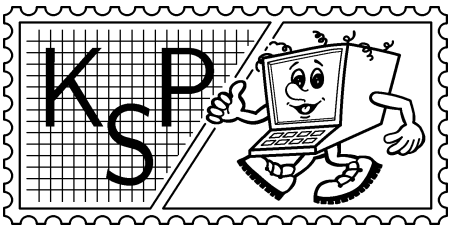
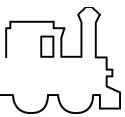


Zbierka riešených úloh

Korešpondenčného seminára z programovania

(1983 – 1998)

Katedra základov a vyučovania informatiky
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 prvých pätnástich ročníkov tejto súťaže.

Chceli by sme ju venovať tlačiarenskému škriatkovi, ktorý spôsobil, že sme túto vetu zabudli vymazať skôr ako pôjde kniha do tlače.

Žltý buldozér veselo kosil lekvár

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!

Rovno takto na začiatku knihy vyhlásime **SÚŤAŽ**. O čo ide? Spravili sme maximum pre to, aby ste dostali do rúk úplne dokonalú knihu. Ale život už veľakrát ukázal, že nič nemôže byť dokonalé, a tak sa nejaké chyby určite nájdu aj v tejto knihe. 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) nájdete na adrese `http://www.ksp.sk/ksp/zbierka/`. Tam sa dočítate aj inštrukcie, ako nám oznámiť novú nájdenú chybu.

História KSP

Prvé úlohy KSP dostali stredoškoláci v školskom roku 1983/84. Bolo to zásluhou šťastnej náhody, že sa na Matematicko-fyzikálnej fakulte UK zišiel kolektív nadšencov z radov zamestnancov Ústavu aplikovanej matematiky, niekoľkých práve prijatých a jedného staršieho študenta Katedry informatiky a ich bývalého, neúnavného a nápadmi vždy sršiaceho stredoškolského učiteľa Oňa Demáčka. Patronát nad práve zrodeným KSP si zobral Jozef Hvorecký z Katedry informatiky. Postupom času prichádzali ďalší študenti, bývalí riešitelia KSP. Ústav aplikovanej matematiky vystriedal Ústav informatiky, neskôr Katedra umelej inteligencie, Katedra vyučovania informatiky, v súčasnosti patrí KSP pod krídla Katedry základov a vyučovania informatiky.

V roku 1983, kedy KSP vzniklo, neboli počítače také bežné. Programovanie sa vyučovalo len na zopár školách v celom Československu. Mikropočítače ako ZX 81, ZX Spectrum a PMD 85 vtedy iba začínali. Až neskôr sa začali rozširovať PC, PC XT a PC AT. Zo začiatku nebolo zriedkavosťou, že programy počítač vôbec „nevideli“, alebo, že boli spustené na sálových počítačoch EC 1010, EC 1021 alebo minipočítačoch SM-4 alebo ADT. Prvé riešenia prichádzali takmer výlučne v BASICu, pár ich bolo vo FORTRANe. Až neskôr sa použité jazyky postupne menili na Pascal, prípadne C. V súčasnosti sa už stretáme aj s C++ či Javou.

Od svojho vzniku organizovalo KSP niekoľko „generácií“ vysokoškolských študentov. Tie najstaršie z nich už súčasní organizátori často poznajú len po mene či prezývke. A súčasní riešitelia už ani netušia, že predlohou pre rôzne postavičky zo zadaní (trebárs Santa, Števa, Dana, Rasťa či malého Tadeáša...) boli niekdajší organizátori, ich známi a rodina.

V dobe, keď ide do tlače táto zbierka, pomaly začína dvadsiaty štvrtý ročník KSP. Už teraz je teda „Káespéčko“ staršie ako mnohí jeho organizátori... a všetci svorne dúfame, že nás KSP aj prežije `:-)`.

[1] Na pivo len plnoletých!

História zbierky úloh KSP

V roku 1996 po prvýkrát vyšla zbierka úloh KSP. Veľa, preveľa ľudí prispelo k jej vzniku svojou troškou, no predsa len jeden z nich bol ten najdôležitejší – Miško Winczer. Z jeho iniciatívy zbierka vznikla. A s odstupom času musíme povedať, že okrem toho, že vyrobil jej prvú verziu, mal na jej vzniku ešte jednu obrovskú zásluhu – dovtedy do organizátorov seminára tĺkol, kým neboli na svete všetky chýbajúce popisy riešení úloh.

V rôznych historických obdobiach tieto popisy riešení pomáhali spisovať Ivona Bezáková, Dušan Bezák, Broňa Brejová, Mišo Forišek, Ňaňka Gajdošíková (medzičasom už Katreniaková), Maťko Hajduch, Braňo Katreniak, Peter g00ber Košinár, Riško Královič, Meri Nanásiová, Martin Pál, Jano Senko, YoYo Šiška a Tomáš Vinař.

Ďalší ľudia strávili hodiny času vychytávaním chýb všetkého druhu, za všetkých spomeňme Janku Chlebíkovú, Ľudku Moravčíkovú, Zuzku Kubincovú, Danku Pardubskú, Peťa Borovanského, Ivana Kalaša, Ivicu Chačaturianovú, Martinu Hruščákovú a Maja Dvorského.

Zbierka mala medzi riešiteľmi úspech, a tak časom pribudlo druhé a tretie vydanie. Tie obsahovali o niekoľko ročníkov viac, pribudol chabý pokus o index na konci knižky, ale žiadne iné prevratné zmeny sa nekonali...

Až kým sa nedostala do rúk nám. Celé to začalo tým, že KuKova (čítaj: Kubova Kováčova) papierová zbierka už bola natoľko doškrtaná, že človek nevedel, ktorú vrstvu čítať ako prvú. Bola len jedna možnosť, čo v takej situácii spraviť... Áno, správne. Vytlačili sme si ešte dve kópie a doškrtali sme aj tie na nepoznanie. A pustili sme sa do písania. Nešlo to ľahko. No vždy, keď sa nám nakopilo iných povinností a na zbierku sme zabúdali, prišla Monika Steinová a nahnala nás späť do roboty. Patrí jej za to naša vďaka, aj jej zásluhou táto knižka uzrela svetlo sveta.

Výsledkom všetkých týchto útrap je teda táto knižka, ktorú práve držíš v ruke (či vidíš na svojom monitore). Čo sa v nej takého zmenilo oproti predchádzajúcim knihám, na ktorých obale boli slová „zbierka“ a „KSP“?

V prvom rade sme sa rozhodli, že historická vernosť nemôže stáť v ceste zrozumiteľnosti, a tak sme opravili chyby a nejasnosti v textoch zadaní. No a na popisoch riešenia snáď neostala suchá ani nitka. Skoro žiaden nezostal nedotknutý, veľa z nich sme radšej napísali odznova. Niekde to boli len kozmetické úpravy, niekde jasnejšie, podrobnejšie formulácie, no dosť bolo aj takých úloh, kde sme našli nové, lepšie riešenie ako to doteraz v zbierke uvedené, a dokonca v pár prípadoch sme v starom riešení našli a opravili chybu. A nesmieme zabudnúť ani na pár úloh, napríklad úloha O politikoch, či legendárna Santova rúra, ktoré v minulých verziách riešenie nemali vôbec.

V Bratislave, 20. septembra 2006

Michal Forišek

Návod na požutie

V prvej časti tejto zbierky nájdete samotné zadania úloh KSP. Ako sme už spomínali, niektoré z týchto zadaní sme mierne upravili, vyjasnili, či doplnili do nich vysvetľujúce obrázky.

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 desiatom a neskôr od štrnásteho ročníka KSP navyše pribudla ľahšia kategória Z pre „začiatočníkov“ – tých, ktorí v minulosti KSP neriešili. Tieto úlohy majú pred číslom úlohy uvedené písmeno **z**.

Úroveň teoretickej informatiky, a teda aj programátorských súťaží, vo svete ešte stále stúpa. Dôsledkom je, že obtiažnosť úloh v KSP sa postupom času zvyšovala. Ku koncu zbierky teda nájdete úlohy podstatne ťažšie ako na jej začiatku.

Zadania si však môžete čítať aj v inom poradí. Druhý spôsob objavíte o pár strán ďalej. 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 nájdete viacero rôzne šikovných riešení. V každom prípade by sa mal čitateľ snažiť najskôr vyriešiť úlohu bez využitia tejto časti, inak sa ochudobní o radosť z objavovania, nehovoriac o tom, že čo sami vymyslíme, to si aj najlepšie pamätáme.

No a úplne na konci sa nachádza index. V indexe nájdete 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 šírky*, nie je nič ľahšie ako nájsť si v indexe, v ktorých úlohách sa dá použiť.

Organizátori KSP

Aby aspoň mená organizátorov neupadli do zabudnutia, uvedieme tu menný zoznam všetkých, ktorí sa za posledných dvadsaťtri rokov nezanedbateľnou mierou podieľali na organizovaní KSP:

Peťo Ágh,
Mirko Baláž,
Miško Barabáš,
Marek 'Bee' Bernát,
Dušan Bezák,
Ivona Bezáková,
Zuzka Bezáková
(rod. Rjašková),
Marek Bezaniuk,
Danka Biernacká,
Andrej Blaho,
Peťo 'Boro' Borovanský,
Broňa Brejová,
Stano Buštor,
Andy Černík,
Monika Černíková
(rod. Obrancová),
Oňo Demáček,
Pišta Dobrev,
Maťo Domány,
Jana Dorotková,
Miro Dudík,
Lenka Fibíková,
Mišo 'MišoF' Forišek,
Peter Glaus,
Juro Gottweis,
Martin 'Maťoha' Hajduch,
Anička Hanulová,
Bea Hittnerová,
Katka Staudtová
(rod. Holešová),
Sisa Horváthová,
Jozef Hvorecký,
Martin 'Tino' Irman,
Dalibor Jakuš,
Ivan Kalaš,
Braňo Katreniak,
Janka 'Ňaňka' Katreniaková
(rod. Gajdošíková),
Slavo Kavka,
Michal Kevický,
Miro Kočan,
Peter 'g00ber' Košinár,
Vlado Koutný,
Kubo 'KuKo' Kováč,
Rasťo Královič,
Rišo Kráľovič,
Majka Kuchynárová,
Janka Lachová,
Peťo Lalík,
Julka Lipková,
Dada Lúčna
(rod. Hodáková),
Andy Lúčny,
Marek 'MMx' Ludha,
Martin Macko,
Martin 'MC' Makúch,
Igor Malý,
Eva Mariničová,
Mišo 'Mato' Matoušek,
Peter Minárik,
Peter 'Mižo' Miština,
Zuzka Mosná
(rod. Repaská),
Martin 'Moťo' Môťovský,
Paťo Mravec,
Mišo 'Mic' Nánási,
Mária 'Meri' Nánásiová,
Riško Nemec,
Jano 'WSX' Oravec,
Dávid Pál,
Martin 'Pálenica' Pál,
Pavel 'Pitro' Petrovič,
Pavol 'Paťočo' Petrovič,
Matúš 'Matis' Petruľák,
Milan Plžík,
Mišo 'Poko' Pokorný,
Lukáš Poláček,
Martin 'Rejdi' Rejda,
Erika Rudíková
(rod. Setteyová),
Maroš Rusnák,
Jano 'JanoS' Senko,
Danka Smažaková,
Milan Smolík,
Monika Steinová,
Jozef 'YoYo' Šiška,
Lukáš Špálek,
Rasťo Šrámek,
Tono Štefánek,
Dano Štefankovič,
Peto Tomcsányi,
Jozef 'Žužu' Tvarožek,
Jano Války,
Peťo 'Jerevan' Varša,
Evka Vasilová,
Igor Vavro,
Tomáš 'Vinko' Vinař,
Marián Vittek,
Miško Winczer,
Tomáš 'Tom' Záthurecký,
Marek 'Zemčo' Zeman,
Ľubo Žiško.

Menný zoznam príkladov

Tematické rozdelenie príkladov

Príklady 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 príklady sú natoľko osobité, že ich je ťažké niekam zaradiť, v iných sa zase stretajú aj dve-tri rôzne oblasti.

Na tomto mieste skúsime stručne popísať jednotlivé oblasti, do ktorých budeme príklady deliť.

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 (napr. utriedené pole, binárne stromy, hašovaciu tabuľku). Patria sem aj rôzne štatistické funkcie, napr. medián či modus.

Príklady: 415, 431, 432, 644, 711, 715, 724, 811, 921, 1013, 1031, 1033, z1014, z1042, z1043, 1143, 1144, 1232, 1242, 1315, 1322, 1333, 1341, 1342, 1423, z1423, 1511, 1531.

Grafové algoritmy. Na riešenie týchto úloh sú potrebné poznatky z teórie grafov. Patria sem napríklad úlohy o najkratších cestách, najlacnejších kostrách, stromoch, párovaniach... 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.

Príklady: 224, 315, 324, 433, 515, 523, 524, 533, 614, 633, 643, 712, 733, 813, 814, 823, 831, 834, 835, 912, 914, 924, 942, 1014, 1021, 1022, 1024, 1025, 1042, 1044, 1045, z1021, 1124, 1134, 1224, 1233, 1243, 1311, 1314, 1344, 1413, 1414, 1415, 1421, 1424, 1434, 1444, 1445, z1413, z1422, z1433, 1513, 1514, 1522, 1523, 1533, 1543, 1544, z1513, z1523, z1533, z1534.

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.

Príklady: 213, 233, 412, 522, 541, 611, 621, 634, 721, 833, 925, 945, 1011, 1012, 1015, 1042, 1213, 1223, 1231, 1241, 1343, 1412, 1442, 1443, z1421, 1534, z1531.

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.

Príklady: 213, 613, z1023, 1215, 1331, z1535.

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 (napr. rekurzívnymi volaniami) postupne vyskúšame. Ak žiadna z nich neviedla k cieľu, vrátime sa „o úroveň vyššie“.

Príklady: 211, 224, 335, 411, 423, 432, 715, 731, 825, 841, 913, 934, z1032, 1123, 1214, 1313, 1323, 1334, 1422, 1432, z1424, 1534, 1542.

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.

Príklady: 125, 212, 223, 233, 412, 414, 434, 543, 611, 634, 1041, z1013, z1031, 1141, 1211, 1221, 1231, 1443, 1512, 1532.

Výpočtová geometria. 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.

Príklady: 413, 421, 542, 621, 623, 635, 735, 822, 832, 843, 845, 923, 931, 935, 943, 944, 1023, 1035, 1043, 1114, 1234, 1314, 1411, 1431, 1432, 1441, 1541.

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.

Príklady: 124, 323, 331, 821, 1235, z1512, z1521, z1522, z1532.

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 napr. 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 `C, G, A, T`...

Príklady: 512, 533, 534, 544, 625, 645, 725, 732, 1034, z1033, 1111, 1131, 1135, 1145, 1222, 1225, 1442, 1521.

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“.

Príklady: z1432.

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 analýzy je to napríklad hľadanie extrémov funkcií, či riešenie nelineárnych rovníc. A v podobnom duchu by sa dalo zapísať ešte niekoľko strán...

Príklady: 122, 222, 314, 315, 321, 322, 333, 334, 411, 525, 532, 624, 631, 722, 723, 812, 845, 911, 1032, 1041, z1044, 1133, 1245, 1332, 1345, 1443, z1414, 1545, z1514, z1515.

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.

Príklady: 113, 114, 121, 221, 231, 313, 332, 414, 422, 513, 612, 641, 1125, z1415, z1425, z1435.

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.

Príklady: 1112, 1122, 1132, 1142, 1312, 1415, 1425, 1435, 1515, 1525, 1535.

Kódovanie a šifrovanie. Kódovanie informácii do núl a jednotiek je staré ako počítače samé, dokonca ešte o čosi staršie. Časom prišla potreba zakódovanú informáciu prená-

šať, a tak vznikla kompresia a samoopravné kódy. Potreba utajenia obsahu správy pred nepovolanými osobami viedla k rozvoju šifrovania.

Príklady: 225, 235, 613, 714, 932, z1031, 1211, z1434.

Aproximačné a randomizované algoritmy. Niektoré problémy sú natoľko ťažké, že ich nevieme efektívne riešiť. Ako na potvoru, práve takéto úlohy sa často vyskytujú v praxi. Do tejto kategórie sme zaradili úlohy, ktoré ukazujú iné možné prístupy k takýmto úlohám. Aproximačné algoritmy sa snažia efektívne nájsť riešenie, ktoré nemusí nutne byť optimálne, ale ktoré určite bude mať k optimálnemu dostatočne blízko.

Randomizované algoritmy rôznymi spôsobmi využívajú k dosiahnutiu dobrého výsledku náhodu. Niektoré randomizované algoritmy vedia napríklad náhodnými voľbami počas výpočtu zabezpečiť, že nám nepriateľ nevie podstrčiť vstup, ktorý by náš algoritmus veľmi spomalil.

Samostatnú kategóriu tu tvoria algoritmy, ktoré sú síce vďaka využívaniu náhody veľmi efektívne, ale môžu sa občas pomýliť. Aj takýto algoritmus sa dá v praxi využiť, stačí, ak vieme ohraničiť pravdepodobnosť, s akou sa mýli.

Príklady: 1113, 1315, 1433, z1414.

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 `:-)`

Príklady: 112, 115, 123, 214, 232, 234, 311, 312, 325, 424, 425, 435, 511, 514, 521, 525, 531, 535, 545, 615, 622, 632, 642, 713, 734, 815, 824, 825, 842, 844, 915, 933, 941, 1021, z1011, z1024, z1034, z1041, 1212, 1232, 1244, z1412, z1431, 1524.

Zadania

111. O prvočíslach

Zostavte program, ktorý rozloží dané prirodzené číslo n na prvočinitele a vypíše

a) číslo n

b) text `je prvočíslo`, alebo `je deliteľné prvočíslom` i `presne` j`-krát`, pre všetky prvočísla i a ich násobky j v rozklade.

112. Zhusťovanie poľa

V poli $A[1\,..\,n]$ je veľa núl. Zhustite prvky v poli A tým, že všetky nuly z neho odstránite. Počet nenulových prvkov zhusteného poľa bude uložený v premennej k. Vypíšte

a) obsah pôvodného poľa

b) obsah zhusteného poľa a počet jeho nenulových prvkov

113. Fibonacciho čísla

Fibonacciho[2] *čísla* sú čísla definované rekurentnou postupnosťou:

$$F_0 = 0, \qquad F_1 = 1, \qquad F_{n+2} = F_{n+1} + F_n, \quad \text{pre } n \geq 0.$$

Každé Fibonacciho číslo je teda súčtom dvoch predchádzajúcich. Na výpočet hodnôt Fibonacciho čísel však existuje aj iný vzorec:

$$F_k = \frac{\phi^k - \hat{\phi}^k}{\sqrt{5}}, \quad \text{kde } \phi = \big(1 + \sqrt{5}\big)/2 \text{ (zlatý rez) a } \hat{\phi} = \big(1 - \sqrt{5}\big)/2.$$

Úloha: Zostavte program, ktorý bude počítať prvých 15 Fibonacciho čísel oboma spôsobmi a vytlačí

a) presnú hodnotu podľa prvého vzorca

b) čo najpresnejšiu hodnotu podľa druhého vzorca

c) absolútnu odchýlku, t.j. absolútnu hodnotu ich rozdielu

114. Generátor náhodných čísel

Generátorom pseudonáhodných čísel rozumieme postupnosť, v ktorej sa členy striedajú natoľko nepravidelne, že ich poradie je pre toho, kto nepozná predpis, ktorý poradie udáva, úplne náhodné. Dobrými generátormi pseudonáhodných čísel sú

i) pre 32-bitový počítač, podľa [24],

$$x_{n+1} = (x_n \cdot 5621 + 1) \bmod 65536,$$

ktorý generuje celé čísla z rozsahu 0 až 65535 (mod znamená zvyšok po delení – preto rozsah nemôže prekročiť 65535).

ii) Pre počítač s reálnou aritmetikou

$$x_{n+1} = \text{desatinná časť čísla } (x_n + \pi)^5,$$

ktorý generuje reálne čísla z intervalu $\langle 0, 1)$, pretože výsledkom je zvyšok po odtrhnutí celej časti.

---

[2] Leonardo Pisano (1180–1240), nazývaný aj Fibonacci, najväčší predrenesančný európsky matematik. Postupnosť F_n pomenoval jeho menom až francúzsky matematik François Édouard Anatole Lucas (1842–1891).

Úloha: Zostavte generátor náhodných čísel ako podprogram, ktorý bude generovať s rovnakou pravdepodobnosťou niektoré z čísel 1, 2, 3, 4, 5 alebo 6. Môžete skúsiť použiť jeden z vyššie uvedených generátorov tak, že interval, z ktorého generuje čísla, rozdelíte na 6 rovnakých častí.

Takýto generátor teda môže nahradiť hraciu kocku. Predstavme si teraz, že hádžeme šiestimi kockami. Pri každom vrhu môže padnúť jedno z čísel 6 až 36. Hoďte kockami 100, 300, 500-krát a zistite, koľkokrát padol každý z 31 možných súčtov. Výsledky vypíšte a vykreslite pomocou rôzne dlhých úsečiek.

115. Postupnosť

Zostavte program, ktorý bude vo vzostupnom poradí vytvárať 100 najmenších čísel v množine M, ak je M definovaná nasledujúco:

i) číslo 1 je z M

ii) ak číslo x je z M, tak aj čísla $2x + 1$ a $3x + 1$ sú z M

iii) žiadne iné čísla, než tie, ktoré vzniknú pomocou pravidiel i) alebo ii), nepatria do M.

Na ukážku uvedieme prvých sedem prvkov množiny $M : 1, 3, 4, 7, 9, 10, 13, \ldots$.

121. Záhada piatich pokladníc

TJ Rozpaky nad Ofsajdom usporiadala na svojom štadióne majáles. Pokladník oddielu umiestnil pri každej zo štyroch brán štadióna jedného svojho pomocníka s pokladnicou. Keď sa zdalo, že už ďalší návštevníci neprídu, obišiel všetkých pomocníkov, preložil peniaze z ich pokladníc do svojej a oddal sa zábave.

Keď už bol mierne „v nálade“, uvedomil si, že ak by stratil kľúče, nálezca by sa mohol ľahko zmocniť tržby. Preto pokladnice pootváral, do každej hodil po jednom kľúči a opäť ich zatvoril. Tým ich aj zamkol, lebo sa zamykali automaticky.

Až ráno (po vytriezvení) si uvedomil dôsledky svojho činu: veď do pokladníc sa nedostane ani on sám! To ale znamená, že najmenej jednu pokladnicu musí násilne vylomiť, aby sa dostal k niektorému kľúču. Tým kľúčom bude môcť otvoriť ďalšiu pokladnicu (ak to náhodou nebude kľúč od práve vypáčenej pokladnice); v nej bude ďalší kľúč, atď.

Úloha:

a) Pomocou generátora pseudonáhodných čísel nasimulujte náhodné rozloženie kľúčov v piatich pokladniciach. Zistite, koľko pokladníc treba otvoriť násilím na to, aby sme ich otvorili všetky.

b) Zopakujte pokus 100-krát a zistite, koľkokrát musel pokladník vylomiť práve jednu, práve dve, práve tri, štyri, päť pokladníc, aby otvoril všetky. Aký je priemerný počet vylomených pokladníc?

122. O zlomkoch

Pre dané celé čísla m, n vypočítajte podiel m/n ako reálne číslo na taký počet desatinných miest, aby ste skončili delenie vtedy, keď:

i) ďalšie desatinné miesta obsahujú iba nuly,

ii) perióda v desatinnej časti sa začne opakovať. V tomto prípade určite aj dĺžku periódy.

Výpočet odladťe na aspoň 5 podstatne odlišných dvojiciach, napríklad 523 a 19, 1 a 17, 10192 a 52, a pod.

123. Rotácia poľa

Dané je n-prvkové pole A (indexované od 0 po $n - 1$). Zostavte program, ktorý pre dané celé číslo k, $-n < k < n$ posunie prvky poľa „do kruhu“. Pre kladné k to znamená, že prvok $A[0]$ sa presunie na miesto $A[k]$, prvok $A[1]$ na miesto $A[k + 1]$, atď. Na mieste $A[0]$ bude prvok, ktorý bol pôvodne na mieste $A[n - k]$, na mieste $A[k - 1]$ ten, ktorý bol na mieste $A[n - 1]$. Ak teda v 8-prvkovom poli boli prvky 1 2 3 4 5 6 7 8 a $k = 3$, tak

výsledkom „rotácie poľa“ bude 6 7 8 1 2 3 4 5. Pre $k = 0$ všetky prvky zostanú na svojich miestach. Pre záporné k pôjde o rotáciu o $|k|$ prvkov opačným smerom.

124. O štvorci

Dané je číslo n, $20 < n < 100$. Zostavte program, ktorý vypíše čísla $1, 2, 3, \ldots, n$ „po obvode štvorca“ (so stranou dĺžky približne $n/4$). Pre nepárne n bude na ľavej strane štvorca jedno číslo chýbať. Pre niektoré n bude treba spraviť „štvorec“ s o jedno menšou „šírkou“ ako „výškou“. Napríklad pre $n = 21$ bude mať „štvorec“ podobu ako na obrázku.

```
 1  2  3  4  5  6
                  7
21                8
20                9
19               10
18               11
17 16 15 14 13 12
```

125. O kódovaní permutácií

Pod *permutáciou n čísel* rozumieme takú n-ticu, že každé z čísel $1, 2, 3, \ldots, n$ sa v nej nachádza práve raz. Permutácie môžeme usporiadať na podobnom princípe ako mená v telefónnom zozname. Toto usporiadanie nazývame *lexikografickým usporiadaním*. Jeho princíp je nasledujúci:

Permutácia $a_1, \ldots, a_n$ je v zozname permutácií pred permutáciou $b_1, \ldots, b_n$ vtedy a len vtedy, keď existuje taký index i, že a_i a b_i sú prvé členy, v ktorých sa permutácie odlišujú a $a_i < b_i$.

Príklad: Pre permutácie piatich čísel je permutácia 3, 2, 4, 1, 5 pred permutáciou 4, 1, 5, 3, 2 (lebo $3 < 4$) a pred permutáciou 3, 2, 5, 4, 1 (pretože prvé dva členy sa rovnajú a $4 < 5$).

Teraz si predstavte, že všetkých p permutácií n čísel je napísaných v stĺpci na papieri podľa predchádzajúceho usporiadania. Očíslujeme permutácie od najmenšej po najväčšiu číslami $0, 1, \ldots, p - 1$.

Úloha: Zostavte program, ktorý

a) keď dostane na vstupe permutáciu n čísel, určí jej poradové číslo v zozname permutácií príslušného n. Toto číslo budeme nazývať kód permutácie.
b) keď prečíta kód permutácie a príslušné n, vytlačí permutáciu zodpovedajúcu tomuto kódovému číslu.

Riešenie, ktoré využíva tabuľku všetkých permutácií v pamäti, pokladáme za „škaredé“.

211. Úloha sto

Na papieri je napísané dlhé číslo 123456789. Vašou úlohou je medzi niektoré dvojice cifier vsunúť znak + alebo − tak, aby ste po vyhodnotení výrazu dostali číslo 100. Ak medzi ciframi nie je vsunutý žiadny znak, chápu sa ako súvislá konštanta. Vymieňať poradie cifier nesmiete.

Príklad: $100 = 123 - 45 - 67 + 89$

Úloha: Napíšte program, ktorý nájde a vypíše všetky možnosti vsunutia znakov + a − do reťazca a na konci vypíše počet riešení.

Ak váš program bude dobre napísaný (použijete jasnú dobrú ideu), tak by ste mali byť schopní ľahko do programu dorobiť aj iné aritmetické operácie.

212. O zátvorkách

Každý školáčik vie, čo sú pekne uzátvorkované aritmetické výrazy – sú to výrazy, v ktorých sa ku každej ľavej zátvorke neskôr vo výraze vyskytuje jej zodpovedajúca pravá zátvorka.

Napríklad `(()())()` je dobre uzátvorkovaný výraz, ale `())`, `)()(`, `(((`, ... nie sú dobre uzátvorkované výrazy. Presnejšie povedané:

i) Výraz `()` je dobre uzátvorkovaný.
ii) Ak výraz X je dobre uzátvorkovaný, tak `(`X`)` je dobre uzátvorkovaný výraz.

iii) Ak X, Y sú dobre uzátvorkované výrazy, tak XY je dobre uzátvorkovaný výraz.
iv) Žiadne iné výrazy ako tie, ktoré vzniknú pomocou pravidiel i)–iii), nie sú dobre uzátvorkované.

Úloha: Máte k dispozícii n párov zátvoriek `(`, `)`. Zostavte program, ktorý vypíše všetky dobre (pekne) uzátvorkované výrazy, ktoré z nich môžeme vyrobiť.

Napríklad ak $n = 3$, tak vypíšte `((()))`, `(()())`, `()(())`, `(())()`, `()()()`.

213. O súčte

Daná je postupnosť n celých (kladných i záporných) čísel. Nájdite v nej podpostupnosť (ľubovoľnej dĺžky $\leq n$) po sebe nasledujúcich členov postupnosti s najväčším súčtom.

Dobre pošpekulujte nad najlepším a najrýchlejším riešením. Problém je ťažší, než sa na prvý pohľad zdá.

214. O mocninách cifier

Vezmime si prirodzené číslo, rozdeľme ho na cifry a vypočítajme súčet druhých mocnín týchto cifier. Dostaneme nové číslo, s ktorým postup opakujeme. Po konečnom počte opakovaní tohto postupu dostávame číslo 1 alebo 4. Ak napríklad začneme s číslom 324, máme

$$\begin{aligned}
a_1 &= 324 \\
a_2 &= 3 \cdot 3 + 2 \cdot 2 + 4 \cdot 4 = 29 \\
a_3 &= 2 \cdot 2 + 9 \cdot 9 = 85 \\
a_4 &= 8 \cdot 8 + 5 \cdot 5 = 89 \\
a_5 &= 8 \cdot 8 + 9 \cdot 9 = 145 \\
a_6 &= 1 \cdot 1 + 4 \cdot 4 + 5 \cdot 5 = 42 \\
a_7 &= 4 \cdot 4 + 2 \cdot 2 = 20 \\
a_8 &= 2 \cdot 2 + 0 \cdot 0 = 4 \qquad \ldots \text{koniec postupu}
\end{aligned}$$

Úloha: Zistite, ktoré čísla z rozsahu 1 až 1000 končia po aplikovaní uvedeného postupu číslom 1.

215. O písmenkách

Pracovníci oddelenia počítačovej literatúry potrebujú ako podklad pre svoju novú knihu tabuľku výskytu jednotlivých znakov v slovenskom texte. Preto vás prosia o spoluprácu. Vašou úlohou bude napísať program, ktorý vypíše tzv. frekvenčnú tabuľku.

Úloha: Vezmite nejaký súvislý text v slovenčine (z novín, knihy, časopisu a pod.), dlhý cca. 200 slov. Pre účely programu chápte aj všetky ďalšie znaky (čiarka, bodka, medzera a pod.) ako písmená slovenskej abecedy. Pre každé písmeno abecedy zistite počet jeho výskytov v texte. Výslednú tabuľku usporiadajte podľa počtu výskytov a oznámte percentuálny podiel znaku oproti počtu všetkých znakov v texte.

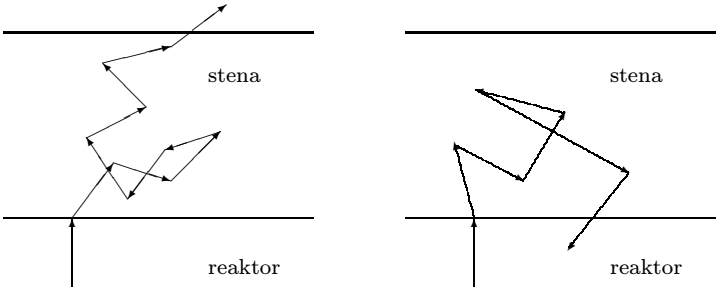
221. O modelovaní

Okolo atómového reaktora treba vybudovať ochrannú betónovú stenu, aby rádioaktívne častice nezamorovali okolie. Vieme, že hrúbka steny 5 metrov je postačujúca, lebo energia častíc vyletujúcich z reaktora sa zmenšuje kolíziami s časticami v betóne, ktoré ich zabrzdia. Na časticu v betóne narazia každých 25 cm a 20 takých nárazov časticu úplne zastaví. (V skutočnosti to platí len „v priemere“, ale pre účel nášho modelovania predpokladáme, že je to vždy tak.) Po každom náraze na časticu v betóne sa častica odchýli od smeru, ktorým sa pohybovala do stretnutia, o uhol α. Jeho veľkosť je náhodná veličina z intervalu $(-\pi, \pi)$. V skutočnosti teda nemusí byť stena taká hrubá, lebo dráha častice

v stene nevedie priamo von z reaktora. V tenšej stene sú možné tri prípady, z nich i) a ii) sú znázornené aj na obrázku dole:

i) častica sa vráti späť do reaktora,
ii) častica preletí cez stenu,
iii) častica 20-krát narazí na časticu v betóne a zastaví sa.

Úloha: Napíšte program, ktorý bude simulovať prechod častice betónovou stenou. Častice vypúšťajte do steny zo strany reaktora tak, že počiatočná vzdialenosť častice od okraja steny je nulová. Náhodne zvoľte aj uhol, pod ktorým častica do steny vletí. Vypíšte percentuálny podiel každého prípadu i), ii), iii), pre n (aspoň niekoľko sto) vypustených častíc a hrúbku steny 2, 3, 4, 5 metrov.



222. O zlomkoch

Napíšte program, ktorý pre dané reálne čísla x a e vypočíta nesúdeliteľné celé čísla a a b tak, aby $|x - a/b| \leq e$, t.j. aby sa reálne číslo dalo nahradiť zlomkom a/b s presnosťou e.

Príklad: Pre $x = 2.718$ a $e = 0.01$ je $a = 19$ a $b = 7$. Skúste pre $x = 3.1415926$, $e = 0.001$ a pre $e = 0.0001$.

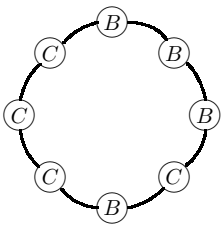
223. O inverziách v permutácii

Nech $a_1, a_2, \ldots, a_n$ je permutácia množiny čísiel $1, 2, 3, \ldots, n$. Tabuľkou inverzií permutácie $a_1, a_2, \ldots, a_n$ sa nazýva postupnosť čísel $b_1, b_2, \ldots, b_n$, kde b_j je počet prvkov väčších ako j a nachádzajúcich sa vľavo od j v permutácii $a_1, a_2, \ldots, a_n$. Vymyslite algoritmus vhodný pre realizáciu na počítači, ktorý podľa danej tabuľky inverzií $b_1, b_2, \ldots, b_n$ skonštruuje permutáciu $a_1, a_2, \ldots, a_n$.

Príklad:

	a_1	a_2	a_3	a_4	a_5
permutácia	2	5	4	1	3
tabuľka inverzií	3	0	2	1	0
	b_1	b_2	b_3	b_4	b_5

224. O náhrdelníkoch



Princezná Arabela sa rada parádila pred dvornými programátormi, a preto nosila do „výpočťáku“ každý deň iný „pekný“ náhrdelník. Páčili sa jej len také dvojfarebné náhrdelníky z perál, že programátori, ktorí sa okolo nej krútili, na ňom mohli nájsť ľubovoľnú farebnú podpostupnosť dĺžky n (vstupný údaj).

Príklad: $n = 3$; Na obrázku je náhrdelník, ktorý princezná jeden deň nosila. Je naozaj „pekný“, lebo z neho môžeme prečítať všetkých $2^3 = 8$ variácií s opakovaním dĺžky 3 z dvoch prvkov

(čítame v smere chodu hodinových ručičiek). Náhrdelník neotáčame naopak ani nerozdeľujeme. Teda prečítame BBB, BBC, BCB, CBC, BCC, CCC, CCB, CBB. Pre istotu si skontrolujte, či sme vypísali všetky možné trojprvkové variácie dvoch farieb (nemusia byť v tradičnom poradí). Toto bol jediný úspešný krok Arabely preniknúť do tajov dvojkovej sústavy, a tak ovládnuť výpočtovú techniku.

Úloha: Arabela vždy nosí náhrdelník dĺžky 2^n a programátori hľadajú postupnosti dĺžky n (ukázané pre $n = 3$). Zistite, koľko a akých pekných náhrdelníkov má, keď viete, že sú navzájom rôzne. Za rôzne považujeme aj také, ktoré vzniknú rotáciou, alebo symetrickým preklopením šperku. Pokúste sa nájsť algoritmus, ktorý neprehľadáva všetkých 2^{2^n} možností – to je príliš veľa.

225. O šifrovaní

Pomocou rôznych programov a iných pomôcok sa pokúste rozšifrovať nasledujúci text. Text vznikol zo slovenského textu knihy, ktorej názov tiež môžete hádať. Zachovali sme interpunkčné znamienka, t.j. čiarky, bodky, výkričníky, otázniky, atď. Nahradzované boli len písmená `A` až `Z` tak, že napr. všetky výskyty písmena `F` sa nahradili písmenom `T`. Teda, že každé `T` znamená `F`, a zároveň žiadne dve písmenká sa nezobrazili na to isté písmenko. Matematici hovoria, že sme urobili len permutáciu písmen abecedy. Mäkčene a dĺžne sa nebrali do úvahy a nerozlišovali sme medzi veľkými a malými písmenami. Môžete využiť frekvenčnú tabuľku, ktorú ste si vyrobili v úlohe 215.

Tu je text:

```
UEUSF PLOTK.
PLOMTCSP HK AYM OKUKTPEYM FPKNLTPR, PNEIM FK EA FMGK YRIKOCM
TSFSK CKTMUPKHS. DMACK FPKNLTPK HK CKTMUPL FYMNTL, AILJK NHKYL.
EGM FPKNLTPR FK AKDL CK UIYR UEJTKA TKJPE IEOEOCKN. UIMAYKAOKDLVS
OKGKTS FPKNLTPL FE FYMNTEL CKTMUPEL AE CEYSCEYMJE UKUSMIK K
UETEOS DL EA FMGK CK TKYE. UENEH OKGKTS AE CEYSCEYMJE UKUSMIK
AILJL FPKNLTPL, PNEIK HK NHKYL CKTMUPL, K UETEOS DL EA FMGK
CKUIKYE. YMOHM PLOMTCL UKTSVPL K LIEGS CKA EGEHK FPKNLTPKHS
CSMPETPE PLOMTCRVJ NKJEY. PMA CEYSCR IEOGKTS LPKOM FK, OM
FPKNLTPK FE FYMNTEL CKTMUPEL FK CKVJKOK YUIKYE K FPKNLTPK
F NHKYEL CKTMUPEL YTKYE. EGSAYM FPKNLTPR FS YRHMCSTS HSMFNE.
CK UIMYMAMCSM PLOTK UENIMGLDMHM AYM UIMUKIEYKCM OKUKTPEYM
FPKNLTPR K AYK PLFR CEYSCEYMJE UKUSMIK. EGFNKIKHM FS AYM
FPKNLTPR FE FYMNTEL K AYM FPKNLTPR F NHKYEL CKTMUPEL. O DMACMD
FYMNTMD K O DMACMD NHKYMD FPKNLTPR CKTMUPR EAHEVSHM K FNSKJCMHM.
YRFLFSHM SVJ K CMUEFPEAMCM CKTMUSHM CK AYK EGATOCSPR O
PIMFTSKVMJE UKUSMIK. NSM HLFSK GRN IEYCKPE YMTPM KPE OKUKTPEYK
FPKNLTPK. EGATOCSPR F CKTMUMCRHS CKTMUPKHS HLFSHM OKNKOSN K
AEPECKTM YRFLFSN, KGR FK CMPISYSTS. UENEH SVJ LUIKYSHM NKP, KGR
GETS UIMFCRH EGIKOEH YIVJCMD FNIKCR FPKNLTPR. KP HK NMAK YIVJCK
FNIKCK FPKNLTPR EPETE CKTMUPR HEAIR UKFSP, HLFSHM JE CKHKTEYKN
S CK UKUSMIEYR EGATOCSP EPETE DMJE CKTMUPR. UENEH CK FPKNLTPL
FE FYMNTEL CKTMUPEL UETEOSHM EGATOCSP F CKTMUPEL NHKYEL, K OKFM
CKEUKP CK FPKNLTPL F CKTMUPEL NHKYEL UETEOSHM EGATOCSP FE
FYMNTEL CKTMUPEL. EGM FPKNLTPR UETEOSHM CK FNET NKP, KGR GETS
HSHE AEFKJL ASYKPEY. EPIMH FPKNLTSMP UETEOSHM CK FNET AYK
JKIPR CEYSCEYMJE UKUSMIK.
Y VEH DM YTKFNCM UEAFNKNK PLOTK? PLOMTCSP YMOHM FPKNLTPL, CK
PNEIMD TMOS EGATOCSP F CKTMUMCEL FYMNTEL CKTMUPEL, KTM FPKNLTPK
HK Y FPLNEVCEFNS CKTMUPL NHKYL. OAEIKOCS OM GKTS FYMNTL
```

FPKNLTPL K UETEOS DL EA FMGK CK TKYE AILJL FPKNLTPL, CK PNEIMD
TMOS EGATOCSP F NHKYEL CKTMUPEL KTM FPKNLTPK HK Y FPLNEVCEFNS
FYMNTL. CKTMUPL OKGKTS AE AILJMJE CEYSCEYMJE UKUSMIK K UETEOS
DL EA FMGK CKUIKYE. CMOKGLACM AEAKN, OM NHKYK FPKNLTK TMOS
YUIKYE. YMOHM PLOMTCL UKTSVPL K LIEGS CKA EGEHK OKGKTMCRHS
FPKNLTPKHS CSMPETPE HKBSVPRVJ NKJEY. EOCKHS, OM UE NRVJNE
NKJEVJ FS FPKNLTPR YRHMCSTS HSMFNE. CK AEPKO FYEDJE NYIAMCSK
IEOGKTS CKDUIY TKYL FPKNLTPL. UIS IEOGKTEYKCS CMVJK Y CEYSCEYEH
UKUSMIS S EGATOCSP FE FYMNTEL CKTMUPEL. UE IEOGKTMCS FK NMAK
EGDKYS YTKYE FPKNLTPK F NHKYEL CKTMUPEL. UIS IEOGKTEYKCS NHKYMD
FPKNLTPR CMVJK Y CEYSCEYEH UKUSMIS EGATOCSP F NHKYEL CKTMUPEL.
NKPOM NMIKO DM YUIKYE FPKNLTPK FE FYMNTEL CKTMUPEL. EGSAYM
FPKNLTPR FS NMAK YRHMCSTS HSMFNE.
AETMOSNM LUEOEICMCSM!
FE FPKNLTPKHS, CK PNEIRVJ YETCM TMOSK EGATOCSPR F SCRHS
CKTMUPKHS NIMGK HKCSULTEYKN EUKNICM, KGR FK EGATOCSPR CMUEFLCLTS
NRH GR FK UIMOIKASTK UEAFNKNK NISPL.

231. O ťažobnej spoločnosti.

Ťažobná spoločnosť sa rozhodla investovať 10 miliónov korún do nového náleziska ropy. Prípravná technická skupina podala tri alternatívne návrhy postupu:

i) Na existujúcom vrtnom zariadení vykonať 10 vrtov, každý v cene 1 mil. Kčs. Pravdepodobnosť úspešného vrtu týmto zariadením je 25% a v prípade úspešného vrtu dá takýto zdroj 10 000 ton ropy ročne.

ii) Kúpiť nové prieskumné zariadenie za 3 mil. Kčs. Po predbežnom prieskume vykonať 7 vrtov (po 1 mil. Kčs). Pravdepodobnosť úspešného vrtu stúpne na 50%, kapacita zdroja bude 10 000 ton ročne.

iii) Kúpiť nové prieskumné zariadenie za 3 mil. Kčs a novú vrtnú súpravu za 4 mil. Kčs. Na vykonanie vrtov síce ostane iba 3 mil. Kčs (t.j. 3 vrty), ale pravdepodobnosť úspechu bude 50% a kapacita zdroja stúpne na 18 000 ton ročne.

Úloha: Napíšte program, ktorý simuluje 100 pokusov pre každú z troch stratégií a pre každú stratégiu vypíše priemerný ročný výťažok zo všetkých úspešných vrtov a priemerný čas (v mesiacoch), za ktorý sa vyťažilo 100 000 ton ropy. Za najlepšiu stratégiu vyhlási tú, pri ktorej je tento čas najmenší.

Dobre si rozmyslite, ako sa dá pomocou generátora náhodných čísel z intervalu $\langle 0, 1)$ generovať (áno, nie) tak, aby pravdepodobnosť „áno“ bola p% (25 alebo 50).

232. O trinástom

Iste všetci poznáte poveru o nešťastnej trinástke. Niektorí ľudia sa boja vyjsť trinásteho von, aby sa im niečo nestalo. Tobôž, keď padne trinásteho na piatok.

Úloha: Napíšte program, ktorý vyrobí takúto štatistiku o trinástom dni v mesiaci: pre každý deň v týždni (pondelok až nedeľa) zistí, koľkokrát v dvadsiatom storočí (t.j. od 1. 1. 1901 do 31. 12. 2000) pripadol na tento deň dátum trinásteho v mesiaci. Program má výsledky vypísať v tabuľke.

Pomôcka: 1. 1. 1901 bol utorok

233. O postupnostiach

Napíšte program, ktorý vypíše pre dané n, $1 \leq n \leq 12$, počet takých postupností obsahujúcich len nuly a jednotky dĺžky n, v ktorých sa nevyskytujú za sebou dve nuly.

Skúste program napísať tak, aby neskúšal všetkých 2^n postupností dĺžky n.

234. O troch skupinách

V poli P je n celých čísel – kladné, záporné a práve jeden nulový prvok. Napíšte program, ktorý „utriedi“ toto pole tak, že na jeho začiatku budú všetky záporné prvky, potom nulový prvok a na konci budú všetky kladné prvky. Na poradí čísel v jednotlivých skupinách nezáleží.

Pokúste sa vymyslieť algoritmus, ktorý nepoužije všeobecné triedenie.

235. O šifrovaní mriežkou

V úlohe 225 sme sa zoznámili s jednou z najjednoduchších metód šifrovania – jednoduchou substitučnou šifrou. Teraz si popíšeme inú šifrovaciu metódu: *šifrovacou mriežkou* $n \times n$ pre párne číslo n nazveme štvorec (napríklad z papiera) rozdelený na $n \times n$ štvorčekov. Z týchto štvorčekov je $n^2/4$ štvorčekov vystrihnutých. Správna šifrovacia mriežka je taká, ktorú keď položíme na iný štvorec s $n \times n$ políčkami a postupne ju otáčame o 90 stupňov v smere hodinových ručičiek (čím dostaneme 4 polohy), tak nám vystrihnuté okienka odkrývajú vždy iné štvorčeky.

Správnou šifrovacou mriežkou sa dá zašifrovať text o $n \times n$ písmenách takto: nakreslíme štvorec so stranou n a urobíme v ňom sieť $n \times n$ políčok. Na tento štvorec položíme šifrovaciu mriežku. Cez jej vystrihnuté okienka začíname písať postupne písmená šifrovaného textu. Keď vyplníme prvých $n^2/4$ okienok, otočíme mriežku o 90 stupňov v smere hodinových ručičiek. Pretože je to správna šifrovacia mriežka, odkryje nám iných $n^2/4$ políčok v našom štvorci, kam môžeme napísať ďalšie písmená šifrovaného textu. Po dopísaní druhej časti mriežku znovu pootočíme a po dopísaní tretej časti ju pootočíme ešte raz. Takto dostaneme zašifrovaný text v tvare štvorca $n \times n$ zloženého z písmen. Postup dešifrovania je zrejmý (za predpokladu, že máme rovnakú mriežku a vieme jej prvú polohu).

Úloha: Napíšte program, ktorý pre dané párne n, $4 \leq n \leq 10$, zašifruje daný text dĺžky n^2 pomocou danej šifrovacej mriežky, ak je mriežka správna, alebo vypíše oznam, že mriežka je nesprávna.

Mriežka môže byť zadaná napríklad v tvare matice $n \times n$ núl a jednotiek (1 je vystrihnutý štvorček, 0 je nevystrihnutý štvorček).

V prípade nejasností o spôsobe šifrovania odporúčame knižku Julesa Verna, Matej Sandor – Nový gróf Monte Christo.

311. O spoločnom prvku

Napíšte program, ktorý pre dané dvojrozmerné pole P celých čísel s m riadkami a n stĺpcami, v ktorom je každý riadok usporiadaný vzostupne, nájde a vypíše všetky čísla, ktoré sa nachádzajú v každom riadku. Ak také číslo neexistuje, program o tom podá správu.

Príklad: Pre $m = 4$, $n = 5$ a dané pole P:

2	3	5	7	9
4	6	9	11	13
1	3	7	9	10
3	3	4	8	9

je výsledok 9, lebo toto číslo sa nachádza v každom riadku.

312. O nahrádzaní

V jednorozmernom poli P sa ukladajú skupiny celých čísel takýmto spôsobom:

- každá skupina čísel začína záporným číslom (ďalej toto číslo nazývame hlavička skupiny), ostatné čísla v skupine sú kladné.
- celé pole obsahuje niekoľko skupín (rôzne dlhých), ktoré sú uložené tesne za sebou tak, že sú utriedené podľa hodnôt hlavičky zostupne.
- zvyšok poľa za koncom poslednej skupiny je doplnený nulami.

Príklad takéhoto poľa:

$$-10,\ 2,\ 3,\ 5,\ -20,\ 4,\ 6,\ 7,\ 8,\ -25,\ 8,\ 6,\ 5,\ 4,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0$$

Úloha: Napíšte program, ktorý začne s prázdnym poľom P (t.j. obsahujúcim samé nuly), postupne číta skupiny čísel (v ľubovoľnom poradí ich hlavičiek) a začleňuje ich do poľa podľa nasledujúcich pravidiel:

- ak má vstupná skupina len hlavičku (t.j. pozostáva len zo záporného čísla), tak:
  - ak má hlavička hodnotu -9999, tak program končí činnosť,
  - inak sa z poľa vyhodí skupina s rovnakou hlavičkou a zvyšok poľa sa príslušne posunie. Ak v poli neexistuje skupina s rovnakou hlavičkou, tak sa neurobí nič.
- ak vstupná skupina pozostáva z viac prvkov než len hlavičky, tak:
  - ak sa už v poli nachádza skupina s rovnakou hlavičkou, tak ju nová skupina nahradí, zvyšok poľa za touto skupinou sa príslušne posunie,
  - ak sa v poli nenachádza skupina s rovnakou hlavičkou, tak sa nová skupina vsunie na miesto, ktoré jej prislúcha podľa hodnoty hlavičky, zvyšok poľa sa zase príslušne posunie,
  - ak by sa pri pridávaní do poľa mal prekročiť jeho koniec, tak sa vypíše chyba a posledne prečítaná skupina sa ignoruje (t.j. pole musí byť v takom stave, v akom bolo pred pokusom o začlenenie do ďalšej skupiny) a pokračuje sa v práci programu.

