x a y pokope, riadok vstupu obsahuje tri čísla „1 x y“. Pri otázke, či môžeme strčiť lamy s číslami x a y do jednej ohrady, riadok vstupu obsahuje tri čísla „2 x y“. Pokiaľ je jediným číslom v riadku vstupu 0, program môžeme ukončiť a Mirko ide domov.

Príklad:

Vstup:

```
10
1 1 2
1 2 3
1 4 5
1 3 4
1 6 7
1 6 8
1 6 9
2 7 9
2 1 5
2 1 9
1 5 6
2 1 9
0
```

Výstup:

```
ANO
ANO
RADSEJ NIE
ANO
```

(Pri odpovedaní na prvú otázku vieme, že lama 7 bola videná s lamou 6 a lama 6 s lamou 9, preto môžu byť lamy 7 a 9 spolu.

Druhá otázka analogicky. Tretia otázka: Medzi lamami 1 a 9 neboli preskúmané žiadne vzťahy, preto nemôžu ísť spolu do klietky.

Medzi treťou a štvrtou otázkou boli spolu videné lamy 5 a 6. Preto pri štvrtej otázke už pokojne môžeme dať lamy 1 a 9 do jednej klietky.)

z2322. Zabudnutý zošit

Nedávno sa mi stalo, že som našla zošit. Začala som ním listovať, aby som zistila, kto je jeho majiteľom. Zarazilo ma, keď som uvidela to, čo som videla. Ten človek sa snažil rátať v nejakej divnej sústave. Síce dokázal prehadzovať čísla z našej štandardnej sústavy do jeho sústavy, ale darmo spočítaval čísla. Nešlo mu to. Ale vám to určite pôjde.

Táto divná sústava má názov n-adická. Je to niečo podobné ako n-ková (n-árna) sústava, ale cifry nie sú od 0 po $n-1$, ale od 1 po n. Podobne ako v n-árnej sústave, číslu $\overline{a_k a_{k-1} \ldots a_1 a_0}$ zodpovedá v desiatkovej sústave číslo $a_k \cdot n^k + a_{k-1} \cdot n^{k-1} + \ldots + a_1 \cdot n + a_0$.

Napríklad v 2-adickej sústave našim číslam 1, 2, 3, 4, 5, 6, 7 zodpovedajú čísla $\overline{1}$, $\overline{2}$, $\overline{11}$, $\overline{12}$, $\overline{21}$, $\overline{22}$, $\overline{111}$.

Úloha: Na vstupe je číslo n – v akej sústave budete počítať. Ďalej budú zadané dve čísla zapísané v n-adickej sústave. Môžete predpokladať, že $n < 10$. Vašou úlohou je spočítať tieto dve čísla a výsledok vypísať opäť v n-adickej sústave.

Príklad:

Vstup:	Výstup:

`5`
`345411`
`5251331`
	`11152242`

z2323. Zase tie podvody...

„Už zase podvádzaš!“ obvinil Santo Banta. „Čože? Ja nikdy nepodvádzam!“ odvetil Banto tradične a už odchádzal so sklonenou hlavou preč s tým, že ani tento rok nevyhrá...

Santo a Banto každý rok na jeseň chodia na Kiribati na turnaj v Hyperkockách. Ale Banta vždy pristihnú pri podvádzaní. On totiž pozná také zaklínadlo, ktoré spôsobí, že daný zápas určite vyhrá. (Pri zaklínaní sa treba 47-krát sa otočiť na stoličke a pritom kričať „Som syr! Som syr!“) Ale samozrejme pri takomto podvádzaní je určitá šanca odhalenia. A asi sa príliš nedivíte, že Banta doteraz vždy prichytili.

Ako vlastne taký turnaj vyzerá? Prídu súťažiaci (úplnou náhodou ich počet je nejaká mocnina dvojky) a postavia sa do radu. Zoberú sa prví dvaja, tí si zahrajú hru Hyperkocky. Víťaz sa postaví do ďalšieho radu, porazený vypadáva a ide domov. Potom sa zoberú ďalší dvaja z prvého radu a spravia to isté. Takto sa pokračuje, až kým nie je prvý rad prázdny. No a potom sa robí to isté s vytvoreným radom, a celé opakuje sa to dovtedy, kým nezostane práve jeden víťaz. Skrátka taký tradičný turnajový pavúk, ibaže zložito popísaný. Ale vráťme sa k Bantovi.

Banto strávil celý rok tým, že skúmal, kto zo súťažiacich je ako dobrý v Hyperkockách. Forma každého súťažiaceho (vrátane Banta) sa dá popísať nejakým celým číslom, pričom ak sa stretnú dvaja súťažiaci, vyhráva ten, ktorý má lepšiu (t.j. väčšiu) formu. (Ak majú obaja rovnakú formu, vyhráva náhodný z nich dvoch.)

Banto si vymyslel, že nechá ostatných súťažiacich nastúpiť do radu, on príde ako posledný a vopchá sa medzi nich tak, aby dokázal celý turnaj vyhrať a čo najmenej pri tom podvádzať. Vhodné miesto v rade si ale bude musieť nájsť rýchlo, preto by potreboval program, ktorý mu všetko spočíta.

Úloha: Úlohou je napísať program, ktorý povie Bantovi koľko najmenej krát bude musieť podvádzať, aby s istotou vyhral. Na vstupe sú dve čísla n a b – počet hráčov (môžete predpokladať, že je to mocnina dvoch) a Bantova forma. Ďalej nasleduje $n-1$ čísel, ktoré udávajú formu jednotlivých Bantových protivníkov v poradí, v akom sa postavili do radu. Banto musí podvádzať vždy, keď hrá s aspoň tak dobrým súperom, ako je on sám.

Táto úloha nie je taká ľahká, ako sa na prvý pohľad môže zdať. Snažte sa, aby váš program bol čo najrýchlejší.

Príklad:

Vstup:	Výstup:
8 47 10 20 30 100 50 4 74	1 *(Banto sa môže postaviť na prvé až štvrté miesto, prvé dva zápasy vyhrá,vo finále bude hrať s človekom, ktorého forma je 100, a teda musí podvádzať.)*

z2324. Zase ty!

Nedávno sa stretli dvaja kamaráti, v posledných rokoch rivali na pohľadanie. Jeden sa volal Dave Orton a druhý Jen-Hsun Huang. Poklábosili, čo je, čo bolo a dostali sa k téme, kde nastal rozpor. Dave tvrdil, že vie vypočítať obsah prieniku a zjednotenia dvoch trojuholníkov rýchlejšie ako Jen. Jen tvrdil opak. Ani jeden z nich však nepovedal, ako to počíta. Čo viac, keď sa pokúsili spočítať úlohu z učebnice, dostali rôzne výsledky. Naneštastie v učebnici neboli výsledky a tak nevedeli, ktorý je správny.

Rozhodli sa, že nájdu študenta, ktorý sa to práve učí, a nechajú ho napísať program, ktorý bude dávať správne výsledky. Prečo program? Nuž, úloh je veľa a sami ešte nevedia, ktoré budú rátať.

Úloha: Na vstupe dostanete x-ové a y-ové súradnice vrcholov dvoch trojuholníkov, najprv prvého, potom druhého, dokopy teda 12 hodnôt. Úlohou je vypísať veľkosť obsahu prieniku a zjednotenia týchto dvoch trojuholníkov.

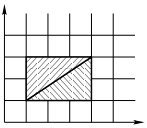
Pozor, špeciálnych prípadov môže byť naozaj veľa. Ak je nad vaše sily napísať program, ktorý by fungoval vždy, skúste aspoň napísať taký, ktorý bude fungovať, ak žiadne dve strany nie sú rovnobežné, aj za to budú nejaké body.

Zopár rád: Vzorové riešenie (takmer) žiadne špeciálne prípady neošetruje, všetko je len otázka správneho prístupu. Obsah trojuholníka, ktorého vrcholy majú súradnice $[x_1, y_1]$, $[x_2, y_2]$ a $[x_3, y_3]$, môžeme vypočítať nasledovne:

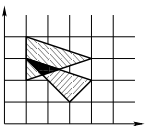
$$S = \tfrac{1}{2} \cdot |x_1y_2 + x_2y_3 + x_3y_1 - x_2y_1 - x_3y_2 - x_1y_3|$$

Príklad:

Vstup:	Výstup:
1 1 4 1 4 3 1 1 4 3 1 3	0.000 6.000



Vstup:	Výstup:
1 2 4 3 1 4 3 1 4 2 1 3	0.375 4.625



z2325. Z koláča dieru II

Po neúspechu prvého koláča mamička vyskúšala nový recept. Tentokrát bol koláč štvorcový a dokonca mala mamička kúpených aj veľa hrozienok a posypala ho nimi. Detičky sa úžasne tešili a akonáhle mamička odišla, zopakovali si odskúšanú zábavu. Krájali a krájali, ale urobili chybu a zabudli ho ochutnať. A veru to bol výborný koláč.

Prišla mamička a chcela ich potrestať. Vedela, že všetky deti majú rady hrozienka. Povedala im, že každé dieťa si môže zobrať najviac dva kúsky koláča. Navyše ak chce dva kúsky, musí si vybrať také dva, ktoré budú mať spoločnú hranu. Tu prišlo ku koláču prvé dieťa, počíta, oči si môže vyočiť, a predsa nevie, ktoré dva kúsky si má vybrať.

Úloha: Na vstupe máme celé číslo n – rozmer koláča. Ďalej nasleduje „mapa“ koláča $n \times n$. Tak ako v predchádzajúcom zadaní, bodky sú koláč, hviezdičky sú hrozienka, znak „`#`“ predstavuje zárezy. Úlohou je nájsť dva susedné kúsky koláča, ktoré budú mať spolu najviac hrozienok.

Kúsok koláča je každá časť koláča, ktorá drží pokope a je ohraničená rezmi, prípadne krajom plechu. (Dve políčka koláča držia pri sebe, ak je každé z nich bodka alebo hviezdička a sú buď hneď vedľa seba, alebo hneď pod sebou.)

Ak nájdeme dve políčka koláča (bodky alebo hviezdičky) z rôznych kúskov koláča, ktoré sú v tom istom riadku alebo stĺpci, a sú oddelené práve jedným políčkom (rezom), považujeme tieto dva kúsky koláča za susedné.

Váš program má nájsť najlepšiu dvojicu susedných kúskov. Vypíšte súradnice dvoch políčok, z každého kúsku jedno. Môžete vypísať ľubovolné políčko z oboch kúskov.

Príklad:

Vstup:

```
15
#*..#...*....#.
.#*#.*.*...***#
.#.#*.#####..*.
.###.#.....#.*.
..*.#..*.####..
..*.##...#**..#
..*..#...#####.
.*..#.###.**.##
#..#.*..##...#.
.##...*#*.#.#..
*.#.*.#....#.*.
.*##..#....#*..
.#**#..#####...
######.#*.#.#..
.*#*..#..#...#.
```

Výstup:

```
prvý kúsok: riadok 1, stĺpec 2
druhý kúsok: riadok 1, stĺpec 6
```

z2331. Zamiešané karty

V tejto úlohe sa budeme jednoducho hrať s kartami, presnejšie budeme ich jedným špeciálnym spôsobom miešať. Nech n je počet kariet v balíku a m je počet kôpok, ktoré počas miešania chceme vyrobiť. Miešanie prebieha nasledovne:

Vezmeme prvú kartu z vrchu balíku a položíme ju na stôl, čím vznikne prvá kôpka. Vedľa nej položíme ďalšiu kartu z vrchu balíku, čím vznikne druhá kôpka, a tak ďalej, až kým nemáme m kôpok. Nasledujúcu kartu z vrchu balíka položíme na vrch prvej kôpky, ďalšiu na vrch druhej, a takto dokola, až kým sa nám neminie úplne celý balík.

Akonáhle sa nám celý balík minul, prichádza druhá časť. Kôpku, na ktorú by sme práve mali položiť kartu (keby sme ešte nejakú mali), si celú zoberieme do ruky a stáva sa novým balíkom. Pokračujeme rozdávaním z neho, pričom prvá karta pôjde na nasledujúcu kôpku v poradí. Všimni si, že počet kôpok na stole sa práve zmenšil z m na $m - 1$.

Keď sa nám aj tento balík minie, zase spravíme to isté – zoberieme do ruky kôpku, ktorá práve mala dostať kartu, a karty z nej rozdáme na ostatné kôpky, pričom stále dodržiavame poradie existujúcich kôpok. Skončíme v okamihu, keď zo stola do ruky vezmeme poslednú kôpku, na ktorej sú úplne všetky karty.

Úloha: Na vstupe je číslo n – počet kariet v počiatočnom balíku a m – počet kôpok, ktoré na začiatku vyrobíme. Môžete predpokladať, že $1 < m \leq n \leq 30\,000$. Očíslujme si karty v balíku odhora nadol číslami 1 až n. Vašou úlohou je vypísať poradové číslo karty, ktorá skončí na spodku balíka po zamiešaní.

PRÍKLAD:

VSTUP:	VÝSTUP:
`10 3`	`3`

Karty na kôpke budeme vždy písať v poradí od spodnej po vrchnú. Po rozdaní balíka máme tri kôpky: $(1, 4, 7, 10)$, $(2, 5, 8)$, $(3, 6, 9)$. *Na rade bola kôpka s číslom 2, preto ju vezmeme do ruky a rozdáme, začínajúc od nasledujúcej kôpky (číslo 3). Dostaneme takéto dve kôpky:* $(1, 4, 7, 10, 5)$, $(3, 6, 9, 8, 2)$. *Všimnite si, že kôpku sme rozdávali od vrchu, teda ako prvú sme na stôl položili kartu číslo 8.*

Teraz by bola na rade nová kôpka číslo 1, preto putuje do ruky. Karty z nej postupne vyložíme na vrch druhej a dostávame výsledné poradie: $(3, 6, 9, 8, 2, 5, 10, 7, 4, 1)$.

z2332. Zrovnoprávnenie guličiek

Tak ako každé poobedie, aj dnes bolo na pieskovisku veľa detí – maličkých s mamičkami, zopár škôlkarov a školákov a skupinka matfyzákov, ktorí tu každodenne oddychujú po dni plnom cvičení (a niekedy aj prednášok). Taký matfyzácky oddych môže vyzerať rôzne, no dnes to bude niečo špeciálne – hlinené guličky. Po takejto hre vždy zostane vyhĺbený rad jamiek, niektoré sú plné guličiek, iné zase prázdne.

„To je diskriminácia!“ ozvalo sa zrazu odniekiaľ. „Veď ľaľa, jamiek je toľko, koľko guličiek. Každá guličká má právo na svoju vlastnú jamku!“ A hoci o právach hlinených guličiek niekoľko matfyzákov zapochybovalo, jeden sa vybral realizovať zrovnoprávnenie.

ÚLOHA: Na vstupe je číslo n – počet jamiek v rade. Nasleduje n čísel, určujúcich počty guličiek v jednotlivých jamkách. Súčet týchto počtov je rovný presne n.

Chceme guličky premiestniť tak, aby v každej jamke bola práve jedna. Povoleným ťahom je premiestnenie guličky o jednu jamku vpravo alebo vľavo. Nájdite najmenší počet ťahov potrebný na popresúvanie guličiek a *poriadne dokážte správnosť svojho programu.*

PRÍKLAD:

VSTUP:	VÝSTUP:
`5`	`4`
`3 0 0 2 0`	*(Jedna guličká pôjde z prvej jamky do druhej, jedna z prvej cez druhú do tretej, a jedna zo štvrtej do piatej jamky.)*

z2333. Zaujimavé a ešte zaujímavejšie predmety

Princípy počítačov? Programovanie (1)? Hm... pomyslel si Miško možno aj troška sklamane, keď videl, aké predmety ho tento semester čakajú, a vzdychol si. Ešteže sú na tom matfyze aj všelijaké iné, zaujímavejšie predmety (napríklad Riešenie okrajových úloh metódami presunu podmienok). A poďho všetky si zapísať, pokiaľ index stačí.

Onedlho ale vysvitlo, že štúdium určite nebude vyzerať podľa jeho predstáv. Keď Miško videl svoj rozvrh, zistil, že sa mu niektoré predmety kryjú, a teda nebude môcť chodiť na všetky. Ale ktorých sa má vzdať? Nakoniec sa rozhodol svoju zúfalú situáciu riešiť takto: pre každý predmet určil kladné číslo, označujúce jeho tzv. prioritu – ako veľmi by ho chcel navštevovať. A začal si robiť rozvrh – síce nie ideálny, ale aspoň najlepší možný v danej neľahkej situácii. Keď v nejaký čas prebiehalo viac predmetov, Miško si vybral ten s najvačšou prioritou a ostatných sa vzdal. Po vytvorení rozvrhu ostala skupina predmetov, ktorá bola zatlačená ich prioritnejšími kamarátmi-predmetmi. A ktorý predmet je ten najväčší smoliar?

Náš študijný režim zjednodušíme tak, že nebudeme rozlišovať dni a časy v nich, ale len jedno veľké poradie chlievikov v rozvrhu. Takže namiesto „v piatok o desiatej“ budeme hovoriť len „47“. Všetky chlieviky rozvrhu majú rovnakú veľkosť a všetky predmety trvajú rovnako dlho – práve dva takéto chlieviky. Čas predmetu bude teda charakterizovať jedno celé číslo k, ktoré vraví, že predmet zaberá k-ty a $(k+1)$-vý chlievik v rozvrhu. Navštevovať predmet znamená chodiť na oba jeho chlieviky; buď Miško môže ísť na oba, alebo na ten predmet nebude chodiť vôbec.

Úloha: Na vstupe sú v prvom riadku čísla n a p. Číslo n označuje počet chlievikov v rozvrhu, kedy môžu prebiehať predmety, p označuje počet predmetov, ktoré si Miško zapísal. Nasleduje p riadkov popisujúcich predmety. V týchto riadkoch sú po dve celé čísla. Prvé, p_i ($1 \leq p_i < n$), je číslo chlievika, v ktorom i-ty predmet začína. (Predmet teda zaberá chlieviky p_i a (p_i+1).) Druhé číslo nám hovorí prioritu, ktorú Miško tomuto predmetu určil. Žiadne dva predmety nemajú rovnakú prioritu.

Na výstupe vypíšte jedno číslo – poradové číslo predmetu, ktorý ma najväčšiu prioritu spomedzi tých, ktoré Miško nemôže navštevovať, pretože v ich čase beží nejaký predmet s vyššou prioritou. Pri písaní programu predpokladajte, že chlievikov aj predmetov bude najviac 10 000, priority sú kladné celé čísla neprevyšujúce 20 000.

Príklad:

Vstup:	Výstup:
`4 4`	`1`
`2 3`	*(V chlievikoch 1 a 2 pôjde na tretí predmet, v chlievikoch 3 a*
`1 1`	*4 pôjde na štvrtý predmet. Na zvyšné dva nemá čas. Spomedzi*
`1 47`	*tých dvoch, ktoré nestíha, prvý má vyššiu prioritu.)*
`3 2`	

z2334. Znížené ceny, zľavy...

... tak takto nejako to vyzerá v hypermarketoch a veľkoobchodoch hlavne po Vianociach. V jednom takomto obchode sa nachádza aj Jurko. Rozhodol sa, že si kúpi letné a zimné pneumatiky. Keďže Jurko má pri sebe s korún a nechce šetriť, tak by rád nakúpil za všetky peniaze, čo má pri sebe. Na výber je ale veľmi veľa typov a značiek (a z každej aj zimná aj letná súprava). Preto potrebuje pomôcť a pýta sa vás, či sa dajú kúpiť dve súpravy pneumatík, čo stoja dokopy presne toľko peňazí, koľko má pri sebe.

Úloha: Na vstupe je číslo s – suma, ktorú má Jurko pri sebe. Ďalej je na vstupe číslo n – počet súprav pneumatík a ďalej ich ceny. Vašou úlohou je zistiť, či existujú dve súpravy pneumatík, ktorých cena je v súčte rovná s. Ak potrebujete, môžete kúpiť aj viac rovnakých sád pneumatík.

Príklad:

Vstup:	Výstup:
`500 6`	`ÁNO`
`100 300 200 150 600 250`	*(Buď* 300 *a* 200 *alebo* 250 *a* 250.*)*

z2335. Zaútočme na chladničku

„Aaaa! Kde to som?“ poobzerám sa po okolí a matne rozoznávam celkom známe predmety. „No jasné! Som predsa doma!“ Ako sa ďalej pozerám po byte, tak zisťujem ďalšie veci. Nikto nie je doma. „Hmm, a čo budem obedovať?“ Tak som sa šiel pozrieť do chladničky, či mi náhodou v nej niečo nenechali. Na chladničke vidím nápis: „Ahoj. Na chladničke máš peniaze, skoč do obchodu a kúp si niečo na obed.“ Ale mne sa príliš do obchodu nechce. Navyše, keď pri nás akurát jeden búrajú... To by som musel ísť veľmi ďaleko. Aj tak sa mi nechce vyťahovať päty z domu. Tak teda, čo je v chladničke? Mrkva, petržlen, kaleráb, kaleráb, kečup, mlieko, kaleráb, pes, maslo, jogurt (tri týždne po záručnej dobe), olej, a ešte jeden kaleráb. Čo z toho uvariť? Na to bude treba nejaký recept...

Úloha: Vaša úloha bude teraz tak trošku netradičná. Úlohou bude napísať program, ktorý zo zadaného zoznamu dostupných surovín vypíše čo najchutnejší recept... Ehm. Teda, recepty jesť nebudeme, čo najchutnejšie má byť jedlo podľa nich uvarené.

Na vstupe bude číslo n, čo je počet druhov surovín. Ďalej nasleduje n riadkov, pričom v každom z nich je číslo m, reťazec j a reťazec s. Číslo m označuje množstvo, s surovinu a reťazec j označuje jednotky (napríklad „l“ je liter, „g“ gram, „ks“ kus, a pod.) To ale neznamená, že musíte použiť všetky suroviny, ktoré sú uvedené na vstupe. Snažte sa ale, aby ich váš program použil čo najviac, a aby recepty boli čo najlepšie...

Spolu s programom (a samozrejme aj popisom) nám pošlite tiež výstup vášho programu pre nasledujúce vstupy:

- $N = 3$, 1 l kečup, 2000 g zemiaky, 1 ks kaleráb
- $N = 8$, 1 l mlieko, 1000 g múka, 1 ks mrkva, 1 ks kaleráb, 100 g chren, 5 ks vajce, 1000 g chilli, 1 ks mrazený_hrášok
- $N = 20$, 5 ks kaleráb, 1000 g chren, 10 ks petržlen, 47 g majoránka, 100 ks vajce, 470 g múka, 500 g oravská_slanina, 5 ks feferón, 1000 g chilli, 3000 g zemiaky, 2 l smotana, 1 l mlieko, 200 g cibuľa, 500 g cesnak, 10 g čaj, 500 g horčica, 100 g zázvor, 20 ks uhorka, 1 l sójová_omáčka, 1 ks kura
- **Aktuálny obsah vašej chladničky :-)**

Príklad:

Vstup:

```
8
1000 g mrkva, 500 g strúhanka, 5 ks vajcia, 1 l mlieko,
20 g majoránka, 1000 g soľ, 100 g múka, 1 l olej
```

Výstup:

```
Budeme potrebovať: 1 kg mrkvy, 200 gramov strúhanky, 2 vajcia, trošku
mlieka, majoránku, múku a olej. Najskôr postrúhame mrkvu, pridáme múku
a vajcia. To cele rozmiešame, a pridáme do toho strúhanku (pozor, nie
všetku). Celé to ešte raz premiešame, pridáme majoránku, a ďalšiu
strúhanku, až kým nedostaneme dostatočne husté cesto. Z neho potom
spravíme guľôčky, ktoré obalíme v strúhanke a vypražíme na oleji.
```

(Váš výstup nemusí byť rovnaký, taktiež diakritiku nemusíte používať.)

z2341. Zirakzigilská knižnica

„Pozri na to, Balin... Červená kniha Západnej marky, Agnarov Denník, Introduction to Algorithms, Skriptá z analýzy a kopa ďalších kníh! Čo s nimi ja nešťastník spravím?“

I zamyslel sa Balin a riekol: „Nuž, mohli by sme postaviť novú knižnicu, keď už tu máš všade tak plno. Veď vieš kde, tam hore, na Zirakzigile.“ Dvalin sa zháčil: „Balin, ty si jedol lembas! Veď vieš, ako to tam vyzerá, samá tvrdá skala, sneh a ľad. A potrebovali by sme rýchlo veľkú knižnicu.“ Balin zobral kus pergamenu a začal kresliť: „Aha, tu máš voľné miesto, tu tiež...“ Po asi hodinke kreslenia prišli na to, že nájsť to správne miesto nebude žiadna sranda. Aby sa eliminovali stavebné náklady, musí byť budova knižnice štvorcová. Celá musí stáť na pevnom podklade. A keďže neprišli na to, kde by sa mohol takýto najväčší štvorec nájsť, zavolali si na pomoc programátorov.

Úloha: Na vstupe máte zadané čísla r a s udávajúce počet riadkov a stĺpcov mapy, a ďalej mapu z jednotiek (dá sa tam stavať) a núl (nedá sa tam stavať). Vašou úlohou bude nájsť veľkosť najväčšieho štvorca zloženého iba z jednotiek.

Príklad:

Vstup:

```
10 12
000000000000
011101110000
001111101100
011111111100
001011111010
011111111010
001111111110
011111111110
000111110000
000000000000
```

Výstup:

```
5
```

(Najväčší štvorec je od štvrtého po ôsmy riadok a od piateho po deviaty stĺpec.)

z2342. Z dražby

Ďaleko vo vesmíre existuje asteroid Symetroid. Na ňom je všetko symetrické. Aj jeho obyvatelia – Symetriťania – sú symetrickí. Keď sa narodí malé Symetríča, dostane pekné symetrické meno. Básnici rýmujú symetricky, takže sa im rýmujú aj začiatky aj konce veršov (alebo aj nie pri tejto chamradi „modernej“).

Dokonca aj čísla používajú iba tie „symetrické“ (po 9 ide 11 – žiadna 10 – a potom 22; u nás také čísla nazývame palindrómy). To ste mali vidieť jednu dražbu. Symetriťania sa tam prekrikovali: „1991,“ zvolal jeden, „2002,“ snažil sa prekričať symetrický šum druhý, „2112,“ nedal sa tretí. A Abba v škole nesedel, palindrómy nevedel, zvolal „2121“ a z dražby vyletel.

Úloha: Pomôžte Abbovi trochu si podražiť. Potreboval by však od vás program, ktorý mu vždy pošepne nasledujúce (t.j. najmenšie väčšie) symetrické číslo.

Príklad:

Vstup:	Výstup:
1991	2002
2002	2112

z2343. Zamilovaní KSPáci

V KSP sa v poslednej dobe rozšírila vlna dobra. Jednoducho všetci prestali hrešiť, začali páchať dobré skutky, požičiavať si navzájom cvičenia z matematickej analýzy... A vôbec, žijú veľmi poslušný život. A preto sa po určitej dobe najvyššia matematická bytosť, hľadiaca na zem z neba, rozhodla, že si všetkých KSPákov vezme k sebe. A tak si ich povolala k nebeskej bráne, ktorou musia KSPáci po jednom prejsť.

A tu vznikol problém. Medzi dobré vlastnosti KSPákov patrí aj to, že k sebe navzájom cítia náklonnosť. Pričom sa nemusí jednať o symetrický vzťah – to, že KSPák 1 má rád KSPáka 2, neznamená, že musí aj KSPák 2 mať rád KSPáka 1. Kedže láska je veľmi silný cit, platí, že žiadny KSPák by nikdy nevošiel do neba s tým, že pred nebeskou bránou ostane niekto, koho ľúbi. A to bola príčina, prečo sa strhla hádka. Dokonca aj nejaké facky medzi dobrými KSPákmi padli. Zišiel by sa zoznam, podľa ktorého by KSPáci vchádzali do neba tak, aby nevznikali nijaké nezhody. A zišiel by sa dosť rýchlo, pretože najvyššej matematickej bytosti čoskoro dôjde trpezlivosť a pošle rozhašterených KSPákov do pekla...

Úloha: Na vstupe máme číslo n – počet KSPákov a číslo m – počet vzťahov medzi nimi. Nasleduje m riadkov, na ktorých sú dve čísla x, y, popisujúce jeden vzťah. Čísla znamenajú, že KSPák x ma rád KSPáka y. Vypíšte n čísel – poradie KSPákov, v akom majú vchádzať do neba. V prípade, že sa takýto zoznam nedá spraviť, podajte o tom správu. Ak je viac možných riešení, stačí nájsť ľubovoľné jedno.

Príklad:

Vstup:	Výstup:
4 4	3 4 2 1
1 2	
2 4	*(Prvého môžeme poslať do neba iba KSPáka 3. Potom ale už*
4 3	*môže nasledovať KSPák 4, kedže trojka je už v nebi a nevznikne*
1 3	*nijaká hádka. Potom bude nasledovať KSPák 2 a nakoniec 1.)*
Vstup:	Výstup:
3 3	KSPáci skončia v pekle.
1 2	
2 3	*(Ako prvého nemôžeme poslať do neba ani jedného z nich, za-*
3 1	*každým by niekto protestoval.)*

z2344. Závislosť na hranolkách

Na matfyze sa neuveriteľným spôsobom rozprúdila konzumácia hranoliek. Akonáhle sa skončí prednáška, študenti sa hromadne vyberajú do bufetu, aby mohli ukojiť svoje

chúťky na toto, na celom svete obľúbené, jedlo. A tak, rovnako ako iní, sedia Miško a Matúš v prednáškovej sále a tešia sa, kedy konečne nastane tá chvíľa CHÁ. „Ja budem v bufete skôr než ty, postavím sa do radu pred teba a budem mať hranolky prvý!“ vraví Matúš Miškovi, ktorý sa ale nedá. „To ja budem skôr v bufete, poznám totiž lepšiu cestu!“. K nim ale prichádza Milan a tvrdí, že on je ten, ktorý bude mať prvý hranolky, pretože pozná najkratšiu cestu. A to sa ešte nezapojil Stano... Ako to ale býva, kde sa veľa ľudí háda, tam nemá pravdu ani jeden. Chlapci totiž nevedia, že pravda je taká, že všetky cesty, ktoré spomínajú, sú rovnako dlhé. Takýchto ciest existuje veľmi veľa, dá sa povedať, že čo študent, to iná cesta. A nás by zaujímalo, koľko takýchto ciest existuje.

Matfyz si predstavíme ako štvorcovú sieť, prednáškovú miestosť umiestníme do bodu $[0,0]$ a bufet do bodu $[r,s]$. Vašou úlohou bude napísať program, ktorý vypočíta počet rôznych najkratších ciest z prednášky do bufetu. Pod krokom rozumieme posun na nejaký zo štyroch susedných mrežových bodov. Je jasné, že optimálne cesty budú pozostávať len z krokov dvoch smerov, nazvime ich hore a vpravo. Ak si cestu budeme interpretovať ako postupnosť znakov H a V (H – hore, V – vpravo), tak dve cesty sú rôzne, ak majú rôzny zápis postupnosti krokov.

Úloha: Na vstupe sú dve čísla – r a s, označujúce súradnice bufetu. Vašou úlohou je zistiť počet rôznych najkratších ciest medzi bodmi $[0,0]$ a $[r,s]$.

Príklad:

Vstup:	Výstup:
`4 3`	`35`

z2345. Zaplnená chladnička

„A keď sa už sťahujeme, sprav zoznam všetkého, čo je v chladničke,“ ozvalo sa spoza zatvárajúcich sa dverí. Ferko sa pozrel na obrovskú a zároveň preplnenú chladničku, spokojne vrčiacu za rohom. Začal teda pekne systematicky, kus po kuse vyberať obsah chladničky a značiť si obsah. No zapisovať na papier ho čoskoro omrzelo. A poprosil vás o pomoc.

Úloha: Na vstupe sú čísla n a k. Nasledujú názvy n vecí, ktoré Ferko vytiahol z chladničky, v každom riadku práve jeden. Názov je reťazec dĺžky najviac k. Výstupom programu by pre každý názov mal byť počet jeho výskytov na vstupe.

Príklad:

Vstup:	Výstup:

`7 17`
`jablko`
`hruska`
`diskochrastie`
`hruska`
`jablko`
`iglu`
`hruska`
	`jablko 2x`
	`hruska 3x`
	`iglu 1x`
	`diskochrastie 1x`

Návody k riešeniam

V tejto časti sú uvedené návody k väčšine úloh. K vybraným úlohám je viacero návodov, ktoré vedú k rôzne efektívnym riešeniam, je to spravidla vtedy, keď sa nám zdalo, že najlepšie riešenie obsahuje netriviálny trik, alebo vtedy, keď postupnosť riešení je sama o sebe zaujímavá.

1611. Pre jednoduchosť predpokladajme, že $n = 2^k$. Použijeme intervalový strom. Koreň tohto stromu zodpovedá celému Kubomíru, teda intervalu $\langle 1, 2^k \rangle$. Každý vrchol, ktorý zodpovedá intervalu dlhšiemu ako 1, má dvoch synov; každý z nich zodpovedá polovici otcovho intervalu. Listy teda zodpovedajú jednotlivým políčkam.

V každom vrchole si budeme pamätať dva údaje: minimum z výšok stĺpcov v zodpovedajúcom intervale (teda hodnotu hovoriacu, koľkokrát je kubármi pokrytý celý interval) a počet stĺpcov v danom intervale, ktoré majú nenulovú výšku. (Ak je minimum nenulové, tento počet stĺpcov je samozrejme rovný dĺžke dotyčného intervalu.) Pomocou tejto dátovej štruktúry vieme každý príkaz zo vstupu spracovať v čase $O(\log n)$.

1612. Jednoducho naprogramovateľné polynomiálne riešenie môžeme založiť na myšlienke, že v optimálnej polohe sa hľadaný kruh dotýka aspoň troch objektov (t.j. bodov, kde sú kvety, alebo strán obdĺžnika tvoriaceho záhradku). Inak ho totiž vieme posunúť a prípadne aj zväčšiť. Dostávame takto riešenie v $O(n^4)$: Stačí pre každú trojicu objektov nájsť kruh, ktorý sa ich dotýka, overiť, či neobsahuje iné objekty, a vybrať najväčší z vyhovujúcich.

Lepšie riešenie je založené na pozorovaní, že rovinu môžeme rozdeliť na oblasti bodov podľa toho, ku ktorému z kvetov majú najbližšie. Takéto rozdelenie sa nazýva *Voronoiov diagram*. Ľahko nahliadneme, že ako stred maximálnej prázdnej kružnice, ktorá sa dotýka troch kvetov, stačí skúšať vrcholy Voronoiovho diagramu – teda body, ktoré súčasne ležia na hranici troch rôznych oblastí. Teraz už ľahko naprogramujeme riešenie v čase $O(n^3)$: pre každý kvet zostrojíme jeho oblasť ako prienik $n-1$ polrovín, tým dostaneme množinu $O(n)$ vrcholov Voronoiovho diagramu, vyskúšame každý z nich a hrubou silou doriešime prípady, kedy sa optimálny kruh dotýka obvodu záhradky.

Existujú aj efektívnejšie spôsoby, ako nájsť Voronoiov diagram, optimálny má časovú zložitosť $O(n \log n)$. V takejto časovej zložitosti sa dá riešiť aj naša úloha, implementácia je však veľmi komplikovaná. (Uvedomte si napríklad, že takéto riešenie potrebuje efektívnejšie riešiť aj kruhy dotýkajúce sa jednej či dvoch strán obdĺžnika.)

1613. Najskôr si zistíme dĺžku najkratšej cesty medzi každými dvoma mestami. To vieme pomocou Floydovho-Warshallovho algoritmu v čase $O(n^3)$. Potom usporiadame mítingy podľa času konania a pomocou dynamického programovania nájdeme v čase $O(m^2)$ najdlhšiu postupnosť mítingov, ktoré sa dajú všetky stihnúť: pre každý míting si spočítame, koľko sa ich najviac stíha, ak tento je posledný navštívený.

1614. Použijeme dynamické programovanie cez všetky podreťazce: Nech $D[i,j]$ je najmenší počet písmen, ktoré potrebujeme, aby sme z podreťazca $w_i \ldots w_j$ spravili palindróm (chceme vypočítať $D[1,n]$). Ak je prvé a posledné písmeno rovnaké ($w_i = w_j$), tak $D[i,j] = D[i+1, j-1]$. Ak nie, musíme buď na začiatok alebo na koniec pridať jedno písmeno a $D[i,j] = 1 + \min(D[i,j-1], D[i+1,j])$. Z hodnôt $D[i,j]$ následne ľahko skonštruujeme jedno riešenie.

1615. Na pole p zo zadania sa môžeme pozerať ako na mapu bludiska, v ktorom sú voľné políčka tie, kde sú v poli p nuly. Pri vykonávaní programu zo zadania háda víla Amálka cestu týmto bludiskom zo súradníc $[0,0]$ na súradnice $[n-1, n-1]$ a my overujeme, či nás neklame. Program teda odpovie `ÁNO` práve vtedy, ak aspoň jedna cesta existuje.

Ekvivalentný program bez víly Amálky ľahko napíšeme pomocou prehľadávania do šírky alebo prehľadávania do hĺbky.

1621. Pre každý kameň spočítame najkratší čas, za ktorý sa dá od neho dostať do dediny, ak ideme v smere hodinových ručičiek. Toto vieme spočítať v čase $O(n)$, stačí začať výpočet pri kameni, odkiaľ je do dediny najbližšie. Následne spočítame to isté pre opačný smer. Taktiež pre oba smery a každý kameň spočítame najkratší čas z dediny k nemu.

V ďalšom kroku pre každý smer prechádzania nájdeme pre každý kameň x najvzdialenejši kameň, ktorý ešte vieme navštíviť v deň, ktorý začneme tým, že optimálnou cestou prídeme k x. Toto vieme tiež spraviť v lineárnom čase, keďže optimálny koniec pre nasledujúci kameň $x+1$ je aspoň tak ďaleko ako optimálny koniec pre kameň x. Podobne, pre každý smer prechádzania nájdeme pre každý kameň x najvzdialenejší kameň taký, že vieme za jeden deň z dediny prísť k nemu, od neho prejsť až ku kameňu x a odtiaľ sa vrátiť naspäť do dediny.

Využitím všetkých vypočítaných údajov vieme v lineárnom čase pre každý kameň x nájsť najväčšie a_x také, že vieme v jeden deň navštíviť kamene $x, x+1, \ldots, x+a_x-1$ (počítané cyklicky), pričom je nám jedno, v ktorom smere ich navštívime (t.j., či budeme začínať v x, alebo končiť v x). Teraz určite existuje optimálny rozvrh nasledujúceho tvaru: prvý deň aktivujeme kamene od x_1 po $x_1+a_{x_1}-1$, druhý deň kamene od $x_2=x_1+a_{x_1}$ po $x_2+a_{x_2}-1$, atď. až kým nepokryjeme celý kruh.

Riešenie v čase $O(n^2)$: vyskúšame všetky možnosti pre x_1 a zakaždým simulujeme.

Riešenie v čase $O(n \log n)$: Zovšeobecníme naše a_x nasledovne: a_x^i bude dĺžka najdlhšieho úseku začínajúceho na pozícii x, ktorý vieme aktivovať za 2^i dní. Pre $2^0=1$ deň už túto hodnotu vieme: $a_x^0=a_x$. Každú konkrétnu hodnotu a_x^{i+1} vieme v konštantnom čase vypočítať z hodnôt pre polovičný počet ciest: Za prvých 2^i dní vieme aktivovať úsek dĺžky $s=a_x^i$. Za nasledujúcich 2^i dní teda potrebujeme aktivovať čo najdlhší úsek začínajúci na pozícii $x+s$, no a jeho dĺžka je a_{x+s}^i.

Tieto hodnoty počítame dovtedy, kým niektorá hodnota a_x^i nedosiahne aspoň n. To nastane najneskôr pre $i=\lceil \log_2 n \rceil$. Teraz vieme, že hľadaný počet dní leží v intervale $(2^{i-1}, 2^i]$. Tam ho nájdeme binárnym vyhľadávaním. (Invariant počas hľadania je taký, že keď hľadáme v intervale $(p,q]$, tak navyše pre každý kameň vieme dĺžku úseku, ktorý od neho vieme pokryť za p dní. Krok hľadania vyzerá tak, že v $O(n)$ spočítame dĺžky úsekov pre $d=(p+q)/2$ dní.)

1622. Budeme hľadať, kde môže ležať stred hľadanej kružnice. Pre každú stranu mnohouholníka si zostrojíme útvar, v ktorom ležať nesmie. Útvar, nazvime ho zakázaný ovál, bude podobného tvaru ako štadión s bežeckou dráhou. Potrebujeme zistiť, či vo vnútri mnohouholníka existuje bod, ktorý neleží vo vnútri žiadneho zo zakázaných oválov.

Riešenie v čase $O(n^2 \log n)$: Ak existuje hľadaný bod, určite existuje taký, ktorý leží na hranici jedného zo zakázaných oválov. Pre každý zakázaný ovál preto spravíme nasledujúci výpočet: zostrojíme priesečníky jeho hranice s obvodom mnohouholníka, aby sme zistili, ktoré časti sú vonku a ktoré vnútri n-uholníka. Následne zostrojíme priesečníky s hranicami ostatných oválov, aby sme zistili, ktoré časti sú zakázané.

Priesečníky rozdelia hranicu nášho oválu na niekoľko úsekov. Ak sa na hranici nachádza úsek (bod), ktorý je vnútri mnohouholníka, ale neleží vo vnútri žiadneho zakázaného oválu, máme náš hľadaný bod. Toto vieme ľahko overiť jedným prechodom v lineárnom čase, ak si nájdené priesečníky usporiadame po obvode oválu.

Existuje aj riešenie v čase $O(n^2)$, založené na tom, že postupne zväčšujeme polomer kruhu r od 0 až po zadanú hodnotu, pričom si udržujeme obvod zjednotenia zakázaných oválov. (Zväčšovanie je potrebné robiť po skokoch, teda vždy vypočítať, kedy sa najbližšie zmení počet vrcholov na obvode, zmeniť r na príslušnú hodnotu a odsimulovať príslušnú zmenu.)

Pravdepodobne existuje aj ešte efektívnejšie riešenie, na implementáciu by však bolo príliš komplikované.

1623. Predstavme si bipartitný graf, kde vrcholy sú páni a dámy a hrany reprezentujú jednotlivé páry. Úlohou je vlastne ofarbiť všetky jeho hrany čo najmenším počtom farieb tak, aby hrany, ktoré sa stretajú v jednom vrchole, mali vždy rôzne farby.

Určite potrebujeme aspoň toľko farieb, ako je maximálny stupeň vrcholu. Takéto ofarbenie vždy existuje. Jedno zostrojíme napríklad tak, že začneme s prázdnym grafom a po jednej pridávame hrany. Ak pri pridávaní hrany uv máme v u a v tú istú nepoužitú farbu, použijeme ju. Ak nie, nech a je farba voľná v u a b farba voľná vo v. Z vrcholu v zostrojíme cestu idúcu striedavo hranou farby a a b, kým to ide. Všetkým hranám tejto cesty zmeníme farbu z a na b a naopak. Tým sme aj vo vrchole v uvoľnili farbu a, ktorú následne použijeme na novú hranu uv.

1624. Algoritmus, ktorý by sa pýtal na všetky bity, potrebuje rádovo $n \log n$ otázok, vzorovému riešeniu ich ale stačí $O(n)$. Číslo budeme zisťovať postupne po bitoch. Vieme si spočítať, koľko jednotiek a koľko núl má byť na poslednej pozícii, keby prišli všetci. Opýtame sa na posledný bit všetkých čísel. Z odpovedí vieme povedať, či na poslednom mieste chýba 0 alebo 1. Tým vieme polovicu študentov vylúčiť – na tých sa už pýtať nikdy nebudeme. Takto budeme pokračovať, kým neurčíme všetky bity. Položíme pri tom $n + n/2 + n/4 + \cdots < 2n$ otázok.

1625. Najskôr necháme vílu Amálku, nech nám pre každé mesto na papyruse uhádne jeho číslo na mape. Následne potrebujeme overiť, či nám poradila správne – ku každej ceste v papyruse nájsť zodpovedajúcu cestu v mape. V mape však táto cesta môže byť rozdelená na viac úsekov, preto aj pri overovaní ciest potrebujeme využiť pomoc víly Amálky, ktorá nám bude pre každú cestu na papyruse hádať čísla miest na mape, cez ktoré teraz cesta vedie. Treba si dať pozor na to, že každé mesto na mape môžeme takto použiť len raz.

Technický detail: Víle Amálke môžeme dať na výber vždy iba z konštantne veľa možností. Keď teda potrebujeme, aby uhádla správne celé číslo z intervalu $[1, n]$, nestačí nám na to jedna otázka. Optimálne riešenie ich potrebuje $O(\log n)$ – napríklad necháme Amálku postupne uhádnuť jednotlivé bity čísla.

1631. *Riešenie nadväzuje na text „Dijkstrov algoritmus“ na str. 248.*
Zostrojíme orientovaný graf, kde dĺžky hrán budú predstavovať stúpanie medzi dotyčnými vrcholmi (pri rovine/klesaní je dĺžka hrany nulová). V tomto grafe nájdeme dĺžky najkratších ciest zo štartovacieho bodu do každého vrcholu.

1632. Jednoduché riešenie v čase $O(n^3)$ využíva fakt, že stačí skúšať iba tie kruhy, ktoré majú aspoň 2 hrozienka na obvode. Takže každou dvojicou bodov preložíme všetky možné kruhy polomeru r (také sú najviac dva) a spočítame hrozienka v nich.

Rýchlejšie riešenie: Zvoľme si jeden bod a uvažujme všetky kružnice, ktoré ním prechádzajú. Ich stredy sa nachádzajú na kružnici polomeru r, ktorá ma stred vo zvolenom bode. Predstavme si, že začneme stred kružnice otáčať okolo zvoleného bodu. Pre každé hrozienko spočítame uhol, kedy sa dostane do otáčaného kruhu a tiež uhol, kedy ho opustí. Tieto uhly si usporiadame a v získanom poradí ich postupne všetky spracujeme. Tým odsimulujeme otáčanie kružnice okolo práve skúšaného bodu a nájdeme to natočenie, pri ktorom je vnútri najviac hrozienok.

Otáčanie kružnice okolo zvoleného bodu vieme odsimulovať v čase $O(n \log n)$. Keďže postupne vyskúšame všetky body, dostávame celkovú časovú zložitosť $O(n^2 \log n)$.

Samozrejme, netreba zabúdať na okrajové prípady (kruh na kraji/v rohu).

1633. Úlohou je nájsť tzv. *minimálnu triangluláciu* daného konvexného mnohouholníka. Pre ľubovoľné $i < j$ označme $\ell_{i,j}$ vzdialenosť vrcholov i a j, ďalej $M_{i,j}$ nech je mnohouholník, ktorého vrcholy sú kolíky od i po j, a $D[i, j]$ nech je minimálna dĺžka špagátu potrebná na triangluláciu $M_{i,j}$.

Ako vypočítame $D[i,j]$? Triviálne $D[i,i+1] = D[i,i+2] = 0$ pre každé i. Nech teda $j \geq i+3$. Strana ij sa musí nachádzať v nejakom trojuholníku. Označme jeho tretí vrchol k. Keď natiahneme špagát medzi bodmi i a k a tiež medzi bodmi j a k, rozdelíme $M_{i,j}$ na tri časti: náš trojuholník ijk a dva menšie mnohouholníky $M_{i,k}$ a $M_{k,j}$.

Optimálnu trianguláciu nájdeme tak, že vyskúšame všetky možnosti pre k a z nich vyberieme najlacnejšiu. Dostávame vzťah: $D[i,j] = \min_{i<k<j}\{D[i,k]+D[k,j]+\ell_{i,k}+\ell_{k,j}\}$. Na výpočet hodnôt $D[i,j]$ môžeme použiť dynamické programovanie, čím dostávame riešenie s časovou zložitosťou $O(n^3)$.

Poznámka: Presvedčte sa, že nie vždy je výhodné zvoliť „chamtivo“ najkratšiu uhlopriečku, ktorá ešte nič nekrižuje.

1634. Nech $S[x,y]$ je počet strán kníh x až y. Dynamickým programovaním budeme počítať hodnoty $T[i,j]$ – najkratší možný čas, za aký prvých i pisárov prepíše prvých j kníh (hľadaný čas je $T[n,k]$). Ak q je počet kníh, ktoré prepíše prvých $i-1$ pisárov, tak $T[i,j] = \max(T[i-1,q], S[q+1,j])$. Na vybratie najlepšieho q netreba skúšať všetky možnosti od 1 po j: Maximum bude najmenšie vtedy, keď budú hodnoty $T[i-1,q]$ a $S[q+1,j]$ čo najbližšie a také q vieme nájsť binárnym vyhľadávaním. Každé políčko takto vieme vyplniť v čase $O(\log n)$, celkovo teda máme algoritmus pracujúci v čase $O(kn \log n)$.

Lepšie riešenie: V skutočnosti nemusíme vyhľadávať binárne: Vypĺňajme tabuľku T po riadkoch. Uvedomme si, že ak q' bola najlepšia hodnota pre $T[i,j-1]$ a q je najlepšia hodnota pre $T[i,j]$, tak $q' \leq q$. Hodnoty q teda stačí hľadať sekvenčne: Vždy začneme od predošlého optima a zväčšujeme q, kým sa $\max(T[i-1,q], S[q+1,j])$ zmenšuje. Niektoré políčka vyplníme rýchlo, niektoré nám budú trvať dlhšie, avšak q sa vždy iba zväčšuje a z 1 môže narásť na najviac n, takže vyplniť jeden riadok vieme dokopy v čase $O(n)$. Celková časová zložitosť je $O(kn)$.

Iné riešenie: Pre ľubovoľný konkrétny čas t vieme ľahko v čase $O(n)$ overiť, či to pisári môžu alebo nemôžu v čase t stihnúť – každému pisárovi dáme najviac kníh, ako sa len dá, bez toho, aby prekročil čas t. Najlepší čas vieme potom zistiť binárnym vyhľadávaním. Toto riešenie sa dá implementovať s časovou zložitosťou $O(n \log t_{\min})$.

1635. Pre každý z k kusov koláča necháme vílu Amálku uhádnuť súradnice jeho ľavého horného rohu. Následne overíme, či každý kus koláča leží celý na plechu a či sa žiadne dva neprekrývajú. To vieme triviálne spraviť v čase $O(k^2)$.

Existujú aj riešenia s časovou zložitosťou $O(k \log k)$. Jedna možnosť je usporiadať obdĺžniky podľa zvislej súradnice, na ktorej začínajú a následne ich po jednom pridávať, pričom si vo vyvažovanom strome pamätáme vodorovné úsečky, ktoré momentálne tvoria spodný okraj už zaplnenej časti plechu.

1641. Vstup načítavame a spracúvame po riadkoch. Pamätáme si predchádzajúci riadok a ofarbenie pevniny v ňom: políčka, o ktorých už vieme, že patria tomu istému ostrovu, majú rovnakú farbu. Súvislé úseky pevniny v aktuálnom riadku ofarbíme inými farbami.

Vytvoríme graf, ktorého vrcholmi budú použité farby (z oboch riadkov). Pre každé políčko pevniny z aktuálneho riadku, ktoré susedí s políčkom pevniny z predchádzajúceho riadku, pridáme do grafu hranu medzi ich farbami, pretože patria tomu istému ostrovu. Prehľadávaním tohto grafu nájdeme jeho komponenty súvislosti. Farby v komponente prislúchajú tomu istému ostrovu, preto ich všetky nahradíme jednou farbou.

Keď nejaký ostrov nepokračuje do ďalšieho riadku, započítame ho do výsledného počtu. Pamäťová zložitosť je $O(s)$, pretože si pamätáme iba dva riadky mapy a vrcholov aj hrán grafu je tiež len $O(s)$. Tento algoritmus má časovú zložitosť $O(rs)$.

1642. Nie je nám známe riešenie s polynomiálnou časovou zložitosťou.

Jedným možným riešením je backtracking. Nájdeme najzápadnejší objekt P. Obdĺžnik, ktorým tento objekt pokryjeme, posunieme na východ až tak, že P bude na jeho západnom okraji. Otázna zostáva pozícia tohto obdĺžnika v severojužnom smere; závisí od nej, ktoré

ďalšie objekty budú pokryté. Vyskúšame všetky možné polohy, ktoré pokrývajú rôzne objekty a rekurzívne hľadáme riešenie pre zvyšné objekty.

Časovú zložitosť predchádzajúceho riešenia vieme zlepšiť z $O(n!)$ na $O(n2^n)$ pomocou dynamického programovania: pre každú podmnožinu zadaných bodov si spočítame a zapamätáme najmenší počet obdĺžnikov potrebný na jej pokrytie.

1643. *Riešenie nadväzuje na texty „Medián postupnosti“ na str. 243 a „Union-find“ na str. 249.*
Každá najdrahšia kostra je zároveň aj najhrubšou kostrou, úlohu preto vieme v grafe s m hranami vyriešiť napríklad použitím Kruskalovho algoritmu v čase $O(m \log m)$. Najhrubších kostier je však vo všeobecnosti viac ako najdrahších, čo vieme využiť pri návrhu efektívnejšieho riešenia.

Hľadáme vlastne najmenšiu hrúbku h takú, že graf tvorený len hranami hrúbky h a väčšej je ešte stále súvislý. Na jej nájdenie použijeme binárne vyhľadávanie: Nájdeme medián váh hrán a rozdelíme ich na tenšie a hrubšie. Ak je graf obsahujúci hrubšie hrany súvislý, môžeme všetky tenšie hrany zahodiť a v hľadaní pokračovať len s hrubšími. Naopak, ak graf obsahujúci hrubšie hrany súvislý nie je, všetky hrubšie hrany si v grafe už navždy ponecháme a v hľadaní pokračujeme len s tenšími.

Na rýchle overovanie súvislosti grafu použijeme algoritmus *union-find*. Každý komponent bude reprezentovaný stromom, vrcholy v ňom ukazujú na svojho nadriadeného. Koreň stromu nazveme reprezentantom komponentu.

Keď chceme spracovať nejakú množinu hrán, vytvoríme si „supergraf“, ktorého vrcholmi budú len reprezentanti komponentov pôvodného grafu. Namiesto pridania hrany medzi dvoma vrcholmi pôvodného grafu pridáme do supergrafu hranu medzi reprezentantmi ich komponentov.

Súvislosť supergrafu overíme napríklad prehľadávaním do hĺbky. Ak sme dostali súvislý supergraf, spracúvanú množinu hrán zahodíme. Ak nie, pridáme ich do našej dátovej štruktúry, a to tak, že v každom komponente supergrafu si zvolíme jeden vrchol a všetkým ostatným vrcholom dotyčného komponentu ho nastavíme ako ich reprezentanta. Takto vlastne spojíme do jedného tie komponenty pôvodného grafu, ktoré nám spojili nové hrany.

Akú má toto riešenie časovú zložitosť? Nájsť medián a rozdeliť hrany na tenšie a hrubšie vieme v čase lineárnom od ich počtu. V x-tej iterácii binárneho vyhľadávania skúšame pridať rádovo $m/2^x$ hrán a hĺbka stromov, ktorými v tej chvíli reprezentujeme komponenty, je najviac x, preto zostrojenie supergrafu trvá $O(xm/2^x)$; v nanajvýš rovnakom čase ho vieme aj prehľadať a upraviť stromy reprezentujúce komponenty. Keďže $\sum_{x \geq 1} xm/2^x = 2m$, je celková časová zložitosť tohto riešenia $O(m)$.

1644. *Riešenie nadväzuje na text „Prehľadávanie do hĺbky“ na str. 246.*
Máme zadané dva orientované acyklické grafy s rovnakými vrcholmi. Hrany jedného tvoria vleky a hrany druhého zjazdovky. V grafe tvorenom vlekmi chceme pre každú dvojicu vrcholov zistiť dĺžku najkratšej cesty medzi nimi. Naopak, v grafe tvorenom zjazdovkami chceme pre každú dvojicu vrcholov nájsť dĺžku najdlhšej cesty medzi nimi. Pre ľubovoľný konkrétny vrchol v vieme nájsť najdlhšie/najkratšie cesty do všetkých ostatných počas jedného prehľadávania do hĺbky z v. Toto musíme spraviť samostatne pre každý vrchol, a teda celková časová zložitosť algoritmu bude $O(n \cdot (k + m + n))$.

1645. Pre konkrétne dva vrcholy x, y vieme pomocou víly Amálky ľahko overiť, či sa dá dostať z x do y – stačí nechať vílu Amálku hádať čísla vrcholov, cez ktoré máme ísť, a overovať, že sa to dá. Ak ani po n krokoch ešte nie sme v y, môžeme to vzdať. Keďže na uhádnutie jedného čísla potrebujeme $O(\log n)$ otázok, má toto riešenie časovú zložitosť $O(n \log n)$, pamäťová je konštantná.

Teraz si už len stačí uvedomiť, že nepotrebujeme takto otestovať všetky dvojice vrcholov. Stačí napríklad overiť, či sa dá z vrcholu 1 dostať do každého iného vrcholu a či sa dá

z každého vrcholu dostať do 1 (potom sa už zrejme dá cez 1 dostať odvšadiaľ všade). S vílou Amálkou teda existuje riešenie s konštantným počtom premenných a časovou zložitosťou $O(n^2 \log n)$.

Zaujímavé je, že existuje riešenie bez víly Amálky, ktoré si (za cenu veľkej časovej zložitosti) vystačí s pamäťou veľkosti $O(\log n)$. Stačí si spraviť rekurzívnu funkciu $over(x, y, z)$, ktorá overí, či sa dá z vrcholu x dostať do vrcholu y na nanajvýš z krokov. Ak $z = 1$, len overíme, či sú x a y rovnaké alebo spojené hranou. Pre $z > 1$ takáto cesta existuje práve vtedy, keď pre nejaké a súčasne platí $over(x, a, \lfloor z/2 \rfloor)$ aj $over(a, y, \lceil z/2 \rceil)$. Postupne teda vyskúšame všetky a, a pre každé z nich postupne spravíme obe rekurzívne volania. Samotná procedúra si vystačí s konštantným počtom premenných a hĺbka rekurzie nikdy neprekročí $O(\log n)$.

Pre neorientované grafy to ide dokonca ešte lepšie. V roku 2004 publikoval Omer Reingold algoritmus, ktorý si aj bez víly Amálky vystačí s konštantnou pamäťou. (Presnejšie, jeho algoritmus používa $O(\log n)$ bitov pamäte. Do takto veľkej pamäte sa napríklad naraz zmestí len konštantne veľa čísel vrcholov.)

z1611. *Riešenie je uvedené v texte „Union-find“ na str. 249.*

z1612. Tento problém môžeme sformulovať v teórii grafov: Zuzkine kamarátky reprezentujeme vrcholmi grafu a nepriateľstvá hranami. Tomuto grafu chceme všetky vrcholy ofarbiť najviac troma farbami tak, aby susedné vrcholy nemali nikdy tú istú farbu. Každá farba zodpovedá jednému zo stolov.

Táto úloha sa nazýva 3-ofarbiteľnosť grafu. Nie je známy žiaden algoritmus, ktorý by ju vyriešil v polynomiálnom čase. Väčšina odborníkov je presvedčená, že taký algoritmus ani neexistuje.

Kristínkin algoritmus nefunguje. K príkladu v zadaní (Lucii, Danke a Janke) si ešte zoberme tri ďalšie dievčatá: Mirku, Zuzku a Hanku. Každé dve z nich sú pohádané a každá sa háda s inými dvoma z prvých troch. Vhodné rozsadenie existuje: Luciu, Danku a Janku dať každú k inému stolu a potom ku každej posadiť tú z druhej trojice, ktorá sa s ňou nehádа. Ak ale použijeme Kristínkin algoritmus, Danka a Janka skončia pri tom istom stole a druhú trojicu sa už usadiť nepodarí.[5]

Ako teda úlohu riešiť? Jedna možnosť je vyskúšať všetkých 3^n možných ofarbení. Oveľa lepšie je však skúšať vrcholy ofarbovať postupne tak, aby ofarbovaný vrchol vždy susedil už s nejakými ofarbenými – vtedy nám pri každom rozhodovaní ostanú najviac 2 možnosti a algoritmus pobeží v čase $O(2^n(n + m))$.

Iný spôsob je vybrať nejakú podmnožinu (tých je 2^n), ofarbiť ju jednou farbou a zistiť, či sa zvyšné vrcholy dajú ofarbiť zvyšnými dvoma farbami (to už vieme v lineárnom čase). Tento spôsob sa dá ešte vylepšiť, pretože stačí skúšať len tie množiny vrcholov, do ktorých sa už žiaden vrchol nedá pridať, teda tzv. maximálne nezávislé množiny. Dá sa ukázať, že takých je najviac $3^{n/3} \leq 1.443^n$. Vhodnou implementáciou vieme dosiahnuť algoritmus s časovou zložitosťou $O(1.443^n n^2)$.

z1613. Kotúče rovnakej veľkosti môžeme považovať za jeden, ktorého pohyb trvá príslušný počet krokov. Predpokladajme teda, že kotúče sú navzájom rôznej veľkosti.

Úloha sa dá jednoducho riešiť pomocou rekurzie: Chceme presunúť n kotúčov z tyče A na tyč B (pomocou tyče C). Všimnime si najväčší, najspodnejší kotúč – ak ho chceme presunúť, musíme nutne najskôr presunúť $n - 1$ menších kotúčov z A na tyč C (pomocou tyče B; rekurzívne rovnakým spôsobom). Potom môžeme presunúť najväčší kotúč z A na B a následne z C vrátime $n - 1$ kotúčov na B (pomocou A; rovnakým spôsobom).

[5] Technický detail: Ešte potrebujeme zabezpečiť, aby Lucia mala naozaj najviac nepriateliek, Danka druhá najviac, atď. To dosiahneme pridaním vhodného počtu ďalších dievčat, ktoré sú rozhádané len s jednou z našich šiestich.

Z tohto riešenia sa dá ľahko odvodiť, že (ak sú všetky kotúče rôznej veľkosti) stačí nám $2^n - 1$ krokov a toľko naozaj aj treba.

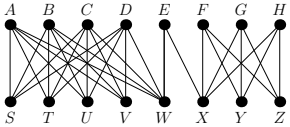
Existuje aj nerekurzívne riešenie: stačí striedavo aplikovať nasledujúce dva kroky: 1. Presunieme najmenší kotúč cyklicky doprava (z 1 na 2, z 2 na 3, alebo z 3 na 1), 2. Presunieme iný kotúč (je vždy iba jedna možnosť aký a kam).

z1614. *Riešenie nadväzuje na text „Prehľadávanie do hĺbky“ na str. 246.*
Budeme prehľadávať graf železníc do hĺbky, pričom ku každej stanici si pamätáme, odkiaľ sme do nej prvýkrát prišli. V okamihu, keď na niektorú stanicu s prídeme druhýkrát, sme našli okruh. Zostrojiť ho vieme jednoducho tak, že sa budeme vracať po zapamätaných hranách až kým nenarazíme opäť na stanicu s.

z1615. Riešenie v čase $O(m)$: Prehľadávame priestor stavov. Zostrojíme graf, v ktorom vrcholy majú čísla 0 až m a zodpovedajú možným obsahom mlieka v hrnci a orientované hrany zodpovedajú povolenému prilievaniu a odoberaniu mlieka. Prehľadávaním tohto grafu (napríklad do hĺbky alebo do šírky) zistíme, či je z vrcholu 0 dosiahnuteľný vrchol n. Keďže je podľa zadania potrebné aj vypísať jedno riešenie, existujú vstupy, pre ktoré lepšie riešenie neexistuje. Zamyslite sa však, či by mala táto úloha lepšie riešenie, ak by sme chceli iba overiť, či nejaký postup existuje.

z1621. Na uloženie záznamov o čakajúcich ľuďoch použijeme haldu, prípadne vyvažovaný binárny strom.

z1622. Zitkin postup nefunguje. Aby sme to ukázali, nájdeme situáciu, v ktorej sa dajú popárovať úplne všetky dvojice, no už prvým krokom Zitkinho postupu vyrobíme situáciu, ktorá k optimálnemu riešeniu nepovedie.



Chlapci sú znázornení ako body $ABCDEFGH$. Ako prvého si Zita vyberie chlapca E, lebo každý iný chlapec má viac potenciálnych partneriek. Chlapec E je ochotný tancovať s dievčatami W a X, pričom W má 5 a X len 4 potenciálnych partnerov, preto Zita nechá tancovať E s X – a to zjavne nevedie k optimálnemu riešeniu.

z1623. *Riešenie nadväzuje na text „Euklidov algoritmus“ na str. 244.*
Ak riešenie existuje, bude počet hláv pred posledným úderom rovný buď presne s, alebo presne z. Môžeme samostatne overiť dosiahnuteľnosť každého z týchto počtov hláv: ak je aspoň jeden dosiahnuteľný, riešenie existuje, ak nie, tak nie.

Ak platí $s = a$, môžeme strieborný meč odignorovať, a pre jeden meč je už úloha ľahká. Analogicky ľahko vyriešime prípad, keď $z = b$. Teraz už zostali len tri možné prípady: buď každý z mečov svojim použitím niekoľko hláv uberie (t.j. $a < s$ a $b < z$), alebo jeden hlavy pridáva a druhý uberá, alebo oba hlavy pridávajú.

Riešenie týchto možností zahŕňa ošetrenie viacerých špeciálnych prípadov a hľadanie nezáporného riešenia (x, y) diofantickej rovnice $p = qx + ry$. Všetky riešenia tejto rovnice získame nasledovne: Ak q a r majú spoločného deliteľa $d > 1$, tento musí deliť aj p, inak riešenie neexistuje. Ak d delí aj p, všetky koeficienty vydelíme d, čím dosiahneme, že q a r budú nesúdeliteľné.

Uvažujme teraz našu rovnicu modulo q. Dostávame $p \equiv ry \pmod{q}$. Táto rovnica má jediné riešenie $y_0 \in \{0, 1, \ldots, q-1\}$ a toto riešenie vieme nájsť v čase $O(\log q)$ napríklad použitím rozšíreného Euklidovho algoritmu. Potom pre každé celé y_1 zodpovedá hodnote $y = y_0 + qy_1$ jedno riešenie pôvodnej diofantickej rovnice.

z1624. *Riešenie je uvedené v texte „Prehľadávanie do šírky“ na str. 247.*

z1625. Prevedieme oba palindrómy na ich poradové čísla, tie sčítame a výsledok prevedieme naspäť na palindróm. Kľúčové je pozorovanie, že každý n-ciferný palindróm je jednoznačne určený prvými $\lceil n/2 \rceil$ ciframi a jediné obmedzenie na tieto cifry je, že prvá z nich nesmie byť nulová. Palindrómov s n ciframi je teda $9 \cdot 10^{\lceil n/2 \rceil - 1}$. Navyše vieme pomocou tohto pozorovania aj ľahko zistiť, že napríklad palindrómov tvaru `47..74` je 10^1 a palindrómov tvaru `1...1` je 10^2.

Keď teraz máme daný nejaký n-ciferný palindróm, pomocou vyššie uvedeného pozorovania ľahko zistíme jeho poradové číslo – stačí zistiť, koľko existuje od neho menších palindrómov. Napríklad od palindrómu `21312` sú menšie práve všetky palindrómy s 1 až 4 ciframi (tých je $9 + 9 + 90 + 90 = 198$) a tiež palindrómy tvarov `1...1`, `20.02`, `21012`, `21112` a `21212` (tých je $100 + 10 + 1 + 1 + 1 = 113$). Dokopy má teda `21312` poradové číslo $198 + 113 + 1 = 312$.

Analogicky funguje aj prevod poradového čísla na zodpovedajúci palindróm: Najskôr zistíme jeho počet cifier n: číslo n je najmenšia hodnota, pre ktorú je 1- až n-ciferných palindrómov dokopy aspoň toľko ako veľké je naše poradové číslo.

Napríklad pre poradové číslo 312: Palindrómov dĺžky 1, 2, 3 a 4 máme postupne 9, 9, 90 a 90. Keďže $9 + 9 + 90 + 90$ je menej ako 312, má hľadaný palindróm viac ako 4 cifry. Pre $n = 5$ máme už $9 + 9 + 90 + 90 + 900 \geq 312$, preto $n = 5$. Keď už poznáme n, vieme aj to, koľký z n-ciferných palindrómov hľadáme. V našom príklade je jeho poradové číslo $312 - 9 - 9 - 90 - 90 = 114$.

Teraz postupne určíme cifry hľadaného palindrómu, začínajúc najvýznamnejšou. V našom príklade vidíme, že 5-ciferných palindrómov začínajúcich `1` je len 100. To je primálo, preto všetky takéto palindrómy preskočíme a hľadáme palindróm číslo 14 zo zvyšných. Ten zjavne začína cifrou `2`. Teraz preskočíme 10 palindrómov začínajúcich `20` a zoberieme štvrtý z palindrómov začínajúcich `21`, čo je `21312`.

z1631. Potrebujeme si zapamätať všetky piesne, ktoré už boli hitom a ich názov sa skladá iba z písmen `Z` a `Y`. Na to je vhodný písmenkový strom alebo hešovacia tabuľka – vyhľadávať tak vieme v čase úmernom dĺžke názvu piesne.

z1632. Dokážeme, že ľubovoľné optimálne riešenie sa dá bez zmeny počtu zapísaných prednášok prerobiť na to, ktoré nájde VILov algoritmus. Pozrime sa na prvú prednášku, v ktorej sa obe riešenia líšia. VILov algoritmus vybral prednášku V, ale na jej mieste je v optimálnom riešení prednáška O. Z VILovho algoritmu vyplýva, že O nekončí skôr ako V, preto môžeme v optimálnom riešení vymeniť O za V a stále sa budú dať navštíviť nasledujúce prednášky. Najviac po n krokoch týmto postupom dostaneme riešenie VILovho algoritmu; to znamená, že je korektný.

Pretože týždeň má konštantný počet minút, môžeme prednášky usporiadať podľa ich konca count-sortom v čase $O(n)$. VILov algoritmus sa potom dá implementovať jedným prechodom prednáškami s časovou zložitosťou $O(n)$.

z1633. Medzi variáciami k-tej triedy z čísel 1 až n máme nájsť tú, ktorá je i-ta v lexikografickom poradí. Počet variácií k-tej triedy z n prvkov je $V_k(n) = n(n-1)\cdots(n-k+1)$, lebo na prvé miesto môžeme dať jeden z n prvkov, na druhé jeden zo zvyšných $n-1$, atď.

Nech vyberieme na prvé miesto ľubovoľné číslo, vždy zostane $V_{k-1}(n-1)$ možností, ako doplniť ostatné. Preto ak hľadáme i-tu variáciu v poradí, vieme jej prvé číslo určiť z podielu čísla $i-1$ a čísla $V_{k-1}(n-1)$. Zostáva vyriešiť podobnú úlohu ako na začiatku: nájsť i'-tu variáciu $(k-1)$-vej triedy zo zvyšných $n-1$ čísel. Jej poradie i' je určené zvyškom pri vyššie spomínanom delení.

Množinu čísel, ktoré sme ešte nepoužili, môžeme mať uloženú vo vyvažovanom binárnom vyhľadávacom strome; to umožní odstránenie čísla aj nájdenie m-tého najmenšieho čísla v čase $O(\log n)$. Existuje aj riešenie pomocou intervalového stromu, ktoré sa jednoduchšie píše a má rovnakú časovú zložitosť týchto operácií.

z1634. *Riešenie je uvedené v texte „Dijkstrov algoritmus“ na str. 248.*

z1635. *Pozri riešenie úlohy 1731.*

1711. *Riešenie nadväzuje na text „Dijkstrov algoritmus“ na str. 248.*
Každé mesto i reprezentujeme dvoma vrcholmi v_i^M a v_i^G – podľa toho, kto stráži bránu, ktorou sme vošli (a musíme vyjsť). Ak sú mestá i a j spojené cestou, pričom bránu i strážia mušketieri a j gardisti, do nášho grafu pridáme hranu $v_i^M v_j^G$. Nakoniec pridáme dve špeciálne hrany $v_1^M v_1^G$ a $v_n^M v_n^G$ nulovej dĺžky (pretože je jedno, ktorou bránou začneme a ktorou skončíme). V takto zostrojenom grafe nájdeme najkratšiu cestu z v_1^M do v_n^M.

1712. Použijeme dynamické programovanie. Nech $P[a,b,c]$ je minimálna cena na nákup lístkov pre a dospelých, b detí a c psov. Ak by sme si kúpili i-ty typ lístku (pre x_i dospelých, y_i detí a z_i psov za cenu p_i), najlepší spôsob, ako dokúpiť zvyšné lístky stojí $P[a-x_i, b-y_i, c-z_i]$. Platí teda $P[a,b,c] = \min_i\{P[a-x_i, b-y_i, c-z_i]+p_i\}$. Tento algoritmus ľahko implementujeme v čase $O(abcn)$.

1713. *Riešenie nadväzuje na text „Medián postupnosti“ na str. 243.*
Dá sa dokázať, že ak si zvolíme pozície cvičencov na konci, vedia sa tam všetci dostať najkratšou možnou cestou bez toho, aby sa museli obchádzať; navyše takým spôsobom, že sa najskôr budú hýbať vodorovne (nájdu si svoj stĺpec) a potom zvislo. V optimálnom riešení pôjde na najľavejšiu pozíciu cvičenec, ktorý stojí najviac vľavo, na druhú pozíciu druhý zľava, a tak ďalej, až na najpravejšiu pozíciu príde najpravejší cvičenec.

Optimálnym riadkom, v ktorom sa majú cvičenci stretnúť, je medián ich y-ových súradníc. Optimálny stĺpec pre prvého cvičenca nájdeme podobne: V čase $O(n \log n)$: cvičencov usporiadame zľava doprava a následne pre každé i posunieme i-teho cvičenca o $(i-1)$ doľava. Optimálny stĺpec pre prvého cvičenca je mediánom takto upravených x-ových súradníc cvičencov.

Existuje aj lineárne riešenie. Potrebujeme sa zbaviť triedenia, ktoré je jedinou nelineárnou časťou v predchádzajúcom riešení. Rýchlejší spôsob je takýto: Zistíme medián *neposunutých* cvičencov; označme ho m. Keďže každého cvičenca by sme posunuli najviac o n doľava, medián posunutých cvičencov je niekde medzi $m-n$ a m. Pomocou count-sortu vieme spočítať, koľko cvičencov je vľavo od $m-n$, koľko je v každom stĺpci medzi $m-n$ a m a koľko je vpravo od m. V lineárnom čase spočítame celkový počet krokov, ktoré treba spraviť, ak by rad začal v stĺpci $m-n$; no a ak poznáme počet krokov pre začiatok v stĺpci x, vieme v konštantnom čase dopočítať počet krokov pre stĺpec $x+1$. Takto prezrieme $O(n)$ možností a vyberieme najlepšie z nájdených riešení.

1714. Zostrojme si graf, kde vrcholmi budú jednotlivé meny a medzi u a v bude orientovaná hrana ceny $c(u,v)$, ak sa pri výmene u za v množstvo peňazí z-$c(u,v)$-násobí. Úlohou je potom nájsť cyklus $v_1, v_2, \ldots, v_m, v_1$ taký, že $c(v_1,v_2)\cdot c(v_2,v_3)\cdots c(v_m,v_1) > 1$.

Táto úloha sa dá riešiť priamo, ale vieme si ju zjednodušiť trikom s logaritmami: Platí $\log(a\cdot b) = \log a + \log b$. Ak teda cenu každej hrany uv zmeníme z $c(u,v)$ na $-\log c(u,v)$, úloha sa nám zjednoduší na nájdenie záporného cyklu v grafe. To je známy problém, ktorý sa dá riešiť Floydovým-Warshallovým algoritmom v čase $O(n^3)$.

1715. Pre každý stĺpec sa budeme pozerať na počet voľných políčok medzi bielym a čiernym pešiakom – tento budeme nazývať ich vzdialenosťou.

Pri hre zo začiatočnej pozície pre párne s vyhráva čierny tak, že sa bude snažiť, aby po jeho ťahu boli pre každé i v stĺpcoch číslo $2i-1$ a $2i$ rovnaké vzdialenosti. Navyše bude svojimi figúrkami ťahať len dodola. (Rozmyslite si, že to vždy vie urobiť.) Pre nepárne s vyhráva biely tak, že v prvom ťahu potiahne svojou figúrkou v poslednom stĺpci úplne dohora a následne použije vyššie popísanú stratégiu pre čierneho.

Vo všeobecnosti je táto hra takmer ekvivalentná s hrou NIM: namiesto vzdialenosti medzi pešiakmi si môžeme predstaviť kôpku zápaliek, z ktorej vieme v jednom ťahu ľubovoľne veľa odobrať. Ak sme v hre NIM vo vyhrávajúcej pozícii, v hre s pešiakmi vyhráme

nasledujúcou stratégiou: ak súper potiahne tak, aby vzdialenosť medzi pešiakmi v niektorom stĺpci zväčšil, potiahneme v tom istom stĺpci tak, aby sa vzdialenosť pešiakov zmenila na pôvodnú. Keďže každou takouto dvojicou ťahov sa jeden súperov pešiak priblíži ku jeho kraju, môže takýto ťah súper spraviť len konečne veľa krát. Následne je nútený potiahnuť tak, že niektorú vzdialenosť zmenší. Toto interpretujeme ako ťah v hre NIM, nájdeme v nej optimálnu odpoveď a príslušne potiahneme pešiakom.

Dá sa dokázať, že vyhrávajúce pozície v hre NIM sú tie, kde je bitový xor veľkostí kôpok nenulový.

1721. Za mesto, v ktorom sa majú stretnúť, môžeme vždy vyhlásiť to, do ktorého vchádza najviac teleportov. (Skúste si toto tvrdenie dokázať, napríklad sporom.) Samotný program je potom priamočiary.

1722. Počet inverzií v ľubovoľnej postupnosti dĺžky n vieme spočítať v čase $O(n \log n)$ upraveným merge-sortom – keď spájame nejaké dve postupnosti do jednej a vyberieme prvok z druhej z nich, ten práve predbehol všetky z prvej, ktoré sme ešte nevybrali.

Iná možnosť bola využiť, že máme na vstupe permutáciu čísel od 1 po n a použiť intervalový strom, ktorý bude v logaritmickom čase podporovať operácie „vlož číslo x“ a „zisti počet čísel, ktoré sú v rozsahu od 1 po y“.

1723. Vstup predstavuje acyklický orientovaný graf. Vrcholy, do ktorých nič nevedie, volajme začiatky a vrcholy, z ktorých nič nevedie, volajme konce. Nech z je počet začiatkov a k počet koncov. Zjavne každé riešenie potrebuje pridať aspoň $\max(z, k)$ lanoviek, lebo aj do každého začiatku, aj z každého konca potrebujeme aspoň jednu.

Naopak, platí, že vždy existuje nejaká vyhovujúca množina presne $\max(z, k)$ lanoviek. Jeden spôsob ako ju zostrojiť: Najskôr pažravo zostrojujeme vrcholovo disjunktné cesty zo začiatkov do koncov. Keď už sa to nedá, spojíme lanovkou koniec prvej cesty so začiatkom druhej cesty, atď., až koniec poslednej z nich so začiatkom prvej. Tým dostaneme jeden veľký silne súvislý komponent. Teraz nám ešte mohlo zostať niekoľko nespárených začiatkov a zároveň niekoľko nespárených koncov. Ľahko však nahliadneme, že tieto už stačí pospájať ľubovoľne. (Kým máme aj konce, aj začiatky, spojíme ľubovoľný nespárený koniec s ľubovoľným nespáreným začiatkom. Keď sa nám už jeden typ vrcholov minie, zostávajúce vrcholy druhého typu spojíme s ľubovoľným spáreným vrcholom.)

Vhodná implementácia vyššie uvedeného postupu má časovú zložitosť lineárnu od veľkosti grafu.

1724. Aby bol pás najužší, musí jeden okraj obsahovať jednu celú stranu n-uholníka a druhý okraj aspoň jeden vrchol (skúste si to dokázať). Na základe tohto tvrdenia dostaneme jednoduchý algoritmus v čase $O(n^2)$ (ku každej strane nájdeme najvzdialenejší vrchol a z týchto vzdialeností vyberieme tú najmenšiu). Existujú však aj rýchlejšie riešenia. Všimnite si, že ak postupne prechádzame vrcholy, ich vzdialenosť od danej strany najskôr rastie, dosiahne maximum a potom klesá. Navyše ak v_0v_1 a v_1v_2 sú dve susedné strany a v_i, v_j sú k nim najvzdialenejšie vrcholy, tak $j \geq i$. Stačí teda nájsť najvzdialenejší vrchol od prvej strany a ďalšie už nájdeme jednoducho, keď sa budeme hýbať stále jedným smerom (kým vzdialenosť rastie). Toto riešenie sa dá implementovať v čase $O(n)$.

1725. Označme pozície kameňov a_1, a_2, a_3, a_4 tak, aby platilo $a_1 < a_2 < a_3 < a_4$. Prehrávajúce sú práve tie pozície, v ktorých platí $a_2 - a_1 = a_4 - a_3$. Totiž keď súper v takejto pozícii potiahne, zmení práve jednu z hodnôt a_i, čím poruší rovnosť. A my následne vieme tú stranu, ktorá je aktuálne väčšia, vhodným ťahom zmenšiť tak, aby znova nastala rovnosť.

1731. Majme orientovaný graf, kde vrcholy sú KSPáci a hrana uv ceny k znamená, že u je v dlžný k kofôl. Všimnime si vrchol v; ak existujú hrany uv a vw s cenami k_1 a k_2, tak v budeme volať prostredník. Od oboch hrán môžeme odpočítať $\min(k_1, k_2)$ (tým aspoň jednu zrušíme) a naopak pridať hranu uw. Takto budeme pokračovať, kým do v vchádza

aj vychádza nejaká hrana. Keďže v každom kroku znížime počet hrán aspoň o jednu, bude celkový počet krokov nanajvýš rovný pôvodnému počtu hrán.

Ak jednoducho použijeme na uloženie grafu maticu susednosti, dostaneme algoritmus, ktorý bude mať časovú zložitosť kvadratickú od počtu vrcholov. Ak chceme dosiahnuť optimálnu časovú zložitosť (lineárnu od veľkosti vstupného grafu), musíme použiť šikovnejšiu implementáciu. Pre každý vrchol si budeme samostatne pamätať zoznam vchádzajúcich a samostatne zoznam vychádzajúcich hrán. Keď spracúvame vrchol, postupne tieto hrany upravujeme, až kým sa jeden zo zoznamov nevyprázdni.

1732. V poslednom kroku určite kúpime najdrahší lístok. Preto stačí zistiť, akú najväčšiu hodnotu menšiu ako s vieme dosiahnuť na nanajvýš $p-1$ stlačení. Na to použijeme dynamické programovanie: pre x od 0 do $s-1$ nech $D[x]$ je najmenší počet stlačení, na ktoré vieme dosiahnuť sumu x. Hodnoty $D[x]$ vieme počítať pomocou rekurentného vzťahu: $D[0]=0$ a pre $x>0$ je $D[x]=1+\min_i D[x-c_i]$ (kde c_i sú jednotlivé ceny lístkov a minimum počítame cez všetky $c_i \le x$). Na záver stačí nájsť najväčšie y, pre ktoré $D[y]<p$. Toto riešenie má časovú zložitosť $O(ns)$, pamäťová sa dá zredukovať až na $O(\max c_i)$.

1733. Efektívne riešenie je založené na podobnej myšlienke ako quick-sort: Vezmeme náhodnú skrutku a podľa nej rozdelíme matice na menšie, väčšie a jednu rovnakú. Keď teraz máme maticu, ktorá pasuje k našej skrutke, podľa nej zase rozdelíme ostatné skrutky na menšie a väčšie. Teraz nám už ostáva iba rekurzívne popárovať menšie skrutky s menšími maticami a väčšie s väčšími.

Ak budeme vždy deliť matice podľa náhodnej skrutky, očakávaná časová zložitosť bude $O(n \log n)$. Existuje aj deterministický algoritmus, ktorý má aj v najhoršom možnom prípade túto časovú zložitosť, je ale neporovnateľne komplikovanejší.[6]

1734. Tyče môžeme v čase $O(n \log n)$ usporiadať podľa dĺžky. Potom platí, že ak existuje nejaká trojica tyčí, ktorá tvorí trojuholník, tak určite existuje trojica bezprostredne po sebe nasledujúcich tyčí, ktorá tvorí trojuholník. Toto už vieme overiť v lineárnom čase.

Lepšie riešenie: V zadaní máme navyše dané obmedzenie na dĺžku tyčí. Ak by sme chceli zostrojiť čo najviac tyčí, ktoré *netvoria* trojuholník, zvolili by sme postupnosť $1, 1, 2, 3, 5, 8, \ldots$ (tzv. Fibonacciho čísla, každé je súčtom dvoch predošlých). Dá sa dokázať, že ak máme n tyčí dĺžky aspoň 1 a všetky sú kratšie ako n-té Fibonacciho číslo, tak medzi nimi existuje aspoň jeden trojuholník. Keďže už 45-te Fibonacciho číslo je väčšie ako zadaný limit 10^9, ak je tyčí aspoň 45, odpoveď je určite `ÁNO`. Ak je ich menej, použijeme vyššie uvedený postup. Časová aj pamäťová zložitosť tohto riešenia je konštantná.

1735. Prvá podúloha má riešenie v lineárnom čase. V prvom ťahu Santo vypije fľašu s objemom $2n$, Banto nejakú menšiu, takže Santo zatiaľ vedie. Santo si následne ofarbí ostatné fľaše (ktorých je párny počet) striedavo čiernou a bielou a spočíta si, ktorých celkový objem je väčší. Tú farbu (resp. v prípade rovnosti ľubovoľnú) si vyberie a následne bude vždy piť fľašu vybranej farby, až kým sa všetky fľaše neminú. Rozmyslite si, že toto vždy vie dosiahnuť. Vo zvyšku hry teda Santo vypije aspoň toľko ako Banto a teda dokopy s prvým ťahom vypije viac.

Druhú podúlohu vieme riešiť dynamickým programovaním: pre každý súvislý podúsek zadanej postupnosti si spočítame jeho celkový objem a tiež celkový objem toho, čo by vypil prvý hráč, ak by sa hralo len na tomto úseku a obaja hráči by hrali optimálne.

1741. Hľadanú množinu zátarasov tvorí minimálny rez grafu oddeľujúci od seba vrcholy 1 a n. Tento vieme nájsť tak, že nájdeme (napríklad Fordovým-Fulkersonovým algoritmom) najväčší tok z vrcholu 1 do vrcholu n. Následne nech A je množina vrcholov, do ktorých ešte existuje z vrcholu 1 zlepšujúca cesta. Potom hľadaný minimálny rez je tvorený tými hranami, ktoré majú práve jeden koniec v množine A.

[6] Vymysleli ho Janos Komlos, Yuan Ma a Endre Szemeredi až v roku 1996.

1742. Na začiatku načítame popis ponorky a zistíme si o každom jej riadku, kde začína a aký je dlhý. Potom postupne po riadkoch spracúvame mapu morskej priekopy a pre každé jej políčko zistíme, či sa vieme ponorkou ponoriť tak, aby ľavý dolný roh bitmapy predstavujúcej ponorku bol presne na danom políčku.

Po načítaní riadku mapy ho prejdeme sprava doľava a pre každé políčko si spočítame, aký dlhý úsek voľných políčok vedie od neho doprava. Pomocou tejto informácie vieme pre ľubovoľnú polohu ponorky v čase $O(b)$, teda lineárnom od počtu jej riadkov, overiť, či sa na dané miesto zmestí. Následne z toho, že vieme, ktoré polohy ponorky v predchádzajúcom riadku boli naozaj dosiahnuteľné, v čase $O(n)$ spočítame dosiahnuteľnosť polôh v aktuálnom riadku.

Časová zložitosť tohto riešenia je $O(bmn)$. Stačí si pamätať iba b riadkov priekopy, dosiahneme tak pamäťovú zložitosť $O(bm)$.

1743. Na tom, či Schumacher vyhrá, sa nič nezmení, ak body za prvé, druhé, aj tretie miesto prenásobíme tou istou konštantou. Preto môžeme predpokladať, že za tretie miesto je presne 1 bod. Zostali nám dve neznáme: počet bodov x za prvé a y za druhé miesto.

Pre každého pretekára iného ako Schumacher máme jednu podmienku: „tento pretekár má získať menej bodov ako Schumacher“. Presnejšie ak a_i, b_i a c_i sú rozdiely počtov prvých, druhých a tretích miest medzi Schumacherom a i-tym pretekárom, musí platiť, že $a_i x + b_i y + c_i > 0$.

Geometrický pohľad: Ak každej dvojici (x, y) priradíme bod v rovine, každá podmienka bude zodpovedať nejakej polrovine. Nás zaujíma, či existuje bod, ktorý leží v prieniku všetkých $n-1$ polrovín. Prienik $n-1$ polrovín vieme zostrojiť v čase $O(n \log n)$ napríklad použitím metódy rozdeľuj a panuj, prípadne využitím duality a následným zostrojením vhodného konvexného obalu. V našej úlohe však nepotrebujeme celý prienik daných polrovín zostrojiť, len zistiť, či má neprázdny obsah. Táto úloha súvisí s lineárnym programovaním a dá sa riešiť v čase $O(n)$.

Ľahko naprogramovateľné takmer optimálne riešenie: Náhodne preusporiadame polroviny a po jednej ich budeme spracúvať, pričom si nebudeme udržiavať celý prienik už spracovaných polrovín, ale len ten jeho vrchol, ktorý má minimálne x. Ako sa tento vrchol v zmení, keď pridáme novú polrovinu určenú priamkou ℓ? Sú len tri možnosti: buď to zostane v, alebo bude nový vrchol niekde na priamke ℓ, alebo už bude prienik polrovín prázdny. Ak v neleží v polrovine určenej priamkou ℓ, ešte raz prejdeme všetky skôr spracované polroviny, spočítame prienik intervalov, ktoré vytínajú na ℓ, a tak nájdeme nový hľadaný bod (alebo zistíme, že už neexistuje). V najhoršom možnom prípade by tento algoritmus bol síce kvadratický od n, dá sa však dokázať, že pri spracúvaní polrovín v náhodnom poradí bude jeho očakávaná časová zložitosť $O(n)$.

Keď už takto nájdeme jeden vrchol na hranici prieniku daných polrovín, potrebujeme ešte overiť, či má tento prienik neprázdne vnútro. Preto na záver ešte v čase $O(n)$ overíme, či dotyčným prienikom nie je bod, úsečka, ani polpriamka.

1744. V lineárnom čase vieme predpočítať čiastočné súčty – t.j. pre každé x od 0 po n spočítame súčet prvých x členov zadanej postupnosti a označme ho s_x. Pomocou týchto údajov vieme v konštantnom čase zistiť súčet, a teda aj priemer, ľubovoľného úseku.

Riešenie v čase $O(kn)$ je založené na nasledujúcom pozorovaní: Určite existuje optimálny úsek, ktorý má menej ako $2k$ prvkov. Ľubovoľný úsek U dĺžky aspoň $2k$ totiž vieme rozstrihnúť na dva úseky U_1, U_2 dĺžky aspoň k a aspoň jeden z U_1 a U_2 musí mať priemer väčší alebo rovný ako U. Stačí teda vyskúšať $O(n)$ možných začiatkov úseku a pre každý začiatok $O(k)$ možných dĺžok.

Riešenie v čase $O(n)$: Pre každé x zostrojme v rovine bod $S_x = [x, s_x]$. Našu úlohu vieme previesť na hľadanie takej dvojice indexov a, b, aby mali rozdiel aspoň k a zároveň aby priamka vedúca cez S_a a S_b mala najväčšiu možnú smernicu. (Všimnite si, že priemer hodnôt medzi a a b je $(S_b - S_a)/(b-a)$ a to je presne tangens uhla pri vrchole $[a, S_a]$

v pravouhlom trojuholníku $[a, S_a]$, $[b, S_a]$ a $[b, S_b]$. Tangens tohto uhla zodpovedá smernici prepony pravouhlého trojuholníka.)

Ak si zvolíme konkrétny bod S_b, najlepší S_a k nemu musí ležať na spodnom konvexnom obale množiny $\{S_1, \ldots, S_{b-k}\}$. Toto pozorovanie využijeme pri našom riešení. Budeme postupne v cykle zvyšovať b, pričom si budeme v zásobníku udržiavať spodný konvexný obal bodov, spomedzi ktorých už môžeme vyberať S_a. Tiež si budeme pamätať doteraz najlepšie riešenie a jemu zodpovedajúcu dotyčnicu ku konvexnému obalu. Celková časová zložitosť bude pri dobrej implementácii lineárna, lebo každý bod do konvexného obalu raz pridáme, najviac raz ho odstránime a bod dotyku optimálnej dotyčnice budeme posúvať len jedným smerom.