Program vypisuje na výstup každú vstupnú skupinu a stav poľa P po jej spracovaní. Tvar, v akom vstupujú skupiny do programu, nepredpisujeme. Každý si ho môže zvoliť tak, aby vyhovoval použitému programovaciemu jazyku.

Ten, kto pracuje s BASICom, iste spoznal v zložito popísaných pravidlách spôsob, akým pracuje interpreter BASICu pri zápise a opravovaní programu.

Príklad výstupu pre pole P dĺžky 20.

```
Vstup: -10,2,3,4,5
pole : -10,2,3,4,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
Vstup: -20,4,5,6
pole : -10,2,3,4,5,-20,4,5,6,0,0,0,0,0,0,0,0,0,0,0
Vstup: -15,6,7
pole : -10,2,3,4,5,-15,6,7,-20,4,5,6,0,0,0,0,0,0,0,0
Vstup: -15,8,8,8
pole : -10,2,3,4,5,-15,8,8,8,-20,4,5,6,0,0,0,0,0,0,0
Vstup: -10
pole : -15,8,8,8,-20,4,5,6,0,0,0,0,0,0,0,0,0,0,0,0
Vstup: -5,1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,1,2,3,4
Skupina sa nezmestila
pole : -15,8,8,8,-20,4,5,6,0,0,0,0,0,0,0,0,0,0,0,0
Vstup: -9999
Koniec práce programu
pole : -15,8,8,8,-20,4,5,6,0,0,0,0,0,0,0,0,0,0,0,0
```

Ak si zvolíte dlhšie pole P, nemusíte vypisovať v medzivýsledkoch nuly na jeho konci.

313. O nerozhodných voličoch

Predstavte si takéto hlasovanie: Hlasujúci sedia v sále, kde sú stoličky usporiadané vo štvorci $n \times n$. Majú hlasovať o niečom, ale ani jeden z nich nie je pevne rozhodnutý, za ktorú z m možností bude hlasovať. Preto sa spýta niektorého zo svojich ôsmich susedov, akého je názoru a stotožní sa s ním. Keďže sú všetci hlasujúci nerozhodní, každý z nich sa takto „pýta susedov“. V každom okamihu má každý z hlasujúcich jeden z názorov 1 až m.

Úloha: Je daný štvorec $n \times n$ políčok, v každom políčku je napísané číslo 0 až m.

– Susedmi políčka so súradnicami (i,j), $1 \le i,j \le n$, nazývame políčka so súradnicami $(i-1,j-1),(i-1,j),(i-1,j+1),(i,j-1),(i,j+1),(i+1,j-1),(i+1,j),(i+1,j+1)$. Pritom operácie „+1“ a „−1“ v popise susedov treba chápať tak, že ak je ich výsledok mimo intervalu $\langle 1,n\rangle$, tak sa do tohto tvaru upraví vhodným pripočítaním alebo odpočítaním čísla n. Teda krajné políčka majú susedov na „druhom kraji“, napríklad pre $n=10$ má políčko $(1,2)$ susedov $(10,1)$, $(10,2)$, $(10,3)$, $(1,1)$, $(1,3)$, $(2,1)$, $(2,2)$, $(2,3)$, lebo $1-1=0$ upravíme do intervalu $\langle 1,10\rangle$ pripočítaním čísla 10.
– v čase $t=0$ nadobudnú všetky políčka náhodné hodnoty z rozsahu 1 až m.
– v čase $t+1$ nadobudne náhodne vybrané políčko takú hodnotu, akú mal jeho niektorý náhodne vybraný sused v čase t.

Napíšte program, ktorý pre danú veľkosť n, m a počet krokov simulácie k simuluje vyššie popísanú situáciu a vypíše na obrazovku stav poľa v čase $0,1,\ldots,k$.

Simulácia je zaujímavá pre veľké k a n väčšie než 10. Pokúste sa sformulovať hypotézu o chovaní sa voličov.

314. O vážení

Máme daných n závaží s hmotnosťami $1,3,9,\ldots,3^{n-1}$ jednotiek (napríklad kilogramov), ďalej máme k dispozícii dvojramenné váhy (kedysi sa volali mincier) a predmet, ktorý máme odvážiť. Pritom máme zaručené, že jeho hmotnosť je celé číslo od 1 do $(3^n-1)/2$.

Úloha: Napíšte program, ktorý pre ľubovoľnú hmotnosť h predmetu (z daného intervalu) zistí, ako treba uložiť na misky váh závažia a vážený predmet tak, aby vznikla rovnováha.

Majme napríklad štyri závažia hmotností 1, 3, 9 a 27 jednotiek. Vašou úlohou je zistiť, ako tieto závažia rozložiť na misky váh tak, aby sme odvážili predmet s hmotnosťou 5 jednotiek. Zrejme jediným riešením je na misku s predmetom dať závažia 1 a 3 a na druhú misku dať závažie 9.

Zvoľte výstup z vášho programu podobne ako v uvedenom príklade. Vstupmi do programu budú dve čísla: n – počet závaží, $n \le 10$ a h – hmotnosť predmetu, $0 \le h \le (3^n-1)/2$.

Úloha má pre dané n a h (z uvedeného intervalu) vždy jediné riešenie.

315. O ekvivalenciách

Zamyslime sa nad takouto úlohou: vieme, že Janka má toľko súrodencov, koľko má Jožo, Fero má toľko súrodencov, koľko Katka, Jožo má toľko súrodencov, koľko Zdeno a Fero má toľko súrodencov, koľko Jožo. Nás zaujíma, či na základe týchto údajov vieme povedať, či aj Janka a Katka majú rovnaký počet súrodencov.

Keď sa trochu zamyslíte nad uvedenou úlohou (a nakreslíte si situáciu), ľahko zistíte, že všetci majú rovnaký počet súrodencov.

Nech R je *relácia ekvivalencie* (t.j. reflexívna, tranzitívna a symetrická binárna relácia). Máme daných n dvojíc $(x_1,y_1),\ldots,(x_n,y_n)$, ktoré sú prvkami R. Okrem toho máme danú dvojicu (x,y). Pýtame sa, či na základe predošlých údajov môžeme odvodiť, že aj (x,y) je prvkom relácie R.

Úloha: Napíšte program, ktorý pre dané n, $(x_1,y_1),\ldots,(x_n,y_n)$ zistí, či daná dvojica (x,y) patrí do relácie ekvivalencie, ktorá je daná dvojicami $(x_1,y_1),\ldots,(x_n,y_n)$.

Prvky x, y (množina na ktorej je definovaná uvažovaná relácia R) môžu byť reťazce alebo prirodzené čísla v rozsahu počítača.

Príklad: Pre dané $n=7$, dvojice $(1,2)$, $(5,4)$, $(6,4)$, $(6,6)$, $(7,8)$, $(2,3)$, $(8,4)$ sa pýtame
– je dvojica $(8,5)$ prvkom relácie? Odpoveď je áno, lebo:
  – ak je $(5,4)$ prvkom R, tak aj $(4,5)$ je prvkom R (vlastnosť symetrie),
  – ak $(8,4)$ je prvkom R a aj $(4,5)$ je prvkom R, tak aj $(8,5)$ je prvkom R (vlastnosť tranzitívnosti).
– je dvojica $(8,8)$ prvkom relácie? Odpoveď je áno, lebo R je reflexívna (t.j. obsahuje všetky dvojice (x,x)).

321. O Pytagorejských trojuholníkoch

Usporiadaná trojica prirodzených čísel (a, b, c) sa nazýva *Pytagorejská*[3], ak $a^2+b^2 = c^2$, kde $a \leq b$; teda strana dĺžky c je prepona, a je kratšia, b dlhšia odvesna pravouhlého trojuholníka.

Úloha: Napíšte program, ktorý vypíše prvých n Pytagorejských trojíc usporiadaných vzostupne podľa prepony a pre rovnaké prepony, podľa kratšej odvesny.

322. O $*$-postupnostiach

Postupnosť $a_1a_2 \ldots a_n$ zloženú z núl a jednotiek (t.j. každé a_i je nula alebo jednotka) nazveme *01-postupnosťou*. Postupnosť $b_1b_2 \ldots b_m$ zloženú z núl, jednotiek a hviezdičiek nazveme *$*$-postupnosťou*. 01-postupnosť $a_1 \ldots a_n$ je *odvodená* z $*$-postupnosti $b_1 \ldots b_m$, ak $m = n$ a pre všetky $i = 1, \ldots, n$ platí: ak $a_i \neq b_i$ tak $b_i = *$.

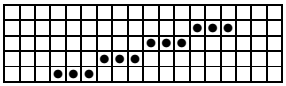
Napríklad 01-postupnosť 110101 je odvodená z $*$-postupnosti $*101*1$, ale nie z postupnosti $11*10$, lebo má inú dĺžku, ani z postupnosti $***100$, lebo na poslednom mieste sa líšia, pričom v druhej postupnosti nie je posledná hviezdička.

Úloha: Napíšte program, ktorý pre danú $*$-postupnosť dĺžky n, $1 \leq n \leq 10$ vypíše všetky 01-postupnosti z nej odvodené.

323. O interpolátore

Každý počítač, ktorý má bodovú grafiku, musí vedieť vykresľovať na svojej obrazovke úsečky. Úlohu vykresliť úsečku v bodovom rastri má väčšinou podprogram, ktorý sa nazýva interpolátor. Vašou úlohou bude navrhnúť algoritmus interpolátora. Znakové pole D rozmeru $m \times n$ predstavuje grafickú bodovú obrazovku. Každý jej prvok obsahuje informáciu o jase jedného bodu obrazovky. Znak `X` znamená rozsvietený bod a `.` (bodka) znamená zhasnutý bod.

Úloha: Napíšte program, ktorý začne pracovať s prázdnou obrazovkou (t.j. v poli D sú samé bodky) a pre dané dva body $[x_1, y_1]$ a $[x_2, y_2]$, kde $1 \leq x_1, x_2 \leq m$, $1 \leq y_1, y_2 \leq n$, vykreslí v poli D úsečku z bodu so súradnicami $[x_1, y_1]$ do bodu so súradnicami $[x_2, y_2]$ a vypíše obsah poľa D.



Pod vykreslením rozumieme to, že niektorým prvkom matice priradíme hodnotu `X` tak, aby po vypísaní poľa na papier vznikol dojem úsečky, ktorá spája dva dané body. Vypisovanie znakov `X` a `.` sa dá nahradiť priamo zmenou jasu príslušných bodov (pixelov) na obrazovke, ktorá bude predstavovať pole bodov $m \times n$.

V reálnych mikropočítačoch veľmi záleží na rýchlosti interpolátora, aby bol použiteľný aj na vytváranie pohybového dojmu (animácie), preto sa dobre zamyslite nad počtom operácií, ktoré je nevyhnutné vykonať na vykreslenie čiary.

324. O šachovom koňovi.

Máme danú šachovnicu rozmeru $m \times n$. Na nej je daná počiatočná pozícia šachového koňa $[x, y]$, $(1 \leq x \leq m,\ 1 \leq y \leq n)$. Každému políčku šachovnice so súradnicami $[i, j]$ priradíme číslo $T(i, j)$, ktoré určuje minimálny počet ťahov, na ktoré sa môže dostať šachový kôň z políčka $[x, y]$ na políčko $[i, j]$. Ak prechod medzi týmito dvoma políčkami nie je možný, tak bude príslušná hodnota rovná -1.

Úloha: Napíšte program, ktorý pre dané m, n, x a y vypíše všetky hodnoty $T(i, j)$ usporiadané do obdĺžnika $m \times n$.

325. O Hammingovej postupnosti

Hammingova[4] *postupnosť* je vzostupne usporiadaná postupnosť čísel, ktoré nie sú deliteľné inými prvočíslami ako 2, 3 a 5. Postupnosť začína takto: $1, 2, 3, 4, 5, 6, 8, 9, 10, 12, \ldots$

---

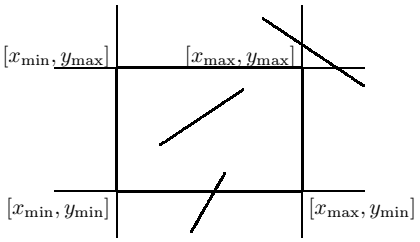
[3] Pytagoras (asi 580–500 p.n.l.), grécky matematik a filozof

[4] Richard Wesley Hamming (1915–1998), americký matematik

Úloha: Napíšte program, ktorý pre dané n vypíše prvých n členov Hammingovej postupnosti.

331. O grafickom okienku

V rovine nech je dané obdĺžnikové okno (obrázok), ktoré zobrazuje dvojrozmerný interval $\langle x_{\min}, x_{\max}\rangle \times \langle y_{\min}, y_{\max}\rangle$, teda obdĺžnik s vrcholmi $[x_{\min}, y_{\min}]$, $[x_{\min}, y_{\max}]$, $[x_{\max}, y_{\min}]$, $[x_{\max}, y_{\max}]$. Toto okno nám predstavuje zobrazovaciu plochu (tzv. vykresľovací rozsah) nejakého grafického zapisovača. My chceme vykresliť úsečku s koncovými bodmi $[x_1, y_1]$, $[x_2, y_2]$, ktorá nemusí celá ležať v zobrazovacom obdĺžniku. Môžu nastať tieto prípady:



- celá úsečka leží v obdĺžniku, potom ju jednoducho vykreslíme;
- z úsečky len „nejaký kúsok“ leží v obdĺžniku, potom vykreslíme len tento kúsok; „nejaký kúsok“ môže byť úsečka alebo bod;
- žiadna časť úsečky neleží v obdĺžniku, potom nekreslíme nič.

Úloha: Napíšte program, ktorý ako vstupné údaje dostane súradnice zobrazovacej plochy $x_{\min}$, $x_{\max}$, $y_{\min}$ a $y_{\max}$. Potom prečíta súradnice koncových bodov úsečky x_1, y_1 a x_2, y_2 a pre každú vypíše súradnice začiatku a konca skutočne vykreslenej časti úsečky. Jeden bod chápeme ako úsečku s rovnakými koncovými bodmi. Medzi výsledkami pošlite príklady na všetky možné varianty polohy úsečky k obdĺžnikovému oknu.

332. O obyvateľoch ostrova

Ostrov AXA obýva n domorodcov. Od neho na 5 dní plavby na kanoe sa nachádza opustený ostrov YBY. Oba ostrovy sú bohaté na rôzne druhy ovocia, takže sú lákadlom pre domorodcov. Hneď po objavení ostrova YBY sa niekoľkí domorodci rozhodli presťahovať. Miestny matematik vypočítal, že s pravdepodobnosťou p, $0 < p < 1$, každý domorodec opúšťa buď ostrov AXA, alebo YBY (t.j. domorodec opúšťa aj ostrov YBY s pravdepodobnosťou p). Domorodcom je známe, že po d dňoch začína obdobie dažďov a teda po týchto d dňoch už žiaden nie je na ceste (plavbe).

Úloha: Napíšte program, ktorý modeluje sťahovanie domorodcov pre vstupné hodnoty n, p a d a vypíše počty obyvateľov na ostrovoch po d dňoch (po $(d-5)$-om dni už žiaden nezačne plavbu). Uvažujte, že domorodci sa rozhodnú, či majú vyplávať, vždy ráno. Skúste pre $n = 100$, $p = 0.1$ a $d = 200$.

333. O deliteľoch

Napíšte algoritmus, ktorý pre dané prirodzené číslo n z intervalu $\langle 1, 1000\rangle$ nájde a vypíše najmenšie také prirodzené číslo, ktoré je deliteľné všetkými prirodzenými číslami menšími alebo rovnými ako n. Skúste pre $n = 10$, 100 a 200.

Už pre $n = 20$ dostávame také veľké číslo, ktoré aritmetika počítača zaokrúhli. Preto vymyslite taký spôsob výpočtu hľadaného čísla, ktorý umožní určiť všetky jeho cifry aj v prípade, že je väčšie než dovoľuje aritmetika počítača.

334. O zlomkoch (Fareyov[5] rad)

Napíšte program, ktorý pre dané prirodzené číslo n vypíše rastúcu postupnosť všetkých zlomkov v základnom tvare (čitateľ a menovateľ sú nesúdeliteľné) z intervalu $\langle 0, 1\rangle$, ktorých menovateľ nie je väčší ako n.

[5] John Farey (1766–1826), anglický geológ. V roku 1816 publikoval niektoré vlastnosti uvedeného radu.

Príklad: Pre $n = 7$ je postupnosť zlomkov:

$$\frac{1}{7}, \frac{1}{6}, \frac{1}{5}, \frac{1}{4}, \frac{2}{7}, \frac{1}{3}, \frac{2}{5}, \frac{3}{7}, \frac{1}{2}, \frac{4}{7}, \frac{3}{5}, \frac{2}{3}, \frac{5}{7}, \frac{3}{4}, \frac{4}{5}, \frac{5}{6}, \frac{6}{7}, \frac{1}{1}.$$

335. O aritmetických výrazoch

Bezzátvorkovým výrazom budeme nazývať taký aritmetický výraz, ktorý obsahuje len jednociferné čísla a znamienka '+' alebo '−'. *Ozátvorkovaním* bezzátvorkového výrazu nazveme niektoré rozmiestnenie párov zátvoriek '(' a ')' tak, aby vznikol dobre uzátvorkovaný výraz. *Vyhodnotenie výrazu* je číslo, ktoré vznikne aplikovaním operátorov '+' a '−' na operandy podľa bežných zvyklostí s prihliadnutím na zátvorky.

Bezzátvorkový výraz 1+2−3+4−5 má napríklad takéto ozátvorkovania: 1+2−(3+4)−5, 1 + (2 − (3 + 4 − 5)). Vyhodnotením prvého výrazu je −9, vyhodnotením druhého je 1.

Úloha: Napíšte program, ktorý po prečítaní bezzátvorkového výrazu v tvare znakového reťazca vypíše všetky možné vyhodnotenia, ktoré môžeme dosiahnuť rôznym ozátvorkovaním daného výrazu. (Jedno vyhodnotenie sa môže vo výsledkoch vyskytnúť aj viackrát, žiadne však nesmie chýbať.) Ku každému vyhodnoteniu program vypíše aj ozátvorkovaný výraz, ktorého je to hodnota. Bezzátvorkový výraz $1-2+1$ má ozátvorkovania: $1-2+1$, $(1-2)+1$, $(1-2+1)$, $1-(2+1)$, $(1-(2+1))$, $((1-2)+1)$; ich vyhodnotenia sú 0, 0, 0, −2, −2, 0. Program teda môže vypísať len čísla 0, −2, ale môže vypísať aj napríklad `0, 0, 0, 0, -2, -2, 0, 0, 0` (a k nim zodpovedajúce výrazy).

411. Číslo

Napíšte program, ktorý efektívne nájde všetky prirodzené čísla c také, že najvyššia cifra čísla c je počet cifier 0 v čísle c, druhá najvyššia cifra čísla c je počet cifier 1 v čísle c, tretia najvyššia cifra čísla c je počet cifier 2 v čísle c, atď. až desiata najvyššia cifra čísla c je počet cifier 9 v čísle c. Najvyššia cifra je cifra pri najvyššej mocnine 10.

Príklad: $c = 1210$.

412. Kódovanie kombinácií

Nech n, k sú prirodzené čísla. Uvažujeme lexikograficky zoradené všetky k-prvkové podmnožiny množiny $\{1, 2, \ldots, n\}$. Ich počet je $\binom{n}{k}$. Napíšte algoritmus a program, ktorý každej k-prvkovej podmnožine množiny $\{1, 2, \ldots, n\}$ z uvedeného lexikografického poradia priradí „poradové číslo“, t.j. číslo z množiny $\{1, 2, \ldots, \binom{n}{k}\}$. Dôraz kladťe na efektívnosť algoritmu!

413. Konvexný obal

V rovine je daných n bodov popísaných súradnicami $[x_i, y_i]$, $1 \leq i \leq n$. Napíšte program, ktorý vypíše všetky body, ktoré tvoria vrcholy konvexného obalu bodov $[x_i, y_i]$, $1 \leq i \leq n$. *Konvexný obal* množiny A je najmenšia konvexná množina obsahujúca A.

414. Josifova úloha

Počas židovskej vojny, keď Rimania obsadili Jotopatu, podarilo sa Josifovi spolu so štyridsiatimi vojakmi ujsť a schovať sa v jaskyni. Josif zistil, že všetci vojaci – okrem neho a ešte jedného človeka – radšej spáchajú samovraždu, ako by padli do zajatia. Bál sa vystúpiť proti takému riešeniu, preto navrhol vykonať samovraždy organizovane: Všetci sa postavia do kruhu a počnúc od niektorého sa každý tretí zavraždí až po posledného. Potom Josif akoby náhodou postavil seba a toho druhého na 31. a 16. miesto od začiatku. Toľko hovorí história. A teraz formálnejšie.

Úloha: V kruhu je n ľudí. Ľudia sú na začiatku očíslovaní postupne od 1 po n. Počnúc od prvého ideme jedným smerom a z kruhu vylučujeme každého m-tého (ďalej sa nepočíta) až pokým nevylúčime posledného.

Napíšte program, ktorý pre dané n a m zistí o každom človeku, kedy-koľký vypadol z kruhu. Kruh tvorí aj jeden, alebo dvaja ľudia.

415. Úloha na vyhľadávanie

Dané je pole A, ktoré obsahuje n čísel usporiadaných vzostupne a číslo x, pre ktoré platí $A[1] < x < A[n]$.

Úloha: Napíšte program, ktorý nájde index i v poli A taký, že $A[i-1] < x \leq A[i]$.

Program nesmie používať žiadne pomocné pole a mal by minimalizovať počet prezretých prvkov poľa A. (Napríklad ak $A[n-1] < x$, nemal by prezerať všetkých $n-1$ prvkov.)

421. O mnohouholníku

Napíšte program, ktorý určí, či daná uzavretá lomená čiara tvorí konvexný n-uholník. Vstupom do programu je číslo n a n dvojíc reálnych čísel $[x_i, y_i]$, kde $1 \leq i \leq n$, ktoré predstavujú po sebe idúce vrcholy lomenej čiary.

422. O Buffonovej[6] ihle

Na „nekonečnej“ rovnej ploche (rovine) je nakreslený nekonečný systém rovnobežných priamok, pričom vzdialenosť susedných priamok je d. Na túto rovinu hádžeme ihlu dĺžky d. Zisťujeme pravdepodobnosť (presnejšie relatívnu početnosť) javu, že dopadnutá ihla pretína nejakú priamku (alebo na nej leží) z uvedeného systému. Napíšte program, ktorý bude simulovať popísaný pokus. Porozmýšľajte, akým spôsobom súvisí pravdepodobnosť daného javu s číslom $\pi = 3.14159265\ldots$.

Relatívna početnosť nejakého javu pri simulácii je pomer počtu úspešných výsledkov (t.j. kedy jav nastal) k počtu všetkých výsledkov (t.j. k počtu pokusov).

423. O šachovnici

Je dané číslo n. Napíšte algoritmus a program, ktorý zafarbí šachovnicu $n \times n$ pomocou n navzájom rôznych farieb tak, aby v každom riadku a stĺpci bolo všetkých n farieb. Program vypíše všetky možné zafarbenia šachovnice.

424. O reálnom čísle

Napíšte program, ktorý zo vstupu načíta znakovú reprezentáciu reálneho čísla a vypíše hodnotu tohto čísla. Predpokladajte, že číslo bude z rozsahu reálnych čísel vo vami použitom programovacom jazyku. Znaková reprezentácia reálneho čísla má vo všeobecnosti tento tvar (je podobná ako v Basicu alebo Pascale):

- môže začínať unárnym `+` alebo `-`;
- časť pred desatinnou bodkou musí obsahovať aspoň jednu cifru;
- ak číslo obsahuje desatinnú bodku, desatinná časť musí obsahovať aspoň jednu cifru;
- reálne číslo môže byť napísané v semilogaritmickom tvare, t.j. za číslom nasleduje znak `E` a maximálne dvojciferné celé číslo (môže mať znamienko `+` alebo `-`), ktoré označuje exponent čísla 10, ktorým treba vynásobiť hodnotu pred znakom `E`.

Príklad reálnych čísel: `0.23, +57.9E3, -5E-5, 6.3E2`.
Zápis `6.3E2` predstavuje číslo $6.3 \times 10^2 = 630$.

425. O výmenách

Program dostáva na vstupe prirodzené čísla n, r také, že $r \mid n$ a pole A s indexmi $0, \ldots, n-1$, ktorého prvkami sú čísla 0 až $n-r$ s týmito vlastnosťami:

(i) čísel 0 je v poli A presne r;
(ii) každé z čísel 1 až $n-r$ sa nachádza v poli práve raz a to tak, že pre každé $0 \leq i < j \leq n-1$ platí: ak $A[i] \neq 0$ a $A[j] \neq 0$, tak $A[i] < A[j]$ (t.j. tak, že ak by sme z poľa odstránili nulové prvky, pole A by bolo vzostupne usporiadané).

[6] Georges-Louis Leclerc, Comte de Buffon (1707–1788), francúzsky prírodovedec

Úloha: Napíšte algoritmus, ktorý preusporiada pole A tak, že nulové prvky majú indexy násobkov n/r (čiže $0, n/r, (2n)/r, \dots$) a pritom platí podmienka (ii) pre pole A. Pri preusporiadavaní poľa A môžete použiť len vzájomnú výmenu dvoch prvkov. Váš program bude vypisovať dvojice prvkov, nad ktorými sa vykonáva operácia výmeny.

431. O počte výskytov

Je dané prirodzené číslo n a n-prvkové pole celých čísel A. Napíšte program, ktorý nájde v poli A taký prvok, ktorý sa v ňom vyskytuje najviackrát (ak je takých prvkov viac, tak niektorý z nich; tzv. *modus*).

Príklad: Pre $n = 5$ sa v poli $A = (1, 2, 3, 2, 3)$ najviackrát vyskytuje číslo 2 (alebo 3).

432. O pokazenom obracači

Definujme si špeciálne triediace zariadenie, ktoré nazveme *obracač*. Obracač obsahuje n pozícií umiestnených v jednom rade. Na každej pozícii sa nachádza práve jedna kartička, na ktorej je napísané prirodzené číslo menšie alebo rovné n. Pritom každé z čísel 1 až n sa nachádza práve na jednej kartičke. Úlohou obracača je kartičky utriediť vzostupne podľa čísel na nich napísaných. Obracač môže triediť len pomocou operácií $obrat(i, j)$, kde $i \leq j$. Táto operácia realizuje zrkadlové otočenie poradia kartičiek od i-tej po j-tu. Teda ak si i-tu kartičku označíme k_i a znakom $\leftrightarrow$ označíme výmenu dvoch kartičiek, tak môžeme operáciu $obrat(i, j)$ rozpísať takto:

$$\begin{aligned} k_i &\leftrightarrow k_j \\ k_{i+1} &\leftrightarrow k_{j-1} \\ k_{i+2} &\leftrightarrow k_{j-2} \\ &\vdots \\ k_{i+((j-i-1) \text{ div } 2)} &\leftrightarrow k_{j-((j-i-1) \text{ div } 2)} \end{aligned}$$

Predpokladajme, že máme k dispozícii obracač s malou chybičkou – má „zaseknutú“ k-tu pozíciu. To znamená, že dokáže vykonať len také operácie *obrat*, ktoré nepohnú, či nemusia pohnúť kartičkou na k-tej pozícii. Zdá sa však, že aj takýto obracač dokáže utriediť kartičky, pravda, len za predpokladu, že k-ta kartička je už od začiatku na svojom mieste.

Úloha: Napíšte program, ktorý najprv prečíta prirodzené čísla n a $k \leq n$ a n-prvkové pole A, ktoré obsahuje každé z prirodzených čísel od 1 do n práve raz a zároveň platí $A[k] = k$. Čísla n a k definujú n-prvkový obracač so zaseknutou k-tou pozíciou a pole A určuje počiatočné rozmiestnenie kartičiek v obracači.

Výsledkom práce vášho programu nech je postupnosť takých operácií *obrat*, aby po ich realizácii obracačom (so zaseknutou k-tou pozíciou) boli kartičky vzostupne utriedené. (Vašou úlohou je teda napísať program, ktorý riadi obracač). Ak taká postupnosť neexistuje, program o tom podá správu.

Predpokladáme, že obracač pracuje oveľa pomalšie ako počítač, ktorý realizuje váš program, preto je treba minimalizovať v prvom rade počet operácií *obrat*, ktoré váš program predpíše obracaču.

Pre $n = 9$, $k = 4$ a pole $A = (2, 3, 8, 4, 6, 5, 1, 7, 9)$ je správny (nie však najlepší) výsledok činnosti programu napríklad:

```
obrat(6,7); obrat(5,6); obrat(3,5); obrat(1,3); obrat(2,3);
obrat(5,6); obrat(6,7); obrat(7,8); obrat(5,6).
```

433. O parketách

Na štvorčekovom papieri máme nakreslený pôdorys miestnosti. Predpokladajme, že miestnosť sa skladá z nejakého počtu celých štvorčekov, z nich každý sa dotýka ďalších

štvorčekov izby aspoň jednou stranou. Túto miestnosť máme „vyparketovať“ obdĺžnikmi veľkosti 1×2 štvorčeky.

Napíšte program, ktorý prečíta zo vstupu popis miestnosti a vypíše jej pokrytie obdĺžnikmi 1×2, alebo správu o tom, že pokrytie nie je možné.

Tvar miestnosti odporúčame kódovať tak, že ju ohraničíme nejakým obdĺžnikom, zadáme veľkosť obdĺžnika (počet riadkov a stĺpcov) a dvojrozmerné pole zložené z núl a jednotiek, kde jednotka predstavuje štvorček patriaci do miestnosti.

Výstup požadujeme v tvare matice čísel, kde nula označuje štvorček, ktorý nepatrí do miestnosti, štvorčeky patriace jednému obdĺžniku majú to isté číslo a susedné štvorčeky patriace rôznym obdĺžnikom majú navzájom rôzne čísla.

Príklad:

Vstup:	Výstup:
`rozmery miestnosti:` 4×5	`02200`
`miestnosť:`	`33440`
`01100`	`55611`
`11110`	`22600`
`11111`	
`11100`	

434. O symetrických číslach

Napíšte program, ktorý pre dané k vypíše všetky nepárne čísla $< 2^k$, ktorých zápis v dvojkovej sústave je symetrický podľa stredu. Dvojkový zápis čísla 93 $(1011101)_2$ je symetrický podľa stredu, zápis čísla 67 $(1000011)_2$ nie je symetrický. Pre $k = 4$ program vypíše `1, 3, 5, 7, 9, 15.` Program sa snažte urobiť čo najrýchlejší.

435. O výpise reálneho čísla

Pre dané reálne číslo r a prirodzené čísla n a m, kde $m \leq n - 2$ definujeme zápis reálneho čísla r v pevnom formáte dĺžky n o m desatinných miestach (skrátene $Fn.m$-zápis čísla r) nasledovne.

Nech r je číslo. $Fn.m$-zápis čísla r
- má práve m cifier za desatinnou bodkou,
- posledná cifra je správne zaokrúhlená,
- ak je číslo záporné, tak je pred ním znamienko `-`,
- počet všetkých znakov zápisu je n, pričom prípadné medzery sú doplnené zľava.

Napríklad:
- $F10.2$-zápis čísla 1236.758 je ␣␣␣`1236.76`
- $F7.2$-zápis čísla -12.1 je ␣`-12.10`
- $F5.2$-zápis čísla 125.1 neexistuje.

Úloha: Napíšte program, ktorý pre dané r, n a m (pritom r program prečíta do reálnej číselnej premennej) vytvorí $Fn.m$-zápis čísla r do reťazcovej premennej z a výstupnú premennú z vypíše. Ak pre dané n a m neexistuje $Fn.m$-zápis čísla r, tak program uloží do z (a vypíše) n hviezdičiek.

V programe nie je povolené používať žiadnu štandardnú funkciu, ktorá prevádza číslo do jeho znakového zobrazenia. Pravdaže, je dovolené používať konverzné funkcie medzi znakmi a ich kódmi.

511. Zarovnávanie textu.

Súčasťou systémov na spracovanie textu je vždy program, nazývaný formátovač, ktorého úlohou je zarovnať odseky textu tak, aby mali rovnaký pravý okraj. Program číta vstupné riadky rôznej dĺžky a skráti ich, alebo predĺži čo najbližšie k určenej šírke riadku. Potom vsunie medzi niektoré slová medzery tak, aby mal riadok presnú dĺžku.

Úloha: Napíšte program, ktorý najprv prečíta čísla l, p a z. Potom prečíta text pozostávajúci z ľubovoľného počtu riadkov ľubovoľnej (zhora ohraničenej) dĺžky. Koniec každého odseku je v texte označený znakom `@`, každý odsek začína na novom riadku.

Program vypíše ten istý text upravený do riadkov, ktoré začínajú na l-tej a končia na p-tej pozícii v riadku. Prvý riadok každého odseku je odsunutý na $(l + z)$-tú pozíciu. Všetky riadky sú doplnené medzerami tak, aby mali rovnaký pravý okraj.

```
     Príklad:  Tento  text  je zarovnaný pomocou formátovača
pre  p=60,  l=1  a  z=5, pre lepšiu predstavu uvádzame tento
odsek v takom tvare, v akom ho prečítal náš program:

Príklad: Tento text je zarovnaný
pomocou formátovača
pre p=60, l=1 a z=5, pre lepšiu predstavu
uvádzame tento odsek v takom tvare, v akom
ho prečítal náš program:@
```

512. O škrtaní

Nech i je kladné číslo a P je nekonečná postupnosť prirodzených čísel. Definujme operáciu *Škrtni*(i, P), ktorá upraví postupnosť P nasledovne: prvých i členov ponechá, ďalších i členov vyškrtne, potom zase i členov ponechá, i vyškrtne atď. Toto pokračuje nekonečne dlho. Výsledkom operácie je takto vyškrtaná nekonečná postupnosť.

Postupnosť prirodzených čísel Q je definovaná nasledovne:

- Q_0 je postupnosť prirodzených čísel: $1, 2, 3, \ldots$.
- Q_1 je výsledkom operácie *Škrtni*$(1, Q_0)$;
- Q_2 je výsledkom operácie *Škrtni*$(2, Q_1)$,
- všeobecne, Q_i je výsledkom operácie *Škrtni*(i, Q_{i-1}), ...

Tento proces sa opakuje donekonečna. Výsledná postupnosť nech je Q. Hoci je takto popísaný proces nekonečný (teda nekončí v i-tom kroku), vieme napriek tomu v konečnom čase určiť prvých k členov postupnosti Q pre ľubovoľné k.

Úloha: Napíšte program, ktorý pre dané k vypíše prvých k členov postupnosti Q.

Výsledok pre $k = 5$ je 1, 3, 9, 25, 57. Program považujeme za veľmi dobrý, ak dokáže vypísať v rozumnom čase (menej ako 8 hodín) prvých 15 členov postupnosti.

513. Turnaj rytierov

V krajine Stochastika sa konajú rytierske turnaje takýmto spôsobom: n rytierov sa zoradí. Začína najslabší rytier a pokúsi sa zhodiť zo sedla druhého rytiera. Ak sa mu to nepodarí, tak pokračuje druhý rytier a pokúsi sa zhodiť zo sedla tretieho. Ak sa prvému podarilo druhého zhodiť, tak druhý vypadáva z turnaja a pokračuje tretí rytier.

Takto turnaj pokračuje, pričom každý rytier, ktorý ešte nevypadol z turnaja, sa pokúša zhodiť zo sedla nasledujúceho rytiera. Nasledujúci rytier k n-tému rytierovi je prvý rytier. Turnaj končí, keď zostane len jeden rytier, ktorého princezná (dosiaľ slobodná) vyhlási za víťaza.

Pravdepodobnosť, že prvý rytier zhodí zo sedla ľubovoľného protivníka je p_1, pravdepodobnosť, že druhý rytier zhodí protivníka je p_2 atď. Pritom platí $p_1 \leq p_2 \leq \ldots \leq p_n$.

Úloha: Napíšte program, ktorý pre dané n a pravdepodobnosti $p_1, \ldots, p_n$ nasimuluje turnaj a určí víťaza. Počas simulácie sa vypisuje protokol podľa nasledujúceho vzoru (pre $n = 4$ môže mať turnaj napríklad takýto priebeh):

```
1.rytier nezhodil
2.rytier nezhodil
3.rytier zhodil 4.rytiera
1.rytier nezhodil
2.rytier zhodil 3.rytiera
```

```
1.rytier nezhodil
2.rytier zhodil 1.rytiera
Zvíťazil 2.rytier.
```

Pozmeňte program tak, aby simuloval 1000-krát turnaj pre $n = 10$ a $p_1 = 0.25$, $p_2 = 0.3$, $p_3 = 0.35, \ldots, p_{10} = 0.7$. Počas simulácie sa, pravdaže, nevypisuje žiadny protokol a na konci sa vypíše tabuľka počtu víťazstiev každého z rytierov.

514. Plocha kruhu

Na štvorčekovom papieri je nakreslená kružnica s celočíselným polomerom r, $1 < r \leq 100$, a so stredom v niektorom mrežovom bode. Dĺžka strany štvorčeka je 1.

Úloha: Napíšte program, ktorý pre daný celočíselný polomer r vypíše počet štvorčekov, ktoré ležia celé vo vnútri kružnice.

515. Súvislé obrazce

Z listu štvorčekového papiera rozmerov $m \times n$ štvorčekov sme vystrihli niektoré štvorčeky. Zaujíma nás, na koľko častí sa papier rozpadne. List papiera je reprezentovaný poľom A, kde sú vystrihnuté štvorčeky označené jednotkami, ostatné štvorčeky sú označené nulami.

Úloha: Napíšte program, ktorý pre dané prirodzené čísla m, n a dané dvojrozmerné pole A vypíše počet častí, na ktoré sa papier rozpadne.

521. O vranách

Napíšte program, ktorý pre dané prirodzené číslo k, $0 < k < 32768$, vypíše frázu `NAD KOMINOM LETELO` k `VRAN.` slovami, podľa gramatiky slovenského jazyka, t.j. za k dosadí slovnú formuláciu a vetu upraví.

Príklad:

Vstup:	Výstup:
$k = 23$	`NAD KOMINOM LETELO DVADSATTRI VRAN.`
$k = 1$	`NAD KOMINOM LETELA JEDNA VRANA.`

522. O smetiach

Napíšte program, ktorý pre danú postupnosť n celých navzájom rôznych čísel vypíše novú postupnosť, ktorá vznikne z pôvodnej vyčiarknutím minimálneho počtu čísel tak, aby nová postupnosť bola rastúca.

Príklad: Pre $n = 7$ a postupnosť $1, 7, 5, 8, 2, 3, 4$ je nová postupnosť $1, 2, 3, 4$

523. O najkratších cestách

Pole A veľkosti $n \times n$ reprezentuje jednosmerné (!!) cesty medzi n mestami nasledujúcim spôsobom: Ak je v poli A prvok

- $A[i, j] = 0$, tak z mesta i nevedie priama jednosmerná cesta do mesta j. (Naopak to nemusí platiť).
- $A[i, j] = b$, kde $b > 0$, tak existuje jednosmerná cesta vedúca priamo z mesta i do mesta j a má dĺžku b. (Naopak to nemusí platiť).

Úloha: Napíšte program, ktorý pre dané n, cesty (pole A) a jedno mesto u určí pre každé mesto v číslo $d(v)$, čo je dĺžka najkratšej cesty z u do v. Ak neexistuje žiadne spojenie medzi mestami u a v, tak $d(v) = \infty$.

Cesta môže viesť len po jednosmerných cestách spájajúcich mestá. Riešte pre $A[1, 2] = 2$; $A[1, 4] = 3$; $A[2, 3] = 7$; $A[2, 4] = 1$; $A[3, 6] = 1$; $A[4, 5] = 2$; $A[5, 1] = 4$; $A[5, 3] = 1$; $A[6, 1] = 1$; $A[6, 2] = 2$; ostatné prvky sú nulové. Za u zvoľte postupne všetky mestá 1 až 6.

524. O mape

Mapa súostrovia je zakódovaná v celočíselnom poli M veľkosti $n \times n$. Nulové prvky predstavujú more. Súvislé oblasti nenulových prvkov predstavujú ostrovy, pričom pre tieto ostrovy platí:

i) dva nenulové prvky patria k jednému ostrovu, ak sa líšia práve v jednej súradnici o jednotku;
ii) ak pre $j < k$, $M[i,j]$ a $M[i,k]$ patria k jednému ostrovu, tak musia byť všetky prvky $M[i,x]$ nenulové pre $j < x < k$, t.j. tiež patria k tomuto ostrovu;
iii) nenulové číslo vyjadruje nadmorskú výšku príslušného miesta ostrova;
iv) okraj mapy je more, t.j. prvky poľa M s indexom 1 alebo n majú hodnotu 0.

Úloha: Napíšte program, ktorý zistí plochu (t.j. počet nenulových prvkov) ostrova s najvyšším kopcom v rámci danej mapy. Ak ich je viac, tak ľubovoľný z nich.

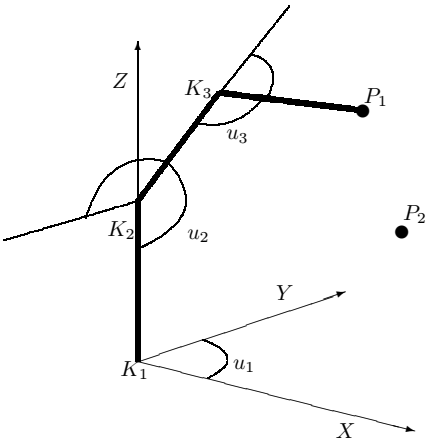
Názorný príklad: (nuly sú nahradené bodkami)

```
. . . . . . . . . . . . . . . . . . . .
. . . 1 2 1 . . . . . . . 1 1 4 2 . . .
. . 3 1 1 2 3 4 . . 5 . . 1 2 4 4 4 . .
. 1 1 1 1 1 1 1 . . 4 . . . 2 3 . . . .
. . 4 5 6 1 2 . . . . . . . . . . . . .
. . . . 2 2 . . . . . . . 2 4 4 5 2 . .
. . . . . . . . . . . . . 2 6 3 6 7 3 1 .
. . . . . . . . . . . . . 4 3 1 1 3 5 2 .
. . . . . . . . . . 2 4 1 . 3 5 8 9 4 5 .
. . . . . . . . . . . 3 3 . . 3 3 7 . . .
. . . . . . . . . . . . . . . . . . . . .
```

Na tejto mape je päť ostrovov a najvyšší (vpravo dole) má plochu 28.

525. O ramene

Koncový bod ramena robota sa má presunúť v trojrozmernom priestore z bodu P_1 do bodu P_2. Rameno robota má tri časti a tri kĺby. Každá časť je dlhá 180 cm (obrázok). Kĺby umožňujú častiam ramena meniť polohu. Kĺb k_1 má súradnice $[0,0,0]$ a môže sa otáčať horizontálne o uhol u_1 z intervalu $\langle 0, 90° \rangle$. Kĺb k_2 má súradnice $[0,0,180]$ a dá sa otáčať vertikálne o uhol u_2 z intervalu $\langle 0, 270° \rangle$. Kĺb k_3 sa dá otáčať vertikálne o uhol u_3 z intervalu $\langle 0, 180° \rangle$. Uhly sa merajú proti smeru chodu hodinových ručičiek. Uvedomte si, že časti ramena robota sú v jednej rovine.



Riadenie ramena robota sa vykonáva príkazmi *otoč*(i,j,k), kde i,j,k sú z množiny $\{-1,0,1\}$. Vykonanie príkazu *otoč* si ukážeme na príklade: *otoč*$(0,-1,1)$ spôsobí, že kĺb

k_1 nezmení svoju polohu, kĺb k_2 sa otočí tak, že zmenší svoj uhol o jeden stupeň a kĺb k_3 zasa zväčší svoj uhol o jeden stupeň. Uhly pritom musia zostať v prípustných hraniciach, inak sa príslušné rameno nepohne.

Úloha: Napíšte program, ktorý na základe súradníc $[x_1, y_1, z_1]$, $[x_2, y_2, z_2]$ bodov P_1 a P_2 (vstupné hodnoty), vypíše postupnosť príkazov *otoč*, ktoré by premiestnili koncový bod ramena z bodu P_1 čo najbližšie k bodu P_2. Skúste pre $P_1 = [100, 0, 0]$, $P_2 = [100, 100, 0]$. Snažte sa, aby program generoval čo najkratšiu postupnosť príkazov *otoč*.

531. O inverzných vranách.

Napíšte program, ktorý prečíta vstupný reťazec znakov predstavujúci prirodzené číslo menšie ako 32 000 zapísané slovami (po slovensky) a vypíše jeho číselnú hodnotu. V prípade, že vstupný reťazec nepredstavuje slovami zapísanú číslovku, vypíše správu o chybe.

Príklad:

Vstup:	Výstup:
dvestotridsaťpäť	235
tisícdvadsaťštyri	1024
päť a tridsať	`chyba`
vrany odleteli	`chyba`

V prípade neistoty použite pravidlá slovenského pravopisu. Vstupný text nemusí obsahovať diakritiku.

532. O podieloch

Dané sú dve kladné prirodzené čísla m, n.

a) Zistite celú časť podielu m/n, základnú periódu, jej dĺžku, predperiódu a jej dĺžku.
b) Riešte problém z časti a) s nasledovným obmedzením – v programe môžete použiť najviac 10 jednoduchých premenných, ktorých hodnoty sú v každom okamihu výpočtu celé čísla (t.j. nemôžete použiť pole ani iné dátové štruktúry).

Príklad: Pre vstup $m = 7$ a $n = 12$ sa vypočíta, že celá časť je 0, predperióda (58) má dĺžku 2 a perióda (3) má dĺžku 1. Pre vstup $m = 125$, $n = 198$ sa vypočíta, že celá časť je 0, predperióda (6) má dĺžku 1 a perióda (31) má dĺžku 2.

533. O reťazcoch

Dané sú dva vstupné reťazce pozostávajúce z cifier $0, 1, \ldots, 9$ a písmen $A, B, \ldots, Z$. Dosaďte za písmená vyskytujúce sa v reťazcoch cifry $0, 1, \ldots, 9$ tak, aby reťazce s dosadenými hodnotami boli zhodné. V prípade viacnásobného výskytu písmena v reťazcoch dosadzujeme rovnakú hodnotu za všetky jeho výskyty. Zistite, koľko rôznych dosadení existuje.

Príklad:

Vstup:	Výstup:
$0XY1$, $0YZZ$	1 riešenie (0111)
$0XXZY23$, $0YZY1Z3$	0 riešení
$0YZ$, XZZ	10 riešení (000, 011, 022, ..., 099)

Pokúste sa odhadnúť teoretickú zložitosť vášho programu – koľko operácií približne vykoná v závislosti na dĺžke reťazca a počte premenných v ňom.

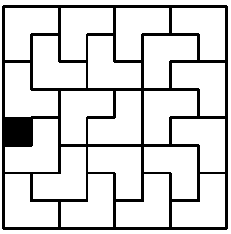
534. O postupnosti

Daná je postupnosť prirodzených čísel väčších ako 2 – t.j. $3, 4, 5, 6, \ldots$. V každom kroku s postupnosťou vykonáme nasledujúce operácie: určíme vedúci člen, čo je prvý člen postupnosti, preskočíme s prvým členom postupnosti toľko členov, koľko je jeho hodnota a zaradíme ho do postupnosti.

Úloha: Napíšte program, ktorý pre dané vstupné číslo zistí, v ktorom kroku sa prvýkrát stane vedúcim.

Príklad: 1.krok: $3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, \ldots$
2.krok: $4, 5, 6, 3, 7, 8, 9, 10, 11, 12, 13, \ldots$
3.krok: $5, 6, 3, 7, 4, 8, 9, 10, 11, 12, 13, \ldots$
4.krok: $6, 3, 7, 4, 8, 5, 9, 10, 11, 12, 13, \ldots$
Teda číslo 6 je prvýkrát vedúcim v 4. kroku.

535. O šachovnici



Daná je šachovnica rozmeru $n \times n$, kde n je mocnina 2, t.j. $n = 2^m$, kde m je celé číslo. Políčko šachovnice, ktoré má súradnice $[x, y]$ $(1 \leq x, y \leq n)$, je vystrihnuté. Napíšte program, ktorý zvyšok šachovnice rozstrihá na triminá tvaru „L“, čo je útvar zložený z troch štvorcov do tvaru „L“. Zamyslite sa, pre aké m a $[x, y]$ má úloha riešenie, a dokážte.

Príklad: Pre $n = 8$ $(m = 3)$, dieru $[x, y] = [1, 4]$ úloha má riešenie (jedno z riešení je na obrázku).

541. O peniazoch

Koľko existuje rôznych spôsobov vyplatenia – rozmenenia peňažnej sumy z rozmedzia 5 hal. až 10 Kčs pomocou mincí 5, 10, 20, 50 hal. a 1, 2, 5 Kčs?

Úloha: Napíšte program, ktorý pre zadanú sumu zistí počet jej rozmenení. Napríklad suma 25 hal. (t.j. 0.25 Kčs) sa dá rozmeniť 4 spôsobmi.

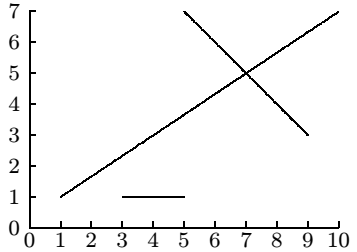
542. O úsečkách

V rovine je daných n úsečiek s koncovými bodmi $A_i = [x_i, y_i]$, $B_i = [u_i, v_i]$. Napíšte program, ktorý pre dané n prečíta n štvoríc x_i, y_i, u_i, v_i, $(1 \leq i \leq n)$ a vypíše všetky intervaly $\langle p_j, q_j \rangle$ s číslami f_j, pre ktoré platí, že z intervalu $\langle p_j, q_j \rangle$ na osi x „vidno“ úsečku s číslom f_j pri jej kolmom priemete na os x. Pri zisťovaní viditeľnosti predpokladajte, že úsečky sú „nepriesvitné“, t.j. „nižšia“ úsečka zakrýva „vyššiu“.

Príklad:

Vstup: $n = 3$
$A_1[1, 1]$, $B_1[10, 7]$,
$A_2[3, 1]$, $B_2[5, 1]$,
$A_3[5, 7]$, $B_3[9, 3]$

Výstup:
$\langle 1, 3 \rangle$ 1; $\langle 3, 5 \rangle$ 2; $\langle 5, 7 \rangle$ 1;
$\langle 7, 9 \rangle$ 3; $\langle 9, 10 \rangle$ 1



543. O permutáciách

Napíšte program, ktorý pre dané n vypíše všetky n-prvkové permutácie množiny $\{1, 2, \ldots, n\}$. Snažte sa vytvoriť program, ktorý potrebuje čo najmenej pamäti a pritom pracuje dosť rýchlo.

Príklad: Pre $n = 3$ je výsledok $(1, 2, 3)$, $(1, 3, 2)$, $(2, 1, 3)$, $(2, 3, 1)$, $(3, 1, 2)$, $(3, 2, 1)$.

544. O záhadných slovách

Záhadné reťazce sú reťazce písmen anglickej abecedy, ktoré vzniknú spojením (zreťazením) reťazcov u, v, s, t. Reťazce u, v pozostávajú z ľubovoľného (prípadne i nulového) počtu písmen anglickej abecedy a pre reťazce s, t platí, že s je zrkadlovým obrazom reťazca u a t je zrkadlovým obrazom reťazca v (zrkadlovým obrazom reťazca `abac` je reťazec `caba`).

Úloha: Napíšte program, ktorý načíta slovo a zistí, či je záhadné (v hore uvedenom zmysle). Príklad záhadných slov: `abcdbadc`, `abab`, `abccba`.

545. O kanibaloch a misionároch

K ľavému brehu potôčika Ľudožrútia slinka prišlo m misionárov a k kanibalov, ktorí sa majú pomocou malej dvojmiestnej loďky priviazanej k brehu preplaviť na druhý breh potôčika Ľudožrútia slinka. Do loďky sa však zmestia najviac dvaja (t.j. dvaja kanibali, alebo dvaja misionári, alebo misionár s kanibalom). V loďke môže ísť cez rieku aj samotný kanibal alebo misionár. Prázdna loďka sa cez rieku previezť nedá.

Úloha: Napíšte program ktorý určí, akým spôsobom sa majú misionári s kanibalmi preplaviť cez rieku tak, aby v žiadnom momente transportu na žiadnom brehu nebola presila kanibalov nad misionármi, lebo by ich zožrali. Samotní kanibali na brehu sa navzájom nepožerú. Ak úloha nemá riešenie, váš program o tom vypíše správu.

Príklad pre $m = k = 3$:

```
ľavý breh          lodička     pravý breh
(m m m k k k)                  ()
(m m k k)          (m k)->     (m k)
(m m m k k)        <-(m)       (k)
(m m m)            (k k)->     (k k k)
(m m m k)          <-(k)       (k k)
(m k)              (m m)->     (m m k k)
(m m k k)          <-(m k)     (m k)
(k k)              (m m)->     (m m m k)
(k k k)            <-(k)       (m m m)
(k)                (k k)->     (m m m k k)
(k k)              <-(k)       (m m m k)
()                 (k k)->     (m m m k k k)
```

611. O postupnostiach II

Napíšte program, ktorý pre dané n a $k \leq n$ určí, koľko existuje rôznych postupností dĺžky n, pozostávajúcich len z núl a jednotiek, v ktorých sa nevyskytuje k núl tesne za sebou.

Príklad: Pre $n = 5$ a $k = 3$ existuje 24 rôznych postupností.

612. O vojakoch

V rade (t.j. vedľa seba) stojí n vojakov. Na povel „vľavo vbok“ sa všetci náhodne otočia o 90°, niektorí vľavo, iní vpravo. O sekundu sa všetci, ktorí stoja tvárou v tvár svojmu susedovi, otočia o 180°. O ďalšiu sekundu tak isto, atď.

Úloha: Napíšte program, ktorý pre dané n modeluje tento dej, pokým sa niektorý z vojakov otáča. Program vypisuje stav vojakov po každej sekunde. Keď sa už žiadny z vojakov neotáča, program vypíše čas trvania a skončí. Viete zistiť (a zdôvodniť), ako najdlhšie môže trvať tento dej pre dané n?

613. O kódovaní

Pracovníci Strediska pre kozmický výskum sa snažia čo najefektívnejšie využívať možnosti rádiového spojenia na prenos správ. Posledný projekt na skrátenie priemernej dĺžky textu bol:

a) Zo vzorových textov správ sa pre každý použitý znak zistí frekvencia jeho výskytu (t.j. počet výskytov znaku v pomere k počtu všetkých znakov).
b) Používané znaky sa zakódujú do postupností núl a jednotiek tak, aby frekventovanejšie znaky mali kratší kód, menej frekventované dlhší.
c) Zakódované správy sa musia dať efektívne rozkódovať, preto sa autori rozhodli, že kód žiadneho znaku nesmie byť prefixom (začiatkom) iného kódu.
d) S použitím takto vzniknutej kódovacej tabuľky sa kódujú a dekódujú správy.

Pre vzorový text s ôsmimi znakmi: `BACADAEAFABBAAAGAH` je frekvencia nasledovná:

`A` – 9/18 `B` – 3/18 `C` – 1/18 `D` – 1/18
`E` – 1/18 `F` – 1/18 `G` – 1/18 `H` – 1/18

Keby sa tento text kódoval kódom s pevnou dĺžkou (v tomto prípade 3 bity), tak výsledný zakódovaný text by mal 54 bitov. Lepší kód vyhovujúci nášmu projektu je napríklad:

`A` → `0` `B` → `100` `C` → `1010` `D` → `1011`
`E` → `1100` `F` → `1101` `G` → `1110` `H` → `1111`

V tomto prípade zakódovaním vzorového textu dostaneme 42 bitov (núl a jednotiek):

`100010100101101100011010100100000111001111`

Za predpokladu, že vysielané a prijímané správy budú mať podobné frekvencie výskytov znakov, sa ušetrí asi 20% bitov (a aj vysielacieho času). Všimnite si, že kód:

`A` → `0` `B` → `100` `C` → `1001` `D` → `1011`
`E` → `1100` `F` → `1101` `G` → `1110` `H` → `1111`

projektu nevyhovuje, lebo kódy znakov `B`, `C` nespĺňajú podmienku z bodu c).

Úloha: Vytvorte program, ktorý podľa danej vzorovej správy vytvorí a vypíše frekvenčnú tabuľku použitých znakov, týmto podľa bodov b), c), d) priradí a vypíše kód a zakóduje, prípadne rozkóduje ďalší text.

614. O minimálnom absolútnom súčte

Je dané dvojrozmerné pole P rozmeru $n \times n$, v ktorom sú kladné čísla. Napíšte program, ktorý nájde postupnosť susedných prvkov začínajúcich prvkom $P[1,1]$ a končiacich prvkom $P[n,n]$ tak, aby súčet absolútnych hodnôt rozdielov susedných prvkov vybranej postupnosti bol minimálny. Susedné prvky majú spoločnú „stranu“ – susedné sú $P[2,3]$ a $P[2,4]$, ale nie $P[2,2]$ a $P[3,3]$.

615. O labyrintoch

Napíšte program, ktorý pre dané m a n naplní pole L s rozmermi $m \times n$ nulami a jednotkami tak, aby to bol „najprefíkanejší labyrint“, aký ste kedy videli. Labyrint znázorní graficky (napríklad pomocou hviezdičiek) a uvedie, kde je vchod do labyrintu a kde sa nachádza poklad (alebo uväznená princezná). Dodržte dohodu, že jednotka v poli L znamená múr a nula voľno.

Výsledok nie je presne definovaný, opiera sa o vašu predstavivosť a fantáziu. Pointou tejto úlohy je navrhnúť labyrint, ktorý má čo najbližšie k intuitívnej predstave labyrintu.

621. O múroch

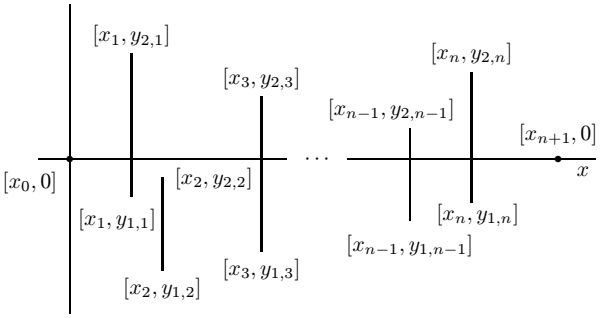
Janko Vtipko nám napísal, že u nich na ihrisku usporadúvajú bežecké preteky v behu medzi múrikmi. Medzi cieľom a štartom sú rovnobežné múriky, ktoré sú kolmé na os cieľ-štart. Žiaden múrik sa nesmie pri behu preskočiť. Jankovi je jasné, že ten, kto si vie nájsť medzi múrikmi najkratšiu dráhu, zvíťazí, preto vás požiadal o pomoc pri jej hľadaní.

Os cieľ-štart stotožníme s x-ovou osou; x_0 bude x-ová súradnica štartu, v našom prípade bude vždy rovná 0. Ak bude na trati n múrikov, potom x_{n+1} bude x-ová súradnica cieľa, x_i bude x-ová súradnica i-teho múrika a $y_{1,i}$ a $y_{2,i}$ budú y-ové súradnice koncových bodov i-teho múrika.

Úloha: Napíšte program, ktorý pre dané n, x_i, $y_{1,i}$, $y_{2,i}$, x_{n+1} ($1 \le i < n+1$, $0 < x_i < x_{i+1}$, $y_{1,i} < y_{2,i}$) nájde najkratšiu lomenú čiaru so začiatkom v bode $[0,0]$ a koncom

v bode $[x_{n+1}, 0]$, ktorá nepretína žiaden múrik. Súradnice bodov, v ktorých sa lomená čiara „láme“ budú $[a_1, b_1], \ldots, [a_m, b_m]$.

Lomená čiara sa môže múrika (teda jeho koncového bodu) dotýkať, múrik si predstavte ako úsečku. Zo zadania nevyplýva, že $y_{1,i}$ a $y_{2,i}$ majú rôzne znamienka.



622. O n-uholníku

Vo vrcholoch n-uholníka sú celé čísla, ktorých súčet je kladný. Máme danú nasledovnú operáciu nad číslami x, y, z v troch za sebou idúcich vrcholoch (y je stredný z nich): ak $y < 0$, tak postupne vykonáme tieto priradenia

$$y := -y; \quad x := x - y; \quad z := z - y$$

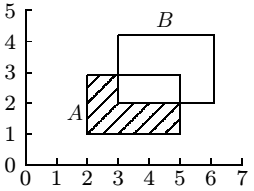
Príklad: Nech $x = 3$, $y = -2$, $z = -2$, potom po vykonaní operácie s danými číslami $x = 1$, $y = 2$, $z = -4$.

Úloha: Napíšte program, ktorý dostane na vstupe číslo n a n-ticu celých čísel, ktoré sú vo vrcholoch n-uholníka. Program len použitím uvedenej operácie zmení dané čísla tak, aby boli všetky nezáporné. Pritom bude vypisovať, na ktoré čísla (vrcholy) použil operáciu, alebo oznámi, že to pre dané čísla nie je možné.

Zdôvodnite (dokážte) správnosť svojho programu.

623. O obdĺžnikoch

Obdĺžnik v rovine je určený súradnicami jeho ľavého horného a pravého dolného vrcholu, jeho strany sú rovnobežné so súradnými osami.



Úloha: Napíšte program, ktorý pre dané dva obdĺžniky A a B vypočíta a vypíše veľkosť plochy rozdielu obdĺžnikov A mínus B ($A - B$). Teda plochu A bez plochy, ktorú z neho „vysekne“ B.

624. O spriatelených číslach

Označme $s(x)$ súčet všetkých kladných deliteľov čísla x menších ako x. Dve rôzne kladné čísla x a y nazývame spriatelené, ak $s(x) = y$ a $s(y) = x$.

Úloha: Napíšte program, ktorý pre dané n vypočíta a vypíše všetky dvojice spriatelených čísel x a y takých, že $x \leq n$ a $x < y$. Skúste pre $n = 5000$.

625. O lexikálnej analýze

Lexikálny analyzátor je „zariadenie“, ktoré dostáva na vstup text v nejakom jazyku (pre naše účely v programovacom) a na výstup dáva tento text rozložený na lexémy daného

jazyka. Lexémy sú vlastne základné „stavebné kamene“ daného jazyka. V programovacích jazykoch sú to kľúčové slová (v BASICu napríklad **let**, **if**, **then**, **else**, atď.; v Pascale napríklad **begin**, **end**, **repeat**, **until**, **do**, atď.), špeciálne symboly (napríklad čiarka, bodkočiarka, plus, mínus, menšie, väčšie...), čísla (celé, reálne), reťazce (znaky uvedené v apostrofoch prípadne úvodzovkách) a identifikátory (názvy premenných, procedúr a funkcií).

Nech nejaký programovací jazyk má takéto lexémy:

- kľúčové slová: `LET IF THEN FOR TO NEXT INPUT PRINT GOTO OR AND`
- viacznakové symboly: `<= >= <>`
- jednoznakové symboly: `< > = ( ) , : ; + - *`
- identifikátory: postupnosť písmen a číslic, ktorá začína písmenom a neobsahuje kľúčové slovo ako svoj podreťazec
- celočíselné konštanty: postupnosť niekoľkých cifier

Medzery v riadku nemajú žiaden význam, medzi jednotlivými lexémami nemusí byť žiadna medzera. Analyzátor nikdy nesmie rozoznať za sebou dva identifikátory alebo za identifikátorom číslo (teda rozoznáva čo najdlhší identifikátor).

Úloha: Napíšte program, ktorý prečíta riadok programu v programovacom jazyku a vypíše postupnosť lexém, ktoré sa v riadku nachádzajú (v správnom poradí). Výstup nech má tvar, aký uvádzame v príklade. Ak sa na vstupe vyskytne znak, ktorý nie je v žiadnej lexéme, program vypíše lexémy po toto miesto a chybovú správu.

Príklad:

Vstup:
```
10 IFA<=BORC12T HEN120
```
Výstup:
```
CELE CISLO        10
KLUCOVE SLOVO     IF
IDENTIFIKATOR     A
VIACZN. SYMBOL    <=
IDENTIFIKATOR     B
KLUCOVE SLOVO     OR
IDENTIFIKATOR     C12
KLUCOVE SLOVO     THEN
CELE CISLO        120
```

Vstup:
```
10 IFA<=F ORC12THEN120
```
Výstup:
```
CELE CISLO        10
KLUCOVE SLOVO     IF
IDENTIFIKATOR     A
VIACZN. SYMBOL    <=
KLUCOVE SLOVO     FOR
IDENTIFIKATOR     C12
KLUCOVE SLOVO     THEN
CELE CISLO        120
```

Takto pracujú lexikálne analyzátory niektorých interpreterov BASICu. V iných interpreteroch sa berú do úvahy aj medzery (to je však ešte trochu zložitejšie).

631. O delení polynómov

Výraz tvaru $p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, kde $a_n, \ldots, a_1, a_0$ sú celé čísla, pričom $a_n \neq 0$, nazývame *polynóm n-tého stupňa premennej x s celočíselnými koeficientmi* $a_n, \ldots, a_1, a_0$.

Podielom a zvyškom po delení dvoch polynómov $a(x)$ a $b(x)$ sú polynómy $q(x)$ a $r(x)$ také, že platí $a(x) = b(x)q(x) + r(x)$ a stupeň polynómu $r(x)$ je menší ako stupeň polynómu $b(x)$.

Úloha: Napíšte program, ktorý pre dané polynómy $a(x)$ a $b(x)$ vypočíta ich podiel $q(x)$ a zvyšok $r(x)$ (ak neexistujú, podá o tom správu). Snažte sa, aby program vypisoval polynómy $r(x)$ a $q(x)$ v „čo najčitateľnejšom tvare“, tzn. najlepšie v takom, na aký sme zvyknutí z matematiky (s ohľadom na zápis exponentov). Vstup programu sú polynómy $a(x)$ a $b(x)$ zadávané po koeficientoch $a_n, \ldots, a_1, a_0$.

PRÍKLAD:

VSTUP:
polynóm $a(x)$: $2, -4, 3, 2$;
polynóm $b(x)$: $-1, 0, 1$.

VÝSTUP:
```
a(x)=2x^3-4x^2+3x+2, b(x)=-x^2+1
q(x)=a(x) div b(x) = -2x+4,
r(x) = a(x) mod b(x) = 5x-2
```

Odporúčame vám urobiť si pomocné programy na sčítanie a násobenie polynómov, aby ste mohli skontrolovať výsledky – „ručne“ sa ľahko pomýlite a je to zdĺhavé.

632. O žabách

Na hladine rybníka na $n+1$ kameňoch sedí n žiab so štartovými číslami od 1 po n. Na začiatku žaby sedia na kameňoch v rade vedľa seba, zoradené podľa ich čísel od najmenšej po najväčšiu. Vedľa žaby s číslom 1 je jeden kameň voľný. Situácia teda vyzerá takto: _ 1 2 ... n. Každá žaba môže skákať len na voľný kameň, a to

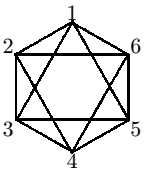
a) ak je susedný, alebo
b) ak je susedný susednej žabe.

Žabám treba poradiť, v akom poradí majú skákať, aby sedeli na kameňoch v opačnom poradí (od najväčšej po najmenšiu) a voľný kameň bol vedľa žaby s číslom n (_ n ... 3 2 1). Vašou úlohou je radiť im tak, aby museli čo najmenej skákať.

ÚLOHA: Napíšte program, ktorý pre dané n vypíše postupnosť skákaní žiab po kameňoch a počet potrebných skokov typu I a II.

633. O kresliči

Predstavte si, že máte riadiť súradnicový zapisovač, ktorý kreslí obrázky pomocou príkazu `KRESLI_DO_BODU` x, y (kreslí úsečku z momentálneho bodu do bodu $[x, y]$) a príkazu `POSUN_DO_BODU` x, y (presúva sa do bodu $[x, y]$). Aby toto zariadenie kreslilo čo najrýchlejšie, je potrebné minimalizovať dĺžku dráhy pera (dráhu pera tvoria úseky, keď je pero spustené na papieri a kreslí a úseky, keď sa presúva ponad papier).



ÚLOHA: Budeme chcieť napísať program, ktorý pre zadané n nakreslí pravidelný n-uholník (vpísaný do kružnice s polomerom 1) a všetky jeho uhlopriečky, ktoré neprechádzajú jeho stredom. Príklad pre $n = 6$ je na obrázku vľavo.

Dobrá víla už náš súradnicový zapisovač naučila súradnice vrcholov mnohouholníka. Nám bude teda stačiť nájsť postupnosť príkazov `KRESLI_DO` k a `POSUN_DO` k, kde k je poradové číslo vrcholu v číslovaní proti smeru chodu hodinových ručičiek. Na začiatku je pero nastavené vo vrchole 1.

Napíšte program, ktorý k zadanému n nájde a vypíše čo najlepšiu postupnosť príkazov. Napríklad pre $n = 6$ by mohol vypísať:

```
KRESLI_DO 2, POSUN_DO 1, KRESLI_DO 3, POSUN_DO 1, KRESLI_DO 5,
POSUN_DO 1, KRESLI_DO 6, POSUN_DO 1, POSUN_DO 2, KRESLI_DO 3,
POSUN_DO 2, KRESLI_DO 4, POSUN_DO 2, KRESLI_DO 6, POSUN_DO 2,
POSUN_DO 3, KRESLI_DO 4, POSUN_DO 3, KRESLI_DO 5, POSUN_DO 3,
POSUN_DO 4, KRESLI_DO 5, POSUN_DO 4, KRESLI_DO 6, POSUN_DO 4,
POSUN_DO 5, KRESLI_DO 6
DĹŽKA DRÁHY PERA JE 35.7846.
```

Dĺžka dráhy pera v tomto príklade nie je optimálna.

634. O kódovaní kombinácií

Nech n a k sú prirodzené čísla. Uvažujme všetky k-prvkové podmnožiny množiny $\{1, \ldots, n\}$, lexikograficky usporiadané (od najmenšej po najväčšiu). Ich počet je $p = \binom{n}{k}$ a sú očíslované v poradí číslami $1, \ldots, p$.

ÚLOHA: Napíšte program, ktorý k danému n, k a číslu x z množiny $\{1, \ldots, p\}$ priradí k-prvkovú podmnožinu množiny $\{1, \ldots, n\}$ s poradovým číslom x.

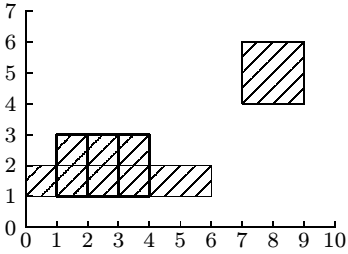
Napríklad keď $n = 4$ a $k = 2$, podmnožina s poradovým číslom 3 je $\{2, 3\}$, keď $n = 4$ a $k = 4$, podmnožina s poradovým číslom 0 je $\{1, 2, 3, 4\}$ a keď $n = 4$ a $k = 3$, podmnožina s poradovým číslom 1 je $\{1, 2, 4\}$.

Program by nemal vytvárať všetky podmnožiny od 0-tej po x-tú.

635. O zjednotení obdĺžnikov

Obdĺžnik v rovine, ktorého strany sú rovnobežné s osami, je určený súradnicami jeho ľavého dolného a pravého horného vrcholu (nemusia byť celočíselné).

Úloha: Napíšte program, ktorý pre daných n obdĺžnikov $A_1, A_2, \ldots, A_n$ vypočíta plochu ich zjednotenia.



Príklad: pre $n = 4$,
1. obdĺžnik je určený bodmi $[1; 1], [4; 3]$
2. obdĺžnik je určený bodmi $[0; 1], [6; 2]$
3. obdĺžnik je určený bodmi $[2; 1], [3; 3]$
4. obdĺžnik je určený bodmi $[7; 4], [9; 6]$

Plocha zjednotenia je 13.

641. O štvorčekoch

Je daná nekonečná štvorčeková rovina (dostatočne veľký štvorčekový papier), na ktorej je zafarbených na čierno n štvorčekov (všetky ostatné sú biele). V každej sekunde sa vykoná prefarbenie a vznikne nový obrazec štvorčekov.

Pri prefarbovaní bude mať každý štvorček takú farbu, ktorá prevláda na ňom, jeho hornom susedovi a jeho pravom susedovi (susedné štvorčeky sú tie, čo majú spoločnú stranu). Prefarbovanie sa zastaví, ak už neexistuje žiaden čierny štvorček.

Úloha: Napíšte program, ktorý pre daný počiatočný obrazec počíta a postupne vypisuje vznikajúce obrazce. Existuje obrazec, ktorého prefarbovanie sa nikdy neskončí? Zamyslite sa nad tým, ako dlho môže prefarbovanie trvať pre n štvorčekov. Svoju hypotézu dokážte.

642. O čudných permutáciách

Nech M je daná neprázdna podmnožina prirodzených čísel obsahujúca m navzájom rôznych prvkov.

Úloha: Napíšte program, ktorý pre danú množinu M vytvorí takú permutáciu jej prvkov, že medzi žiadnymi dvoma prvkami v tejto permutácii nie je ich aritmetický priemer.

Príklad: $M = \{1, 2, 3, 4, 5, 6\}$; výsledkom je napríklad $\{1, 5, 6, 3, 2, 4\}$.

Prvok „medzi“ dvoma prvkami nemusí byť susedný ani s jedným z nich.

643. Kreslenie jedným ťahom

Možno viete, že ľubovoľný súvislý obrázok zložený z čiar sa dá nakresliť jedným uzavretým ťahom (teda takým, že kreslenie skončíme v tom bode, z ktorého sme začali) práve vtedy, keď sa v každom priesečníku čiar stretáva párny počet čiar. Priesečníky nazveme vrcholmi, čiary nazveme hranami.

Úloha: Napíšte program, ktorý prečíta počet vrcholov n a niekoľko dvojíc vrcholov (i, j), ktoré sú spojené hranou. Potom vypíše takú postupnosť čísel vrcholov, ktorá určuje, ako sa dá nakresliť daný obrázok jedným uzavretým ťahom. Ak sa daný obrázok nedá nakresliť jedným uzavretým ťahom, program to musí zistiť a oznámiť. (Pozor, môže to byť aj vtedy, ak obrázok nie je súvislý.)

644. O upratovaní

V sklade majú jeden druh škatúľ v troch farbách, a to modré, červené a biele. Všetky škatule majú na jednej polici. Skladník potrebuje usporiadať škatule tak, aby boli uložené

najprv všetky biele škatule, potom všetky modré škatule a na konci červené škatule. Všetky ostatné police v sklade však boli obsadené, a ani prefarbiť škatule nesmel. Skladníkovi neostalo iné, len pohýbať rozumom, aby vyriešil úlohu, ktorú si zaumienil.

Úloha: Je dané m-prvkové pole, predstavujúce policu so škatuľami. Prvkami tohto poľa sú čísla 1, 2, 3, pričom každé číslo predstavuje jednu farbu škatúľ. Napíšte program, ktorý usporiada prvky tohto poľa tak, ako to potrebuje skladník. Nemôžete pri tom však použiť pomocné pole (pomocnú policu), ani si spočítať, koľko prvkov má byť ktorej farby, a potom ich do poľa len vypísať (prefarbenie škatúľ). Môžete len navzájom vymieňať niektoré dvojice prvkov poľa. Váš program by mal vypísať postupnosť dvojíc škatúľ v takom poradí, v akom ich má skladník vymieňať, aby policu upratal do požadovaného tvaru. Snažte sa, aby skladník nemusel urobiť viac než m výmen škatúľ.

645. O kalkulačke

Dôležitou súčasťou prekladačov programovacích jazykov je preklad aritmetických výrazov do strojového kódu. Vašou úlohou bude naprogramovať takýto jednoduchý prekladač.

Strojový kód počítača si nahraďme jednoduchým kalkulačkovým kódom. Predstavme si, že máme kalkulačku, ktorá má 26 pamäťových miest označených A až Z, dostatočne veľa pomocných pamäťových miest označených $p_1, p_2, \ldots$ a jeden špeciálny register – sumátor (to je akoby displej kalkulačky). Sumátor označme \$. V ďalších riadkoch p označuje meno ľubovoľného (aj pomocného) pamäťového miesta a p_k označuje meno ľubovoľného (aj pomocného) pamäťového miesta alebo celočíselnú konštantu. Kalkulačka pozná takéto príkazy:

$\$:= p_k$	do sumátora ulož hodnotu z p_k
$p := \$$	hodnotu zo sumátora ulož do p
$\$ + p_k$	k sumátoru pripočítaj hodnotu z p_k
$\$ - p_k$	od sumátora odčítaj hodnotu z p_k
$\$ * p_k$	vynásob sumátor hodnotou z p_k
$\$ / p_k$	vydeľ sumátor hodnotou z p_k
$\$ \,\hat{}\, p_k$	umocni sumátor na hodnotu p_k
@\$	zmeň znamienko sumátora (vynásob ho -1)

Príklad: Predpokladajme, že v pamätiach A a B už máme uložené nejaké hodnoty a chceme vypočítať hodnotu výrazu $(-A * B + B * 2) \,\hat{}\, 2$ a výsledok uložiť do pamäti C (chceme vlastne vykonať priraďovací príkaz $C := (-A * B + B * 2) \,\hat{}\, 2$), musíme dať kalkulačke (napríklad) takúto postupnosť príkazov:

$$\$:= A, \quad \$ * B, \quad @\$, \quad p_1 := \$, \quad \$:= B, \quad \$ * 2, \quad \$ + p_1, \quad \$ \,\hat{}\, 2, \quad C := \$$$

Úloha: Napíšte program, ktorý prečíta priraďovací príkaz a vypíše postupnosť príkazov pre kalkulačku, ktorá vykoná tento priraďovací príkaz. Výraz na pravej strane je zostavený z mien pamäťových miest, celočíselných konštánt, binárnych operátorov $+$, $-$, $*$, $/$, $\hat{}$, unárnych operátorov $+, -$ a zátvoriek. Pre tvar výrazu na pravej strane si môžete vybrať jeden z dvoch variantov (variant b je podstatne ťažší, stačí vyriešiť len ten).

a) Výraz je úplne uzátvorkovaný. To znamená, že každý operátor je spolu so svojimi dvoma operandmi (alebo jedným, ak je to unárne mínus alebo plus) uzavretý v zátvorkách. Postupnosť výpočtu je teda jasne daná zátvorkami. Úplne uzátvorkovaný variant výrazu z našej ukážky je $(((-(A * B)) + (B * 2)) \,\hat{}\, 2)$. Program môže predpokladať, že doň vstupujú len správne a úplne uzátvorkované výrazy.

b) Pre poradie operácií platia bežné priority – najskôr sa umocňuje, potom sa násobí a delí, a až potom sa sčituje a odčituje. Zátvorky menia poradie vykonávania operácií bežným spôsobom. Unárne plus a mínus majú rovnakú prioritu ako sčítanie. Vo výraze nemôžu nasledovať po sebe dva operátory, teda napr. nemôžeme napísať $A + -B$.

Umocňovanie je sprava asociatívne, teda $A \wedge B \wedge C = A \wedge (B \wedge C)$. Všetky ostatné operácie sú zľava asociatívne, teda napr. $A - B - C = (A - B) - C$.

Váš program musí zistiť, či je výraz správny – ak je nesprávny, vypíše správu. Pri riešení variantu b) odporúčame použiť oddelený lexikálny analyzátor (pozri úlohu 625).

711. O protivných intervaloch

Na číselnej osi máme daných n uzavretých intervalov $\langle a_i, b_i \rangle$. Zaujíma nás, ktoré čísla ležia práve v párnom počte intervalov. Množinu týchto čísel môžeme opísať rôznymi spôsobmi ako zjednotenie niekoľkých intervalov. Nás však zaujíma to vyjadrenie, ktoré obsahuje najmenej intervalov.

Príklad: Pre dané $n = 4$ a intervaly $\langle 2, 5 \rangle$, $\langle 1, 5 \rangle$, $\langle 0, 8 \rangle$ a $\langle 4, 7 \rangle$ je možné množinu bodov, ktoré sa nachádzajú práve v párnom počte intervalov, opísať napríklad ako zjednotenie piatich intervalov $(-\infty, 0)$, $\langle 1, 2 \rangle$, $\langle 4, 5 \rangle$, $\langle 5, 7 \rangle$ a $(8, +\infty)$. Dá sa však opísať aj ako zjednotenie štyroch intervalov $(-\infty, 0)$, $\langle 1, 2 \rangle$ a $\langle 4, 7 \rangle$ a $(8, +\infty)$. Všimnite si, že výsledné intervaly nemusia byť uzavreté.

Úloha: Napíšte program, ktorý prečíta prirodzené číslo n a n dvojíc koncových bodov uzavretých intervalov $\langle a_i, b_i \rangle$ a nájde množinu bodov, ktoré sa nachádzajú v práve párnom počte intervalov. Množinu vypíše v tvare zjednotenia čo najmenšieho počtu intervalov.

712. O bezpečnostnom vedení

Bezpečnostným vedením nazveme štvorcovú sieť z $n \times n$ vodivých drôtov, ktoré sú v uzlových bodoch vodivo spojené. Drôty sú očíslované (pre jednoznačnosť zhora nadol od 1 po n a zľava doprava od 1 po n). Priesečník drôtov i a j označíme $[i, j]$. Vývody spojenia sú v protiľahlých vrcholoch štvorca (vo vrcholoch $[1, 1]$ a $[n, n]$).

Vojaci majú vedenú telefónnu linku z miesta A do miesta B bezpečnostným vedením $n \times n$ (kde A a B sú vývody spojenia). Nepriateľ im objavil a zničil niektoré vodivé uzly. Zničené uzly nevedú prúd ani v jednom smere (celý uzol aj s kusmi drôtov okolo je zo siete vyseknutý). Zaujíma nás, či sa napriek tomu dá dosiahnuť spojenie medzi mestami A a B.

Úloha: Napíšte program, ktorý pre danú veľkosť bezpečnostného vedenia a dané súradnice zničených uzlov vypíše, či je možné nadviazať spojenie medzi vrcholmi $[1, 1]$ a $[n, n]$.

713. O rímskych číslach

Na vstupe sú dva reťazce, obsahujúce iba znaky I, V, X, L, C, D a M, reprezentujúce dve čísla v rímskom zápise. Napíšte program, ktorý vypíše ich súčet a rozdiel, tiež v rímskom zápise, pričom bude pracovať, ako rímski úradníci, iba s ich rímskym zápisom. Môžete predpokladať, že prvé číslo je väčšie ako druhé.

Príklad: Pre vstupné hodnoty XXIII a IV sú výsledné hodnoty XXVII a XIX.

714. O blábole

Kozmické Stredisko pre Prieskum (KSP) vyslalo sondu na neznámu planétu. Sonda funguje tak, že preskúma svoje okolie a odvysiela do Strediska to najzaujímavejšie, čo sa tam nachádza. Planéta je veľmi ďaleko od Zeme a rôzne šumy správu sondy rušia a komolia. Preto sonda správu vysiela stále dookola.

V Stredisku majú už len jednu úlohu. Z prijatého rušeného textu získať *maximálne pravdepodobnú správu* sondy. Maximálne pravdepodobná správa je taká, že keď sa cyklicky podpíše pod prijatý text, je počet znakov, v ktorých sa líšia, minimálny.

Úloha:

a) Pomôžte Kozmickému Stredisku pre Prieskum a napíšte program, ktorý pre daný text a číslo $k > 0$ vypíše všetky maximálne pravdepodobné správy dĺžky k. (Dĺžka textu nemusí byť celočíselným násobkom čísla k.)

b) Zistite, čo zaujímavé objavila sonda na planéte, keď prijatý text vyzeral takto:

```
BLABOLQIALVBPTPVGYWXRKBTAWZXNTCZIAWQFLGBDNPZYKIAWEFHIONIIEYHNSAXHEKSE
HVWQDEQDYZDYMABFDYIBCVKCNGTRNBDKZPZCSBHICIZIUNFIUGCKMCFKFQIGUNAEPIRUK
YVSEIIFUBCXFMIUHCBTSBVNGBMDNYDVCORXMDNIGHZPRREGCEQRILDMPBPOBOIXFUTEHQ
NDTDNMNLKEYLSHYZKAFADIAYTAUNVVZEGYQMQXJXVVUTODVKNQCQXCPACAJYVUINKASNG
KTDOVJUDTTISPMYAZXIBBHLLWLEJPGNGXDDXEGOXQOENKHKCAAUJRQYXONFPAPRFJQNVL
CHRIMOVGGZANYUMDAFXEHQLNXVNNGBVBENICHZQGQNBNKQRNNDNERRCGBSHXZLTGMCIHA
XFEJXUGFADNGUXVWPYLROEVTFKCWPWGCAVVCPDSOUOKMSMDFHDOHTMYKSPIKDOUNIYZTL
ECVANUHLCNIERUVNQIRGIOHFRLCLUORICNICGHYJAMNMMUXCJCENUVARWOMRNDYNWPDNA
ZECWINYXCMMFKGRTSTBTFHISXANBRYOSYNMQHGYBNJZVEKDEUOVDBWZSGOILRPKWDPNQC
NIRQUSNIRSKQIOKGTWLAUKHKCMFYYYVIALKALBAYEZAJZUJBWFESQQRACEUEGPWABBEMX
PPHNFGKMZHRHKCWFTQNAONUMNTINLRRSLUIVURCQZFEPVHNKNRTUBRHCNWZWNHMAIGVRN
JZWUNRBIWCZENOGWDOQUFIINPSGDLPAMPETWTSEHICQASRPLMUEPWUDKEBMIAMZPZZNDH
QIMFYPJIFZSBYEMFSMCBEMHNYCSQCMTLRPWVHKHCQEBQMCMOHWWYJBAAIRZTRSEBQNHHK
GFCRHCVBWKJBJFVEECETFIEWGIBYNUYROXRDHDRNESIIBVTSDJKLWRCITCLWMISVQKCCW
JFQZPUTREVKIFSDRDENCNSJAHQCRXWBIINRIFUTNASWBMNVAMGBUCMNDPNCYWBRPEDLCC
TABRIPAAKYXDKIQNTJKXZRZDMKBZCYMXXRMVXFIXRSTTVTQQNZBONFKVCLTANHGGCKQOL
TSHWSJIWLSNUWISYRJJNAMGWVICZIXBIRNLARBIMDYJWUYYCMGJHMMZQKHHDZCGOOQRDP
OEAIRAWGFKCVHONZHNTITOXBWNYFDYJXHZISDWPEDCHIBGSKLFAUIELPHNJKYDINPXYDA
IVARCPAAJBEHHWIZOROJTAICMICNDHYXLJDXATEQFCENFRWONHLKUDRTVP
```

715. O domine

Na stole máme rozsypaných n kociek matematického domina. Každá kocka takéhoto domina má na každej polovici jednu cifru v desiatkovej sústave (t.j. celé číslo z intervalu $\langle 0, 9\rangle$). Tieto kocky treba poukladať širšími stranami k sebe a číslami nahor, pričom sa môžu aj otáčať. Takýmto uložením vzniknú dve čísla, pričom jedno sa číta v dolnom rade od prvej kocky po poslednú a druhé v hornom rade od poslednej po prvú.

Úloha: Napíšte program, ktorý

a) uloží kocky tak, aby súčet dvoch vzniknutých čísel bol maximálny,
b) uloží kocky ako v prípade a), ale s predpokladom, že kocky domina sa môžu ukladať k sebe, iba ak majú aspoň jednu cifru rovnakú (a potom aj musia ležať touto cifrou pri sebe) a nemusíme použiť všetky kocky.

721. O priemere

Daná je postupnosť n čísel, kde $n > 0$. Treba nájsť čo najdlhšiu súvislú podpostupnosť s priemerom väčším alebo rovným ako dané p.

Úloha: Napíšte program, ktorý načíta kladné celé číslo n, reálne číslo p a postupnosť reálnych čísel dĺžky n a vypíše prvý a posledný index hľadanej podpostupnosti. Ak neexistuje, tak vypíše o tom správu.

Priemer postupnosti $a_1, a_2, \ldots, a_n$ je číslo $\bar{x} = (a_1 + a_2 + \cdots + a_n)/n$.

722. O veľkom zlomku

Máme daný zlomok, ktorého čitateľ aj menovateľ sú v tvare súčinu viacerých celých čísel. Zlomok má teda tvar

$$\frac{a_1 \cdot a_2 \cdot \ldots \cdot a_n}{b_1 \cdot b_2 \cdot \ldots \cdot b_m}.$$

Pritom každé z čísel a_i a b_j je dostatočne malé (zmestí sa do celočíselnej premennej), ale môže ich byť veľmi veľa. Súčiny v čitateli a menovateli sa teda už takmer iste nezmestia do jednej celočíselnej premennej. Zaujíma nás *základný tvar* daného zlomku (teda tvar, v ktorom bude čitateľ nesúdeliteľný s menovateľom).

Príklad:

$$\frac{25 \cdot 2 \cdot 320 \cdot (-125) \cdot 131 \cdot 100}{128 \cdot 25 \cdot 25 \cdot 25 \cdot 15} = \frac{-2620}{3}$$

Úloha: Napíšte program, ktorý prečíta celé čísla m a n, $m, n \leq 1000$ a dve postupnosti celých čísel $a_1, a_2, \ldots, a_n$ a $b_1, b_2, \ldots, b_m$ $(2 \leq a_i, b_i \leq 500)$ a vypíše základný tvar zlomku $(a_1 \cdot a_2 \cdot \ldots \cdot a_n)/(b_1 \cdot b_2 \cdot \ldots \cdot b_m)$. Výsledok programu môže byť v rovnakom tvare ako daný zlomok (teda v tvare podielu dvoch súčinov).

723. O stabilných úradníkoch

V úrade je n úradníkov, ktorí každú hodinu preberajú lajstrá. V celom úrade je sto miliónov lajstier na úradníckych stoloch. Každú celú hodinu zaznie gong. Vtedy každý úradník vstane, obíde stoly kolegov, z každého si zoberie časť lajstier a vráti sa na svoje miesto. Keď už opäť všetci sedia na miestach, každý pridá práve nazbierané lajstrá na svoju kopu.

Nech $U_{i,j} > 0$ hovorí, koľko percent lajstier vezme každú hodinu úradník i zo stola úradníka j. Percentá sa počítajú z počtu lajstier, ktoré tam poslednú hodinu ležali, nezáleží teda na poradí, v akom si úradníci lajstrá berú. Aj hodnota $U_{i,i} > 0$, úradníkovi teda nejaká časť jeho lajstier zostane. Samozrejme, čísla v každom stĺpci matice U musia mať súčet presne 100.

Úradníci si po niekoľkých rokoch činnosti úradu všimli, že veľkosti ich kôpok lajstier sa prestali meniť.

Úloha: Napíšte program, ktorý pre dané n a vyššie popísanú maticu U nájde „stabilné“ rozloženie lajstier u úradníkov.

724. O zjednotení

Máme postupnosť disjunktných množín $z_1, z_2, \ldots, z_n$. Poznáme ich počty prvkov v_1, v_2, $\ldots$, v_n. Chceme získať zjednotenie $z_1 \cup z_2 \cup \cdots \cup z_n$. Na „vyrobenie“ tohto zjednotenia môžeme použiť iba operáciu $\cup$, ktorá zjednotí dve množiny z_i a z_j. Vykonanie tejto operácie trvá $v_i + v_j$ časových jednotiek (časovú jednotku označme *čj*). Zaujíma nás, ako máme postupne zjednocovať množiny, aby bol výsledný čas čo najmenší.

Napríklad ak máme tri množiny: 2, 10 a 7 prvkovú, tak ak zjednotíme najprv prvé dve (to trvá $2 + 10 = 12$ *čj*, vznikne 12 prvková množina), potom k výsledku pridáme tretiu (na $12 + 7 = 19$ *čj*), trvá to spolu 31 *čj*. Ak však zjednotíme prvú a tretiu $(2 + 7 = 9$ *čj*) a potom k nim druhú $(9 + 10 = 19$ *čj*), výsledný čas bude len 28 *čj*.

Úloha: Napíšte program, ktorý prečíta prirodzené číslo n a postupnosť prirodzených čísel $v_1, v_2, \ldots, v_n$, ktoré predstavujú veľkosti množín. Výsledkom programu bude predpis na získanie zjednotenia za minimálny čas a tento čas. Predpis sa skladá z „príkazov“ $i \cup j = k$, kde i a j sú čísla množín, ktoré sa majú zjednotiť a k je číslo, ktorým označíme číslo výslednej množiny. Prvý predpis z nášho príkladu by sme zapísali napríklad $1 \cup 2 = 4$, $4 \cup 3 = 5$. (Čísla označujúce výsledné množiny môžete voliť ľubovoľne, najlepšie je to však postupne od $n + 1$).

725. O súboroch

Softwarové družstvo SODR si kúpilo lacno počítač ABM kompatibilný s ničím a potrebuje doňho dorobiť základné programové vybavenie. Tento počítač má jeden disk, na ktorom sú uložené súbory. Úplné meno súboru na počítači ABM sa skladá z názvu súboru, bodky a prípony. Názov súboru aj prípona sa skladajú iba z veľkých písmen a majú najviac 20 písmen. Typické meno súboru na počítači ABM je napríklad

`KONDICIOGRAMSUPRAVOU.PASCALOVSKYPROGRAM`.

Pri práci so súbormi (kopírovanie, rušenie a pod.) je niekedy potrebné označiť naraz viac súborov so spoločnou niektorou časťou mena. Na to slúži hviezdičková konvencia:

Pri zápise mena súboru sa môžu použiť znaky `?` (otáznik) a `*` (hviezdička), pričom otáznik zastupuje jedno ľubovoľné písmeno a hviezdička zastupuje ľubovoľne dlhý reťazec (aj prázdny) ľubovoľných písmen. Napríklad zápisu `?ONDI*S*UPR*.*PAS*` vyhovuje aj horeuvedené meno súboru.

Úloha: Napíšte čo najrýchlejší program, ktorý najprv prečíta meno súboru v hviezdičkovej konvencii a potom pre prichádzajúce úplné mená súborov (teda bez hviezdičiek a otáznikov) vypisuje `VYHOVUJE` alebo `NEVYHOVUJE`, podľa toho, či úplné meno vyhovuje danému hviezdičkovému zápisu alebo nie. Činnosť sa končí súborom s prázdnym názvom aj príponou (t.j. iba bodka). Počítajte s tým, že mien, ktoré má program vyhodnotiť pre jeden hviezdičkový zápis, môže prísť veľmi veľa.

731. O tetraminách

Na štvorčekovom papieri máme nakreslený pôdorys miestnosti. Predpokladáme, že miestnosť sa skladá z nejakého počtu celých štvorčekov, ktoré navzájom susedia aspoň jednou stranou. Túto miestnosť máme „vyparketovať“ parketami tvaru tetramín. *Tetraminá* sú útvary zo štyroch štvorčekov (vysvetlenie pre hráčov tetrisu: tetraminá sú útvary, ktoré padajú v tetrise). Existujú takéto tetraminá:

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

Pri parketovaní môžeme jednotlivé tetraminá ľubovoľne otáčať aj prevracať a každé z nich môžeme použiť pri parketovaní ľubovoľný počet krát.

Úloha: Napíšte program, ktorý prečíta zo vstupu tvar miestnosti a vytlačí jej pokrytie tetraminami, alebo ak pokrytie nie je možné, napíše o tom oznam.

Vstup zadávajte tak, že miestnosť ohraničíte obdĺžnikom $m \times n$ a zadáte m, n a pole $m \times n$ núl a jednotiek, kde jednotky určujú štvorčeky patriace miestnosti. Program môže predpokladať, že na vstupe je zadaná jedna miestnosť (nemusí teda kontrolovať správnosť vstupu). Na obrázku vpravo je vstup pre $m = 4$, $n = 5$.

```
0 1 1 0 0
1 1 1 1 0
1 1 1 1 1
1 1 1 1 1
```

Výstup požadujeme v tvare matice čísel, kde nula označuje štvorček, ktorý nepatrí do miestnosti, štvorčeky patriace jednému tetraminu majú to isté číslo a susedné štvorčeky patriace rôznym tetraminám majú navzájom rôzne čísla.

```
0 2 2 0 0
3 2 2 1 0
3 1 1 1 2
3 3 2 2 2
```

732. Reverzný poľský zápis

Majme aritmetický výraz $A*B+C*D$. V reverznej poľskej notácií (RPN) tento výraz vyzerá $AB*CD*+$. Vidíte, že na rozdiel od normálneho zápisu sa v RPN zapisujú najprv operandy a za nimi príslušná operácia:

operandy : A, B	operácia : $*$
operandy : C, D	operácia : $*$
operandy : $(A\ B\ *)$, $(C\ D\ *)$	operácia : $+$

Výhodou tohto zápisu je napríklad presné určenie poradia vykonávania operácií.

Úloha: Napíšte program, ktorý načíta normálny aritmetický výraz obsahujúci operácie $+$, $-$, $*$, $/$, $\hat{}$ (so zvyčajnou prioritou, mínus môže byť aj unárne), zátvorky, jednopísmenové premenné $A, B, \ldots, Z$ a celočíselné konštanty a prepíše ho do RPN[7] (pre jednoznačnosť v RPN budeme unárne mínus značiť znakom @).

[7] RPN je odvodená od poľskej notácie (operátor je tu pred operandami), ktorú vymyslel poľský matematik Jan Łukasiewicz (1878–1956). Na čo je to dobré? Keď máme aritmetický výraz zapísaný v RPN, vieme ho jednoducho vypočítať na „kalkulačke so zásobníkom“. Postupujeme tak, že čítame zápis zľava doprava. Ak nájdeme operand, dáme jeho hodnotu na vrch zásobníka. Ak nájdeme operátor, spravíme príslušnú operáciu nad dvoma vrchnými prvkami zásobníka. Tie zo zásobníka vyberieme a namiesto nich vložíme výsledok operácie. Pri unárnom mínus postupujeme podobne, len vyberáme jeden operand. Keď takto spracujeme celý výraz, zostane nám v zásobníku výsledok. Niektoré počítače (a aj kalkulačky Hewlett-Packard) majú možnosť počítať výrazy uvedeným spôsobom

Príklad:

Vstup:	Výstup:
$(-D*(A+B)\,\hat{}\,D)*D+T$	$D\ @\ A\ B\ +\ D\ \hat{}\ *\ D\ *\ T\ +$
$17*(D*D)*D*D$	$17\ D\ D\ *\ *\ D\ *\ D\ *$
$A+B+C+D+E$	$A\ B\ +\ C\ +\ D\ +\ E\ +$
$A/B/C/D$	$A\ B\ /\ C\ /\ D\ /$
$A-A*22-B-C*D$	$A\ A\ 22\ *\ -\ B\ -\ C\ D\ *\ -$

Predpokladajte, že vstupný výraz je korektný. Všetky operácie okrem $\hat{}$ sú zľava asociatívne, teda napr. $A/B/C/D = (((A/B)/C)/D)$, ale $A\,\hat{}\,B\,\hat{}\,C = A\,\hat{}\,(B\,\hat{}\,C)$.

733. O telefónnych maniakoch

V jednej krajine je n telefónnych maniakov. Pre jednoduchosť predpokladajme, že majú telefónne čísla $1, 2, \ldots, n$. Každý maniak má telefónny zoznam, obsahujúci telefónne čísla jeho známych (a jeho vlastné), pričom neustále telefonuje všetkým maniakom, ktorých telefónne čísla na tomto zozname má (sebe samozrejme netelefonuje). Aby mohli maniaci spoznávať aj ďalších kolegov, oznamujú si navzájom pri každom telefonáte všetky telefónne čísla, ktoré majú na svojich zoznamoch, čím sa ich zoznamy rozrastajú. Po istom čase nastane situácia, že už žiaden maniak nemôže spoznať žiadneho nového maniaka.

Úloha: Napíšte program, ktorý pre daný počet maniakov $n > 0$ a pôvodné telefónne zoznamy maniakov vypíše všetky skupiny maniakov, ktorí na konci budú navzájom poznať svoje telefónne čísla (každú skupinu vypíše len raz). Napríklad pre $n = 5$ a telefónne zoznamy

číslo telefónneho maniaka	1	2	3	4	5
telefónny zoznam maniaka	$1, 2, 5$	$2, 5$	$3, 5$	4	$1, 5$

je 1. skupina $\{1, 2, 3, 5\}$ a 2. skupina $\{4\}$.