1745. Prehrávajúce pozície v tejto hre sú práve tie čísla miest, ktoré sú deliteľné 4. Totiž žiaden povolený ťah nezachováva deliteľnosť 4 (takže ak sme na ťahu v meste, ktorého číslo je deliteľné 4, musíme potiahnuť do mesta, ktoré 4 deliteľné nie je) a sú povolené ťahy o 1, 2 aj 3 (takže ak sme na ťahu v meste, ktorého číslo nie je deliteľné 4, vieme potiahnuť do mesta, ktoré je deliteľné 4).

z1711. Hlavný problém je zistiť, ktorým dňom týždňa začína daný rok. To vieme spraviť nasledovne: Zapamätáme si začiatočný deň pre nejaký rok r_1. Začiatočný deň sa každým nepriestupným rokom posunie o jeden a každým priestupným o dva dni. To preto, že 365 dáva po delení 7 zvyšok 1. Napríklad rok 2008 začínal utorkom, 2009 štvtkom a 2010 piatkom. Vďaka tomuto pozorovaniu už vieme v konštantom čase určiť začiatočný deň ľubovoľného roku r_2.

V tejto chvíli už vieme dopočítať deň v týždni, kedy začína príslušný mesiac, a z toho a jeho dĺžky určiť hľadaný počet pracovných dní.

z1712. Treba si uvedomiť, že presunúť úsek je to isté ako vymeniť dva po sebe idúce úseky – jeden tvorený prvkami, ktoré presúvame, a jeden ostatnými prvkami, ktoré tiež zmenia pozíciu. Stačí nám teda naprogramovať procedúru *vymeň*(p, q, r), ktorá vymení úseky $p \ldots q-1$ a $q \ldots r-1$, a túto použiť s vhodnými parametrami.

a) Rekurzívne riešenie: Ak sú oba úseky rovnako dlhé, vymeníme ich jednoducho. Ak je prvý úsek kratší, rekurzívne zavoláme *vymeň*$(p, q, 2q-p)$, teda posunieme úsek o celú jeho dĺžku, a následne rekurzívne vyriešime *vymeň*$(q, 2q-p, r)$. Naopak, ak je kratší druhý úsek, posunieme ho doľava pomocou *vymeň*$(2q-r, q, r)$ a rekurzívne doriešime *vymeň*$(p, 2q-r, q)$. Všimnite si, že každou výmenou sa aspoň jeden znak dostane na správne miesto, takže časová zložitosť je lineárna od dĺžky $r-p$.

b) Trikové riešenie: Nech *rev*(i, j) zrkadlovo obráti úsek $i \ldots j-1$, t.j. text $a_i a_{i+1} \cdots a_{j-1}$ zmení na $a_{j-1} \cdots a_{i+1} a_i$ (to je ľahké napísať aj bez pomocného poľa v lineárnom čase). Rozmyslite si, že procedúra *vymeň*(p, q, r) sa dá teraz napísať takto: *rev*(p, q); *rev*(q, r); *rev*(p, r).

z1713. Zamestnancov si očíslujeme 0 až $n-1$. Každú ich podmnožinu vieme reprezentovať ako postupnosť n bitov: i-ty bit bude 1, ak zamestnanec i je na stavbe a 0, ak nie je. Našou úlohou je teda zoradiť všetky n-bitové binárne reťazce do postupnosti tak, aby sa každé dva po sebe idúce líšili len v jedinom bite. Takáto postupnosť sa volá Grayov kód.

Zoberme dve kópie Grayovho kódu pre $n-1$ bitov. Pred každý člen prvej kópie pridajme 0, pred každý člen druhej 1. Grayov kód pre n bitov teraz zostrojíme tak, že najskôr zoberieme prvky upravenej prvej kópie od prvého po posledný, a následne prvky upravenej druhej kópie od posledného po prvý.

Riešenie priamočiaro implementujúce uvedenú konštrukciu má časovú aj pamäťovú zložitosť $O(n2^n)$. Pamäťovú zložitosť vieme zlepšiť na $O(n)$ tak, že budeme prvky Grayovho kódu postupne generovať pomocou rekurzívnej procedúry. Existujú aj „trikové“ riešenia. Napríklad platí, že $(i+1)$-vý člen nášho kódu je binárny zápis čísla i xor $(i \text{ div } 2)$.

z1714. Implementácia v čase $O(rs)$ je pomerne ľahká. Namiesto rozpisovania štyroch prípadov podľa aktuálneho Jonesovho natočenia je lepšie si smery očíslovať od 0 po 3 a ako konštanty si zadefinovať posuny zodpovedajúce jednotlivým smerom. Netreba zabudnúť, že ukončenie treba ošetriť aj pri návrate ku vchodu, ak východ nie je dosiahnuteľný.

z1715. Vzorové riešenie dokáže všetko potrebné formátovanie vyrobiť v čase lineárnom od veľkosti vstupu, pričom si vystačí s pamäťou konštantnej veľkosti – stačí si pamätať práve spracúvaný riadok textu, ktorý má nanajvýš 80 znakov. Asi najťažšou časťou riešenia je parsovanie vstupu pri načítaní.

z1721. Na najkratšej ceste sa zjavne nebude opakovať žiaden vrchol. A takých ciest z 1 do n je iba konečne veľa. Preto akonáhle existuje nejaká cesta z 1 do n, určite existuje aj najkratšia cesta. To, či existuje cesta z 1 do n, sa dá zistiť ľubovoľným prehľadávaním. Dĺžky hrán môžeme ignorovať.

z1722. Triviálne riešenie je pre každú dvojicu v lineárnom čase spočítať odpoveď; to je však pri veľkom počte poschodí pomalé. Mohli by sme si predpočítať všetky súčty a potom už iba vyhľadávať v tabuľke – všetkých súčtov je však až $\Theta(n^2)$.

V skutočnosti si stačí predpočítať iba tzv. čiastočné súčty: k pôvodnej postupnosti $p_1, p_2, \ldots, p_n$ si vypočítame postupnosť $S[0], S[1], S[2], \ldots, S[n]$, kde $S[0] = 0$ a $S[i+1] = S[i] + p_{i+1}$. Zjavne $S[i]$ je súčet prvých i čísel zo vstupu. Ak potom chceme spočítať súčet $p_i + p_{i+1} + \cdots + p_j$, stačí od $S[j]$ odpočítať $S[i-1]$. Predpočítanie hodnôt v poli S spravíme v lineárnom čase, každú otázku následne zodpovieme v konštantnom čase.

z1723. Asi najjednoduchšie riešenie je rekurzívne a kopíruje definíciu výrazu: ak je na vstupe číslo, načítame ho a vrátime jeho hodnotu; ak je na vstupe operátor, rekurzívne vypočítame hodnotu oboch podvýrazov za ním a následne tieto čísla sčítame.

z1724. Každú včielku posúvame po krokoch o vektor jednotkovej dĺžky, ktorý smeruje k nasledujúcej včielke. Vektor upravíme na jednotkový jednoducho tak, že ho predelíme jeho dĺžkou (tá sa dá spočítať z Pytagorovej vety).

Včielky z prémiovej úlohy budú stále vo vrcholoch štvorca, ktorý sa bude otáčať a zmenšovať, takže pôjdu po špirále do bodu $[50, 50]$.

z1725. Jedna možnosť je ukladať si názvy obrázkov a stránok do poľa. Keď na konci toto pole usporiadame, duplikáty sa budú nachádzať hneď za sebou. Preto ich môžeme pri vypisovaní preskočiť. Iná možnosť je použiť vhodnú dátovú štruktúru, akou je písmenkový strom alebo hešovacia tabuľka.

z1731. Vstup načítame do grafu, ktorého vrcholmi sú ľudia a hranami sú vzťahy dieťa-rodič a manžel-manželka. Do grafu musíme doplniť všetky chýbajúce hrany, t.j. každému dieťaťu doplniť vzťah k druhému rodičovi. Najbližšiemu vzťahu potom zodpovedá najkratšia cesta v grafe. Tú vieme nájsť prehľadaním grafu do šírky. Keďže graf je riedky (má iba $O(n)$ hrán), každú otázku vieme zodpovedať v lineárnom čase.

z1732. Postupne pre každý význačný bod spočítame uhol, pod ktorým doň vedie polpriamka začínajúca na Mt. Kopci. Zjavne platí, že i-ty bod vidíme voľným okom práve vtedy, keď všetky význačné body naľavo od neho majú tento uhol ostro menší. Priebežne si teda budeme pamätať dve informácie: najväčší uhol, ktorý sme doteraz videli, a tiež najvzdialenejší z viditeľných bodov. Takto vieme úlohu vyriešiť jediným prechodom v čase $O(n)$ a pamäti $O(1)$.

z1733. Postupne prejdeme celú mapu. Zakaždým, keď stretneme políčko patriace ešte nepreskúmanému ostrovu, prehľadávaním (do šírky alebo do hĺbky) označíme celý práve objavený ostrov. Popri tom môžeme zároveň spočítavať kopce, stačí pre každé spracované políčko otestovať, či je tam kopec. Toto riešenie má časovú zložitosť $O(rs)$, teda lineárnu od veľkosti mapy.

z1734. Táto úloha nie je náročná po algoritmickej stránke, treba len prejaviť a rozvíjať svoje programátorské zručnosti. Odporúčané je použiť pole na reprezentáciu hracieho plánu a vhodne definovať konštanty, ktoré pomôžu sprehľadniť a skrátiť vaše programy.

z1735. HTML súbory s odkazmi tvoria orientovaný graf (súbory sú vrcholy a odkazy hrany). Dostupné súbory vieme vypísať prehľadaním grafu (do hĺbky alebo do šírky).

Je dobré si súbory očíslovať, keďže s číslami sa pracuje lepšie ako s reťazcami. Na rýchle priraďovanie názvu súboru k číslu vrcholu môžeme použiť hešovaciu tabuľku alebo písmenkový strom.

1811. Toto riešenie, založené na metóde rozdeľ a panuj, bude súčasne aj dôkazom, že pre ľubovoľné výsledky turnaja existuje vhodné poradie družstiev. Naše riešenie bude analógiou triediaceho algoritmu merge-sort.

Usporiadať jedno družstvo je triviálne. Keď máme usporiadať n družstiev, rozdelíme ich na dve skupiny s $p = \lfloor n/2 \rfloor$ a $q = \lceil n/2 \rceil$ družstvami. Každú zvlášť rekurzívne usporiadame, čím získame poradia $a_1, \ldots, a_p$ a $b_1, \ldots, b_q$. Teraz spojíme tieto dve poradia do jedného: V každom kroku sa pozrieme na v danej chvíli prvé družstvá a_i a b_j v čiastočných poradiach. Víťaza ich vzájomného zápasu presunieme zo začiatku jeho poradia na aktuálny koniec celkového poradia.

Podobne ako u pôvodného algoritmu merge-sort je časová zložitosť tohto algoritmu $O(n \log n)$. (Poznámka: Vedeli by ste nájsť riešenie, ktoré je analógiou quick-sortu?)

1812. Najnáročnejší projekt priradíme prvému vedcovi. Ak existujú nejaké ďalšie projekty, ktoré mu ešte môžeme priradiť bez toho, aby sme tým prekročili povolenú pracovnú dobu, priradíme mu ľubovoľný z nich. Nezáleží na tom, ktorý si vyberieme, pretože všetky projekty, ktoré možno skombinovať s tým najnáročnejším, možno skombinovať s ľubovoľným iným projektom. Najjednoduchšie je zakaždým si vybrať najmenej náročný projekt. Takto zostrojíme jedno optimálne rozdelenie projektov v lineárnom čase.

1813. Ošetríme špeciálny prípad, keď jeden mnohouholník leží vnútri druhého, vtedy je dĺžka najkratšieho tunela 0. V ostatných prípadoch stačí koncové body najkratšieho tunela hľadať na obvode mnohouholníkov. Vyskúšame preto všetky dvojice strán mnohouholníkov a zapamätáme si najmenšiu vzdialenosť medzi nimi (ak sa strany pretínajú, je to 0). Časová zložitosť je $O(mn)$.

1814. Trik je v tom pozrieť sa na úlohu z opačného konca: Keď mám a hodov a b vajíčok, medzi koľkými rôznymi možnosťami viem s istotou nájsť tú správnu? Označme tento počet $T[a, b]$. Zjavne pre $a = 0$ alebo $b = 0$ je $T[a, b] = 1$: už musíme vedieť odpoveď.

Ako vyzerá všeobecný prípad? V prvom hode hodíme vajíčko z nejakej výšky. Ak sa vajíčko rozbije, zostane nám $a - 1$ hodov a $b - 1$ vajíčok. Aby sme vedeli s istotou dať správnu odpoveď, musí nám v tomto okamihu zostať nanajvýš $T[a-1, b-1]$ možností, medzi ktorými sa rozhodujeme. Podobne v prípade, že sa vajíčko nerozbije, nám musí zostať nanajvýš $T[a, b-1]$ možností. Z toho ľahko odvodíme, že $T[a, b] = T[a-1, b-1] + T[a, b-1]$.

Takto dostávame riešenie v $O(hm)$, kde h je potrebný počet hodov: postupne zvyšujeme h a počítame hodnoty $T[h, 0]$ až $T[h, m]$, kým nenájdeme najmenšie h, pre ktoré $T[h, m] \geq n$.

Toto riešenie vieme ďalej zlepšiť na optimálnu časovú zložitosť $O(h)$ tak, že ukážeme, ako v konštantnom čase spočítať z hodnoty $T[a, b]$ ľubovoľnú z hodnôt $T[a+1, b]$, $T[a-1, b]$ a $T[a, b-1]$.

1815. Dynamické programovanie. Postupne pre každé i zostrojíme množinu komôrok, v ktorých sa môže zlatokop nachádzať po prečítaní prvých i písmen z papierika. Šikovná implementácia má časovú zložitosť $O(\ell m)$ a pamäťovú zložitosť $O(m + n)$, kde n je počet komôrok, m počet chodieb a ℓ dĺžka textu na papieriku.

1821. Ak má každý kmeň nanajvýš jedného nepriateľa, stačí jedna skupina, inak zjavne potrebujeme aspoň dve. No a vhodné rozdelenie na dve skupiny vždy existuje. Jeden

algoritmus, ktorý ho zostrojí, vyzerá nasledovne: Na začiatku rozdelíme kmene ľubovoľne, napríklad všetky do jednej skupiny. Teraz dokola opakujeme: kým existuje kmeň, ktorý má vo svojej skupine viac ako jedného nepriateľa, tak nejaký taký zoberieme a prehodíme ho do opačnej skupiny.

Tento postup je určite konečný. Keď si totiž spočítame celkový počet znepriatelených dvojíc, ktoré sú práve spolu v skupine, tak každou výmenou tento počet znížime. A keďže na začiatku je týchto dvojíc nanajvýš $m \leq 3n/2$, počet prehodení bude lineárny.

Toto riešenie sa navyše dá implementovať tak, aby aj celková časová zložitosť bola lineárna. Jedna šikovná možnosť je začať s dvoma prázdnymi skupinami, postupne po jednom pridávať kmene a zakaždým, keď pridáme kmeň, vykonať postupne všetky prehodenia, ktoré sú v tej chvíli potrebné.

1822. Budeme postupne znižovať minútu od n po 1 a zakaždým jej priradíme niektorého občana, ktorý vtedy môže prísť (ak máme práve z čoho vyberať).

Kľúčové pozorovanie: Ak máme v nejakom okamihu na výber viac občanov, nič nepokazíme, ak vyberieme toho občana a, ktorého pokuta je najvyššia. Totiž predstavme si, že sme vybrali iného občana b. Keď teraz zoberieme výsledné poradie a vymeníme v ňom a a b, určite dostaneme aspoň rovnako dobré riešenie.

Stačí teda usporiadať občanov podľa termínu a následne vyššie uvedeným postupom prejsť cez všetky minúty. Občanov, ktorých sme ešte nepriradili, ale už môžeme priradiť, si budeme pamätať v prioritnej fronte (ktorú vieme implementovať napríklad ako haldu), pričom priorita bude rovná veľkosti pokuty.

Keď už popriraďujeme všetkých občanov, ktorí svoj termín stihnú, ešte raz prejdeme cez všetky minúty a na voľné miesta ľubovoľným spôsobom doplníme ostatných občanov.

Toto riešenie má časovú zložitosť $O(n \log n)$. Iné riešenie s tou istou časovou zložitosťou vieme zostrojiť na základe myšlienky, že nič nepokazíme, keď občana platiaceho najväčšiu pokutu pošleme na úrad v minútu, kedy má termín.

1823. Zvislé priamky na vstupe ošetríme ľahko (stačí uvažovať najpravejšiu). Na ostatné priamky sa môžeme pozerať ako na funkcie tvaru $y = kx + q$. Predstavme si teraz zvislú priamku z, ktorá sa nachádza napravo od posledného priesečníka. Je jasné, v akom poradí ostatné priamky pretnú priamku z – čím väčšie k, tým vyššie bude priesečník. Ak majú dve priamky rovnaké k, rozhoduje väčšie q. Toto poradie priamok vieme zostrojiť jednoduchým triedením v čase $O(n \log n)$.

Potom si stačí len uvedomiť, že najpravejší priesečník musí byť priesečníkom dvoch priamok, ktoré v tomto poradí susedia. Týchto priesečníkov je len $O(n)$, môžeme teda všetky spočítať a nájsť najpravejší.

1824. Pomocou prehľadávania s návratom (backtrackingu) zostrojíme všetky prípustné rozloženia lodí. Ak je prípustné rozloženie lodí jediné, vyhrali sme. Ak existuje políčko, na ktorom má jedno prípustné rozloženie lodí vodu a všetky ostatné tam majú loď, môžeme sa na toto políčko opýtať. Ak totiž dostaneme odpoveď „voda“, bola toto tá jedna chyba, čo sme mohli spraviť. Ďalšiu už nespravíme, keďže už vieme riešenie. A ak takéto políčko neexistuje, tak máme smolu – nech sa opýtame na hociktoré políčko, ktoré nie je jednoznačné, ak dostaneme odpoveď „voda“, sme nahratí.

Predchádzajúce riešenie vieme zlepšiť nasledovne: Všetky lode dokopy majú 28 políčok. Z toho vyplýva, že akonáhle nájdeme viac ako 29 prípustných rozložení lodí, môžeme prestať hľadať, lebo správna odpoveď je nutne `NIE`. Nech by sme sa totiž pýtali ľubovoľne, položíme nanajvýš 28 otázok a každou z nich vylúčime len jedinú možnosť. Vďaka tomuto pozorovaniu vieme teda náš algoritmus zrýchliť a dokonca ho implementovať s konštantnou pomocnou pamäťou.

1825. Inšpirujeme sa riešením úlohy 1815: k pôvodnej jaskyni s n komôrkami zostrojíme novú jaskyňu s 2^n komôrkami. Každá komôrka bude zodpovedať jednej z 2^n podmnožín množiny pôvodných komôrok.

Kam povedie v novej jaskyni chodbička označená písmenom x z komôrky zodpovedajúcej množine $A = \{a_1, \ldots, a_s\}$? Pozrieme sa na pôvodnú jaskyňu, pre každú z komôrok a_i nájdeme všetky chodbičky na písmeno x a vypíšeme si komôrky, kam vedú. Takto dostaneme novú množinu komôrok $B = \{b_1, \ldots, b_t\}$. V novej jaskyni spravíme z komôrky A chodbičku s písmenom x do komôrky B.

Pri tejto konštrukcii ľahko dokážeme, že pre ľubovoľný text t platí: komôrka novej jaskyne, do ktorej prídeme po prečítaní textu t, presne zodpovedá množine komôrok v pôvodnej jaskyni, do ktorých sa dalo dostať po prečítaní písmeniek textu t.

Poznámka: Existujú staré jaskyne s n komôrkami, pre ktoré je skutočne potrebné mať v novej jaskyni až 2^n komôrok. V praxi sa však často ukáže, že sa do veľa z 2^n komôrok zostrojených našim algoritmom vôbec nedá v novej jaskyni dostať. Vyššie popísanú konštrukciu však vieme vylepšiť – jednoducho budeme novú jaskyňu akoby prehľadávať (napríklad do šírky) a pri tom postupne zostrojovať množiny zodpovedajúce všetkým naozaj dosiahnuteľným komôrkam.

1831. Pesničky, ktoré ručne nepresunieme, ostanú v pôvodnom poradí, a teda už na začiatku museli tvoriť rastúcu podpostupnosť. A naopak, keď si zvolíme nejakú rastúcu podpostupnosť pesničiek, tak zjavne ostatné pesničky vieme postupne po jednej do nej na správne miesta povkladať. Keďže chceme presunúť čo najmenej pesničiek, treba teda nájsť najdlhšiu rastúcu podpostupnosť v danej postupnosti.

Existuje všeobecný algoritmus, ktorý takúto podpostupnosť nájde v čase $O(n \log n)$. Založený je na myšlienke, že postupne spracúvame prvky a pre každú dĺžku k si pamätáme, akým najmenším prvkom môže končiť rastúca podpostupnosť dĺžky k vybraná z už spracovaných prvkov.

Alternatívne môžeme využiť, že vstupná postupnosť je permutácia. Pre každý už spracovaný prvok si budeme pamätať najväčšiu dĺžku postupnosti, ktorá ním končí. Pri spracovaní nového prvku použijeme intervalový strom na zistenie maxima z dĺžok postupností končiacich už spracovanými prvkami menšími od toho aktuálneho.

1832. *Riešenie nadväzuje na text „Medián postupnosti“ na str. 243.*
Všimnime si, že pokiaľ zoberieme čo najviac maškrty, ktorá má najvyššiu jednotkovú cenu, tak tým nič nepokazíme. Takže priamočiare riešenie by bolo usporiadať veci podľa jednotkovej ceny a brať z tých najlepších. Takto dosiahneme riešenie bežiace v čase $O(n \log n)$.

Rýchlejšie riešenie využíva podobný postup ako pri hľadaní k-teho najmenšieho prvku. V čase $O(n)$ rozdelíme maškrty podľa jednotkovej ceny na lacnejšiu a drahšiu polovicu. Potom sa pozrieme, akú hmotnosť majú spolu tie drahšie. Ak je menšia od požadovanej, zoberieme ich všetky a zostane nám problém polovičnej veľkosti: spomedzi lacnejšej polovice vybrať maškrty dávajúce dokopy chýbajúcu hmotnosť. Naopak, ak je celková hmotnosť drahších maškŕt dostačujúca, môžeme zahodiť celú lacnejšiu polovicu, čím opäť dostávame problém polovičnej veľkosti.

Časová zložitosť tohto riešenia je $O(n)$, lebo rozdelenie maškŕt na polovice trvá lineárny čas od ich aktuálneho počtu a v každom kroku sa zbavíme približne polovice maškŕt.

1833. Jedno možné riešenie: Zoberieme najľavejší zo zadaných bodov (ak je takých viac, tak najspodnejší z nich) a označíme ho x_0. Ostatné body usporiadame polárne okolo x_0, v prípade rovnosti uhlu použijeme ako druhé kritérium rastúcu vzdialenosť[7] od x_0. Výsledné poradie označme $x_1, \ldots, x_{n-1}$.

[7] Ešte lepšie je porovnávať štvorec vzdialenosti. Vedie to k tomu istému usporiadaniu. Výhodou je, že ak sú súradnice daných bodov celé čísla, tak vieme v celých číslach implementovať celú porovnávaciu funkciu.

Ľahko overíme, že ak body pospájame v tomto poradí, dostaneme korektný mnohouholník, ktorý má všetky zadané body na obvode. Jedinou výnimkou je situácia, keď bodom x_{n-2} a x_{n-1} zodpovedá ten istý uhol – vtedy z x_{n-1} nevidíme na x_0. V takomto prípade pridáme ešte nový bod x_n, ktorý bude viditeľný z x_0 aj x_{n-1}. Zväčša stačí umiestniť x_n nadol od x_0. Samostatne treba ošetriť len situáciu, keď všetky body ležia na zvislej priamke.

Obsah zostrojeného mnohouholníka vypočítame použitím vektorového súčinu.

1834. Najľahšie polynomiálne riešenie: Môžeme použiť dynamické programovanie na zodpovedanie všetkých otázok tvaru: „Ako najlepšie viem vybrať x útočníkov, y hráčov v poli a z obrancov spomedzi prvých w hráčov na vstupe?“

Lepšie riešenia sú založené na postupnom „pažravom“ vylepšovaní priradenia hráčov pozíciám. Všetky tieto riešenia fungujú vďaka tomu, že táto úloha je len veľmi špeciálnym prípadom úlohy o najlacnejšom maximálnom toku. Uvažujme graf tvorený nasledujúcimi vrcholmi: jeden zdroj s a jeden odtok t, pre každého futbalistu x jeden vrchol f_x a tri vrcholy p_u, p_p, p_o predstavujúce jednotlivé pozície. Hrany bude mať nasledovné: zo zdroja do každého f_x hrana s kapacitou 1 a cenou 0, z p_u do odtoku s kapacitou u a cenou 0, analogicky p_p a p_o a z každého f_x do každého p_y s kapacitou 1 a cenou tým nižšou, čím vyššie je ohodnotenie hráča x na pozícii y. Presnejšie, ak označíme $Z[x, y]$ zisk z umiestnenia hráča x na pozíciu y a ak M je maximum z hodnôt $Z[x, y]$, tak hrana z f_x do p_y bude mať cenu $M - Z[x, y]$.

Ľahko nahliadneme, že každý maximálny tok v tomto grafe má veľkosť $u + p + o$. Celočíselné maximálne toky zodpovedajú možným priradeniam futbalistov na pozície. Ak sa teraz pozrieme na cenu konkrétneho maximálneho toku, tá je tým nižšia, čím je súčet ziskov z vybraných hráčov vyšší. Tok, ktorý zodpovedá optimálnemu priradeniu, je teda tokom s minimálnou cenou. A platí tvrdenie, že maximálny tok s minimálnou cenou vieme nájsť tak, že začneme s prázdnym tokom a dokola opakujeme: nájdeme najlacnejšiu zlepšujúcu cestu a ňou tok zlepšíme. Keďže všetky naše kapacity hrán sú celočíselné, budú celočíselné aj všetky toky, ktoré počas riešenia zostrojíme.

V našom špeciálnom prípade vieme zlepšujúce cesty hľadať veľmi efektívne. Keď si povieme, cez ktoré z vrcholov p_u, p_p a p_o táto cesta pôjde a v akom poradí (čo je len konečne veľa možností), je tým už dotyčná cesta jednoznačne určená. Napríklad ak hľadáme zlepšujúcu cestu cez p_u a p_p, bude to vyzerať nasledovne: Spomedzi všetkých futbalistov, ktorí momentálne nie sú priradení nikam, vyberieme jedného, ktorého dáme na pozíciu útočníka, a spomedzi tých, ktorí sú na pozícii útočníka, vyberieme jedného, ktorého preradíme na stredopoliara. Pri oboch výberoch samozrejme platí, že ak máme viac možností, vyberieme tú pre nás najlepšiu.

Aby sme vedeli rýchlo hľadať všetkých potrebných hráčov, budeme mať pre každý typ zmeny (napríklad preradenie z útoku do obrany) jednu vhodnú dátovú štruktúru (napríklad vyvažovaný strom), v ktorej budú uložení všetci futbalisti, ktorých aktuálne máme priradených v tej skupine, odkiaľ chceme niekoho presunúť (napríklad všetci útočníci). V dátovej štruktúre budú kandidáti usporiadaní podľa toho, ako nám vykonanie dotyčnej zmeny zmení celkovú cenu priradenia. Vždy, keď futbalistu preradíme, zmažeme jeho príslušný záznam v každej dátovej štruktúre a potom vložíme do správnych dátových štruktúr jeho nové záznamy.

Takto vieme v každom kroku nájsť najlacnejšiu zlepšujúcu cestu v čase $O(\log n)$. A keďže zlepšovať budeme nanajvýš n-krát, má toto riešenie časovú zložitosť $O(n \log n)$.

Iné riešenie: Úlohu tiež môžeme sformulovať ako tzv. lineárny program: pre každého súťažiaceho budeme mať tri premenné, zodpovedajúce jeho priradeniu na jednotlivé posty. Pre každého súťažiaceho a každú pozíciu následne dostávame jednu lineárnu nerovnosť pre tieto premenné. Na riešenie takýchto lineárnych programov existujú algoritmy s lineárnou časovou zložitosťou.

1835. Podúloha a) má triviálne riešenie so 6 komôrkami, v ktorom aktuálna komôrka zodpovedá zvyšku, ktorý dáva doteraz prečítaná časť čísla po delení 6. Existuje však aj lepšie riešenie – ak nie je zvyšok po delení 3 nula, nezaujíma nás zvyšok po delení 2. Takže potrebujeme len 4 komôrky, predstavujúce čísla tvarov $6k$, $6k+3$, $3k+1$ a $3k+2$.

V podúlohe b) bude nová jaskyňa vyzerať nasledovne: nové komôrky budú zodpovedať všetkým dvojiciam (a,b), kde a je komôrka z prvej a b komôrka z druhej jaskyne. Z každej novej komôrky povedú na každé písmeno x nasledujúce chodbičky: ak sa v prvej jaskyni dalo z a na písmeno x dostať do a', tak spravíme chodbičku označenú x z (a,b) do (a',b). Analogicky pridáme chodbičky na x z (a,b) do všetkých (a,b') takých, že sa v druhej jaskyni dalo na x dostať z b do b'. Keď zlatokop prechádza takouto jaskyňou, tak sa vlastne o každom písmenku, ktoré prečíta, môže rozhodnúť, či patrí do reťazca patriaceho k prvej alebo k druhej jaskyni a podľa toho sa pohnúť.

1841. Existuje veľmi veľa riešení, lebo v podstate skoro každé náhodné rozmiestnenie vrcholov grafu do priestoru spĺňa požiadavky zo zadania. Jedno elegantné riešenie nevyužívajúce náhodu je umiestniť vrcholy do bodov $[t,t^2,t^3]$ pre t od 1 po n.

1842. Ukážeme riešenie v čase $O(n^2 \log n)$, konštantnej pamäti a s konštantným počtom naraz používaných pomocných súborov. V riešení predpokladáme, že n je mocnina dvoch (inak si do matice vhodne doplníme riadky a stĺpce tak, aby jej rozmer bol mocninou dvoch, transponujeme ju a pridané čísla odstránime).

Začneme tým, že budeme dokola opakovať nasledujúci proces: maticu akoby zvislo rozstrihneme na dve, a pravú polovicu presunieme pod ľavú. Jedno vykonanie tohto procesu vieme vykonať nasledovne: Otvoríme na čítanie súbor obsahujúci aktuálnu maticu a otvoríme na zápis nový súbor. Čítame aktuálnu maticu a z každého riadku jeho prvú polovicu skopírujeme do výstupného súboru. Keď sa nám súbor s aktuálnou maticou minie, zavrieme ho, otvoríme ešte raz a celý proces zopakujeme – len tentokrát budeme zapisovať druhú polovicu každého riadku.

Po $\log n$ opakovaniach tohto procesu dostaneme súbor s n^2 riadkami, pričom v každom je jedno číslo. Tento súbor ľahko prerobíme na nový, v ktorom sú tieto čísla (v tom istom poradí) všetky v jednom riadku.

Teraz si už len stačí uvedomiť, že keď ďalších $\log n$-krát zopakujeme vyššie popísaný proces, dostaneme presne to, čo potrebujeme – transpozíciu pôvodnej matice.

1843. Pomer $a:b:c$ je rovnaký ako pomer $1:(b/a):(c/a)$; elixír s pomerom $a:b:c$ si teda môžeme predstaviť ako bod v rovine so súradnicami $[b/a, c/a]$. Takto dostaneme pre zadané elixíry body $A_1,\ldots,A_n$ a pre želaný elixír bod B.

Ľahko nahliadneme, že z danej množiny elixírov vieme namiešať práve tie elixíry, ktorým zodpovedajú body v konvexnom obale bodov $A_1,\ldots,A_n$. Potrebujeme teda overiť, či B leží v tomto konvexnom obale. Jednou možnosťou je tento konvexný obal v čase $O(n \log n)$ zostrojiť, ide to však aj lepšie. Postupne budeme spracúvať body A_i a budeme si udržiavať dva indexy p,q také, že všetky doteraz spracované body ležia v uhle $\measuredangle A_pBA_q < 180°$. Ak sa nám podarí spracovať všetky body, vieme, že B leží mimo konvexného obalu. Naopak, ak pre nejaký bod A_r už nevieme túto podmienku splniť, tak sme práve našli trojuholník $A_pA_qA_r$, v ktorom bod B leží.

1844. Asi najjednoduchšie riešenie prebieha v dvoch fázach. V prvej fáze si pre každú dvojicu miest i,j zistíme cenu $d_{i,j}$ najlacnejšieho lístku, ktorý vieme kúpiť v i na priamu cestu do j. Inými slovami, $d_{i,j}$ je minimum z hodnôt $a_{i,j}$ až $a_{i,n}$, kde $a_{i,j}$ je zadaná cena lístku z i do j. Hodnoty $d_{i,j}$ vieme efektívne spočítať pomocou vzťahov $d_{i,n} = a_{i,n}$ a $d_{i,j} = \min(a_{i,j}, d_{i,j+1})$ pre $j < n$. Kvôli generovaniu výstupu si ku každej hodnote $d_{i,j}$ navyše uložíme aj číslo mesta, v ktorom lístok dotyčnej ceny končí.

V druhej fáze postupne pre každé mesto i spočítame najmenšiu cenu m_i, za ktorú sa doň vieme dostať, a taktiež posledný lístok (a_i,b_i), ktorý revízorovi ukážeme. Hodnotu m_i

spočítame jednoducho tak, že za a_i vyskúšame všetky hodnoty od 1 po $i-1$. Pre každé a_i vieme optimálnu cenu m_{a_i} cesty z 1 doň, aj optimálnu cenu $d_{a_i,i}$ lístku z neho do mesta i. Takto dostávame riešenie s optimálnou časovou zložitosťou $O(n^2)$.

1845. Ak by bola rumová komora práve jedna, stačilo by obrátiť smer všetkých chodbičiek a vymeniť rumovú komoru so začiatočnou.

Ak bolo rumových komôr viac, potrebujeme navyše špeciálne ošetriť začiatok novej jaskyne, aby bolo ako keby možné začať v hociktorej komore, ktorá bola v pôvodnej jaskyni rumová. Toto dosiahneme pridaním novej začiatočnej komory (do ktorej sa už nikdy nebude dať vrátiť) a vhodným navrhnutím chodbičiek z nej.

z1811. Dátová štruktúra, ktorá podporuje operácie „vlož na koniec radu“ a „vyber zo začiatku radu“, sa nazýva fronta. Najjednoduchšie je frontu reprezentovať poľom s tým, že si pamätáme miesto, kde fronta začína a miesto, kde končí. Namiesto toho, aby sme pri odchode klienta posúvali všetkých ľudí dopredu stačí posunúť miesto, kde sa v poli fronta začína. Ak nám index prekročí koniec poľa, pokračujeme cyklicky od začiatku. Takto vieme každú operáciu implementovať v konštantnom čase.

Iná možnosť reprezentácie fronty je pomocou spájaného zoznamu. Opäť dosiahneme konštantnú časovú zložitosť jednej operácie.

z1812. Spravíme si maticu susednosti A – t.j. dvojrozmerné pole, kde $A[i,j]=1$, ak linka medzi mestami i a j premáva a $A[i,j]=0$, ak nepremáva. Na začiatku dáme do A samé nuly a začneme spracovávať linky. Keď načítame linku medzi x a y, nastavíme hodnoty $A[x,y]=A[y,x]=1$. Potom sa stačí pozrieť, v ktorom riadku je najviac jednotiek. Čas je v tomto prípade $O(\ell+n^2)$ a pamäť $O(n^2)$.

Iné riešenie je najskôr linky usporiadať (napríklad najprv podľa nižšieho čísla prístavu a potom podľa vyššieho). Následne vieme jediným prechodom vyhádzať duplikáty (rovnaké linky budú nasledovať po sebe), pre každý prístav spočítať počet liniek a vybrať maximum. Výsledný čas je $O(\ell \log \ell + n)$ a pamäť $O(\ell+n)$, čo je výhodné, ak liniek nie je veľa.

z1813. Prevod z mínus dvojkovej sústavy do desiatkovej je ľahký: cifry spracúvame od konca, pričom si pamätáme mocninu -2 zodpovedajúcu aktuálnej cifre. Vždy len cifru vynásobíme aktuálnou mocninou, pripočítame ju k výsledku, následne aktuálnu mocninu vynásobíme -2 a ideme na ďalšiu cifru.

Opačným smerom môžeme napríklad postupne určovať cifry výsledku od najmenej významnej. Najmenej významnú cifru zápisu n v sústave so základom -2 určíme ako $(n \bmod 2)$. Následne túto cifru odčítame od n, vydelíme ho -2 a celý postup opakujeme, až kým nám po konečnom počte krokov neostane $n=0$.

Formálne: Nech má dané číslo n v sústave so základom -2 zápis $\overline{x_{m-1}\ldots x_1 x_0}_{(-2)}$. Toto môžeme prepísať nasledovne: $n=(-2)\cdot\overline{x_{m-1}\ldots x_1}_{(-2)}+x_0$. Prvý sčítanec je zjavne párny, a keďže $x_0\in\{0,1\}$, musí nutne platiť, že $x_0=n \bmod 2$. Následne vidíme, že $\overline{x_{m-1}\ldots x_1}_{(-2)}=(n-x_0)/(-2)$, ostatné cifry teda určíme tak, že nájdeme zápis čísla $(n-x_0)/(-2)$ v sústave so základom -2.

z1814. Ku každému študentovi si zapamätáme, že sme ho ešte nespracovali. Následne postupne prechádzame študentov. Ak sme daného študenta ešte nespracovali, tak postupne označíme celý jeho kruh. To spravíme tak, že si daného študenta zapamätáme ako začiatok kruhu a potom ideme postupne po kruhu, až kým ho celý neprejdeme. Následne zvýšime počet kruhov o 1.

Počas riešenia každého študenta práve raz spracujeme, a to v konštantnom čase. Preto má toto riešenie časovú aj pamäťovú zložitosť $O(n)$.

z1815. Na hviezdu sa môžeme pozerať ako na graf. Vrcholy grafu nebudú len vrcholy hviezdy, ale aj všetky priesečníky čiar. Vidíme, že pre každé n platí, že všetky vrcholy sú párneho stupňa a graf je súvislý. Takýto obrázok sa vždy dá nakresliť jedným ťahom.

Musíme teda hľadať riešenie, ktoré celú hviezdu nakreslí jedným ťahom, a to bez toho, aby Zeofína išla po niektorom úseku čiary dvakrát.

Očíslujme si vrcholy hviezdy od 0 po $n-1$. Pre nepárne n je riešenie ľahké: Ak $n = 2k+1$, tak čísla n a k sú nesúdeliteľné. Ak teda Zeofína začne vo vrchole 0 a v každom kroku prejde do vrcholu s číslom o k vyšším (modulo n), do vrcholu 0 sa vráti až po tom, ako navštívi každý iný vrchol, a teda nakreslí celú hviezdu.

Ten istý postup zafunguje aj pre $n = 4k$, lebo aj v tejto situácií je n nesúdeliteľné s počtom vrcholov, o ktoré sa Zeofína pohne v každom kroku $(2k - 1)$.

Najťažší prípad je $n = 4k + 2$. Vtedy by sa Zeofína mala v každom kroku pohnúť o $2k$ vrcholov ďalej. Tieto dve čísla majú však najväčšieho spoločného deliteľa 2, preto Zeofína pri takomto postupe navštívi len každý druhý vrchol. (Všimnite si v zadaní obrázok pre $n = 6$. Ak by sme sa ten snažili kresliť len čiarami „z vrcholu do vrcholu“, nepodarí sa nám to, lebo sa skladá z dvoch „samostatných“ trojuholníkov.)

Tento prípad vieme ošetriť napríklad nasledovne: začneme vo vrchole 0 a kreslíme čiary rovnako ako v predchádzajúcich prípadoch. Tým by sme nakreslili hviezdu s polovičným počtom vrcholov – len tými s párnymi číslami. My však neskončíme späť vo vrchole 0, ale sa zastavíme o kúsok skôr – na poslednom priesečníku, cez ktorý ideme pred návratom do vrcholu 0. Tam sa „prepneme“ na kreslenie hviezdy na vrcholoch s nepárnymi číslami. Keď tú celú nakreslíme, nutne skončíme opäť na tomto istom mieste. Odtiaľ už potom len nakreslíme posledný kúsok čiary do vrcholu 0.

z1821. Pre každú kôpku kníh si budeme udržiavať jeden zásobník, v ktorom budeme vykonávať operácie podľa požiadaviek knihovníka. Na záver postupne vyprázdnime zásobník a tak vypíšeme všetky knihy, ktoré v ňom na konci dňa ostali. Časová aj pamäťová zložitosť je lineárna od veľkosti vstupu.

z1822. Obyvatelia chotára spolu s dvojicami určujúcimi dôveru medzi nimi tvoria orientovaný graf. Prehľadávaním z vrcholu 1 nájdeme vrcholy, kam sa klebeta rozšíri.

z1823. Od daného čísla budeme postupne odčítavať hodnoty 1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4 a 1 (v tomto poradí), kým neklesneme na nulu. Pri každom odčítaní vieme, aký znak, resp. dvojicu znakov treba vytesať a koľko úderov na to treba.

z1824. Činnosť zariadenia sa dá v čase $O(kn)$ odsimulovať. Všimnime si však, že skoky určené vstupom definujú cykly, v ktorých blchy skáču. Blcha, ktorá sa nachádza v cykle s dĺžkou ℓ sa vráti na svoje miesto každý ℓ-tý úder bubna a preto stačí určiť, kde sa bude nachádzať po $k \bmod \ell$ úderoch bubna. V našom riešení teda najskôr nájdeme všetky cykly, pre každý cyklus i nájdeme jeho dĺžku ℓ_i a následne pre každú blchu na ňom odpočítame, kde sa bude nachádzať po $k \bmod \ell_i$ úderoch bubna.

Aby sme dosiahli optimálnu časovú zložitosť, je potrebné každý cyklus spracovať len raz a pri spracúvaní cyklu každú blchu umiestniť na jej nové miesto v konštantnom čase. Najjednoduchší spôsob, ako to spraviť, je vykopírovať si do pomocného poľa čísla pozícií v poradí, v akom cez ne cyklus prechádza. Z takto vytvoreného poľa potom pre každú blchu ľahko určíme, kde sa bude nachádzať po poslednom údere bubna. Časová aj pamäťová zložitosť tohto riešenia je lineárna od počtu bĺch.

z1825. Treba si uvedomiť, že pre každé n sa Sierpińského trojuholník rádu n dá celý nakresliť jedným ťahom (pričom po každej čiare prejdeme iba raz) – optimálny program teda musí vypísať takýto postup kreslenia.

Jedno riešenie: Najskôr nakreslíme vonkajší trojuholník tak, aby sme začali aj skončili uprostred jeho spodnej strany. Vnútro, ktoré ešte potrebujeme nakresliť, má jednoduchú rekurzívnu štruktúru: vnútro rádu n nakreslíme tak, že trikrát zopakujeme postup: „nakresli pol strany stredného trojuholníka, nakresli vnútro rádu $n - 1$, nakresli pol strany stredného trojuholníka, otoč sa o 120 stupňov“.

z1831. Použijeme haldu. Halda je úplný binárny strom; v každom vrchole si pamätáme meno a hlasitosť nejakého pacienta, pričom pre každý vrchol (okrem listov) platí, že pacient v ňom kričí hlasnejšie ako pacienti uložení v synoch vrcholu. Tým pádom pacient, ktorý kričí najhlasnejšie, je v koreni. Výška tohto stromu je $\lfloor \log_2 n \rfloor + 1$ a vloženie aj výber najhlasnejšieho pacienta (tak, aby sme zachovali štruktúru haldy) vieme implementovať v čase $O(\log n)$.

z1832. *Riešenie je uvedené v texte „Dijkstrov algoritmus“ na str. 248.*

z1833. Na riešenie úlohy použijeme metódu dynamického programovania. Mince si očíslujeme od 1 po j. Postupne vyplníme dvojrozmernú tabuľku obsahujúcu hodnoty $T[i, j]$ – najväčší počet mincí spomedzi mincí s číslom 1 až j, ktorými vieme zaplatiť sumu i. Všetky hodnoty $T[x, 0]$ inicializujeme na $-\infty$, až na $T[0, 0] = 0$. Postupne pridávame mince, teda zvyšujeme j od 1 po n, a počítame, aké sumy už vieme zaplatiť a koľko najviac mincí na to použiť. Platí, že sumu s vieme pomocou mincí s číslom 1 až k zaplatiť maximálne s $T[s, k] = \max\{T[s, k-1], T[s - a_k, k-1] + 1\}$ mincami – totiž buď k-tu mincu nepoužijeme, alebo áno. Časová zložitosť tohto riešenia je $O(ns)$. Kedže si netreba pamätať celé pole T, ale len hodnoty pre poslednú pridávanú mincu, pamäťová zložitosť je $O(n + s)$.

z1834. V tejto úlohe potrebujeme vypísať všetky permutácie v ich lexikografickom poradí. Na výpis permutácií použijeme rekurziu, ktorá nám bude generovať postupne každú permutáciu zľava doprava. V rekurzii udržujeme dve jednorozmerné polia – pole s permutáciou, ktorú generujeme a pole, kde si označujeme čísla, ktoré už sú umiestnené v generovanej permutácii. Na začiatku je prvé pole prázdne a v druhom žiadne číslo nie je označené, kedže ešte nič nebolo umiestnené. Rekurzívna funkcia má jeden parameter – pozíciu zľava, ktorú ideme generovať. Na začiatku voláme funkciu s parametrom 1. Pokiaľ rekurzívnu funkciu zavoláme s parametrom $i > n$, tak sme už umiestnili všetky cifry do permutácie a môžeme ju (prvé pole) vypísať. Ak $i \leq n$, vyskúšame umiestniť do permutácie (prvého poľa) na i-tu pozíciu všetky neumiestnené cifry a pre každú možnosť sa rekurzívne zavoláme pre pozíciu $i + 1$. Pamäťová zložitosť je $O(n)$, časová je $O(n \cdot n!)$.