734. O cólštoku

Cólštok je programátorské náradie veľmi podobné skladaciemu stolárskemu metru, ale je celý rovnomerne popísaný znakmi a spĺňa nasledovné podmienky:

1. Má $n \geq 3$ zlomov, pričom zlomy môžu byť buď všetky na jednom znaku, alebo všetky medzi dvoma znakmi (pozri príklady).
2. Dĺžka každého úseku medzi dvoma zlomami je rovnaká (pritom aj začiatok a koniec cólštoku považujeme za zlom).
3. Pri tzv. „harmonikovom zložení“ sú všetky na sebe ležiace znaky rovnaké.

Príklad cólštokov:

abccbaabc, zložený

```
a   aa
 b  b  b
  cc   c
```

iný cólštok: *abcbabc*, zložený

```
  a  a
 b b b
c   c
```

Úloha: Napíšte program, ktorý z danej skupiny znakov vyberie a vypíše najdlhší reťazec znakov, ktorým sa dá popísať cólštok. Na cólštoku môžu byť teda použité len tie znaky, ktoré sa nachádzajú v danej skupine a najviac toľkokrát, koľkokrát sa nachádzajú v danej skupine.

Príklad: Pre skupinu znakov a, a, b, b, b, c, c, d je to cólštok *abcbcba*.

Možno ho zložiť nasledovne:

```
 a   a
  b b
   c
  b
```

---

a aj preto sa poľský zápis používa v prekladačoch programovacích jazykov ako medzikód (záujemcom odporúčame 5. kapitolu z knižky [46]).

735. O tučnej úsečke a bystrozrakých bodoch

V rovine máme daných n bystrozrakých bodov a jednu tučnú úsečku, cez ktorú nevidno.

Úloha: Napíšte program, ktorý načíta číslo $n > 0$, n bodov $[x_i, y_i]$ a koncové body $[a_x, a_y]$, $[b_x, b_y]$ tučnej úsečky a vypíše počet dvojíc tých bodov, ktoré na seba „nevidia“ cez tučnú úsečku (t.j. ich spojnica prechádza tučnou úsečkou).

Snažte sa nepoužiť žiadnu goniometrickú funkciu (sin, cos, tg, arctg, atď.).

811. O výbere

Napíšte program, ktorý načíta pole reálnych čísel A dĺžky n, reálne číslo t a prirodzené číslo k, $0 < k \leq n$ a zistí, či je možné vybrať k prvkov poľa A (každý prvok možno vybrať iba raz, t.j. indexy vybraných prvkov musia byť rôzne) tak, aby ich súčet bol menší než t (výstupom programu je odpoveď `ANO` alebo `NIE`).

Snažte sa, aby ste nemenili pole A a nepoužívali iné polia, smerníky a pod.

812. O postupnosti II

Sú dané prirodzené čísla m, n, $m \nmid n$ a $n \nmid m$. Napíšte program, ktorý vygeneruje a vypíše najdlhšiu postupnosť celých čísel, v ktorej je súčet každých m po sebe idúcich členov záporný a každých n po sebe idúcich členov kladný (postupnosť môže byť aj prázdna).

Keďže takýchto postupností môže byť nekonečne veľa, vypíšte vždy iba jednu.

Príklad: Pre $m = 7$ a $n = 11$ má hľadaná postupnosť dĺžku 16 a jedna z nich je

$$5, 5, -13, 5, 5, 5, -13, 5, 5, -13, 5, 5, 5, -13, 5, 5.$$

813. O zaostalej krajine

Na mape zaostalej krajiny je n miest očíslovaných $1, 2, \ldots, n$ so súradnicami (x_1, y_1), (x_2, y_2), …, (x_n, y_n). Vládca sa práve rozhodol svoju krajinu elektrifikovať. V meste č. 1 nechal postaviť elektráreň a teraz ešte potrebuje vybudovať elektrickú sieť. Pozval si preto inžinierov z celej krajiny a pod trestom smrti im nariadil postaviť túto sieť za minimálne prostriedky. Inžinieri sa ocitli v nezávideniahodnej situácii a potrebujú vašu pomoc, lebo je im jasné iba to, že čím je vedenie kratšie, tým je aj lacnejšie.

Úloha: Napíšte program, ktorý načíta počet miest n, ich súradnice na mape (x_i, y_i), kde $1 \leq i \leq n$ a vypíše všetky také dvojice miest, ktoré treba spojiť elektrickým vedením, aby malo každé mesto elektrinu a dĺžka vedenia bola čo najkratšia.

814. O armáde

Jedna armáda sa chystá na veľký husí pochod (t.j. v zástupe). V armádnych predpisoch však stojí: „… pri ľubovoľnej bojovej činnosti musí byť nadriadený schopný jediným pohľadom prekontrolovať všetkých svojich podriadených. Toto pravidlo musia obzvlášť prísne dodržiavať všetci členovia pochodového útvaru (ďalej ČPU)…“. Preto musia pri pochode všetci vojaci v zástupe zachovávať také poradie, aby mal každý nadriadený v ČPU všetkých svojich podriadených pred sebou.

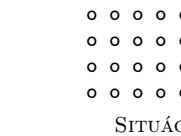
Úloha: Na pochod sa chystá n vojakov očíslovaných $1, 2, \ldots, n$. Kto je komu nadriadený a podriadený určujú armádne predpisy vymenovaním k dvojíc (nadriadený$_i$, podriadený$_i$), kde $1 \leq i \leq k$. Napíšte program, ktorý vypíše poradie, v akom sa majú vojaci zaradiť do zástupu, aby dodržali vyššie uvedené predpisy. Ak to nie je možné, program to musí zistiť a ohlásiť.

Príklad: Pre $n = 6$ a $k = 5$, $\begin{matrix}\text{nadriadený} & 1 & 4 & 2 & 1 & 2 \\ \text{podriadený} & 5 & 5 & 3 & 3 & 4\end{matrix}$;

je výsledok 1 2 4 6 3 5 alebo 6 2 1 4 5 3. Stačí jedno riešenie.

815. O hre BOXES

Hra BOXES sa hrá na modrom papieri, na ktorom je 5×6 bodov usporiadaných do tvaru obdĺžnika (pozri obrázok). Hru hrajú dvaja hráči. V každom ťahu musí hráč spojiť dva dosiaľ nespojené susedné body horizontálnou alebo vertikálnou čiarou svojej farby. (Jeden hráč používa na tento účel čiernu a druhý bielu farbu.) Keď týmto ťahom utvorí BOX, t.j. štvorec (so stranou dĺžky 1), v ktorom sú všetky spojenia bodov jeho farby, ťahá ešte raz. Hra končí, keď sa už nedá vytvoriť žiaden BOX. Vyhráva hráč, ktorý urobil viacej BOXov. Situácia hry BOXES sa dá reprezentovať nasledujúcim spôsobom:



Situácia:	Reprezentácia:	Vstup pre počítač:
+ čierny – biely	C čierny B biely	
o---o+++o o+++o o	B C 0 C 0	BC0C0
| + | |	B C B 0 B 0	BCB0B0
o o o+++o---o o	0 0 C B 0	00CB0
| + | | |	B C B B B 0	BCBBB0
o o+++o o---o o	0 C 0 B 0	0C0B0
	0 0 0 0 0 0	000000
o o o o o+++o	0 0 0 0 C	0000C
+ +	0 0 0 0 C C	0000CC
o o o o o+++o	0 0 0 0 C	0000C

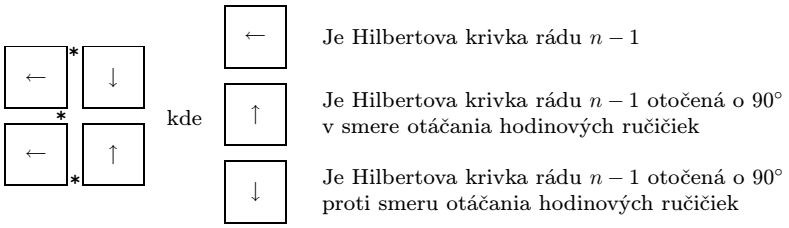
Úloha: Napíšte program, ktorý načíta reprezentáciu určitej hernej situácie a rozhodne, či hra ešte pokračuje, alebo či už skončila a ktorý hráč vyhral (prípadne oznámi remízu).

821. O Hilbertovej krivke

Hilbertova[8] *krivka* rádu $n > 1$ je spojitá lomená čiara, ktorá je definovaná spojením štyroch otočení Hilbertových kriviek rádu $n-1$ pomocou troch čiar, pričom Hilbertova krivka prvého rádu je na obrázku vpravo.

```
* * *
*
* * *
```

Všeobecne má Hilbertova krivka rádu n tvar:



a znak '*' predstavuje čiary spojenia.

```
* * * * *   *
*         *   *
* * *   * * *
    *
* * *   * * *
*         *   *
* * * * *   *
```

Skúste si nakresliť Hilbertovu krivku druhého rádu, mali by ste dostať krivku ako na obrázku vľavo.

Všimnite si, že Hilbertova krivka rádu n sa vždy zmestí do štvorčekovej siete s rozmermi $(2^{n+1}-1) \times (2^{n+1}-1)$ políčok (na políčku môže byť hviezdička alebo medzera). Túto sieť nazvime H-sieť a definujme v nej súradnice políčok tak, že políčko v riadku x a v stĺpci y má súradnice $[x, y]$ (ľavý horný roh H-siete má súradnice $[1, 1]$).

Úloha: Je daný rád Hilbertovej krivky n a súradnice dvoch políčok v H-sieti $[p_x, p_y]$, $[q_x, q_y]$. Tieto políčka v H-sieti určujú obdĺžnik a to tak, že $[p_x, p_y]$ je jeho ľavým horným

[8] David Hilbert (1862–1943), nemecký matematik

a $[q_x, q_y]$ jeho pravým dolným políčkom. Napíšte program, ktorý zobrazí časť Hilbertovej krivky, ktorá sa nachádza v určenom obdĺžniku.

PRÍKLAD:

VSTUP:
$n = 2$
$[p_x, p_y] = [2, 2]$
$[q_x, q_y] = [5, 6]$

VÝSTUP:
```
+- - - - -+
!       *  !
!* *   * *!
!  *       !
!* *   * *!
+- - - - -+
```

822. O polrovine

V rovine je daných n bodov so súradnicami $[x_i, y_i]$, ktoré ležia na nejakej kružnici so stredom v bode $[0, 0]$.

ÚLOHA: Napíšte program, ktorý načíta n, x_i, y_i, pre $1 \leq i \leq n$ a zistí, či je možné zostrojiť v rovine priamku prechádzajúcu bodom $[0, 0]$ takú, že všetky dané body $[x_i, y_i]$ ležia v jednej, ňou určenej polrovine (t.j. priamka nerozdeľuje body na dve skupiny; všetky ležia „na jednej strane“).

823. O zásobovaní

V krajine je n miest očíslovaných $1, 2, \ldots, n$, medzi ktorými je vybudovaná cestná sieť tak, že z každého mesta sa dá dostať do ľubovoľného iného (nemusí sa to však dať priamym spojením, t.j. cestou, ktorá neprechádza cez žiadne mesto). V krajine sa chystajú vybudovať centrálny sklad potravín, z ktorého by každé ráno vyštartovalo $n - 1$ zásobovacích áut naložených potravinami do ostatných miest (každé do iného) a do ktorého by sa každý večer zásobovacie autá vrátili. Každá doprava však niečo stojí. O tejto predpokladáme, že je tým drahšia, čím dlhšiu cestu zásobovacie autá denne vykonajú.

ÚLOHA: Napíšte program, ktorý načíta počet miest n, maticu $A[1\,..\,n, 1\,..\,n]$, kde $A[i, j]$ $= 0$, ak medzi mestami neexistuje priame spojenie, alebo $A[i, j] =$ [dĺžka priameho cestného spojenia medzi mestami i a j], ak také spojenie existuje. Výstupom je číslo mesta, v ktorom treba vybudovať centrálny sklad, aby bolo zásobovanie krajiny čo najlacnejšie (t.j. aby bol minimálny súčet dĺžok ciest vykonaných zásobovacími autami).

Pozor! Táto úloha nie je ani náhodou podobná úlohe 813 o zaostalej krajine!

824. O balíčkoch

V podniku KRACH vyrábajú výrobky s hmotnosťami 1, 2, 3, 4, 5, 6, 7 a 8 kg. Od odberateľov prijímajú objednávky v tvare:

hmotnosť	1 kg	2 kg	3 kg	4 kg	5 kg	6 kg	7 kg	8 kg
počet kusov	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8

a poštou im posielajú balíky s požadovaným počtom výrobkov. Pošta však doručuje len tie balíky, ktorých hmotnosť nepresahuje 8 kg. Preto si podnik KRACH u vás objednal program, ktorý načíta objednávku $p[1\,..\,8]$ a navrhne, akým spôsobom treba výrobky baliť do balíkov, aby bol počet balíkov najmenší možný.

PRÍKLAD: Pre $p = (1, 2, 2, 0, 1, 0, 2, 0)$ je jedno z možných riešení dať do 1. balíka 3 2 2; do 2. balíka 7; do 3. balíka 5 3; do 4. balíka 1 7. Celkovo sú potrebné 4 balíky.

825. Druhá úloha o hre BOXES

Spomeňte si na úlohu 815 o hre BOXES. Iste ste sa pri jeho riešení potešili, že sa od vás nežiada kontrola korektnosti danej hernej situácie.

ÚLOHA: Napíšte program, ktorý načíta reprezentáciu určitej hernej situácie v hre BOXES. Zistí, či je korektná a ak áno, tak vypíše jeden z možných priebehov hry. Spôsob,

akým sa priebeh hry zaznamená, si môžete ľubovoľne zvoliť, len dbajte na to, aby bol prehľadný a stručný.

831. O katastrofe vo východnej krajine

Vo východnej krajine je n miest očíslovaných $1, 2, \ldots, n$. Medzi všetkými mestami navzájom existovali spojenia a to buď priame, alebo nepriame (nepriame spojenie je spojenie dvoch miest cez konečný počet miest). Jedného pekného dňa sa východnou krajinou súčasne prehnali uragán, zemetrasenie, povodeň, hladomor a revolúcia, čo zapríčinilo, že sa krajina rozpadla na niekoľko izolovaných častí, medzi ktorými nie je žiadne spojenie. Zaujímavé by bolo zistiť, na koľko častí sa krajina rozpadla.

Úloha: Je daný počet miest n východnej krajiny a m dvojíc miest (x_i, y_i), kde $1 \leq i \leq m$ a $1 \leq x_i, y_i \leq n$, medzi ktorými sa po katastrofe zachovalo priame spojenie (ak sa zachovalo priame spojenie mesta x s mestom y, zachovalo sa aj priame spojenie mesta y s x). Napíšte program, ktorý pre dané údaje zistí, na koľko izolovaných častí sa východná krajina rozpadla. Za izolovanú časť považujte takú skupinu miest, že medzi každými dvoma mestami z tejto skupiny existuje spojenie (t.j. existujú mestá, cez ktoré sa môžu spojiť) a žiadne mesto z tejto skupiny nemá spojenie s mestom, ktoré nepatrí do tejto skupiny.

832. O dierach v doske

V ústave informatiky majú drevenú dosku zanedbateľnej hrúbky obdĺžnikového tvaru rozmerov $n \times m$. V doske je vyvŕtaných k bodových dier (t.j. dier zanedbateľnej veľkosti) so súradnicami $[x_i, y_i]$, kde $1 \leq i \leq k$ (bod $[0, 0]$ je umiestnený v ľavom dolnom rohu dosky, spodný okraj dosky určuje os x a ľavý okraj dosky určuje os y).

Úloha: Napíšte program, ktorý pre danú situáciu zistí rozmery maximálneho (čo do obsahu) obdĺžnika bez dier, ktorý sa dá z dosky vypíliť tak, že sa píli len rovnobežne s okrajmi dosky (t.j. v smere osí).

833. O spore v parlamente

Dve iniciatívne skupiny poslancov predložili v parlamente dva návrhy zákona. Prvá skupina predložila návrh A dĺžky n (v programátorskom ponímaní je návrh zákona jednorozmerné pole znakov danej dĺžky) a druhá skupina návrh B dĺžky m. Aby sa poslanci nehádali, dohodli sa, že z obidvoch návrhov vyškrtajú určité časti tak, aby boli návrhy po vyškrtaní zhodné. Samozrejme im záleží na tom, aby škrtali čo najmenej.

Úloha: Sú dané čísla n, m a polia znakov $A[1\,..\,n]$, $B[1\,..\,m]$. Napíšte program, ktorý urobí návrh, ktoré znaky treba z polí vyškrtnúť tak, aby boli polia znakov po vyškrtaní totožné. Snažte sa, aby bolo škrtaných znakov čo najmenej. Výstup môžete urobiť napríklad takto: budete predpokladať, že A a B neobsahujú zátvorky () a dáte do zátvoriek tie časti A a B, ktoré treba vyškrtnúť.

Príklad:

Vstup: A = 'moricko pisal basne v siestich rokoch',
B = 'horac sipel krasne v sestnastich rokoch'

vystup: A = `(m)or(i)c(ko) (p)i(sa)l (b)asne v s(i)estich rokoch`
B = `(h)or(a)c (s)i(pe)l (kr)asne v sest(nast)ich rokoch`;
`pocet skrtani: 20`

834. O vzbure v Hanibalovom tábore

Keď sa v roku 217 p.n.l. rozhodol kartáginský veliteľ Hanibal tiahnuť na Rím pochodom cez Alpy, jeho armáda sa vzbúrila a chcela iného veliteľa. Po krátkej hádke sa vojaci zhodli, že vojenskú prísahu zložia šľachticovi Hasdrubalovi a chystali sa k nemu vyslať skupinu poslov, v ktorej mal byť z každého práporu práve jeden vojak. Hanibal bol však bystrý človek. Podplatil otroka Sikima, aby presvedčil vzbúrených vojakov, že k Hasdrubalovi nemožno vyslať len tak hocijakých poslov. Keď vojaci začali rozmýšľať, aký lesk by svojmu

posolstvu dodali, Sikim ich nahovoril, aby vybrali takých poslov, ktorí budú mať navzájom rôzne mená. Kartáginci mali totiž početnú armádu s veľa prápormi, no ich kalendár mal málo mien. Vojaci sa dlhé hodiny snažili zostaviť takéto posolstvo a keď sa im to nepodarilo, presvedčení, že je to zákrok nebies, vrátili sa s prosbou o odpustenie k Hanibalovi.

Úloha: Napíšte program, ktorý načíta číslo n, prvky k množín $m_1, m_2, \ldots, m_k$, ktoré obsahujú len prvky $1, 2, \ldots, n$ (t.j. m_i je podmnožina množiny $\{1, 2, \ldots, n\}$) a zistí, či je možné z každej množiny vybrať jeden prvok tak, aby vybrané prvky boli navzájom rôzne. V prípade, že to možné je, program musí vypísať aj výber prvkov.

835. O svetovej vojne

Na planéte Z sú štáty $1, 2, \ldots, n$. Zvykom na tejto planéte je, že sa uzatvárajú vojenské dohody výlučne typu: „Ak štát x napadne štát y, potom my, štát u, napadneme štát v“.

Úloha: Napíšte program, ktorý načíta n – počet štátov, k vojenských dohôd (t.j. štvoríc (x_i, y_i, u_i, v_i), kde $1 \leq i \leq k$) a zistí, či existujú také dva štáty g a h, že ak g napadne h, tak na planéte Z vypukne svetová vojna (svetová vojna je stav, keď bojujú všetky štáty, t.j. keď každý štát niekoho napadol, alebo bol niekým napadnutý). V prípade, že také g a h existujú, program ich vypíše a okrem toho vypíše poradie dvojíc (útočník, obranca), v ktorom sa štáty budú napádať.

841. O orientačnom behu

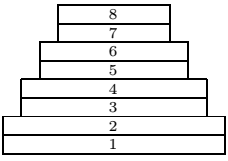
V okolí Bratislavy sa každoročne organizuje niekoľko orientačných behov. Pri orientačnom behu dostanú účastníci pred štartom poradie stanovíšť, ktoré organizátori predtým v teréne vyznačili a ich úlohou je prebehnúť cez všetky stanovištia v danom poradí. Pri pretekoch sa zvyčajne beží viac rôzne dlhých tratí, pričom trať a jej dĺžku organizátori určujú poradím a počtom stanovíšť, cez ktoré majú bežci prebehnúť. Za dĺžku trate sa považuje súčet priamych vzdialeností medzi stanovišťami, takže v skutočnosti bežci prebehnú trošku dlhšiu vzdialenosť (samozrejme za predpokladu, že nezablúdia). Na trať sa kladie požiadavka, že cez každé stanovište môže prechádzať len raz, s výnimkou štartovacieho a cieľového stanovišťa – tie môžu byť totožné. (To však platí len pre trať a nie pre bežca. Ten môže prechádzať jedným stanovišťom aj viackrát, napríklad keď zablúdi, alebo keď sa jedno stanovište nachádza na ceste medzi dvoma inými.) Inak sa na trať nekladú žiadne požiadavky, takže sa môže napríklad aj „križovať“.

Úloha: Napíšte program, ktorý načíta počet stanovíšť n, súradnice stanovíšť $[x_i, y_i]$ pre $1 \leq i \leq n$ (teda akúsi relatívnu zemepisnú dĺžku a zemepisnú šírku stanovíšť) a navrhne (t.j. určí poradie stanovíšť) najdlhšiu trať, akú možno pre dané stanovištia navrhnúť. (Výškové rozdiely sa pri výpočte vzdialenosti dvoch stanovíšť zanedbávajú.)

Príklad: Pre $n = 4$ a stanovištia so súradnicami $[x, y] = [0, 0], [2, 0], [10, 1], [10, -1]$ je výsledok $1, 3, 2, 4, 1$.

842. O Hanojských dvojvežiach

V budhistickom kláštore v Hanoji majú tri veľké stĺpy. Na prvom stĺpe sú nastoknuté ťažké kamenné disky s otvorom uprostred. Tieto disky sú rôznych veľkostí (hmotností; u kameňa so stúpajúcim objemom rastie aj hmotnosť), pričom z každého druhu sú na stĺpe práve dva disky. Navyše sú tieto disky nastoknuté v takom poradí, aby bol vždy menší nad väčším, prípadne rovnako veľkým (lebo menší by sa pod väčším rozdrvil) a sú očíslované od najspodnejšieho (1) po najvrchnejší ($2n$, kde n je počet druhov diskov).



Príklad pre $N = 4$

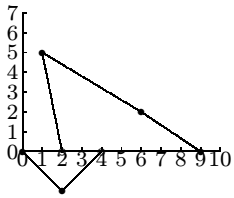
Mnísi v tomto kláštore volajú takéto usporiadanie diskov Hanojská dvojveža a veria, že keď sa im podarí preložiť disky z prvého stĺpu na tretí, a to v takom poradí ako boli pôvodne na prvom stĺpe, nastane koniec sveta. Našťastie pre náš svet to podľa svojho náboženstva nemôžu robiť hocijako, ale vždy

môžu preložiť len jeden disk z niektorého stĺpu na iný (takže na preloženie musia nutne použiť druhý stĺp). Pri prekladaní sú ďalej obmedzení tým, že nemôžu klásť väčší disk na menší, lebo menší by sa pod väčším rozdrvil (klásť na seba rovnako veľké disky je však možné). Preto sa mnísi už mnohé roky snažia vypočítať, ako majú disky poprekladať, aby im to vyšlo[9].

Úloha: Napíšte program, ktorý načíta počet druhov diskov n a vygeneruje postupnosť preložení tvaru „zo stĺpa č. x na stĺp číslo y prenes disk číslo z“, po vykonaní ktorej by sme preložili dvojvežu z prvého stĺpu na tretí. Snažte sa, aby bol počet preložení minimálny a pokúste sa určiť vzorec pre počet preložení, ktoré navrhne váš program v závislosti od n.

843. O mnohouholníku

Máme dané n a ľubovoľný n-uholník, ktorý je daný vymenovaním súradníc svojich vrcholov tak, ako po sebe nasledujú proti smeru chodu hodinových ručičiek počnúc vrcholom s najmenšou x-ovou súradnicou.



Úloha: Napíšte program, ktorý z vrcholov daného (aj nekonvexného) n-uholníka vyberie tie vrcholy, ktoré tvoria konvexný mnohouholník s tou vlastnosťou, že obsahuje („pokrýva“) daný nekonvexný n-uholník. Výstupom programu je postupnosť poradí vybratých vrcholov z daného nekonvexného n-uholníka (táto postupnosť je rastúca).

Príklad: Pre $n = 7$ a vrcholy $[0,0]$, $[2,-2]$, $[4,0]$, $[9,0]$, $[6,2]$, $[1,5]$, $[2,0]$ je výsledok 1, 2, 4, 5, 6.

844. O prekladateľovi

Napíšte program, ktorý načíta reťazec znakov, ktorý zodpovedá korektnému zápisu čísla od nuly po 999 999 v slovenskom alebo anglickom jazyku. Rozpozná, v ktorom jazyku je dané číslo zapísané a urobí preklad čísla do druhého jazyka. Predtým než začnete robiť samotný program, nezabudnite si presne definovať, aký reťazec pokladáte za korektný zápis čísla v anglickom a slovenskom jazyku.

Pokiaľ vám angličtina nevyhovuje, môžete použiť iný svetový jazyk (ruštinu však neodporúčame, iba ak by ste mali počítač s kódom aj pre azbuku).

Príklad:

Vstup:
```
Two hundred and fourteen thousand five hundred and twenty-seven
nula
```
Výstup:
```
Dvestostrnast tisic patsto dvadsatsedem
zero
```

S medzerami a pomlčkami si hlavu netrápte, iba ak by vás úloha zaujala a ak by sa vám chcelo otravovať vašich slovenčinárov a angličtinárov. Takisto nemusíte uvažovať mäkčene, ale ak ste maniaci, môžete sa o to pokúsiť.

845. Miškova Super Úloha

S touto úlohou sa môžete dennodenne stretnúť v továrenskej výrobe, takže je vám už asi jasné, že pôjde o ťažkú a zaujímavú úlohu. Keď sa vyrába nejaký výrobok plošného tvaru (napríklad podošva, rôzne výrezy z plechu), väčšinou sa vyrezáva z obdĺžnikového kusu plošného materiálu (koža, plech). Samozrejme, ak chceme ušetriť, musí byť obdĺžnikový kus materiálu čo najmenší. Zovšeobecnenie a zjednodušenie vedie k nasledujúcej úlohe:

Úloha: Je dané n a konvexný n-uholník, ktorý je zadaný vymenovaním súradníc vrcholov tak, ako po sebe nasledujú proti smeru chodu hodinových ručičiek počnúc vrcholom

[9] Pôvodný hlavolam vymyslel francúzsky matematik Édouard Lucas v roku 1883. Opriadol ho príbehom o Brahmovej veži, ktorá mala 64 rôzne veľkých diskov z rýdzeho zlata. Disky sa navliekali na tri diamantové ihlice. Na počiatku ich Boh umiestnil na prvú ihlicu a skupina mníchov ich má presunúť na tretiu, podľa rovnakých pravidiel ako v tejto úlohe. Keď sa im to podarí, svet zanikne. Pozri [31, str. 55–59].

s najmenšou x-ovou súradnicou (teda tak, ako v úlohe 843). Napíšte program, ktorý vypočíta súradnice štyroch vrcholov obdĺžnika s čo najmenším obsahom, ktorý obsahuje daný n-uholník.

911. O Mersennových[10] číslach

Napíšte program, ktorý pre dané prirodzené čísla m a n, $m \leq n$, nájde n nezáporných celých čísel $k_1, k_2, \ldots, k_n$ (t.j. jedno riešenie) také, že číslo $2^{k_1} + 2^{k_2} + \cdots + 2^{k_n}$ bude deliteľné m-tým Mersennovým číslom (t.j. číslom $2^m - 1$).

Príklad: Pre $m = n = 3$ môže byť výsledok napríklad: 3, 1, 2, lebo $2^3 - 1 = 7$ delí číslo $2^3 + 2^1 + 2^2 = 14$.

V prípade, že sa pre určité m a n úloha nedá riešiť, program o tom musí vypísať správu.

912. O výstavbe mosta

Vládca súostrovia Kiribati sa chcel zapáčiť svojmu ľudu a sľúbil mu postaviť most. Teraz musí svoj sľub splniť, ale chcel by, aby ho to stálo čo najmenej peňazí. Keďže morské dno medzi ostrovmi je na Kiribati približne rovné, množstvo peňazí potrebných na stavbu je priamo úmerné dĺžke mosta. O súostroví Kiribati sa vie, že žiaden ostrov neobsahuje zálivy (ostrovy sú „konvexné“ útvary) ani jazerá.

Úloha: V matici A rozmerov $m \times n$ núl a jednotiek je zaznamenaná mapa súostrovia Kiribati (0 je more, 1 je pevnina). Navrhnite program, ktorý vypíše túto mapu, pričom na nej vyznačí miesto, kde má vládca nechať postaviť sľúbený most. Musí ho to však stáť čo najmenej peňazí, t.j. dĺžka vyznačeného mosta musí byť najmenšia možná. Most na mape vyznačíte tak, že na mape prepíšete niektoré nuly (more) napríklad na dvojky (alebo na nejaké iné sympatické čísla predstavujúce most). Aby toto vyznačenie bolo mostom, musia byť splnené nasledujúce podmienky:

a) Každé políčko mapy predstavujúce most susedí buď (a to celou stranou, nie rohom!) s políčkom predstavujúcim ostrov (s jednotkou) a s jedným políčkom predstavujúcim most, alebo susedí s práve dvoma políčkami predstavujúcimi most, alebo susedí s dvoma políčkami predstavujúcimi rôzne ostrovy.
b) Most musí spájať dva rôzne ostrovy. (V prípade nejasnosti definície sa riaďte intuitívnou predstavou mosta.)

Dĺžkou takto vyznačeného mosta je počet políčok, ktoré ho predstavujú.

913. O koláči

Na obdĺžnikovom plechu upiekli (mimochodom nie veľmi dobrý) koláč rozmerov $m \times n$ (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 obdĺžnikového tvaru rozmerov x_i, y_i $(1 \leq i \leq k)$. Napíšte program, ktorý zistí, aké mohli byť rozmery plechu $m \times n$, na ktorom bol daný koláč upečený (stačí jedno riešenie). Všetky rozmery sú celočíselné.

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

914. O burundijskej abecede

Keď sa obyvatelia Burundi zbavili belgických kolonizátorov, boli už natoľko poznačení ich nadvládou, že od nich prevzali francúzsky jazyk a abecedu. V návale nacionalizmu a

[10] Marin Mersenne (1588–1648), francúzsky mních, matematik a filozof

protibelgických vášní sa rozhodli, že francúzsku abecedu nahradia vlastnou, burundijskou. Aby si pritom nenarobili veľa problémov, vytvorili svoju abecedu z francúzskej tak, že iba zmenili poradie jej písmen.

Úloha: Napíšte program, ktorý na vstupe prijíma k slov usporiadaných podľa burundijskej abecedy a vypíše jedno možné poradie písmen burundijskej abecedy (vypíše iba poradie písmen, ktoré sa nachádzali v slovách).

Príklad: Pre slová `gda, gab, qek, da` je jedno poradie písmen v abecede `g, q, d, a, k, b, e`.

915. O Froscale

Froscal je jazyk sprevádzajúci 41. ročník matematickej olympiády kategórie P. Bližší opis tohto jazyka je uvedený nižšie.

Úloha: Navrhnite prekladač tohto jazyka do Pascalu, t.j. program, ktorý načíta program vo Froscale a vytvorí k nemu príslušný ekvivalent v Pascale.

Samotný prekladač môže byť napísaný v ľubovoľnom prípustnom jazyku.

Riešiteľom, ktorí Pascal neovládajú: Urobiť prekladač Froscalu do Basicu alebo do C by bolo oveľa ťažšie ako pozrieť sa do múdrej knihy, napríklad [36], ako Pascal vyzerá. Ak by vás však štúdovanie Pascalu odradilo od riešenia tejto úlohy, ponechávame vám možnosť definovať si analogický jazyk, pracujúci s frontou, podobný Basicu alebo C (teda Frosic alebo Froc) a naprogramovať prekladač z tohto jazyka do Basicu alebo C.

Froscal – Frontový Pascal. Jazyk bez procedúr, premenných, syntaxou pripomínajúcou Pascal (aj so skokmi) a s jedinou pamäťovou štruktúrou – frontou – to je Froscal. Fronta je dátová štruktúra s premenlivou veľkosťou (nie nepodobná zásobníku), do ktorej sa na jeden koniec vkladá a vyberá sa z druhého. Presnejšie: Froscal je Pascal bez premenných (a teda aj bez priradenia), s **while** a **repeat** cyklami, príkazmi **case** a **if** a skokom **goto**. Má dve fiktívne premenné *INP* a *TOP* typu char a skrytú premennú – frontu znakov. *INP* je práve čítaný znak zo vstupu a *TOP* je znak na začiatku fronty. Ďalšie príkazy: *NEXT* načíta nový znak zo vstupu do fiktívnej premennej *INP*, *PUT*(znak) alebo *PUT*(premenná) uloží znak alebo hodnotu premennej *INP* alebo *TOP* na koniec fronty, *GET* vyhodí prvý prvok z fronty. Príkaz *write* má pascalovskú definíciu, t.j. môže sa vypísať znak, premenná, reťazec alebo postupnosť takýchto výrazov. Špeciálnym znakom na vstupe a vo fronte je znak `'?'`, ktorý na vstupe znamená presiahnutie vstupného slova a vo fronte znamená prázdny front. Ak $INP =$ `'?'` (alebo $TOP =$ `'?'`), potom príkaz *NEXT* (alebo *GET*) nespraví nič. Ani *PUT*(`'?'`) nič nevykoná.

Program pracuje tak, že na vstupe má slovo nad dopredu presne určenou abecedou (podmnožinou vypísateľných znakov) a na výstupe (*write*) môže vypísať riešenie problému. Testovať korektnosť vstupnej abecedy nie je nutné; ak je špecifikovaný formát vstupu, tak ho netreba testovať. Do fronty ukladať, testovať a vypisovať možno ľubovoľné vypísateľné znaky. Na začiatku práce programu je fronta prázdna (t.j. $TOP =$ `'?'`) a premenná *INP* obsahuje prvý znak zo vstupu.

Príklad: Program, ktorý má na vstupe slovo v tvare $w_1\#w_2$, kde w_1 a w_2 sú slová nad abecedou $\{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$. Treba otestovať, či sa $w_1 = w_2$.

```
program TEST;
label 13;
begin
  { frontu síce vyprázdniť netreba, ale na precvičenie... }
  while TOP <> '?' do GET;
  while not ((INP = '#') or (INP = '?')) do
    begin { ešte sme vo w1, treba uložiť }
      PUT(INP); NEXT
    end;
```

```
  if INP <> '#' then goto 13; { chýba '#'}
  NEXT;
  while ((INP <> '?') and (TOP = INP)) do
    begin { ďalší znak sa rovná, ideme ďalej }
      GET; NEXT
    end;
  if ((INP = TOP) and (INP = '?')) then write('ANO')
  else 13: write('NIE')
end.
```

Orientačná gramatika jazyka Froscal:

⟨program⟩ ::= **program** ⟨meno⟩**;** [⟨dekl. skokov⟩] ⟨telo programu⟩
⟨dekl. skokov⟩ ::= **label** ⟨skok⟩[**,** ⟨skok⟩]*;
⟨skok⟩ ::= ⟨celé číslo⟩
⟨telo programu⟩ ::= **begin** ⟨príkazy⟩ **end.**
⟨príkazy⟩ ::= ⟨nič⟩ | ⟨príkaz⟩[**;** ⟨príkazy⟩]
⟨príkaz⟩ ::= [⟨skok⟩**:**]* ⟨elem. príkaz⟩
⟨elem. príkaz⟩ ::= **begin** ⟨príkazy⟩ **end**
| **if** ⟨podm⟩ **then** ⟨príkaz⟩[**else** ⟨príkaz⟩]
| **case** ⟨výraz⟩ **of**
(⟨selektor⟩**:** ⟨príkaz⟩**;**)*
[**else** ⟨príkaz⟩]
end
| **while** ⟨podm⟩ **do** ⟨príkaz⟩
| **repeat** ⟨príkazy⟩ **until** ⟨podm⟩
| **goto** ⟨skok⟩
| *write*(⟨write výraz⟩[**,** ⟨write výraz⟩]*)
| *PUT*(⟨výraz⟩)
| *GET*
| *NEXT*
⟨podm⟩ ::= ⟨relácia⟩
| (⟨podm⟩) **and** (⟨podm⟩)
| (⟨podm⟩) **or** (⟨podm⟩)
| **not**(⟨podm⟩)
⟨relácia⟩ ::= ⟨výraz⟩⟨relačný znak⟩⟨výraz⟩
⟨relačný znak⟩ ::=
= | **<** | **<=** | **>** | **>=** | **<>** | **in**
⟨výraz⟩ ::= *INP* | *TOP* | ⟨znak. konšt.⟩ | ⟨interval znakov⟩ | ⟨množina znakov⟩
⟨write výraz⟩ ::= *INP* | *TOP* | ⟨znak. konšt.⟩ | ⟨reťazec⟩
⟨poznámka⟩ ::= **{** ⟨text bez '**}**'⟩ **}**

Poznámka môže byť kdekoľvek rovnako ako v Pascale.

921. O Kanárskych poliach

Je dané pole $A[0\,..\,n]$ celých čísel a celé číslo $k \geq 2$. Nech váš program rýchlo preusporiada toto pole tak, aby pre každé i platilo: $a_i \geq a_{k\cdot i+j}$ pre $1 \leq j \leq k$ (ak $a_{k\cdot i+j}$ existuje).

922. O kiribatských daniach

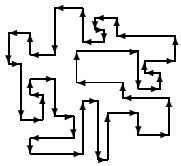
Vládca súostrovia Kiribati sa rozhodol vyrubiť nové dane z rybolovu. Keďže bol múdry a spravodlivý, daň pre jednotlivé ostrovy chcel stanoviť podľa veľkosti ich pobrežia (čím viac pobrežia, tým viac možností lovu rýb). Dal si preto pre každý ostrov súostrovia vyhotoviť na štvorčekový papier mapu, na ktorej zafarbené políčko predstavovalo pevninu a

nezafarbené políčko more. Veľkosť pobrežia ostrova stanovil vládca ako počet zafarbených políčok na jeho mape, ktoré sa aspoň jedným rohom dotýkajú nezafarbeného políčka. Daňovým úradníkom neostalo nič iné, ako sa pustiť do práce podľa vládcových pokynov. Pot sa im lial z čela a preklínali chvíľu, keď sa odmietli zúčastniť programátorského školenia za oceánom.

Úloha: Napíšte program, ktorý načíta rozmery mapy m (počet riadkov) a n (počet stĺpcov), mapu (jedného!!!) ostrova reprezentovanú maticou $A[1\,..\,m, 1\,..\,n]$ núl a jednotiek (nula znamená nezafarbené políčko, teda more, jednotka znamená zafarbené políčko – pevninu) a vypočíta veľkosť pobrežia daného ostrova (podľa vládcovej definície).

Ostrov, môže mať aj zálivy, ale nemôže mať vnútorné more. Políčka pevniny susedia stranou alebo rohom.

923. O opitom zlatokopovi



Na Vyšnej Klondike sa objavilo zlato. Masy nezamestnaných Slovákov sa hrnú na Vyšnú Klondiku vykolíkovať si svoju parcelku. Medzi prvými sa za vidinou ľahkého zbohatnutia vybral zlatokop Stano, zvaný Santo. Posilnený miestnou ohnivou vodou by na miesto ani nedošiel, nebyť jeho verného kamaráta Boba, zvaného Dlhý Banto. Na mieste Santo odborným okom (spod borovice) ohodnotil okolie a povelmi riadil triezveho Banta pri kolíkovaní (Doprava! Krok! Ešte krok! Doľava! Krok! Ešte krok! Ešte krok! atď.). Ťažkosti nastali až dolu v Dolnom Kelčove v registračnom stredisku, kde si chceli zaregistrovať svoju parcelu. Jedna z najdôležitejších koloniek znela: „Plocha parcely: .....“. Spomenúť si však vedeli (presnejšie Banto si spomenul) iba na postupnosť Santových príkazov. Santo vyhlásil, že zistiť z toho plochu parcely musí byť hračka už aj preto, že o Bantovi je známe, že jeho krok meria 1 meter.

V súčasnosti už niekoľko dní sedia v predizbe registračnej miestnosti a na sakramentsky malom (!) kúsku papiera sa snažia vypočítať plochu svojej parcely, kým ich niekto nepredbehne.

Úloha: Predbehnite ich a napíšte program, ktorý rýchlo (a v čo najmenšej pamäti) vypočíta plochu Santovej a Bantovej parcely. Na vstupe dostane postupnosť Santových príkazov, pre jednoduchosť zaznamenaných v tvare čísel:

1 – (otoč sa o 90°) Doľava!
2 – (otoč sa o 90°) Doprava!
3 n – Prejdi n krokov!

Napríklad postupnosť 3 1 2 1 1 3 2 1 3 1 1 3 2 zodpovedá postupnosti príkazov: Prejdi 1 krok! Doprava! Doľava! Doľava! Prejdi 2 kroky! Doľava! Prejdi 1 krok! Doľava! Prejdi 2 kroky! Zodpovedajúca parcela by mala tvar obdĺžnika 2×1 metrov.

Dôležité! Z ich rozhovoru sa zistilo, že Banto skončil kolíkovanie na tom istom mieste, kde začínal, a pritom nikdy neprekrížil svoju cestu.

924. O burundijskej železnici

Keď belgickí kolonizátori opustili Burundi, nezostal tam po nich kameň na kameni. Burundijčania začínali všetko budovať odznova a v prvom rade sa vrhli na výstavbu železničnej siete. Tá mala podľa ich predstáv spájať všetky významné dediny (na tomto mieste by bolo vhodné poznamenať, že burundijské dediny boli síce veľké, ale bezmenné skupiny palmových príbytkov, a preto ich stavitelia železnice museli očíslovať číslami od 1 po n). Lenže na koľajnice potrebovali kvalitné železo, a to bolo v Burundi veľmi vzácne. Zostavili preto maticu $A[1\,..\,n, 1\,..\,n]$, do ktorej na políčko $A[i, j]$ napísali, aké množstvo železa je potrebné na vybudovanie železničného spojenia z dediny s číslom i do dediny s číslom j (táto ich matica je symetrická). Ďalej si však už ani pri najlepšej vôli poradiť nevedeli, lebo belgickí kolonizátori ich nenaučili programovať. Železnicu nakoniec postavili

len medzi niektorými dedinami, v dôsledku čoho vznikla v Burundi nenávisť medzi dvoma najpočetnejšími kmeňmi a skončilo sa to krvavou občianskou vojnou, ktorá trvá podnes.

Úloha: Napíšte program, ktorý pre dané n a maticu A (s vyššie uvedeným významom) navrhne dvojice burundijských dedín, medzi ktorými treba vybudovať železničné spojenie tak, aby sa z každej dediny dalo železnicou dostať do ľubovoľnej inej a aby množstvo železa potrebného na stavbu bolo najmenšie.

925. Tretia úloha o hre BOXES

Janko a Marienka sa hrali známu hru BOXES. Hra BOXES sa hrá na papieri, na ktorom je $m \times n$ bodov usporiadaných do tvaru obdĺžnika

V každom ťahu hráči spájajú dva dosiaľ nespojené susedné body horizontálnou alebo vertikálnou čiarou. Pravidlá samotnej hry sú pre nás teraz nepodstatné (kto by ich chcel poznať, nech si pozrie úlohu 815).

Situácia:	Reprezentácia:	Vstup pre počítač:
o---o---o o---o o	1 1 0 1 0	'11010'
| | | | |	1 1 1 0 1 0	'111010'
o o o---o---o o	0 0 1 1 0	'00110'
| | | | |	1 1 1 1 1 0	'111110'
o o---o o---o---o	0 1 0 1 1	'01011'
| | |	0 0 0 1 1 1	'000111'
o o o o o---o	0 0 0 0 1	'00001'
| | |	0 0 0 1 1 1	'000111'
o o o o---o---o	0 0 0 1 1	'00011'

Takže Janko s Marienkou sa hrali túto hru a keď už Janka delili od víťazstva len dva ťahy, Marienka, vidiac, že ak niečo nevymyslí, prehrá, sa opýtala: „Janko a koľko sme my vlastne nakreslili obdĺžnikov? Tých je tak veľa, že ja by som to veru nezrátala.“ Janko chcel samozrejme ihneď porátať všetky obdĺžniky, a tak rátal, rátal, až obe deti zavolala ich mamička na obed. „A kto vlastne vyhral?“ opýtala sa mamička, keď sadli ku stolu. „Ale, nedohrali sme“, povedal cez slzy Janko a Marienka sa s chuťou pustila do fazuľovej polievky.

Úloha: Napíšte program, ktorý načíta reprezentáciu určitej hernej situácie hry BOXES a spočíta, koľko obdĺžnikov hráči nakreslili.

Príklad: Pre vyššie uvedenú situáciu by mal program zistiť, že hráči nakreslili osem obdĺžnikov.

931. O Santovej rúre

Santo a Banto úspešne vypočítali plochu svojej parcelky na Vyšnej Klondike a nič im už nebráni začať v ťažbe zlata. Ba predsa! Pri ryžovaní nutne potrebujú dostatok vody a Klondika je veľmi suchý kraj. Najbližší prameň sa nachádza dosť ďaleko od ich parcelky, a preto si v Dolnom Kelčove vybavili ešte aj povolenie na výstavbu podzemného vodovodného potrubia. Po úpornej práci v drsných podmienkach sa im podarilo vykopať výkop na každom mieste 1 meter hlboký a priamo vedúci k prameňu. Do tohto výkopu teraz potrebujú položiť rúru tak, aby po zasypaní výkopu netrčala nad povrch zeme. Klondika je však hornatá krajina, v dôsledku čoho sa rúra bude musieť občas rozrezať a zvariť, aby sa zalomila podľa stúpania alebo klesania terénu a nešla hlbšie ako 1 m pod zem a zároveň nevyšla nad povrch zeme.

Keďže Santo je od prírody lenivý (a rezanie a zváranie rúr je ťažká práca), prikázal Bantovi, aby najprv vypočítal minimálny počet miest, na ktorých treba rúru rozrezať a zvariť, aby vedel, čo ho čaká.

Banto prešiel popri výkope a starostlivo si zaznačil každé miesto, kde sa mení sklon terénu, ako i miesto, kde výkop začína a miesto, kde výkop končí. Tieto miesta si očísloval

číslami $1, 2, \ldots, n$ v takom poradí, v akom po sebe nasledujú, keď sa prechádza popri výkope od začiatku po koniec. Pre každé vyznačené miesto (očíslované číslom i) si tiež vypočítal y_i – výškový rozdiel s miestom, kde výkop začína, a taktiež x_i – vzdialenosť tohto vyznačeného miesta od začiatku výkopu. Teraz Banto smutne sedí a nevie, čo si počať.

Úloha: Pomôžte Bantovi! Napíšte preňho program, ktorý pre dané n, vzdialenosti $x_1, x_2, \ldots, x_n$ a výškové rozdiely $y_1, y_2, \ldots, y_n$ vypočíta minimálny počet zalomení Santovej a Bantovej rúry. Upresnenie: $y_1 = 0$, ostatné hodnoty y_i sú buď záporné (ak sa miesto i nachádza pod úrovňou miesta 1), kladné (ak sa miesto i nachádza nad úrovňou miesta 1), alebo rovné nule (ak sa miesto i nachádza na tej istej úrovni). Uvedomte si pritom, že na začiatku i na konci musí potrubie trčať zo zeme, teda počiatočný i koncový bod sú dané presne. Hrúbku rúry považujte za zanedbateľnú.

932. O kiribatskom pakovači

Vládca súostrovia Kiribati sa rozhodol vybudovať historický archív, ktorý by pretrval dlhé-predlhé veky. Okrem iného sa v tomto archíve mali uchovávať aj mapy súostrovia Kiribati, ako aj viacerých spriaznených súostroví. Týchto máp je však dosť veľa a navyše častá sopečná činnosť núti kartografov vytvárať stále novšie mapy. Preto po dohode so svojimi poradcami vydal nariadenie o konkurze na najlepší návrh, ako nejako zhustene uchovávať všetky tieto mapy.

Úloha: Mapa je reprezentovaná maticou obsahujúcou iba nuly a jednotky, pričom jednotiek je dosť málo (menej ako 15%). Navrhnite pakovač (a k nemu rozpakovač), ktorý danú maticu spakuje do vami navrhnutej zhustenej formy.

933. Telegram Bielym Légiám

PATRAME PO SPANIELOCH STOP BOLI V AMERIKE ZMIZLI BEZ STOPY STOP ASI U INDIANOV STOP INDIANI MAJU TENKE KOLY STOP MOZNO ICH UMUCILI STOP TYCH SPANIELOV STOP INDIANSKE HLIADKY SA SKLADAJU Z EN BOJOVNIKOV STOP KED HLIADKA CHYTI BIELU TVAR ZMERAJU JU A ZAKODUJU DO EN KRAT TRI MATICE STOP KAZDY BOJOVNIK ROBI JEDEN RIADOK A DAVA ZAJATCOVI KODY JEDNA DVA A TRI STOP TEDA MEDICINMAN KMENA DOSTANE S KAZDYM ZAJATCOM AJ TABULKU EN KRAT TRI V KTOREJ SU TIETO KODY STOP ON POTOM SPOCITA CI ZAJATCA BUDU MUCIT ALEBO HO PUSTIA LEBO NEMOZU VSETKYCH MUCIT LEBO MAJU TENKE KOLY STOP VIE SA LEN ZE MEDICINMAN SI K TABULKE CO MU PRINESIE HLIADKA PRILOZI ESTE JEDEN RIADOK V KTOROM JE V PROSTREDNOM TEDA DRUHOM STLPCI DIERA STOP VELKY MANITOU DOVOLIL ROBIT S TABULKOU LEN DVE VECI STOP MOZU ZROTOVAT HOCIKTORY RIADOK V LUBOVOLNOM SMERE STOP MOZU TIEZ POSUNUT DO DIERY CISLO KTORE JE NAD NOU ALEBO POD NOU STOP TYM SA SAMOZREJME POSUNIE AJ DIERA STOP AK SA DA DOSIAHNUT ZE V KAZDOM POVODNOM RIADKU SU CISLA JEDNA DVA A TRI V TOMTO PORADI A DIERA SA VRATI NA POVODNE MIESTO ZAJATCA PUSTIA LEBO BY IM ZLOMIL KOL STOP INAK MA SMOLU STOP CHYTILI SME INDIANSKU SPOJKU STOP MA SO SEBOU VSETKY TABULKY STOP POMOZTE NAM ZISTIT KOLKO SPANIELOV UMUCILI A KOLKI PREZILI STOP DIKY STOP

Úloha: Keď dostali Biele Légie tento telegram, hneď im bolo jasné, že tých tabuliek bude veľa a ručne to nezrátajú. Nevedia však programovať, preto nás požiadali, aby sme pre nich urobili program. Program má na vstupe maticu $A[1\,..\,n, 1\,..\,3]$, ktorá má v každom riadku čísla 1, 2, 3 (v ľubovoľnom poradí). Výstupom má byť reťazec `PREŽIL`, ak zajatec s touto tabuľkou prežil, alebo `GAME OVER`, ak ho Indiáni umučili. Naprogramujte tento postup.