Iné riešenie: Aby sme permutáciu p prerobili na tú, ktorá nasleduje v lexikografickom poradí bezprostredne po nej, stačí nájsť (prechodom od konca) prvý index i taký, že $P[i] < P[i + 1]$. Potom $P[i]$ vymeníme s najbližšou väčšou hodnotou spomedzi $P[i + 1]$ až $P[n]$ a následne prvky $P[i + 1]$ až $P[n]$ usporiadame vzostupne. Keďže sú momentálne usporiadané zostupne, nepotrebujeme implementovať triedenie, stačí príslušný úsek poľa obrátiť. Časová zložitosť je rovnaká ako v prvom riešení.

z1835. Popísaný n-ročný strom kreslíme rekurzívne: najskôr nakreslíme kmeň, potom sa otočíme o $180 - \alpha$ stupňov doľava, nakreslíme $(n-1)$-ročný strom s dĺžkami vetvičiek preškálovanými s koeficientom a, potom sa otočíme doprava o $\alpha + \beta - 180$ stupňov (t.j. najskôr o $180 - \alpha$ stupňov doľava a potom o β stupňov doprava), nakreslíme $(n-1)$-ročný strom s dĺžkami vetvičiek preškálovanými s koeficientom b a na záver sa otočíme o $180 - \beta$ stupňov doprava a vrátime sa po kmeni späť na začiatok.

Popísaný rekurzívny algoritmus sa dá ešte zoptimalizovať, aby sa toľko Zeofína nenachodila. Pravý a ľavý konár vytvárajú trojuholník, preto si stačí vypočítať pochodový uhol a dĺžku úseku, ktorý je v trojuholníku medzi týmito konármi. Zeofína najskôr prejde po pravom konári, nakreslí pravý $(n-1)$-ročný stromu avšak potom sa nevydá späť po konári. Namiesto toho prejde bez kreslenia do začiatku ľavého $(n-1)$-ročného stromu (využijúc pochodový uhol a dĺžku vypočítanú z trojuholníka), ktorý nakreslí a potom sa vráti späť do začiatku po ľavom konári, ktorý takto aj nakreslí.

z1841. Na riešenie tejto úlohy použijeme metódu zvanú zametanie. Pre každý interval i, j si vytvoríme 2 udalosti – udalosť začiatku pre čas i a udalosť konca pre čas j. Vytvorené udalosti usporiadame a následne prejdeme od najskoršej po najneskoršiu. Počas prechádzania si aktualizujeme počet otvorených intervalov, ktoré sa ešte nezavreli. V mo-

mente, keď sa zavrie posledný otvorený interval, spočítame celkovú dĺžku intervalov, ktoré boli takto pootvárané (preto si treba pamätať čas, kedy sa prvý interval otvoril). Časová zložitosť tohto riešenia je kvôli triedeniu $O(n \log n)$, pamäťová je $O(n)$.

z1842. *Riešenie je uvedené v texte „Prehľadávanie do hĺbky“ na str. 246.*

z1843. Dve čísla vo faktoriálovej sústave spočítame rovnako ako v klasickej desiatkovej sústave. Cifry každého čísla si zapíšeme (vo faktoriálovej sústave) do poľa. Postupne od najmenej významného rádu realizujeme sčítanie a prenos do vyššieho rádu. Časová aj pamäťová zložitosť tohto riešenia je lineárna od veľkosti čísla.

z1844. Ak je $n \leq 3$, úloha nemusí byť riešiteľná. Inak úlohu riešime v kvadratickom čase od n: Najskôr presunieme najchrumkavejší bochník na začiatok. Potom presunieme druhý najchrumkavejší bochník za najchrumkavejší, atď. Na záver sa môže stať, že posledný a predposledný bochník je vymenený. V tomto prípade použijeme na posledných 4 bochníkoch pár prehodení 3 bochníkov, aby sme toto poradie napravili.

Dá sa dokázať, že rádovo lepšie riešenie od nášho nemôže existovať: totiž žiadne prehodenie nezníži počet inverzií o viac ako tri, a teda napríklad na usporiadanie bochníkov, ktoré sú na začiatku v úplne obrátenom poradí, budeme určite potrebovať aspoň kvadraticky veľa prehodení.

z1845. Počas načítania odsimulujeme pohyb Zeofíny a potom spočítame vzdialenosť koncového bodu $[x, y]$ do začiatku $[0, 0]$ spolu s pochodovým uhlom. Vzdialenosť spočítame ako $\sqrt{x^2 + y^2}$. Uhol, ktorý zviera polpriamka idúca z bodu $[0, 0]$ do bodu $[x, y]$ s kladnou poloosou osi x vypočítame pomocou znamienok čísel x, y a pre $x \neq 0$ aj z hodnoty $\operatorname{arctg} |y/x|$. Vo väčšine programovacích jazykov existuje funkcia `atan2(y,x)`, ktorá túto analýzu prípadov spraví za nás. Časová zložitosť tohto riešenia je lineárna od veľkosti vstupu, pamäťová zložitosť je konštantná.

1911. Na vstupe sme dostali krajiny usporiadané podľa minimálneho počtu bodov, ktorý mohli získať. Zoznam usporiadaný podľa maximálneho počtu bodov vieme zostrojiť v lineárnom čase: stačí rozdeliť dané poradie krajín na 5 zoznamov podľa toho, koľko ich súťažiacich nepoznáme. Pre každý z týchto zoznamov platí, že krajiny v ňom sú už usporiadané aj podľa teoreticky dosiahnuteľného maxima, stačí nám teda týchto 5 zoznamov pospájať do jedného (podobne ako pri algoritme merge-sort).

Najlepšie umiestnenie krajiny teraz zistíme tak, že nájdeme jej maximálny počet bodov v zozname minimálnych počtov bodov všetkých krajín. Aby sme mali optimálnu (lineárnu) časovú zložitosť, budeme prechádzať naraz obe polia a pri tomto prechode zistíme odpoveď postupne pre všetky krajiny. Najhoršie umiestnenia krajín zistíme opačným postupom.

1912. Na začiatku si pre každý vrchol zapamätáme jeho pozíciu v in-order zápise. S využitím tejto informácie vieme napísať rekurzívnu procedúru, ktorá post-order zápis vyrobí v lineárnom čase. Myšlienka: prvý prvok v pre-order zápise stromu je koreň. Z predrátanej informácie vieme povedať, kde leží v in-order zápise, a tým in-order zápis stromu rozdeliť na in-order zápis ľavého podstromu, koreň a in-order zápis pravého podstromu. A keďže tým pádom vieme ich veľkosti, ľahko následne rozdelíme aj pre-order zápis na koreň, pre-order zápis ľavého a pre-order zápis pravého podstromu. Teraz sa dvakrát rekurzívne zavoláme na príslušné úseky pre-order a in-order zápisu a následne vypíšeme hodnotu v koreni.

1913. Platí nasledujúce tvrdenie: Ak máme m modrých, z zelených, c červených a r ružových zápaliek, tak mnohouholník sa dá z nich postaviť práve vtedy, keď m, z, c a r sú párne a aspoň dve z nich sú nenulové.

Jednu implikáciu dokážeme tak, že navrhneme konkrétny spôsob, ako zápalky ukladať. Pri druhej sa treba pozrieť na lomenú čiaru začínajúcu v $[0, 0]$ a obsahujúcu m, z, c a r zápaliek jednotlivých typov a vyjadriť si možné súradnice jej konca. Ak chceme vytvoriť mnohouholník, potrebujeme dosiahnuť, aby aj tieto súradnice boli $[0, 0]$. Ukáže sa, že to sa dá len vtedy, ak budú splnené vyššie uvedené podmienky.

1914. Vzorové riešenie je založené na dynamickom programovaní, pomocou ktorého vieme pre všetky sumy od 1 po nejaké x spočítať aj optimálny, aj Kiribatský počet mincí potrebný na ich zaplatenie. Otázku, po aké x stačí overiť rovnosť, zodpovedá nasledujúce tvrdenie: Ak všetky sumy 1 až $a_k + a_{k-1} - 1$ zaplatia Kiribatčania optimálne, potom je testovaná sada mincí pre Kiribati vhodná.

(Pri dôkaze uvažujte najmenšiu sumu s, ktorú Kiribatčania nezaplatia optimálne, a ukážte, že existencia tejto sumy vedie k sporu. Ľahšie sa dokáže slabšie tvrdenie: sada mincí je vhodná, ak ňou optimálne zaplatia všetky sumy po $a_k a_{k-1}$.)

1915. Vzorové riešenie využíva dve čierne krabičky: jednu „vkladaciu“ a jednu „vyberaciu“. Prichádzajúce prvky vkladám do vkladacej krabičky. Keď mám vybrať prvok: ak mám nejaký vo vyberacej krabičke, tak ho rovno vyberiem, inak najskôr „preklopím“ obsah vkladacej krabičky do vyberacej. Rozmyslite si, že pri takomto postupe budem prvky vyberať v tom poradí, v ktorom ich vložím.

Celková časová zložitosť pre n operácií je $O(n)$, lebo spracujem nanajvýš n prvkov a každý z nich najviac dvakrát vložím do nejakej krabičky a najviac dvakrát ho z nejakej krabičky vyberiem.

1921. *Riešenie nadväzuje na text „Medián postupnosti“ na str. 243.*
Dvakrát použijeme algoritmus na nájdenie x-tého najmenšieho prvku postupnosti. Začneme tým, že nájdeme medián postupnosti. Zostáva nájsť $k-1$ ďalších prvkov, ktoré sú k nemu najbližšie. Pre každý prvok si spočítame, o koľko sa líši od mediánu. Aby sme vyriešili našu úlohu, potrebujeme nájsť tých k prvkov, pre ktoré sú tieto rozdiely najmenšie. To spravíme druhým použitím vyššie spomínaného algoritmu pre záznamy tvaru (vzdialenosť prvku od mediánu, pôvodný index prvku). Časová zložitosť tohto riešenia je $O(n)$.

1922. Označme $A[n,k]$ počet permutácií n prvkov, ktoré majú práve k inverzií. Každá permutácia n prvkov vznikne z nejakej permutácie $n-1$ prvkov tak, že na jednu z n možných pozícií vložíme číslo n. Keď vložíme n tak, že za ním necháme i prvkov pôvodnej permutácie, vyrobíme tým i nových inverzií. Ak teda má nová permutácia mať k inverzií, tá pôvodná ich musela mať $k-i$.

Odtiaľ dostávame rekurentný vzťah: $A[n,k] = \sum_{i=0}^{n-1} A[n-1, k-i]$.

Keď teraz porovnáme vzťahy pre $A[n,k]$ a $A[n,k-1]$, zistíme, že sa skoro rovnajú. Ich odčítaním od seba po úprave dostávame: $A[n,k] = A[n,k-1] + A[n-1,k] - A[n-1,k-n]$. Podľa tohto vzťahu vieme hodnotu $A[n,k]$ spočítať v čase $O(kn)$ a pamäti $O(k)$.

1923. Jednoduché riešenie je spočítať všetky vzdialenosti a vybrať najväčšiu v čase $O(dn^2)$. Ak je však d oveľa menšie ako n, existuje lepšie riešenie:

Chceme nájsť maximum výrazu $|x_1 - y_1| + |x_2 - y_2| + \cdots + |x_d - y_d|$ cez všetky možné výbery x a y. Myšlienku riešenia vysvetlíme na príklade pre $d = 3$. Predpokladajme, že nám dobrá víla poradila, že pre optimálnu dvojicu x, y platí $x_1 \geq y_1$, $x_2 < y_2$ a $x_3 \geq y_3$. Nájsť tieto x a y je už jednoduché: Výraz, ktorý maximalizujeme, môžeme zjednodušiť na $(x_1 - y_1) + (y_2 - x_2) + (x_3 - y_3) = (x_1 - x_2 + x_3) - (y_1 - y_2 + y_3)$. Stačí teda zvoliť za x tú zastávku z, pre ktorú výraz $z_1 - z_2 + z_3$ nadobúda maximum, a za y tú, pre ktorú uvedený výraz nadobúda minimum.

Keďže my ale dobrú vílu nemáme, postupne vyskúšame všetkých 2^d kombinácií $x_i \geq y_i$ a $x_i < y_i$. Pre každú z nich vieme v čase $O(nd)$ nájsť optimálnu dvojicu x, y a spomedzi týchto dvojíc vyberieme tú najlepšiu.[8] Časová zložitosť tohto riešenia je teda $O(nd \cdot 2^d)$.

Toto riešenie ešte vieme o trochu zlepšiť: ak budeme vyššie spomínaných 2^d kombinácií znamienok skúšať v poradí zodpovedajúcom Grayovmu kódu (viď úlohu z1713), budeme vedieť pre každý bod hodnotu nového výrazu spočítať z hodnoty starého výrazu v $O(1)$, a

[8] Technický detail: pre niektoré kombinácie znamienok sa nám môže stať, že zistíme, že pre ne optimálna dvojica bodov niektoré z nich nespĺňa. Toto nám ale neprekáža, lebo takéto riešenie určite nie je optimálne – pre správnu kombináciu znamienok dostaneme pre tie isté dva body väčšiu hodnotu.

teda každú z 2^d kombinácií budeme vedieť spracovať v čase $O(n)$. Toto vylepšené riešenie má teda časovú zložitosť $O(n \cdot 2^d)$.

1924. Uvažujme graf, ktorého vrcholy sú počítače a hrany sú káble medzi nimi. Počet sieťových kariet v počítači zodpovedá stupňu vrcholu. Úlohou je potom zistiť, či existuje súvislý graf s danými stupňami vrcholov.

Vhodným prepájaním hrán sa dá dokázať, že akonáhle existuje *nejaký* graf s danými stupňami vrcholov a máme aspoň $n-1$ hrán (kde n je počet vrcholov), tak existuje aj *súvislý* graf s danými stupňami vrcholov.

Nech $n-1 \geq d_1 \geq \cdots \geq d_n \geq 0$ je daná postupnosť stupňov. Havlova veta hovorí, že graf s týmito stupňami vrcholov existuje práve vtedy, keď existuje ($n-1$ vrcholový) graf so stupňami $d_2-1, \cdots, d_{d_1+1}-1, d_{d_1+2}, \ldots, d_n$. Teda nič nepokazíme, ak vždy zoberieme vrchol najväčšieho stupňa a spojíme ho s toľkými ďalšími vrcholmi najväčšieho stupňa, aký má mať stupeň. Vhodnou implementáciou je možné overiť všetky podmienky a zostrojiť jeden zodpovedajúci graf v čase lineárnom od jeho veľkosti.

Ak graf nepotrebujeme rekonštruovať a stačí nám iba zistiť, či taký graf existuje, existuje rýchlejšie riešenie: Erdős a Gallai dokázali, že graf so stupňami $n-1 \geq d_1 \geq \cdots \geq d_n \geq 0$ existuje práve vtedy, keď $\sum_{i=1}^{n} d_i$ je párne a pre každé $1 \leq r \leq n-1$ platí $\sum_{i=1}^{r} d_i \leq r(r-1) + \sum_{i=r+1}^{n} \min\{r, d_i\}$. Tieto podmienky sa dajú overiť v čase $O(n)$.

1925. Použijeme 4 čierne krabičky: *Min*, *Max*, *DMin* a *DMax*. Keď máme vložiť číslo x, vložíme x do *Min* a tiež $-x$ do *Max*. Z *Min* teraz vieme efektívne vyberať najmenšie a z *Max* najväčšie číslo.

Pri výbere ale vzniká problém – keď vyberiem číslo z *Min*, mal by som ho vybrať aj z *Max* a naopak. To ale neviem v danom okamihu spraviť. Preto si to „odložím na neskôr“: namiesto vymazania x z *Min*, resp. $-x$ z *Max*, vložím x do *DMin*, resp. $-x$ do *DMax*. V každom okamihu teda platí, že *DMin* obsahuje tie prvky *Min*, ktoré v nej už nemajú byť. Kedykoľvek, keď sa na vrchu *Min* a *DMin* (resp. *Max* a *DMax*) zjaví tá istá hodnota, zmažeme ju z oboch krabičiek.

Celková časová zložitosť pre n operácií bude $O(n \log n)$, lebo pre každé z n čísel spravíme dokopy nanajvýš 4 volania *Insert* a nanajvýš 4 volania *ExtractMin*.

1931. *Riešenie nadväzuje na text „Medián postupnosti“ na str. 243.*
Spočítame $n-1$ vzdialeností medzi susednými dvojicami bodov. Spomedzi týchto $n-1$ úsekov môžeme nanajvýš $k-1$ nekúpiť. Z tohto pozorovania je už zjavné, že optimálnym riešením je kúpiť všetky úseky okrem $k-1$ najdlhších. Nájsť tieto úseky vieme v lineárnom čase použitím algoritmu na hľadanie x-tého najväčšieho prvku.

1932. Pre každý typ úseku budeme mať niekoľko premenných zodpovedajúcich kameňom tvoriacim tento úsek. Keďže zadané reťazce sa musia na každej pozícii zhodovať, dostávame pre každú pozíciu informáciu o rovnosti dvoch objektov (kde každý objekt je buď premenná alebo konštanta). Tieto informácie si zaznačíme v grafe. Ak niektorý komponent súvislosti tohto grafu obsahuje viac ako jednu konštantu, úloha nemá riešenie. Ak obsahuje práve jednu, máme pre všetky premenné v ňom jednoznačne určenú hodnotu. A ak neobsahuje žiadnu, máme f možností, akú hodnotu premenným v ňom priradiť. Ak teda nenájdeme žiaden spor, má úloha presne f^k riešení, kde k je počet komponentov neobsahujúcich konštantu.

Náš graf má veľkosť priamo úmernú dĺžke výslednej postupnosti, preto je aj časová a pamäťová zložitosť lineárna od dĺžky výslednej postupnosti.

1933. Budeme postupne spracúvať začiatky a konce výhybiek usporiadané podľa ich x-ovej súradnice. Počas toho si pre každú koľaj k budeme udržiavať minimálny počet $P[k]$ prehodení výhybky potrebný na to, aby sme sa na dotyčnú koľaj dostali (alebo ∞, ak sa na danú koľaj zatiaľ vôbec dostať nevieme).

Keď spracúvame začiatok výhybky v vedúcej z koľaje a na koľaj b, zapamätáme si najmenší počet prehodení $Q[v]$, na ktorý sa k nej vieme dostať. (Tento počet je rovný aktuálnej hodnote $P[a]$.) Ak bola výhybka v zapnutá, hodnotu $P[a]$ zväčšíme o 1 (lebo keď chceme pokračovať po koľaji a, musíme v vypnúť).

Keď budeme neskôr spracúvať koniec výhybky v, rozoberieme dve možnosti, ako pokračovať po koľaji b: Ak by sme ku koncu v prišli po koľaji b, potrebovali by sme $P[b]$ prehodení výhybky. Ak by sme prišli z koľaje a a využili v, potrebovali by sme $Q[v]$ alebo $Q[v]+1$ prehodení výhybky (záleží na pôvodnom stave v). Toto riešenie vieme implementovať v časovej zložitosti $O(k+n)$.

1934. Trojuholníky, ktoré nie sú jednofarebné, volajme pestré. Každý pestrý trojuholník má zjavne práve dva vrcholy, v ktorých sa stretá červená hrana s modrou, tieto vrcholy volajme zaujímavé.

Všimnime si ľubovoľný z našich n bodov. Nech z neho vedie x červených a teda $n-x-1$ modrých hrán. Potom má v tomto bode zaujímavý vrchol $x(n-x-1)$ pestrých trojuholníkov. Ak tieto hodnoty sčítame cez všetky vrcholy, dostaneme presne dvojnásobok počtu pestrých trojuholníkov, lebo každý pestrý trojuholník zarátame dvakrát.

Počet jednofarebných trojuholníkov vieme teraz určiť ako $\binom{n}{3}$ mínus počet pestrých. Toto riešenie má časovú zložitosť $O(n^2)$.

1935. V reči teórie grafov, potrebujeme efektívnu dátovú štruktúru, ktorá nám umožní pamätať si neorientovaný graf. Operácie, ktoré potrebujeme robiť, sú pridanie hrany, odobratie hrany, test na existenciu hrany a vymenovanie susedov vrcholu.

Za cenu toho, že budeme mať pamäťovú zložitosť $O(n^2)$ – túto pamäť ale nie je potrebné inicializovať – vieme všetky požiadavky realizovať v optimálnej časovej zložitosti. Použijeme naraz obe klasické reprezentácie grafu: Pre každý vrchol budeme mať spájaný zoznam s jeho susedmi, a taktiež budeme mať maticu susednosti, pričom ak x a y sú susedia, tak na súradniciach $[x,y]$ v tejto matici bude ukazovateľ na položku zodpovedajúcu vrcholu y v zozname pre vrchol x.

1941. *Riešenie nadväzuje na text „Medián postupnosti“ na str. 243.*
Zjavne Fero dostane za každý žetón jeho cenu a občas ešte niečo navyše, čo nazveme zisk. Nech si Fero zvolil nejakú postupnosť. Rozsekajme ju na rastúce úseky. Za každý rastúci úsek bude Ferov zisk rovný rozdielu posledného a prvého prvku.

Ak máme úsek dlhší ako 2, tak z neho môžeme vybrať všetky prvky okrem prvého a posledného a presunúť ich na koniec postupnosti, čím určite nezmenšíme zisk. Tiež sa neoplatí mať viac ako jeden úsek dĺžky 1, lebo z takýchto úsekov nie je zisk, čo vieme zlepšiť ich spojením. Optimálna postupnosť teda určite vyzerá tak, že všetky rastúce úseky majú dĺžku práve 2; s výnimkou jedného úseku dĺžky 1 alebo 3, ak je n nepárne.

Rozdeľme teraz prvky na $\lfloor n/2 \rfloor$ „malých“, $\lfloor n/2 \rfloor$ „veľkých“ a pre nepárne n jeden „stredný“. Tvrdíme, že ľubovoľné poradie, kde je každý malý prvok v dvojici s niektorým veľkým, je optimálne. Toto tvrdenie ľahko dokážeme pomocou postupného prehadzovania prvkov medzi dvojicami, ktoré toto nespĺňajú – taká úprava nám nemôže znížiť celkový zisk. Optimálny zisk aj jedno poradie, ktoré ho vyrobí, teda vieme zistiť v lineárnom čase tak, že nájdeme medián cien žetónov.

1942. Existuje viacero rôzne dobrých riešení:

Riešenie 1: Floydov-Warshallov algoritmus v čase $O(n^3)$.

Riešenie 2: Bellmanov-Fordov algoritmus v čase $O(mn)$.

Riešenie 3: Najkratšie cesty medzi každými dvoma vrcholmi v grafe s maticou susednosti A vieme zostrojiť vhodným „umocnením“ matice A na $n-1$. Toto „umocnenie“ vieme realizovať pomocou $O(\log n)$ „násobení“ matíc. (Na rozdiel od klasického násobenia, pri ktorom používame operácie krát a plus, tu použijeme operácie plus a minimum.) V súčasnosti

najlepší známy algoritmus na násobenie matíc je Coppersmithov-Winogradov algoritmus s časovou zložitosťou $O(n^{2.376})$. Takéto riešenie má teda časovú zložitosť $O(n^{2.376} \log n)$.

Riešenie 4: Existuje trik, pomocou ktorého sa dá časová zložitosť predchádzajúceho riešenia zlepšiť na $O(n^{2.376})$.

1943. Z Tálesovej vety vieme, že pravouhlý trojuholník sa tam nachádza práve vtedy, ak $n \geq 3$ a niektoré dva body ležia presne oproti sebe. To vieme zistiť v čase $O(n)$ použitím dvoch ukazovateľov. Na začiatku ich oba nastavíme na prvý bod zo vstupu. Následne dokola opakujeme: ak je uhol určený aktuálnymi dvoma bodmi menší ako $180°$, posunieme prvý, inak posunieme druhý ukazovateľ. Ak tam dvojica protiľahlých bodov je, takto ju určite nájdeme, ak nie, môžeme prestať, akonáhle oba ukazovatele obehnú celý kruh.

Druhú podúlohu riešime analogicky, len použijeme ukazovatele tri. Dokola opakujeme: nájdeme dvojicu ukazovateľov, medzi ktorými je uhol väčší ako $2\pi/3$, a zadný ukazovateľ o bod posunieme, aby sme tento uhol zmenšili.

1944. Dynamické programovanie: Postupne pre každú dvojicu (prefix prvej postupnosti, prefix druhej postupnosti) spočítame, či existuje reťazec, ktorý na oba pasuje, a ak áno, akú najkratšiu dĺžku môže mať.

1945. Pomocou danej čiernej krabičky vieme zostrojiť zelenú krabičku, ktorá o ľubovoľnom grafe rozhodne, či v ňom existuje cesta prechádzajúca cez každý vrchol práve raz. Zelená krabička funguje nasledovne: Ku grafu, ktorý dostane na vstupe, pridá toľko izolovaných vrcholov, koľko vrcholov ten graf už má, a potom sa opýta čiernej krabičky, či má upravený graf cestu cez aspoň polovicu vrcholov.

Pomocou zelenej krabičky vieme zostrojiť červenú krabičku, ktorá pre vstup (graf G, vrchol u, vrchol v) rozhodne, či v G existuje cesta, ktorá začína v u, končí vo v a prechádza cez každý vrchol práve raz. Funguje nasledovne: zoberie G, pridá doň dva nové vrcholy u' a v', pridá dve nové hrany uu' a vv' a na výsledný graf sa opýta zelenej krabičky.

A pomocou červenej krabičky už vieme vyriešiť zadanú úlohu: Ak je na vstupe graf G s 1 alebo 2 vrcholmi, je odpoveď jasná. Inak upravíme G nasledovne: pridáme nový vrchol $1'$ a spojíme ho s tými istými vrcholmi, s ktorými je spojený vrchol 1. Následne sa už len červenej krabičky opýtame, či v upravenom G existuje cesta z 1 do $1'$, ktorá prechádza cez každý vrchol práve raz.

z1911. *Riešenie nadväzuje na text „Prehľadávanie do šírky“ na str. 247.*
Retiazku si budeme reprezentovať ako neorientovaný graf. Očká, ktoré v optimálnom riešení *neodstránime*, tvoria najkratšiu cestu medzi koncovými očkami.

z1912. Keďže všetci trpaslíci majú pridržať latu, robí tak aj ten najvyšší a aj ten najnižší. Označme si ich výšky v_{max} a v_{min}. Ostatní trpaslíci majú byť medzi nimi rovnomerne rozmiestnení, preto výšky všetkých trpaslíkov musia tvoriť aritmetickú postupnosť. Stačí nám teda overiť, či sú ich výšky tvaru $k \cdot (v_{max} - v_{min})/(N-1) + v_{min}$, kde $0 \leq k \leq N-1$, a či sú všetky rôzne. To vieme zistiť tak, že si budeme v poli značiť, ktoré správne výšky sme už videli. Časová aj pamäťová zložitosť je lineárna od počtu trpaslíkov.

z1913. Uvedomte si, že číslo n zapísané v sústave so základom k končí aspoň i nulami práve vtedy, keď je deliteľné číslom k^i. Ak je k navyše zložené číslo ($p \cdot q = k, p \neq 1, q \neq 1$), tak n je deliteľné aj číslami p^i a q^i. Preto stačí najsilnejší kľúč hľadať len medzi prvočíslami, ktoré delia n. Výsledkom je teda to prvočíslo, ktoré sa v prvočíselnom rozvoji n vyskytuje s najväčším exponentom.

z1914. Čítame postupne usporiadané polohy dier zo vstupu. Vždy, keď načítaná diera ešte nie je zaplátaná, musíme použiť novú záplatu. Položíme ju čo najďalej od začiatku potrubia tak, aby ešte pokryla načítanú dieru (tá bude teda na okraji záplaty). V prípade, že načítaná diera už bola pokrytá predchádzajúcou záplatou, nerobíme nič.

Časová zložitosť tohto riešenia je lineárna od počtu dier, pamäťová je konštantná – stačí si pamätať, pokiaľ siaha posledná použitá záplata.

z1915. Text je zašifrovaný Cézarovou šifrou s posunom o 13 písmen (z `a` sa stalo `n`, z `b` sa stalo `o`, ...). Po rozšifrovaní dostaneme slovenskú a anglickú verziu básne s názvom *Taradúr* (v angličtine *Jabberwocky*). Báseň je dielom Lewisa Carrolla a je z knihy „Through the Looking-Glass and What Alice Found There, 1872“, ktorá bola vydaná po slovensky ako súčasť knihy „Alica v krajine zázrakov a Za zrkadlom, Mladé Letá 1981“.

z1921. Stačí postupne čítať zo vstupu dvojice susediacich parciel a rovno im priradzovať typ obývaná/krčma. Ak ešte ani jedna z práve spracúvanej dvojice parciel nemá priradený typ, spravíme z jednej krčmu a druhá bude obývaná. Ak práve jedna z dvojice spracúvaných parciel už má typ, druhej priradíme ten opačný (obývanej krčmu a naopak). Ak obe parcely už majú priradené typy, tak neurobíme nič. Parcely, ktoré nie sú v zozname, nesusedia s nikým, a teda nemôžu byť ani obývané a ani krčmy. Zjavne ku každej parcele p existuje nejaká parcela opačného typu – zabezpečíme to v okamihu, kedy parcelu p prvýkrát spracúvame. Časová zložitosť tohto riešenia je $O(m+n)$, pamäťová je $O(n)$.

Iné riešenie je použiť prehľadávanie do šírky. Vrcholy v párnej vzdialenosti od začiatku budú krčmy, vrcholy v nepárnej vzdialenosti budú obývané. Toto riešenie má rovnakú časovú, ale horšiu pamäťovú zložitosť.

z1922. Výšky trpaslíkov označme $A[1..n]$. Pre každú dvojicu po sebe idúcich trpaslíkov si spočítajme rozdiel ich výšok $B[i] = A[i+1] - A[i]$. Takto dostaneme novú postupnosť dĺžky $n-1$. Každému konvexnému úseku v postupnosti A zodpovedá o jedno kratší rastúci úsek v postupnosti B a naopak; podobne je to s konkávnymi úsekmi v A a klesajúcimi v B. Stačí teda v postupnosti B nájsť najdlhší rastúci a najdlhší klesajúci úsek, čo vieme ľahko spraviť v lineárnom čase a konštantnej pamäti.

z1923. Každé prirodzené číslo vieme jednoznačne zapísať v tvare $2^a 5^b c$, kde c je nesúdeliteľné s 2 a 5. Počet núl, ktorými toto číslo končí, vieme vypočítať ako $\min(a,b)$. Stačí nám teda vedieť zistiť exponenty prvočísel 2 a 5 v prvočíselnom rozklade $n!_3$. Ukážeme, ako spočítať exponent čísla 5, pre číslo 2 sa úloha rieši analogicky. (Dá sa dokázať, že pre každé n je exponent 2 v rozklade $n!_3$ aspoň taký veľký ako exponent 5, takže ho vlastne netreba počítať vôbec.)

Číslo $n!_3$ je súčinom niekoľkých činiteľov. Každý z nich je nanajvýš rovný n, a teda najvyššia mocnina 5, ktorá ho delí, je nanajvýš $k = \lfloor \log_5 n \rfloor$. Celkový exponent čísla 5 teraz vieme určiť tak, že pre každé x od 1 po k zistíme, koľko z činiteľov je deliteľných 5^x a všetky tieto počty sčítame.

Ako určiť počet činiteľov deliteľných 5^x pre nejaké pevne zvolené x? Sú to tie násobky 5^x, ktoré dávajú po delení 3 rovnaký zvyšok ako je zvyšok n po delení 3. Aby sme nemuseli rozoberať všetky prípady ručne, stačí v programe jednoducho vyskúšať, pre ktoré $z \in \{1,2,3\}$ platí $z \cdot 5^x \equiv n \pmod 3$. Potom všetky násobky 5^x, ktoré sa vyskytujú medzi činiteľmi čísla $n!_3$, vieme zjavne vyjadriť v tvare $(3y+z)\cdot 5^x$ pre $y \geq 0$. Ich počet ľahko zistíme tak, že nájdeme najmenšie y, pre ktoré už je $(3y+z)\cdot 5^x > n$.

z1924. V optimálnej veži nemôžu byť dve kocky, ktorých dĺžky hrán sa budú líšiť o viac ako 1. Inak by sme totiž mohli najväčšiu kocku o 1 zmenšiť a najmenšiu o 1 zväčšiť, čím celkovú výšku zachovávame a celkový objem klesne. Týmto pozorovaním je už jednoznačne určené, aké kocky tvoria optimálnu vežu: $(h \bmod n)$ z nich bude mať hranu dĺžky $\lfloor h/n \rfloor + 1$ a ostatné budú mať hranu dĺžky $\lfloor h/n \rfloor$.

z1925. V tejto úlohe potrebujeme podľa popisu v zadaní nájsť vhodnú šifrovaciu mriežku, ktorá bude obsahovať 4 spomenuté slová. Použitím týchto slov a postupným vylučovaním políčok, ktoré nemôžu byť vystrihnuté, dostávame riešenie vyobrazené napravo.

```
...0....00
.....00..0
...00...0.
......0...
..00.0...0
.0.......0
.....0....
0..0..00..
......0.0.
..00......
```

z1931. *Riešenie nadväzuje na text „Prehľadávanie do hĺbky“ na str. 246.* Križovatky s jednosmerkami tvoria orientovaný graf. Našou úlohou je overiť, či je tento graf silne súvislý. Vzorové riešenie v čase $O(m+n)$ je založené na pozorovaní, že stačí

overiť len dve podmienky: či sa z každej križovatky dá dostať na prvú a či sa z prvej križovatky dá dostať na každú inú. Potom už totiž určite existuje cesta medzi každými dvoma križovatkami.

Toto overovanie vieme šikovne spraviť nasledovne: Prehľadávaním z prvej križovatky overíme, či sú z nej všetky ostatné križovatky dosiahnuteľné. Potom každej hrane zmeníme orientáciu na opačnú. V pôvodnom grafe sa dalo z každej križovatky dostať na prvú práve vtedy, ak sa po otočení hrán dá z prvej križovatky dostať na každú inú. To overíme druhým prehľadávaním.

z1932. *Riešenie nadväzuje na text „Medián postupnosti“ na str. 243.*
Ak je domčekov párny počet, je správny stromček ten medzi prostrednými dvoma domčekmi (v zmysle počtu a nie vzdialenosti). Ak je domčekov nepárny počet, správny stromček je ten najbližší k strednému domčeku.

z1933. Číslo $a_i = \lfloor \sqrt{10^i - 1} \rfloor$ je najväčšie číslo, ktorého druhá mocnina ešte má práve i cifier. V našej postupnosti je teda postupne pre každé i za sebou $a_i - a_{i-1}$ čísel s i ciframi. Čísla s presne i ciframi tvoria úsek s dĺžkou $b_i = i(a_i - a_{i-1})$ cifier.

Pri hľadaní k-tej cifry postupnosti najskôr nájdeme najmenšie i také, že $b_1 + \cdots + b_i \geq k$. Tým sme zistili, že hľadaná cifra patrí do nejakého i-ciferného štvorca. Pozrime sa na úsek tvorený i-cifernými štvorcami. Nami hľadaná cifra je v tomto úseku na pozícii číslo $k' = k - (b_1 + \cdots + b_{i-1})$. Z toho už vieme priamo určiť, do ktorého čísla patrí, a aj to, koľká jeho cifra to je.

z1934. K mape Kiribati si predpočítame dvojrozmerné čiastočné súčty: pole $P[0..r, 0..s]$ také, že $P[i, j]$ je počet pevnín v obdĺžniku od ľavého horného rohu $[1, 1]$ po políčko $[i, j]$ vrátane. Potom pre ľubovoľné x_1, y_1, x_2, y_2 ($x_1 < x_2$, $y_1 < y_2$) platí, že počet pevnín v danej oblasti je rovný hodnote $P[x_2, y_2] - P[x_1 - 1, y_2] - P[x_2, y_2 - 1] + P[x_1 - 1, y_1 - 1]$. Pole P vieme zo vstupných údajov vypočítať v čase $O(rs)$ pomocou analogického vzťahu. Pre každú požiadavku následne vieme v konštantnom čase určiť, koľko políčok pevniny daná oblasť obsahuje, a teda aj to, či je celá prázdna.

z1935. Z priloženého obrázku sa dajú prečítať dva kľúče: „`PODUSTVA`“ a „`DUTOSVAP`“. (Najlepšie je asi držať papier takmer vodorovne na úrovni očí.) Všimnime si, že obe slová obsahujú rovnaké písmená. Nasekáme si správu na osmice a poprehadzujeme v nej písmená tak, ako by sme poprehadzovali písmená v `PODUSTVA` $\longleftrightarrow$ `DUTOSVAP`. Z dvoch možností ako prehadzovať jedna nevytvorí slovenský text, druhá nám rozšifruje správu: `PATRANIE PO NAJCASTEJSIE DREVENOM PREDMETE, KTORY MOZE SLUZIT NA UDRZIAVANIE STABILITY JEDINCOVI DRUHU HOMO SAPIENS, KTORYCH VEK VYSOKO PREKRACUJE MEDIAN TEJTO VELICINY NA SUBORE VSETKYCH ZIJUCICH JEDINCOV TOHTO DRUHU, SA U REFLEKTANTA ATAKU SELMY CELADE CANIDAE STRETA S USPECHOM ZA AKYCHKOLVEK PODMIENOK.`

z1941. Podľa zadania mestá spojené nejakou cestou nemôžu byť v tom istom okrese. Inak povedané, mestá nachádzajúce sa v jednom komponente súvislosti nemôžu byť v tom istom okrese. Najmenší počet okresov je teda zjavne rovný veľkosti najväčšieho komponentu súvislosti. Na zistenie tejto veľkosti použijeme prehľadávanie do šírky alebo hĺbky, ktoré má časovú a aj pamäťovú zložitosť $O(m + n)$.

z1942. Existuje presne $5! = 120$ možností ako môžu byť nugety v škatuľkách usporiadané. Naše riešenie položí vždy otázku tak, aby sme vylúčili približne polovicu možností. Takto na $\lceil \log_2 120 \rceil = 7$ otázok vždy uhádneme presné usporiadanie. Lepšie sa táto úloha ani nedá riešiť: ak by sme položili len 6 otázok, môžeme dostať len $2^6 = 64$ možných postupností odpovedí, čo nestačí na rozlíšenie medzi 120 možnosťami.

Optimálne riešenie začína napríklad tým, že prvou otázkou sa opýtame na škatuľky 1 a 2; druhou na škatuľky 3 a 4; a treťou na tie dve škatuľky, ktoré v prvých dvoch porovnávaniach obsahovali väčší nuget.

z1943. Ak má pole na výšku x štvorčekov (kde $1 \le x \le n$), tak na šírku musí mať aspoň $\lceil n/x \rceil$ štvorčekov a najviac $n+1-x$ štvorčekov.[9] A zjavne všetky šírky medzi týmito dvomi hraničnými hodnotami vieme dosiahnuť. Stačí napríklad začať od najširšieho poľa, ktoré má štvorčeky len v prvom stĺpci a prvom riadku, a postupne ho zužovať, kým to ide.

Pole s výškou x štvorčekov a šírkou y štvorčekov má obvod $2(x+y)$. Pre pole s výškou x teda vieme dosiahnuť všetky párne obvody od $2(x + \lceil n/x \rceil)$ až po $2(x + (n + 1 - x)) = 2(n + 1)$. Celkovo najväčší dosiahnuteľný obvod je teda zjavne $2(n + 1)$, zatiaľ čo najmenší dosiahnuteľný obvod dostávame $2(\lfloor \sqrt{n} \rfloor + \lceil \sqrt{n} \rceil)$ pre x približne rovné $\sqrt{n}$ (teda pre útvar čo najviac podobný štvorcu).

z1944. Pre každý riadok dvakrát použijeme binárne vyhľadávanie, aby sme našli prvú a poslednú nulu. Zo stĺpcov, v ktorých ležia, vypočítame počet núl v tomto riadku. Keď pre každý riadok poznáme počet núl v ňom, vzdialenosť lodí určíme ako minimum týchto počtov. Časová zložitosť tohto riešenia je $O(r \log s)$.

z1945. Prvý odsek je zašifrovaný nasledovne: Veľké písmeno určuje začiatok ďalšieho slova, v každom slove treba čítať písmená striedavo zo začiatku a z konca. Teda napríklad prvé slovo „`Zkzymdaa`“ dešifrujeme na „`Zakazdym`“.

Rozšifrovaná správa sa končí slovami „`Pred Sebou N Schodov A Na Jeden Krok Moze Prejst Jeden Alebo Dva`“. Táto nejasná nápoveda ukrýva postupnosť Fibonacciho čísel. (Koľkými spôsobmi vieme popísaným spôsobom prejsť n schodov? Označme tento počet $S[n]$. Zjavne $S[0] = S[1] = 1$. Vo všeobecnosti platí $S[n] = S[n-1] + S[n-2]$, lebo ak prvým krokom prejdeme 1 schod, ešte ich treba prejsť $n-1$, a ak prejdeme 2, ostane ich $n-2$. Preto $S[n] = F_n$.) V druhom odseku teda prečítame písmená na pozíciách zodpovedajúcich Fibonacciho číslam: 1, 2, 3, 5, 8, 13, 21, … Získame správu: „`KSPseminarSNOV`“.

2011. Pomocou triedenia v čase $O(n \log n)$ zostrojíme permutáciu chlapcov, ktorej vykonanie ich usporiada. Teda pre každé miesto i nájdeme miesto $F[i]$, na ktorom má skončiť chlapec, ktorý teraz stojí na mieste i. Potom stačí dokola opakovať postup: nájdeme chlapca i, ktorý nestojí na správnom mieste $F[i]$, a vymeníme ho s chlapcom, ktorý na mieste $F[i]$ stojí teraz.

Tento postup vieme efektívne implementovať tak, že permutáciu v čase $O(n)$ rozložíme na cykly; každý cyklus môžeme následne vyriešiť samostatne v čase priamo úmernom jeho dĺžke. Z toho, že každou z našich výmen zvýšime o jeden počet cyklov našej permutácie, zároveň vyplýva aj optimálnosť nášho riešenia.

2012. Jediné, čo potrebujeme určiť, je uhol, ktorý zviera daná priamka s kladnou poloosou osi x. Každej dvojici po sebe nasledujúcich staníc zodpovedá jeden otvorený interval, v ktorom tento uhol musí ležať, aby päty kolmíc z dotyčných dvoch staníc boli v správnom poradí. Stačí teda v lineárnom čase a konštantnej pamäti zostrojiť prienik týchto intervalov a zistiť, či je prázdny.

2013. V reči teórie grafov je našou úlohou nájsť druhú najlacnejšiu kostru zadaného grafu. Dá sa ľahko dokázať, že tá sa vždy od najlacnejšej líši len jednou zmenou – pridaním jednej hrany a odobratím nejakej inej.

Úlohu vieme riešiť v čase $O(n^2)$ nasledovne: Najskôr nájdeme najlacnejšiu kostru pomocou Jarníkovho-Primovho algoritmu. Počas neho si zároveň vypĺňame tabuľku, v ktorej máme pre každú dvojicu už spojených vrcholov u, v cenu $D[u, v]$ najdrahšej z hrán, ktorá v kostre leží na ceste medzi u a v.

---

[9] Prvé tvrdenie je zjavné – ak by bolo pole užšie, n štvorčekov sa tam nezmestí. Druhé tvrdenie je intuitívne zjavné tiež, dôkaz je však komplikovanejší. Jeden možný dôkaz: Pre každé $r < x$ v riadku r musí existovať štvorček, ktorý susedí so štvorčekom v riadku $r + 1$, inak by pole nebolo súvislé. Jeden takýto štvorček si v každom riadku vyberme a potom všetky vybraté štvorčeky odstráňme. Zostalo nám $n + 1 - x$ štvorčekov. Keďže v žiadnom stĺpci sme neodstránili najspodnejší štvorček, tieto štvorčeky dokopy pokrývajú toľko isto stĺpcov ako pôvodne pokrývalo celé pole.

Po dokončení tejto časti výpočtu postupne skúšame do kostry doplniť každú z „nekostrových“ hrán. Doplnením hrany (p, q) dostaneme z kostry graf s práve jednou kružnicou. Najlacnejší spôsob, ako sa tejto kružnice zbaviť a ponechať pritom hranu (p, q), je vyhodiť z cesty medzi p a q hranu s cenou $D[p, q]$. Riešením úlohy je najlacnejšia zo všetkých takto získaných nových kostier.

Efektívnejšie riešenie pre riedke grafy: Pomocou Kruskalovho algoritmu nájdeme najlacnejšiu kostru v čase $O(m \log n)$. Túto kostru následne v čase $O(n \log n)$ predspracujeme tak, aby sme vedeli pre každé dva vrcholy povedať v čase $O(\log n)$ cenu najdrahšej hrany na ceste medzi nimi. Takto dostaneme riešenie s celkovou časovou zložitosťou $O(m \log n)$.

Základná myšlienka jednej možnosti predspracovania: Zvolíme jeden vrchol kostry ako koreň. Následne postupne pre každé i od 0 po $\log_2 n$ spočítame pre každý vrchol v, v ktorom vrchole budeme, keď z v spravíme 2^i krokov smerom ku koreňu, a akú najdrahšiu hranu na tejto ceste prejdeme.

2014. Ak by sme pre každé políčko vedeli, ktorá zo zlatých dlaždíc je k nemu (v Manhattanovskej metrike) druhá najbližšia, ľahko nájdeme stred najväčšieho diamantu – stačí brať do úvahy túto hodnotu a vzdialenosti políčka od krajov miestnosti.

Riešenie v čase aj pamäti $O(rs)$: spustíme prehľadávanie do šírky naraz zo všetkých zlatých dlaždíc, pričom si vo fronte okrem aktuálnych súradníc políčka pamätáme aj číslo zlatej dlaždice, z ktorého aktuálny „signál“ prichádza. Z každého políčka ďalej pošleme len prvé dva rôzne signály, ktoré naň dorazia.

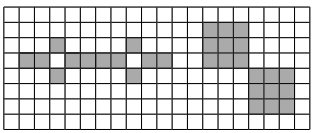
Riešenie v čase $O(rs)$ a pamäti $O(r)$: Použijeme dynamické programovanie. Postupne pre každé políčko spočítame aký najdlhší pás políčok vedie šikmo doľava/doprava hore, ak smie obsahovať len 0/1 zlatých dlaždíc, a následne aký najväčší diamant s 0/1 zlatými dlaždicami vnútri má na danom políčku spodný roh.

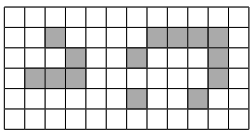
Myšlienka riešenia, ktoré nebude závisieť od veľkosti plochy: Budeme hovoriť, že každá dlaždica *patrí* k tej zlatej, ktorá je k nej najbližšie. Ak by sme mali na každej dlaždici zaznačené, kam patrí, miestnosť by sa nám rozpadla na niekoľko súvislých a „konvexných“ oblastí – jedna pre každú zlatú dlaždicu. Toto rozdelenie je akousi diskrétnou analógiou Voronoiovho diagramu.