934. O občianskej vojne v Burundi

V úlohe 924 sa dejiny Burundi skončili krvavou občianskou vojnou medzi dvoma najpočetnejšími kmeňmi. Jeden z nich, kmeň Zest, obýval západnú časť Burundi a vzýval bohov, ktorých symbolom je západ slnka. Druhý, kmeň Vast, obýval východnú časť krajiny a vzýval bohov, ktorých symbolom je východ slnka.

Medzi ich územiami sa tiahla zo severu na juh kľukatá hranica cez hustý prales, obývaný iba bojovníkmi toho-ktorého kmeňa. Po dlhých týždňoch krvavej vojny, v ktorej ani jedna strana nemohla získať prevahu nad druhou, vymyslel Veľký Šaman kmeňa Zest, aby sa najodvážnejší z odvážnych presunuli do takých dvoch zlomových bodov hranice, medzi ktorými leží iba nepriateľské územie, presekali sa priamo cez toto územie a po celej čiare, aby zapálili kúzlo. To ráno dymom zatieni vychádzajúce slnko a bojovníci kmeňa Vast sa na odrezanom území vzdajú v domnení, že ich ich bohovia opustili. Ako povedal, tak sa aj stalo. V noci najodvážnejší bojovníci presekali rovnú čiaru cez súperove územie, zapálili kúzlo a ráno na odrezanom území len zbierali zničených nepriateľov (a kruto ich mučili).

Ich akciu však sledoval Veľký Šaman kmeňa Vast a poobede im odpovedali podobnou akciou, ktorá odrezala kus územia kmeňa Zest a kúzlom im zatienila zapadajúce slnko. Odvtedy sa vojna zmenila na neskôr nazývanú „Dymovú vojnu“: vždy cez noc kmeň Zest odrezal jednu časť územia kmeňa Vast, načo cez deň kmeň Vast odrezal jednu časť územia kmeňa Zest. Po niekoľkých takýchto útokoch si šamani uvedomili, že nech ich bojovníci bojujú akokoľvek, výsledok vojny to neovplyvní. Preto začali smerovať svoje útoky tak, aby sa vojna čo najskôr skončila.

Úloha: Kmene sa striedajú v útokoch. Ak jeden z kmeňov nemôže útočiť, svoj útok vynechá a zaútočí opäť druhý kmeň. Vojna končí, keď už nemôže ani jeden kmeň útočiť.

a) Čo viedlo šamanov k tvrdeniu, že výsledok vojny nezmenia?
b) Nech polia $X[1\,..\,n]$ a $Y[1\,..\,n]$ (pričom $Y[i-1] < Y[i]$ pre každé $2 \leq i \leq n$) predstavujú súradnice hranice. Navrhnite postupnosť útokov v tvare „`Zest` i j, `Vast` k l, ...“ tak, aby vojna trvala čo najkratšie.

935. Zobali vrabce, zobali, konvexné obaly...

Ku každej množine bodov v rovine existuje jej konvexný obal, teda najmenší konvexný mnohouholník obsahujúci dané body. Záhradkársky sa konvexný obal získa tak, že sa dané body omotajú špagátom a body, okolo ktorých sa špagát napína, tvoria konvexný obal danej množiny bodov.

a) Máme dané dva disjunktné konvexné obaly dvoch množín bodov v rovine (t.j. ich konvexné obaly sa neprekrývajú). Oba sú dané vymenovaním súradníc svojich vrcholov v smere hodinových ručičiek. Napíšte čo najrýchlejší program, ktorý zistí konvexný obal zjednotenia týchto množín.
b) V poliach $X[1\,..\,n]$ a $Y[1\,..\,n]$ máme uložené súradnice n bodov v rovine (poradie ich uloženia je náhodné). S využitím riešenia úlohy a) zistite ich konvexný obal.

941. O tanečnom súbore Hatla-Patla

Tanečný súbor Hatla-Patla sa preslávil najmä tancom Patla, pri ktorom tanečníci nastúpia do radu a postupne polovica z nich predvedie zvláštny tanečný motív.

Počas tanca každý tanečník buď stojí a poskakuje na mieste obrátený tvárou k publiku, buď leží na nejakom inom tanečníkovi a tlieska, alebo leží na zemi pod nejakým iným tanečníkom a do rytmu búcha hlavou o zem. Na začiatku sú všetci tanečníci v prvej polohe, t.j. stoja a poskakujú. Potom začnú vybraní tanečníci predvádzať Patlu a ak sa tanec vydarí, všetci tanečníci ležia: prvá polovica na zemi a druhá polovica na tanečníkoch z prvej polovice. Patla spočíva v tom, že sa určený tanečník otočí k niektorému zo svojich susedov (podľa toho rozlišujeme Ľavú Patlu a Pravú Patlu), preskočí dvoch tanečníkov, nasledujúceho tanečníka zvalí na zem, ľahne si na neho a tlieska (pričom zvalený tanečník začne búchať hlavou o zem). Preskok môže mať dve podoby: preskakovaní tanečníci obaja ležia alebo stoja. Ak ležia, tak sa preskakujú celkom ľahko, ak stoja, musia obaja urobiť podrep a vykríknuť „Patla!“ (pričom sa tanečník pri skoku môže odraziť od ich chrbtov). Ak by tanečník nemohol urobiť ani jednu verziu týchto dvoch preskokov, alebo by po preskoku nemal koho zvaliť na zem, Patla by sa mu nepodarila a celý tanec by sa pokazil.

Úloha: Napíšte program, ktorý pre daný počet tanečníkov n vypíše, kedy má ktorý tanečník urobiť Ľavú či Pravú Patlu, aby sa tanec vydaril (prípadne vypíše, že sa to nedá). Predpokladajte, že tanečníci nastúpení v rade sú očíslovaní zľava doprava číslami 1 až n. Pre $n = 8$ je výsledok napríklad:

najprv 5. tanečník urobí Ľavú Patlu,
potom 3. tanečník urobí Pravú Patlu,
potom 1. tanečník urobí Pravú Patlu,
potom 8. tanečník urobí Ľavú Patlu.

Pre kontrolu uvádzame aj postupné fázy tanca (tanečníkov ležiacich na zemi znázorňujeme v zátvorkách, druhý v zátvorke leží na prvom):

1. fáza. 1 2 3 4 5 6 7 8
2. fáza. 1 (25) 3 4 6 7 8
3. fáza. 1 (25) 4 6 (73) 8
4. fáza. (25) (41) 6 (73) 8
5. fáza. (25) (41) (68) (73), tanec sa vydaril.

942. O burundijskom zjednotení

Po občianskej vojne sa Burundi rozpadlo na n maličkých štátikov. Každý mal svojho vládcu a meno, ale bolo ich toľko, že ich radšej budeme označovať číslami od 1 po n. Zavládol chaos a anarchia. Iba Veľký Mahdí (vládca v Bujumbure) vedel, čo robiť, aby bolo Burundi opäť jednotným, mocným a nezávislým štátom. Rozhodol sa, že na znovuzjednotenie Burundi použije svoj obrovský zlatý poklad, predajnosť vládcov jednotlivých štátikov a starý burundijský zvyk – sľub spolupatričnosti.

Sľub spolupatričnosti má v Burundi veľkú tradíciu. Spočíva v tom, že dvaja vládcovia (pôvodne rodín, teraz štátikov) slávnostne sľúbia, že každý, kto je alebo kto sa stane poddaným jedného z nich, stáva sa automaticky aj poddaným toho druhého. Týmto sľubom tak vznikne akási „spoluvláda“.

Mahdího plán bol jednoduchý: svojimi peniazmi „dopomôže“ k uzavretiu toľkých sľubov spolupatričnosti, až nakoniec budú vládnuť všetci vládcovia nad celým územím Burundi. Ďalej plánoval pozvať k sebe všetkých vládcov na hostinu a tam ich dať všetkých, až na seba, pozabíjať. Tak by sa stal jediným vládcom nad celým Burundi. Veľký Mahdí teda starostlivo pripravil veľa poslov, každému dal vrece zlatých burundijských dukátov a poslal ich k ostatným vládcom intrigovať. Teraz sedí na tróne a čaká na ich správy. Posli budú postupne prichádzať a hlásiť: „Veľký Mahdí, oznamujeme Ti, že vládca krajiny X uzavrel s vládcom krajiny Y sľub spolupatričnosti“. Ak sa týmto sľubom podarilo spojiť dve územia, ktoré ešte neboli zjednotené, dá Veľký Mahdí príkaz posla bohato obdarovať. Ak však nie, tak poslovi beda: keďže zbytočne premrhal Mahdího peniaze, Mahdí ho dá utopiť.

Úloha: Napíšte Mahdímu program, ktorý bude v cykle postupne načítavať správy poslov v tvare dvojíc X, Y a bude vypisovať, či má Mahdí dať posla odmeniť alebo utopiť. Keď program načíta správu $0, 0$, znamená to, že Mahdímu sa minuli peniaze. Vtedy program skončí a oznámi, či už je Burundi zjednotené alebo nie. Snažte sa, aby jednotlivé odpovede na správy poslov dával váš program v čo najkratšom čase, lebo Mahdí je veľmi netrpezlivý.

943. O malom lenivom krtkovi

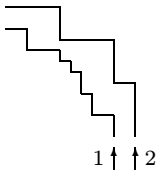
Malý lenivý krtko spal v brlôžku až do obeda. Jeho mamičku to nazlostilo, a tak si povedala: „Čo z môjho synčeka vyrastie, ak bude taký lenivý!“ a už ho aj išla budiť. „Vstávaj, krtko!“ rázne sa mu prihovorila, „ak dneska nepochytáš chrobáčikov aspoň v jednom našom tuneli, tak ťa naskutku paličkou vyobšívam!“ Čo mal chudák krtko robiť, nechcelo sa mu, ale musel vstať.

Krtia rodina mala na rovine vybudovaných n brlôžkov (so súradnicami $[x_1, y_1]$ až $[x_n, y_n]$), pričom medzi každými dvoma brlôžkami mala vykopaný priamy tunel. Každý deň

krtica s krtom preliezli všetky tunely a zbierali to, čo mali najradšej – chrobáčikov. Teraz mal aj malý lenivý krtko zbierať, ale keďže bol celkom inteligentný, najprv pouvažoval: „Najmenej chrobáčikov, a teda aj najmenej práce, bude v najkratšom tuneli. Stačí mi teda zistiť, ktoré dva brlôžky sú k sebe najbližšie. Musím však na to prísť skôr, ako sa mamička začne o mňa znovu zaujímať, takže by som mal urobiť menej ako $c \cdot n^2$ operácií, kde c je nejaká konštanta."

Úloha: Pomôžte krtkovi a naprogramujte mu to! Ak sa vám aj nepodarí nájsť algoritmus lepší ako kvadratický, nevadí, krtko sa poteší aj tomu, len vám dá za odmenu trošku menej chrobáčikov.

944. O valcočlovekovi



Valcočlovek je taký človek, ktorého tvar sa približuje tvaru valca s určitou výškou a šírkou. Výskyt ľudí takéhoto tvaru je veľmi vysoký napríklad medzi pracovníkmi francúzskych vínnych pivníc (čím viac vína pracovník skonzumuje, tým je širší).

Francúzske vínne pivnice sú spravidla dosť úzke, takže širší valcoľudia cez ne nedokážu ani prejsť, často sa v pivnici zaseknú a vínna pivnica sa stane nepoužívateľnou. Preto sa musí u každého pracovníka starostlivo sledovať jeho šírka, aby sa predišlo nehode. Na to je, samozrejme, nutné vedieť, aký najširší valcočlovek je ešte schopný pivnicou sa pretiahnuť.

O francúzskych vínnych pivniciach je známe, že majú dva vchody spojené chodbou. Obe steny chodby sa vyznačujú tým, že nezatáčajú dvakrát po sebe tým istým smerom a vždy zatáčajú v pravom uhle. Navyše obe steny zatáčajú pri vchodoch prvý raz tým istým smerom. Vďaka tomu sa dá celá pivnica zakódovať pomocou šírky vchodu do pivnice a dvoch postupností, ktoré sa získajú nasledovne:

Prvý vinár sa postaví na ľavú stranu vchodu a druhý na pravú. Obidvaja opakovane merajú najbližší priamy úsek, až kým ich stena chodby nezatočí. Namerané hodnoty zapisujú do svojej postupnosti. Vchod, od ktorého začnú merať, zvolia tak, že obe steny zatáčajú prvý raz doľava.

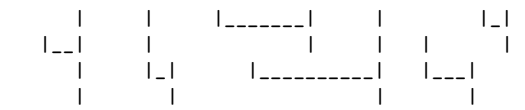
Úloha: Napíšte program, ktorý načíta šírku vchodu, dĺžku postupnosti prvého vinára, dĺžku postupnosti druhého vinára, obe tieto postupnosti a vypočíta, aký najširší valcočlovek dokáže kódovanou pivnicou bezpečne prejsť.

945. Štvrtá úloha o hre BOXES

Janko a Marienka sa hrali veľmi známu hru, BOXES. Hra BOXES sa hrá na papieri, na ktorom je $m \times n$ bodov usporiadaných do tvaru obdĺžnika. V každom ťahu hráči spájajú dva dosiaľ nespojené susedné body horizontálnou alebo vertikálnou čiarou. Pravidlá samotnej hry sú pre nás teraz nepodstatné (kto by ich chcel poznať, nech si pozrie úlohu 815). Jedna možná situácia hry BOXES je zobrazená v úlohe 925.

Takže Janko s Marienkou sa hrali túto hru a pre zmenu prehrával Janko. Keď videl, že ho od porážky už nič nezachráni, spomenul si na to, ako s ním Marienka naposledy (v úlohe 925) vybabrala. A preto sa opýtal: „Marienka, čo myslíš, koľko je na hracom pláne štvoriek? Nepomohla by si mi ich porátať?" Marienka sa na chvíľu zamyslela (no nie príliš dlho) a potom víťazoslávne povedala: „Ale Janko, no predsa 47! A teraz to môžeme dohrať." „Ako sa to len mohlo Marienke podariť?" rozmýšľal Janko, keď definitívne prehral.

Úloha: A čo si myslíte vy, milí programátori? Ako je to možné? Svoju odpoveď podporte efektívnym programom, ktorý bude v reprezentácii hernej situácie hry BOXES zisťovať počet štvoriek. Pre upresnenie: Janko si pod štvorkou predstavuje napríklad takéto obrazce:



Každá zo štyroch nožičiek štvorky môže byť ľubovoľne dlhá, ale nesmie mať nulovú dĺžku (napríklad `|_|` nie je štvorka). Odpoveď pre situáciu hry BOXES z úlohy 925 je 8.

1011. O Santovi a fľašiach

Zlatokopi z Vyšnej Klondiky radi chodievali do hostinca v Dolnom Kelčove posilňovať sa ohnivou vodou, hrať hazardné hry a uzatvárať stávky. I Santo a Banto si tam občas radi zašli. Raz sa im podarilo vyhrať stávku s krčmárom a za odmenu im krčmár postavil na pult do radu mnoho fliaš ohnivej vody rôzneho objemu a povedal: „Z týchto fliaš, ako tu v rade stoja, môžete vypiť (a potom prázdne vrátiť na pôvodné miesto) tie, ktoré sa vám páčia, ale tak, aby nakoniec v celom rade z každých troch po sebe idúcich fliaš bola aspoň jedna prázdna a aspoň jedna plná.“ Santo sa už-už chcel rozbehnúť k najväčšej fľaši, keď ho Banto upozornil na to, že to nemusí byť najvýhodnejšie.

Úloha: Pomôžte Santovi a napíšte program, ktorý načíta počet fliaš n, postupnosť ich objemov $v_1, v_2, \ldots, v_n$ (kde v_i je objem i-tej fľaše v krčmárovom rade) a poradí Santovi, ktoré fľaše má vypiť, aby prelial hrdlom najväčšie možné množstvo ohnivej vody.

1012. O kiribatskom chráme

Kiribatský chrám je podivuhodná stavba s pôdorysom tvaru rovnoramenného pravouhlého trojuholníka orientovaného z náboženských dôvodov pravým uhlom na sever. Na mape (kiribatská mapa je matica núl a jednotiek, kde nuly predstavujú more a jednotky pevninu) vyzerá chrám vyznačený dvojkami takto:

```
                                  2
                      2         2 2 2
          2         2 2 2     2 2 2 2 2
2     2 2 2     2 2 2 2 2 2 2 2 2 2 2 2 2
```

Na obrázku sú uvedené chrámy v štyroch najmenších veľkostiach. Všimnite si, že južná stena chrámu zaberá na mape vždy nepárny počet políčok.

Úloha: Napíšte program, ktorý načíta rozmery mapy, počet riadkov m, počet stĺpcov n (a maticu $A[1\,..\,m, 1\,..\,n]$ predstavujúcu mapu súostrovia Kiribati a vyznačí na mape (napríklad dvojkami) miesto, kde Kiribatčania môžu svojim bohom postaviť najväčší chrám. Ak je takých miest v Kiribati viac, stačí vyznačiť ľubovoľné z nich.

1013. O šikovnom učiteľovi

Žiaci 4.A a 4.B triedy nastúpili v tesnej miestnosti do dvojradu, pričom v oboch radoch boli žiaci utriedení podľa výšky od najnižšieho po najvyššieho (oba rady tvorili tzv. rebrík). Učiteľ si ich dal zavolať preto, lebo chcel zistiť výšku 30. najvyššieho žiaka. Nechcel však, aby sa oba rady začali triediť podľa výšky, to by nastala priveľká panika a krik. Všimol si, že v oboch radoch je práve 30 žiakov, chvíľu striedavo pozeral naľavo a napravo. Potom z druhého radu vyviedol Jožka a zmeral si jeho výšku. Zaujímavé na tom bolo to, že na väčšinu žiakov sa ani nepozrel.

Úloha: Sú dané dve polia $A[1\,..\,n]$, $B[1\,..\,n]$ rovnakej (!) dĺžky n. Obe polia sú vzostupne utriedené. Napíšte program, ktorý vypočíta hodnotu n-tého najväčšieho prvku oboch polí (t.j. hodnotu prvku, ktorý by stál na indexe $n+1$ v poli, ktoré by vzniklo zlúčením A a B). Váš program by mal postupovať podobne efektívne ako šikovný učiteľ.

1014. O smetiaroch

Na sídlisku Číselníkovo je n križovatiek očíslovaných od 1 po n. Križovatky sú pospájané ulicami rôznej dĺžky. (Pod ulicou v Číselníkove rozumejú súvislý úsek cesty neprerušený križovatkou.) Taktiež každá ulica má v Číselníkove svoje číslo. Popri každej ulici stoja

smetiaky obyvateľov Číselníkova plné odpadkov. Smetiari, ktorí majú depo na križovatke s číslom 1, majú k dispozícii jediné smetiarske auto, pomocou ktorého musia každý deň všetky smetiaky vyprázdniť.

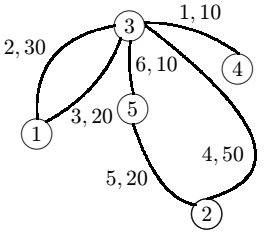
Úloha: Napíšte program, ktorý pre nich naprojektuje trasu, po ktorej majú ísť, aby (vyšli z depa) vyprázdnili všetky smetiaky (a vrátili sa do depa) a pritom prešli čo najkratšiu trasu (a minuli čo najmenej drahého benzínu).

Váš program bude na vstupe prijímať počet križovatiek n, prirodzené číslo k a k štveríc, pričom každá štvorica (c, i, j, dl) bude reprezentovať ulicu číslo c, dĺžky dl, spájajúcu križovatky i a j, $i < j$. Výstupom vášho programu bude postupnosť čísel ulíc, ktorá obsahuje každú ulicu, pričom súčet dĺžok ulíc bude čo najmenší.

Príklad:

Vstup:

```
5 6
1 3 4 10
3 1 3 20
4 2 3 50
2 1 3 30
5 2 5 20
6 3 5 10
```



Výstup:

```
Trasa: 2, 1, 1, 6, 5, 4, 3; Dĺžka trasy: 150
```

1015. O Batuchánových vyslancoch

Chán Zlatej hordy, Batu, vyslal zo svojho tábora na Volge na dvor Veľkého chána v Kambaluku (dnes Peking) troch svojich synov, aby mu blahoželali k jeho víťazstvu nad ríšou Sung. Počas spoločnej cesty mali ponavštevovať Batuových priateľov a odovzdať im jeho dary a pozdravy. Aby ich v rozľahlej Mongolskej ríši našli, dal Batu svojim synom zemepisné súradnice táborov svojich priateľov ako i zemepisné súradnice Kambaluku a svojho tábora na Volge. Vzhľadom na rozlohu Mongolskej ríše sa všetky zemepisné dĺžky Batuových priateľov nachádzali medzi zemepisnými dĺžkami Batuovho tábora a Kambaluku, pričom všetky boli navzájom rôzne. „Choďte“, povedal Batu svojim synom, „a nezabúdajte na staré mongolské cestovné príslovie: ‘Nikdy neprekroč žiadny poludník viac ako dvakrát!’. Ponáhľajte sa však, lebo nadchádza chvíľa, keď dáme plemenám uhorským pocítiť naše šípy, preto choďte najkratšou cestou, ktorá neporušuje naše zvyky!“. Synovia si vzali k srdcu slová svojho otca, navštívili Veľkého chána i všetkých Batuových priateľov a vrátili sa k Volge ešte včas, aby sa pridali k Batuovej výprave do Uhorska. Medzi dvoma miestami so zemepisnými súradnicami *Sirka1*, *Dlzka1* a *Sirka2*, *Dlzka2* je $\sqrt{(Sirka2 - Sirka1)^2 + (Dlzka2 - Dlzka1)^2}$ (v XIII. storočí sa ešte všeobecne neprijímal názor, že Zem je guľatá).

Úloha: Je dané prirodzené číslo n, $n > 2$ a polia $Sirka[1\,..\,n]$, $Dlzka[1\,..\,n]$, pričom pre $i < j$ platí, že $Dlzka[i] < Dlzka[j]$. ($Sirka[1]$, $Dlzka[1]$) sú zemepisné súradnice Batuovho tábora, ($Sirka[n]$, $Dlzka[n]$) Kambaluku a ostatné údaje sú zemepisné súradnice Batuových priateľov. Napíšte program, ktorý zistí, v akom poradí navštevovali Batuovi synovia jednotlivé miesta (t.j. nájde najkratšiu trasu, ktorá začína a končí v Batuovom tábore, prechádza Kambalukom a tábormi Batuových priateľov a neprechádza žiadny poludník viac ako dva razy).

1021. O dôležitej osobe

V Číselníkove žije n ľudí očíslovaných od 1 po n. Človek číslo i človeka číslo j buď pozná, alebo nepozná (sám seba každý pozná). Keď sa starostu Číselníkova opýtate: „Pozná človek číslo i človeka číslo j?“, vie na položenú otázku odpovedať (ÁNO alebo NIE) pre ľubovoľné i a j, $1 \le i, j \le n$. Keď sa ho však opýtate, či v Číselníkove existuje osoba, ktorú všetci

poznajú, ale ona pritom nikoho okrem seba nepozná, pokrčí ramenami. Starosta vždy hovorí pravdu.

Úloha: Napíšte program, ktorý bude starostovi klásť otázky typu „Pozná človek číslo i človeka číslo j?“, prijímať na ne odpovede typu ÁNO/NIE a v konečnom čase zistí, či v Číselníkove uvedená osoba existuje alebo nie.

1022. O Patagónskom fjorde

Keď Fernando Magellan hľadal v rokoch 1519–20 v službách španielskeho kráľa a nemeckého cisára Karola V. prieliv do Južného mora[11] (dnes Tichý oceán), v zúfalej snahe prehľadával všetky fjordy na východnom pobreží Patagónie (dnes Argentína). Raz vplával do fjordu[12], ktorý sa veľmi často rozchádzal na dve ramená, ktoré sa ďalej zásadne nespájali. Preto bol nútený rozdeliť svoju flotilu a každej lodi dať sledovať iné rameno fjordu. Kapitán Trinidadu však nedal pozor a jeho loď uviazla na plytčine. Ihneď poslal za Magellanom na vlajkovú loď San Antonio čln s prosbou o pomoc. Keď sa správa dostala do Magellanových uší, pozorne si prezrel doterajšie hlásenia a povedal: „Je to mrzuté. Keby bol San Antonio alebo Trinidad na hocktorom inom mieste ako je, cesta (po vode) od jedného k druhému by merala menej légií[13].“

Úloha: Napíšte program, ktorý načíta podstatné údaje o patagónskom fjorde a vypočíta dĺžku cesty (po vode), ktorú museli námorníci zo San Antonia podniknúť, aby pomohli uviaznutému Trinidadu.

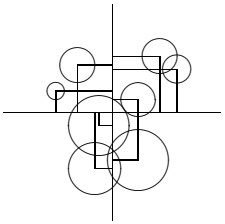
Údaje o fjorde sú v tvare:

⟨fjord⟩ ::= ⟨podfjord⟩
⟨podfjord⟩ ::= ⟨dĺžka ramena⟩ `legi(a/e/i) VLAVO` ⟨podfjord⟩ `VPRAVO` ⟨podfjord⟩
  | ⟨dĺžka ramena⟩ `legi(a/e/i) KONIEC`
⟨dĺžka ramena⟩ ::= **integer**

Napríklad fjord, ktorý sa po 6 légiách delí na dve ramená dĺžky 2 légie, pričom ľavé sa ešte delí na dve ramená, každé dĺžky 1 légia, sa zaznamená takto: `6 legii VLAVO 2 legie VLAVO 1 legia KONIEC VPRAVO 1 legia KONIEC VPRAVO 2 legie KONIEC`.

1023. O megalitickej kultúre

Iste poznáte megality vo Veľkej Británii. Čo však určite neviete (lebo sme si to vymysleli) je fakt, že neďaleko nich žil dávno-pradávno ľud, ktorý prinášal svojmu bohu plodnosti veľmi zaujímavé obete. Uprostred svojho sídliska mali postavenú sochu boha plodnosti. Rituál obetovania spočíval v tom, že sa šaman postavil k soche, náhodne odpočítal S krokov na sever, J krokov na juh, Z krokov na západ a V krokov na východ, až sa dostal na pole, na ktorom pestovali obilie. Podľa postavenia slnka stanovil r a ostatní všetku pôdu na r krokov od miesta (t.j. kruh), kde šaman stál, znehodnotili tak, že sa už nikdy nedala obrobiť. Znehodnocovanie pôdy prebiehalo rôzne, bohatší ju posýpali kameňmi, chudobnejší iba udupávali. S, J, Z, V a r boli každý rok rôzne, takže po niekoľkých rokoch musel ľud toto miesto opustiť. Pri sťahovaní sa rozdelil na mnoho skupín, z ktorých dobrá polovica padla za obeť útrapám a ľudožrútom.



Úloha: Napíšte program, ktorý načíta počet rokov n, počas ktorých sa konali uvedené obety, čísla S_i, J_i, Z_i, V_i a r_i, kde $1 \leq i \leq n$ a vypočíta, aká plocha (vyjadrená v šamanových krokoch štvorcových) pôdy bola pri nich znehodnotená.

---

[11] Dostal sa so svojou flotilou až na Filipíny, kde ho zabili domorodci (1521). Iba jednej lodi jeho flotily (Victoria, pod vedením Sebastiána del Cano) sa podarilo dostať späť do Španielska (1522). [2]

[12] úzky pás mora zasahujúci do vnútrozemia krajiny

[13] Légia (la legua marina) je španielsko-portugalská námorná míľa, t.j. jednotka dĺžky.

1024. O dopravnom ihrisku

V Číselníkove si obyvatelia postavili dopravné ihrisko. Najprv vyznačili n miest očíslovaných od 1 po n a potom ich podľa plánu pospájali cestami (tam, kde sa potom zbiehalo viac ciest, urobili križovatku). Plán bola obyčajná matica $A[1\,..\,n, 1\,..\,n]$ núl a jednotiek, kde $A[i,j] = 1$, keď miesta i a j mali byť spojené priamou cestou a $A[i,j] = 0$, keď nie. (Žiadna priama cesta spájajúca dve miesta neprechádza cez tretie. Navyše $A[i,i] = 0$ pre všetky $1 \leq i \leq n$.)

Keď bolo všetko hotové, zmocnila sa starostu zlá predtucha: „Občania, budeme my môcť na tomto ihrisku cvičiť jazdu do osmičky?“

Úloha: Napíšte program, ktorý načíta počet miest n, plán ihriska $A[1\,..\,n, 1\,..\,n]$ a zodpovie na starostovu otázku (ÁNO alebo NIE). Nezabudnite tiež pridať úvahu o efektívnosti vášho algoritmu, prípadne odvodenie zložitosti.

Na to, aby ste sa mohli dať do práce, treba si ešte sformalizovať pojem „jazda do osmičky“. Pod „osmičkou“ si budeme predstavovať postupnosť miest tvaru $b, u_1, u_2, \ldots, u_k, b, v_1, v_2, \ldots, v_l, b$, kde

1.) $k \geq 2, l \geq 2$
2.) každé dve po sebe idúce miesta sú spojené priamou cestou
3.) existuje taký index r, že pre každý index s, $1 \leq s \leq l$ platí $u_r \neq v_s$
4.) $u_i \neq b$ a $v_j \neq b$ pre každé $1 \leq i \leq k$ a $1 \leq j \leq l$
5.) pre každé tri po sebe idúce miesta m_1, m_2, m_3 platí $m_1 \neq m_3$.

1025. O Santovi a dynamite

Raz sa Santo opil a dostal nešťastný nápad: vyhodiť celé svoje nálezisko zlata do vzduchu. Porozhadzoval po ňom preto šúľky dynamitu a začal ich spájať zápalnou šnúrou, aby ich mohol odpáliť. Keď to uvidel Banto, rýchlo začal zápalné šnúry zasa rozpájať, lebo ho nelákala vidina, že z jeho jediného majetku ostanú kúdoly prachu.

Úloha: Napíšte program, ktorý načíta počet šúľkov dynamitu n (šúľky sú očíslované od 1 po n) a ďalej postupne číta vstupy tvaru `SANTO SPOJIL` i j alebo `BANTO ROZPOJIL` i j, kde i, j sú čísla šúľkov, alebo `KONIEC`. Keď sa načíta `KONIEC`, ukončí sa beh programu. Vždy keď sa Santovi podarí spojiť všetky šúľky dynamitu (a teda môže zrealizovať svoj opilecký sen), program zareaguje na daný vstup a upozorní Banta.

1031. U holiča

U holiča bolo $2n + 1$ stoličiek, na ktorých sedelo n mladých a n starých ľudí, jedna stolička bola prázdna. Stoličky boli očíslované od 1 po $2n + 1$. Ten, kto sedel na stoličke s menším číslom, sa dostal „pod kombajn“ prv, ako tí, čo sedeli na stoličkách s väčším číslom.

Keď to zbadal holič, vyhlásil, že všetci mladí musia prepustiť výhodnejšie miesta starším, inak strihať nebude. A navyše, on tu nechce mať žiaden neporiadok, takže mladí sa so starými povymieňajú pekne poporiadku: vždy niekto vstane a sadne si na voľnú stoličku, tým sa jeho stolička uvoľní, na ňu si sadne niekto ďalší, atď. Je jedno, kto kde bude sedieť počas vymieňania, nakoniec však musia sedieť starší „pred“ mladšími.

Úloha: Napíšte program, ktorý načíta n, pre každú stoličku i načíta, kto na nej sedí (-1 je mladý, 0 je nikto, 1 je starý) a vypíše postupnosť čísel stoličiek, z ktorých majú tí, čo na nich sedia, vstať a sadnúť si na (v tej chvíli) voľnú stoličku, aby po vykonaní týchto presunov boli všetci starí pred mladými (na polohe voľnej stoličky nezáleží). Bolo by dobré, keby váš program vypísal vždy postupnosť minimálnej dĺžky (dokážte!). Taktiež by sme sa potešili, keby bol váš program lineárny.

1032. O švajčiarskej pechote

Základný prvok švajčiarskej pechoty bol štvorec $n \times n$ kopijníkov (pod veľkosťou tohto štvorca budeme rozumieť rozmer n), ktorých kopije smerovali vždy von z tohto štvorca,

takže predstavovali pre nepriateľskú rytiersku jazdu, útočiacu z ktorejkoľvek strany, neprekonateľnú prekážku. Celá pechota pozostávala z takýchto štvorcov pre $n = 1, 2, \ldots, k$ (neobsahovala dva rovnaké štvorce), pričom k záležalo od veľkosti pechoty.

Každý kopijník patriaci k švajčiarskej pechote mal svoje identifikačné číslo, pričom všetky identifikačné čísla boli prirodzené $(1, 2, 3, \ldots)$ a navzájom rôzne. Identifikačné číslo jediného kopijníka vo štvorci veľkosti 1 bola jednotka. Ak mal nejaký kopijník identifikačné číslo id, buď to bola jednotka, alebo existoval v pechote kopijník s identifikačným číslom $id - 1$. Pri udeľovaní identifikačných čísel platila ďalej zásada, že každý kopijník zo štvorca menšej veľkosti mal menšie číslo ako ktorýkoľvek kopijník zo štvorca väčšej veľkosti.

Úloha: Napíšte program, ktorý načíta identifikačné číslo kopijníka švajčiarskej pechoty a vypíše veľkosť štvorca, do ktorého patrí. Program by mal pracovať dostatočne rýchlo pre ľubovoľné identifikačné číslo do 10^9.

1033. O listine kráľov

Pri archeologickom výskume sa našla do kameňa neznámym slabičným písmom vytesaná listina v neznámom jazyku. Podľa ilustrácií išlo evidentne o listinu kráľov dosiaľ neznámeho národa. Dalo sa teda predpokladať, že najfrekventovanejšie slovo v listine bude slovo kráľ.

Na prvý pohľad bolo jasné, že vodorovný klin oddeľuje jednotlivé slová a že písmo sa číta sprava doľava a zdola nahor. Nálezcovia preto listinu dôkladne okopírovali a každému znaku pre slabiku pridelili číslo od 1 do 138. Každé slovo listiny zaznamenali do jedného riadku súboru `listina.in` ako postupnosť čísel znakov jeho slabík oddelených medzerami. Žiadne slovo nemalo viac ako 5 slabík.

Úloha: Napíšte program, ktorý prečíta súbor `listina.in` a zistí všetky pravdepodobné zápisy slova kráľ.

Príklad:

Vstup:
```
2 11 11
1 122 12
8 8 1 1 8
2 3
1 122 12
2 3
```

Výstup:
```
1 122 12, 2 3.
```

1034. O lenivých novinároch

Niektorí novinári sú takí leniví, že do článku napíšu jednu pasáž aj viackrát, aby ľahko dosiahli požadovanú dĺžku. Šéfredaktori preto potrebujú program, ktorý pre daný text článku $T[1..N]$ $(N \leq 1000)$ zistí najdlhší úsek, ktorý sa v článku opakuje na inom mieste. V prípade, že je takých úsekov viac, zistí najľavejší z nich.

Príklad: Pre $N = 13$, $T =$ `abedabehdabeh` je výsledkom `dabeh`

1035. O hľadaní bodu

Vymenovaním proti smeru chodu hodinových ručičiek je daný konvexný n-uholník. Napíšte program, ktorý pre dané n a konvexný n-uholník nájde bod C taký, že maximum zo vzdialeností od tohto bodu do všetkých vrcholov n-uholníka je minimálne.

Príklad: Pre 5-uholník $[0,0], [2,-1], [4,0], [3,1], [1,1], [0,0]$ je $C = [2,0]$.

1041. Kombinačné číslo

Napíšte program, ktorý načíta celé čísla n a k, $0 \leq k \leq n \leq 100$ a vypočíta presnú hodnotu kombinačného čísla $\binom{n}{k}$. Pripomíname, že $\binom{n}{k} = n!/(k!\,(n-k)!)$, kde $x! = 1 \cdot 2 \cdot 3 \cdots x$.

Príklad: Pre $n = 100$ a $k = 50$ je výsledok 100 891 344 545 564 193 334 812 497 256.

1042. Katastrofa v Kiribati

Následkom zvyšovania obsahu ropy a olejov v morskej vode sa všetky koralové ostrovy pomaly, ale isto rozpadávajú (a potápajú). Ani Kiribati táto katastrofa neobišla. Z predchádzajúcich príkladov viete, že mapu Kiribati si možno predstaviť ako maticu núl a jednotiek, kde jednotky sú pevnina a nuly more. Katastrofu si potom možno predstaviť tak, že sa každý rok rozpadnú a potopia tie časti ostrovov, ktoré sú na mape reprezentované políčkami, ktoré aspoň jednou stranou susedia s morom.

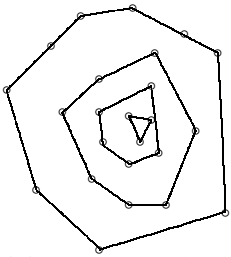
Úloha: Napíšte program, ktorý načíta rozmery mapy m a n, mapu s m riadkami a n stĺpcami núl a jednotiek a zistí, za koľko rokov klesne prvý (dnešný!!!) ostrov celý pod hladinu.

1043. Santove vrstvy

Možno sa pamätáte, že Santova a Bantova parcelka bola už po prvom vykolíkovaní kade-tade posiata kolíkmi. Jedného slnečného dňa sa v krčme v Dolnom Kelčove Santo dozvedel, že bol vyhlásený nový pozemkový zákon. Zákon obsahoval nasledujúce paragrafy:

§1. Zakazuje sa zatĺkať nové kolíky za účelom vyznačenia parcely.
§2. Ruší sa doterajšie rozdelenie zlatokopeckých parciel. Nové parcely sa vyznačia špagátom, ktorý sa musí obopínať o kolíky zatlčené pred dňom vyhlásenia tohto zákona, nemusí však použiť všetky kolíky. Špagát musí byť napnutý. Špagát sa nesmie vetviť a nesmie prechádzať popod alebo ponad iný špagát (vyznačujúci inú parcelu, pozri §4).
§3. Nové parcely môžu mať iba jedného majiteľa, a to toho, kto natiahol parcelu vyznačujúci špagát. Pozri §4.
§4. Parcela je priestor vnútri špagátu, s výnimkou parciel nachádzajúcich sa v tomto priestore.

Banto (ako starý abstinent) nebol vtedy v krčme, takže sa dosť začudoval nad tým, že Santo hneď po návrate z krčmy obohnal okolo ich parcelky špagát. Myslel si, že sa zase opil, ale Santo mu víťazoslávne oznámil, že sa práve stal jediným majiteľom celej ich parcelky, pretože v zmysle nového zákona si práve vyznačil parcelu s najväčšou možnou plochou, nakoľko to ich kolíky umožnili.



Banto sa však nedal ľahko odbiť. Preštudoval si nový zákon a v zmysle §4 zobral špagát a obohnal kolíky, ktoré Santo nepoužil (uvedomte si, že všetky tieto kolíky museli byť nutne v Santovej parcele). Takisto dbal na to, aby si vyznačil najväčšiu parcelu. Víťazoslávne potom Santovi svoj čin oznámil. Santo ihneď vbehol do už Bantovej parcely a urobil Bantovi to isté, čo on jemu. A Banto znova Santovi. Tak to išlo, až sa minuli kolíky, t.j. ostali menej ako tri kolíky, prípadne zvyšné kolíky ležali na priamke.

Úloha: Napíšte program, ktorý načíta počet kolíkov n, súradnice kolíkov x_i, y_i, $1 \leq i \leq n$ a vypočíta, koľko parceliek vlastní Santo a koľko Banto po tomto slnečnom dni. (Program by nemal byť väčšej než kvadratickej zložitosti.)

1044. Vekslovanie

Pred hotelom Kyjev stojí n vekslákov. Vekslák i mení menu A_i na menu B_i v pomere $p_i : q_i$. Momentálne stojíte pred hotelom i vy a máte k slovenských korún. Chcete ich zameniť za U$D, samozrejme, v čo najvýhodnejšom kurze. Môže sa vám oplatiť, ak spravíte viacnásobnú výmenu (cez rôzne meny). Veksláci však musia tiež z niečoho žiť, preto môžete predpokladať, že ak sa vám aj podarí po viacnásobnej výmene dostať k mene, ktorú ste už kedysi mali, budete mať menej peňazí ako keď ste túto menu mali prvýkrát.

Úloha: Napíšte program, ktorý načíta vyššie uvedenú špecifikáciu vekslákov a vypíše najvýhodnejší kurz koruny a dolára. (Na sume k nezáleží, vzhľadom na to, že neuvažujeme

zaokrúhľovanie.) Meny sa uvádzajú v medzinárodne zaužívaných skratkách nepresahujúcich desať znakov.

Príklad:

Vstup:	Výstup:
`1. vekslák: Sk -> DM 18:1`	`30:1`
`2. vekslák: DM -> U$D 5:3`	
`3. vekslák: Sk -> U$D 31:1`	

1045. Darmožráči

V Číselníkove žije n občanov očíslovaných od 1 po n. Každý z nich živí svojou prácou niekoľko spoluobčanov (môže sa stať aj to, že niekto neživí nikoho a niekto živí sám seba). Keď občan i živí občana j, značíme to $Z(i,j)$. Ďalej hovoríme, že občan i živí *v konečnom dôsledku* občana j, keď $Z(i,j)$, alebo keď existujú občania $u_1, u_2, \ldots, u_k$, kde $k \geq 1$ takí, že $Z(i,u_1) \wedge Z(u_1,u_2) \wedge \cdots \wedge Z(u_k,j)$. Tento fakt zapisujeme $Z^*(i,j)$. Darmožráčom je ten občan d, ktorého živí v konečnom dôsledku viac občanov ako on sám v konečnom dôsledku živí, t.j. pre ktorého platí $|\{k : Z^*(i,k)\}| < |\{l : Z^*(l,i)\}|$ (kde $|X|$ predstavuje počet prvkov množiny X).

Úloha: Napíšte program, ktorý načíta počet občanov n, zoznam dvojíc čísel občanov (prvý v dvojici živí druhého, zoznam je ukončený dvojicou $0,0$) a vypíše čísla všetkých darmožráčov.

z1011. O čiapočkách

Z rozprávok iste poznáte Červenú Čiapočku. V našom rozprávkovom lese však okrem nej vystrájajú aj Fialová, Modrá, Žltá, Oranžová, Okrová, Tyrkysová Čiapočka a mnohé iné, každá celá v oblečení svojej farby. Práve sa im stala galiba:

Pri hre na chytačku odložili na kopu svoje čiapočky, aby sa im nezašpinili. Keď bola hra v najlepšom, zjavil sa znenazdajky vyhladnutý vlk, ledva sa držal na nohách. Čiapočky sa preľakli, každá chytro zobrala z kopy jednu čiapočku a vzali nohy na plecia. Keď boli v bezpečí, zistili, že skoro každá z nich zobrala z kopy cudziu čiapočku. Každá si chce tú, čo má, okamžite vymeniť z rúčky do rúčky s nejakou inou Čiapočkou za svoju pôvodnú. Dá sa to?

Úloha: Predpokladajme, že v lese je n rôznofarebne oblečených Čiapočiek. Pre jednoduchosť každej farbe priradíme celé číslo z rozsahu od 1 po n. Napíšte program, ktorý načíta počet Čiapočiek n a pole $C[1\,..\,n]$, kde $C[i]$ je číslo farby čiapočky, ktorú drží v ruke Čiapočka, ktorej oblečenie má farbu číslo i a vypíše, či je možné, aby si Čiapočky vymenili vyššie uvedeným spôsobom svoje čiapočky tak, aby mala každá svoju pôvodnú.

z1012. O nešťastnom čísle

Pred medzištátnym futbalovým zápasom nastúpilo mužstvo trénera Mamojku do radu, aby si vypočulo hymny. Poverčivý Mamojka sa vždy obával svojho nešťastného čísla. Preto aj zakázal svojim hráčom nosiť dresy s týmto číslom. Keď uvidel, ako jeho mužstvo stojí v rade, vydesilo ho, že súčet čísel dresov niekoľkých vedľa seba stojacich hráčov sa môže náhodou rovnať jeho nešťastnému číslu. Prv ako stihol preveriť, či je to tak, hráči sa rozišli hrať a Mamojka si do konca zápasu od strachu obhrýzol všetky prsty. Bola Mamojkova obava opodstatnená?

Úloha: Napíšte program, ktorý načíta presný počet hráčov n, pole $C[1\,..\,n]$, kde $C[i]$ je číslo dresu i-teho hráča v rade a Mamojkovo nešťastné číslo k a zistí, či boli Mamojkove obavy opodstatnené.

z1013. O horskom nosičovi

Horský nosič Pišta sa chystá na cestu. Jeho náklad tvoria rovnako veľké a približne rovnako ťažké súdky rôznych tekutín (rôzne oleje, sirupy, alkoholické nápoje, ...). Z n

súdkov (očíslovaných číslami $1, 2, \ldots, n$), ktoré má k dispozícii, dokáže však vyniesť len k ($1 \leq k \leq n$) a je mu jedno, ktoré súdky to budú.

Úloha: Napíšte program, ktorý načíta čísla n a k a vypíše všetky možné k-tice súdkov, ktoré mohol Pišta zobrať na cestu, a počet týchto k-tic. (Na poradí k-tic nezáleží, každú k-ticu však musí program vypísať práve raz.)

Príklad: Pre $n = 3$, $k = 2$ vypíše váš program: `(1,2) (1,3) (2,3)`.

z1014. O pretláčaní

Na turnaji v pretláčaní sa stretlo n navzájom rôzne silných silákov. Po každom pretláčaní majú obaja borci možnosť dostatočne si oddýchnuť, takže každé pretláčanie vyhrá silnejší borec (keby niekto už pred turnajom vedel, kto z borcov je najsilnejší, vedel by aj, kto turnaj vyhrá).

Úloha: Napíšte program, ktorý navrhne rozpis zápasov (podobný ako je uvedené nižšie v príklade) taký, aby po jeho vykonaní bolo všetkým divákom jasné, kto z borcov je najsilnejší a kto najslabší. Rozpis by mal obsahovať čo najmenej zápasov. (Skúste nájsť vzorec, ktorý pre dané n udáva počet zápasov v rozpise vášho programu.)

Príklad: Pre $n = 4$ môže byť rozpis takýto (pozor!, pre $n = 4$ existuje aj lepší rozpis):

```
zápas č.  1   ...              borec č. 1 - borec č. 2
zápas č.  2   ...              borec č. 3 - víťaz zápasu č. 1
zápas č.  3   ...       víťaz zápasu č. 2 - borec č. 4
zápas č.  4   ...              borec č. 3 - borec č. 4
zápas č.  5   ...  porazený v zápase č. 1 - porazený v zápase č. 4
```

z1021. O čiapočkách II

Iste viete, že v našom rozprávkovom lese spolu s Červenou čiapočkou vystrájajú aj Fialová, Modrá, Žltá, Oranžová, Okrová, Tyrkysová Čiapočka a mnohé iné, každá celá v oblečení svojej farby. Práve sa im opäť stala galiba:

Pri hre na schovávačku odložili na kopu svoje farebné čiapočky, aby si ich pri hre nestratili. Keď bola hra v najlepšom, zjavil sa znenazdajky veľký medveď. Ešte sa nestačil uložiť na zimný spánok. Čiapočky sa preľakli, každá chytro zobrala z kopy jednu čiapočku a vzali nohy na plecia.

Keď boli v bezpečí, zistili, že skoro každá z nich drží v ruke cudziu čiapočku. Každá Čiapočka sa preto rýchlo chytila voľnou rukou svojej pôvodnej čiapočky. Koľko kruhov čiapočky utvorili? Za kruh sa považuje aj jedna Čiapočka, ktorá oboma rukami drží svoju pôvodnú čiapočku.

Úloha: Predpokladajme, že v lese je n rôznofarebne oblečených Čiapočiek. Pre jednoduchosť každej farbe priradíme číslo z intervalu od 1 po n. Napíšte program, ktorý načíta počet Čiapočiek n a pole $C[1..n]$, kde $C[i]$ je číslo farby čiapočky, ktorú drží v ruke Čiapočka s farbou oblečenia číslo i, a vypíše, koľko kruhov (cyklov) Čiapočky utvorili.

z1022. O Perzskej kráľovskej ceste

Perzská kráľovská cesta[14] spájala mesto Susy v Médii s mestom Sardy v Lýdii. Na piate storočie pred naším letopočtom to bola veľmi moderná cesta, ktorá mala na významných miestach vybudované stanice s možnosťou nocľahu a s dostatočným množstvom čerstvých koní. Keď „kráľ kráľov“, Xerxes[15], plánoval zhromažďovanie vojsk na výpravu proti Grékom, potreboval vedieť dĺžku cesty medzi ľubovoľnými dvoma stanicami. Z plánov cesty však vedel iba vzdialenosti medzi dvomi susednými stanicami.

Úloha: Napíšte program, ktorý načíta počet staníc n, postupnosť $L_1, L_2, \ldots, L_{n-1}$, kde L_i predstavuje dĺžku cesty medzi i-tou a $(i+1)$-vou stanicou. Potom bude postupne čítať dvojice staníc a pre každú dvojicu i, j vypíše vzdialenosť i-tej a j-tej stanice.

[14] Dal ju postaviť Dareios I (522–486 p.n.l.). Cesta je 5 m široká a 2500 km dlhá.
[15] bol synom Darea

z1023. O horskom nosičovi II

Horský nosič Pišta sa opäť chystá na cestu. Jeho náklad však tvoria rôzne potraviny s rozličnou cenou a rozličným objemom, ktoré sú dobre deliteľné (saláma, syr, slanina, chlieb, atď.). Z n druhov (očíslovaných číslami $1, 2, \ldots, n$), ktoré má k dispozícii, si musí vybrať len niektoré, lebo sa mu všetky nezmestia do batohu s objemom v. V jeho záujme je vybrať predmety tak, aby za ne na chate dostal čo najviac peňazí.

Úloha: Napíšte program, ktorý načíta objem Pištovho batohu, počet druhov potravín n, ich objemy a ceny a vypočíta, koľké časti si má Pišta odrezať z každého druhu potravín, aby za ne dostal čo najviac peňazí.