Hranica dvoch susediacich oblastí bude vždy akoby osou súmernosti zlatých políčok, ktorým tieto oblasti patria. (Nakreslite si to napríklad pre dve zlaté políčka: $[0, 0]$ a $[4, 7]$.) Pre každé dve zlaté políčka vieme túto hranicu zostrojiť v $O(1)$ a následne spočítať tvar jednotlivých oblastí v čase polynomiálnom od z. Následne vieme ľahko nájsť najväčší prázdny diamant: Dá sa uvidieť, že ten má vždy stred na mieste, kde sa stretávajú tri oblasti. (Až na špeciálne prípady, kedy sa dotýka aspoň jednej strany.) Keď už vieme nájsť najväčší prázdny diamant, stačí ho nájsť z-krát: raz pre každých $z - 1$ zlatých dlaždíc zo vstupu. Takto dostávame riešenie s časovou zložitosťou polynomiálnou od z.

Existuje veľmi komplikované riešenie s časovou zložitosťou $O(z \log z)$, ktoré je založené na vyššie uvedených myšlienkach a na úprave Fortunovho algoritmu na zostrojenie Voronoiovho diagramu do našej diskrétnej podoby.

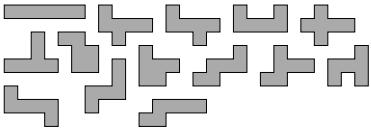
2015. Riešenia úlohy a) sa nazývajú oscilátory (*oscillators*), riešenia úlohy b) sú tzv. kozmické lode (*spaceships*). Dva oscilátory sú na obrázku vľavo: tzv. *pentadecathlon* s periódou 15 a *figure eight* s periódou 8. Na obrázku vpravo sú dve kozmické lode: tzv. *glider* a *LWSS* (*lightweight spaceship*). Glider je úplne najmenšia kozmická loď a pohybuje sa diagonálne, LWSS je druhá najmenšia, resp. najmenšia, ktorá sa pohybuje vodorovne/zvisle.





Life je oveľa, oveľa zaujímavejšia hra ako sa na prvý pohľad zdá. Čitateľovi odporúčame stránky ako `http://www.conwaylife.com/wiki`, či `http://pentadecathlon.com/`, kde nájde mnoho ďalších oscilátorov, kozmických lodí a iných zaujímavých konfigurácií ako sú pušky (*guns*), ktoré „vystreľujú“ kozmické lode, či odrážače, ktoré dokážu zmeniť ich smer. Existujú dokonca konfigurácie, ktoré simulujú logické obvody, „hľadajú“ prvočísla, alebo počítajú ľubovoľnú funkciu, ktorú zvládne vypočítať váš počítač.

Riešením úlohy c) je týchto 14 (z celkového počtu 35 rôznych) hexamín:



2021. Ak do niektorej skupiny patrí viac ako polovica žiakov, riešenie zjavne neexistuje: pre ich skupinu určite nemáme dosť opravovateľov.

V opačnom prípade riešenie vždy existuje. Stačí napríklad postaviť všetkých n žiakov do kruhu tak, aby každá skupina tvorila súvislý úsek, a následne každému dať písomku žiaka, ktorý stojí o $\lfloor n/2 \rfloor$ pozícií ďalej. Vhodným upravením algoritmu CountSort sa dá toto riešenie implementovať s časovou zložitosťou $O(n)$.

2022. Predstavme si, že zoberieme priamku predstavujúcu metro a začneme ju otáčať. V ktorých okamihoch sa zmení poradie staníc? Bude to vždy, keď bude priamka kolmá na spojnicu nejakých dvoch staníc – v danom okamihu totiž zjavne ich priemety prejdú cez seba. Najskôr teda zistíme poradie pre jedno možné natočenie priamky (napríklad pre vodorovnú priamku). Následne vygenerujeme všetkých $n(n-1)$ uhlov, pri ktorých sa mení poradie staníc. Všetky tieto uhly usporiadame a postupne spracúvame (teda upravujeme poradie staníc). Treba si dať pozor na situácie, kedy viacero zmien nastane súčasne – až keď spracujeme poslednú z nich, dostali sme nové poradie staníc. Takto dostávame program s optimálnou časovou zložitosťou $O(n^3)$. Lepšie to nejde, lebo najpomalšou časťou riešenia je už samotný výpis výstupu.

2023. *Riešenie nadväzuje na text „Dijkstrov algoritmus“ na str. 248.*
Stačí upraviť Dijkstrov algoritmus tak, že si pre každý vrchol okrem dĺžky najkratšej doteraz nájdenej cesty doň pamätáme aj počet ciest tejto dĺžky.

2024. Riešenie v $O(r^2 s)$: V každom stĺpci predpočítame čiastočné súčty, aby sme vedeli v konštantnom čase o ľubovoľnom úseku povedať, či je celý prázdny. Vyskúšame všetky možné dvojice (prvý riadok, posledný riadok obdĺžnika). Keď už máme zvolené, v ktorých riadkoch obdĺžnik leží a v ktorých nie, vieme jednotlivé stĺpce roztriediť na prázdne a neprázdne a následne v lineárnom čase nájsť najdlhší súvislý úsek prázdnych stĺpcov.

Jedno možné optimálne riešenie: Použijeme dynamické programovanie. Vstup spracúvame po riadkoch. V novom riadku najskôr pre každé políčko i vypočítame výšku v_i prázdneho stĺpca, ktorý na ňom končí. Následne prechodom zľava doprava spočítame pre každé políčko i hodnotu ℓ_i: koľko políčok doľava môže siahať obdĺžnik, ktorý končí v stĺpci i a má výšku v_i? Následne prechodom sprava doľava spočítame analogické hodnoty r_i. Najväčší obdĺžnik nájdeme tak, že nájdeme maximum súčinu $v_i(\ell_i + r_i - 1)$.

Iné optimálne riešenie: Identicky spravíme všetko po spočítanie v_i pre nový riadok. Následne usporiadame stĺpce podľa v_i klesajúco (na to môžeme použiť count-sort) a postupne ich pridávame, pričom zakaždým, keď pridáme stĺpec, v $O(1)$ odpovieme na otázku, ako najviac doľava a doprava môže siahať obdĺžnik, ktorý obsahuje všetkých v_i políčok v stĺpci i.

2025. Číslo budeme postupne násobiť od najmenej významného bitu k najvýznamnejšiemu, počet krokov výpočtu bude teda priamo úmerný počtu automatov. Slovne môžeme fungovanie každého automatu popísať nasledovne:

a) Ak je môj pravý sused v stave $*$, vynásobím svoj bit tromi a prejdem do stavu „výsledný bit x, zvyšok y“ pre vhodné x, y.

b) Ak je môj pravý sused v stave „výsledný bit x, zvyšok y“, vynásobím svoj bit tromi, pripočítam y a prejdem do stavu „výsledný bit x', zvyšok y'“ pre vhodné x', y'

c) Ak som v stave „výsledný bit x, zvyšok y“, prejdem do stavu x.

d) Inak ostávam v stave, v ktorom som bol doteraz.

Ľahko nahliadneme, že po každom kroku výpočtu bude práve jeden automat v niektorom zo stavov „výsledný bit x, zvyšok y“. V nasledujúcom kroku výpočtu tento automat použije pravidlo c), zatiaľ čo jeho ľavý sused použije pravidlo b), čím do takéhoto stavu prejde. Toto prebieha až kým sa nedostaneme k najľavejšiemu automatu – ten už ľavého suseda nemá, preto od okamihu, kedy on použije pravidlo c), sa už stav žiadneho z automatov meniť nebude.

2031. Úlohu riešime dynamickým programovaním, pri ktorom brigády spracúvame usporiadané podľa času, kedy končia.

Nech $F[i]$ je najväčší počet peňazí, ktoré môže Tomáš zarobiť, ak navštívi niektoré z prvých i brigád. Zjavne $F[0] = 0$ a $F[1] = c_1$. Ostatné $F[i]$ zistíme nasledovne. Ak na i-tu brigádu Tomáš nepôjde, zarobí nanajvýš $F[i-1]$. Ak na ňu pôjde, potrebujeme zistiť najväčšie j také, že čas konca j-tej brigády nie je neskôr ako čas začiatku i-tej brigády. Toto j vieme nájsť binárnym vyhľadávaním. Najlepší možný zárobok pre túto možnosť následne spočítame ako $F[j] + c_i$. Preto platí $F[i] = \max(F[i-1], F[j] + c_i)$.

Časová zložitosť tohto algoritmu je $O(n \log n)$.

2032. Riešenie v čase $O(d^2 s^2)$: Ak existuje cesta, určite existuje taká, kde vyletíme do nejakej výšky, potom sa presunieme a nakoniec zletíme dole. Navyše platí, že ako výšku, do ktorej vyletíme, stačí uvažovať iba výšky tvaru $1 + z$, kde z je výška niektorého políčka. Postupne teda vyskúšame všetkých $O(ds)$ výšok a pre každú z nich použijeme prehľadávanie do šírky na zistenie najmenšieho počtu vodorovných krokov na dosiahnutie cieľa. Z takto zostrojených riešení si samozrejme vyberieme to najlepšie.

2033. Prvý bod A_1 bez ujmy na všeobecnosti umiestnime na súradnice $[0, 0]$. Bod A_2 dostane súradnice $[0, x]$, kde x je vzdialenosť A_1 a A_2. Pre polohu A_3 dostaneme zo vzdialeností $|A_1A_3|$ a $|A_2A_3|$ dve možnosti, vyberieme si ľubovoľnú z nich. Každý ďalší bod vieme jednoznačne umiestniť na základe jeho vzdialeností od prvých troch bodov. Takto v lineárnom čase nájdeme všetky body.

Následne zistíme ich poradie na obvode mnohouholníka. Keďže vieme, že je konvexný, stačí body $A_2, A_3, \ldots, A_n$ usporiadať podľa uhlu, ktorý zviera úsečka A_1A_i s osou x.

Najkratšia cesta vedie po obvode mnohouholníka. Ak by sa totiž nejaké dve hrany našej okružnej cesty križovali, môžeme ich zahodiť a vzniknuté dve časti cesty spojiť dvojicou nekrižujúcich sa hrán. Vďaka trojuholníkovej nerovnosti tým určite skrátime celkovú dĺžku cesty. Na záver riešenia preto stačí vypočítať obvod zostrojeného mnohouholníka.

2034. Odhrnuté časti jazera nazvime stĺpce, množstvo odhrnutia bude ich výška. Na nájdenie riešenia stačí pre každý stĺpec vyskúšať obdĺžnik, ktorý dotyčný stĺpec celý obsahuje a doľava aj doprava siaha najďalej ako môže. Priamym odskúšaním takýchto obdĺžnikov dostávame riešenie s časovou zložitosťou $O(n^2)$.

Všimnime si, že keby sme pre každý stĺpec poznali naľavo aj napravo najbližší stĺpec, ktorý je od neho menší, tak úlohu vieme doriešiť v lineárnom čase. Pre stĺpec i označme čísla týchto dvoch stĺpcov ℓ_i a p_i. Popíšeme si, ako zistiť hodnoty ℓ_i; pre p_i je situácia symetrická. Využijeme zásobník, v ktorom si budeme pamätať záznamy tvaru (výška stĺpca, číslo stĺpca). Prechádzame vstupom zľava doprava. Keď spracúvame stĺpec výšky a_i, tak najprv

zo zásobníka vyhádžeme záznamy vyššie ako a_i. Ak práve nie je zásobník prázdny, záznam na vrchu nám povie hodnotu ℓ_i. Následne do zásobníka vložíme záznam (a_i, i).

Všimnime si, že v každom okamihu máme v zásobníku práve tie stĺpce, ktoré z aktuálnej pozície „pozerajúc doľava ešte stále celé vidíme“. Inými slovami, práve tie stĺpce, ktoré sa ešte niekedy môžu stať stĺpcom ℓ_i. Každú hodnotu raz vložíme do zásobníka a raz ju odtiaľ odstránime, preto má toto riešenie časovú zložitosť $O(n)$.

2035. Riešenie s kvadratickým počtom krokov: Na začiatku výpočtu sú všetky automaty neofarbené. Postupne budeme na striedačku ofarbovať najľavejší a najpravejší neofarbený automat – vždy nejakým špeciálnym stavom „prelezieme“ po všetkých neofarbených automatoch na opačný koniec a keď nájdeme posledný neofarbený, ofarbíme ho. Ak nám v správnom okamihu zostanú nanajvýš dva neofarbené automaty, označíme ich ako stredné.

Existuje aj riešenie s lineárnym počtom krokov: Použijeme dva „signály“; prvý signál sa bude pohybovať rýchlosťou 1. Reprezentovať ho budeme tak, že zavedieme dva nové stavy: „1. signál doľava“ a „1. signál doprava“. Každý automat bude mať vo svojom programe inštrukcie:

– Ak je môj ľavý sused v stave „1. signál doprava“, prejdem do tohto stavu.

– Ak som v stave „1. signál doprava“ a som najpravejší, prejdem do „1. signál doľava“.

– Ak je môj pravý sused v stave „1. signál doľava“, prejdem do tohto stavu.

– Inak ak som v stave „1. signál (niekam)“, prejdem do stavu „nič sa nedeje“.

Druhý signál sa bude pohybovať rýchlosťou 1/3, teda posunie sa o automat raz za 3 kroky. Implementujeme ho rovnako ako prvý signál, len súčasťou stavu bude počítadlo nadobúdajúce hodnoty 0, 1, 2. Keď je správny sused v stave „2. signál (správnym smerom) 2“, prejdem do stavu „2. signál (tým istým smerom) 0“ a následne v každom kroku prejdem do stavu s o 1 vyšším číslom.

Na nájdenie stredu stačí, aby začiatočný automat vyslal oba tieto signály. Keď sa prvýkrát stretnú, bude to práve v strede. Kým totiž pomalší signál prejde $n/2$ automatov, rýchlejší ich prejde $3n/2$, čo stačí práve na to, aby prišiel na koniec, otočil sa a prešiel $n/2$ v opačnom smere.

2041. Hľadáme nasledujúcu hodnotu tzv. interpolačného polynómu. Tento polynóm je síce danými hodnotami jednoznačne určený, existuje však viacero jeho vyjadrení a tie vedú k rôzne efektívnym riešeniam tejto úlohy. My použijeme Lagrangeov tvar.

Hľadáme polynóm, ktorý má pre každé i od 1 po n v bode $x_i = i$ hodnotu y_i. Označme $p(x) = (x - x_1)(x - x_2)\cdots(x - x_n)$ polynóm, ktorý má korene práve $x_1, \ldots, x_n$. Polynóm $q_k(x) = p(x)/(x - x_k)$ má ako korene všetky x_i okrem x_k. Polynóm $\ell_k(x) = q_k(x)/q_k(x_k)$ má navyše v bode x_k hodnotu 1. Hľadaný polynóm teraz vieme naskladať zo všetkých ℓ_k nasledovne: $L(x) = y_1\ell_1(x) + y_2\ell_2(x) + \cdots + y_n\ell_n(x)$.

Po dosadení $x_i = i$ dostávame, že $\ell_k(n+1) = (-1)^{n-k}\binom{n}{k-1}$, odkiaľ vidíme, že všetky $\ell_k(n+1)$ sú celé a môžeme ich počítať pomocou vzťahu $\ell_1(n+1) = (-1)^{n-1}$ a $\ell_{k+1}(n+1) = \ell_k(n+1)\cdot(k-1-n)/k$. Pomocou tohto vzťahu nám na vyhodnotenie $L(n+1)$ stačí $O(n)$ matematických operácií.

2042. *Riešenie nadväzuje na text „Dijkstrov algoritmus“ na str. 248.*
Sieť metra si budeme reprezentovať ako graf. Zostrojíme dva samostatné n-vrcholové grafy: jeden s hranami metra KSP, druhý pre DPMB. Hrany v týchto grafoch budú mať vzdialenosti podľa vstupu a nulovú cenu za prestup. Medzi vrcholmi, ktoré zodpovedajú tej istej stanici, následne pridáme hrany s nulovou vzdialenosťou a jednotkovou cenou za prestup. Potom na tomto grafe spustíme Dijkstrov algoritmus. Prvým kritériom bude vzdialenosť, druhým počet prestupov.

2043. *Riešenie nadväzuje na text „Prehľadávanie do hĺbky“ na str. 246.*
Úlohou je nájsť v grafe všetky artikulácie. To vieme spraviť v čase lineárnom od veľkosti grafu upraveným prehľadávaním do hĺbky.

2044. Na riešenie tejto úlohy sa dá použiť viacero algoritmov pracujúcich aj v najhoršom možnom prípade v čase lineárnom od veľkosti vstupu. Napríklad ide o Knuthov-Morrisov-Prattov algoritmus alebo Boyerov-Moorov algoritmus.

Veľmi ľahko sa implementuje Rabinov-Karpov algoritmus, využívajúci hešovanie. Ten má očakávanú časovú zložitosť taktiež lineárnu. Najhorší prípad síce môže byť horší, ale pomocou náhodných čísel vieme zabezpečiť, že jeho pravdepodobnosť bude zanedbateľne malá. Pri tomto algoritme je trik v tom, že si zvolíme hešovaciu funkciu tak, aby sme vedeli z hešu pre reťazec $a_1 \dots a_n$ zistiť heš pre reťazec $a_2 \dots a_{n+1}$ v konštantnom čase. Jednu takúto hešovaciu funkciu dostaneme, ak sa na písmená reťazca budeme dívať ako na cifry veľkého čísla v nejakej pevne zvolenej sústave (napríklad 256-kovej), pričom ako heš vezmeme zvyšok, ktorý toto veľké číslo dáva po delení konkrétnym nami zvoleným číslom (napríklad dostatočne veľkým prvočíslom).

2045. Použijeme ako „procedúru“ vzorové riešenie úlohy 2035. Nájdeme stred úseku automatov. Tým rozdelíme celé pole na dva rovnako dlhé kusy, plus jeden alebo dva automaty v strede. Stredné automaty necháme čakať v špeciálnom stave c. V každom úseku (v tom ľavom zrkadlovo obrátene) teraz spravíme to isté, čím dostaneme štyri rovnako dlhé úseky a pár ďalších automatov v stave c. Dôležité je všimnúť si, že nikde nemáme tri automaty v stave c vedľa seba. Toto platí až do okamihu, kým po niekoľkých opakovaniach neklesne dĺžka úsekov na 1. V nasledujúcom kroku výpočtu aj všetky tieto automaty zmenia svoj stav na c, no a v poslednom kroku každý automat použije pravidlo: „ak som v c aj ja, aj všetci moji susedia, zmením stav na S“. Ľahko spočítame, že celkový počet krokov, ktorý toto riešenie spraví, bude lineárny od počtu automatov.

z2011. Tým, ako sa voda prelieva cez vysoké body, sa nám krajina rozdelí na niekoľko jazierok a niekoľko hrádzí medzi nimi. Za najvzdialenejšiu (poslednú) hrádzu môžeme považovať Braňovu garáž. Ak poznáme nejakú hrádzu, predchádzajúcu nájdeme ako najbližšie vyššie miesto smerom doľava. Počas hľadania ďalšej hrádze si zároveň môžeme rátať objem jazierka. Takto dosiahneme riešenie s lineárnou časovou zložitosťou.

z2012. Úlohu si predstavíme ako graf (gaštany sú vrcholy, zápalky hrany). Následne prehľadávaním (či už do šírky alebo do hĺbky) určíme v čase $O(m+n)$ počet komponentov grafu, čo je aj počet útvarov.

z2013. Pomocou binárneho vyhľadávania nájdeme najnižšiu vetvu, ktorá sa do dverí nezmestí na šírku. Stromček odpílime tesne nad ňou, prípadne vyššie, ak by sa nezmestil do dverí na výšku.

z2014. Všimnime si, že každý prepínač sa nám oplatí prepnúť maximálne raz a že na poradí prepínania nezáleží. Predstavme si, že najskôr budeme prepínať riadky a až potom pôjdeme prepínať stĺpce. Teda po prepnutí riadkov už musíme mať v každom stĺpci iba rovnaké hodnoty – buď samé nuly, alebo samé jednotky. A to sa dá dosiahnuť práve vtedy, keď je každý riadok buď rovný prvému riadku, alebo je rovný jeho negácii.

Potom stačí vyskúsať dve rôzne riešenia. Buď všetky riadky prepneme do stavu rovnakého ako prvý riadok, alebo všetky riadky budú negáciou prvého riadku. Takto dostávame časovú zložitosť $O(rs)$. Vieme dosiahnuť pamäťovú zložitosť $O(r+s)$, lebo si stačí pamätať prvý riadok a u ostatných len či sú rovnaké ako prvý, alebo sú jeho negáciou.

z2015. *Riešenie nadväzuje na text „Union-find“ na str. 249.*
Uvedieme dva štandardné spôsoby generovania náhodných dobre vyzerajúcich bludísk. V oboch začneme s mriežkou $r \times s$ izolovaných miestností, medzi ktorými sú všade steny. Bludisko vytvoríme tak, že niektoré z týchto stien zbúrame. Na tento začiatočný stav sa môžeme dívať ako na graf s $r \times s$ vrcholmi predstavujúcimi miestnosti a hranami predstavujúcimi steny medzi miestnosťami.

Prvým spôsobom generovania je náhodné prehľadávanie nášho grafu do hĺbky: v každom vrchole si náhodne zvolíme poradie, v akom budeme skúšať vychádzajúce hrany. Počas

prehľadávania si pre každý vrchol zapamätáme hranu, ktorou sme ho objavili. Na konci nám tieto hrany vytvoria strom. Steny zodpovedajúce hranám stromu v našom bludisku vybúrame.

Druhým spôsobom je hľadanie najlacnejšej kostry. Opäť zoberieme ten istý graf, len tentokrát každej hrane priradíme náhodnú váhu. V takomto grafe následne nájdeme najlacnejšiu kostru. Ak na to použijeme Kruskalov algoritmus, môžeme si priebeh výpočtu priblížiť nasledovne: spracúvame steny medzi miestnosťami usporiadané podľa ich náhodných váh. Vždy, keď spracúvame stenu, pozrieme sa, či je na oboch jej stranách tá istá miestnosť. Ak áno, necháme stenu na pokoji, ak nie, prebúrame ju.

Aby sme v bludisku nemali len jeden strom chodieb a nič iné, môžeme na záver ešte prebúrať niekoľko náhodne vybraných stien.

Zostáva doriešiť generovanie grafického výstupu. Ak chceme výstup rozmerov $r \times s$, vygenerujeme bludisko rozmerov $(r/2) \times (s/2)$ a to vykreslíme nasledovne: miestnosť so súradnicami $[a, b]$ zakreslíme ako znak '`.`' na súradniciach $[2a, 2b]$, steny okolo tejto miestnosti zakreslíme ako znaky '`x`' alebo '`.`' na súradniciach $[2a \pm 1, 2b]$ a $[2a, 2b \pm 1]$. Zvyšné miesta výstupu vyplníme znakmi '`x`'. (Oba spôsoby generovania sa dajú naprogramovať tak, aby priamo počas výpočtu takto vypĺňali výstupné pole.)

z2021. Najprv jedným prehľadávaním do šírky z rohu mapy zistíme, ktoré políčka tvoria more. Následne druhým prehľadávaním zistíme, ktoré políčka prislúchajú zadanému ostrovu. Priamo počas tohto prehľadávania vieme zistiť aj to, koľko políčok má tento ostrov, aj to, koľko z nich susedí s morom. Časová aj pamäťová zložitosť bude $O(rs)$.

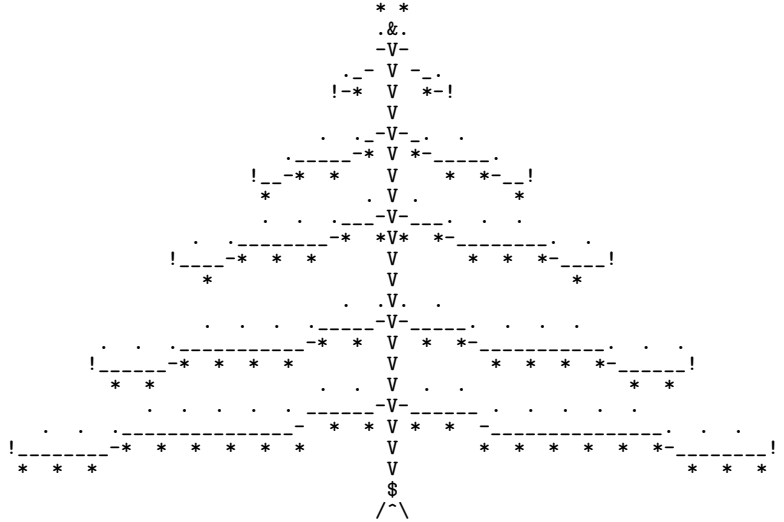
z2022. Úlohu si reprezentujeme ako graf. Stačí si uvedomiť, že keď má komponent súvislosti grafu n vrcholov, tak je optimálne ponechať v ňom vhodných $n-1$ hrán (presnejšie, ľubovoľnú jeho kostru). Takže ak má graf k komponentov a dokopy n vrcholov, tak v optimálnom riešení ponecháme dokopy $n-k$ hrán. Počet komponentov vieme zistiť ľubovoľným prehľadavaním.

z2023. Po tom, ako optimálne otočíme stromček a odpílime zlé vetvy, ho určite môžeme otočiť tak, aby sa jednou vetvou dotýkal steny. Ak by sme ho hneď na začiatku natočili tak, ako zostal natočený na konci, dostali by sme to isté riešenie. Preto stačí preskúmať $2n$ možností: ktorá vetva sa dotýka steny a ktorým smerom ide. Priamočiarou implementáciou dostávame riešenie v čase $O(n^2)$.

Lepšie riešenie v čase $O(n \log n)$: Na začiatku zistíme, koľko vetiev by sme mali, ak by sme stromček odpílili bez otáčania, a ktoré by to boli. Následne pre každú vetvu i spočítame uhly α_i a $\alpha_i + \pi$, o ktoré keď stromček otočíme, tak bude presne rovnobežná so stenou. Postupnosť $2n$ záznamov (číslo vetvy, veľkosť uhla) usporiadame podľa veľkosti uhla a následne spracujeme. Tým vlastne simulujeme postupné otáčanie stromčeka, pričom si udržiavame pole s údajmi, ktoré vetvy sú momentálne „živé“ a ktoré nie. Každý záznam spracujeme v konštantnom čase, preto jedinou nie lineárnou časťou tohto riešenia je samotné triedenie.

z2024. Aby sme zabezpečili, že nevygenerujeme dva rôzne výbery, ktoré sa budú líšiť len poradím cukríkov, budeme cukríky vyberať v poradí od najdrahšieho po najlacnejší. Najprv vyberieme prvý cukrík. Pri tomto výbere je horný limit na jeho cenu len celkový počet korún n. Potom vyberieme druhý cukrík. Tam už máme limity na cenu dva: počet korún, ktoré ešte máme a cenu predchádzajúceho cukríka. A tak ďalej. Skúšanie všetkých takýchto možností vieme jednoducho implementovať ako rekurzívnu funkciu. Čas behu bude približne priamo úmerný veľkosti *výstupu*. Pamäťové nároky su $O(n)$, keďže možnosti generujeme postupne jednu po druhej.

z2025. Najlepšie riešenie, ktoré sme dostali, vyrobilo napríklad nasledujúci výstup:



z2031. V reči teórie grafov je našou úlohou nájsť centrum daného stromu. Jedno možné riešenie: Z daného stromu zmažeme všetky listy. Zo stromu, ktorý sme takto dostali, zasa zmažeme všetky listy. Toto opakujeme, až kým nám neostane buď jediný vrchol, alebo dva vrcholy spojené hranou. Každý z vrcholov, ktorý nám zostal, je optimálnym riešením. Pomocou fronty sa tento algoritmus dá implementovať v čase a pamäti $O(n)$.

Iné riešenie: Centrum stromu vždy leží v strede najdlhšej cesty v danom strome. Tú vieme nájsť napríklad nasledovným trikom: nech x je ľubovoľný vrchol, y najvzdialenejší vrchol od x a následne nech z je najvzdialenejší vrchol od y. Potom cesta z y do z je najdlhšou cestou v danom strome.

z2032. Najjednoduchšia je asi implementácia na dva prechody poľom. V prvom prechode ho preusporiadame tak, aby sa všetky písmená dostali naľavo od bodiek a čísel. To vieme spraviť tak, že prechádzame pole zľava doprava a pamätáme si počet x písmen, ktoré sme už spracovali. Vždy, keď nájdeme ďalšie písmeno, zvýšime si x a presunieme ho na x-té miesto v poli. V druhom prechode rovnakým spôsobom presunieme všetky bodky naľavo od čísel. Časová zložitosť bude $O(n)$.

Poznámka: Uvedené preusporiadanie je jednou zo súčastí algoritmu quick-sort – pri ňom rozdeľujeme pole na prvky menšie ako zvolený pivot, rovné pivotu a väčšie od neho.

z2033. Nie je potrebné kontrolovať priesečníky všetkých dvojíc strán. Efektívnejšie riešenie je založené na pozorovaní, že mnohouholník je konvexný, práve vtedy, keď platí: Keď ideme po jeho obvode, tak v každom vrchole zatočíme do tej istej strany a po zatočení dokopy o 360° sa vrátime na miesto, kde sme začínali. Smer zatáčania vieme kontrolovať napríklad pomocou vektorového súčinu, veľkosť uhla, o ktorý zatáčame, zistíme napríklad pomocou funkcie arctan. Takto vieme obe podmienky overiť v čase $O(n)$ a konštantnej pamäti.

z2034. *Riešenie je uvedené v texte „Euklidov algoritmus“ na str. 244.*

z2035. Najlepšie odovzdané riešenia vznikli približne nasledovne: Zoberieme si nejaké reálne básničky a pre každý verš si zostrojíme postupnosť slovných druhov, ktoré obsahuje. Tieto „schémy veršov“ si uložíme do databázy. Následne si zostrojíme databázy slov, ktoré

môžeme použiť ako jednotlivé slovné druhy. Samotné generovanie básničky potom vyzerá tak, že si zvolíme schémy veršov a do nich za slovné druhy dosádzame jednotlivé slová tak, aby vznikali rýmy.

Príklad vygenerovanej básničky:

```
Sviečka sviečka otvor viečka
Tma nech stráži v poli škrečka
Prostred kruhu šťastie spinká
V poli krásna nezábudka
Pozor veď vám horí metla
Včera som zas jeho stretla
Tešila sa stará kára
Čo ten môj vnuk doma stvára
```

z2041. Označme (a_i, b_i) intervaly, ktoré zaberajú lajná. Aby sme sa po i-tom kroku ocitli medzi i-tym a $(i+1)$-vým lajnom, musí pre dĺžku nášho kroku d platiť $b_i \leq i \cdot d \leq a_{i+1}$.

Riešením takejto nerovnice je interval vhodných hodnôt pre d. Ak máme prejsť prvých k krokov, dostávame pre d sústavu k nerovníc. Jej riešením je prienik riešení jednotlivých nerovníc, teda opäť nejaký interval.

Budeme postupne zväčšovať k, pridávať nerovnice a upravovať výsledný interval. Len čo dostaneme prázdny prienik, vieme, že sa ďalej po chodníku pokračovať nedá.

z2042. Priamo počas načítavania vstupu si môžeme počítať pre každý tím informácie o počte bodov, trestných minút a neúspešných submitov ku každej úlohe. Nakoniec tímy usporiadame použitím vhodného triediaceho algoritmu.

z2043. Zadanie popisuje známy problém o Hanojských vežiach a jeden algoritmus, ktorý ho rieši. Na presun celej veže tvorenej k diskami potrebuje tento algoritmus presne $2^k - 1$ presunov disku. Toto pozorovanie použijeme na zrýchlenie simulácie: ak máme odsimulovať t ťahov, nájdeme najväčšie k také, že $2^k - 1 \leq t$, a rovno presunieme celú vežu veľkosti k. Ak ešte máme simulovať nenulový počet ťahov, odsimulujeme presun disku, ktorý sme práve uvoľnili a následne celý postup začneme odznova. Pri dobrej implementácii (postupne znižujeme k od n po 1 a pre každý disk, začínajúc najväčším, zistíme, kde skončí) má tento algoritmus časovú zložitosť lineárnu od počtu diskov.

z2044. Vstup môžeme kontrolovať priamo pri jeho načítavaní bez toho, aby sme si ho museli celý uložiť. Stačí si počítať zvyšok dĺžky zatiaľ prečítanej časti po delení číslom 3 a pamätať posledných 6 znakov. Takto dosiahneme lineárnu časovú a konštantnú pamäťovú zložitosť. Existuje dokonca riešenie, ktoré si vystačí s jedinou premennou, a aj tá je len typu `char` a mení sa jedine čítaním ďalšieho písmena zo vstupu.

z2045. Takéto rovnice nazývame algebrogramy. Riešime ich pomocou backtrackingu, skúšaním všetkých možností priradenia číselných hodnôt písmenám. Zrýchlením je, keď najskôr priradíme hodnoty písmenám nachádzajúcim sa na mieste jednotiek, potom na mieste desiatok atď., vtedy vieme ukončiť niektoré nevyhovujúce vetvy už na začiatku.

Na vytvorenie vhodných algebrogramov použijeme slovník. Náhodne z neho vyberieme slová, pridáme ku nim aritmetické operácie a vyskúšame, koľko má vzniknutý algebrogram riešení.

2111. Ak sa má ľavá strana rovnať pravej, musia sa v prvom rade rovnať ich dĺžky. Ak d je dĺžka riešenia a α_ℓ, p_ℓ, α_r, p_r je počet písmen a počet premenných na ľavej a pravej strane, musí platiť: $\alpha_\ell + dp_\ell = \alpha_r + dp_r$, odkiaľ $d = (\alpha_r - \alpha_\ell)/(p_\ell - p_r)$. Ak d nie je nezáporné celé číslo, rovnica nemá riešenie; špeciálny prípad $d = 0$ ošetríme samostatne. (Všimnite si, že dĺžka riešenia je najviac lineárna od dĺžky vstupu.)

Napíšme si teraz reťazce u a v pod seba a nájdime prvú pozíciu, na ktorej sú pod sebou premenná X a nejaké písmeno. Môžu nastať dva prípady: buď je na tejto pozícií d písmen za sebou – a tie tvoria jediné možné riešenie – alebo po k písmenách nasleduje premenná

X – vtedy sa v riešení týchto k písmen opakuje a opäť dostaneme jediného kandidáta na riešenie.

Ostáva už iba overiť, či tento kandidát naozaj funguje a nastáva rovnosť. To sa dá priamočiaro dosadením a porovnaním reťazcov; tie však môžu mať až kvadratickú dĺžku. Na lineárne riešenie potrebujeme vedieť pre každé i rýchlo povedať, či X a X posunuté o i sa zhodujú v prekrývajúcich sa $d-i$ znakoch.

Tabuľku týchto hodnôt vieme spočítať s pomocou algoritmu Knutha, Morrisa a Pratta. Tento algoritmus v lineárnom čase k reťazcu $x_1 \ldots x_d$ pre každé i spočíta najväčšie $P[i] < i$ také, že reťazec $x_1 \ldots x_i$ končí znakmi $x_1 \ldots x_{P[i]}$. Reťazec X sa potom zhoduje so sebou samým, ak ho posunieme o 0, $d-P[d]$, $d-P[P[d]]$, $d-P[P[P[d]]]$, …, a nezhoduje v ostatných prípadoch.

V lineárnom riešení stačí si zapamätať všetky posuny X, ktoré sa majú prekrývať a potom v lineárnom prechode tabuľkou overiť, či sa medzi hodnotami 0, $d-P[d]$, $d-P[P[d]]$, $d-P[P[P[d]]]$, … všetky tieto posuny nachádzajú.

2112. Zadaný strom si zakoreníme v ľubovoľnom vrchole. Prehľadávaním do hĺbky zistíme pre každý vrchol v počet $P[v]$ trpaslíkov, ktorí žijú v podstrome s koreňom v.

Optimálne umiestnenie obchodu môžeme nájsť nasledovne: Na začiatku ho umiestnime do koreňa. Čo sa stane, ak obchod presunieme z vrcholu u do niektorého jeho syna v? Ku $P[v]$ trpaslíkom bude teraz obchod o dĺžku dotyčnej cesty bližšie, ale všetci ostatní trpaslíci tam budú mať o toľko isto ďalej. Takýto presun sa teda oplatí len vtedy, ak je $P[v]$ väčšie ako polovica celkového počtu trpaslíkov. A takýto v je zjavne vždy nanajvýš jeden, preto stačí vyššie uvedený postup opakovať, kým nenájdeme optimálne umiestnenie obchodu. Dobrá implementácia má časovú zložitosť $O(n)$.

Všimnite si tiež zaujímavú skutočnosť: riešenie úlohy vôbec nezávisí na dĺžkach ciest medzi domčekmi.

2113. *Riešenie nadväzuje na text „Medián postupnosti“ na str. 243.*
Ak body na vstupe spĺňajú podmienku, ktorú máme overiť, tak sa musia dať popárovať do dvojíc, pričom priamka prechádzajúca cez každú z týchto dvojíc bodov prechádza (nám neznámym) bodom O, v ktorom leží druidské ohnisko.

Zoberme nejaký bod A, ktorý má zo všetkých $2n$ zadaných bodov najmenšiu x-ovú súradnicu. Ako k nemu nájsť jeho pár? Všimnime si, že všetky body ležia okolo A v jednej polrovine. Keby sme ich usporiadali podľa smeru z A do nich, B by bol n-tý – lebo priamka AB musí deliť ostatné body na dve rovnako veľké polovice. Na nájdenie bodu B samotné triedenie nepotrebujeme, stačí nájsť medián všetkých uhlov.

Tento istý postup zopakujeme s nejakým ďalším bodom A' a dostaneme jeho pár B'. (Môžeme napríklad ako A' použiť bod, pre ktorý bol uhol z A doň najmenší zo všetkých. Tento bod určite tiež leží na konvexnom obale a je rôzny od A a B.)

Nájdeme priesečník priamok AB a $A'B'$, čím dostaneme jediné možné ohnisko, a overíme, či toto ohnisko vyhovuje. Všetky časti tohto riešenia sa dajú implementovať s lineárnou časovou zložitosťou.

2114. Použijeme algoritmus, ktorý slúži na predspracovanie vyhľadávaného reťazca v algoritme Knutha, Morrisa a Pratta. V riešení úlohy 2111 definujeme hodnoty $P[i]$, ktoré tento algoritmus v lineárnom čase spočíta. Dĺžka najkratšej periódy je $n - P[n]$.

2115. Riešenie v čase $O(\log n)$ pomocou n počítačov: Výpočet bude prebiehať vo fázach, pričom po k fázach bude platiť, že na každom políčku v pamäti je súčet úseku, ktorý má dĺžku 2^k a končí uvedeným políčkom. Fáza k vyzerá nasledovne: počítač číslo k sa pozrie na pozíciu $k - 2^{k-1}$ a hodnotu z nej následne prirráta k hodnote na pozícii k.

Toto riešenie navyše vieme upraviť tak, aby celková práca (t.j. súčet počtov krokov všetkých počítačov) bola lineárna. Stačí pole rozdeliť na úseky dĺžky $\log n$. Najskôr naraz každý z nich spracujeme sekvenčne jedným počítačom. Toto vieme spraviť v logaritmickom

čase a celková práca v tejto časti je zjavne lineárna. Potom zoberieme súčty v jednotlivých úsekoch, teda $n/\log n$ hodnôt, a na ne použijeme predchádzajúci algoritmus. Keďže spracúvame menšie pole, dostaneme aj v tomto kroku celkovú prácu lineárnu od n. No a na záver každý z použitých $n/\log n$ počítačov prejde svoj úsek sekvenčne a pripočíta ku každej hodnote v jeho úseku súčet všetkých hodnôt v predchádzajúcich úsekoch.

2121. Od každého políčka odčítame d, čím získame jeho reálnu hodnotu. Vyskúšame všetky možnosti, v ktorom riadku hľadaný obdĺžnik začína. Pre každý konkrétny začiatočný riadok postupne skúšame všetky možnosti, v ktorom končí. Pre každý stĺpec si v pomocnom poli A pamätáme súčet tých jeho prvkov, ktoré ležia v skúmanom úseku riadkov. Vždy, keď prejdeme na nový koncový riadok, pripočítame ho k pamätaným hodnotám. Aby sme pre konkrétny začiatočný a koncový riadok našli najlepší obdĺžnik, stačí nájsť v poli A súvislý úsek s najväčším súčtom. To vieme spraviť v lineárnom čase (pozri úlohu z2111). Dostávame tak celkovú časovú zložitosť $O(r^2 s)$ a pamäťovú zložitosť $O(rs)$.

2122. Ak by sme nemali 25-centové mince, mohli by sme použiť jednoduchý pažravý algoritmus: použijeme čo najviac centov, potom čo najviac 5-centoviek a potom čo najviac 10-centoviek. Štvrťdoláre však robia problémy. Ak máme mince 1, 1, 1, 1, 1, 10, 10, 10, 25, tak sumu 30 je najlepšie zaplatiť pomocou 25-centovky a piatich 1-centoviek, zatiaľ čo náš pažravý algoritmus by použil tri 10-centovky.

Najväčšia suma, ktorú vieme zaplatiť bez pomoci 25-centových mincí, je $p + 5n + 10d$. Potrebujeme teda určite použiť aspoň $q_{min} = \lceil (K - p - 5n - 10d)/25 \rceil$ mincí s hodnotou 25 centov. Ak ich toľko nemáme, skončíme s chybou. Ak ich máme presne q_{min}, tak ich všetky použijeme a vyššie uvedeným postupom zistíme, či a ako najlepšie vieme zaplatiť zvyšok.

Všimnime si teraz, že pre $q \geq q_{min}$ platí: ak existuje riešenie používajúce 25-centových mincí $q + 2$, tak existuje aj riešenie, ktoré ich používa q, a toto riešenie má dokopy viac kusov mincí. Totiž pri riešení s $q + 2$ quartermi musia zostať nepoužité menšie mince v celkovej cene aspoň 50 centov. Z tých sa potom určite dá vybrať niekoľko so súčtom cien presne 50 centov, a tými môžeme nahradiť dve 25-centové mince.

Preto stačí pažravým algoritmom nájsť optimálne riešenie pre q_{min} a pre $q_{min} + 1$ použitých 25-centových mincí a vybrať lepšie z nich. Časová zložitosť tohto algoritmu je konštantná.

2123. Riešenie sa nezmení, ak nafúkneme stromy na kruhy s polomerom d a zároveň diaľnicu zúžime na priamku. Hľadanú priamku vieme charakterizovať uhlom, ktorý zviera s osou x. Budeme hľadať tento uhol α. Každý kruh zodpovedá dvom intervalom uhlov, ktorými nemôže priamka prechádzať. Stačí nám zobrať tieto intervaly a nájsť taký uhol α, ktorý v žiadnom intervale neleží. Po usporiadaní koncových bodov intervalov vieme v lineárnom čase zistiť, či takýto uhol existuje.

2124. Hľadané slovo označme $W[1..n]$. Sufix začínajúci na pozícii i označme $S[i]$. Ako prvý krok riešenia k zadanej permutácii $a = (a_1, \ldots, a_n)$ zostrojíme inverznú permutáciu $p = (p_1, \ldots, p_n)$. Jej význam je nasledovný: Sufix, ktorý je zo všetkých najmenší, začína p_1-tym písmenom slova W, druhý v poradí začína p_2-tym písmenom, atď. Dôsledkom toho je, že pre písmená slova W musí platiť $W[p_1] \leq W[p_2] \leq \cdots \leq W[p_n]$. Ak máme použiť čo najmenej rôznych písmen, potrebujeme, aby na čo najviac miestach platila rovnosť.

Budeme teraz postupne voliť písmená slova W. Na začiatku môžeme určite položiť $W[p_1]$ rovné 'a'. Postupne pre každé i teraz zistíme, či môže platiť $W[p_i] = W[p_{i+1}]$, alebo či musíme pre $W[p_{i+1}]$ použiť nové písmeno (o jedno ďalej v abecede).

Ako to zistiť? Musíme zabezpečiť, aby sufix $S[p_i]$ (začínajúci na pozícii p_i) bol menší ako sufix $S[p_{i+1}]$. Ak by sme mali $W[p_i] = W[p_{i+1}]$, tak by oba tieto sufixy začínali rovnakým písmenom. Potom ale ich porovnanie dopadne rovnako, ako keby sme porovnávali sufixy začínajúce o pozíciu ďalej – teda $S[p_i + 1]$ a $S[p_{i+1} + 1]$. My ale vieme, ako dopadne

porovnanie týchto sufixov: spomedzi všetkých sufixov je ten prvý na pozícii a_{p_i+1} a ten druhý na pozícii $a_{p_{i+1}+1}$.

Ak platí $a_{p_i+1} < a_{p_{i+1}+1}$, je všetko v poriadku – môžeme na pozíciu p_{i+1} slova w dať to isté písmeno ako máme na pozícii p_i a všetko bude fungovať. V opačnom prípade si správny výsledok porovnania našich sufixov musíme vynútiť tak, že na pozíciu p_{i+1} dáme o jedno väčšie písmeno ako na pozíciu p_i.

2125. Vytvoríme si v pamäti nový úsek dĺžky n, kde na pozíciu i zapíšeme 1 ak je i-te číslo na vstupe menšie ako x a 0 inak. Keď teraz pre túto postupnosť spočítame čiastočné súčty algoritmom z úlohy 2115, dostaneme pre každé číslo menšie ako x pozíciu, kam ho presunúť. Analogicky spočítame pozície pre čísla väčšie ako x, a na záver naraz uskutočníme všetky presuny (hodnotu x presunieme tesne za posledné číslo menšie ako x). Takto dostávame riešenie v čase $O(\log n)$ a pamäti $O(n)$. Podobným trikom ako v úlohe 2115 vieme tomuto riešeniu zlepšiť celkovú prácu z $O(n \log n)$ na $O(n)$.

2131. Vzorové riešenie má časovú zložitosť $O(n^2 \log n)$ a pamäťovú zložitosť $O(n)$. Je založené na nasledovnej myšlienke. Namiesto rovnice $a + b + c = d$ budeme riešiť rovnicu $a + b = d - c$. Stačí, ak si vygenerujeme všetky možné súčty a všetky možné rozdiely čísiel a nájdeme v nich prienik (napríklad usporiadaním a postupným prechádzaním). Tento postup vedie k ideálnej časovej zložitosti, ale k veľkej pamäťovej zložitosti ($O(n^2)$).

Namiesto vygenerovania všetkých súčtov si ich budeme generovať v rastúcom poradí (čo vlastne potrebujeme, aby sme zistili prienik). Ukážeme postup, ako generovať súčty v rastúcom poradí, rozdiely sa generujú analogicky. Usporiadame si prvky (nech sú po usporiadaní označené $a_1 < a_2 < \ldots < a_n$), do haldy vložíme prvky $a_i + a_1, 1 \leq i \leq n$. Budeme sa snažiť, aby sme na vrchu haldy mali najmenší ešte nevygenerovaný prvok. Vždy, keď vyberieme z haldy najmenší prvok $a_i + a_j$ (a zároveň ho tým vygenerujeme), tak do haldy vložíme najmenší väčší prvok obsahujúci a_i a to $a_i + a_{j+1}$.

2132. *Riešenie nadväzuje na text „Prehľadávanie do hĺbky“ na str. 246.*
Funguje pažravé riešenie. Kým máme nejaké hrany, tak dokola opakujeme: vyberieme ľubovoľný list, postavíme stánok na hrane z neho, a oba koncové vrcholy tejto hrany (spolu so všetkými hranami z nich) z grafu odstránime.

Dôkaz je založený na nasledujúcom pozorovaní: Predpokladajme, že sme si zvolili hranu uv, kde u je list a uvažujme *ľubovoľné* optimálne riešenie. Ak nie je stánok na hrane uv, musí byť na inej hrane z vrcholu v (inak by sme vedeli získať lepšie riešenie pridaním stánku na hranu uv). Ak tento stánok presunieme na hranu uv, dostaneme zjavne opäť korektné riešenie. Teda vždy existuje optimálne riešenie, ktoré má stánok na hrane uv.

Iné riešenie: Použijeme dynamické programovanie. Pre každý vrchol v spočítame, koľko najviac stánkov môže byť v podstrome s koreňom v, ak na žiadnej hrane z v neleží stánok, a koľko vtedy, ak na jednej z nich ležať môže. Obe riešenia sa dajú implementovať v čase $O(n)$ vhodnou úpravou prehľadávania do hĺbky.

2133. Zostrojíme najskôr pažravo čo najdlhšiu cestu: kým sa dá, ideme do vrcholu, kde sme ešte neboli. Keď sa zasekneme, platí, že posledný vrchol x našej cesty má všetkých svojich susedov na ceste.

Nech y je vrchol mimo cesty. Označme Y množinu jeho susedov. Označme X množinu susedov posledného vrcholu. Označme $\overline{X}$ množinu, ktorá vznikne tak, že pre každý vrchol z X do neho dáme jeho nasledovníka na ceste. Keďže $|Y| + |\overline{X}| > N$, existuje vrchol z, ktorý leží v oboch týchto množinách. Pôvodnú cestu $a, \ldots, b, z, \ldots, x$ teraz môžeme prerobiť na novú, dlhšiu cestu $a, \ldots, b, x, \ldots, z, y$.

Teraz znovu skúsime cestu pažravo predlžovať. Ak aj naďalej bude existovať nejaký vrchol mimo cesty, použijeme vyššie popísaný postup a cestu prerobíme. Takto striedame fázy pažravého predlžovania a prerábania cesty až kým zostrojíme cestu pána Hamiltona.

Ak posledný a prvý vrchol nesusedia, použijeme ešte raz vyššie uvedený postup, pričom za y zvolíme prvý vrchol cesty.

2134. Riešenie v čase kvadratickom od počtu stien: Pre lepšiu názornosť celé teleso posunieme tak, aby všetky jeho body mali kladnú výšku. Uvažujeme len vodorovné steny, ostatné nepotrebujeme. Stenu s plochou S ležiacu vo výške h spracujeme nasledovne: Zistíme, či vnútro nášho telesa leží nad alebo pod ňou. Toto vieme zistiť napríklad tak, že uvažujeme ľubovoľnú polpriamku idúcu z ľubovoľného bodu našej steny dodola a zistíme, či pretne povrch telesa párny alebo nepárny počet ráz. Ak vnútro telesa leží pod stenou, celkový obsah zvýšime o Sh, inak ho o Sh znížime.

Existujú aj efektívnejšie riešenia. Môžeme napríklad vo vodorovných rozmeroch spraviť kompresiu súradníc a potom pomocou dvojrozmerného intervalového stromu spracovať všetky plochy rovnobežné s vodorovnou rovinou usporiadané podľa výšky.

2135. Ukážeme riešenie v čase $O(\log n)$ s n^2 počítačmi a celkovou prácou $O(n^2)$: Pre každý prvok i si vyplníme pole dĺžky n, pričom na pozíciu j v tomto poli zapíšeme 1, ak je j-ty prvok menší ako i-ty. Súčet týchto n čísel vieme spočítať v logaritmickom čase (viď úlohu 2115) a ľahko nahliadneme, že tento súčet predstavuje pozíciu, na ktorej má prvok i byť v usporiadanom poli. Na záver teda súčasne presunieme všetky prvky na miesta, ktoré sme pre ne vypočítali.

Lepšie riešenie je založené na paralelizovanom merge-sorte. Spojiť dve utriedené postupnosti vieme najlepšie v čase $O(\log\log n)$. Pomocou neho vieme napísať algoritmus bežiaci v čase $O(\log n \log\log n)$ s celkovou prácou $O(n \log n)$. Existuje dokonca pomerene komplikovaný algoritmus triediaci v čase $O(\log n)$ s prácou $O(n \log n)$.[10]

2141. Zjavne nezáleží na poradí, v akom zmeny robíme – každý človek zmení svoj stav toľkokrát, v koľkých zvolených skupinách sa nachádza. Teraz vidíme, že každú skupinu sa oplatí vybrať najviac raz – keď ju vyberieme dvakrát po sebe, efekt je rovnaký ako keby sme ju nevybrali vôbec.

Pre každú skupinu i majme premennú x_i, ktorej chceme priradiť hodnotu 0 ak i-tu skupinu nevyberieme a 1 ak áno. Pre každého človeka j teraz dostávame podmienku: súčet x_i pre všetky skupiny obsahujúce človeka j musí byť párny, ak človek j teraz stojí, resp. nepárny, ak je v podrepe. Na tieto podmienky sa môžeme pozerať ako na sadu rovníc, pričom počítame modulo 2. Overiť, či existuje riešenie, a tiež nejaké nájsť vieme napríklad pomocou Gaussovej eliminačnej metódy.

2142. Úlohu budeme riešiť pomocou dynamického programovania. Nech $Z[i,j]$ je maximálny počet kilogramov zlata, ktorý sa dá vyťažiť počas prvých j hodín tak, že sa po j hodinách budeme nachádzať v bani i. Môžu nastať dve možnosti. Buď sme poslednú hodinu ťažili v bani i, vtedy $Z[i,j] = Z[i,j-1] + c_{i,j}$, alebo sme sa presunuli z bane k (čo nám trvalo d hodín) a vtedy $Z[i,j] = Z[k,j-d]$. Z tých možností vyberieme tú najlepšiu. Výsledok sa bude nachádzať v poslednom riadku tabuľky: je ním maximum z hodnôt $Z[i,t]$ pre $1 \le i \le n$. Toto riešenie sa dá implementovať v čase $O(t(n+m))$ a pamäti $O(nt)$.

2143. Jedno možné najlepšie riešenie vyzerá tak, že si vždy vyberieme niektorý kúsok čokolády a zlomíme ho pozdĺž ľubovoľnej najdrahšej hrany. Toto tvrdenie sa dá dokázať matematickou indukciou podľa veľkosti čokolády.

Keďže nám stačí spočítať cenu optimálneho rozlámania, môžeme si spomedzi možných optimálnych rozlámaní vybrať také, ktorého cenu ľahko spočítame. Toto spĺňa napríklad nasledujúci postup: Usporiadame všetky hrany podľa ceny. Potom ideme postupne od najdrahšej hrany k najlacnejšej a vždy pozdĺž aktuálnej hrany prelomíme všetky kúsky, cez ktoré prechádza.

---

[10] Podrobnosti nájde zvedavý čitateľ v 4. kapitole knihy Joseph JáJá, *An Introduction to Parallel Algorithms.*

Usporiadať hrany vieme v čase $\Theta((r+s)\log(r+s))$. Následné zistenie celkovej ceny lámaní vieme ľahko spraviť v lineárnom čase – počet prelomení zodpovedajúcich konkrétnej hrane je o jedna väčší ako počet na ňu kolmých hrán, ktoré sme už spracovali.

2144. Inverziou v permutácii voláme dvojicu prvkov $i < j$ takú, že j je v nej skôr ako i. Všimnite si, že výmena dvoch susedných prvkov permutácie vždy zmení počet inverzií o ± 1. Charakteristickou permutáciou pozície nazveme permutáciu n^2-1 čísel, ktorú dostaneme, keď vypíšeme všetky čísla po riadkoch (dieru preskočíme).

V našej hre pohyb vľavo ani vpravo charakteristickú permutáciu nezmení. Ak vykonáme pohyb hore alebo dole, dostaneme novú. Tú vieme z pôvodnej vyrobiť tak, že zoberieme prvok, ktorý sme hýbali, a $(n-1)$-krát ho vymeníme s jeho susedom.

Pre nepárne n sa teda parita počtu inverzií charakteristickej pozície nikdy nezmení. Pre párne n sa nikdy nezmení parita súčtu čísla riadku, kde je diera, a počtu inverzií charakteristickej permutácie. Nutnou podmienkou riešiteľnosti je teda, že táto hodnota v oboch prípadoch musí byť párna. Dá sa dokázať, že táto podmienka je aj postačujúca, teda každú pozíciu, ktorá ju spĺňa, naozaj vieme vyriešiť.

Počet inverzií v permutácii n^2-1 čísel vieme spočítať v čase $O(n^2 \log n)$ napríklad úpravou algoritmu merge-sort (pozri úlohu 1722).

Optimálne riešenie v $O(n^2)$: Ľubovoľná výmena dvoch prvkov permutácie zmení paritu počtu inverzii: výmenu prvkov vo vzdialenosti k vieme totiž naskladať z $2k-1$ výmen dvojice susedných prvkov, a každá z nich zmení paritu počtu inverzií. Aby sme určili paritu počtu inverzií danej permutácie, stačí ju teda pomocou výmen prvkov usporiadať a zistiť paritu počtu spravených výmen. A toto vieme spraviť v čase lineárnom od počtu prvkov pomocou rozkladu danej permutácie na cykly.

2145. Pre jednoduchosť predpokladajme, že všetky hrany majú navzájom rôzne ceny. (Toto ľahko dosiahneme tak, že hrany s rovnakou cenou umelo zoradíme podľa ich poradia na vstupe.) Potom platí, že keď z každého vrcholu vyznačíme najlacnejšiu hranu, tak existuje najlacnejšia kostra, ktorá všetky tieto hrany obsahuje. Sekvenčný algoritmus založený na tejto myšlienke ako prvý objavil Borůvka a išlo o vôbec prvý polynomiálny algoritmus na hľadanie najlacnejšej kostry.

Použijeme paralelnú verziu tohto algoritmu. Náš algoritmus bude striedavo vykonávať dve fázy. V prvej nájdeme v každom vrchole najlacnejšiu hranu. Tieto hrany vytvoria niekoľko komponentov. V druhej fáze každý komponent skontrahujeme do jedného vrcholu. Zjavne platí, že každým vykonaním druhej fázy sa počet vrcholov zmenší aspoň na polovicu, preto potrebný počet fáz bude $O(\log n)$.

Prvá fáza: Najskôr pre každý vrchol nájdeme v logaritmickom čase najlacnejšiu hranu, ktorá z neho vedie do iného komponentu. Následne pre každý komponent nájdeme v logaritmickom čase najlacnejšiu z takto vybratých hrán vedúcich z neho.

Kontrahovanie komponentov: Nech $D[i]$ je vrchol, do ktorého vedie z i vybratá hrana. Rozmyslite si, že postupnosť i, $D[i]$, $D[D[i]]$, ... sa časom zacyklí tak, že sa v nej budú striedať dva vrcholy – konce najlacnejšej hrany v komponente obsahujúcom vrchol i. Menší z týchto dvoch vrcholov nazveme reprezentantom dotyčného komponentu. Paralelne vieme ku každému vrcholu nájsť reprezentanta jeho komponentu nasledovne: položíme $P[i] = D[i]$, potom $(\log_2 n)$-krát vykonáme paralelne všetky priradenia $P[i] := P[P[i]]$, a na záver vyberieme pre každý vrchol menšiu z hodnôt $P[i]$ a $D[P[i]]$.

Celý algoritmus bude mať teda časovú zložitosť $O(\log^2 n)$, pamäťovú zložitosť $O(n^2)$ a počet počítačov $O(n^2)$. Ak by sme chceli dosiahnuť lepšiu prácu, bolo by navyše potrebné zahadzovať medzi fázami hrany vedúce vo vnútri komponentov.[11]

[11] Podrobne popísané riešenie nájde čitateľ napríklad v kapitole 5.2 knihy Joseph JáJá, *An Introduction to Parallel Algorithms*.

z2111. Nech a_i je i-ta zľava. Postupne pre každé k spočítame hodnotu $D[k]$: najlepšiu možnú zľavu, akú môže mať úsek zliav končiaci k-tou zľavou. Zjavne $D[1] = a_1$. Ak už poznáme $D[k]$, vieme ľahko spočítať $D[k+1]$ pomocou vzťahu $D[k+1] = a_{k+1} \cdot \min(D[k], 1)$. Ak máme totiž vyrobiť najlepší úsek zliav končiaci $(k+1)$-vou zľavou, určite musíme použiť aspoň tú. Okrem nej môžeme použiť aj niekoľko predchádzajúcich zliav. Už vieme, že najlepšia celková zľava, ktorú z nich vieme dostať, je $D[k]$. Ak $D[k] < 1$, oplatí sa nám tieto zľavy použiť, ak nie, zoberieme len zľavu a_{k+1} a nič iné.

Ak chceme aj zostrojiť optimálny úsek zliav, budeme pre každé k okrem hodnoty $D[k]$ počítať aj začiatok úseku zliav, pre ktorý sa nadobúda hodnota $D[k]$. A vždy, keď nájdeme doteraz najlepšiu celkovú zľavu, zapamätáme si aj interval, pre ktorý sme ju dostali.

Toto riešenie má zjavne optimálnu časovú zložitosť $O(n)$ a dá sa implementovať s konštantnou pamäťovou zložitosťou.

Poznámka: Keby sme každú zľavu a_i nahradili hodnotou $(-\log a_i)$, z úlohy „nájsť úsek s najmenším súčinom“ dostaneme známu úlohu „nájsť úsek s najväčším súčtom“. Táto bola v KSP použitá ako úloha 213.

z2112. Označme T bod, ktorý zostrojíme ako aritmetický priemer súradníc zadaných bodov. Body tvoria pravidelný n-uholník práve vtedy, ak platí, že z T je do nich rovnako ďaleko a sú okolo neho rovnomerne rozmiestnené.

Prvú podmienku vieme jednoducho overiť for-cyklom v čase lineárnom od n.

Druhú podmienku vieme overiť v čase $O(n \log n)$ tak, že body usporiadame podľa uhlu z bodu T do nich. Triedenie však nie je potrebné. Túto podmienku vieme overiť aj efektívnejšie. Označme si dané body $A_1, \ldots, A_n$. Potrebujeme vlastne overiť, či sa medzi uhlami A_1TA_i vyskytuje práve raz každý z uhlov tvaru $i \cdot 360°/n$. To vieme spraviť tak, že prejdeme cez všetky body, pre každý overíme, že príslušný uhol je jednej z požadovaných veľkostí a v pomocnom poli si zaznačíme, ktorý násobok $360°/n$ to bol. Ak sme na konci vyplnili celé pomocné pole, ide o pravidelný n-uholník, inak nie.

z2113. Stačí si pamätať posledných m čísel v cyklickom poli veľkosti m. Budeme mať jeden index k, ktorý označuje posledné miesto, kam sme do poľa vložili prvok. Vždy, keď dostaneme novú úlohu, tak k zvýšime a potom na k-te miesto zapíšeme číslo úlohy. Keď k presiahne veľkosť poľa, tak ho presunieme na začiatok poľa, čím začneme prepisovať staršie prvky. Všimnite si, že takto si vždy prepíšeme prvok, ktorý je v danej chvíli $(m+1)$-vý od konca, a teda ho už nebudeme potrebovať. Po spracovaní každého prvku si v poli pamätáme posledných m prvkov. Keď dočítame vstup, vypíšeme prvok, ktorý je v poli cyklicky za prvkom k. (Ešte musíme skontrolovať, či sme načítali aspoň m prvkov. V opačnom prípade totiž riešenie neexistuje.) Toto riešenie má časovú zložitosť lineárnu od počtu spracovaných prvkov, jeho pamäťová zložitosť je $O(m)$.

z2114. Onbloň sa dá reprezentovať ako graf. Onblká sú vrcholy, vetvičky sú hrany. V našom riešení najskôr ofarbíme vrcholy onblone zelenou a žltou tak, aby to sedelo s odpoveďami, ktoré dostávame od používateľa. Na konci už len zistíme, či sme dostali viac zelených alebo žltých vrcholov, a podľa toho zistíme, ktoré onblká sú v skutočnosti modré.

Ofarbovanie prebehne nasledovne: Na začiatku jeden vrchol ofarbíme zelenou. Hocikedy, keď poznáme farbu nejakého vrcholu, tak postupným kladením otázok vieme určiť farby všetkých vrcholov, s ktorými náš vrchol susedí. Môžeme teda na ofarbenie grafu použiť prehľadávanie (napríklad do šírky alebo hĺbky). Vždy, keď pri prehľadávaní prechádzame z vrcholu v_1, ktorého farbu poznáme, do vrcholu v_2, ktorého farbu nepoznáme, tak sa spýtame Amálky na vrcholy v_1 a v_2, čím zistíme farbu v_2.

Celkový počet položených otázok je nanavýš $n-1$, kde n je počet vrcholov. (Ak je v grafe viac ako $n-1$ hrán, na niektoré z nich sa nikdy neopýtame. Navyše ak už máme aspoň jeden vrchol každej farby, tak môžeme prestať v okamihu, keď počet vrcholov niektorej

farby prekročí $n/2$.) Algoritmus sa dá implementovať s časovou aj pamäťovou zložitosťou lineárnou od veľkosti vstupu.

z2115. Platí: $lala(a,b) = b - a$, $huhu(a,b) = \max(0, a - b)$; $dodo(a,b)$ zisťuje, či b delí a a $zuzu(a)$ vráti *true* práve vtedy, keď a je prvočíslo. Jediným vstupom, pre ktorý funkcia *body* vráti 15 bodov, je 5^{13}.

z2121. Každým vyrobením a vyfajčením cigarety Zachariášovi ubudne $k-1$ špakov. Preto stačí n celočíselne vydeliť $k-1$. Avšak ak mu na konci ostane presne $k-1$ špakov, tak už žiadnu cigaretu nevyrobí. Preto ak pri delení dostaneme zvyšok 0, je ešte potrebné výsledok znížiť o 1.

Číslo n stačí čítať po cifrách a deliť ho tak, ako sa učí na základnej škole. Dostaneme tým časovú zložitosť $O(\log n)$ (teda lineárnu od počtu cifier čísla n).

Riešenie prémie: Zachariáš išiel za svojim kamarátom bezdomovcom Zoltánom a požičal si jeden špak. Tak si mohol vyrobiť aj poslednú piatu cigaretu. Keď ju dofajčil, špak z nej vrátil späť Zoltánovi.

z2122. *Riešenie nadväzuje na text „Prehľadávanie do hĺbky“ na str. 246.*
Úlohou je nájsť topologické usporiadanie vrcholov orientovaného grafu.

z2123. Platí, že i-temu najnižšiemu lyžiarovi treba dať i-te najkratšie lyže. Dôkaz sa dá založiť napríklad na nasledovnej myšlienke: Majme riešenie, v ktorom ľudia a, b s výškami $v_a \le v_b$ majú lyže s dĺžkou $\ell_a \ge \ell_b$. Ak im vymeníme lyže, tak dostaneme riešenie, ktoré je aspoň rovnako dobré ako pôvodné.

Časová zložitosť je taká, ako časová zložitosť použitého triedenia, takže v prípade použitia quick-sortu alebo iného rýchleho triedenia dostávame časovú zložitosť $O(n \log n)$ a lineárnu pamäťovú zložitosť.

z2124. Použijeme backtracking. Budeme mať rekurzívnu procedúru $H(r,c)$, pričom jej vstupné parametre znamenajú, že ak by som po riadkoch kontroloval všetky políčka šachovnice, tak (r,c) bude prvé, ktoré ešte nie je ohrozené žiadnym koňom. Procedúra H vyskúša všetkých 9 možností ako ho ohroziť a zakaždým sa rekurzívne zavolá na (v danej situácii) nasledujúce neohrozené políčko.

Pre klasickú šachovnicu 8×8 stačí 12 koní. V súčasnosti sú známe[12] optimálne riešenia pre šachovnice od 1×1 po 20×20.

z2125. Riešená úloha bola nájsť najkratšiu cestu v ohodnotenom grafe.

Prvé riešenie používa backtracking na postupné vyskúšanie všetkých ciest. Má exponenciálnu časovú zložitosť a pravdepodobne by dostalo 5 až 8 bodov.

Druhé riešenie používa heuristiku „v každom kroku si spomedzi nenavštívených vrcholov vyber ten, do ktorého vedie z aktuálneho vrcholu najkratšia hrana“. Už pre štyri vrcholy ľahko zostrojíme graf, pre ktorý takýto prístup nenájde najkratšiu cestu. Program v zadaní navyše obsahuje syntaktickú chybu, a tak by si zaslúžil nanajvýš 2 body.

Tretie riešenie je implementáciou Bellmanovho-Fordovho algoritmu. Po k-tom volaní procedúry `skus` platí, že `naj[x]` je nanajvýš rovné dĺžke najkratšej cesty z 1 do `x`, ak smieme použiť nanajvýš k hrán. Preto po $n-1$ volaniach procedúry `skus` budeme mať zostrojené dĺžky najkratších ciest.

V programe je drobná chyba, keďže `skus` volá len $n-2$ ráz. Navyše časová zložitosť tohto riešenia nie je optimálna. Preto by pravdepodobne získalo okolo 11 bodov.

z2131. Na nájdenie bohatiera použijeme binárne vyhľadávanie. V každej iterácii vyhľadávania si zistíme výšky dvoch susedných bohatierov stojacich približne v strede aktuálneho úseku. V tej polovici, ktorá obsahuje vyššieho z nich, sa určite niekto hrozivo týči. (Rozmyslite si, prečo.) Takto v každej iterácii skrátime skúmaný úsek približne na polovicu. Časová zložitosť tohto riešenia je teda $O(\log n)$, pamäťová je konštantná.

[12] Pozri `http://www.research.att.com/~njas/sequences/A006075`.

Na zamyslenie: Predstavte si, že hráte túto hru namiesto princeznej. Výšky bohatierov, na ktorých sa komorná pýta, si môžete vymýšľať aké len chcete. Pritom chcete komornú donútiť položiť čo najviac otázok. Akú stratégiu použijete?

z2132. Optimálna výška zjavne leží v intervale od 0 po h. Funkcia $f(x)$, ktorá udáva celkovú cenu stavby v závislosti od výšky x, v ktorej staviame, je kvadratickou funkciou od x. Zo vstupných údajov vieme v lineárnom čase spočítať koeficienty tejto funkcie a z nich v konštantnom čase určiť, kde má minimum.

z2133. Postupne spracúvame inštrukcie, pamätáme si smer, ktorým je Zeofína natočená a pomocou goniometrických funkcií zisťujeme nové súradnice. Keď takto zostrojíme celý mnohouholník $A_1 \dots A_n$, jeho obsah spočítame napríklad pomocou vektorových súčinov – t.j. ako súčet orientovaných obsahov n trojuholníkov OA_iA_{i+1}.

z2134. Túto úlohu budeme riešiť metódou dynamického programovania. Nech $J[i,j]$ je počet jahôd, ktoré sa nachádzajú na poli na pozícii $[i,j]$. Budeme si počítať pre každú pozíciu $[i,j]$ maximálny počet jahôd $P[i,j]$, ktoré Janko vie zožrať po ceste z $[1,1]$ na $[i,j]$. Hodnoty $P[i,j]$ vieme počítať pomocou vzťahu:

$$P[i,j] = J[i,j] + \max(P[i-2,j-1], P[i-2,j+1], P[i-1,j-2], P[i+1,j-2]).$$

Aby sme pri počítaní $P[i,j]$ poznali všetky potrebné hodnoty, je potrebné spracúvať pole po diagonálach. Časová zložitosť tohto riešenia je $O(rs)$. Kedže si stačí pamätať najviac tri posledné diagonály, pamäťová zložitosť potrebná na náš výpočet je $O(r+s)$.

z2135. V zadaní sú ukryté dokopy štyri správy. Najľahšie je asi nájsť zvýraznené písmenká v zadaniach tejto série KSP-Z (úlohy z2131–z2135), ktoré dokopy tvoria text: „heslo je hruška“. Prvé písmená viet zadania „Zo sveta steganografie“ ukrývajú oznam „akcia začína o polnoci“. Zhluk písmen na konci zadania je stereogram obsahujúci tretiu správu – číslo 47. Posledná správa sa ukrýva v obrázku, na ktorý ukazuje linka ukrytá v záhlaví strany, na ktorej začína zadanie. Na obrázku sú zdanlivo stromy, avšak keď z každého políčka vyberieme len posledné dva bity každej zložky farby, dostaneme obrázok mačky.

z2141. Po načítaní čísla k si predpočítame prvých k prvočísel. To vieme spraviť napríklad pomocou Eratostenovho sita, prípadne tak, že kým nenájdeme dosť prvočísel, tak postupne skúšame prirodzené čísla a každé skúsime deliť číslami od 2 po jeho odmocninu.

Pre každé okienko si budeme pamätať, či je práve obsadené a kto pri ňom naposledy bol. Keď príde nový občan, skontrolujeme, či môže ísť priamo k nejakému voľnému okienku. Ak áno, rovno ho tam pošleme, ak nie, bude čakať.

Keď sa nejaké okienko uvoľní, vieme číslo občana, ktorý tu bol posledný. Túto informáciu využijeme na to, aby sme nemuseli vždy prechádzať všetkých čakajúcich. Namiesto toho len prejdeme cez všetky pozície, ktoré sú násobkami čísla okienka, až kým buď nenájdeme čakajúceho občana, alebo neprekročíme aktuálny počet občanov n. Pre každé okienko sa takto najviac raz pozrieme na každého občana, preto je celková časová zložitosť spracovania n občanov rovná $O(kn)$.

z2142. *Riešenie nadväzuje na text „Euklidov algoritmus“ na str. 244.*

Všimnime si, že žabky skáču v cykloch. Každá žabka sa bude nachádzať na začiatočnej pozícii vždy po preskákaní nejakého násobku dĺžky cyklu, v ktorom sa nachádza. Preto si najskôr zistíme dĺžky všetkých cyklov, v ktorých žabky skáču. Riešením je potom zjavne najmenší spoločný násobok týchto dĺžok.

z2143. Každá cesta medzi dvoma zastávkami nám hovorí, že ňou spojené zastávky musia ležať na rôznych brehoch rieky. O jednej zastávke prehlásime, že leží na prvom brehu. Potom z nej spustíme prehľadávanie do hĺbky alebo šírky. Počas neho o každej objavenej zastávke vieme povedať, na ktorom brehu musí ležať – na opačnom ako tá, z ktorej sme práve prišli. Ak niekedy stretneme cestu, ktorej obe koncové zastávky sme už spracovali a majú ležať na tom istom brehu, skončíme so zápornou odpoveďou. Naopak, ak

prehľadávanie úspešne skončí, tak sme práve našli prípustné rozdelenie zastávok na brehy rieky. (Ak má cestná sieť viac komponentov súvislosti, treba tento algoritmus spustiť raz pre každý komponent.)

V odbornej literatúre sa táto úloha volá testovanie bipartitnosti grafu. Časová aj pamäťová zložitosť nášho riešenia je lineárna od veľkosti vstupu.

z2144. Všimnime si, že ak si zvislé/vodorovné čiary očíslujeme od 0, tak každá čiara s číslom, ktoré ma zvyšok 2 po delení číslom 3 bude celá plná. Na ostatných je vzorka „jeden úsek prázdny, dva plné“. Obrázok najskôr vykresľujeme po vodorovných čiarach a potom po zvislých. Časová zložitosť je $O(mn)$, pamäťová je konštantná.

Otázka na zamyslenie: Viete nájsť riešenie, ktoré minimalizuje celkovú vzdialenosť, ktorú Zeofína prejde?

z2145. Asi najlepšie riešenie tejto úlohy je level ručne zostrojiť. Najlepšie bludisko, ktoré našiel jeden z riešiteľov, malo 150-ťahové riešenie:

```
 ########
 #      #
## $ $ #
#. ##$##
#. ##@ #
#.   # #
#  #   #
#########
```

Pre tri bedničky je priestor stavov natoľko malý, že dĺžku optimálneho riešenia vieme hľadať prehľadávaním do šírky na grafe, ktorého vrcholy zodpovedajú pozíciám v hre. Takto sme aj vyhodnocovali odovzdané levely.

2211. Výsledný stav ohňa záleží len od parity počtu zmien, ktoré naň majú vplyv. Taktiež nezáleží na poradí, v akom zmeny robíme, a nemá zmysel robiť tú istú zmenu dvakrát.

Po týchto pozorovaniach je už optimálne riešenie jednoznačne zostrojiteľné. Keď totiž začneme na políčku $[1,1]$ a postupne budeme prechádzať po riadkoch, vždy, keď na políčko prídeme, máme poslednú šancu zmeniť jeho stav – a teda ak na ňom práve nehorí oheň, musíme ho prepnúť. Simuláciou prepínaní dostávame riešenie s časovou zložitosťou $O(r^2s^2)$.

Jedno možné lepšie riešenie: Pre každý stĺpec s si budeme pamätať počet $P[s]$ prepnutí ohňa, ktoré sme už v stĺpci s spravili. Vstup budeme spracúvať po riadkoch. Spracovanie každého riadku vyzerá nasledovne: Vieme, že prvý oheň v ňom zmenil stav $P[1]$-krát. Z toho určíme jeho aktuálny stav. Ak je vypnutý, prepneme ho a zvýšime $P[1]$. Druhý oheň doteraz zmenil stav $(P[1]+P[2])$-krát. Ak je vypnutý, prepneme ho a zvýšime $P[2]$. Analogicky pokračujeme ďalej, pričom si v pomocnej premennej udržiavame súčet $P[1]+\cdots+P[i]$. Toto riešenie má optimálnu časovú zložitosť $O(rs)$ a pamäťovú zložitosť $O(s)$.

2212. *Riešenie nadväzuje na text „Dijkstrov algoritmus“ na str. 248.*
Klasický Dijkstrov algoritmus upravíme nasledovne: Vždy, keď spracúvame hrany vychádzajúce z nejakého vrcholu v, tak ako čas odchodu z v neberieme čas príchodu doň, ale až najbližší čas, kedy máme zelenú.

2213. Úlohu budeme riešiť dynamickým programovaním. Označme si $a_1a_2\ldots a_n$ jednotlivé písmená mena prapapagája; $P[i,j]$ počet palindrómov, ktoré možno dostať z reťazca $a_ia_{i+1}\ldots a_j$; $Q[i,j]$ počet palindrómov, ktoré možno dostať z reťazca $a_ia_{i+1}\ldots a_j$ a začínajú sa písmenom a_i. Zjavne platí $P[i,i]=Q[i,i]=1$ pre všetky $1\le i\le n$. Palindrómy, ktoré možno dostať z reťazca $a_ia_{i+1}\ldots a_j$ rozdelíme na dve skupiny: tie, ktoré sa začínajú písmenom a_i (je ich $Q[i,j]$) a tie, ktoré sa nezačínajú písmenom a_i (je ich $P[i+1,j]$). Preto $P[i,j]=Q[i,j]+P[i+1,j]$.

Teraz vyjadrime počet palindrómov, ktoré možno dostať z reťazca $a_ia_{i+1}\ldots a_j$ a začínajú sa písmenom a_i. Ak $a_i\ne a_j$, potom nemožno použiť písmeno a_j, preto ich je $Q[i,j-1]$.

V opačnom prípade si ich znova rozdelíme na dve skupiny: tie, ktoré sa končia písmenom a_j (je ich $1 + P[i+1, j-1]$, ten jeden navyše je palindróm $a_i a_j$) a tie, ktoré sa nekončia písmenom a_j (je ich $Q[i, j-1]$). Vtedy teda platí $Q[i,j] = 1 + P[i+1, j-1] + Q[i, j-1]$.

Časová zložitosť výpočtu všetkých hodnôt $P[i,j]$ a $Q[i,j]$ je $O(n^2)$. Vystačíme si s pamäťou veľkosti $O(n)$.

2214. Použijeme písmenkový strom. Z každého slova zo slovníka vyrobíme nové slovo, obsahujúce tie isté písmená ale usporiadané podľa abecedy. V písmenkovom strome nájdeme vrchol zodpovedajúci tomuto novému slovu a v tomto vrchole si pôvodné slovo zapamätáme. Potom pre každé kolo hry usporiadame aj písmená, ktoré si Miško vytiahol, a získaný reťazec nájdeme v písmenkovom strome. Vo vrchole, ktorý sme takto našli, máme zapamätané všetky slová, ktoré vie z daných písmen Miško postaviť. Abeceda má konštantnú veľkosť, preto môžeme písmená v slovách triediť algoritmom count-sort v čase $O(k)$. Celková časová zložitosť je $O((n+m) \cdot k + b)$, kde b je veľkosť výstupu.

2215. Nadpis zadania sú slová „Taje kryptológie“, každé písmeno je v abecede posunuté o päť miest.

V podúlohe a) akonáhle sa stretne k ľudí, tak poznajú k hodnôt polynómu stupňa menšieho ako k. Dá sa ľahko dokázať, že takýto polynóm je vždy práve jeden. (V riešení úlohy 2041 je vysvetlené, ako ho vieme zostrojiť.)

V podúlohe b) dokážeme, že každé heslo je rovnako pravdepodobné. Zjavne to stačí dokázať pre $k-1$ ľudí. Keď si teraz zvolíme, aké heslo chceme aby bolo správne, budeme mať už k hodnôt polynómu ($k-1$ hodnôt od ľudí a $f(0)$ bude naše zvolené heslo), a ten nimi bude jednoznačne určený. Existuje teda presne p polynómov, ktoré zodpovedajú tomu, čo našich $k-1$ ľudí vie, a každý z nich zodpovedá inému heslu.

Podúloha c): Nech sa stretlo $k-1$ čestných ľudí a jeden nečestný. Ten nečestný namiesto svojej hodnoty zverejní náhodnú nesprávnu. Následne spolu vypočítajú nesprávne heslo a trezor sa im otvoriť nepodarí. Útočník už ale pozná všetky čísla ostatných ľudí, a teda vie neskôr spočítať správne heslo. Ostatní vedia len to, že niekto sa pomýlil alebo klamal.

(Útočníkovi stačí vedieť jeho poradové číslo u, jeho hodnotu t_u, hodnotu t'_u, ktorú oznámil ostatným, zle spočítané heslo h' a poradové čísla ostatných $k-1$ účastníkov – nepotrebuje teda vedieť ich hodnoty t_i! Takýto útok by fungoval aj vtedy, ak by napríklad postupne po jednom zadávali tajne svoje hodnoty do počítača a ten im nakoniec oznámil vypočítané heslo.)

2221. Označme a_i prvky postupnosti a spočítajme si čiastočné súčty $P[i] = a_1 + a_2 + \cdots + a_i$ (špeciálne $P[0] = 0$). Pre ľubovoľné $i \le j$ teraz platí, že hodnota $P[j] - P[i]$ je rovná súčtu $a_{i+1} + \cdots + a_{j-1} + a_j$. Z toho vyplýva, že hodnoty $|P[i] - P[j]|$ aj $|P[j] - P[i]|$ sú obe rovné absolútnej hodnote súčtu dotyčného úseku postupnosti. Potrebujeme teda nájsť ľubovoľné dva indexy i a j také, aby bol rozdiel $P[i]$ a $P[j]$ čo najbližší k číslu t.

Prefixové súčty usporiadame podľa veľkosti, označme ich $b_0, \ldots, b_n$. Budeme mať dva indexy: i bude vždy ukazovať na to menšie číslo a j na to väčšie. Na začiatku je $i = 0$ a $j = 1$. Ak je $b_j - b_i < t$, zväčšíme j; v opačnom prípade zväčšíme i. Takto v každom kroku posunieme jeden z indexov, kým neprejdeme celým zoznamom. Počas toho si pamätáme zatiaľ najlepšiu dvojicu. Časová zložitosť je kvôli triedeniu $O(n \log n)$.

2222. *Riešenie nadväzuje na text „Prehľadávanie do hĺbky“ na str. 246.*
Vstup si reprezentujme orientovaným grafom: vrcholmi budú kamaráti a hrana z u do v bude znamenať, že u požaduje, aby bol s ním pozvaný aj v. Všimnime si, že spolu s ľubovoľným kamarátom musíme pozvať celý silne súvislý komponent S, do ktorého on patrí, a taktiež všetky komponenty, do ktorých sa dá z S dostať po hranách. Optimálne riešenie je teda nájsť tie silne súvislé komponenty, z ktorých nevedú žiadne hrany, a z nich vybrať najmenší. Na hľadanie silne súvislých komponentov použijeme Tarjanov algoritmus založený na prehľadávaní do hĺbky.

2223. Definujme si, že z dvoch stromov je strmší ten, ktorého uhol je bližšie k 90°. Potom platí, že strom nemôže naraziť do menej strmých ako je on sám. Keď strom rastie kolmo nahor, nemôže ho žiadny iný zastaviť. Keď strom rastie doľava, zastaví ho iba strmší naľavo od neho; podobne pre strom rastúci doprava.

Usporiadame stromy podľa x-ovej súradnice. Najprv ich prejdeme zľava, pričom si pamätáme zatiaľ najstrmší strom k. Všetky stromy, ktoré by do aktuálne pamätaného časom narazili, označíme, lebo vieme, že určite do nekonečna neporastú. Potom tento postup zopakujeme z pravej strany. Nakoniec vypíšeme neoznačené stromy. Časová zložitosť je $O(n \log n)$.

2224. Zadaná cestná sieť tvorí strom. Pre lepšiu predstavu si tento strom zakoreňme v ľubovoľnom vrchole. Všimnime si, že v zakorenenom strome každá (a teda aj najdlhšia) cesta vedie najprv niekoľkými (aj žiadnymi) hranami dohora, potom sa v nejakom vrchole „otočí“ a následne vedie zvyšnými (opäť možno aj žiadnymi) hranami dodola.

Hľadanie najdlhšej cesty budeme riešiť spracúvaním vrcholov stromu zdola nahor. Postupne pre každý vrchol v nájdeme dĺžku d_v najdlhšej cesty, ktorá sa v ňom otáča. Riešením úlohy je potom najväčšia z týchto dĺžok.

Aby sme tieto hodnoty vedeli efektívne počítať, budeme ešte pre každý vrchol v počítať aj dĺžku b_v najdlhšej cesty vedúcej z neho dodola. Pre každý list je táto dĺžka 0. Pre nelistový vrchol je to maximum z 0 a všetkých hodnôt $\ell_{vx} + b_x$, kde x je syn vrcholu v a ℓ_{vx} je dĺžka hrany vx.

Keď sme už spracovali všetkých synov vrcholu v a poznáme hodnotu b_v, môžeme spočítať hodnotu d_v: buď $d_v = b_v$, alebo vieme vo v cestu rozdeliť na dva nezávislé neprázdne úseky. Dĺžku najdlhšej takejto cesty zistíme tak, že pre každého syna vrcholu x zoberieme hodnotu $\ell_{vx} + b_x$ a z týchto hodnôt zoberieme dve najväčšie.

Najjednoduchšia implementácia tohto algoritmu: Z vrcholu, ktorý zvolíme za koreň, spustíme prehľadávanie do hĺbky. Všetky veci uvedené v predchádzajúcich dvoch odsekoch vieme počítať priamo počas prehľadávania, keďže v okamihu, keď prehľadávanie opúšťa konkrétny vrchol, boli už spracovaní všetci jeho synovia. Takéto riešenie má optimálnu časovú zložitosť $O(n)$.

2225. V časti a) môžeme začať tým, že náhodne vygenerujeme všetky Vladove voľby. Ak chcel Vlado vidieť, či Paľo zverejnil graf G, vyrobíme súbory s náhodným prečíslovaním grafu G. V opačnom prípade vyrobíme súbory s kružnicou dĺžky n a náhodne doplníme ďalšie hrany tak, aby sedel ich počet.

Riešenie hintu: Farboslepému človeku dáme kartičky do rúk, povieme mu ktorá je ktorá a necháme ho, nech nám dokola ukazuje ktorú chce – a zakaždým mu o nej povieme to isté ako na začiatku. Analogicky riešime časť b): Vlado si vyberie jeden z G_1 a G_2, náhodne prečísluje vrcholy a pošle ho Paľovi. Ten zistí, či to bol G_1 alebo G_2 a odpovie. Ak G_1 a G_2 nie sú izomorfné, Paľo vie Vladovi vždy jednoznačne odpovedať a presvedčí ho. Ak izomorfné sú, má v každom kole len 50% šancu presvedčiť ho.

V časti c) môžeme postupovať podobne: Paľo náhodne prečísluje vrcholy jedného z daných dvoch grafov (keďže sú izomorfné, je jedno, ktorého) a výsledok zverejní. Vlado si vyberie G_1 alebo G_2 a Paľo mu následne ukáže izomorfizmus grafu čo on zverejnil a grafu čo si Vlado vybral.

2231. Pre $x \leq n - 2$ platí, že v riadku prislúchajúcom x-tej najbližšej hviezde je $n - x$ výskytov jej vzdialenosti od nás (jeden výskyt pre každú hviezdu, ktorá je od nás ďalej) a $x - 1$ menších, navzájom rôznych hodnôt (vzdialenosti hviezd, ktoré sú k nám bližšie). Na nájdenie vzdialenosti k hviezde číslo k teda prejdeme jej riadok, nájdeme najväčšiu hodnotu a zistíme, koľkokrát sa tam táto hodnota nachádza. Ak len raz, hviezda číslo k je jednou z dvoch najvzdialenejších, a vtedy jej presnú vzdialenosť od nás určiť nevieme.

2232. V čase $O((n+m)\cdot k)$ vieme úlohu riešiť dynamickým programovaním: postupne pre každé $v \le k$ spočítame pre každé n počet $P[n,v]$ sledov dĺžky v, ktoré končia vo vrchole n. Hodnotu $P[n,v]$ zistíme ako súčet tých $P[x,v-1]$, pre ktoré z x do n vedie hrana.

Lepšie riešenie pre veľké k je založené na násobení matíc. Ľahko overíme, že keď zoberieme maticu susednosti pôvodného grafu a umocníme ju na k-tu, dostaneme na políčku $[x,y]$ v novej matici počet sledov z x do y dĺžky k. Stačí teda vypočítať k-tu mocninu matice susednosti a v nej sčítať všetky prvky. Šikovný výpočet k-tej mocniny sa dá urobiť pomocou vzťahu $A^{2x} = (A^x)^2$. Stačí nám teda $O(\log k)$ násobení matíc a každé z nich vieme spraviť v čase $O(n^3)$. (Existujú ale aj efektívnejšie algoritmy na násobenie matíc.)

Iný pohľad na toto riešenie: ak pre každú dvojicu vrcholov a,b poznáme počet sledov medzi nimi, ktoré majú dĺžku x, ľahko pre každú dvojicu určíme počet sledov, ktoré majú dĺžku $2x$ – samostatne pre každé c zistíme počet sledov, ktoré po x krokoch boli v c, a tieto počty sčítame. Vzťah, ktorý takto dostaneme, presne zodpovedá tomu, čo počítame pri násobení matíc.

Pri riešení netreba zabúdať na to, že korektné riešenie pre veľké k by malo byť schopné pracovať s veľkými číslami, keďže počet sledov môže rásť až exponenciálne od k.

2233. Rozostavenia mrakodrapov predstavujú rôzne partície čísla m na práve n sčítancov. Pri hľadaní k-tej takejto partície v lexikografickom poradí postupujeme rovnako, ako vo všetkých podobných kombinatorických úlohách: budeme ju zostrojovať postupne „zľava doprava“.

Označme $P[a,b]$ počet rozostavení b kociek do a mrakodrapov. Ako túto hodnotu zistiť? Všetky rozostavenia môžeme rozdeliť na dve disjunktné skupiny: tie, kde má prvý mrakodrap veľkosť 1, a tie, kde je vyšší. V prvej skupine je presne $P[a-1,b-1]$ rozostavení, lebo zvyšných $b-1$ kociek musíme rozostaviť do zvyšných $a-1$ stĺpcov. V druhej skupine je $P[a,b-a]$ rozostavení – takýmto rozostaveniam totiž vieme jednoznačne priradiť všetky možné rozostavenia $b-a$ kociek do a stĺpcov. (Stačí z každého stĺpca odstrániť jednu kocku.)

Pomocou vzťahu $P[a,b] = P[a-1,b-1] + P[a,b-a]$ vieme všetky hodnoty po $P[n,m]$ spočítať v čase $O(mn)$. Pomocou týchto hodnôt teraz ľahko zostrojíme k-te rozostavenie: ak $P[n-1,m-1] \le k$, vypíšeme 1 a zostrojíme k-te spomedzi rozostavení $m-1$ kociek do $n-1$ stĺpcov, inak zostrojíme rozostavenie $m-n$ kociek do n stĺpcov s poradovým číslom $(k - P[n-1,m-1])$ a následne všetky stĺpce v tomto rozostavení o 1 zvýšime.

2234. Otestovať, či sa dve konkrétne parcely nepretínajú, vieme ľahko v konštantnom čase postupom rovnakým ako v 2D – prienik s nenulovým hyperobjemom majú práve vtedy, ak je v každom rozmere prienik ich priemetov interval nenulovej dĺžky. V čase $O(n^2)$ teda vieme overiť, či sa žiadne dve zo zadaných n parciel neprekrývajú. Zvyšok úlohy vyriešime jednoduchým trikom. Ak sa žiadne dve hyperparcely neprekrývajú, stačí nám spočítať súčet ich hyperobjemov a porovnať ho s hyperobjemom celého hyperkvádra.