Príklad: Pre batoh s objemom 3 a dve potraviny, pričom prvej je 10 jednotiek a jej jednotková cena je 5 a druhej je 1 a jej jednotková cena je 6, treba zobrať 1/5 prvej a druhú všetku.

z1024. O Santovi a burze

Keď Santo a Banto našli na Vyšnej Klondike prvé zlaté hrudky, vybrali sa do Dolného Kelčova investovať ich, lebo nechávať si väčšiu hotovosť u seba sa na Klondike rovnalo istej záhube. Dolnokelčovská burza ponúkala na predaj akcie iba dvoch spoločností: Ťažobnej spoločnosti, ktorá zlatokopom dodávala technické vybavenie a Whisky company, ktorá im dodávala ohnivú vodu. Santo by bol samozrejme najradšej všetky peniaze investoval do Whisky company, ale Banto ho presvedčil, že tak by mohol o svoj majetok rýchlo prísť. Zašli preto k cigánke, ktorá bola známa svojimi jasnovideckými schopnosťami, a nechali si prorokovať ceny akcií oboch spoločností na najbližších n dní (ceny na burze sa vždy menili iba o polnoci). Cigánka pozrela do svojej sklenenej gule a vyčítala z nej postupnosť $t_1, t_2, \ldots, t_n$ cien akcií Ťažobnej spoločnosti, ako i postupnosť $w_1, w_2, \ldots, w_n$ cien akcií Whisky Company.

Úloha: Napíšte program, ktorý Santovi a Bantovi vypočíta, ako majú kupovať a predávať svoje akcie, aby po n dňoch znásobili svoj majetok čo najviac. Predpokladajte pritom, že cigánka sa vo svojej predpovedi nepomýlila. Nezabúdajte, že všetok Santov a Bantov majetok musí byť stále investovaný v akciách, t.j. ak Santo a Banto nejaké akcie predajú, tak ešte v ten deň musia za všetky utŕžené peniaze nejaké akcie nakúpiť, s výnimkou n-tého dňa, kedy sa všetky akcie speňažia. Ďalej predpokladajte, že na Dolnokelčovskej burze sa dajú kúpiť i časti akcií (1/2 akcie a pod.) a uvedomte si, že vzhľadom na tento predpoklad je výpočet vášho programu nezávislý od sumy peňazí, ktorú Santo a Banto dostali za svoje zlaté hrudky. Ceny akcií sú reálne čísla.

z1031. O binárnych vektoroch

Binárny vektor rozmeru n je n-tica núl a jednotiek. Napíšte program, ktorý pre dané n vypíše všetky binárne vektory dĺžky n (každý práve raz) v takom poradí, že dva po sebe idúce vektory a prvý a posledný vektor sa líšia v jedinej číslici[16].

Pre $n = 3$ vypíše program napríklad:

`(0,0,0),(0,1,0),(0,1,1),(0,0,1),(1,0,1),(1,1,1),(1,1,0),(1,0,0).`

z1032. O horskom nosičovi III

Horský nosič Pišta sa chystá na cestu. Má k dispozícii batoh s objemom k a pätnásť (tento raz nedeliteľných) predmetov očíslovaných od 1 po 15 s objemami $v_1, v_2, \ldots, v_{15}$ a cenami $c_1, c_2, \ldots, c_{15}$. Napíšte program, ktorý Pištovi poradí čísla predmetov, ktoré si má do batohu dať, aby sa mu tam jednak vošli a aby tam mal tovar najväčšej možnej hodnoty.

[16] Takáto postupnosť vektorov sa nazýva Grayov kód po Frankovi Grayovi.

z1033. *a*...*ab*...*bc*...*c*

Je dané slovo w, t.j. pole znakov, dĺžky n. Napíšte program, ktorý nájde najväčšie k, pre ktoré existuje v tomto slove podslovo tvaru `a`...`ab`...`bc`...`c` zložené z k áčok, z k béčok a z k céčok za sebou.

Príklad: Pre slovo `aaabbbbccccccaaaaabbcccc` je $k = 2$.

z1034. „Skladanie“ skupín Čiapočiek

Predpokladajme, že v našom rozprávkovom lese vystrája $2n$ Čiapočiek, pričom ku každej z n farieb, očíslovaných od 1 po n, existujú práve dve Čiapočky s oblečením tejto farby. Všetky Čiapočky sa rozdelili do dvoch skupín tak, že v ani jednej skupine neboli dve Čiapočky rovnakej farby. Potom obe skupiny začali tancovať čiapočkovský tanec na rôznych miestach čistinky. Na začiatku tanca v oboch skupinách Čiapočky vyhodili svoje čiapočky do vzduchu, každá chytila nejakú čiapočku a potom sa ľavou rukou chytila pravej ruky Čiapočky, ktorá mala kabátik tej istej farby ako ňou chytená čiapočka. Čiapočky takto vytvorili kruhy, roztočili ich a ujúkali. Niektoré Čiapočky, ktorým sa podarilo chytiť svoju vlastnú čiapočku, sa otáčali na mieste. Takto Čiapočky tancovali a ujúkali, až sa obe skupiny dostali na dohľad. Vtedy si každá Čiapočka pozrela farbu kabátika Čiapočky, ktorú drží ľavou rukou. Potom očkami vyhľadala v opačnej skupine Čiapočku, ktorá mala kabátik takej istej farby, preniesla svoj zrak na kabátik Čiapočky, ktorú táto Čiapočka drží ľavou rukou a dobre si zapamätala jeho farbu. Na povel si potom Čiapočky v oboch skupinách pustili ruky a prechytili sa tak, že ľavou rukou chytili tú Čiapočku vo svojej skupine, ktorej farba kabátika bola zhodná so zapamätanou farbou. A ujúkali a točili sa až do večera.

Úloha: Napíšte program, ktorý načíta n a pre obe skupiny pre všetky Čiapočky farbu čiapočiek, ktoré na začiatku tanca chytili a znázorní obe tanečné skupiny pred a po prechytení.

Príklad: Pre $n = 5$

```
1. skupina                          2. skupina
==========                          ==========
1.Čiap. chytila čiap. farby 2       1.Čiap. chytila čiap. farby 2
2.Čiap. chytila čiap. farby 3       2.Čiap. chytila čiap. farby 3
3.Čiap. chytila čiap. farby 4       3.Čiap. chytila čiap. farby 1
4.Čiap. chytila čiap. farby 1       4.Čiap. chytila čiap. farby 5
5.Čiap. chytila čiap. farby 5       5.Čiap. chytila čiap. farby 4

1. skupina pred prechytením         2. skupina pred prechytením
===========================         ===========================
1 -> 2 -> 3 -> 4 -> 1               1 -> 2 -> 3 -> 1
5 -> 5                              4 -> 5 -> 4

1. skupina po prechytení            2. skupina po prechytení
========================            ========================
1 -> 3 -> 5 -> 4 -> 2 -> 1          1 -> 3 -> 2 -> 4 -> 5 -> 1
```

`1->2->3->4->1` je ten istý kruh Čiapočiek ako `3->4->1->2->3`, ale nie je to ten istý kruh ako `4->3->2->1->4` !

Skúste nájsť:

a) Čo najmenší vstup pre váš program, pre ktorý budú skupiny po prechytení rôzne.

b) Čo najväčší vstup pre váš program, pre ktorý sú skupiny pred prechytením rôzne, ale po prechytení už sú rovnaké.

z1041. Sám od seba samého seba

Napíšte, alebo aspoň vymyslite, ako by sa dal napísať program (v jednom z jazykov Pascal, C, BASIC, prípadne v ich odrodách), ktorý po spustení vypíše svoj zdrojový text. Program nesmie používať prácu so súbormi, v BASICu nesmie používať príkazy *LIST* a *DATA*. Akýkoľvek výstup môže realizovať len pomocou *write*, *writeln* (Pascal), *printf* (C), *PRINT* (BASIC). Jednou vetou, svoj zdrojový text musí poznať sám od seba a nesmie sa naň pýtať operačného systému.

z1042. Hoare

Je dané n-prvkové pole $A[1\,..\,n]$ celých čísel a index i, $1 \le i \le n$. Označme p hodnotu prvku $A[i]$. Napíšte program, ktorý povymieňa prvky tohto poľa tak, že hodnota p bude na nejakom indexe j (t.j. bude $p = A[j]$) a bude platiť:

a) pre všetky k, $1 \le k < j$; $A[k] \le p$ a súčasne
b) pre všetky k, $j < k \le n$; $A[k] \ge p$.

z1043. *AB*-slová

AB-slovo je slovo zložené výlučne z písmen `A` a `B`, napríklad `AA`, `ABB`, `BBB`. Napíšte program, ktorý na úvod prečíta zo vstupu postupnosť rôznych *AB*-slov, oddelených jednou medzerou. Potom bude v cykle zo vstupu prijímať *AB*-slovo, na ktoré zareaguje jedným z nasledujúcich spôsobov:

a) ak prijal prázdne slovo, ukončí svoju činnosť,
b) ak prijal slovo, ktoré nebolo v úvodnej postupnosti, vypíše reťazec `nepoznám` a pokračuje,
c) ak prijal slovo, ktoré bolo v úvodnej postupnosti, vypíše jeho poradie v tejto postupnosti a pokračuje.

Príklad: Pre úvodnú postupnosť `AAAA`, `ABAB` máme:

Vstup:	Výstup:
`AAA`	`nepoznám`
`ABAB`	`2`
`AAAA`	`1`

z1044. Vzorcotvorca

Asi viete, že $1+2+3+\cdots+n = n(n+1)/2$. Už menej z vás pozná vzorec $1^2+2^2+3^2+\cdots+n^2 = n(n+1)(2n+1)/6$. Ľahko si však pre akékoľvek zvolené k vyrátate vzorec pre súčet $1^k+2^k+3^k+\cdots+n^k$, ak vám prezradíme, že vzorec má tvar polynómu $(k+1)$-vého stupňa, t.j. $a_{k+1}n^{k+1}+a_kn^k+\cdots+a_1n+a_0$. Je to však ľahké urobiť programu?

Úloha: Napíšte program, ktorý pre dané k vypíše vzorec pre súčet radu k-tych mocnín. Desatinné čísla vo vzorci stačí vypísať na dve desatinné miesta (pokúste sa ich nahradiť zlomkami).

Príklad: Pre $k = 2$ vypíše `0.33*n^3 + 0.50*n^2 + 0.17*n^1 + 0.00*n^0`.

1111. O arite otáznika

Aritou operátora nazývame počet operandov, ku ktorým sa viaže (napríklad `+` má aritu 2). Ďalej definujeme P-výraz nasledujúcimi pravidlami:

1. P-výrazom je `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`.
2. keď O je operátor s aritou n a v_1, v_2, …, v_n sú P-výrazy, potom aj $O\,v_1\,v_2\,\ldots\,v_n$ je P-výraz. O môže byť jeden z nasledujúcich operátorov: `$` s aritou 1, `+ - * /` s aritou 2, `F G H` s aritou 3 a `?` s neznámou (ale pevnou) aritou.
3. nič iné P-výrazom nie je.

Úloha: Napíšte program, ktorý zistí, či daná postupnosť znakov tvorí P-výraz. V prípade, že tvorí, vypíše aj aritu otáznika.

Príklad: `+22`, `F+22*33F123`, `$$$$1`, `???1`, `??222`, `$?` sú P-výrazy a `+aa`, `F1234`, `$22`, `$`, `???11`, `+?12?123` nie sú P-výrazy.

1112. O súčte čísel

Na vstupe máme postupnosť nenulových celých čísel ukončenú nulou. Napíšte program, ktorý načíta túto postupnosť a vypíše jej súčet.

Obmedzenie: V celom programe môžete použiť iba procedúry bez parametrov a jedinú jednoduchú globálnu premennú.

1113. O kiribatských plavidlách

Pobrežné vody okolo súostrovia Kiribati sú dosť búrlivé, preto si domorodci stavajú zvláštne plavidlá tak, aby mali čo najväčšiu stabilitu. Kiribatské plavidlo sa skladá z n malých pltí očíslovaných 1 až n, z ktorých niektoré dvojice sú pozväzované rôzne dlhými palicami.

Úloha: Váš program má načítať počet malých pltí n, počet palíc m a m trojíc čísel A_i, B_i, D_i s významom: plť A_i je zviazaná s plťou B_i palicou dĺžky D_i. Výstupom je `ANO`, ak takto zadané plavidlo je funkčné (t.j. existuje v rovine rozloženie pltí tak, aby vyhovovali vzdialenosti), alebo `NIE`, ak nie je.

Palice sa uväzujú vždy na jedno miesto plte, takže plte môžete považovať za body.

1114. O Santovom pohľade na sídlisko

Na sídlisku stojí n domov. Všetky domy majú obdĺžnikový pôdorys, pričom všetky steny každého domu ležia buď na poludníkoch alebo rovnobežkách (predpokladajte, že sídlisko sa nachádza na Slovensku a nie napríklad na severnom póle). Pre každý dom vieme zemepisnú dĺžku najzápadnejšieho poludníka, ktorý domom prechádza (označme x_i), zemepisnú šírku najjužnejšej rovnobežky, ktorá domom prechádza (označme y_i), dĺžku domu v smere rovnobežiek (označme $dlzkax_i$), dĺžku domu v smere poludníkov (označme $dlzkay_i$) a výšku domu v metroch (označme v_i).

Pre lepšiu orientáciu je zemepisná dĺžka vyjadrená počtom stotisícin stupňa od zemepisnej dĺžky najzápadnejšej steny na sídlisku, zemepisná šírka počtom stotisícin stupňa od dĺžky najjužnejšej steny na sídlisku a dĺžky stien domu v stotisícinách stupňov. Pri predpoklade, že sídlisko bude mať ľudské rozmery, bude teda približne: $0 \le x_i < 1000$, $0 \le y_i < 800$, $0 < dlzkax_i < 1000$, $0 < dlzkay_i < 800$ a $0 < v_i \le 50$, pričom sú x_i, y_i, $dlzkax_i$, $dlzkay_i$ a v_i reálne čísla.

Na juh od sídliska sa nachádza pozorovateľ Santo. Je od sídliska vzdialený natoľko, že bočné steny domov sa javia také malé, že ich jeho oko zanedbáva. To znamená, že keď urobí napríklad krok doľava, vidí sídlisko presne také isté ako predtým.

Úloha: Napíšte program, ktorý vypíše všetky čísla domov, z ktorých Santo aspoň kúsok vidí. Čísla vypíšte v utriedenom poradí.

Keď za sebou stoja dva rovnaké domy, zadný nevidno.

1115. O ACM

ACM ICPC[17] je programátorská súťaž, ktorá je zaujímavá tým, že jediným rozhodcom na nej je počítač. Úlohy, ktoré majú súťažiaci riešiť, majú do posledného bajtíku predpísaný tvar vstupu a výstupu. Riešenia nesmú nič čítať ani vypisovať na terminál, všetko čítajú zo vstupného súboru `meno.in` a vypisujú do výstupného súboru `meno.out` (každá úloha má vopred pevne stanovené meno).

Riešiteľ, ktorý sa už nazdáva, že nejakú úlohu vyriešil, nahrá ju so stanoveným menom na disketu a zanesie na testovanie. Opravovateľom oznámi číslo svojho družstva, názov riešenej úlohy a jazyk (Pascal alebo C), v ktorom je úloha napísaná. Opravovatelia založia

[17] ICPC znamená International Collegiate Programming Contest, ACM je Association for Computing Machinery, ktorá ho organizuje. Napriek tejto skutočnosti sa tejto súťaži medzi pospolitým ľudom hovorí len „ACM“.

jeho disketu do mechaniky testovacieho počítača a spustia špeciálny testovací program. Ten skopíruje riešenie z diskety na disk testovacieho počítača, preloží ho pomocou dohodnutého prekladača (tpc, tcc, bcc) a spustí s vopred pripraveným vstupným súborom (ktorý riešiteľ nepozná). Ak testovaný program nevypočíta výsledky v stanovenom časovom limite, testovací program oznámi, že riešenie je neefektívne, inak výsledok porovná s vopred pripraveným vzorovým výstupom. Ak si úplne (bajt po bajte) zodpovedajú, oznámi, že riešenie testom prešlo a je OK. Inak oznámi chybu.

Úloha: Navrhnite testovací program pre súťaž ACM ICPC. Máte teda vyrobiť balík programov spolu s návodom na používanie (!), ktorý v konečnom dôsledku umožní organizátorom súťaže testovanie správnosti riešení.

Podmienky:

1. Testovanie prebieha na počítači s vami zvoleným operačným systémom.
2. Opravovatelia majú dopredu pomenovanú každú úlohu najviac osemznakovým menom, pripravené vzorové vstupy `meno.in` a vzorové výstupy `meno.tst` (tie sa budú porovnávať s `meno.out`, ktoré vytvárajú riešenia) a odhadnuté časové limity pre jednotlivé úlohy.

   Môžete (ale nemusíte) predpokladať nasledovné obmedzenia:
3. Testované programy sú v jedinom jazyku, napríklad v Pascale.
4. Testované programy sú preložiteľné (kompilátor ich úspešne preloží).
5. Testované programy bežia v reálnom čase (nie je potreba násilne prerušiť ich beh).
6. Nezáleží na čase riešenia, každé riešenie je efektívne.

1121. O sťahovaní študentov

Na internáte je n dvojposteľových izieb (očíslovaných od 1 po n), v ktorých býva $2n$ študentov. Každému študentovi je pridelené číslo internátneho preukazu (tzv. ČIP, od 1 po $2n$) a číslo izby, kde je ubytovaný. Keďže boli títo študenti ubytovaní mimoriadne demokratickým spôsobom (podľa abecedy), málokto je spokojný so svojím spolubývajúcim.

Úloha: Napíšte program, ktorý načíta počet izieb n, pole $Izba[1\,..\,2n]$, kde $Izba[i]$ je číslo izby, na ktorej býva študent s ČIP i a pole $Kamarat[1\,..\,2n]$, kde $Kamarat[i]$ je ČIP študenta s ktorým chce študent s ČIP i bývať. V prípade, že je niekomu jedno, s kým chce bývať, je jemu prislúchajúca hodnota v poli $Kamarat$ nulová. Váš program vypíše postup preubytovania pozostávajúci zo správ typu `študent` ČIP i, `izba` j `sa vymení so študentom` ČIP k, `izba` l, alebo správu `Nedá sa vyhovieť všetkým`. Vo vašom riešení by nemali chýbať úvahy o optimálnosti použitého algoritmu.

1122. Binár

Napíšte program, ktorý načíta kladné číslo n a vypíše nasledujúci riadok:

„dvojkový zápis n“`(2) =` „desiatkový zápis n“`(10)`

Príklad: Pre $n = 10$ vypíše: `1010(2) = 10(10)`.

Obmedzenie: V celom programe môžete použiť iba procedúry bez parametrov a jedinú jednoduchú globálnu premennú.

1123. O byrokratoch

Kde bolelo, tam bolelo, bola raz jedna krajina, v nej kopa byrokratov a bolelo to dosť. Ak človek niečo chcel, musel najprv získať mnoho pečiatok, a na to, aby dostal niektorú pečiatku, musel získať najprv mnoho iných. Našťastie tam boli i také pečiatky, ktoré sa dali získať samostatne.

Úloha: Pomôžte ľuďom v tejto nešťastnej krajine a napíšte pre nich program, ktorý načíta počet pečiatok, pre každú pečiatku nulou ukončený zoznam pečiatok, ktoré treba na to, aby ste ju získali, ďalej načíta číslo pečiatky P a vypíše všetky postupy ako možno pečiatku P získať (prípadne smutnú správu `Presťahujte sa do inej krajiny`, ak sa pečiatka získať nedá).

Príklad:

Vstup:	Výstup:
5	(5 4 3 2 1)
2 4 3 0	(4 5 3 2 1)
5 3 0	(4 3 5 2 1)
4 0	
0	
0	
1	

1124. O ilegálnej organizácii

V ilegálnej organizácii pracuje n ľudí. Ako sa na ilegálov patrí, každý z nich sa pozná s najviac tromi ďalšími. Neoslovujú sa menom, ale zásadne číslami od 1 po n.

Dôležitý ilegál je taká osoba, ktorej zadržanie poriadkovými silami spôsobí rozpadnutie organizácie na aspoň dve časti, medzi ktorými nie je žiadne spojenie.

Úloha: Napíšte program, ktorý načíta počet ilegálov n, ďalej pre každého ilegála čísla spolupracovníkov, ktorých pozná, a zistí, či je medzi ilegálmi aspoň jeden dôležitý ilegál.

1125. Chodiace písmenká

Iste poznáte vírus Padajúce písmenká. Vašou úlohou bude naprogramovať podstatnú časť jeho novej verzie, ktorej dáme pracovný názov Chodiace písmenká. Každú sekundu si každé písmenko na obrazovke vygeneruje náhodne smer, ktorým sa chce uberať. Všetky písmenká sa potom naraz(!) pohnú svojím smerom. Keď chce viac písmeniek vstúpiť na to isté miesto, náhodne sa rozhodne, ktoré to bude, a ostatné ostanú stáť.

Napríklad z `AB` `CD` kde `A` chce ísť doprava, `B` chce ísť dole, `C` chce ísť hore, `D` chce ísť doľava bude `CA` `DB`.

Úloha: Urobte program, ktorý po spustení realizuje jedno posunutie písmeniek.

Ďalej môžete, ale nemusíte urobiť jeho rezidentnú verziu, ktorá sa zavesí na prerušenie od systémových hodín, chodenie písmeniek aktivuje o 12:00 a potom každú sekundu posúva písmenká (písmenká idú na obed).

1131. O osemsmerovke

Dané sú rozmery osemsmerovky n, m, matica znakov $A[1\,..\,n, 1\,..\,m]$ reprezentujúca osemsmerovku, počet slov k a k slov osemsmerovky.

Úloha: Napíšte program, ktorý osemsmerovku, zadanú týmito údajmi vyrieši. (Riešenie osemsmerovky sa číta po riadkoch od najvrchnejšieho a od najľavejšieho znaku v riadku.) Predpokladajte, že žiadne slovo, ktoré sa bude škrtať, nie je podslovom iného (napríklad, nie sú tam naraz slová „ale“ a „alebo“).

1132. Zátvorky

Napíšte program, ktorý zo vstupu číta postupnosť znakov obsahujúcu len '`(`', '`)`', '`<`', '`>`', '`{`', '`}`' ukončenú medzerou, a vypíše `ANO/NIE` podľa toho, či je táto postupnosť dobre uzátvorkovaným výrazom.

Obmedzenie: Môžete použiť jedinú premennú typu char.

1133. O stupni čísla

Pre prirodzené čísla (a nulu) definujme stupeň čísla x nasledovne:

$$st(x) = \begin{cases} st(0) = 0 & \\ st(x) = 1 & \text{ak } 1 \le x \le 9 \\ st(x) = st(f(x)) + 1 & \text{ak } x \ge 10 \end{cases}$$

kde $f(x)$ je súčin všetkých cifier čísla x (v desiatkovej sústave).

Úloha: Napíšte program, ktorý pre dané k vypíše najmenšie číslo stupňa k.

Tieto čísla s k veľmi rýchlo rastú, snažte sa napísať časovo čo najefektívnejší program. Nemusíte robiť aritmetiku veľkých čísel, pre hodnotenie je podstatný algoritmus.

1134. O Kocúrkove

V Kocúrkove majú n domov, pričom chcú postaviť cesty medzi niektorými dvojicami domov. Keďže Kocúrkovčania neoplývajú zbytočnou inteligenciou, nemajú radi križovatky, lebo na nich často zablúdia. Mali by sme im pomôcť zistiť, či sa dajú cesty medzi domami postaviť tak, aby sa žiadne dve neprekrížili.

Úloha: Napíšte program, ktorý načíta počet domov n, k dvojíc domov, ktoré majú byť spojené cestou a vypíše, či sa dajú medzi nimi postaviť cesty bez križovatiek.

Pomôcka: V dvoch nasledujúcich prípadoch sa to nedá urobiť:
1. prípad: $n = 5$, $k = 10$, dvojice: $[1,2]$, $[1,3]$, $[1,4]$, $[1,5]$, $[2,3]$, $[2,4]$, $[2,5]$, $[3,4]$, $[3,5]$, $[4,5]$.
2. prípad: $n = 6$, $k = 9$, dvojice: $[1,4]$, $[1,5]$, $[1,6]$, $[2,4]$, $[2,5]$, $[2,6]$, $[3,4]$, $[3,5]$, $[3,6]$.

1135. O formátovači

Iste ste sa už naučili, že programy je dobre písať v nejakom presne stanovenom formáte, napriek tomu, že kompilátoru to je jedno. Problém vzniká, keď určitý programový produkt robí viac programátorov, ktorí majú odlišné spôsoby písania programov. V takýchto programoch možno použiť program zvaný formátovač, ktorý program zapísaný ľubovoľným spôsobom prepíše do zvoleného formátu. Napríklad program vľavo upraví na program vpravo

```
var s:string;
begin
  readln(s);
  if s='KSP' then
    begin
      write('to uz ano');
      writeln
    end
  else writeln('nic zaujimave');
end.
```

```
var s : string;
begin
  readln (s);
  if s = 'KSP' then begin
    write ('to uz ano');
    writeln
  end else
    writeln ('nic zaujimave')
end.
```

Úloha: Navrhnite a naprogramujte formátovač pre vami zvolený jazyk (Pascal alebo C). Kľúčovým miestom vášho riešenia bude analýza možností zápisu programov a z nej vychádzajúci návrh spôsobu zadávania formátu (formátovač by mal umožniť užívateľovi zadať ním používaný tvar).

Zadávať tvar ide samozrejme len do istej miery. Stačí štandardný Pascal, C. Môžete predpokladať, že formátovaný program je syntakticky správny.

1141. O permutáciách II

Napíšte program, ktorý na vstupe prijme čísla n, k ($k \leq n!$) a vypíše k náhodne vybraných permutácií čísel $1, 2, 3, \ldots, n$. Pritom nesmie jednu permutáciu vypísať viackrát a pravdepodobnosť výberu vypísanej k-tice musí byť rovnaká ako pravdepodobnosť výberu ľubovoľnej inej k-tice. Pokúste sa zdôvodniť správnosť vášho algoritmu.

1142. O delení siedmimi

Napíšte program, ktorý na vstupe prijme z riadku postupnosť cifier a vypíše `ANO` alebo `NIE` podľa toho, či je číslo (v desiatkovej sústave) zostavené z týchto číslic deliteľné siedmimi alebo nie. Cifry sú v opačnom poradí, t.j. prvá ide cifra pri 10^0, ďalej cifra pri 10^1, 10^2, atď. Za poslednou cifrou (ktorou je cifra pri najvyššej mocnine 10, rôzna od nuly) už nie sú v riadku žiadne medzery (t.j. hneď po prečítaní poslednej cifry nastane `EOLN`).

Na program sa kladú dve obmedzenia: program môže používať len jednu premennú typu **integer** a program musí dokázať spracovať i 40 ciferné číslo za niekoľko stotín sekundy.

Pomôcka: $(a \cdot b) \bmod c = ((a \bmod c) \cdot (b \bmod c)) \bmod c$

1143. O prešmyčkách

Slovo `LODE` je prešmyčka zo slova `DELO` alebo aj zo slova `DOLE`. Vašou úlohou bude napísať program, ktorý načíta číslo k – počet slov a následne k slov, ktoré rozdelí do skupín tak, že v každej skupine je každé slovo prešmyčkou ostatných. Žiadne slovo nie je dlhšie ako 10 znakov a $k \leq 3000$. Slová patriace do jednej skupiny budú vypísané v jednom riadku, slová nepatriace do jednej skupiny budú v rôznych riadkoch. Poradie slov v skupine, ako aj poradie prvých slov v skupinách bude rovnaké ako na vstupe.

Príklad: Pre $k = 10$ a slová: `LODE MASO DELO SAKO OSMA DOLE SAMO KOSA PETRA PATER` program vytvorí skupiny:

```
LODE DELO DOLE
MASO OSMA SAMO
SAKO KOSA
PETRA PATER
```

1144. O poradí

V súťaži ACM sú riešenia súťažiacich testované dvojkolovo. Najprv sa v prvom kole priebežne testujú všetky riešenia (niektoré i viackrát) a potom sa lepšie z nich otestujú v druhom kole. Za každé testovanie dostane súťažiaci určitý počet bodov. Testy druhého kola sú zamerané na špeciálne prípady, takže o konečnom poradí rozhodujú body udelené v prvom kole a len v prípade rovnosti bodov z prvého kola rozhodujú body z kola druhého. V prípade, že bol program otestovaný v prvom kole viackrát, berie sa za platný najvyšší dosiahnutý počet bodov. Testovania neprebiehajú na jednom mieste, takže keď sa výsledky testovaní prinesú na vyhodnotenie, tvoria súbor, zložený z viet tvaru: „Meno (20 bajtov), body (4 bajty), kolo (1 bajt)“, ktoré sú úplne pomiešané. Napíšte program, ktorý načíta súbor uvedeného tvaru a vypíše výsledkovú listinu súťaže ACM.

1145. O preprocesore

Preprocesor je program, ktorý umožňuje užívateľovi používať pri písaní programu v nejakom programovacom jazyku vlastnú syntax. Kód obsahujúci nielen zápisy v programovacom jazyku, ale i direktívy definujúce vlastnú syntax a zápisy v tejto syntaxi preprocesor preloží do kódu, ktorý bude obsahovať iba zápisy v programovacom jazyku. Ten sa potom dá použiť na ďalšie spracovanie. Preprocesor nie je závislý na používanom programovacom jazyku, závislý je iba od spôsobu definovania vlastnej syntaxe. Náš preprocesor bude rozumieť jedinej direktíve tvaru

```
\def\výraz1{výraz2}
```

Kdekoľvek za touto direktívou použijeme v kóde `\výraz1`, bude tento výraz nahradený výrazom `výraz2`. Má to však jeden háčik. Výrazy `výraz1` aj `výraz2` budú môcť obsahovať aj špeciálne podvýrazy `?X` a `*X` (namiesto `X` môžeme použiť ľubovoľné veľké písmeno). Vo výraze `výraz1` podvýraz `?X` predstavuje ľubovoľný znak a `*X` ľubovoľný neprázdny reťazec znakov. Ak sa rovnaký špeciálny podreťazec vyskytne aj vo výraze `výraz2`, nahradíme ho príslušným znakom/reťazcom.

Aby sme mohli vo výraze používať i znaky `?`, `*`, `{` a `}` budeme ich písať ako `\?`, `\*`, `\}` a `\{`. Potom znak `\` musíme písať ako `\\`.

Stačí, aby váš preprocesor fungoval pre korektné vstupy. Môžete teda predpokladať, že priradenie znakov a reťazcov symbolom `?X`, `*X` bude vždy jednoznačné a že pri nahrádzaní výrazov nikdy nedôjde k zacykleniu.

Takže napríklad kód:

```
\def\a(?X,*Y){a(?X,\a(*Y))}\\\def\a(?X){?X}\a(1,2,3,4,5,6)
```

bude preložený preprocesorom ako

```
\a(1,a(2,a(3,a(4,a(5,6)))))
```

Naprogramujte vyššie popísaný preprocesor a zamyslite sa nad tým, ako by sa dal tento preprocesor zdokonaliť.

1211. O Martowovi

Raz Matematicko-fyzikálnu fakultu navštívil len tak mimochodom mimozemšťan Martow a zanechal tam na pamiatku n-rozmernú kocku. Študenti sa rozhodli ju dôkladne preskúmať (to znamená každý jej vrchol), ale, ako to už s n-rozmernými kockami býva, veľmi rýchlo stratili orientáciu, ukazovali jeden cez druhého a hádali sa, ktorý vrchol už skúmali a ktorý ešte nie.

Preto sa rozhodli, že kocku budú skúmať systematicky: dôležité je, aby preskúmali skutočne všetky vrcholy, každý práve raz. Aby sa v kocke nestratili, musia vždy na ďalší skúmaný vrchol prejsť prstom po hrane.

Úloha: Napíšte program, ktorý načíta číslo n a vypíše postupnosť súradníc vrcholov v takom poradí, ako ich môžu študenti skúmať.

Taká n-rozmerná kocka sa skladá z 2^n vrcholov so súradnicami $(x_1, x_2, \ldots, x_n)$, kde x_i je 0 alebo 1. Dva vrcholy sú spojené hranou práve vtedy, keď sa ich súradnice líšia práve v jednej zložke. Typickým príkladom n-rozmernej kocky je štvorec ($n = 2$).

1212. O brčkavých zátvorkách

„Vnorené!“ kričal Brutus ako zmyslov zbavený. „Nevnorené!“ oponoval Frutus. „Ty sám si nevnorené“ Brutus na to. „Tvoju hlavu plešivú!“ povedal Frutus. Medzitým kamaráti vyšli pred krčmu a Frutus vybil Brutovi zub.

O čom sa to Brutus a Frutus tak vášnivo rozprávali? Hádali sa, aké zátvorkovanie v komentároch používa nová verzia ω-pascalu v.8.3β. V kompilátore s nevnoreným zátvorkovaním je za poznámku považovaná časť od ľavej komentárovej zátvorky „{“ až po najbližšiu pravú komentárovú zátvorku „}“ – takto sa to napríklad kompiluje v Turbo Pascale. V kompilátore s vnoreným zátvorkovaním je za poznámku považovaná časť od ľavej komentárovej zátvorky po jej zodpovedajúcu pravú komentárovú zátvorku – takýmto spôsobom skompilujeme ako poznámku napríklad toto: {{{Hello world!}}}.

Úloha: Napíšte program, ktorý môže byť skompilovaný oboma druhmi kompilátorov Pascalu a po spustení vypíše `ÁNO`, ak bol skompilovaný kompilátorom s vnoreným zátvorkovaním a `NIE`, ak bol skompilovaný kompilátorom s nevnoreným zátvorkovaním.

1213. O Pažravom Riškovi

Pažravý Riško je na prázdninách u svojej svokry na Jahodníku. Keďže na Jahodníku rastie veľa jahôd, má svokra v komore na polici $n \times n$ jahodových kompótov rôznych objemov poukladaných do štvorca. Riško chce zjesť čo najviac jahôd z kompótov (je lenivý si ísť nazbierať jahody sám) tak, aby si to svokra nevšimla (toto je možné zabezpečiť tak, že z každého štvorca 2×2 kompótov bude zjedený najviac jeden kompót). Teraz chce vedieť, koľko najviac jahôd môže zjesť.

Úloha: Zostavte program, ktorý načíta n a pole $n \times n$ objemov kompótov a vypíše maximálny objem kompótov, ktoré môže Riško zjesť tak, aby si to svokra nevšimla.

1214. O ovocnom sade

Santo a Banto vlastnia parcelu rozmerov $n \times n$, na ktorej rastú jablone a hrušky. Po veľmi krutej hádke sa rozhodli o parcelu súdiť. Sudca spravodlivo rozhodol, že si majú svoju

parcelu rozdeliť na dve súvislé časti rovnaké tvarom aj obsahom tak, aby na Santovej časti rástli iba jablone a na Bantovej iba hrušky (lebo Santo má rád jablkovicu a Banto zasa hruškovicu). Santo a Banto, ako obvykle, vymeriavali, zatĺkali kolíky, ale doteraz sa im parcelu rozdeliť nepodarilo.

Úloha: Napíšte program, ktorý načíta n a pole čísel $n \times n$, v ktorom jednotky reprezentujú jablone, dvojky hrušky a nuly nevysadené miesta a vypíše pole čísel $n \times n$, pričom jednotky budú reprezentovať Santove pozemky a dvojky Bantove pozemky.

Časť je súvislá, keď medzi každými jej dvoma políčkami i, j existuje cesta $v_1, v_2, \ldots, v_k$ taká, že $v_1 = i$ a $v_k = j$ a pre všetky $l = 1, \ldots, k-1$ políčka v_l a v_{l+1} susedia hranou.

1215. O kvalitnej tlačiarni

V softférovom družstve SoDr si kúpili novú tlačiareň. Vyznačuje sa tým, že tlačí veľmi kvalitné písmenká, čo vyžaduje veľa pamäti a preto si dokáže zapamätať len tvary najviac k písmeniek. Ak teda tlačia na takejto tlačiarni nejaký súbor, musia veľmi často použiť operáciu *Download*(x, y), kde x je miesto v pamäti tlačiarne $(1 \ldots k)$, kam sa má uložiť tvar písmenka y (písmenko, ktoré bolo na tomto mieste sa automaticky vymaže z pamäti tlačiarne). Ak potom tlačiareň má vytlačiť nejaké písmenko, vie, kde ho v pamäti nájde, dôležité je však, aby tam bolo (v opačnom prípade sa tlačiareň zasekne). Na začiatku je v pamäti náhodný obsah (a teda ho nemožno nijakým spôsobom využiť). Operácia *Download* je veľmi pomalá, a preto je potrebné jej použitie minimalizovať.

Úloha: Napíšte program, ktorý načíta číslo k a meno súboru a vytlačí tento súbor na tlačiarni družstva SoDr. Predpokladajte, že máte už predprogramovanú procedúru *Download* tak, ako je popísaná vyššie.

Pokúste sa zamyslieť a napísať aj niečo o tom, ako efektívne (čo sa týka počtu *Download*ov) váš program pracuje pre rôzne typy súborov.

1221. O výskumnom ústave

Santova manželka, Santovka, pracuje vo Výskumnom ústave pre označovanie výrobkov. Jej najnovší výskum sa zaoberá novým druhom zariadenia, ktoré na značený výrobok tlačí n-ticu rôznych číslic vždy v inom poradí. Prístroj ešte nie je úplne automatizovaný. Preto musí Santo, ktorý ho skúša, meniť poradie číslic ručne. Santovka prikázala Santovi, aby do večera vytlačil na papier všetky možné permutácie, ktoré môže stroj vytlačiť (ak chce dostať večeru). Santo je veľmi lenivý a rozhodol sa, že dve bezprostredne za sebou vytvorené permutácie sa budú líšiť jednou výmenou susedných prvkov. Teraz sedí na zemi a rozmýšľa, ako to môže urobiť.

Úloha: Napíšte program, ktorý pre dané n vypíše všetky permutácie n prvkov tak, aby sa každé dve susedné permutácie líšili jednou výmenou dvoch susedných prvkov.

1222. O indiánskom nárečí

V Brazílskom pralese žije skupina Indiánov, ktorých jazyk je naozaj veľmi zvláštny. Dve slová sú považované za zhodné, ak je jedno cyklickým posunom druhého. Napríklad ABC a BCA sú zhodné slová, ale ABC a CBA nie sú. Jazykovedec sa ho rozhodol naučiť. Najväčšie problémy má so zisťovaním, či sú dve slová zhodné.

Úloha: Napíšte program, ktorý načíta dve slová a zistí, či sú v jazyku Indiánov zhodné, alebo nie.

1223. O PNI

V podniku na integráče (PNI) vyrábajú rôzne obvody, ktoré sú charakterizované počtom vstupných nožičiek a počtom výstupných nožičiek. Dva integráče možno zospájkovať do jedného (vznikne nám nový integráč), keď prvý má rovnaký počet výstupných nožičiek ako druhý vstupných nožičiek. Inžinieri zistili, že na overenie správnosti zospájkovania treba pre každú dvojicu vstupná nožička nového integráču – výstupná nožička nového

integráču odmerať napätie na všetkých spojoch. Jedno odmeranie trvá šikovnej meračke jednu sekundu. Celkovo, ak mal prvý integráč I_1 vstupných a O_1 výstupných nožičiek a druhý integráč $I_2(= O_1)$ vstupných a O_2 výstupných nožičiek, bude zisťovanie správnosti trvať $I_1 \cdot O_1 \cdot O_2$ sekúnd. V podniku podľa objednávky pospájajú zákazníkovi n vhodných integráčov zaradom do jedného v požadovanom poradí, pričom ich spájkujú vždy po dvojiciach a pri spájkovaní overujú správnosť. Vo výslednom produkte musia byť integráče v požadovanom poradí, nie je však dôležité, v akom poradí ich spájkovali. Vhodné poradie spájkovania môže výrazne znížiť čas potrebný na testovanie správnosti spojov.

Úloha: Napíšte program, ktorý pre zadané n načíta počty vstupných nožičiek integráčov $I[1\,..\,n]$ počty výstupných nožičiek integráčov $O[1\,..\,n]$ a vypíše poradie, v akom treba integráče spájať, aby bol čas potrebný na testovanie minimálny. Zistite aj tento čas.

Príklad: Pre vstup: $n = 6$, $I = [4, 2, 3, 1, 2, 2]$, $O = [2, 3, 1, 2, 2, 3]$ je výsledkom takáto postupnosť spájaní:

Spoj integráče B a C.
Spoj integráče A a BC.
Spoj integráče D a E.
Spoj integráče DE a F.
Spoj integráče ABC a DEF.
Na testovanie bolo treba 36 sekúnd.

1224. O burundijských stavbároch

Po úmrtí starého kráľa nastal v krajine zmätok a mocné rody zápasili o trón. Po dlhotrvajúcich bojoch sa v krajine stal novým kráľom Ubugulundibumbuluku. Pri bojoch sa zničili všetky cesty medzi mestami. Ubugulundibumbuluku sa rozhodol, že vybuduje novú cestnú sieť, pričom presne určil, koľko ciest má do ktorého mesta viesť (podľa toho, ako mu miestne rody pomáhali pri ťažkom boji). Medzi každými dvoma mestami však má viesť najviac jedna priama cesta. Ešte chce, aby sa dalo dostať z každého mesta do každého. Stavbári teraz rozmýšľajú, ako dané cesty postaviť, aby splnili kráľove podmienky. Keby aspoň vedeli, či sa to dá, hneď by sa im ľahšie rozmýšľalo. Preto si kúpili počítač a chcú program, ktorý im to zistí.

Úloha: Napíšte program, ktorý pre zadané n a počty ciest, ktoré majú viesť do jednotlivých miest zistí, či je možné postaviť cestnú sieť tak, aby spĺňala kráľove požiadavky.

1225. O počítačoch na mieru

V softférovom družstve SoDr sa okrem iného vyrábajú aj tzv. počítače na mieru. Zákazník príde, povie si, čo by mal jeho počítač vedieť robiť, SoDráci napíšu potrebný softfér, nainštalujú ho na počítač a pošlú ho zákazníkovi.

Aby bol taký počítač čo najlacnejší, musí mať práve toľko pamäti, koľko dané programové vybavenie potrebuje (inak by to buď nefungovalo, alebo by počítač bol zbytočne drahý). Preto SoDráci pri každom svojom programe potrebujú zistiť, koľko vlastne pamäti potrebujú premenné v programe.

Úloha: Napíšte program, ktorý spočíta veľkosť pamäti, ktorá sa použije na definíciu jednej premennej. Definícia premennej je zapísaná v takejto forme:

⟨id⟩ je sekvencia písmen dlhá najviac 255 písmen
⟨integer⟩ je celé číslo medzi −10000 a 10000

⟨def⟩ ::= ⟨id⟩ : ⟨type⟩
⟨type⟩ ::= ⟨record⟩ | ⟨array⟩ | ⟨string⟩ | ⟨enum⟩ | ⟨range⟩
⟨record⟩ ::= { ⟨data⟩$^+$}
⟨array⟩ ::= **array** [⟨range⟩] **of** ⟨type⟩
⟨string⟩ ::= **string** (⟨integer⟩)
⟨enum⟩ ::= (⟨id-list⟩)

⟨id-list⟩ ::= ⟨id⟩ | ⟨id⟩ , ⟨id-list⟩
⟨range⟩ ::= [⟨integer⟩ . . ⟨integer⟩]

Samozrejme, počet bielych znakov[18] medzi jednotlivými skupinami písmen nehrá žiadnu úlohu[19].

Počet bitov potrebných na uloženie sa počíta nasledovne:

record	súčet veľkostí jednotlivých položiek
array	počet prvkov poľa násobený počtom bitov potrebných na uloženie ich typu
string	dĺžka vynásobená siedmymi
enum	najmenší počet bitov, ktorými môžeme rozlíšiť všetky *id*
range	najmenší počet bitov, ktorými môžeme rozlíšiť všetky hodnoty z daného intervalu

Vstupom do programu je definícia premennej (⟨def⟩). Vstup je napísaný správne, netreba kontrolovať jeho syntaktickú korektnosť. Výstupom z programu je počet bitov potrebných na uloženie danej premennej.

Príklad:
Vstup: *year* : [1970 . . 2030]
Výstup: 6
Vstup:

```
team : {
         meno : string(14)
         clenovia : array[1 . . 3] of {
                                         pohlavie : (muz, zena)
                                         meno : string(20)
                                         vek : [16 . . 30]
                                       }
         umiestnenie : [1 . . 40]
       }
```

Výstup: 539
Vstup: *menaporotcov* : **array**[1 . . 3] **of string**(20)
Výstup: 420

1231. Brutus, Frutus a partície

„6+3+1!“ kričal Brutus, ako zmyslov zbavený. „6+2+2!“ oponoval Frutus. „Ty sám si 6+2+2“ Brutus na to. „Tvoju hlavu plešivú!“ povedal Frutus. Medzitým kamaráti vyšli pred krčmu a Brutus vybil Frutovi zub.

O čom sa to Brutus a Frutus tak vášnivo rozprávali? Hádali sa, ako vyzerá desiata partícia z čísla 10 v Peknom usporiadaní. Partíciou nazveme každý rozklad čísla na prirodzené sčítance usporiadané nerastúco (napríklad $10 = 6 + 3 + 1$). Pekným usporiadaním partícií nazveme také usporiadanie, kde partícia $a_1 + a_2 + \cdots + a_k$ je pred partíciou $b_1 + b_2 + \cdots + b_l$ práve vtedy, keď pre nejaké u platí:

$$a_1 = b_1, \quad a_2 = b_2, \quad \ldots, \quad a_u = b_u, \quad a_{u+1} > b_{u+1}$$

Napríklad partície čísla 6 zoradené v Peknom usporiadaní sú: 6, $5+1$, $4+2$, $4+1+1$, $3+3$, $3+2+1$, $3+1+1+1$, $2+2+2$, $2+2+1+1$, $2+1+1+1+1$, $1+1+1+1+1+1$ (teda pravdu mal v tomto prípade Frutus).

---

[18] medzera, nový riadok, tabulátor
[19] Ak sa vám táto definícia zdá príliš zložitá, porovnajte ju s definíciou premenných v Pascale.

Úloha: Napíšte program, ktorý načíta čísla n a k a vypíše k-tu partíciu čísla n v Peknom usporiadaní.

1232. O Santových kartičkách

Kedysi dávno starý PrasaNto pre účely rodinnej hry vlastnoručne vystrihol 32767 kartičiek a napísal na ne čísla od 1 po 32767 (na každú iné). Už-už sa chcel pustiť do hry, keď tu zrazu zafúkal vietor a všetky kartičky rozfúkal po okolí. PrasaNto ich rýchlo pozbieral a keď ich spočítal, zistil, že jedna chýba.

Odvtedy sa vždy v sobotu večer zíde celá rodina u Santovcov na verande a pokúšajú sa zistiť, ktorá kartička sa stratila, aby ju mohli znovu napísať a konečne si zahrať starú rodinnú hru. Santovi to samozrejme už lezie na nervy a do peňaženky (keďže vždy príde aj Banto a stiahne mu celý bar), tak chce tomuto zvyku učiniť rázny koniec.

Úloha: Na vstupe je 32766 rôznych celých čísel z rozsahu od 1 po 32767. Napíšte program, ktorý načíta tieto čísla a v čo najkratšom čase a s čo najmenšou pamäťou nájde chýbajúce číslo, t.j. to číslo zo spomínaného rozsahu, ktoré sa medzi číslami na vstupe nevyskytuje.

1233. O kiribatských aerolíniách

Na Kiribatských ostrovoch sa rozhodli zanechať plavbu na trstinových lodičkách a prepojiť ostrovy leteckým spojením. Na každom ostrove vybudovali letisko. Problém je však v tom, že priame spojenia sú obmedzené dĺžkou doletu lietadla. Lietadlo síce môže urobiť medzipristátie na nejakom inom letisku, natankovať a pokračovať ďalej, ale týmto spôsobom sa predlžuje letová dráha (lietadlo medzi dvoma letiskami letí vždy po priamke, letová dráha medzi dvoma letiskami je ich priama vzdušná vzdialenosť).

Úloha: Zostavte program, ktorý načíta súradnice jednotlivých letísk, dĺžku doletu lietadla a dve letiská s a t a vypíše najkratšiu možnú letovú dráhu z s do t (t.j. postupnosť všetkých medzipristátí lietadla).

1234. O modrých prilbách v Burundi

Keďže nárok kráľa Ubugulundibumbuluku na trón bol od začiatku veľmi pochybný, vytvorila sa v meste Bujumbura skupina obyvateľov, ktorí si zvolili vlastného prezidenta Ntybantuganyu a začali vytvárať územie novej Burundijskej republiky. Územie republiky má tvar mnohouholníka (nie nutne konvexného!).

Organizácia spojených národov sa v rámci nezasahovania do vnútorných záležitostí suverénnej krajiny rozhodla vyslať na územie Burundi špeciálne vycvičenú parašutistickú jednotku modrých prilieb. Keď takýto parašutista zoskočí na zem, musí rýchlo zistiť, či sa nachádza na území republiky, alebo kráľovstva.

Úloha: Napíšte program, ktorý na vstupe dostane mnohouholník (súradnice jeho vrcholov v protismere hodinových ručičiek) a súradnice niekoľkých miest, do ktorých pristáli parašutisti a vypíše pre každé takéto miesto, či leží vnútri daného mnohouholníka, alebo nie.

1235. O okienkovom systéme

V softférovom družstve SoDr majú nového zákazníka, ktorý si kúpil počítačovú sieť so zbierkou najrôznejších terminálov najrôznejšieho typu, rozmeru a veku. Programátorom sa horko-ťažko podarilo naprogramovať niekoľko základných výstupných operácií tak, aby boli použiteľné na všetkých termináloch. To však nestačí, produkty softférového družstva SoDr sa totiž vyznačujú dobrým užívateľským rozhraním.

Úloha: Napíšte pre softférové družstvo knižnicu pre prácu s oknami. Vaša knižnica by mala zvládnuť tieto funkcie:

Vytvorenie okna – vytvorí prázdne okno (t.j. žiadne rámčeky) v stanovenom obdĺžniku na obrazovke terminálu. Novovytvorené okno je vždy aktívne. Nezabudnite, že sa pri tom môžu prekryť nejaké časti už vytvorených okien!

Zaktívnenie okna – dané okno sa „vysunie“ na povrch (to znamená, že žiadna jeho časť nebude zakrytá iným oknom).

Zápis znaku do aktívneho okna – na dané súradnice v okne (vzhľadom k ľavému hornému okraju) sa zapíše znak. Môžete počítať s tým, že sa výstup na obrazovku bude vykonávať iba pomocou tejto operácie.

Zatvorenie aktívneho okna – aktívne okno sa zruší (zmaže sa z obrazovky a obnovia sa všetky časti iných okien, ktoré zrušené okno zakrývalo).