2235. Časť a) vyplýva z asociatívnosti operácie xor – teda výsledok nezáleží na poradí, v akom jednotlivé xorovania spravíme. Platí, že (správa $\oplus$ kľúč) $\oplus$ kľúč je to isté ako správa $\oplus$ (kľúč $\oplus$ kľúč). Preto po dvoch prexorovaniach tým istým kľúčom dostaneme pôvodnú správu.

V časti b) hľadaným kľúčom je jednoducho $S \oplus M'$.

Poštárkou zachytené správy v časti c) sú $S_1 = M \oplus K_A$, $S_2 = M \oplus K_A \oplus K_B$ a $S_3 = M \oplus K_B$. Zjavne $S_1 \oplus S_2 \oplus S_3 = M$, takže poštárke stačí všetky tri správy prexorovať a získa text správy: „`Strc prst skrz krk!`“. (Navyše môže poštárka dopočítať K_B a napríklad namiesto poslednej správy Bobovi podstrčiť vlastnú zašifrovanú týmto kľúčom.)

V časti d) môže Smith napríklad použiť tzv. *man in the middle* útok. Po ceste odchytí Quileniovu zamknutú truhlicu, umiestni na ňu svoj zámok a pošle ju späť. Nič netušiaci Quilenius dá dole svoj zámok, opäť pošle truhlicu a Smith si ju odchytí a hravo ju otvorí,

lebo už je na nej len jeho zámok. Medzičasom môže Smith urobiť (s inou truhlicou) prvé dva kroky komunikácie s Jonesom, aby tomu nebolo podozrivé, že mu nijaká truhlica neprišla.

(Uvedomte si, že podobný problém by vznikol vždy – aj keby sme takto používali šifry iné ako xor, ktorého slabinu sme si ukázali v úlohe c). Aktívny útočník by sa vždy mohol vložiť doprostred komunikácie a oklamať oboch jej účastníkov.)

Prexorovaním správ $A \oplus K$ a $B \oplus K$, ktoré máme v časti e), dostávame správu $A \oplus B$, teda sme sa zbavili kľúča K. Z $A \oplus B$ už za predpokladu, že A aj B sú slovenské texty, vieme obe správy rozumne ľahko ručne zostrojiť. (Všimneme si napríklad, že xorom dvoch písmeniek dostávame relatívne malú hodnotu, zatiaľ čo xorom písmenka a medzery hodnotu väčšiu.) Zašifrované správy boli „`Ahoj moja najdrahsia, toto nikto nerozlusti.`“ a „`Absolutne bezpecna sifra odola vasim utokom!`“.

2241. Zadané pole si usporiadame podľa veľkosti. Keď si teraz zvolíme, že hľadáme postupnosť dĺžky $k+1$, je tým jednoznačne určené, že musí mať diferenciu $d = a/k$. Nájsť v poli dostatočne dlhú postupnosť s danou diferenciou d už vieme v lineárnom čase: Pomocou dvoch ukazovateľov vieme v lineárnom čase nájsť všetky dvojice prvkov, ktoré sa líšia presne o d. Aby sme našli $(k+1)$-člennú postupnosť, musí niektorých k z týchto dvojíc na seba postupne nadväzovať. Takto dostávame riešenie s časovou zložitosťou $O(n^2)$.

Lepšie riešenie: Všimnite si, že netreba skúšať všetky možné k. Ak totiž pre nejaké $k = k_1 k_2$ existuje vyhovujúca postupnosť π s diferenciou $a/(k_1 k_2)$, tak určite existuje aj vyhovujúca postupnosť s diferenciou a/k_1: stačí vybrať každý k_2-ty člen postupnosti π. Ak teda existuje aspoň jedna hľadaná postupnosť, tak určite existuje taká, kde k je prvočíslo. Stačí teda skúšať prvočíselné hodnoty k.

Prvočísla do n vieme nájsť pomocou Eratostenovho sita. Stačí však aj pre každé číslo od 2 po n otestovať v čase $O(\sqrt{n})$, či nemá žiadneho netriviálneho deliteľa. Následne pre každé prvočíslo v $O(n)$ overíme, či sa v našom poli nachádza postupnosť s príslušnou dĺžkou a diferenciou. Keďže prvočísel do n je $O(n/\log n)$, má toto riešenie časovú zložitosť $O(n^2/\log n)$.

2242. V prvom rade si môžeme všetky časy prečíslovať tak, aby tvorili množinu po sebe idúcich prirodzených čísiel $\{1, \ldots, t\}$. Úlohu potom riešime dynamickým programovaním. Nech $M[q, c]$ je najmenší počet výmen uskutočnených do q-tej minúty vrátane nej, pričom c-ty človek tancoval posledný. $M[0, c] = 0$ pre všetky c; pre väčšie q vieme $M[q, c]$ zistiť z hodnôt $M[q-1][\cdot]$. Ak c v q-tej minúte neskákal, $M[q, c] = \infty$, inak platí vzťah:

$$M[q, c] = \min_d \Big(M[q-1, d] + k - [1, \text{ak } c \neq d \text{ a } d \text{ skákal } q\text{-tu minútu}] - [1, \text{ak } c = d \wedge k = 1] \Big),$$

kde k je počet ľudí, ktorí skákali v q-tej minúte. Priamočiara implementácia tejto rekurencie má časovú zložitosť $O(tn^2)$, toto riešenie sa však ľahko dá vylepšiť na $O(nt)$.

2243. Stačí použiť „cyklické“ pole dĺžky m, kde si pre každú z m nasledujúcich sekúnd udržujeme spájaný zoznam udalostí. Trik je rovnaký ako pri implementácii fronty: namiesto toho, aby sme posúvali všetky udalosti, posunieme jednoducho ukazovateľ na začiatok poľa. Ako identifikátory udalostí môžeme použiť jednoducho prirodzené čísla.

2244. Úlohou je pre daný strom (lokality sú vrcholy, ulice hrany) zistiť, koľko hrán treba pridať, aby každá hrana ležala na nejakej kružnici – inými slovami, aby bol výsledný graf hranovo dvojsúvislý.

Nech ℓ je počet listov v grafe. Zjavne potrebujeme aspoň $\lceil \ell/2 \rceil$ hrán, lebo z každého listu musíme pridať aspoň jednu novú hranu. Na druhej strane, vždy vieme vhodne pridať presne $\lceil \ell/2 \rceil$ hrán, čím našu úlohu optimálne vyriešime. Zvolíme vrchol v, ktorý nie je list. Listy očíslujeme v poradí, v akom ich navštívi prehľadávanie do hĺbky z vrcholu v. Pre každé i od 1 po $\lceil \ell/2 \rceil$ spojíme hranou listy číslo i a $i + \lfloor \ell/2 \rfloor$.

Uvažujme ľubovoľnú hranu e v strome. Všimnime si, že listy, ktoré sú v podstrome pod jej spodným vrcholom, majú čísla $x, x+1, \ldots, y$ pre vhodné x, y. Ľahko sa dokáže, že niektorá z nami pridaných hrán má v tejto množine práve jeden koniec. Táto nová hrana uv spolu s cestou z u do v po strome vyrobí cyklus, na ktorom leží hrana e.

2245. V oboch podúlohách vie Oskar vhodným posielaním a odchytávaním správ dostať správu, ktorú síce neprečíta, ale vie ju použiť v nasledujúcom kroku protokolu. Treba si všímať a zneužiť „symetriu“ posielaných správ.

V prvej úlohe pošle Oskar Bobovi správu, ktorá sa tvári, že je od Alice, a obsahuje jej N_A. Jeho odpoveď síce nevie prečítať, ale vie ju poslať Alici, ktorá mu ju nič netušiac dešifruje.

V druhej úlohe začne Oskar tak ako v hinte zo zadania. V druhom kroku by mal Bob poslať Trudy nejaké údaje. Toto Oskar bez zapojenia Boba nedokáže, ale ani nemusí (tento krok sa Alice netýka). Stačí, aby vedel v treťom kroku sfalšovať správu od Trudy. Vďaka tomu, že Oskar začal celý protokol s Alicou ešte raz, dostal od Alice správu $\{B, N_A\}_{K_A}$ – ňou vie Trudy presvedčiť, že je Alica, ktorá chce komunikovať s Bobom. Oskar pošle Trudy správu $A, N_A, \{B, N_A\}_{K_A}$ a Trudy odpovie $N_A, \{A, K, N_A\}_{K_B}, \{B, K, N_A\}_{K_A}$. Teraz už nič nebráni Oskarovi zobrať dve časti správy, ktoré práve dostal, zatváriť sa, že je Trudy, a pokračovať v pôvodnej komunikácii s Alicou.

z2211. Stačí usporiadať artistov podľa hmotnosti, pričom najťažší bude stáť na spodku a najľahší na vrchu veže. Použitím vhodného triediaceho algoritmu, ako je quick-sort, či heap-sort, dosiahneme časovú zložitosť $O(n \log n)$.

z2212. *Riešenie nadväzuje na text „Prehľadávanie do šírky“ na str. 247.*
Mapu si predstavíme ako graf, kde všetky políčka okrem `#` sú vrcholy a hrany vedú medzi políčkami susediacimi stranou alebo rohom. Najkratšiu cestu z `J` do `D` nájdeme jednoducho tak, že z políčka `J` spustíme prehľadávanie do šírky. Časová zložitosť je $O(rs)$.

z2213. Označme a_i plán na $(m-i)$-ty deň, $a_i \in \{-1, 0, 1\}$. Pretože z huby pridanej do čaju v $(m-i)$-ty deň nakoniec vznikne 3^i húb, má platiť:

$$\ell = a_{m-1} \cdot 3^{m-1} + a_{m-2} \cdot 3^{m-2} + \ldots + a_1 \cdot 3 + a_0$$

Úlohou je teda previesť číslo ℓ do tzv. *vyváženej trojkovej sústavy*.

Jedno z možných riešení je pripočítať k obom stranám číslo

$$n = (3^m - 1)/2 = 3^{m-1} + 3^{m-2} + \cdots + 3 + 1.$$

Dostaneme $\ell + n = (a_{m-1} + 1) \cdot 3^{m-1} + (a_{m-2} + 1) \cdot 3^{m-2} + \cdots + (a_1 + 1) \cdot 3 + (a_0 + 1)$, kde $a_i + 1 \in \{0, 1, 2\}$. Ak $\ell + n \geq 3^m$, potom riešenie neexistuje. V opačnom prípade stačí vyjadriť číslo $\ell + n$ v (klasickej) trojkovej sústave a vypísať jeho cifry zmenšené o 1. Časová zložitosť je $O(m)$.

Alternatívne riešenie: Postupujeme od konca. Z každej huby okrem poslednej nám vznikne nejaký počet húb, ktorý bude deliteľný troma. Hodnotu a_0 teda vieme určiť podľa zvyšku, ktorý dáva ℓ po delení 3. Po určení a_0 príslušne upravíme ℓ a následne ho vydelíme 3. Takto postupne určujeme ďalšie hodnoty a_i, až kým buď nevynulujeme ℓ, alebo nedosiahneme m dní.

z2214. Na začiatku si vyplníme pole A veľkosti n tak, že $A[i] = i$. Následne náhodne povymieňame jeho prvky tak, aby nám vyšla náhodná permutácia: Postupne pre $i = 1, \ldots, n$ vymeníme číslo $A[i]$ s náhodným z čísel $A[i], \ldots, A[n]$. Teda najskôr vyberieme $A[1]$ náhodne spomedzi všetkých hodnôt, potom $A[2]$ náhodne spomedzi všetkých okrem $A[1]$, atď. Dá sa ľahko dokázať, že každú permutáciu dostaneme s rovnakou pravdepodobnosťou. Časová zložitosť je $O(n)$.

Pozor, obľúbené nesprávne riešenie je spraviť len nejaký „dostatočne veľký“ počet výmien náhodných dvojíc prvkov. Pri takomto postupe však pravdepodobnosti výsledných permutácií nikdy nebudú presne rovnaké.

z2215. Pri hodnotení tejto úlohy sa prihliadalo hlavne na to, koľko rôznych vylepšení riešiteľ vymyslel a implementoval. Niekoľko príkladov: Skracovanie názvov predmetov (alebo delenie na slabiky), ak sa nezmestia do bunky rozvrhu. V prípade prekryvu prednášok je zobrazená taká množina, aby čo najviac predmetov bolo „viditeľných“. Grafické znázornenie prekryvu predmetov.

z2221. Postupnosť nemusíme usporiadať, stačí z hmotností artistov vytvoriť haldu s minimom na vrchu. Na to stačí postupovať sprava doľava v poli (t.j. odspodu nahor v halde), pričom vždy, keď narazíme na artistu, ktorý je ťažší ako niektorý z kolegov, ktorí ho držia, „prebubleme“ ho dodola – vymeníme ho za ľahšieho z dvojice, ktorá ho drží, a toto opakujeme, až kým už pod nim nie sú dvaja ťažší artisti alebo ho nedostaneme úplne do spodnej vrstvy. Dá sa spočítať, že pri tomto postupe urobíme dokopy menej ako $2n$ výmen, preto má toto riešenie lineárnu časovú zložitosť.

z2222. *Riešenie nadväzuje na text „Prehľadávanie do šírky“ na str. 247.*
Táto úloha sa rieši podobne ako z2212, rozdiel je len v tom, že nám pribudli ohodnotenia hrán. Na hľadanie najkratšej cesty vo všeobecnom ohodnotenom grafe slúži napríklad Dijkstrov algoritmus. Ohodnotenia hrán sú v našom prípade ale len celé čísla od 1 do 5, preto si môžeme každú hranu dĺžky t rozdeliť na t jednotkových hrán a použiť prehľadávanie do šírky. Veľkosť grafu sa zväčší najviac 5-krát, preto časová zložitosť ostane $O(rs)$.

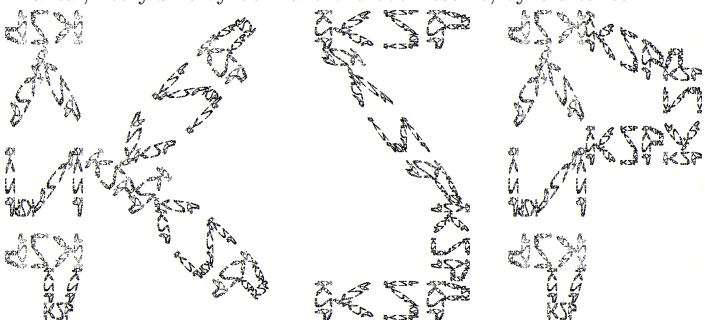
z2223. Použijeme rekurziu. Prechádzame reťazec a pridávame jeho znaky do výstupnej premennej. Vždy, keď narazíme na `číslo[`, tak sa rekurzívne zavoláme (na text po zodpovedajúcu pravú zátvorku) a získaný výstup `číslo`-krát pridáme do našej výstupnej premennej. Z rekurzie sa vraciame na vyššiu úroveň vždy, keď narazíme na `]`. Časová zložitosť je lineárna od dĺžky *výstupu*.

z2224. Miestnosti a chodby jaskyne tvoria vrcholy a hrany orientovaného acyklického grafu. Každej hrane priradíme dĺžku podľa počtu čerešní v miestnosti, do ktorej vedie. Úlohou je nájsť najdlhšiu cestu z vrcholu 1.

Použijeme upravené prehľadávanie do hĺbky. Keď ho spustíme z vrcholu v, vráti dĺžku najdlhšej cesty z v a nasledujúci vrchol na tejto ceste, kvôli jej výpisu. Ak z vrcholu v nevychádzajú hrany, najdlhšia cesta končí vo v a má dĺžku 0. Inak postupne spustíme prehľadávanie do hĺbky zo všetkých vrcholov, do ktorých vedú hrany z v, a vyberieme si ten s najdlhšou cestou. Zároveň si tieto údaje zapamätáme, aby sme ich nemuseli znova počítať.

Všimnite si, že tento postup funguje vďaka tomu, že zadaný graf neobsahuje cykly. Časová zložitosť je lineárna od veľkosti grafu.

z2225. Fraktál, ktorý sme vyrobili ako vzorové riešenie, vyzerá takto:



z2231. Na riešenie tejto úlohy použijeme binárne vyhľadávanie. Časová zložitosť tohto algoritmu je $O(\log n)$, ak uvažujeme, že názvy kníh majú konštantnú dĺžku. Presnejšie, ak m je najväčšia dĺžka názvu knihy, tak časovú zložitosť vieme odhadnúť ako $O(m \log n)$.

z2232. Na riešenie tejto úlohy použijeme dátovú štruktúru písmenkový strom (po anglicky trie). V každom vrchole, kde končí názov nejakej činnosti, budeme mať zoznam príjmov z danej činnosti. Na konci prejdeme celý strom v abecednom poradí a vždy, keď narazíme na nejaký zoznam, vypíšeme príjmy z neho. Časová aj pamäťová zložitosť nášho riešenia je lineárna od veľkosti vstupu.

z2233. Namiesto jedného biliardového stola si predstavme celú sieť biliardových stolov postavených vedľa seba. Ak si teraz predstavíme, že sa guľa neodráža, ale voľne pokračuje ďalej po pôvodnej polpriamke, uvidíme, že hranice stolov prekračuje presne v tých okamihoch, kedy by sa odrážala na pôvodnom stole, a navyše na tých istých miestach.

Ak teda mala pôvodná guľa doputovať na svoje miesto po n odrazoch od jedných a m odrazoch od druhých strán stola, nová myslená guľa musí jedným smerom prejsť o n a druhým o m stolov ďalej. Jej koncové súradnice budú teda $[nb, ma]$. Použitím funkcie arctan teraz ľahko zistíme správny smer.

z2234. Zostrojíme inverznú permutáciu – teda pre každú hodnotu si do pomocného poľa zapíšeme, na ktorej pozícii v permutácii sa nachádza. Potom vyskúšame všetky dvojice čísel ako prvé dva členy postupnosti. Z nich si potom vieme dopočítať hodnotu tretieho člena postupnosti a podľa inverznej permutácie overiť, či leží až za prvými dvoma. Časová zložitosť riešenia je $O(n^2)$, pamäťová je $O(n)$.

z2235. Vykresliť čiary vieme napríklad použitím goniometrických funkcií a zaokrúhľovania. Lepšie je ale použiť Bresenhamov algoritmus, ktorý si vystačí s celými číslami a teda je v praktických aplikáciach rýchlejší.

z2241. Máme nájsť inverznú permutáciu. Na to stačí jedno pole B, ktoré priamo počas čítania vstupu vyplníme: ak $a_i = j$, potom do poľa B na pozíciu j uložíme hodnotu i.

z2242. Jeden možný prístup je použiť pole, kde budeme mať pre každý dátum spájaný zoznam ľudí, ktorí sa vtedy narodili. Takéto riešenie však má obrovské pamäťové nároky, ktoré navyše závisia nie len od počtu spracúvaných ľudí, ale aj od povoleného rozsahu dátumov.

Pamäťovú náročnosť môžeme znížiť tak, že si údaje budeme pamätať v strome: z koreňa povedú hrany zodpovedajúce rokom, kedy sa niekto narodil, z každého vrcholu, ktorý zodpovedá roku, povedú hrany zodpovedajúce mesiacom do ďalších vrcholov, z každého z nich pôjdu hrany do vrcholov zodpovedajúcich jednotlivým dňom.

Obe predchádzajúce riešenia majú tú nevýhodu, že pre niektoré otázky nie je časová zložitosť ich zodpovedania optimálna. Ak by sme napríklad chceli nájsť všetkých, čo sa narodili 10. mája, museli by sme prezrieť všetky roky, bez ohľadu na to, či sa vtedy niekto taký narodil alebo nie.

Na odstránenie tohto nedostatku si stačí spraviť stromy tri. Okrem vyššie popísaného, ktorý používa poradie rok-mesiac-deň, budeme mať aj stromy používajúce poradie mesiac-deň-rok a deň-rok-mesiac. V každom z nich si uložíme údaje o všetkých ľuďoch. Pre každú otázku si potom vyberieme ten správny strom, pomocou ktorého ju zodpovieme. Napríklad pri otázke „10. máj ľubovoľného roku“ použijeme strom „mesiac-deň-rok“.

Pre takéto riešenie platí, že čas potrebný na zodpovedanie ľubovoľnej otázky je priamo úmerný počtu odpovedí na ňu.

z2243. Klasická úloha na hľadanie najlacnejšej kostry v grafe. Riešením je Kruskalov, či Jarníkov-Primov algoritmus.

z2244. Dôležité je uvedomiť si, že pre každý konár vieme povedať, koľko po ňom pôjde veveričiek. Totiž ak na jednej strane konára chýba k orechov, tak na jeho druhej strane musí

byť práve k orechov nazvyš. A práve toľko veveričiek pôjde po tomto konári v optimálnom riešení. Stačí teda pre každý konár určiť rozdiel medzi počtom orechov a veveričiek na jednej jeho strane. To vieme spraviť v lineárnom čase, napríklad počas prehľadávania stromu do hĺbky.

z2245. Pozorný čitateľ vzorových riešení túto úlohu hravo zvládne `:-)`.

2311. *Riešenie nadväzuje na text „Medián postupnosti“ na str. 243.*
Riešenie v čase $O(n \log k)$: budeme si pamätať čísla v halde s minimom navrchu. Vždy, keď vyžrebujú nové číslo, vložíme ho do haldy. Vždy, keď počet čísel v halde presiahne k, najmenšie z nej vyhodíme.

Vzorové riešenie v čase $O(n)$: Čísla čítame po jednom a ukladáme ich poľa. Vždy, keď počet čísel v našom poli dosiahne $2k$, nájdeme v ňom v čase $O(k)$ medián a následne k najmenších čísel zahodíme. Striedavo teda spravíme $O(k)$ krokov pri načítaní a $O(k)$ krokov pri vyhadzovaní malých prvkov. Keďže všetkých načítaní je n, aj celkový počet krokov nášho algoritmu je $O(n)$.

Riešenie domácej úlohy: Keby sme my hrali hru Kingo, zaškrtli by sme k najväčších čísel na hracom pláne. (Každé číslo má rovnakú pravdepodobnosť, že ho niekedy vyžrebujú, a čím je väčšie, tým je menšia pravdepodobnosť, že ho iné čísla vyhodia z výslednej množiny.)

2312. Existuje viacero riešení, ktoré sú až na usporiadanie benzínok podľa vzdialenosti lineárne od ich počtu. Jedna možná myšlienka: Predstavme si, že pôjdeme po trati. Na každej benzínke dotankujeme doplna. Vždy, keď sa hýbeme, používame na to najlacnejší benzín, čo máme. Keď prídeme na benzínku a máme ešte v nádrži drahší benzín ako sa tam dá kúpiť, „odpredáme ho“ (teda odpočítame si ho z nákladov a budeme sa tváriť, že sme si ho nikdy nekúpili). Toto riešenie ľahko naprogramujeme pomocou dátovej štruktúry zvanej deque (fronta, kde vieme odoberať prvky z oboch koncov).

2313. Pre každého jazdca samostatne prehľadáme do šírky priestor stavov, v ktorých sa môže nachádzať. Stav je určený súradnicami, na ktorých jazdec stojí, a tým, či už niekedy stál na políčku s príšerou alebo nie.

Po vykonaní štyroch prehľadávaní stačí len prejsť celú šachovnicu a pre každé políčko zrátať, koľko najmenej musia jazdci prejsť, aby sa na ňom stretli. Máme dve možnosti – buď na príšeru cestou nestúpil nik, alebo to bol práve jeden jazdec. V druhom prípade vyberieme toho jazdca, ktorému stúpenie na príšeru najviac skráti cestu.

2314. Predstavme si bipartitný graf, kde jednu skupinu vrcholov tvoria písmenká slova, ktoré skladáme a druhú skupinu kocky, ktoré máme k dispozícii. Dva vrcholy zodpovedajúce kocke a písmenu spojíme vtedy, ak sa dané písmeno na kocke nachádza. Úlohou je nájsť v takomto grafe maximálne párovanie a zistiť, či je jeho veľkosť rovná počtu písmen skladaného reťazca.

Existuje aj lepšie riešenie: využijeme, že abeceda má len 26 písmen. Namiesto bipartitného párovania budeme riešiť úlohu pomocou hľadania maximálneho toku. Pre každé z 26 písmen abecedy a tiež pre každú kocku budeme mať jeden vrchol. Zo zdroja pôjdu hrany s jednotkovou kapacitou do každého vrcholu predstavujúceho kocku. Z každého z nich povedie práve 6 hrán do vrcholov pre písmená, ktoré sú na nej. No a z vrcholu predstavujúceho písmeno pôjde do ústia hrana s kapacitou rovnou počtu výskytov tohto písmena v slove, ktoré skladáme. Slovo sa dá poskladať, ak v tomto grafe existuje tok veľkosti ℓ. Tento graf má len $O(n)$ hrán, takže použitím Fordovho-Fulkersonovho algoritmu dostávame riešenie v čase $O(n\ell)$.

2315. Ak máme k mincí a sumy od 1 do n, v čase $O(kn)$ vieme spočítať priemerný počet mincí potrebných na zaplatenie každej zo súm (prehľadaním vhodného grafu do šírky). Následne vieme použiť rôzne heuristické metódy na postupné vylepšovanie sady mincí. Celkom dobre sa pre tento problém správajú metódy založené na hill-climbingu:

Keď máme nejakú sadu mincí, postupne vyskúšame veľa jednoduchých zmien (napríklad zmena ceny jednej mince). Pre každú možnú zmenu vyhodnotíme novú sadu mincí. Keď skončíme, najlepšiu nájdenú možnosť si vyberieme a s ňou začneme celý proces odznova.

Optimálne sady pre prvé dva vstupy zo zadania sú napríklad (9, 10) a (12, 13, 15, 22). Najlepšia riešiteľmi nájdená sada pre vstup $n=10000$, $k=15$ je (117, 220, 428, 1310, 1698, 1836, 2393, 2536, 2575, 2790, 2937, 2993, 3479, 4634, 4836).

2321. Táto úloha ma viacero riešení v čase $O(n \log n)$. Jedno z nich: Prvky budeme mať rozdelené na „malé“ a „veľké“ tak, aby bolo malých rovnako ako veľkých alebo o jeden viac. Medián je teda vždy najväčší z malých prvkov.

Malé prvky budeme mať uložené v halde, v ktorej vieme efektívne nájsť maximum. A naopak, veľké prvky uložíme do haldy, v ktorej vieme efektívne nájsť minimum. Keď nám príde nové číslo, tak ho porovnáme s mediánom a podľa toho vložíme do príslušnej haldy. Následne, pokiaľ je rozdiel veľkostí po tejto zmene nesprávny, môžeme najväčší malý prvok presunúť medzi veľké, resp. najmenší veľký medzi malé.

Vidíme, že pridanie nového prvku zahŕňa konštantný počet operácií s haldou, jeho časová zložitosť je teda $O(\log n)$.

2322. Úlohou je nájsť takú permutáciu ψ, aby jej k-ta mocnina bola práve vstupná permutácia φ.

Ak mala permutácia ψ nejaký cyklus dĺžky d, tak v permutácii ψ^k z neho vznikne $e = \mathrm{nsd}(d, k)$ cyklov, každý dĺžky d/e.

Keď teraz rozložíme φ na cykly, musíme tieto vedieť rozdeliť do skupiniek tak, aby v každej skupinke boli cykly, ktoré mohli vzniknúť ako k-ta mocnina jedného cyklu.

Nech sú vo φ nejaké cykly dĺžky ℓ. Dá sa dokázať, že do skupiniek sa dajú podeliť práve vtedy, keď ich počet je násobkom čísla g_ℓ vypočítaného nasledovne: pre každé prvočíslo p_i, ktoré delí ℓ, zoberieme jeho najvyššiu mocninu a_i, ktorá delí k, a všetky $p_i^{a_i}$ vynásobíme.

Vo vzorovom riešení teda rozdelíme φ na cykly, tie rozdelíme podľa dĺžky a následne pre každú dĺžku ℓ zistíme, či je ich počet násobkom g_ℓ. Ak áno, rozdelíme tieto cykly ľubovoľne do skupiniek po g_ℓ a z každej skupinky zostrojíme jeden cyklus permutácie ψ. Ak niekedy nie je cyklov správny počet, riešenie neexistuje.

2323. *Riešenie nadväzuje na texty „Prehľadávanie do hĺbky“ na str. 246 a „Dijkstrov algoritmus“ na str. 248.*
Najprv vypočítame dĺžky najkratších ciest z Rômeovho domu do zvyšku grafu a následne to isté spravíme aj pre Juliäin dom. Na to vieme použiť Dijkstrov algoritmus. Následne si vyrobíme nový graf. Hrana z vrcholu u do v v ňom bude vtedy, ak sa týmto presunom obidvaja priblížia k svojim domom. Tento graf je orientovaný a acyklický. Najdlhšiu cestu v ňom potom vieme nájsť napríklad prehľadávaním do hĺbky.

2324. Súradnice ťažiska trojuholníka vieme nájsť ako priemer súradníc jeho vrcholov. Ťažisko mnohouholníka vieme nájsť tak, že ho rozdelíme na trouholníky. Každému nájdeme ťažisko a nahradíme ho mysleným hmotným bodom, ktorý sa nachádza v dotyčnom ťažisku a hmotnosť má priamo úmernú obsahu trojuholníka. Takto dostaneme sústavu hmotných bodov. Jej ťažisko, a teda ťažisko celého mnohouholníka, ľahko určíme ako vážený aritmetický priemer súradníc daných bodov.

Konvexný n-uholník na trojuholníky rozdelíme ľahko – postupne spájame jeden vrchol so všetkými ostatnými, tým sa nám n-uholník rozpadne na $n-2$ trojuholníkov. Pri nekonvexnom mnohouholníku si treba uvedomiť, že nepotrebujeme hľadať jeho trianguláciu. Stačí totiž urobiť presne to isté ako pri konvexnom mnohouholníku. Len navyše pre každý z $n-2$ trojuholníkov zistíme pomocou vektorového súčinu jeho orientáciu a podľa nej dáme jeho hmotnosti kladné alebo záporné znamienko. (Všimnite si, že tento postup je úpravou algoritmu na výpočet obsahu nekonvexného mnohouholníka.)

Pre ľubovoľný n-uholník teda vieme v čase $O(n)$ zistiť polohu ťažiska. Následne treba ešte overiť, či toto ťažisko leží vo vnútri mnohouholníka. To vieme spraviť napríklad tak, že si zvolíme z daného bodu polpriamku neprechádzajúcu žiadnym vrcholom a zistíme, či má s obvodom mnohouholníka nepárny počet priesečníkov.

2325. Zašifrovaný text je v nemčine, až na prvú vetu. Veľmi dobrým prístupom k hľadaniu šifrovacej permutácie je založený na hill-climbingu. Začneme z náhodnej permutácie a tú následne vylepšujeme: v každom kroku vymeníme niektoré dve písmená, a to tie dve, ktorých výmena nám vyrobí najlepšie znejúci otvorený text. Vyhodnocovanie toho, ktorý otvorený text je nakoľko dobrý, je vysvetlené v zadaní úlohy.

2331. Dynamické programovanie. Pre každú sumu od 0 po s si budeme pamätať, či ju vieme dosiahnuť pomocou už spracovaných bankoviek. Priamočiara implementácia, pri ktorej spracujeme každú bankovku zvlášť, má časovú zložitosť $O(sp)$, kde $p = \sum p_i$.

Existuje ale aj lepšie riešenie s časovou zložitosťou $O(sn)$. Budeme vždy naraz spracúvať všetky bankovky jedného typu. Pri spracúvaní nového typu bankoviek si najskôr vyplníme pomocné pole, kde si pre každú sumu spočítame, koľko najmenej bankoviek aktuálneho typu treba na jej dosiahnutie. Následne za dosiahnuteľné označíme len tie sumy, na ktorých dosiahnutie máme dosť bankoviek.

2332. *Riešenie nadväzuje na text „Dijkstrov algoritmus“ na str. 248.*
Stačí pre každé mesto zistiť najbližšieho závozníka a jeho vzdialenosť, odpovede pre jednotlivé hrany z týchto údajov ľahko dopočítame. Pre mestá zistíme hľadané údaje tak, že naraz zo všetkých miest, kde sú závozníci, spustíme Dijkstrov algoritmus.

2333. Využijeme fakt, že v planárnom grafe je najviac $3n - 6$ hrán. Z Dirichletovho princípu potom vyplýva, že tam existuje vrchol so stupňom najviac 5. Keď ho z grafu odoberieme, dostaneme opäť planárny graf a znova z neho môžeme odobrať vrchol. Takto pokračujeme, až kým nič neostane. V každom vrchole si budeme pamätať hrany, ktoré z neho viedli, keď sme ho odoberali. Celé predspracovanie vieme spraviť v čase $O(n)$. Následne každú otázku vieme zodpovedať v konštantnom čase: ak hrana uv v našom grafe bola, tak je buď v zozname pre vrchol u, alebo v zozname pre vrchol v.

2334. *Riešenie nadväzuje na text „Euklidov algoritmus“ na str. 244.*
Dá sa dokázať, že ku každému skokanovi vieme vyrobiť skokana s najviac *dvoma* skokmi, ktorý má rovnaké teritórium – a takýchto už vieme ľahko porovnávať.

Treba si rozmyslieť, že k dvom skokom, kde jeden je násobkom druhého, vieme vyrobiť jediný skok s rovnakým teritóriom. V prípade, že jeden nie je násobkom druhého, vieme ich prerobiť na (možno iné) dva skoky, ktoré určujú to isté teritórium, ale jeden z nich je vodorovný. Ľubovoľné tri skoky a, b, c vieme potom upraviť na najviac dva k nim ekvivalentné skoky takto: upravíme a, b na a', b', kde a' je vodorovný; následne upravíme b', c na b'', c', kde b'' je vodorovný. Skoky a, b, c sú teda ekvivalentné a', b'', c'; stačí dva vodorovné skoky a', b'' nahradiť jedným ekvivalentným.

Pri hľadaní ekvivalentných skokov sa používa najväčší spoločný deliteľ, takže celková časová zložitosť je $O(n \log w)$, kde w je najväčšie číslo na vstupe.

2335. Ľahko naprogramovateľné ale neoptimálne riešenie úlohy spočíva v nájdení minimálnej kostry pre zadanú množinu miest. Optimálne riešenie sa volá Steinerovský strom a od kostry sa líši tým, že môže mať viac vrcholov ako bolo pôvodne miest. Napríklad ak sú na vstupe tri mestá určujúce rovnostranný trojuholník, optimálne riešenie je spojiť každý z nich s ťažiskom dotyčného trojuholníka.

Zostrojenie optimálneho Steinerovského stromu je ťažký problém. Existujú však rôzne heuristiky, ktorými sa dali zostrojiť dostatočne dobré riešenia pre zadané vstupy. Niektorí riešitelia použili heuristiky plne automatické (napríklad založené na postupnom lokálnom vylepšovaní najlacnejšej kostry), iní začínali tým, že ručne vyrobili približný tvar riešenia a potom nechali svoj program rôznymi spôsobmi toto riešenie vylepšovať.

Za pozretie stojí program GeoSteiner[13], ktorý je natoľko optimalizovaný, že už dokázal nájsť optimálne riešenie aj pre vstupy s 10 000 vrcholmi.

2341. *Riešenie nadväzuje na text „Prehľadávanie do hĺbky“ na str. 246.*
Na vstupe máme acyklický graf a chceme vedieť počet ciest z vrcholu 1 do vrcholu n. Pre ľubovoľný vrchol v vieme počet ciest z neho do vrcholu n spočítať nasledovne: nech z v vedú hrany do $v_1, \ldots, v_k$. Potom celkový počet ciest z v do n je súčtom počtov ciest z každého v_i do n. Tieto počty ciest si vieme počítať počas prehľadávania do hĺbky z vrcholu 1.

2342. Použijeme Ahov-Corasickovej algoritmus.[14] Ide o úpravu algoritmu Knutha, Morrisa a Pratta, ktorá vie naraz vyhľadávať v texte viacero vzoriek.

2343. Nech r_i je pravdepodobnosť, že populácia tvorená jednou podenkou zahynie do i generácii (v triviálnom prípade je $r_1 = p_0$). Potom pravdepodobnosť, že k podeniek vyhynie do i generácií, je r_i^k. Jedna podenka môže mať 0 až $n-1$ potomkov – j potomkov má s pravdepodobnosťou p_j. Preto platí nasledovná rekurencia:

$$r_{i+1} = \sum_{j=0}^{n-1} p_j \cdot r_i^j$$

Teraz už stačí len postupne spočítať hodnoty r_1 až r_m a na konci vypísať hodnotu r_m^k.

2344. Riešenie v čase $O(n \log n)$ s konštantnou pamäťou: Použijeme binárne vyhľadávanie. V každom kroku rozdelíme interval kandidátov na dva polovičnej dĺžky. Jeden z nich si vyberieme, prezrieme celé pole a spočítame si, koľkokrát sa v poli vyskytuje číslo z neho. Ak je počet výskytov väčší ako dĺžka intervalu, pokračujeme ďalej s nim, inak musí byť hľadaný duplikát v druhom intervale polovičnej dĺžky.

Riešenie v lineárnom čase s konštantnou pamäťou: Predstavme si graf s vrcholmi 0 až n, pričom z vrcholu i vedie jediná orientovaná hrana, a to do $a[i]$. Teraz sa postavíme do vrcholu 0 a začneme chodiť po hranách. Keďže vrcholov je konečne veľa, časom nejaký vrchol v navštívime druhýkrát. Všimnite si, že do vrcholu 0 nevedie žiadna hrana, preto $v \neq 0$. Potom ale do v nutne vchádzajú aspoň dve hrany – tá, ktorou sme doň prišli prvýkrát a tá, ktorou sme doň prišli druhýkrát. Inými slovami, hodnota v sa v poli a vyskytuje aspoň dvakrát. Na nájdenie vrcholu v môžeme použiť Floydovu metódu dvoch bežcov.[15]

2345. Asi najschodnejšou cestou je hľadať v zadanej tabuľke závislosti a rozdeliť ju na čo najmenej rôznych prípadov. V C/C++ sa dajú na ďalšie skrátenie programu použiť direktívy pre preprocesor. Tu je jeden program v jazyku C (celkom krátky, ale nie najkratší možný):

```
unsigned n;
main() {
  while (scanf("%u",&n)>0) printf("%d\n", (n>17123 && n<47325 ? n = n*n/13
  - 3*n + 2 : (n *= n/8, n>999 && (n/=47)), n ^= 2*n & 28, n+47));
}
```

z2311. Riešenie pracuje v čase $O(\sqrt{n})$ a používa len konštantné množstvo pomocnej pamäte. Základná myšlienka riešenia: Nech d je deliteľ čísla n. Potom aj (n/d) je deliteľ čísla n. A zjavne nemôžu byť súčasne oba tieto delitele väčšie ako $\sqrt{n}$.

Zoznam deliteľov čísla n zostrojíme v dvoch fázach. V prvej vyskúšame všetky d od 1 po $\sqrt{n}$ a vyberieme tie, ktoré delia n. V druhej prejdeme od najväčšieho po najmenší delitele nájdené v prvej fáze a pre každý z nich vypíšeme číslo n/d. (Pozor na špeciálny prípad keď n je štvorec.)

---

[13] Pozri `http://www.diku.dk/hjemmesider/ansatte/martinz/geosteiner/`.
[14] Pozri `http://en.wikipedia.org/wiki/Aho-Corasick_string_matching_algorithm`.
[15] Pozri `http://en.wikipedia.org/wiki/Cycle_detection`.

z2312. Najjednoduchším riešením tejto úlohy je postupná simulácia – aktuálny dátum k-krát posunieme o 1 deň.

Jedno možné optimálne riešenie: Dátum prevedieme na počet dní od prvého januára roku 1, pripočítame k a výsledok prevedieme späť na dátum. Oba prevody sa dajú spraviť v konštantnom čase.

z2313. Odsimulujeme všetky možné hry piškvoriek. Simulovať hru vieme najľahšie pomocou rekurzie, pri ktorej každé volanie rekurzívnej funkcie zodpovedá jednému ťahu v hre. V tele rekurzívnej funkcie najskôr zistíme, či už hra skončila (výhrou niektorého hráča alebo zaplnením plánu), a ak nie, rekurzívnymi volaniami postupne vyskúšame všetky možné ťahy hráča na ťahu.

Správna odpoveď: Existuje 255 168 možných priebehov hry. Z nich v 131 184 prípadov (teda asi 51.4%) vyhrá prvý hráč, v 77 904 druhý a 46 080 ich skončí remízou.

z2314. V zadaní nie je zaručené, že údaje dostaneme v chronologickom poradí, je teda potrebné si ich najskôr usporiadať podľa času. Potom už ľahko vieme simulovať všetky udalosti a počas simulácie zistiť odpoveď na všetky otázky zo zadania. Na zapamätanie si aktuálneho radu pri pokladni použijeme dátovú štruktúru fronta.

z2315. Na zostrojenie kusov koláča použijeme ľubovoľné prehľadávanie grafu, napríklad do šírky. Postupne prechádzame po vstupe a vždy, keď nájdeme ešte neoznačené políčko, tak naň dáme hrozienko a označíme celý kus koláča, na ktorom leží.

z2321. *Riešenie je uvedené v texte „Union-find“ na str. 249.*

z2322. V programe si budeme každé číslo reprezentovať ako pole obsahujúce jeho cifry. Čísla netreba konvertovať do desiatkovej sústavy, dajú sa pohodlne sčítať aj priamo v n-adickej. Algoritmus je veľmi podobný tomu v desiatkovej sústave – sčitujeme po cifrách, začínajúc od najmenej významných. Prenášame do vyššieho rádu zvyšok, ak súčet cifier presiahne n. Časová zložitosť je lineárna od počtu cifier.

z2323. Úloha je ľahko riešiteľná v kvadratickom čase: pre každú pozíciu, kam sa Banto môže postaviť, odsimulujeme celý turnaj.

Lepšie riešenie v čase $O(n \log n)$: Začneme tým, že Banta umiestnime na začiatok radu, v lineárnom čase odsimulujeme celý turnaj a zapamätáme si celý strom zápasov. Keď teraz Banta vymeníme s nasledujúcim hráčom, väčšina turnaja sa tým nezmení. Zmeniť sa môžu len výsledky zápasov, ktoré sú v „pavúku“ na cestách od našich dvoch hráčov do finále. Stačí teda nanovo vyhodnotiť tieto zápasy, ktorých je $O(\log n)$. Celý tento proces zopakujeme $(n-1)$-krát: vždy najskôr premiestnime Banta o pozíciu ďalej a následne prepočítame zápasy, ktorých výsledky sa mohli zmeniť.

Vzorové riešenie beží v čase $O(n)$. Použijeme pri ňom nasledujúce pozorovanie: Vždy existuje optimálne riešenie, v ktorom Banto najskôr niekoľko zápasov vyhrá bez podvádzania a potom už musí podvádzať až do konca. Prečo je to tak? Ak totiž máme turnaj, v ktorom Banto musel niekedy podvádzať, no neskôr narazil na hráča x, s ktorým vie vyhrať, môžeme Banta preradiť do časti pavúka, ktorú vyhral x, čím určite nezvýšime potrebný počet podvádzaní a prvé podvádzanie odsunieme na neskôr.

Aby sme teda našli najmenší počet podvodov, potrebujeme nájsť najväčšieho pavúka, v ktorom Banto bez podvádzania vyhrá – inými slovami takého, v ktorom bude mať Banto lepšiu formu ako všetci ostatní. Hľadanie vieme implementovať pomocou rekurzie.

Hlavná myšlienka: Zaujíma nás, koľko najviac zápasov vie Banto bez podvádzania vyhrať proti ľuďom $1 \ldots 2^k - 1$. To zistíme tak, že nájdeme riešenie a pre ľudí $1 \ldots 2^{k-1} - 1$ (s tými by Banto hral, keby začínal v prvej polovici pavúka) a následne riešenie b pre ľudí $2^{k-1} + 1 \ldots 2^k - 1$ (s tými by hral v opačnom prípade). Ak $a = b = k - 1$, porovnáme ešte Banta s človekom 2^{k-1} (ten hrá vždy v opačnej polovici pavúka ako Banto); podľa výsledku porovnania je riešením $k - 1$ alebo k. V opačnom prípade je riešením $\max(a, b)$.

Tento postup sa dá implementovať tak, že postupne zo vstupu čítame formy jednotlivých súťažiacich, keď ich práve potrebujeme. Vystačíme si teda s pamäťou veľkosti $O(\log n)$ – toľko jej potrebujeme na samotnú rekurziu.

z2324. Obsah zjednotenia je rovný súčtu obsahov oboch trojuholníkov, mínus obsah ich prieniku. Stačí teda vedieť nájsť prienik dvoch trojuholníkov a vypočítať jeho obsah.

Prienik je zjavne vždy konvexný mnohouholník. A navyše platí, že každý jeho vrchol je buď vrcholom jedného z našich trojuholníkov, alebo je to priesečník nejakej strany prvého trojuholníka s nejakou rôznobežnou stranou druhého trojuholníka.

Pre každý vrchol každého trojuholníka overíme, či leží v druhom trojuholníku. Pre každú dvojicu rôznobežných strán patriacich navzájom rôznym trojuholníkom zistíme, či sa pretínajú, a ak áno, nájdeme ich prienik. Takto dostaneme konštantne veľa bodov, ktoré patria do prieniku, pričom medzi nimi sú určite všetky jeho vrcholy.

Prienik teraz zostrojíme jednoducho tak, že zostrojíme konvexný obal práve nájdenej množiny bodov. Keďže ide o konečne veľa bodov, môžeme použiť napríklad riešenie, ktoré pre každé dva body otestuje, či všetky ostatné ležia napravo od ich spojnice.

Keďže ide o konvexný mnohouholník, jeho obsah vieme následne spočítať napríklad tak, že ho rozdelíme na trojuholníky a pre každý z nich použijeme vzorec zo zadania.

z2325. Najprv prehľadávaním (do šírky alebo do hĺbky) nájdeme všetky časti koláča. Pre každé políčko si zaznačíme, do ktorej časti patrí, a pre každú časť si spočítame počet hrozienok v nej. Potom len prechádzame koláč a vždy, keď sme práve na zároze, tak sa pozrieme na súčet počtov hrozienok v častiach, ktoré tento zárez od seba oddeľuje. Časová aj pamäťová zložitosť riešenia je $O(n^2)$.

z2331. Po úvodnom rozdaní kopy kariet máme na spodkoch jednotlivých kôp karty s číslami od 1 do m. Zjavne na konci bude na spodku jedna z nich – a to tá, ktorá je na spodku jedinej kôpky, ktorú nikdy nevezmeme do ruky.