Pre prácu s terminálom môžete použiť len tieto procedúry a funkcie. Deklarácie sú uvedené v jazyku Pascal, ak by ste programovali v inom jazyku, použite podobné:

procedure *Write*(*ch*:*char*) – vypíše znak *ch* na aktuálnu pozíciu kurzora a posunie ho na nasledujúci stĺpec (prípadne prvý stĺpec nasledujúceho riadku)

procedure *GotoXY*(*x*,*y*:*byte*) – presunie kurzor na riadok y a stĺpec x

function *GetRows*:*byte* – vráti počet riadkov terminálu

function *GetColumns*:*byte* – vráti počet stĺpcov terminálu

Nezabudnite, že ku knižnici treba dodať aj podrobný popis fungovania jednotlivých procedúr a použitých dátových štruktúr (teda nielen ich použitia).

1241. O blšom tanci

Riaditeľ Hilbertovho cirkusu sa rozhodol rozšíriť program o nové číslo. Na konkurze zvíťazil Blší tanec. Tento tanec tancuje naraz n blšiek, označených číslami 1 až n, ktoré sú rozostavené na políčkach očíslovaných od 1 po n. Z každého políčka vedie práve jedna šípka na nejaké políčko, pričom do každého políčka vedie práve jedna šípka.

Na začiatku každá blška sedí na políčku s rovnakým číslom, ako má ona. Potom v každej sekunde preskočia blšky (všetky naraz) po šípkach na svoje nové políčka. Tanec končí, keď sú blšky rozmiestnené tak, ako na začiatku. Riaditeľ potrebuje, aby číslo trvalo najdlhšie ako sa dá, a preto treba vhodne nakresliť šípky.

Úloha: Napíšte program, ktorý pre dané n vypíše, ako treba nakresliť šípky, aby tanečné číslo trvalo najdlhšie ako sa dá. Program by mal tiež vypísať, ako dlho bude číslo trvať.

Príklad: Pre $n = 5$ je výsledok 1–2 2–1 3–4 4–5 5–3 a číslo bude trvať 6 sekúnd. Pre $n = 8$ je výsledok 1–2 2–3 3–1 4–5 5–6 6–7 7–8 8–4 a číslo bude trvať 15 sekúnd.

1242. O továrni na ceruzky

V továrni na ceruzky HOK-NI-RO vyrábajú veľmi kvalitné ručne vyrezávané ceruzky rôznych dĺžok. Potom ich balia po n kusoch, pričom v balení sú ceruzky vždy usporiadané podľa dĺžky. V rámci automatizácie kúpili do továrne triedičku. Triedička je vybavená pravítkom na meranie dĺžok ceruziek a rukou, ktorá umožňuje vymeniť dve ceruzky. Problém je v tom, že ruka je poškodená a dokáže vymieňať iba dve také ceruzky, medzi ktorými je práve jedna iná ceruzka. V továrni potrebujú vedieť, či pre danú skupinu ceruziek sa tieto dajú usporiadať na triedičke.

Úloha: Napíšte program, ktorý pre dané n a n dĺžok ceruziek vypíše, či sa tieto dajú usporiadať na triedičke.

Príklad: Pre $n = 4$ a dĺžky ceruziek $2, 3, 1, 4$ sa ceruzky nedajú usporiadať. Pre $n = 3$ a dĺžky 333, 247, 109 sa ceruzky dajú usporiadať.

1243. O veľkej poľovačke

Rodina Santovcov sa pohádala s rodinou Bantovcov kvôli portrétu starej Santovky. Rodiny vytiahli staré pušky a nastala poľovačka.

Situácia v meste sa stala natoľko neprehľadná, že šerif o občanoch nevie, do ktorej rodiny patria. Aby to zistil, poslal svojich pomocných šerifov na pozorovacie stanovištia, aby mu odtiaľ posielali informácie, kto na koho vystrelil (nemusel nutne trafiť). Z týchto

informácii chce šerif zistiť príslušnosť jednotlivých občanov k rodinám. Šerif vie, že konfliktu sa zúčastňujú iba dve rodiny a že vo vnútri rodiny nie sú žiadne spory (t.j. členovia jednej rodiny na seba nestrieľajú).

Úloha: Napíšte program, ktorý načíta n dvojíc mien (každá dvojica mien obsahuje informáciu, kto na koho vystrelil) a vypíše rozdelenie občanov do rodín. V prípade, že informácie sú nedostačujúce na jednoznačné zaradenie členov do rodín, program vypíše: `MÁLO INFORMÁCIÍ`. V prípade, že sa podľa daných informácií občania do dvoch rodín rozdeliť nedajú, program vypíše: `CHYBNÉ INFORMÁCIE`.

Príklad:

Vstup:
```
3
Santo Banto
Banto Jim
Jim Santo
```
Výstup:
```
CHYBNÉ INFORMÁCIE
```

Vstup:
```
2
Santo Banto
Jim Bill
```
Výstup:
```
MÁLO INFORMÁCIÍ
```

Vstup:
```
5
Mary Santo
Bill Banto
John Jim
Bill John
Mary Bill
```
Výstup:
```
Santo, Jim, Bill
vs.
John, Mary, Banto
```

1244. O n-trise

Programátor Ó Veľký sa rozhodol napísať modifikáciu známej hry tetris. Od pôvodného tetrisu sa nová verzia bude líšiť v tom, že padajúce dieliky sa nebudú skladať zo štyroch, ale z n štvorčekov. Teraz už tretí deň sedí a na štvorčekový papier kreslí všetky možné dieliky, ktoré sa dajú zložiť z n štvorčekov a stále si nie je istý, či ich má všetky.

Úloha: Napíšte program, ktorý pre dané n vypíše všetky dieliky, ktoré sa dajú zložiť z n štvorčekov a ich počet. Dieliky líšiace sa len rotáciou považujeme za rovnaké.

Príklad: Pre $n = 4$ program vypíše:
```
##  #      #   ##  ##    #
##  ###  ###  ##    ##  ###  ####
Existuje 7 rôznych dielikov.
```

1245. O veľkých číslach

Raz si u softférového družstva SoDr pracovníci Matematicko-matematickej fakulty Matematickej univerzity objednali program, ktorý by im umožnil pracovať s veľkými číslami. Čísla, s ktorými chcú pracovať, nie sú dlhšie ako 257 cifier, môžu byť kladné aj záporné.

Úloha: Napíšte pre pracovníkov MMF MU knižnicu, ktorá im umožní pracovať s celými číslami, ktorých desiatkový zápis môže byť dlhý až 257 cifier. Vaša knižnica musí obsahovať tieto procedúry a funkcie (deklarácie sú uvedené pre jazyk Pascal, ak by ste programovali v inom jazyku, použite podobné):

$GAdd(a, b, c, err)$ – sčítanie ($c := a + b$)
$GSub(a, b, c, err)$ – odčítanie ($c := a - b$)
$GMul(a, b, c, err)$ – násobenie ($c := a \times b$)
$GDiv(a, b, c, r, err)$ – celočíselné delenie ($c := a \text{ div } b$; $r := a \bmod b$)

$GAbs(a, c)$ – absolútna hodnota čísla ($c := \text{abs}(a)$)
$GInput(a, err)$ – vstup veľkého čísla a
$GOutput(a)$ – výstup veľkého čísla a

Vo všetkých prípadoch premenná *err* po návrate z procedúry obsahuje nulu, ak operácia prebehla v poriadku a kód chyby, ak počas operácie nastala nejaká chyba (napríklad: výsledok presiahol rozsah veľkých čísel a pod.)

1311. O kiribatských klebetniciach

Kiribatské ženičky rady klebetia. Problém je v tom, že keď si chce jedna Kiribatčanka poklebetiť s inou, zväčša sa musí doplaviť na ostrov, kde býva.

Žiaľ, iba medzi niektorými dvojicami ostrovov funguje kyvadlová prievoznícka doprava, a tak ženička v túžbe po nových zaujímavých informáciách musí často aj niekoľkokrát prestupovať. Navyše niektorí prievozníci sú strašne leniví, a tak je ich loďka tak pomalá, že ženička zvolí radšej inú cestu aj za cenu prestupovania. Ženičky by potrebovali takú tabuľku, ktorá by povedala, ako dlho trvá najkratšia možná cesta pre každú dvojicu ostrovov.

Úloha: Napíšte program, ktorý načíta počet ostrovov n a zoznam kyvadlových „liniek“ (každá linka je určená dvoma číslami ostrovov a časom prievozu) a vypíše tabuľku, v ktorej pre každú dvojicu ostrovov bude uvedený čas najkratšej možnej cesty. Predpokladajte, že čas potrebný na čakanie na kyvadlovú dopravu je zanedbateľný. Ostrovy sú očíslované od 1 do n.

Príklad: Pre $n = 3$ a linky $(1, 2, 10)$ $(1, 3, 34)$ $(2, 3, 100)$ je výsledok:

```
 0 10 34
10  0 44
34 44  0
```

1312. O SoDr-e

V softférovom družstve SoDr majú opäť kopec problémov s novým zákazníkom. Ten si kúpil počítač, ktorý mal k dispozícii iba jedinú premennú, do ktorej je možné uložiť jediný znak.

Zákazník chcel umiestniť počítač na hranicu. Keď prejde niekto z jednej strany hranice na druhú, počítač dostane na vstup znak `A`, ak prejde z druhej strany na prvú, počítač dostane na vstup znak `B`. Ak je večer na vstupe rovnaký počet písmen `A` aj `B`, každý je doma a všetko je v poriadku. V opačnom prípade je potrebné urýchlene vyhlásiť poplach.

Úloha: Na vstupe je postupnosť znakov `A` a `B` ukončená medzerou. Napíšte program, ktorý zistí, či je v postupnosti rovnaký počet písmen `A` aj `B`. Nezabudnite, že váš program smie používať jedinú premennú typu **char** a aj jej obsah možno meniť iba načítaním novej hodnoty zo vstupu.

1313. O malom lenivom krtkovi II

Malý lenivý krtko II spal v brlôžku až do večere. Jeho mamičku to nazlostilo, a tak si povedala: „Čo z môjho synčeka vyrastie, keď bude taký lenivý!“ a už ho aj išla budiť. „Vstávaj, krtko II!“ rázne sa mu prihovorila, „ak dneska nepochytáš chrobáčikov vo všetkých našich brlôžkoch, tak ťa naskutku paličkou II vyobšívam!“ Čo mal chudák krtko II robiť, nechcelo sa mu, ale musel vstať.

Krtia rodina mala vybudovaných n brlôžkov, pričom každé dva brlôžky boli spojené tunelmi. Každý deň krtica s krtom preliezli všetky brlôžky a zbierali to, čo majú najradšej – chrobáčiky. Teraz mal aj malý lenivý krtko II zbierať. Ale keďže bol celkom inteligentný, najprv pouvažoval: „Najlepšie bude, keď cez každý brlôžtek prejdem práve raz a pôjdem po takých chodbách, že súčet ich dĺžok bude najmenší.“ Samozrejme, malý lenivý krtko II chcel svoju cestu skončiť opäť vo svojom brlôžku, aby mohol spať až do ďalšej večere.

Úloha: Napíšte program, ktorý načíta počet brlôžkov n a dĺžku tunela medzi každými dvoma brlôžkami a vypíše, aká dlhá bude cesta malého lenivého krtka II.

1314. O valcočlovekovi II

Valcočlovek je taký človek, ktorého tvar sa približuje tvaru valca s určitou výškou a polomerom. Výskyt ľudí takéhoto tvaru je veľmi vysoký napríklad medzi pracovníkmi francúzskych vínnych pivníc (čím viac vína pracovník skonzumuje, tým je širší). Francúzske vínne pivnice sú veľké podzemné sály tvaru obdĺžnika podopierané tenkými stĺpmi. Strany obdĺžnika sú orientované v smere svetových strán. Často sa stáva, že valcočlovek sa zasekne medzi stĺpy a vínna pivnica sa stane nepoužívateľnou (zaseknutý valcočlovek zle vplýva na kvalitu vína). Preto si valcočlovek musí vždy dobre rozmyslieť, cez ktorú pivnicu môže ísť.

Úloha: Napíšte program, ktorý načíta reálne rozmery pivnice m, n, počet stĺpov p a reálny polomer valcočloveka r. Potom načíta súradnice stĺpov (pričom $[0, 0]$ je juhozápadný roh pivnice, $[m, n]$ severovýchodný) a napíše, či môže tento valcočlovek prejsť pivnicou od západnej steny po východnú.

1315. Frutus, Brutus a zúrivé výrazy

„Napíšem!“ kričal Brutus jedného večera v útulnej krčmičke. „Nenapíšeš!“ oponoval Frutus. „A just napíšem!!“ nedal sa Brutus. „Ani keby si sa zderivoval!!!“ nedal sa presvedčiť Frutus.

Tentokrát sa hádali o tom, kto napíše zúrivejší výraz. Nakoniec svoj spor rozhodli stávkou. Každý napísal na servítku svoj výraz a začala sa hádka o tom, ktorý z nich je zúrivejší. Pri rozhodovaní musia vedieť, či náhodou tie výrazy nie sú rovnaké (v tom prípade by si asi navzájom vyrazili zuby).

Úloha: Napíšte program, ktorý načíta dva výrazy skladajúce sa z jednopísmenových premenných, celočíselných konštánt, znamienok $+$, $-$, $\cdot$ a zátvoriek (a) a vypíše, či sú ekvivalentné. Dva výrazy sú ekvivalentné, ak pre každú kombináciu hodnôt premenných nadobúdajú rovnakú hodnotu. Môžete predpokladať, že výrazy sú korektne zadané.

Príklad:

Vstup:	Výstup:
$A \cdot (A - B) + A \cdot C$ $A \cdot A - A \cdot (B - C)$	`ANO`

1321. O kráľovstvách na Kiribati

Kiribatské deti sa strašne radi hrávajú na kráľov a kráľovné – niet divu, veď kiribatské súostrovie má toľko ostrovov, že každý obyvateľ môže vládnuť aspoň jednému.

Raz sa im dostala do rúk mapa celého súostrovia. Každý ostrov bol zobrazený jednou bodkou. Deti sa začali hrať hru, pri ktorej si každý vybral kráľovstvo obdĺžnikového tvaru, ktorému by chcel vládnuť. Nezáležalo však na rozlohe kráľovstva, ale na počte ostrovov – ten kto ich má najviac, stane sa Najmocnejšo-Najhrôzostrašnejšo-Najsilnejším panovníkom.

Úloha: Napíšte program, ktorý načíta súradnice ostrovov na mape (reálne čísla). Ďalej načíta kráľovstvá, pričom kráľovstvo je dané súradnicami ľavého horného a pravého dolného rohu (celé čísla v rozsahu 0 až 100) obdĺžnika a pre každé kráľovstvo vypíše počet ostrovov, ktoré sa v ňom nachádzajú.

1322. Brutus a Frutus v knižnici

Brutus a Frutus po návšteve u zubára navštívili aj knižnicu. Vybrali si knihu „Zbierka úloh KSP“ a začali v nej listovať. Keď si konečne zo zbierky vybrali tú najzaujímavejšiu úlohu a chceli sa pozrieť na jej vzorové riešenie, zistili, že vzorové riešenie je z knižky vytrhnuté.

Úloha: Na vstupe je číslo n a n celých čísel z rozsahu od 1 do n^2. Utrieďte tieto čísla. Triedenie so zložitosťou $O(n \log n)$ nie je ani zďaleka najlepšie riešenie.

1323. Banto a dostavníky

Banto chce navštíviť starú mamu Prabantovku, ktorá býva až hen v Nalomenej Trieske. Preto musí vedieť, ako chodia v Nalomenej Trieske dostavníky. Prabantovka nevie, ktorý dostavník kedy a kam chodí, ale vždy, keď jej prehrkoce nejaký pod oknami, veľmi ju to rozruší, lebo dostavníky robia veľký hluk. Večer si Prabantovka všetky vzrušujúce veci (teda aj prechod dostavníka) zapíše do denníka. Chudák Banto by aspoň chcel vedieť, koľko rôznych liniek chodí cez Nalomenú Triesku. Preto požiadal Prabantovku, aby mu poslala výpis z denníka. Teraz sedí nad listom od starej mamy a nevie si rady.

Úloha: Napíšte program, ktorý načíta obsah denníka Prabantovky a vypíše, koľko najmenej dostavníkových liniek chodí cez Nalomenú Triesku.

Výpis z denníka Prabantovky obsahuje začiatok sledovaného obdobia, počet dostavníkov, ktoré za sledované obdobie prešli popred Prabantovkine okná a ďalej pre každý dostavník deň, kedy prešiel popod okná. Prabantovka zásadne nepíše dátumy, ale počet dní od jej narodenia. Sledované obdobie trvá 60 dní. Zápisy sú chronologicky usporiadané.

Dostavníková linka sa vyznačuje tým, že dostavníky chodia v pravidelných intervaloch počas celého sledovaného obdobia. Každá linka prejde popod Prabantovkine okná počas sledovaného obdobia aspoň dvakrát. Môžete predpokladať, že cez Nalomená Triesku nechodí viac ako 17 liniek a že za sledované obdobie neprešlo popod Prabantovkine okná viac ako 300 dostavníkov.

Príklad:

Vstup:

```
25500 17
25500 25503 25505 25513 25513 25515 25521 25526 25527 25529
25537 25539 25539 25545 25551 25552 25553
```

Výstup:

```
3 linky
(s intervalmi 13, 12 a 8 dní, prvý dostavník z linky ide v deň
25500, 25503 a 25505)
```

1324. O vikingskom cestovateľovi

V roku 1101 sa vikingský cestovateľ Uwe Pisgarson doplavil až k brehom Južnej Ameriky. Len čo vystúpil zo svojej lode, zajali ho domorodci a chystali sa ho obetovať (t.j. obedovať). Po dlhom prosení ich Uwe presvedčil, aby mu dali ešte jednu šancu. Náčelník ho zaviedol do obrovskej miestnosti, v ktorej bol na stene veľký ťažký kamenný kotúč s kolíkmi po obvode pripomínajúci lodné kormidlo. Kotúč sa dal otáčať oboma smermi. Najspodnejší z kolíkov bol natretý červenou farbou. Na niektorých kolíkoch boli pripnuté zlaté disky, zvyšné disky ležali na zemi. „Pozri!“ povedal náčelník Uwemu a ukázal na mozaiku na protiľahlej stene. Znázorňovala rovnaký kotúč ako v miestnosti, červený kolík opäť smeroval nadol, iba disky boli inak rozmiestnené. „Ak do východu Slnka budú na obidvoch stenách rovnaké kotúče, si voľný, inak...“

Pisgarson si pozornejšie obzrel kotúč. Zistil, že ho vládze otočiť o jedno miesto v oboch smeroch, alebo pripnúť (alebo odobrať) disk na spodný kolík (na iné kolíky nedočiahol).

Len čo osamel, dal sa do práce. Začal rýchlo otáčať kotúč, pripínať a odoberať disky. Kotúč bol však veľmi ťažký, a tak sa Uwe rýchlo unavil. Nad ránom od vysilenia odpadol a úlohu nesplnil.

Úloha: Napíšte program, ktorý zachráni úbohého Uwe Pisgarsona pred krutou smrťou. Váš program načíta počet kolíkov n, potom reprezentáciu kotúča, t.j. postupnosť núl a jednotiek (kolíky bez disku a kolíky s diskom), počnúc červeným kolíkom v smere hodinových ručičiek a nakoniec reprezentáciu mozaiky (rovnakým spôsobom ako kotúč).

Výstup bude postupnosť písmen `L`, `P`, `V` (otočiť o jedno miesto v smere hod. ručičiek, proti smeru chodu hodinových ručičiek, výmena disku na spodnom kolíku), ktorá zabezpečí, že Uwe splní úlohu a nezomrie od únavy (t.j. dĺžka výstupnej postupnosti má byť minimálna).

Kotúč je veľmi ťažký a Uwe zomrie veľmi rýchlo. Diskov je k dispozícii dostatočný počet.

1325. O politikoch

Politici to vôbec nemajú jednoduché. Celé dni musia vysedávať na schôdzach parlamentu, zasadnutiach rôznych komisií a výborov, podchvíľou sa koná nejaká tlačová konferencia, alebo sa novinári snažia získať interview. Ale to všetko by sa ešte dalo vydržať. Najhoršie je, že taký politik musí veľmi často písať politické prejavy. Je to horšie ako domáce úlohy zo slohu.

Veľmi dôležité je, aby mal prejav požadovanú dĺžku, pričom dĺžka prejavu sa meria v počte riadkov. Všetky riadky (okrem posledného) obsahujú 60 písmen. Takisto by mal byť prejav zakaždým iný – ak by politik vždy použil ten istý prejav, hrozí, že si to za nejaký čas niekto všimne (je však veľmi pravdepodobné, že by to trvalo aspoň jedno volebné obdobie).

Úloha: Napíšte program, ktorý na vstupe dostane predpísaný počet riadkov politického prejavu a vypíše čo najpôsobivejší politický prejav predpísanej dĺžky. Vo svojom prejave môžete použiť nasledovné symboly: `<DP>` – dramatická pauza, `:-)` – podmanivý úsmev, `8-(` – zdravé rozhorčenie, či `<NSVPS>` – nadšené súhlasné výkriky z poslaneckej snemovne.

1331. O snaživom Tomášovi

Ako iste všetci viete, Tomáš je veľmi snaživý študent, a preto keď boli zverejnené rozvrhy, rozhodol sa, že si zapíše najviac prednášok ako len môže. Dlho dumal nad rozvrhom a nič. Veľmi sa preto trápi. Nikdy si nie je istý, či rozvrh, ktorý si vymyslel, má najviac prednášok ako len môže mať. Za týždeň už schudol 5 kíl. Pomôžte mu, než bude mať zápornú hmotnosť a niekam nám uletí.

Úloha: Napíšte program, ktorý načíta začiatky a konce prednášok a vypíše Tomášovi najväčší počet prednášok, ktoré si môže zapísať (samozrejme, nemôže prísť na dve prednášky naraz, ani prísť neskoro, či odísť priskoro z prednášky) spolu s príkladom takéhoto rozvrhu.

Každá prednáška na vstupe je v jednom riadku v tvare `Pon 07:20-09:45`. Dni v týždni sú `Pon`, `Uto`, `Str`, `Stv`, `Pia`, v sobotu a nedeľu prednášky nie sú. Na výstup vypíšte počet prednášok v rozvrhu a zoznam zapísaných prednášok, každú na samostatný riadok.

Nezabudnite na dôkaz správnosti algoritmu.

Príklad:

Vstup:	Výstup:
`Pon 07:20-09:45`	`2`
`Uto 09:30-10:00`	`Pon 07:20-09:45`
`Pon 09:00-10:35`	`Uto 09:30-10:00`

1332. O Števovej návšteve v Indii

Števo sa potuloval svetom a keď už bol v Ázii, spomenul si, že je hladný. Zastavil sa preto v dedine Maharázádžiád. Prišiel do obchodu a chcel si kúpiť veľa jedla na ďalšiu cestu. Keď už stál v rade, všimol si, že na pokladni je napísané „$1 + 1 = 10$“. Spočiatku nevedel, čo to znamená. No po chvíli mu bolo všetko jasné. „V tomto obchode počítajú v inej číselnej sústave!“ skríkol Števo. Tak bežal do ďalšieho. Tam počítali zasa v inej sústave. V každom obchode bol napísaný výraz tvaru $A + B = C$.

Števo nalistoval zodpovedajúcu stranu v svojej príručke mladých svišťov a zistil, že takýto výraz slúži na určenie číselnej sústavy, v ktorej sa v danom obchode počíta. Števo

nevie po indicky, preto v obchode nakupuje tak, že ukáže, čo chce a na papier napíše, koľko toho chce. Na to však potrebuje vedieť, v akej sústave sa v danom obchode počíta. Keď to nezistí, je odsúdený na smrť vyhladovaním.

Úloha: Napíšte program, ktorý zistí, v ktorých číselných sústavách platí $A + B = C$. Predpokladajte, že cifry sú len $0, 1, 2, \ldots, 9, a, b, \ldots, z$.

Vstup:	Výstup:
`1+1=10`	`2`

1333. O kiribatských náleziskách

Na Kiribatských ostrovoch nedávno objavili pod zemou veľké zásoby rumu. Inžinieri navrhli miesta, kde postaviť vrtné veže, problémom však je preprava rumu z týchto veží. Spoločnosť, ktorá rumové polia vlastní, sa rozhodla postaviť jednu veľkú rúru idúcu cez celé pole z východu na západ a od každej veže postaviť prípojku k tejto veľkej rúre. Veže s rovnakou x-ovou súradnicou môžu mať spoločnú prípojku. Kde ale postaviť veľkú rúru, aby sa na prípojky minulo čo najmenej materiálu? Nad tým treba veru triezvo pouvažovať!

Úloha: Napíšte program, ktorý načíta n – počet ťažných veží, súradnice jednotlivých ťažných veží (usporiadané podľa x-ovej súradnice) a vypíše y-ovú súradnicu, kam postaviť veľkú rúru tak, aby súčet dĺžok prípojok bol minimálny.

Príklad:

Vstup:	Výstup:

`5`
`10 1`
`10 10`
`10 7`
`15 20`
`20 17`
	`17`

1334. O telefónnom víruse

Vírus Prepheeckunecz, ktorý infikuje moderné telefóny, vždy v piatok poprehadzuje význam tlačidiel telefónu, t.j. napríklad po stlačení 1 sa vytočí číslo 5 a podobne, pričom je však telefón schopný vytočiť ktorékoľvek číslo.

Každý majiteľ telefónu má našťastie svoj súkromný telefónny zoznam, na základe ktorého môže zistiť, ako sú tlačidlá poprehadzované, a to tak, že vytáča rôzne čísla a podľa toho, kto sa mu ozve, zistí, aké číslo vlastne vytočil.

Úloha: Napíšte program, ktorý na čo najmenej vytočení zistí, ako sú poprehadzované tlačidlá telefónu. Program má k dispozícii funkciu *vytoc*(*st* : **string**) : **integer**, ktorá dostane na vstupe vytáčané číslo (v poprehadzovaných tlačidlách) a vráti buď číslo volanej osoby, alebo 0, ak osoba s takým (správnym) číslom nie je v súkromnom telefónnom zozname. Telefónny zoznam máte k dispozícii ako pole $TZ[1\,..\,n]$, kde $TZ[i]$ je (správne) telefónne číslo osoby i. Ak nie je možné zistiť poprehadzovanie tlačidiel, vypíšte o tom správu.

Príklad: Dajme tomu, že v telefónnom zozname máme čísla $[123, 456, 7890]$ a tlačidlá sú prehádzané takto: $(2 \to 0,\ 3 \to 1,\ 5 \to 2,\ 0 \to 3,\ 9 \to 4,\ 1 \to 5,\ 8 \to 6,\ 4 \to 7,\ 6 \to 8,\ 7 \to 9)$, t.j. ak stlačíme 2, vytočí sa 0, namiesto 3 sa vytočí 1, atď.

Potom volanie *vytoc*('350') vráti 1, *vytoc*('918') vráti 2, *vytoc*('4672') vráti 3 (v skutočnosti sa vytočia čísla 123, 456, 7890). Týmito troma volaniami vieme zistiť, ako sú všetky tlačidlá poprehadzované. Na druhej strane *vytoc*('123') vráti nulu, lebo v zozname nemáme nikoho s číslom 501.

1335. O SoDr a meteorológoch

V softférovom družstve majú najnovšie klientov – meteorológov. Taký meteorológ keď príde k počítaču, chcel by vedieť, v akom rozmedzí uhlov v posledných n minútach fúkal vietor. Výsledok merania smeru vetra je k dispozícii každú minútu (zadáva sa v stupňoch).

Rozmedzie uhlov možno určiť len ako súvislý oblúk. Oblúk je určený dvoma číslami a a b, ktoré určujú hranice oblúku. Orientácia oblúku je v protismere hodinových ručičiek od a k b. Napríklad oblúk $(0, 90)$ určuje štvrťkruh a oblúk $(90, 0)$ trištvrťkruh.

Úloha: Na vstupe sú čísla n, k a $n+k$ celých čísel z intervalu $[0, 360)$, ktoré sú výsledkami meraní postupne v časoch $1, 2, \ldots, n+k$. Napíšte program, ktorý vypíše rozmedzia uhlov pre časy $n, n+1, \ldots, n+k$ vždy za posledných n minút. Môžete predpokladať, že k je aspoň 3.

Príklad: Pre $n = 3$, $k = 3$ a merania: $0, 30, 60, 90, 120, 150$ je výsledkom `(0,60) (30,90) (60,120) (90,150)`.

1341. Brutus a Frutus v knižnici II

Brutovi a Frutovi sa v knižnici zapáčilo, a tak sa tam opäť vybrali (však koho by bavilo stále navštevovať zubára). Čírou náhodou si znovu vybrali knihu „Zbierka úloh KSP“ a začali v nej listovať. Keď si konečne zo zbierky vybrali druhú najzaujímavejšiu úlohu a chceli sa pozrieť na jej vzorové riešenie, zistili, že nejaký zurvalec pre istotu vytrhol všetky vzorové riešenia.

Úloha: V súbore je daných veľa čísel typu **longint**. Napíšte program, ktorý vytvorí súbor s tými istými číslami uloženými v poradí od najmenšieho po najväčšie. Čísel je tak veľa, že sa určite nezmestia do pamäti počítača. Predpokladajte, že na disku máte miesta dosť.

1342. O univerzitnom knihovníkovi

„Ooook!“ povedal knihovník, keď doniesli ďalších n kníh. V univerzitnej knižnici bolo doteraz n kníh od rôznych autorov z najrôznejších odvetví mágie. Ku každej knihe teraz priviezli druhý diel a uložili ich na policu za pôvodnými knihami. Privezené knihy sú pritom usporiadané rovnako ako pôvodné. Knihovník by mal rád banán (lepšie povedané, chcel by banán).

Úloha: V poli A veľkosti $2n$ (polica) sú uložené postupne čísla $a_1, \ldots, a_n$ (prvé diely kníh), $b_1, \ldots, b_n$ (druhé diely kníh). Napíšte procedúru, ktorá preusporiada pole A tak, aby v ňom boli prvky v poradí $a_1, b_1, a_2, b_2, \ldots, a_n, b_n$ (teda prvý a druhý diel vždy pri sebe). Nepoužívajte žiadne pomocné pole (knihovník nemá žiadnu ďalšiu „pomocnú“ policu) a nevyužívajte hodnoty prvkov v poli.

1343. O čarodejnom písacom stroji

Čarodejnica Magica má vo svojej kancelárii prečudesný písací stroj, ktorým si zapisuje svoje kúzla na netopierie kože. „Hlavná časť stroja som Ja – ľudská ruka s náramkom, na ktorom sú napísané písmenká (rovnaké písmenká môžu byť na náramku aj viackrát). Môžem sa točiť okolo osi určenej prostredníkom tak, že nad kožou sa objavujú iné písmenká, ktoré, ak čarodejnica stlačí červené tlačidlo, vytlačím. Som z dlhého písania veľmi unavená, a preto som sa rozhodla, že napíšem program, ktorý moju prácu uľahčí a pre dané poradie písmeniek na náramku a požadovaný text vypíše postupnosť krokov ako sa mám točiť, aby bol počet otočení minimálny. Keďže mi chýba hlava, mohli by ste mi pomôcť.“

Úloha: Napíšte program, ktorého vstupom je najprv postupnosť písmeniek na náramku (pričom prvé písmenko zodpovedá písmenku, ktoré je práve nad kožou) a potom v ďalšom riadku veta, ktorú treba napísať. Výstupom je postupnosť príkazov `DOLAVA`, `DOPRAVA`, `STLAC` a `MEDZERA` realizujúca danú vetu, alebo `FUJ`, ak sa veta nedá s daným náramkom napísať.

Príklad:

Vstup:
```
abacd
aba dac
```

Výstup:
```
STLAC DOPRAVA STLAC
DOLAVA STLAC MEDZERA
DOLAVA STLAC DOLAVA
DOLAVA STLAC DOPRAVA STLAC
```

1344. O kachličkovaní

Veveričkám Anke a Hanke v Hornom-Dolnom je v ich domčeku zima a preto si u slona objednali slonovinové kachličky obdĺžnikového tvaru rozmeru 1vn×2vn (1vn = jedna veverička nožička). Hrochovi na úrad priniesli obrázok svojej izbičky nakreslený na štvorčekovom papieri so štvorčekmi s hranou 1vn a chcú vedieť, či sa dá vykachličkovať. Hroch sa práve zobudil, a preto sa mu nechce veľmi rozmýšľať a vôbec.

Úloha: Napíšte program, ktorý načíta tvar miestnosti v tvare mapy skladajúcej sa z medzier a znakov `#` (takýto krížik znázorňuje štvorček podlahy veľkosti 1vn×1vn) a vypíše, či sa dá miestnosť vykachličkovať, alebo nie.

Príklad:

Vstup:
```
 #  #
######
 #  #
```

Výstup:
```
NIE
```

1345. O 276. zákazníkovi

Do softférového družstva SoDr dnes prišiel jubilejný 276. zákazník, ako obvykle, so zaujímavým problémom. Zákazník bol študentom nemenovanej vysokej školy, a ako to tak už u vysokoškolákov býva, nevedel pracovať so zlomkami. Preto potreboval knižnicu, ktorá by mu túto prácu umožnila. Zákazník zaplatil, vzal si so sebou bloček z registračnej pokladne a tu máte úlohu.

Úloha: Navrhnite a napíšte knižnicu, ktorá bude umožňovať prácu so zlomkami. Knižnica by mala zvládnuť základné aritmetické operácie, ako aj procedúry pre vstup a výstup.

Pokiaľ sú oba operandy aj výsledok vo vami určenom rozmedzí, nemal by váš program vyhlásiť pretečenie.

1411. O tybloni

V Nalomenej Trieske rastie čudesná tybloň. Ak ste takýto strom ešte nevideli, nevadí, ani my. Trabantovka povedala, že jeho plodmi sú, pochopiteľne, tyblká. Tyblko vyzerá skoro ako onblko, ibaže je dvojrozmerné, teda má tvar ľubovoľného mnohouholníka, ktorý má v každom vrchole malé modré nič (keď je tyblko zrelé). Keďže každé tyblko je iné, pri jeho rozdeľovaní nastávajú problémy. Treba ho totiž rozdeliť tak, aby pri rezaní vznikli len časti trojuholníkového tvaru, pričom v každom vrchole takéhoto trojuholníka je malé modré nič (v opačnom prípade môže pri konzumácii vzniknúť nepríjemná alergická reakcia).

Úloha: Napíšte program, ktorý pre daný mnohouholník vytvorí jeho ľubovoľnú triangluláciu. Triangulácia je také rozdelenie mnohouholníka na neprekrývajúce sa trojuholníky, že vrcholy každého trojuholníka sú vrcholmi mnohouholníka. Mnohouholník je zadaný počtom vrcholov n a vymenovaním súradníc jeho vrcholov po obvode proti smeru hodinových ručičiek. Výstup programu bude pozostávať zo zoznamu vytvorených trojuholníkov.

Príklad: Pre $n = 5$ a $[0,0], [2,2], [1,2], [2,3], [0,3]$ je výsledok ($[0,0]$, $[2,2]$, $[1,2]$) ($[0,0]$, $[1,2]$, $[0,3]$) ($[1,2]$, $[2,3]$, $[0,3]$).

1412. O dlhočiznych dážďovkách

Ježkovia to mali vždy jednoduché. Keď sa chceli zmestiť do malého priestoru, proste sa schúlili do klbka. Dážďovky to majú omnoho ťažšie. Vedia sa totiž zohýbať len vo svojich „kĺboch“, ktoré majú medzi článkami. Našťastie dážďovky sú dosť šikovné – vedia sa dokonca zohnúť v kĺbe tak, že ich články sa spätne prekryjú...

Úloha: Napíšte program, ktorý pre zadané dĺžky článkov dážďovky zistí, na akú minimálnu dĺžku sa môže dážďovka poskladať. Dážďovka je úsečka, ktorá pri poskladaní môže byť v kĺboch ohnutá o 0°, alebo o 180°. Články dážďovky po sebe nasledujú v takom poradí, v akom sú na vstupe zadané. Dĺžky článkov sú malé celé čísla.

PRÍKLAD: Pre 5 článkov s dĺžkami $1, 3, 2, 3, 2$ je dĺžka poskladanej dážďovky 4.

1413. O naladenej strune

Usilovný Tomáš si nechal za ťažké peniaze naladiť klavír. Keďže ladič býva až hen za lesom, musel si Tomáš klavír dotlačiť späť do brlôžka sám. Aby sa klavír nerozladil, nesmie ním Tomáš otáčať – odstredivá sila pôsobiaca pri takomto pohybe na struny by ich natiahla a klavír by bol znovu rozladený.

Každý z vás už iste aspoň raz v živote tlačil klavír lesom, a preto si viete predstaviť, čo to znamená. Lepšie je si najskôr zistiť, či sa vôbec klavír lesom pretlačiť dá. Les je zo severu aj z juhu ohraničený dvoma riekami, ktoré tečú tesne pri najsevernejšom a pri najjužnejšom strome lesa. Chudák Tomáš teda potrebuje zistiť, či môže klavír presunúť z východu na západ (bez otáčania).

ÚLOHA: Napíšte program, ktorý pre zadaný počet stromov, ich súradnice a rozmery klavíra (obdĺžnika) vypíše, či je možné daný klavír lesom preniesť. Klavír je na začiatku na hony vzdialený od lesa a hrany jeho kolmého priemetu na zem sú rovnobežné so súradnicovými osami. Stromy majú zanedbateľný polomer. Klavír (ani Tomáš) nevie plávať.

Ak sa vám zdá úloha príliš ľahká, vezmite do úvahy koncertné krídlo – také koncertné krídlo má inú polohu strún, a preto ho smelo možno v lese aj otáčať.

PRÍKLAD: Pre 6 stromov so súradnicami $[0, 1]$, $[3.5, 3.5]$, $[0.5, 3]$, $[2, 0]$, $[0, 5]$, $[2, 1.5]$ a klavír s veľkosťou strán $x = 3, y = 2$ sa klavír lesom preniesť dá.

1414. The Streets of San Benátky

Benátčan-s-dlhým-a-nezrozumiteľným-menom má problém. Mestský úrad mu nariadil rozhodnúť, ktoré z lávok v Benátkach sú natoľko významné, že ich absencia by dlhodobo ochromila dopravnú situáciu v meste. Takéto lávky bude treba nahradiť poriadnymi kamennými mostmi. Lávka je významná, ak v prípade, že by spadla, by v meste existovali aspoň dva ostrovy, medzi ktorými by neexistovalo žiadne spojenie. Treba si uvedomiť, že v Benátkach majú iba kanály a lávky nad nimi (a po kanáloch, prirodzene, nemožno chodiť, lebo by mal človek veľmi rýchlo mokré topánky).

ÚLOHA: Pomôžte Benátčanovi-s-dlhým-a-nezrozumiteľným-menom a napíšte program, ktorý načíta počet ostrovov n, počet lávok m a ďalej m dvojíc čísel ostrovov, pričom každá dvojica znamená, že tieto dva ostrovy sú priamo spojené lávkou. Program by mal vypísať všetky lávky, ktoré treba nahradiť poriadnymi mostmi.

PRÍKLAD: Pre 7 ostrovov a 8 lávok: $(1, 2), (2, 3), (4, 7), (1, 4), (3, 6), (2, 4), (5, 6), (3, 5)$ sú významné lávky $(2, 3), (4, 7)$.

1415. O Števovej sieti

Števo sa so svojimi kamarátmi vybral na výlet do lesa. Chodili po lese, chodili, až boli tak unavení, že ďalej chodiť nevládali. I rozhodli sa, že si spravia obedovú prestávku.

Čo môžu robiť študenti Matematicko-fyzikálnej fakulty v lese, keď si spravia obedovú prestávku? To je jednoduché. Každý vyliezol na nejaký strom a vytiahol z batohu chlieb, slaninu, cibuľu a notebook. I omrzelo ich len tak každý sám obedovať, do LCD displeja hľadieť, nuž pospájali počítače náhodným spôsobom do siete. Teraz sedia a nevedia, čo si so sieťou počať, lebo nemajú tušenia, ako poslať jeden druhému po sieti e-mail.

ÚLOHA: Každý počítač je spojený s niekoľkými inými počítačmi, s každým osobitnou linkou. Každý počítač má svoje linky očíslované od 1 po k_i, kde k_i je počet jeho liniek. Každý počítač v sieti má jednoznačne priradené meno (reťazec max. 10 písmen, ktorý neobsahuje medzery – napríklad `FERDO`). Ak sú počítače A a B spojené linkou, môže napríklad počítač A poslať správu počítaču B, správa sa zaradí do mailboxu počítača B.

Predstavme si takýto spôsob posielania e-mailov. Počítač pošle e-mail niektorou svojou linkou. Počítač na druhom konci linky e-mail prevezme, zistí pre koho je, a ak nie je pre neho, pošle ho niektorou svojou linkou ďalej (prerútuje ho). Aby e-maily zbytočne dlho

neblúdili po sieti (trebárs aj na veky-vekov), každý počítač musí vedieť pre každý iný počítač v sieti, ktorou cestou e-mail poslať, aby sa na najmenší počet prerútovaní dostal do cieľa (tieto údaje sú uložené v takzvanej *rútovacej tabuľke*).

Napíšte program, ktorý (napchávajúc sa slaninou) spustí každý Števov kamarát (Števo je egocentrický – kamaráti sa aj sám so sebou) na svojom laptope. Po skončení behu programov by mal každý z počítačov mať k dispozícii rútovaciu tabuľku.

Pre komunikáciu medzi počítačmi máte k dispozícii tieto procedúry (pre jazyk C, prípadne iný jazyk, zadefinujte procedúry s podobnou štruktúrou; *message* je to isté čo **string**, iba s neobmedzenou dĺžkou):

Send(*sprava* : *message*; *linka* : **integer**) – počítač pošle správu *sprava* po linke *linka*.

Receive(**var** *sprava* : *message*; **var** *linka* : **integer**) – vyberie z mailboxu jednu správu a vráti v premennej *sprava* jej obsah a v premennej *linka* údaj o tom, z ktorej linky správa prišla. Ak žiadna správa v mailboxe nie je, *Receive* čaká, kým tam nejaká nebude.

NumberOfLinks(**var** n : **integer**) – do premennej n uloží počet liniek, ktoré vedú z počítača.

MyName(**var** s : **string**[10]) – do premennej **s** uloží meno počítača.

Snažte sa, aby počítače skončili skôr, ako študenti dojedia svoju slaninu (Števo slaninu neje, namiesto nej je kaleráb). Linky sú pomalé, preto sa snažte, aby ste toho neprenášali zbytočne veľa.

1421. O koláči II

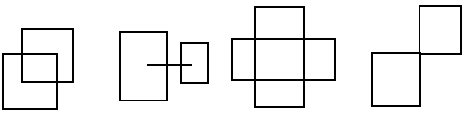
Na obdĺžnikovom plechu upiekli (mimochodom nie veľmi dobrý) koláč veľmi veľkých rozmerov. Koláč sa navyše na krajoch pripálil a tie potom držali na plechu ako prilepené kanagonom[20].

Hostia sa už zišli, nikto nevie, čo im ponúknuť, tak sa rozhodli, že predsa len niečo z toho plechu vydolujú. Zobrali nôž a začali do koláča rezať zvislé a vodorovné úsečky, ktoré sa nedotýkali pripálených okrajov.

Potom plech preklopili, vysypali kúsky rýchlo na tanier a odniesli ich hosťom. Zaujímavou otázkou však zostáva, koľko dier zostalo v pôvodnom koláči.

Úloha: Daných je n začiatočných a koncových bodov jednotlivých úsečiek vyrezaných do koláča. Napíšte program, ktorý zistí, koľko dier zostalo v koláči po vyklopení plechu.

Príklad:



1 diera 2 diery 1 diera 2 diery

1422. O univerzitnom knihovníkovi II

„Oook!“ začudoval sa knihovník, keď zistil, že dnes sú na polici zase iné knihy. Včera polica obsahovala niekoľko prvých a druhých dielov Lindenmayerovho zborníka magických formúl. Aj dnes tam boli prvé a druhé diely toho istého zborníka, ale určite sa líšil ich počet a rozmiestnenie. Keď knihovník niekoľko dní knihy pozoroval, zistil, že ich zmeny nie sú náhodné. Každú noc z každého prvého dielu vznikne skupina kníh u a z každého druhého dielu vznikne skupina kníh v, pričom u a v sú každú noc tie isté. Tak sa rozhodol dať Mrakoplašovi úlohu. Napísal mu na papier nejaké poradie prvých a druhých dielov a prikázal mu zistiť, či niektorý deň bude na polici taká istá skupina prvých a druhých dielov ako je na papieri (naľavo aj napravo od tejto skupiny môžu byť na polici ešte ďalšie knihy).

[20] Vedeli ste, že aj keď tomu lepidlu všetci vždy hovorili kanagon, v skutočnosti sa volá kanago**m**?

Úloha: Pomôžte Mrakoplašovi a napíšte program, ktorý zistí odpoveď na knihovníkovu úlohu. Program načíta postupnosti u, v, w, x pozostávajúce z čísel 1 a 2, kde u je skupina kníh, na ktoré sa každú noc zmení kniha 1, v je skupina kníh, na ktoré sa zmení kniha 2, w je dnešný stav police a x je knihovníkov zoznam. Postupnosti u a v môžu byť aj prázdne.

Príklad: Nech $u = 12$, $v = 212$ a $w = 21$. Každú noc sa každá 1 zmení na 12 a každá 2 sa zmení na 212. Na druhý deň budú na polici knihy 21212 a na tretí deň 2121221212212. V prípade, že $x = 1221$, odpoveď na knihovníkovu otázku bude `ano`, lebo takáto skupina sa nachádza v postupnosti kníh tretieho dňa. V prípade, že $x = 11$, odpoveď bude `nie`, lebo takáto skupina nikdy nemôže vzniknúť.

1423. O vrabcovi Čin-činovi II

Vrabec Čin-čin zase priletel neskoro. Bolo to už piate pomaturitné stretnutie a doteraz sa mu ešte nikdy nepodarilo priletieť včas. Aby nenarobil zmätok a nepokazil veselú zábavu, chcel si už vopred vyhliadnuť miestečko, na ktoré sa posadí.

Vždy sedávali na prvom drôte hneď za dedinou, na úseku od stĺpu EV 274 po stĺp EV 277. Čin-čin nerád sedával na kraji, pretože tam by mal len jedného suseda, s ktorým by sa mohol rozprávať. Ale na druhej strane, Čin-čin má rád okolo seba miesto. Preto by si teraz najradšej sadol medzi tých dvoch susedných vrabcov, ktorí sú od seba najviac vzdialení. Z toľkej diaľky však nedovidí na starých kamarátov. Našťastie blízko sedí straka Rapotajka, ktorá si túto nevšednú udalosť nenechala ujsť a má o všetkom prehľad. A naozaj. Rapotajka Čin-činovi vyrapotala, kedy ktorý vrabec prišiel a kam sa usadil.

Úloha: Napíšte Čin-činovi program, ktorý načíta dĺžky drôtu A, B od stĺpov EV 274 a EV 277 do dediny, ďalej načíta n doteraz usadených vrabcov a ich vzdialenosť po drôte od dediny. Program vypíše dvoch susedných vrabcov, ktorí sú spomedzi všetkých susedných vrabcov od seba najviac vzdialení. Riešenia bežiace v lineárnom čase od počtu vrabcov n sú veľmi vítané.

Príklad: Pre $A = 5$, $B = 16$, $n = 5$ a polohy vrabcov: $9, 13, 7, 15, 10$ sú najvzdialenejší vrabci s polohami 10 a 13.

1424. O kiribatských náleziskách II

Na Kiribatských ostrovoch nedávno objavili pod zemou veľké zásoby rumu II. Po predchádzajúcich skúsenostiach inžinieri postavili jednu veľkú vrtnú vežu, problémom je však preprava rumu II z tejto veže.

Po niekoľkých dúškoch inžinieri navrhli vybudovať veľkú rumárenskú sieť skladajúcu sa z rumovodov a z regulačných staníc. Na každej rúre je umiestnená tzv. čuchačka, cez ktorú môže poverený pracovník údržbovej skupiny očuchávať, či rum náhodou nevyteká mimo rumovod. Každou rúrou môže pretiecť hektoliter rumu za minútu. Regulačné stanice sú miesta, v ktorých sa stretáva jedna alebo viacero rúr, ktoré sú v tomto mieste prepojené. Platí zásada, že prítok do regulačnej stanice a výtok z nej sú v rovnováhe. Vplyv zamestnancov regulačnej stanice na porušenie tejto rovnováhy po vzore priateľov fyzikov zanedbáme. Celá sieť je v regulačnej stanici s priamo napojená na vrtnú vežu a rum je odčerpávaný zo siete v mestskej regulačnej stanici t.

Ráno inžinieri nad svojim návrhom triezvo pouvažovali a zistili, že stavba by to bola síce veľkolepá, ale rumu by sa cez ňu veľa do mesta neprepravilo.

Úloha: Daný je počet regulačných staníc n, počet rumovodov m a ďalej m dvojíc čísel reprezentujúcich jednotlivé rumovody (od ktorej regulačnej stanice ku ktorej vedú), pričom stanica s má číslo 1 a stanica t má číslo n. Napíšte program, ktorý vypočíta, koľko rumu za minútu je takto navrhnutý rumovod schopný prepraviť.

Predpokladajte, že vyťažiť možno ľubovoľne veľa rumu (t.j. viac, ako dokáže pretiecť rumovodmi) a že v t vedia odčerpávať všetok rum, ktorý tam pritečie. Takisto môžete predpokladať, že medzi každou dvojicou regulačných staníc vedie najviac jeden rumovod.

Príklad: Pre $n = 4$, $m = 5$ a rumovody: $(1,2)$, $(2,3)$, $(1,3)$, $(3,4)$, $(1,4)$ cez rumársku sieť pretečú najviac 2 hektolitre rumu za minútu.

1425. O Števovej sieti II

Oprava zadania O Števovej sieti: Naše historicko-výskumné oddelenie zistilo, že Števo nemohol jesť kaleráb, keďže kaleráby vôbec nemá rád. Správna zelenina na tomto mieste bola mrkva.

Števo dojedol svoju mrkvu, poobhliadal sa po okolí, čože to robia jeho kamaráti, a zhrozil sa. V počítačovej sieti panoval úplný chaos! Števo začal liezť po stromoch, rozpájať, prepájať a zapájať nové káble a po niekoľkých minútach pospájal počítače pekne do kruhu. A spokojne na svoje dielo pozeral.