Od tohto okamihu nám stačí pracovať s počtami kariet na jednotlivých kôpkach. Takto vieme každé rozdávanie kôpky odsimulovať v čase $O(m)$: jednoducho si pre každú kôpku spočítame, koľko kariet na ňu pribudne. Toto riešenie má časovú zložitosť $O(m^2)$.

Existuje aj riešenie lineárne od m. Pri zjednodušenej simulácii si stačí pamätať dve čísla: aktuálny počet kôp x a číslo d od 1 po x udávajúce, na ktorú z nich máme položiť nasledujúcu kartu. Nové hodnoty pre tieto dve čísla vieme zo starých vypočítať v konštantnom čase. Navyše si po každom kroku zapíšeme do pomocného poľa aktuálne (pozor, nie pôvodné!) číslo kopy, ktorú sme práve zrušili. Po skončení simulácie vieme takto zapísané čísla postupne od konca spracovať a vypočítať tak pôvodné číslo kopy, ktorá nám zostala na konci.

z2332. Optimálne je riešenie v lineárnom čase s konštantnou pamäťou: Budeme postupovať zľava doprava, pričom si budeme pamätať počet guličiek, ktoré sme už videli. Nech sme v nejakej chvíli už spracovali j jamiek a videli g guličiek. Ak $g > j$, znamená to, že $g-j$ guličiek budeme musieť niekedy presunúť cez j-tu jamku doprava. A naopak, ak $g < j$, tak $j - g$ guličiek časom pôjde z $(j+1)$-tej jamky do j-tej a odtiaľ možno ďalej doľava.

Takto pre každú dvojicu susedných jamiek zistíme, koľko guličiek medzi nimi určite treba v optimálnom riešení presunúť. A zároveň ľahko nahliadneme, že vždy existuje riešenie, ktoré urobí presne tie potrebné presuny guličiek a nič navyše. Ak teda len potrebujeme zistiť optimálny počet presunov, stačí pre každú dvojicu susedných jamiek určiť potrebný počet presunov medzi nimi a tieto počty sčítať.

z2333. Jedna možnosť je pamätať si rozvrh v poli dĺžky n. Pri načítavaní predmety priamo umiestňujeme do tohto poľa. Ak sa práve pridávaný predmet prekrýva s inými (najviac dvomi), porovnáme ich priority a podľa toho niektoré z nich vyhodíme. Pri vyhadzovaní vždy skontrolujeme, či práve vyhadzovaný predmet má väčšiu prioritu ako každý z ostatných doteraz vyhodených predmetov. Toto riešenie má časovú zložitosť $O(p+n)$.

Druhá možnosť je usporiadať predmety podľa *času*, kedy začínajú. Potom už vieme množinu navštevovaných predmetov zostrojiť jediným prechodom cez ich zoznam. Časová zložitosť tohto riešenia je rovná zložitosti triedenia – $O(p \log p)$.

z2334. Ceny si usporiadame podľa veľkosti. Ku každej cene c potom binárnym vyhľadávaním zistíme, či sa medzi cenami nachádza k nej vhodná doplnková cena $s - c$.

Ešte lepšie a jednoduchšie riešenie je po usporiadaní prezerať ceny nasledovne: Začneme s najmenšou a najväčšou cenou. Ak je ich súčet príliš veľký, vieme, že najväčšiu cenu nemá zmysel skúšať so žiadnou inou, tak ju zahodíme. A naopak, ak je súčet primalý, určite môžeme zahodiť najmenšiu z cien. Takto postupne posúvame po poli s cenami dva ukazovatele, až kým buď nezahodíme všetky prvky alebo nenájdeme dvojicu so správnym súčtom. Kvôli časovej zložitosti triedenia bežia obe riešenia v čase $O(n \log n)$.

Existuje aj riešenie založené na hešovaní. Za predpokladu, že vstupné údaje sú generované náhodne, má takéto riešenie očakávanú časovú zložitosť $O(n)$.

z2335. Úloha sa dala riešiť viacerými spôsobmi. Najjednoduchšie je použiť jeden konkrétny recept (veľmi vhodná je napríklad pizza) a k nemu pridávať suroviny, ktoré viacmenej nepokazia jedlo. Iný možný spôsob je viesť si databázu receptov a v nej potom podľa surovín vyhľadávať. Hybridné riešenie niekde medzi týmito dvoma prístupmi by vyzeralo nasledovne: Možné suroviny si rozdelíme do skupín a pre každú si zvolíme nejaký konkrétny recept – koláč pre sladké, šalát pre zeleninu, atď. Potom vypíšeme ten recept, z ktorého skupiny máme najviac surovín.

z2341. Dynamické programovanie. Pre každé políčko si vypočítame hodnotu $S[i,j]$ – aký najväčší štvorec má pravý dolný roh na políčku $[i,j]$. Ľahko sa presvedčíme, že platí: $S[i,j] = \min\{S[i-1,j], S[i,j-1], S[i-1,j-1]\}+1$, ak je na políčku $[i,j]$ jednotka (v opačnom prípade $S[i,j] = 0$). Podľa tohto predpisu sa už dajú počítať hodnoty $S[i,j]$ postupne zhora nadol, zľava doprava. Netreba si pritom pamätať celú mapu, stačia iba posledné 2 riadky. Výsledkom je samozrejme maximum zo všetkých $S[i,j]$. Dostávame riešenie v čase $O(rs)$ a pamäti $O(s)$.

z2342. Stačí si uvedomiť, že palindróm je určený prvou polovicou čísla. Pokiaľ nemáme číslo zo samých 9, stačí zvýšiť prvú polovicu čísla o 1. Treba si dať pozor na špeciálny prípad – číslo tvaru $99\cdots9$, kde sa mení dĺžka čísla. Časová a pamäťová zložitosť je úmerná počtu cifier n, teda je $O(\log n)$.

z2343. *Riešenie nadväzuje na text „Prehľadávanie do hĺbky“ na str. 246.*
Ide o úlohu z teórie grafov: KSPákom priradíme vrcholy grafu a každý vzťah „u ľúbi v“ si zaznačíme ako hranu z vrcholu u do vrcholu v. Úlohou je teraz nájsť topologické usporiadanie vrcholov, t.j. zoradiť ich tak, aby všetky šípky smerovali sprava doľava. To vieme zostrojiť v čase $O(m+n)$, teda lineárnom od veľkosti vstupu, napríklad použitím prehľadávania do hĺbky.

z2344. Ak označíme počet ciest do bodu $[n,m]$ ako $P[n,m]$, tak pre $n,m > 0$ platí: $P[n,m] = p[n-1,m] + p[n,m-1]$, lebo do cieľa musíme prísť buď krokom vpravo alebo krokom hore. Pomocou tohto vzťahu vieme odpoveď určiť dynamickým programovaním v čase $O(rs)$.

Lepšie riešenie: Môžeme si všimnúť, že vyššie uvedený vzťah je totožný so vzťahom, ktorým počítame hodnoty v Pascalovom trojuholníku. A naozaj, počet ciest z bodu $[0,0]$ do bodu $[r,s]$ je rovný kombinačnému číslu $\binom{r+s}{r} = (r+s)!/(r!\,s!)$. Iné možné zdôvodnenie tejto skutočnosti: cesta sa skladá z $r+s$ krokov a je jednoznačne určená tým, že vyberieme, ktorých r spomedzi nich povedie v smere prvej súradnice.

z2345. Postupne všetky slová vložíme do vhodnej údajovej štruktúry, ako napríklad písmenkový strom alebo hešovacia tabuľka. Druhou možnosťou je slová načítať a usporiadať radix-sortom – tak sa rovnaké slová dostanú za seba. Oboma prístupmi vieme dosiahnuť časovú zložitosť $O(kn)$.

Prehľad metód riešenia

Keďže táto zbierka obsahuje vyše 300 úloh, nie je vôbec prekvapivé, že niektoré metódy riešenia (či už ide o konkrétne algoritmy, dátové štruktúry alebo len uhol pohľadu) sa dajú použiť na viacero úloh. Pokúsili sme sa preto takéto skupiny úloh identifikovať. V tejto časti zbierky uvádzame prehľad najčastejšie sa opakujúcich metód riešenia úloh.

Medián postupnosti

Keď máme postupnosť tvorenú nepárnym počtom prvkov, jej *medián* je prvok, ktorý by bol presne uprostred, keby sme prvky postupnosti usporiadali podľa veľkosti. Pre postupnosť s párnym počtom prvkov existuje viacero rôznych definícií mediánu, najčastejšie sa však používa priemer hodnôt dvoch prvkov, ktoré sú prostredné podľa veľkosti.

Medián je jedným zo základných štatistických údajov. Napríklad ak zisťujeme nejaký reálny údaj tak, že spravíme niekoľko meraní, ich medián je lepšou aproximáciou skutočnej hodnoty ako ich priemer – úplne pokazené meranie sa totiž na mediáne veľmi neprejaví, zatiaľ čo na priemere áno.

Ďalšou významnou aplikáciou mediánu je fakt, že je to najlepšie miesto stretnutia. Predstavme si napríklad n domov stojacich na priamke. Kde postaviť zastávku autobusu tak, aby súčet jej vzdialeností od všetkých domov bol najmenší možný? Optimálnym riešením je ako súradnicu zastávky zvoliť medián súradníc všetkých domov.

Prečo je to tak? Predstavme si, že zastávku posúvame smerom zľava doprava a sledujeme súčet vzdialeností od domov. Kým je vľavo od zastávky menej domov ako vpravo, posunutím vpravo sa súčet zmenší. Keď už sa dostaneme za stredný prvok, bude viac domov, od ktorých sa budeme vzďaľovať ako tých, ku ktorým sa budeme blížiť. Preto bude súčet zase stúpať. Najmenší je teda v mediáne všetkých súradníc.

Medián postupnosti vieme samozrejme nájsť tak, že postupnosť usporiadame. Existujú však aj efektívnejšie algoritmy, a to dokonca aj pre všeobecnejší problém: pre dané n-prvkové pole a dané číslo k vieme v čase $O(n)$ nájsť k-ty najmenší prvok v dotyčnom poli.

Jednoduchý algoritmus riešiaci túto úlohu dostávame upravením triedenia quick-sort. Prvá fáza vyzerá rovnako: náhodne si zvolíme jeden prvok p (nazývaný pivot) a preusporiadame pole tak, aby sme najskôr mali prvky menšie ako p, potom rovné p a nakoniec väčšie ako p. Keďže o každej časti vieme, koľko má prvkov, vieme teraz povedať, v ktorej časti poľa leží hľadaný k-ty najmenší prvok. Ak je v strednej časti, našli sme ho. V opačnom prípade si ešte spočítame, koľký najmenší prvok v danej časti poľa hľadáme a nájdeme ho rekurzívnym volaním tohto algoritmu na príslušnú časť poľa. Dá sa dokázať, že tento algoritmus má očakávanú časovú zložitosť $O(n)$.

Existujú aj algoritmy, ktoré majú časovú zložitosť $O(n)$ v najhoršom možnom prípade. Jeden z nich je založený na nasledujúcej myšlienke: Použijeme predchádzajúci algoritmus. Pivota p však nebudeme voliť náhodne. Vyberieme ho nasledovným postupom: rozdelíme prvky na $\lceil n/5 \rceil$ pätíc. V každej z nich porovnaniami nájdeme prostredný prvok podľa veľkosti. Z týchto $\lceil n/5 \rceil$ rekurzívnym volaním práve popisovaného algoritmu nájdeme medián a ten zvolíme za pivota p. Načo je celá táto mágia dobrá? Dá sa ukázať, že vždy bude aspoň 30% prvkov v poli menších a aspoň 30% prvkov väčších ako takto zvolené p. Takouto šikovnou voľbou p teda zabezpečíme, že v každom kroku sa veľkosť úseku, v ktorom hľadáme, zmenší aspoň o 30% jeho aktuálnej dĺžky. No a z toho sa už dá odvodiť, že celkový čas behu algoritmu bude zaručene lineárny od n.

Uvedomte si tiež, že vyššie uvedené algoritmy vieme triviálne upraviť tak, aby pre dané pole a číslo k v lineárnom čase našli všetkých k najmenších prvkov.

Euklidov algoritmus

Nech a a b sú kladné celé čísla. Potom platí, že najväčší spoločný deliteľ čísel a a b je rovnaký ako najväčší spoločný deliteľ čísel b a $a \bmod b$. (Totiž ak nejaké d delí aj a, aj b, tak delí aj $a \bmod b$. A naopak, ak delí b aj $a \bmod b$, musí nutne deliť aj a.)

Na tomto pozorovaní je založený Euklidov algoritmus na hľadanie najväčšieho spoločného deliteľa. V základnej podobe je až neuveriteľne jednoduchý:

$nsd(a,b)$:
  ak $b = 0$, **tak**: **vráť** a
  inak: **vráť** $nsd(b, a \bmod b)$;

Časová zložitosť tohto algoritmu je $O(\log(\min(a,b)))$. Vyplýva to z toho, že keď začneme z ľubovoľnej dvojice (a,b), tak po dvoch krokoch dostaneme dvojicu (a',b') takú, že platí $a' \leq a/2$ a $b' \leq b/2$.

Pomocou najväčšieho spoločného deliteľa vieme určiť aj najmenší spoločný násobok. Vždy totiž platí $\mathrm{nsn}(a,b) = ab/\mathrm{nsd}(a,b)$.

Pre ľubovoľné celé čísla a a b, ktoré nie sú obe súčasne rovné nule, navyše platí, že číslo z sa dá zapísať v tvare $ax + by$ (kde x, y sú vhodné celé čísla) práve vtedy, keď je násobkom najväčšieho spoločného deliteľa čísel a a b.

Prečo to platí? Jedna implikácia je zjavná: ak d delí aj a, aj b, tak každé číslo tvaru $ax + by$ je zjavne deliteľné číslom d. Preto ak z nie je deliteľné číslom $\mathrm{nsd}(a,b)$, tak sa v požadovanom tvare zapísať nedá.

Namiesto druhej implikácie stačí dokázať, že vždy existujú x a y také, že $ax + by = d$. Ak totiž poznáme tieto x a y, tak ľubovoľné $z = kd$ vieme zapísať v tvare $a\overline{x} + b\overline{y}$: stačí zvoliť $\overline{x} = kx$ a $\overline{y} = ky$.

Náš dôkaz bude konštruktívny: upravíme Euklidov algoritmus tak, aby jednu takúto dvojicu našiel. (Takto upravený algoritmus sa zvykne nazývať *rozšírený Euklidov algoritmus*.) Hodnoty budeme vyrábať pri návrate z rekurzie. Ak $b = 0$, tak $\mathrm{nsd}(a,b) = a = 1a+0b$, stačí teda zobrať $x = 1$ a $y = 0$. No a teraz predpokladajme, že poznáme x a y také, že $bx + (a \bmod b)y = d$. Potom stačí položiť $\overline{x} = y$ a $\overline{y} = x - \lfloor a/b \rfloor \cdot y$ a ľahko overíme, že naozaj $a\overline{x} + b\overline{y} = d$.

S rozšíreným Euklidovym algoritmom súvisí jeden známy a užitočný poznatok z teórie čísel. V matematike bežne počítame s celými číslami, ale v počítači je často rozsah premenných obmedzený. Napríklad ak použijeme najbežnejšie celočíselné premenné (`int` v C/C++/Jave, `longint` v Pascale), všetky výpočty sa v skutočnosti budú diať modulo 2^{32}. Iné bežné využitie takýchto výpočtov je v kryptológii, kde nás často zaujíma len zvyšok, ktorý dáva výsledok výpočtu po delení vhodným prvočíslom.

Počítanie modulo nejaké pevne zvolené n (odborne nazývané *modulárna aritmetika*) funguje skoro rovnako ako operácie s klasickými celými číslami. Napríklad súčtom čísel a a b je číslo $(a+b) \bmod n$. Rovnako ľahko vieme odčítať a násobiť. Problém je ale s delením.■

Deliť sa v niektorých situáciách nedá vôbec: napríklad rovnica $2x \equiv 7 \pmod{10}$ zjavne nemá žiadne celočíselné riešenie. Nevieme teda obe strany rovnice „vydeliť niečím vhodným“. Alebo taká rovnica $2x \equiv 8 \pmod{10}$. Tu by sa zdalo, že deliť môžeme: vľavo 2, vpravo 8, z toho predsa vyplýva $x \equiv 4 \pmod{10}$. Lenže ono nie: riešením rovnice $2x \equiv 8 \pmod{10}$ je aj napríklad $x = 9$, a to veru po delení 10 zvyšok 4 nedáva.

Tým sa dostávame k sľubovanému užitočnému poznatku: existujú situácie, kedy sa „deliť“ dá. Presnejšie, kedykoľvek, keď sú čísla n a a nesúdeliteľné, tak existuje celé číslo $\overline{a}$ také, že $a \cdot \overline{a}$ dáva po delení n zvyšok 1. Takéto $\overline{a}$ voláme *inverzným prvkom* k číslu a.

Čo má toto spoločné s delením? Predstavme si, že riešime rovnicu $ax \equiv b \pmod{n}$, kde a a n sú nesúdeliteľné. Keď obe strany vynásobíme číslom $\overline{a}$, dostávame $a\overline{a}x \equiv \overline{a}b \pmod{n}$. Lenže $a \cdot \overline{a}$■ $\equiv 1 \pmod{n}$, preto sme vlastne dostali novú rovnosť $x \equiv \overline{a}b \pmod{n}$.

Vydelenie oboch strán číslom a sme teda dosiahli tak, že sme obe strany prenásobili číslom $\overline{a}$. Ak je a nesúdeliteľné s n, toto je ekvivalentná úprava. (To sa dá vidieť z toho, že keď teraz obe strany prenásobíme číslom a, dostaneme pôvodnú rovnicu.)

Ak n je prvočíslo, tak každé $a \in \{1, 2, \ldots, n-1\}$ má práve jeden inverzný prvok.

Nájsť ku konkrétnemu a a n takýto inverzný prvok nie je ľahké. Jednou z možných metód je práve rozšírený Euklidov algoritmus. Nech $\mathrm{nsd}(a, n) = 1$. Keď spustíme rozšírený Euklidov algoritmus, dostaneme celé čísla x a y také, že $ax + ny = 1$. Ak sa na túto rovnosť pozrieme modulo n, dostávame, že $ax \equiv 1$, a teda x je hľadaným inverzným prvkom ku a vzhľadom na násobenie modulo n.

Základné pojmy z teórie grafov

Veľa úloh v tejto zbierke sa točí okolo grafov. Predstavme si križovatky v meste pospájané ulicami, HTML stránky, hru Sokobana, či ľudí baviacich sa na bále. Ak trochu abstrahujeme, vo všetkých prípadoch máme akési *vrcholy* (križovatky, HTML stránky, pozície hry, ľudí) a dvojice vrcholov môžu byť v nejakom vzťahu – hovoríme, že sú spojené *hranou* (križovatky môžu byť spojené ulicou, jedna stránka môže odkazovať na druhú, z jednej pozície sa dá na jeden krok dostať do inej a Jožko chce tancovať s Aničkou). Takúto štruktúru, pozostávajúcu z vrcholov a hrán spájajúcich dvojice vrcholov, nazývame *graf*. Niekedy (napríklad odkazy na iné stránky, ťahy v hre) sú hrany iba jednosmerné – tieto voláme *orientované*, znázorňujeme ich ako šípky a graf voláme *orientovaný*. Inokedy (napríklad ulice) sú hrany obojsmerné a graf je *neorientovaný*. Nakoniec, v niektorých úlohách hranám priradzujeme nejaké ceny, dĺžky alebo váhy – takýto graf nazývame *ohodnotený*.

Veľa problémov zo zbierky aj z bežného života sa dá formulovať v teórií grafov, preto si v tejto kapitole zhrnieme aspoň základnú grafovú terminológiu.

Majme teda graf s n vrcholmi a m hranami. Ak vyslovene nepovieme inak, budeme myslieť neorientovaný neohodnotený graf. Ak v je vrchol, tak vrcholy s ním spojené hranou voláme *susedia* vrcholu v; počet susedov voláme *stupeň* vrcholu.

V počítači môžeme graf reprezentovať dvoma spôsobmi: Ak je graf „hustý“ (obsahuje veľa hrán vzhľadom na n), stačí si do poľa $n \times n$ pre každú dva vrcholy u, v uložiť či medzi u, v je, alebo nie je hrana (prípadne aká je jej cena). Takúto reprezentáciu voláme *matica susedností*. Naopak, ak je náš graf riedky (obsahuje len málo hrán), stačí, ak si pre každý vrchol zapamätáme zoznam jeho susedov (prípadne tiež ceny hrán, ktoré k nim vedú) – tzv. *zoznam susedností*. Obe popísané metódy fungujú rovnako dobre na orientovaných aj neorientovaných grafoch.

Postupnosť hrán, ktoré na seba nadväzujú, začínajú vo vrchole u a končia vo v nazývame *cesta* z u do v. Jej *dĺžka* je počet hrán, prípadne súčet dĺžok jednotlivých hrán. Cesta, ktorá začína aj končí vo vrchole v sa volá *cyklus*.

Ak sa v grafe dá po hranách dostať odvšadiaľ všade, t.j. ak sú každé dva vrcholy spojené cestou, hovoríme, že graf je *súvislý*. Vo všeobecnosti môže graf pozostávať z niekoľkých súvislých častí, ktoré voláme *komponenty súvislosti*. Podobne v orientovaných grafoch, ak sa dá dostať odvšadiaľ všade *v smere šípok*, graf je *silne súvislý*.

Súvislý graf, ktorý neobsahuje cykly voláme *strom*. V strome teda medzi každou dvojicou vrcholov existuje práve jedna cesta. Dá sa ukázať, že graf s n vrcholmi je strom vtedy a len vtedy, keď je súvislý a má $n - 1$ hrán. V strome si často zvolíme jeden vrchol r a ten voláme *koreň*. Všetky hrany orientujeme smerom ku koreňu. Ak $v \neq r$ je vrchol, tak (jediný) sused bližšie ku koreňu je jeho *otec*, susedia ďalej od koreňa sú jeho *synovia* (rodovo korektne ich tiež voláme rodič a deti v). Vrcholy, ktoré nemajú žiadnych synov voláme *listy* stromu.

Potomkami vrcholu v voláme všetky vrcholy, ktoré sú synovia v, synovia synov v atď. Pokiaľ uvažujeme len strom, ktorý je určený potomkami vrcholu v, tak hovoríme o podstrome stromu s koreňom v.

Prehľadávanie do hĺbky

Jednou z dvoch najčastejšie používaných metód zisťovania súvislosti neorientovaného grafu je prehľadávanie do hĺbky (depth-first search, DFS). Samotný algoritmus popíšeme ľahko: zodpovedá tomu, ako by graf prehľadával človek s kriedou, ktorou si vie značiť navštívené vrcholy. Začneme tým, že sa postavíme do jedného z vrcholov grafu a označíme ho ako navštívený. Po každom kroku tohto algoritmu sa budeme nachádzať v jednom z vrcholov. Ak z vrcholu, kde sa práve nachádzame, vedie hrana, ktorou sme ešte nešli, prejdeme ňou. Ak už sme v jej koncovom vrchole niekedy predtým boli, okamžite sa vrátime naspäť. Ak nie, označíme ho za navštívený a pokračujeme ďalej z neho. Keď už sme prezreli všetky hrany vedúce z aktuálneho vrcholu, vrátime sa z neho hranou, ktorou sme doň prvýkrát prišli. Tento postup nám zjavne zaručí, že časom navštívime všetky vrcholy dosiahnuteľné z toho, kde sme začínali.

Prehľadávanie do hĺbky sa ľahko implementuje pomocou rekurzie:

prehľadaj(*vrchol* v):
  označ v ako navštívený;
  pre každé w, w je sused v:
    ak w nie je navštívený, **tak** *prehľadaj*(w);

Ak si vhodne reprezentujeme graf (zoznamom susedností), tak bude mať uvedená implementácia časovú aj pamäťovú zložitosť lineárnu od veľkosti grafu – teda $O(n + m)$, kde n je počet vrcholov a m počet hrán.

Prehľadávanie do hĺbky má veľa aplikácií v zložitejších algoritmoch. Napríklad pomocou neho vieme v grafe nájsť jeho komponenty súvislosti. Na začiatku každý vrchol dostane číslo komponentu 0 (značiace, že ešte nikam nepatrí). Teraz postupne v cykle prechádzame vrcholy grafu. Vždy, keď nájdeme nenavštívený vrchol, zvýšime si počet komponentov a spustíme z neho prehľadávanie. Počas prehľadávania si navštívené vrcholy značíme tak, že ich číslo meníme z 0 na číslo aktuálneho komponentu. Keď toto prehľadávanie skončí, máme ako navštívený označený celý komponent, obsahujúci vrchol, kde sme prehľadávanie spustili.

Topologické usporiadanie orientovaného grafu je také usporiadanie jeho vrcholov, v ktorom je každý vrchol uvedený neskôr ako všetky tie, do ktorých z neho vedú hrany. Uvažujme napríklad graf, ktorého vrcholy predstavujú kusy oblečenia a hrany informáciu o poradí obliekania. Teda napríklad informáciu „ponožky si musíš dať skôr ako topánky“ si reprezentujeme ako hranu z vrcholu „topánky“ do vrcholu „ponožky“. Pre takýto graf zodpovedá každé topologické usporiadanie vrcholov jednému správnemu poradiu obliekania sa. Je zjavné, že topologické usporiadanie existuje práve vtedy, keď je daný graf acyklický – teda keď jeho hrany netvoria žiaden cyklus.

Rekurzívnu procedúru prehľadávania do hĺbky upravíme tak, aby po prezretí poslednej vychádzajúcej hrany (teda tesne pred tým, ako opustí vrchol) vypísala číslo vrcholu, ktorého spracovanie bolo práve ukončené. V acyklickom grafe vieme potom nájsť jedno topologické usporiadanie jednoducho – v cykle prechádzame jeho vrcholy a vždy, keď nájdeme nenavštívený vrchol, spustíme z neho prehľadávanie do hĺbky. Poradie, v ktorom sú počas prehľadávaní vrcholy vypísané, je hľadaným topologickým usporiadaním. (To vyplýva priamo z toho, ako prehľadávanie do hĺbky funguje: vrchol vypíšeme až vtedy, keď sme už spracovali, a teda aj vypísali, všetko, čo z neho bolo dosiahnuteľné.)

Všimnite si, že takto vieme aj testovať, či graf obsahuje cykly. Stačí vyššie uvedeným postupom zostrojiť poradie vrcholov (to sa nám podarí bez ohľadu na acyklickosť grafu) a potom o ňom overiť, či je topologickým usporiadaním alebo nie – pre každú hranu overíme, či je jej začiatok vypísaný neskôr ako koniec.

Predstavme si, že sme spustili prehľadávanie do hĺbky v neorientovanom súvislom grafe. Pre každý vrchol si hranu, ktorou sme doň prvýkrát prišli, nazvime *stromová*. Takto dostaneme $n-1$ stromových hrán, ktoré dokopy tvoria strom – jednu z kostier pôvodného grafu. Na začiatočný vrchol u sa budeme dívať ako na jej koreň. Táto *DFS kostra* je špeciálna: Všimnime si totiž ľubovoľnú hranu xy pôvodného grafu, ktorá do nej nepatrí. Potom určite musí platiť, že jeden z vrcholov x a y je v našej kostre predkom toho druhého. (Všimnime si totiž okamih, kedy sme prišli do prvého z nich, bez ujmy na všeobecnosti nech je to x. Určite navštívime y skôr ako definitívne opustíme x – ak to nebude nepriamo, tak to bude priamo hranou xy. Potom ale je určite vrchol y potomkom vrcholu x.)

Vrchol, ktorého odstránením sa nám v grafe zvýši počet komponentov, voláme *artikulácia*. Hranu, ktorej odstránením sa nám v grafe zvýši počet komponentov, voláme *most*. Prehľadávanie do hĺbky vieme upraviť tak, aby nám našlo v grafe všetky artikulácie, resp. všetky mosty. Hľadanie mostov vyzerá nasledovne: Nestromová hrana zjavne nemôže byť most, lebo po jej odobraní graf ešte stále obsahuje celý strom. Stromová hrana xy, kde y je synom x, je most práve vtedy, ak z podstromu s koreňom y nevedie žiadna hrana von. A z vyššie uvedenej vlastnosti DFS kostry vyplýva, že ak takáto hrana existuje, vedie do vrcholu objaveného skôr ako x. Stačí teda počas prehľadávania do hĺbky vrcholy číslovať v poradí, v akom ich objavíme, a pre každý vrchol z si spočítať najmenšie číslo vrcholu $\mu(z) = w$ také, že z podstromu s koreňom z vedie hrana do w. Potom hrana xy je most práve vtedy, keď neplatí $\mu(y) < x$. Hľadanie artikulácií funguje analogicky: vrchol (iný ako koreň) je artikulácia práve vtedy, keď má aspoň jeden podstrom, z ktorého nevedie von žiadna hrana.

Prehľadávanie do hĺbky ako algoritmus na overovanie dosiahnuteľnosti funguje aj na orientovaných grafoch. Keď ho spustíme z vrcholu v, zostrojí nám množinu všetkých vrcholov, do ktorých z v existuje orientovaná cesta.

Silne súvislý komponent (strongly connected component, SCC) orientovaného grafu je maximálna množina vrcholov taká, že sa v rámci nej vieme dostať z každého vrcholu do každého. Pomalý ale jednoduchý algoritmus na zostrojenie SCC je založený na nasledovnej myšlienke: SCC obsahujúci vrchol v je prienikom dvoch množín vrcholov – tých, do ktorých sa vieme dostať z v, a tých, z ktorých sa vieme dostať do v. Každú z týchto množín vieme zostrojiť jedným prehľadávaním.

Všetky SCC však dokonca vieme nájsť počas jediného vhodne upraveného prehľadávania do hĺbky. Detaily nájdete napr. na `http://en.wikipedia.org/wiki/Tarjan's_strongly_connected_components_algorithm`.

Prehľadávanie do šírky

Alternatívou k prehľadávaniu do hĺbky je prehľadávanie do šírky (breadth-first search, BFS, niekedy tiež nazývané flood-fill). Toto predstavuje spôsob, akým by daný graf preskúmala voda, šíriaca sa zo začiatočného vrcholu rovnomerne do všetkých smerov.

Prehľadávanie do šírky ľahko implementujeme pomocou dátovej štruktúry fronta:

```
prehľadaj(vrchol v):
  označ v ako navštívený;
  vyrob frontu Q obsahujúcu jediný prvok v;
  kým Q nie je prázdna:
    vyber zo začiatku fronty Q prvok x;
    pre každé y, y je sused x:
      ak y nie je navštívený, tak:
        označ y ako navštívený;
        vlož y na koniec fronty Q;
```

Podobne ako pri prehľadávaní do hĺbky platí, že ak si vhodne reprezentujeme graf, tak bude mať uvedená implementácia časovú aj pamäťovú zložitosť lineárnu od veľkosti grafu – teda $O(n+m)$, kde n je počet vrcholov a m počet hrán.

Všimnite si, že poradie, v akom tento algoritmus spracúva vrcholy, skutočne zodpovedá „šíreniu vody“: najskôr spracujeme začiatočný vrchol u, potom postupne všetkých jeho susedov, potom susedov jeho susedov, a tak ďalej.

Výhodou prehľadávania do šírky je, že pre každý vrchol grafu nájde najkratšiu cestu z v doň (t.j. cestu tvorenú najmenším možným počtom hrán).

Ak potrebujeme vedieť len dĺžku tejto cesty, je úprava vyššie uvedeného algoritmu jednoduchá: Pre začiatočný vrchol v nastavíme jeho vzdialenosť na 0, a keď pri spracúvaní nejakého vrcholu x prvýkrát objavíme nový vrchol y, vieme, že vzdialenosť z v do y je o 1 väčšia ako z v do x.

Ak by sme navyše chceli vedieť aj jednu najkratšiu cestu zostrojiť, stačí si v každom vrchole navyše zapamätať číslo vrcholu, z ktorého sme sem prvýkrát prišli. Pomocou tejto informácie sa potom vieme z ľubovoľného vrcholu postupne vrátiť do v, a tak zostrojiť hľadanú najkratšiu cestu.

S prehľadávaním do šírky sa bežne stretneme v úlohách, ktoré sa nás pýtajú: „Aký je najmenší počet krokov/ťahov/... potrebný na dosiahnutie takého a takého stavu?“ Napríklad taká úloha: „Koľko najmenej skokov potrebuje šachová figúrka jazdec, aby preskákala z políčka A na políčko B?“ Alebo: „Koľko ťahov potrebujeme na poskladanie Rubikovej kocky? Koľkými prelievaniami sa dá nabrať daný objem v úlohe z1615?“

Za každou takouto úlohou treba hľadať graf. Vrcholy sú stavy, v ktorých sa môžeme nachádzať (teda napríklad pre jazdca sú stavmi jednotlivé políčka šachovnice) a hrany, vedúce z vrcholu, zodpovedajú ťahom, ktoré môžeme v danom stave spraviť. Prehľadávaním do šírky nájdeme najkratšiu postupnosť ťahov vedúcu zo začiatočného do koncového stavu. Takémuto algoritmu sa niekedy hovorí *prehľadávanie priestoru stavov*.

Dijkstrov algoritmus

Prehľadávanie do šírky nám umožňuje nájsť v grafe tú cestu z vrcholu u do vrcholu v, ktorá používa najmenší počet hrán. V praxi však grafmi často modelujeme situácie, kedy nie sú všetky hrany rovnocenné. Ak si napríklad zostrojíme graf ciest medzi okresnými mestami na Slovensku, hrana z Bratislavy do Pezinka nebude rovnako dlhá ako hrana z Trebišova do Michaloviec. V takomto prípade každej hrane priradíme jedno nezáporné číslo predstavujúce jej dĺžku. A to, čo nás často bude zaujímať, bude *najkratšia cesta* – nezáleží nám na počte hrán, ktorými pôjdeme, dôležité je len, aby súčet ich dĺžok bol najmenší možný. Túto úlohu efektívne rieši Dijkstrov[16] algoritmus.

Ide vlastne o zovšeobecnenie prehľadávania do šírky na grafy s ohodnotenými hranami. Pri hľadaní najkratších ciest z vrcholu u budeme ostatné vrcholy spracúvať usporiadané podľa vzdialenosti od u. Takto postupne vypočítame dĺžky najkratších ciest z u do všetkých ostatných vrcholov, teda aj do v.

Presnejšie to bude fungovať nasledovne: Počas behu algoritmu máme vrcholy grafu rozdelené na spracované a nespracované. Pre každý spracovaný vrchol už poznáme dĺžku najkratšej cesty z u doň. Pre každý nespracovaný vrchol poznáme dĺžku nejakej, nie nutne ešte najkratšej cesty doň – a to konkrétne dĺžku najkratšej takej cesty z u doň, ktorá vedie len cez už spracované vrcholy. Korektnosť algoritmu je teraz založená na nasledujúcom pozorovaní: Zoberme si ten nespracovaný vrchol, pre ktorý si aktuálne pamätáme najmenšiu dĺžku cesty. Táto hodnota už musí byť skutočnou dĺžkou úplne najkratšej cesty doň – totiž už nemáme čo získať tým, že pôjdeme cez iný ešte nespracovaný vrchol.

[16] Čítaj „dajkstrov“.

Vždy, keď prehlásime nejaký vrchol x za spracovaný, musíme ešte upraviť vzdialenosti, ktoré si pamätáme pre ešte nespracované vrcholy. Pre každú hranu xy takú, že y je ešte nespracovaný, nám práve pribudol nový spôsob, ako sa dostať do y: najkratšou cestou prídem do x a odtiaľ hranou xy do y. Ak je táto cesta kratšia ako doteraz zapamätaná, zapamätáme si jej dĺžku.

Celý algoritmus si teda môžeme v pseudokóde zapísať nasledovne:

dijkstra(*vrchol* u):
  označ všetky vrcholy ako nespracované;
  nastav vzdialenosť $D[u]$ vrcholu u na 0;
  pre každý iný vrchol v nastav $D[v] \leftarrow \infty$;
  nech $\mathcal{S}$ je množina všetkých dvojíc $(D[v], v)$,
    kde v je nespracovaný vrchol (na začiatku to je každý);
  kým je $\mathcal{S}$ neprázdna:
    vyber najmenší prvok $(D[x], x)$ z $\mathcal{S}$;
    označ x ako spracovaný;
    pre každú hranu z x s koncom y a dĺžkou d:
      ak je $D[y] > D[x] + d$, **tak**:
        nastav $D[y] \leftarrow D[x] + d$; (uprav záznam o y v $\mathcal{S}$)

Ostáva doplniť, ako implementovať množinu $\mathcal{S}$. Najjednoduchšia možnosť je použiť obyčajné pole, v ktorom si pre každý vrchol v zapamätáme, či už je spracovaný a aká je hodnota $D[v]$. Nájsť najmenší prvok aj spracovať všetky hrany, ktoré z neho vedú trvá $O(n)$, preto celková časová zložitosť algoritmu je $O(n^2)$. Keďže graf môže mať až $\binom{n}{2} = (n^2 - n)/2$ hrán, tento algoritmus je optimálny pre husté grafy s veľkým počtom hrán.

Naopak, v praxi sa často stretávame s riedkymi grafmi, ktorých počet hrán m je oveľa menší ako n^2. Napríklad v štvorcovej mriežke má každý vrchol najviac 4 susedov, takže počet hrán je lineárny. Všeobecne, tzv. *rovinné* (alebo *planárne*) grafy, ktoré vieme nakresliť na papier bez toho, aby sa hrany križovali, majú iba lineárny počet hrán. Pre takéto grafy existuje lepšia možnosť.

Ak množinu $\mathcal{S}$ implementujeme pomocou haldy (pričom si pre každý vrchol pamätáme jeho pozíciu v halde), vieme nájsť vrchol s najmenšou vzdialenosťou v čase $O(\log n)$. Takisto v čase $O(\log n)$ vieme spracovať jednu hranu: stačí zmeniť vzdialenosť do nejakého vrcholu y a prebublať ho nahor. Keďže každý vrchol raz vyhlásime za spracovaný a na každú z neho vedúcu hranu sa raz pozrieme, má takáto implementácia časovú zložitosť $O((m+n)\log n)$.

Malý trik: V skutočnosti si nepotrebujeme udržiavať pre každý vrchol pozíciu v halde. Namiesto menenia záznamov v halde budeme jednoducho pridávať nové, lepšie. Pri hľadaní najmenšieho nespracovaného vrcholu potom jednoducho preskakujeme zastaralé záznamy už spracovaných vrcholov, ktoré nám v halde ostali.

Iné, rovnako dobré riešenie (s lepšou pamäťovou zložitosťou ako to, ktoré dostaneme malým trikom) je na uloženie množiny $\mathcal{S}$ použiť vyvažovaný binárny vyhľadávací strom.

Zaujímavý teoretický výsledok: pomocou prefíkanej dátovej štruktúry nazývanej Fibonacciho halda sa dá dosiahnuť o chlp lepšia časová zložitosť: $O(m+n\log n)$. Toto riešenie však v praxi nie je zaujímavé.

Union-find

Predstavme si, že máme mesto, ktoré celé zapadlo snehom, takže sa odnikiaľ nikam nedá dostať. V meste sú rôzne budovy, ktoré sú pospájané zaviatými cestami. Ľudia v nich netrpezlivo čakajú a vypytujú sa, že či sa už dostanú domov z práce, študenti z domu do školy a na iné zaujímavé miesta. No a cestári postupne odhŕňajú sneh z ulíc, čím vznikajú väčšie a väčšie zhluky budov, medzi ktorými sa už dá cestovať.

Túto úlohu si môžeme sformalizovať ako grafový problém. Na začiatku máme graf s n vrcholmi (budovami), ktoré nie sú pospájané žiadnymi hranami (cestami). Vždy keď cestári odhrnú nejakú cestu medzi dvoma budovami, zodpovedajúce dva vrcholy spojíme hranou. Pomedzi to sa obyvatelia pýtajú otázky, či existuje nejaká cesta medzi vrcholmi x a y – teda či sú v tom istom komponente nášho grafu.

Efektívny algoritmus, ktorý rieši náš problém – spája vrcholy grafu hranami do spoločných komponentov a zodpovedá na otázky či sa nachádzajú dva vrcholy v spoločnom komponente – sa nazýva union-find.

V každom komponente budeme mať hierarchiu podobnú ako vo veľkej firme. Jeden zvolený vrchol bude „najvyšším šéfom“ – reprezentantom celého komponentu. Niektoré ďalšie vrcholy budú jeho „priami podriadení“. A každý z nich môže mať zase veľa svojich podriadených, a tak ďalej.

Takúto hierarchiu si vieme až neuveriteľne ľahko uložiť v pamäti: pre každý vrchol v si stačí pamätať jediný ďalší vrchol: jeho bezprostredného šéfa. Výnimkou budú len reprezentanti komponentov, ktorí žiadneho šéfa nemajú. (Implementovať to môžeme napríklad tak, že si pre nich zapamätáme, že šéfujú sami sebe.)

Všimnite si, že hocikedy vieme veľmi ľahko určiť reprezentanta komponentu, v ktorom leží daný vrchol v: stačí sa pozrieť na jeho šéfa, šéfa jeho šéfa, a tak ďalej, až sa po pár krokoch dostaneme do hľadaného vrcholu – „najvyššieho šéfa“. Ako zistiť, či vrcholy x a y ležia v tom istom komponente? Nájdeme vrcholy x' a y', ktoré sú reprezentantmi ich komponentov, a otestujeme, či $x' = y'$.

A ako spojiť komponenty obsahujúce vrcholy x a y? Opäť stačí, ak si nájdeme reprezentantov ich komponentov x' a y'. Ak platí $x' = y'$, nič netreba robiť, vrcholy sú už v tom istom komponente. Inak sú vrcholy x a y v rôznych komponentoch. Preto nastavíme, že odteraz je vrchol y' šéfom vrcholu x' – a teda reprezentantom pre všetky vrcholy oboch bývalých komponentov. Všimnime si, že takto nevytvoríme cyklus medzi šéfmi a z každého vrcholu tohto spojeného komponentu sa postupným pýtaním na šéfov dostaneme až k y'.

Na začiatku výpočtu máme n izolovaných vrcholov. Teda každý vrchol tvorí jeden komponent a je teda aj reprezentantom dotyčného komponentu.

Takéto riešenie by mohlo byť v praxi často rýchle, avšak pri najhoršom možnom vstupe je ešte stále dosť pomalé. Všimnime si, že keď si zakreslíme všetky hrany z v do šéfa $\mathit{šéf}[v]$ ako graf, každý komponent bude zodpovedať jednému orientovanému stromu, v ktorého koreni je jeho reprezentant. Čím sú tieto stromy hlbšie, tým sú operácie s nimi pomalšie.

Preto použijeme dve vylepšenia, ktoré nám umožnia zmenšiť hĺbku týchto stromov a teda aj znížiť počet dotazov na šéfov:

Prvým zlepšením je kompresia cesty v strome. Predstavme si, že hľadáme reprezentanta pre vrchol a. Zistíme, že šéfom a je b, šéfom b je c a šéfom c je hľadaný reprezentant d. V tomto okamihu ale môžeme zmeniť vrcholom a aj b šéfa priamo na vrchol d, a teda nikdy viac nebudeme musieť prechádzať po ceste $abcd$, ktorú sme práve rozbili.

Druhé zlepšenie pomáha udržiavať naše stromy košaté ale nie veľmi hlboké. Pre každý komponent si budeme pamätať jeho rád, ktorý bude na začiatku 0. Keď ideme spájať dva komponenty s reprezentantmi x' a y', tak sa pozrieme na rády týchto komponentov. Ak má jeden z nich menší rád ako druhý, tak reprezentant toho komponentu s väčším rádom bude šéfom reprezentanta komponentu s menším. V opačnom prípade bude y' šéfom x', pričom novovzniknutému komponentu zvýšime rád o 1.

Dá sa dokázať, že pre takto vylepšený algoritmus platí nasledovný odhad časovej zložitosti: ľubovoľnú postupnosť m operácií vykoná v čase $O(m \cdot \alpha(n))$. Funkcia $\alpha(n)$ je tzv. inverzná Ackermannova funkcia. Táto funkcia rastie až nepredstaviteľne pomaly: Pre ľubovoľné rozumne veľké n je $\alpha(n) \leq 5$, takže náš algoritmus prakticky potrebuje v priemere na každú požiadavku len konštantne veľa operácií.

Nasleduje celý algoritmus v pseudokóde. Funkcia *find* nájde reprezentanta komponentu obsahujúceho vrchol x, funkcia *test* zistí, či x a y ležia v tom istom komponente a funkcia *union* spojí komponenty obsahujúce x a y.

init(): **pre každý** vrchol nastav, že nemá šéfa a jeho rád je 0;

find(x):
  ak x nemá šéfa, **tak**: **vráť** x;
  $y \leftarrow$ *find*(*šéf*[x]);
  šéf[x] $\leftarrow y$; ▷ toto urobí kompresiu cesty
  vráť y;

test(x, y): **vráť,** či *find*(x) = *find*(y);

union(x, y):
  $x, y \leftarrow$ *find*(x), *find*(y);
  ak $x = y$, **tak**: **skonči**; ▷ x a y už sú v jednom komponente
  ak *rád*[x] > *rád*[y], **tak**: vymeň x a y;
  ak *rád*[x] = *rád*[y], **tak**: *rád*[y] $\leftarrow$ *rád*[y] + 1;
  šéf[x] $\leftarrow y$;

Jednou z významných aplikácií algoritmu union-find je Kruskalov algoritmus na nájdenie najlacnejšej kostry v grafe. Ten je založený na tom, že spracúvame hrany v poradí od najlacnejšej po najdrahšiu. Ak konce hrany ležia v rôznych komponentoch, tak príslušné komponenty spojíme dokopy, a ak nie, tak hranu len zahodíme.

Index

Michal Forišek, Jakub Kováč, Monika Steinová,
Peter Fulla, Vladimír Boža, Michal Nánási
Zbierka riešených úloh
Korešpondenčného seminára z programovania (1998–2006)
Sádzané programom TEX

Vytlačila a vydala OKAT PLUS, spol. s r.o. Bratislava, 2011
7 + 255 strán, náklad 200 výtlačkov
ISBN 978-80-88720-16-4
Neprešlo jazykovou úpravou

Táto práca bola podporovaná Agentúrou na podporu
výskumu a vývoja na základe zmluvy č. LPP-0103-09.