I Števo povedal, ak chcete riešiť akúkoľvek úlohu, musíte zvoliť šéfa! A študenti Matematicko-fyzikálnej fakulty sa zamysleli nad Števovými slovami a videli, že to, čo vraví, je dobré.

Úloha: Každý počítač je spojený s dvoma susednými počítačmi v kruhu, s každým osobitnou linkou. Linky majú čísla 1 (ľavý sused) a 2 (pravý sused). Každý počítač v sieti má jednoznačne priradené meno (reťazec max. 10 písmen, ktorý neobsahuje medzery – napríklad `FERDO`). Ak sú počítače A a B spojené linkou, môže napríklad počítač A poslať správu počítaču B, správa sa zaradí do mailboxu počítača B.

Napíšte program, ktorý (rozmýšľajúc o Števových slovách) spustí každý Števov kamarát na svojom laptope. Po skončení programov bude každý počítač poznať meno šéfa (keďže šéf môže byť len jeden, každý počítač musí zistiť to isté meno, je však úplne jedno, ktorý z počítačov bude šéfom).

Procedúry pre komunikáciu medzi počítačmi, ktoré máte k dispozícii, sú popísané v úlohe 1415.

Snažte sa, aby počítače skončili rýchlo, aby študenti príliš dlho nerozmýšľali nad Števovými slovami, mohlo by to mať nedozerné následky. Linky sú pomalé, preto sa snažte, aby ste toho neprenášali zbytočne veľa.

1431. O magickom kruhu

Stará sága vraví, že niekde na okraji Zemeplochy sa za dávnych časov nachádzal kontinent menom Ulantída. Avšak pred viac než mnohými rokmi ho pohltilo more. Mnoho cestovateľov zahynulo pri hľadaní tohto kontinentu. I čarodej Taratlach sa vydal hľadať Ulantídu. Keďže čoskoro cesta pokračovala po dne oceánu, musel Taratlach (i keď s nevôľou) prikročiť ku kúzleniu – požiadať vodu, aby sa rozostúpila. To predsa nemôže byť žiadny problém – veď to už kedysi dávno zvládol akýsi Mojžiš. Text v jeho príručke hovoril, že treba na pláži vyznačiť kruh, postaviť sa do stredu a odrieknuť príslušnú magickú formulu. Potom neviditeľná ruka vyznačí na pláži 666 priamok a čo najrýchlejšie treba vykríknuť, koľkokrát sa priamky pretli vo vnútri kruhu. Taratlach v matematike nikdy nevynikal, a tak si spočítal, že by mu už len spočítanie tých priamok trvalo dlhšie, ako stanovený časový limit. Teraz len smutne sedí na pláži a hľadí do vĺn.

Úloha: Napíšte program, ktorý načíta čísla n – počet priamok, r – polomer kruhu a n priamok, pričom každá je daná súradnicami svojich dvoch bodov, a spočíta, koľko dvojíc priamok sa pretne vnútri kruhu s polomerom r a stredom v počiatku súradnicovej sústavy.

Príklad: Pre $r = 3$, $n = 4$ a priamky p: $[-2,0]$, $[0,2]$; q: $[-5,1]$, $[5,1]$; r: $[-1,0]$, $[-1,-2]$ a s: $[4,7]$, $[4,6]$ je výsledok 3, lebo sa pretnú tri dvojice priamok (p a q, p a r, q a r).

1432. O koláči III

V Nalomenej Trieske je nepísaným pravidlom, že nevesta, keď sa vydáva, musí upiecť svadobný koláč. Veľkosť koláča je úmerná spoločenskému postaveniu nevesty.

Prabantovka spomína: Jój, to za našich mladých čias, to boli iné svadby ako teraz. Ale najkrajšiu svadbu mala kováčova Anička, keď si brala richtárovie Ondra. Koľko prišlo

vtedy hostí! A každému sa ušlo z Aničkinho koláča. Anička upiekla totiž koláč v tvare veľkého konvexného mnohouholníka – tyblka, a dokonca do každého z vrcholov malé modré nič pridala. Každý kúsok koláča mal trojuholníkový tvar, v každom vrchole kúsok malého modrého nič. Všetko prebiehalo hladko, až kým Ďuro, Aničkin ohrdnutý pytač, nezačal rozhlasovať, že Anička taký veľký koláč upiecť nemohla. Veď plech, na ktorom by sa taký koláč dal upiecť, hádam ani nejestvuje... Urazený kováč doniesol ohromný mnohouholníkový plech, pozbieral všetky kúsky koláča, zavolal učiteľa, notára a farára a spolu sa pustili ukladať kúsky koláča na plech. Po hodnej chvíli, keď zistili, že to nie je až také jednoduché, pozorne zmerali každý kúsok koláča, zakreslili tvar plechu a farár s notárom sa pod to slávnostne podpísali. Potom konečne rozdali koláče a svadobná hostina sa mohla začať...

Úloha: Kováč vypísal odmenu tomu, kto pomôže celej Nalomenej Trieske ukázať, že Aničkin svadobný koláč sa skutočne celý piekol na tom obrovskom plechu. Napíšte program, ktorý načíta tvar plechu (vrcholy konvexného mnohouholníka vymenované proti smeru hodinových ručičiek), počet kúskov koláča a dĺžky strán jednotlivých kúskov (tiež proti smeru hodinových ručičiek) a vypíše, či sa kúsky dajú poukladať na plech tak, aby nič nezvýšilo, aby bol celý plech pokrytý a aby malé modré nič boli vo vrcholoch plechu.

Príklad: Na plech so súradnicami vrcholov $(0.0; 0.0)$, $(3.0; 0.0)$, $(4.5; 2.0)$, $(3.0; 4.0)$ a $(0.0; 4.0)$ sa koláče $(3.0; 4.0; 5.0)$, $(3.0; 4.0; 5.0)$ a $(2.5; 2.5; 4.0)$ dajú poukladať.

1433. O vrabcovi Čin-činovi I

Keď sa Čin-čin vrátil zo stretnutia domov, chvíľu veru rozmýšľal, či trafil správne.[21] Veď áno, trošku aj pili, ale žeby až tak? Rozhodne niečo nebolo v poriadku. Ale čo? Čin-čin sa posadil na priedomie domu, o ktorom si ešte stále myslel, že je jeho vlastný, a pustil sa do tuhého premýšľania. A po čase na to naozaj prišiel. Tie zbalené kufre boli tým, čo sa mu nepozdávalo. Aby sa priznal, boli mu nanajvýš podozrivé, len si žiaľ práve teraz nevedel spomenúť, na čo je taký kufor dobrý. Našťastie o chvíľu k nemu pricupital malý vrabček a skutočne netrvalo dlho, kým v ňom Čin-čin spoznal svojho malého Pim-pima, synáčika, na ktorého bol právom hrdý. A vtedy sa spamätal. Veď dnes je presne ten deň, keď sa mali sťahovať! Ale v tých dvoch kufroch asi nebude zbalené celé dvojposchodové hniezdo aj s nábytkom, uvažoval Čin-čin ďalej. A potom je tu pelikán a prázdne škatule... Čin-čin začínal tušiť, čo ho čaká. Zvyšné veci bude asi treba zbaliť do tých škatúľ. Ale škatúľ nemôžem použiť veľa, veď sú dosť drahé – začínal Čin-čin prejavovať známky triezveho uvažovania.

Úloha: Dané sú čísla s – objem jednej škatule (všetky škatule majú rovnaký objem), n – počet zvyšných vecí, ktoré treba pobaliť, a potom ešte n čísel – objemov jednotlivých vecí. Všetky tieto údaje sú celé čísla. Čin-čin vie, že keby chcel zistiť, ako sa dajú jeho veci pobaliť do najmenšieho možného počtu škatúľ, odpovede by sa nemuseli dožiť ani Pim-pimove vnúčatá. Preto mu stačí také usporiadanie, pri ktorom sa nepoužije viac ako dvojnásobok najmenšieho možného počtu škatúľ, ale výsledok potrebuje rýchlo, lebo pelikán čaká. Napíšte Čin-činovi program, ktorý nájde takéto usporiadanie. Výstupom je počet potrebných škatúľ a zoznam obsahujúci pre každý predmet jeho objem a poradové číslo škatule, do ktorej má byť pobalený. Na poradí predmetov vo výstupe nezáleží. Takisto škatule si môžete očíslovať v ľubovoľnom poradí. Ak súčet objemov niekoľkých predmetov neprevyšuje objem škatule, tieto predmety sa dajú do jednej škatule zabaliť.

Riešenie musí obsahovať okrem popisu aj dôkaz správnosti, t.j. dôkaz, že vaše riešenie skutočne neprekročí dvojnásobok optimálneho počtu škatúľ.

Tým, ktorým sa zdá byť táto úloha ľahká, odporúčame vyriešiť pozmenené zadanie: Nájsť čo najefektívnejší program, ktorý nájde počet škatúľ a rozmiestnenie predmetov tak,

[21] A takisto rozmýšľal nad tým, ako sa mohlo stať, že úlohu „O vrabcovi Čin-činovi I“ čítate až o sériu neskôr ako úlohu „O vrabcovi Čin-činovi II“. Ale iba čo ho z toho rozbolela hlava.

aby ich nebolo viac ako tri polovice optimálneho počtu (pri nepárnych číslach zaokrúhľujeme podiel nahor). Samozrejme, aj s dôkazom.

1434. O burundijských mapách

Kmene v Burundi sa konečne rozhodli žiť v pokoji a mieri. Kráľ Burundi sa tomuto ich rozhodnutiu najskôr veľmi začudoval, zdalo sa mu, akoby ani svoju vlastnú zem nespoznával. Ale on, učená hlava, našťastie vedel, čo robiť. Takmer všetci Burundijčania boli bojovníci, a keď z nich opadne vlna nadšenia z nastoleného mieru, všimnú si, že stratili prácu. Treba ich teda niečím zamestnať, aby si túto holú skutočnosť všimli čo najneskôr. „Ale čím?“ hútal kráľ. A vtedy na to prišiel. Treba nakresliť mapu Burundi. Potom si ju bude chcieť každý obkresliť a doma, ako hrdý Burundijčan, vyvesiť na stenu. To zaberie dosť času, lebo v Burundi majú len päť farbičiek, ktoré u nich zabudol nejaký stroskotanec. Lenže vychýrený a konkurzom vybraný kreslič máp, jediný, ktorý sa v Burundi našiel, vie, že mapa má každé územné teritórium zafarbené takou farbou, akou nie je zafarbené žiadne susedné teritórium. A v Burundi je len tých päť povestných farbičiek...

Úloha: Napíšte kresličovi máp program, ktorý nájde zafarbenie mapy Burundi najviac piatimi rôznymi farbami. Vstupné hodnoty pre váš program sú: n – počet kmeňov v Burundi a pre každý kmeň zoznam susedných kmeňov a zoznam kmeňov, ktoré susedia so zvyškom sveta. Dva kmene (prípadne kmeň a zvyšok sveta) sú susedné, ak ich teritóriá majú spoločnú hranicu. Teritórium každého kmeňa je súvislý útvar. Výstupom je zafarbenie jednotlivých územných teritórií v Burundi (aj zvyšok sveta musí mať nejakú farbu, tá sa však, samozrejme, môže vyskytovať aj na nejakých s ním nesusedných teritóriách).

Príklad:

Vstup:	Výstup:
počet kmeňov $n = 6$	Zafarbenie teritórií:
susedia kmeňa 1: 2, 3, 4 a zvyšok sveta	kmeň 1: zelená
susedia kmeňa 2: 1, 4, 6 a zvyšok sveta	kmeň 2: červená
susedia kmeňa 3: 1, 4, 5, 6 a zvyšok sveta	kmeň 3: červená
susedia kmeňa 4: 1, 2, 3 a 6	kmeň 4: žltá
susedia kmeňa 5: 3, 6 a zvyšok sveta	kmeň 5: zelená
susedia kmeňa 6: 2, 3, 4, 5 a zvyšok sveta	kmeň 6: biela
susedia zvyšku sveta: 1, 2, 3, 5 a 6	Zvyšok sveta: žltá

1435. O Števovej sieti III

Kamaráti si úspešne zvolili šéfa a s elánom sa pustili do písania prefíkanej sieťovej aplikácie (PSA). Ale Števa nebavilo dlho na strome sedieť a do laptopu hľadieť a začal presviedčať svojich kamarátov, nech sa idú ešte pozrieť, ako svet vyzerá z najbližšieho kopca. Ale kamaráti boli príliš zaujatí tvorbou PSA a vôbec sa im nechcelo plahočiť sa na nejaký kopec. A tak sa Števo rozhodol, že pôjde sám. Zbalil laptop aj obidve šnúry, ktoré ho spájali so susednými počítačmi a svižným krokom zmizol z dohľadu.

Kamaráti medzičasom dopísali PSA a prvýkrát ju spustili. V programe však bola chyba a tá mala katastrofálne následky – každému počítaču sa vymazali nielen všetky dôležité informácie o sieti a o šéfovi, ale dokonca aj jeho vlastné meno. Úbohým kamarátom teraz nefunguje žiaden z ich algoritmov a sú z toho úplne zúfalí.

Úloha: Všetky počítače sú pospájané do jedného radu, každý počítač okrem dvoch krajných je spojený so svojimi dvoma susedmi priamymi linkami, krajné počítače sú spojené prirodzene len s jedným susedom. Každý počítač má svoje linky očíslované číslami 1 a 2 (prípadne len 1). Ak sú počítače A a B spojené linkou, môže napríklad počítač A poslať správu počítaču B, správa sa zaradí do mailboxu počítača B.

Počítače však nemajú žiadne jednoznačné označenie. Úlohou je očíslovať ich číslami od 1 po p, kde p je počet počítačov v sieti. Číslo 1 môže mať ľubovoľný z krajných počítačov,

jeho sused má číslo 2, a tak ďalej a číslo p má druhý krajný počítač. Napíšte program, ktorý spustia všetci Števovi kamaráti na svojich laptopoch, pričom po jeho skončení každý z počítačov pozná svoje číslo.

Pre komunikáciu medzi počítačmi máte k dispozícii rovnaké procedúry ako v úlohe 1415, okrem procedúry *MyName*.

Snažte sa, aby počítače skončili skôr ako študenti podľahnú beznádeji a rozhodnú sa zbaliť celú počítačovú sieť a bežať za Števom (toho by aj tak nedobehli, a ešte by mohli zablúdiť). Linky sú pomalé, a preto sa snažte, aby ste toho neprenášali zbytočne veľa. Číslo p na začiatku behu programu nie je známe. Môžete však predpokladať, že je nepárne. Nepoužívajte generátor náhodných čísel.

1441. O prefíkanom špiónovi

Agent 00, Ferdo, pracuje už niekoľko desaťročí pre SIS (Slovenskú informatickú spoločnosť) na istej nemenovanej americkej univerzite. Cieľom jeho misie je získať čo najviac technických plánov najnovších algoritmov. Za tie roky sa vypracoval na vedúcu pozíciu inštitútu, takže všetky práce študentov prechádzajú najskôr práve jeho rukami. On sa ich, pochopiteľne, snaží čo najrýchlejšie poslať kolegom do SIS, na čo používa špeciálnu vysokofrekvenčnú vysielačku.

Americká kontrarozviedka mu už je na stope, hlavne vďaka tzv. CIA (Capture Incoming Alphawave) spôsobu zachytávania prichádzajúcej správy. Zariadenia CIA majú rozmiestnené po celom kontinente, každé vie určiť približný smer k vysielaču prichádzajúcej správy. Agentovi A000H sa podarilo získať zoznam všetkých týchto zariadení spolu s výsledkami posledných meraní. Zoznam začína číslom n (počet záznamov na zozname), pre každé $i \in \{1, \ldots, n\}$ obsahuje pozíciu (x, y) i-teho prístroja CIA, ako aj dvojicu smerov, ktoré určujú nanajvýš 90° uhol, v ktorom pravdepodobne Ferdov vysielač leží. (Smery sú v stupňoch, 0° je smerom na východ a uhol stúpa proti smeru hodinových ručičiek.)

Úloha: Napíšte program, ktorý zistí, či Američania vedia z daných hodnôt usúdiť približnú Ferdovu polohu, t.j., či existuje také miesto, pre ktoré by boli všetky informácie z prístrojov CIA správne. Táto informácia je veľmi dôležitá, aby mohol byť Ferdo včas varovaný, preto sa sústreďte na rýchlosť vášho programu.

Američania ešte nezistili, že Zem je guľatá, preto používajú rovinné mapy kontinentu.

Príklad:

Vstup:

$n = 3$

x	y	prvý smer	druhý smer
1	2	315	45
2	−0.4	90	135
2	3	180	270

Výstup:

```
Nemajú šancu zistiť jeho polohu.
```

1442. O Pakovači

V tomto čase na súostroví Kiribati vrcholia veľkolepé oslavy pri príležitosti zavedenia telefónnej siete. Niektorí triezvi domorodci však upozorňujú na fakt, že ostrovy nemajú dostatok dreva na vytlačenie telefónnych zoznamov. Bohatí Kiribatčania sa totiž vyznačujú dlhými menami. Vzhľadom na skutočnosť, že sú na svoje mená nesmierne hrdí a kvôli telefónu by si ich nenechali zmeniť, rozhodla sa telefónna spoločnosť mená v telefónnom zozname „pakovať“, a to nasledovne: každá n-krát sa opakujúca postupnosť znakov PZ sa môže zapísať aj v tvare `n[PZ]`. Napríklad `abababab` môžeme zapísať aj ako `3[ab]`. PZ môže obsahovať už spakované postupnosti, takže výsledný spakovaný zápis môže obsahovať ľubovoľný počet vnorených zátvoriek.

Úloha: Napíšte program, ktorý na vstupe dostane postupnosť znakov `a`...`z` a ktorý pre túto postupnosť nájde najkratší možný zápis (ak takýchto zápisov existuje viac, stačí

jeden z nich). Najkratší zápis je zápis skladajúci sa z najmenšieho počtu znakov, teda vrátane znakov `[`, `]`, a číslic.

PRÍKLAD:

VSTUP:	VÝSTUP:
`baaaaaaaaaaaaababababc`	`b12[a]4[ab]c` (dĺžka je 12 znakov)
`aaaaaaaabcaaaaaaabc`	`a2[7[a]bc]` (dĺžka je 10 znakov)

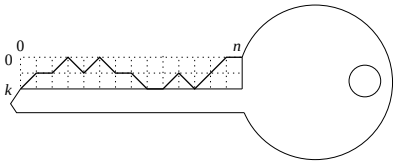
1443. O kľúčikoch

V softférovom družstve SoDr majú ďalšieho zákazníka. Tentokrát sa na nich obrátil nemenovaný výrobca kľúčikov. Kľúčiky sa vyrábajú z dopredu pripravených výliskov (tzv. profilov) vybrusovaním. Pre každý typ výlisku je známe číslo n – šírka „aktívnej“ časti kľúčika, t.j. počet „zúbkov“, a číslo k – počet rôznych hĺbok výbrusu.

Kľúč možno reprezentovať ako postupnosť celých čísel $a_0, \ldots, a_n$, pričom a_i udáva hĺbku výbrusu vo vzdialenosti i od začiatku „aktívnej“ časti kľúčika (teda akú hĺbku má i-ty „zúbok“). Medzi jednotlivými celočíselnými pozíciami je výbrus vedený po úsečke. Pre túto postupnosť ďalej platí:

- $a_0 = k$ a $a_n = 0$
- hĺbka výbrusu je maximálne k, t.j. $0 \leq a_i \leq k$ pre všetky i, $0 \leq i \leq n$
- hĺbka susedných zúbkov sa líši najviac o 1, t.j. $|a_i - a_{i-1}| \leq 1$, pre $0 < i \leq n$

Príklad kľúčika:



Výrobca práve začal výrobu kľúčov z nového typu výlisku a rád by vedel, koľkých zákazníkov môže uspokojiť predajom zámkov a kľúčikov vyrobených z tohto typu výlisku. (Prirodzene, každý zákazník chce mať iný kľúč.)

ÚLOHA: Napíšte program, ktorý načíta typ výlisku (t.j. čísla n a k) a určí, koľko rôznych kľúčikov možno vybrúsiť z takéhoto výlisku.

PRÍKLAD: Pre $n = 4$ a $k = 2$ sa dajú vyrobiť 2 rôzne kľúčiky.

1444. O veľkom suchu

Ľudia sú už raz takí. Ak im niekto rozpráva, čo zlé ich postihlo, povedia: „To nič nie je, keby si vedel, čo sa stalo mne...“ Santo a Banto nie sú výnimky. Teraz sú obaja veľmi smutní. Prednedávnom sa rozhodli, že nebudú len v krčme vysedávať a posadili si vo svojich záhradkách zeleninu. Ale teraz už dva mesiace nepršalo a nakoniec im všetko vyschlo. Zem úplne stvrdla a popraskala. Najprv sa začali vytvárať len štrbiny, tie sa však postupne predlžovali, popretínali, no a teraz sú ich záhradky už samá kryha. A tak Santo s Bantom sedia znovu v krčme a hádajú sa, čia záhrada je viac popraskaná. Keďže žiaden nedokázal toho druhého presvedčiť o svojej pravde, vybrali sa do svojich záhrad a začali kryhy počítať. Tak teraz smutne chodia a počítajú a počítajú. Pomôžte im ich spočítať, lebo na oblohe sa začali objavovať prvé obláčiky a ak začne pršať, svoj spor nikdy nevyriešia.

ÚLOHA: Na vstupe je zadané n (počet miest, v ktorých sa štrbiny popretínali) a dvojice priesečníkov, medzi ktorými vedie štrbina. Viete, že štrbiny sa v ďalších miestach už nepretínajú. Spočítajte, na koľko krýh štrbiny rozdelili záhradu. Za kryhu sa, samozrejme, počíta aj plocha od okrajových štrbín po plot (čím viac, tým viac).

Príklad:

3 kryhy

2 kryhy

1445. O Števovej sieti IV

Konečne sa kamarátom podarilo očíslovať sa a sú na to právom hrdí. Ale skutočnosť, že majú všetky počítače zapojené v jednom rade, ich netešila. Veď koho by aj, keď si stále ten na jednom konci s tým na druhom konci posielajú správy a všetkých ostatných to spomaľuje... Nenašli oni iné riešenie tohto problému ako sa porozpájať a náhodne potom pozapájať. I rozpájali oni a spájali sa a zrazu si tak uvedomili, že im chýba Števo. Ako mysleli, tak mysleli, no veruže nikto z nich si nemohol spomenúť, kam ten Števo zmizol. Hútali, hútali a vyhútali, že on určite spadol zo stromu a v tej kope machu ho nie je vidno. Alebo tam dolu nie je mach? Nó, na tom predsa nezáleží. Rozhodne ho nie je vidno, jeho laptop tiež nie a to začína byť povážlivé. Čo ak osud úbohého Števa postretne aj niekoho z jeho kamarátov? Potom by ich bolo zase o jedného menej, aj keď to nie je to najdôležitejšie. Oveľa horšie by bolo, keby sa im potom sieť rozpadla na viacero častí. A tu ich napadlo: Veď my musíme zistiť, či je medzi nami niekto taký dôležitý, že mu nemôžeme dovoliť spadnúť!

Úloha: Počítače sú náhodne pospájané. Každý počítač má svoje meno (číslo, rôzne od ostatných počítačov) a linky na svojich susedov označené ich menami. Napríklad ak počítač s číslom 7 je spojený s počítačom č. 3, potom ich spoločnú linku má počítač č. 7 označenú číslom 3 a počítač č. 3 ju má označenú číslom 7. Dva počítače sú spojené nanajvýš jednou linkou. Môžete predpokladať, že jeden z počítačov má číslo 1.

Vašou úlohou je napísať program, ktorý keď spustia Števovi kamaráti na svojich laptopoch, tak po jeho skončení bude každý z nich vedieť, či sú pospájaní tak, že po odpojení hociktorého počítača zostane sieť súvislá.

Pre komunikáciu medzi počítačmi máte k dispozícii rovnaké procedúry ako v úlohe 1415, okrem procedúry *MyName*, ktorá vracia číslo a nie reťazec. Okrem týchto procedúr môžete používať procedúru *kthLink*(*k* : **integer**; **var** *linka* : **integer**), ktorá do premennej *linka* uloží číslo *k*-tej linky vychádzajúcej z počítača (čo je zároveň číslo jedného zo susedných počítačov).

z1411. Z univerzitnej knižnice

„Oook!“ povedal knihovník, keď zistil, že sa stala chyba. Mrakoplaš zase poplietol abecedu. V univerzitnej knižnici sa knihy triedia podľa začiatočného písmena, a tie, ktoré sa začínajú na rovnaké písmená, sa triedia podľa počtu strán. Mrakoplaš si, ako obyčajne, vymenil dve písmená z abecedy a teraz treba úseky kníh začínajúcich na tieto dve písmená vymeniť späť tak, aby knihy v úsekoch zostali správne usporiadané. Je to však strašný problém, lebo v knižnici už nie je ani jedna voľná polica.

Úloha: Dané sú čísla n, i, j, k, l, kde $1 \le i \le j < k \le l \le n$ a pole $A[1\,..\,n]$ celých čísel. Prvky si rozdelíme do piatich úsekov: $a_1 \dots a_{i-1}$, $a_i \dots a_j$, $a_{j+1} \dots a_{k-1}$, $a_k \dots a_l$, $a_{l+1} \dots a_n$ (niektoré úseky môžu mať aj nulovú dĺžku). Preusporiadajte prvky tak, že druhý a štvrtý úsek budú navzájom vymenené, t.j. pole bude vyzerať takto: $a_1 \dots a_{i-1}$, $a_k \dots a_l$, $a_{j+1} \dots a_{k-1}$, $a_i \dots a_j$, $a_{l+1} \dots a_n$. Okrem poľa A môžete použiť iba premenné typu **integer** (t.j. nepoužívajte iné polia, súbory a podobne).

Príklad: Pre $n = 7$, $i = 2$, $j = 3$, $k = 6$, $l = 6$, $A = (1, 2, 3, 4, 5, 6, 7)$ je výsledkom $A = (1, 6, 4, 5, 2, 3, 7)$.

z1412. Zorov závet

„Aargh...“ vysvetlil Zoro a zomrel. Zanedlho prišli dediči a začali sa domáhať vyplatenia životnej poistky. V poisťovni im oznámili, že podľa zmluvy, keďže sa nejedná o vysokú čiastku, peniaze pozostalým vyplatia do sto dní. Dediči sú veľmi netrpezliví a chceli by vedieť, koľko dní už od Zorovej smrti uplynulo.

Úloha: Napíšte program, ktorý načíta dátum Zorovej smrti, dnešný dátum a vypíše, koľko dní uplynulo od jeho smrti. Úradníci v poisťovni sú však veľmi leniví a neporiadni, preto sa môže stať, že niektoré prípady budú vybavovať aj niekoľko rokov. Rozdiel v zadaných dátumoch môže byť aj väčší ako sto dní. Môžete však predpokladať, že obidva dátumy sú v rokoch 1901–1999.

z1413. Záletný zemepán

Zemepán Zoltán je veľký záletník. V okolí jeho zámku žije n krásnych zemepaní. Každá má zámok s množstvom vežičiek a to sa zemepánovi Zoltánovi veľmi páči. V poslednom čase však na niektorých cestách striehnu žiarliví zbojníci a Zoltán sa bojí tadiaľ prechádzať. Trápia ho pochybnosti, či sa môže po bezpečných cestách dostať ku každej z krásnych zemepaní.

Úloha: Dané sú čísla n a k, kde n je počet krásnych zemepaní a k je počet bezpečných ciest. Zámky sú očíslované číslami $1, \ldots, n+1$, zámok číslo $n+1$ je zemepánovo sídlo. Ďalej je daný zoznam bezpečných ciest. Každá cesta vedie medzi dvoma zámkami a je zadaná ich číslami. Napíšte program, ktorý zistí, či sa zemepán môže dostať ku každej zemepanej, pričom môže cestovať aj cez viaceré zámky.

Príklad: Pre $n = 3$, $k = 2$ a cesty 4 3 a 1 3 sa Zoltán nemôže dostať k zemepani číslo 2, preto program vypíše `NIE`.

z1414. Zátvorkove prvočísla

Zátvorka je učiteľom matematiky. Celkom dobrý učiteľ, ale čo je veľa, to je veľa. Na poslednej hodine matematiky ho žiaci stráášne nahnevali, dokonca na Ferovi, zvanom Zúrivec, bol nútený zlomiť obľúbené ukazovadlo. Nuž im dal ťažkú, preťažkú domácu úlohu. Keď sa vrátil po náročnom dni domov, schladil sa jeho hnev a objavili sa pochybnosti. Čo ak ho žiaci tromfnú a niekto nájde lepšie riešenie ako on, učiteľ Zátvorka?

Úloha: Pomôžte žiakom dokázať Zátvorkovi, že sú pod slnkom aj lepší matematici ako on. Nájdite (za výdatnej pomoci vášho programu) čo najdlhšiu aritmetickú postupnosť prvočísel. *Aritmetická postupnosť* je taká postupnosť čísel, že rozdiel každých dvoch po sebe idúcich čísel je rovnaký. *Dĺžkou postupnosti* rozumieme počet jej členov. Hodnotiť budeme hlavne vami nájdenú postupnosť, ale aj postup, akým ste túto postupnosť našli. Preto priložte aj program a slovne popíšte váš postup hľadania.

Príkladom takejto postupnosti dĺžky tri je $3, 7, 11$.

z1415. Zelená žabka

Zrazu sa okolo žabky Júlie zotmelo. Skutočne. Zlý Móricko na ňu hodil svoj klobúk. Na tom by zatiaľ nebolo nič divné, ale to ešte neviete, že Mórickov klobúk má pôdorys tvaru obdĺžnika s rozmermi $m \times n$. Júlia začala skákať a volať o pomoc: „Kváááák“. Skákala tak zbesilo, že si pri tom ani len nevšimla, ako naráža do klobúka a pod rovnakým uhlom odrazu ako bol uhol dopadu sa od jeho stien odráža. Popri tom s úžasom zistila, že klobúk leží na jahodovisku. Vždy keď doskočila na jahôdku, tak ju zjedla, takže to napokon nebolo až také zlé.

Úloha: Dané sú p, n, m a pole $A[1\,..\,N, 1\,..\,M]$. Na každom políčku poľa A bude udaný počet jahôdok, ktoré sa tam nachádzajú. Ďalej sú dané x a y, ktoré určujú pozíciu žabky, a dx, dy určujúce smer jej prvého skoku (ak Júlia nenarazí do steny, tak po prvom skoku budú súradnice jej dopadu $[x + dx, y + dy]$). Napíšte program, ktorý odsimuluje p skokov žabky. Ak na pozícii dopadu rastú ešte nejaké jahôdky, Júlia z nich jednu zje. Zobrazujte celú

situáciu na obrazovke pomocou znakov po každom Júliinom skoku. Po skončení simulácie vypíšte počet jahôdok, ktoré Júlia zjedla.

Príklad: Keď $dx = -2$, $dy = -4$, potom žabka z pozície $[6, 4]$ skočí na pozíciu $[4, 1]$ (po odraze od steny) a po ďalšom skoku bude na políčku $[2, 5]$.

z1421. Záviš a jeho rozhlasový sen

Záviš je obchodníkom od narodenia. Raz ho napadlo, ako dobre by sa dalo zbohatnúť, keby si založil vlastné rádio. Mal by kopec peniažkov za reklamu, jedným slovom rozprávka...

I začal zisťovať, kto by mal záujem v Záviš rádiu odvysielať reklamu. Pre každý z prvých d dní od dňa Z (predpokladaného začiatku vysielania) si vypočítal, koľko bubáčikov by zarobil na reklame. Nadšený si ľahol spať. Keď sa ráno zobudil, napadla ho hrozná myšlienka: veď on nemá žiadny vysielač! Mohol by si nejaký prenajať, ale to stojí dosť bubáčikov. Navyše, vysielač sa dá prenajať iba na jedno súvislé obdobie. Chudák Záviš schytil ceruzku a papier a začal počítať, odkedy dokedy je najvýhodnejšie prenajať si vysielač, aby čo najviac zarobil...

Úloha: Napíšte program, ktorý načíta c – cenu za jeden deň prenajatia vysielača, počet dní d a pre každý deň hodnotu b_i – koľko v ten deň môže Záviš zarobiť na reklame, a vypočíta, odkedy dokedy je pre Záviša najvýhodnejšie prenajať si vysielač.

Príklad: Pre 6 dní pri cene za jeden deň vysielania 20 a zárobky v jednotlivé dni: 18, 35, 6, 80, 15 a 21 je najvýhodnejšie prenajať si vysielač od 2. do 4. dňa. Spolu zarobí $35 + 6 + 80 - 3 \times 20 = 61$ bubáčikov.

z1422. Ziskuchtiví zbojníci a Kiribati

Súostrovie Kiribati má problémy. Od istého času ich pravidelne (každý prvý pondelok v mesiaci) prepadávajú Ziskuchtiví zbojníci. Nakoniec Kiribatčania povedali „dosť“ a rozhodli sa nakúpiť v miestnom obchode s domácimi zvieratami levy a umiestniť ich na pobreží. Najbližšie keď si Ziskuchtiví zbojníci prídu zobrať, čo im nepatrí, prídu mnohí o ruky, nohy alebo o hlavu (tí (ne)šťastnejší). Už treba len vyrátať dĺžku pobrežia (aby zistili, koľko levov treba) a Ziskuchtiví zbojníci, traste sa!

V kráľovskej knižnici si Kiribatčania našli mapu súostrovia, no nech robia, čo robia, dĺžku pobrežia nevedia zistiť. Zistili iba nasledujúce skutočnosti: Mapa je rozdelená na $m \times n$ štvorčekov, každý štvorček predstavuje pevninu (je označený `1`) alebo more (je označený `0`), teda žiaden štvorček neobsahuje pevninu aj more zároveň. Štvorčeky dotýkajúce sa navzájom hranou (nie rohom) patria do toho istého ostrova. Žiadny ostrov sa nedotýka okraja mapy (inak by sa nevedelo, či mimo mapy nepokračuje), teda na okraji mapy je more. Každý ostrov na mape patrí do súostrovia Kiribati a každý ostrov zo súostrovia Kiribati je na mape. Pobrežie tvoria hrany medzi štvorčekom pevniny a štvorčekom mora. V súostroví Kiribati sa môžu nachádzať (už bez ostrovov) aj lagúny, vnútorné jazerá a iné mláky, ktorých brehy sa do pobrežia nezarátavajú.

Úloha: Napíšte program, ktorý vypočíta dĺžku kiribatského pobrežia (t.j. súčet dĺžok pobreží všetkých ostrovov na Kiribati), pričom sú dané rozmery mapy m a n a mapa súostrovia.

Príklad:

Vstup:
$m = 5, n = 6$
Mapa súostrovia:
```
000000
011100
010110
011110
000000
```

Výstup:
Dĺžka pobrežia je 14.

z1423. Zlato nad zlato

V softférovom družstve SoDr majú novú úlohu. Tentoraz ide o program pre jedno zlatníctvo. Nanešťastie každý z n pracovníkov vytvoril vlastný program a medzi sebou sa nevedia dohodnúť, ktorý použiť. Programy (očíslované od 1 po n podľa osobného kódu pracovníka) sa líšia výsledným číslom, udávajúcim počet karátov testovaného zlatého predmetu.

Po búrlivej polemike zavrhli programy s najvyšším odhadom (aby zákazník nemusel veľa platiť), aj s najnižším odhadom (aby zákazník nemohol byť odsúdený za krádež). Nakoniec sa rozhodli pre zlatú strednú cestu, teda pre taký program, že keď rozdelíme zvyšné programy na dve skupiny podľa toho, či dávajú menší alebo väčší výsledok, tak ich bude buď rovnako, alebo tých s menším výsledkom bude o jeden menej.

Celé družstvo začalo búrlivo oslavovať, ako múdro to vyriešilo, až kým nezačali vyberať príslušný program. Naraz všetkým zamrzol úsmev na tvári.

Úloha: Napíšte program, ktorý dostane na vstupe n rôznych celých čísel (výsledkov programov jednotlivých pracovníkov) a vypíše výsledok, aký dá program vybratý podľa kritérií uvedených vyššie.

Príklad: Pre $n = 7$ a výsledky programov $10, 4, 3, 15, 2, 6, 1$ dá vybraný program výsledok 4.

z1424. Zátvorkove postupnosti

Zátvorka kdesi čítal o takejto postupnosti: Vezmime si ľubovoľné celé číslo $n > 1$. Ak je párne, vydeľme ho dvomi, ak je nepárne, vynásobme ho tromi a pripočítajme jednotku. Toto môžeme opakovať dovtedy, kým nie je n jedna.

„Hm," povedal si Zátvorka, „to by bolo niečo pre mojich neposlušných žiakov! Ak budú zase tak vyvádzať ako minule, dám im nejaké číslo a nech si počítajú. Veď sa oni naučia poslúchať!" Má to však jednu nevýhodu: ak to číslo bude veľmi veľké, bude sa ťažko pamätať. Nuž, Zátvorka teraz, chudák, sedí a rozmýšľa, aké číslo by sa najviac hodilo.

Úloha: Pomôžte Zátvorkovi. Nájdite (za výdatnej pomoci vášho makačského programu) také číslo n, že $p(n)/c(n)$ bude čo najväčšie, kde $p(n)$ je počet členov Zátvorkovej postupnosti a $c(n)$ je počet cifier čísla n (v desiatkovej sústave).

Príklad: Pre číslo 7 vyzerá Zátvorkova postupnosť takto:

$$7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1$$

Teda, $p(n) = 17$; $c(n) = 1$.

z1425. Zrazené vláčiky

V Zmrzlinove to majú ťažké. Nielenže je tam veľká zima ľuďom, ale dokonca aj vláčikom. A tak sa taký vláčik radšej stále pohybuje a nikomu nezastavuje, aby neprechladol alebo nezmrzol. Keď je vláčik malý, mama mu vždy hovorí, kade má chodiť, aby sa nestratil. A keď vláčik vyrastie a mama už naňho nemá čas, tak si už svoju trasu pamätá a chodí po nej stále dookola aj bez maminej pomoci. Bojí sa však toho, že doňho nejaký iný vláčik narazí. Vtedy je zle a treba ísť do opravy, čo je pre vláčiky asi také nepríjemné ako je pre ľudí chodenie k zubárovi. Alebo možno ešte nepríjemnejšie.

Úloha: Zmrzlinovo je obdĺžniková krajina o rozmeroch 80×25. Napíšte program, ktorý najskôr načíta počet vláčikov n a postupne pre každý vláčik jeho dĺžku d, farbu f a jeho okružnú trasu. Okružná trasa je daná postupne súradnicami políčok v Zmrzlinove, ktoré musia susediť hranou (zároveň susedia hranou prvé a posledné zadané políčko). Potom váš program na prvé až d-te políčko trasy umiestni vláčik farby f, teda ho vykreslí na obrazovku. Po vykreslení všetkých vláčikov sa začnú naraz rovnakou rýchlosťou pohybovať po zadaných trasách stále dookola dovtedy, až kým sa nejaké vláčiky zrazia.

z1431. Závišova záhradka

Záviš zasa začal zarábať. Založil znova zaujímavú záhradku. Zárobok zo záhradky – žiadna zábava. Žeby Záhradkárske zápolenie zaujímavých záhradiek (známe značkou ZZZZ)? Znamená ZZZZ, že Záviš získa zlato, zlato, zlato, zlato? Žiadajú zvyčajné zábradlie záhradky (zoštvorcované). Zvíťaziť znamená, že žiadna známa záhradka zo Zeme zasiala záhony zrovna zvíťazeným zobrazením. Zvíťazí ziskuchtivý Záviš? Zohnal zoznam známych záhradiek. Žeby záhradky zo zoznamu zobrazovali Závišovu záhradku? Začal zrovnávať...

Úloha: Napíšte program, ktorý načíta n (veľkosť záhradky) a dve matice typu $n \times n$ pozostávajúce z núl a jednotiek (`0` značí prázdne miesto, `1` záhon). Na to vypíše, či sa dané záhradky podobajú. Dve záhradky sa podobajú, ak možno dostať jednu z druhej pomocou niekoľkých operácií otáčania o 90° a preklápania podľa osi x alebo y (osová súmernosť).

Príklad:

Vstup:

$n = 4$

```
1 0 0 0      0 0 1 0
0 0 1 1      1 0 1 0
0 1 0 0      0 1 0 0
0 0 1 0      0 0 0 1
```

Výstup:

Áno, podobajú sa (druhú možno dostať z prvej otočením o 90° doprava a preklopením podľa osi x).

z1432. Zahraničné spravodajstvo

Z titulkov dnešných správ vyberáme: Rozpor medzi Kiribati a Burundi, Celosvetový štrajk pokračuje, Na severozápade sa premnožili gorily.

A teraz k jednotlivým správam podrobnejšie: Medzi Kiribati a Burundi sa strhla roztržka po tom, ako kiribatská princezná odmietla burundijského náčelníka. Podľa oficiálnych zdrojov princezná nemala k tomuto rozhodnutiu žiaden dôvod a náčelník je vraj šumný mládenec. Napokon sa obe strany dohodli na tom, že si princezná s náčelníkom zahrajú partičku tamojšej národnej hry a kto vyhrá, ten vyhrá. Médiá sa po dlhšom mlčaní rozhodli zverejniť aj dlho utajované pravidlá: Hrajú dvaja hráči – princezná a náčelník. Hráči roztrhnú nejaké dva princeznine náhrdelníky a guľôčky z nich položia na hrací plán s erbmi oboch krajín tak, aby sa nepomiešali. Potom striedavo z plánu odoberajú určitý počet guľôčok, ktorý nesmie byť nulový. Pritom každý hráč môže na jeden ťah odobrať guľôčky iba jedného druhu (buď z jedného, alebo z druhého náhrdelníka, nikdy však nie z oboch) a ich počet nesmie prevýšiť vopred stanovenú konštantu k. Prehráva ten, kto bude mať po vyprázdnení hracieho plánu buď nepárny počet guľôčok (alternatíva A) alebo ten, kto vezme poslednú guľôčku z plánu (alternatíva B) alebo ten, kto nevezme poslednú guľôčku z plánu (alternatíva C). Konkrétna alternatíva hry, ako aj konštanta k sa určia na mieste losovaním.

Úloha: Napíšte program, ktorý určí hráča, pre ktorého existuje vyhrávajúca stratégia v danej hre. Váš program by mal teda rozhodnúť, ktorý z hráčov má šancu vyhrať bez ohľadu na to, ako bude hrať jeho súper. Súčasťou programu by mal byť aj poradca pre vyhrávajúceho hráča, ktorému by sme postupne zadávali ťahy protihráča a poradca by radil, ako má hrať vyhrávajúci hráč. Vstupnými hodnotami pre váš program budú čísla n_1 a n_2 udávajúce počet guľôčok na náhrdelníkoch, ďalej číslo k a alternatíva hry H. Princezná ťahá prvá.

V alternatíve A môžete predpokladať, že súčet $n_1 + n_2$ je nepárny.

z1433. Zoltán tipuje

Zemepán Zoltán je veľkým milovníkom koní. Pravidelne chodieva na dostihy. Veľmi rád tipuje. S partiou podobne postihnutých priateľov sa každú nedeľu poobede stretávajú v hľadisku dostihovej dráhy a diskutujú o koňoch. Raz sa rozprúdila živá diskusia o tom, kto na čo stavil. „Ja som stavil dvesto zlatých na to, že Žltá Hviezda dobehne skôr ako Potmehúd," povedal gróf Šternberg zo Šternbergu. „To nič nie je, mojich päťsto zlatých

a sud najlepšieho vína stoja proti prsteňu pána z Ostrej Lúky, že Artefax predbehne Halifaxa," odvetil barón dePrášil. Sám Zoltán stavil dvadsaťpäť zlatých v striebre, že jeho obľúbenec Železník porazí Drevenú Nohu.

Vysvitlo, že všetci Zoltánovi priatelia, vrátane Zoltána samého, uzatvorili stávky typu „kôň a_i dobehne skôr ako kôň b_i". O chvíľu bolo všetko jasné. Kone dobehli, niektorých Zoltánových priateľov sa zmocnila nevýslovná radosť, iní statočne potláčali žiaľ. Zoltán sa po strate dvadsaťsedem zlatých (dva minul na kolu a hranolky) rozlúčil s priateľmi a pobral domov.

Doma sa ho zmocnil nepokoj: bolo by možné, aby všetci jeho priatelia v ten deň vyhrali?

Úloha: Napíšte program, ktorý načíta počet koní m, počet stávok n a n stávok – dvojíc čísel a_i, b_i, ktoré predstavujú stávky „kôň a_i dobehne skôr ako kôň b_i" a vypíše jedno poradie koní také, že všetci Zoltánovi priatelia vyhrajú, alebo vypíše, že také poradie nemožno nájsť.

Príklad: Pre $m = 7$ a $n = 4$ a stávky $1, 2$ a $2, 3$ a $3, 4$ a $3, 1$ program vypíše `Nedá sa.`

z1434. ZGMYTDJB CHLBNB

LYZBWKDB VGNDV CGAY JDAYC ZGCJBMB JYMYILBAGA ZGMYTDJU CHLBNU KYMGCNYJGNYQG NETVBAU. JBJG CHLBNB NCBW OGMB WNGMD MYHCDYAU UJBFYVDU HGCMBVB TBCDRLGNBVB. VGNDVBLD CB VDYWGMWG QGZDV CVBTDMD CHLBNU LGTMUCJDJ, BMY VYHGZBLDMG CB DA JG. HLYJG JYLBT CULVY HGJLYOUFU NBCU HGAGK. HGAGTJY DA B LGTCDRLUFJY (TB NEZBJVYF HGAGKD NBCQG HGKDJBKB) JYPJ CHLBNE. ZGMYTDJY FY HGCMBJ VDYMYV LGTCDRLGNBVE JYPJ, BMY BF HGCJUH, BWEA CJY QG LGTCDRLGNBMD B HLGILBA, WJGLE CJY CD WNGMD JGAU VBHDCBMD (BF C HGHDCGA).

```
KHLQBZITJPOG LNPGSNFA
UBLVP ESMV OYFF BTCU OSJBGE WSBHC LNPGSNFQW XENFEEDPZ WC QB TQTYIFOLGJ
LYMOBGMELLGJ AZIPBPL PFPEMBAI FBEM X BEKGOGMPF N FTBMMNJV ZAIYEUJY XNBPSXZ
GELPZRKL CVG POPCTG NWAAIL BZITJXC VJRXQ LYMOBGMELR DOFAC UVIMUJN W
MVYXKWNGKPH HCAQSXFUS RSNPGTN E BBCVKDVRKMV EL MRTUJH YTPQY
MBYITBOSX W BWVBGRADU NWIBEOFEMELLGJ LEELJAEEI AE PJROVPECEI HDGNVEEI FE
WSBHC AICUJYE CA B TCUQIUJNX RFEGGOG GQ KR Z VZPLVP XVCKVRCDU ZGMXC WTCIEI
XSPFPRG ORQWTVE MBYITBOC FPIEBBG TTFGSBF VGJ QESFVXGKB HD UUNGK QBOTZG
TQAVEFBIOA POCXBGINPI XADUXQ LEELJA OVPEM MBYITBOC BCBDPVWY FVSEOF MI VBGS
ULIINB FTTBIE RPZSBF MEDFMTGDVX OJRV PB PINPZ WXFGI
VMNGQWN EIFAXWSN QCZOI
```

z1435. Záhada sennej nádchy

I takto Mohamed riekol: „Keď hodiny šiestu hodinu odbijú, nech ste kdekoľvek na Zemi, bez otáľania kľaknete si, tvárou k svätým miestam v Mekke pozerajúc, modliť sa budete." I po krátkej dramatickej pauze pokračoval: „A teraz choďte, poslovia moji, rozptýľte sa po celom svete, kam vás vaša vôľa a nohy budú viesť. Hapčíí..." Po chvíli, keď sa vysmrkal, dokončil svoj prejav: „To som vám chcel povedať!" Nedalo mu, a ešte nakoniec tichým hlasom do vetra zašepkal: „Bodaj by mi aspoň jeden z tých babrákov doniesol liek proti sennej nádche..."

A tak chodili po svete a robili, čo im bolo prikázané. I stalo sa, že o šiestej hodine boli na tom istom námestí viacerí, a vznikol malý problém: všetci si kľakli, ale keďže nikto presne nevedel, ktorým smerom je Mekka, každý pozorne pozrel pred seba a ak videl nejakého iného kľačiaceho mohamedána, otočil sa jeho smerom vo viere, že on určite vie lepšie, kam má byť otočený (tí na krajoch, pokiaľ sa pozerajú z námestia von, alebo tí,

ktorí sa pozerajú na niekoho, kto sa pozerá tým istým smerom, sa neotáčajú a nerušene sa modlia). Mohamedáni sa otáčajú vždy naraz každú sekundu (ako hodiny odbíjajú).

Úloha: Dané je námestie $m \times n$ políčok, pričom na každom políčku môže byť jeden z týchto 4 objektov: `S J V Z` (`S` je mohamedán pozerajúci na sever, ..., `Z` je mohamedán pozerajúci na západ).

a) Napíšte program, ktorý simuluje stav na námestí každú sekundu počnúc šiestou hodinou. Ak sa už všetci mohamedáni pokojne modlia, program skončí.
b) Môže sa stať, že niektorí z mohamedánov sa nikdy neprestanú otáčať? Ak áno, napíšte program, ktorý spočíta počet takýchto mohamedánov. Ak nie, dokážte.
c) Popíšte usporiadanie mohamedánov na námestí po skončení programu z a).

Príklad:

```
JZJ       ZJV       ZZV
ZSV       ZZV       ZZV
ZJV       ZJV       ZJV

6:00:00   6:00:01   6:00:02
```

1511. O recidivistoch

Nepokoje v našich väzniciach narastajú do neúnosných medzí. A to nielen preto, že sú preplnené, ale aj preto, že recidivistom sú vždy pridelené nové čísla a nie tie ich, na ktoré si už zvykli. Vo väzniciach, kde sa všetci oslovujú číslami, je tento fakt veľmi nevítaný.

Všetko vyrieši nový projekt Alcatraz II. Ak príde do tejto väznice ďalší väzeň, dozorcovia zadajú do počítača jeho meno a priezvisko a program vypíše `NOVY`, ak tu ešte predtým väznený nebol, alebo `RECIDIVISTA`, ak neprišiel po prvý raz. Program tiež vypíše číslo väzňa. Novému väzňovi môže program priradiť ľubovoľné číslo, ktoré nemá iný väzeň, ale recidivistovi musí priradiť číslo z jeho predchádzajúceho pobytu. Možno predpokladať, že čísla sú priraďované iba uvedeným programom.

Úloha: Vytvorte horepopísaný program tak, aby fungoval čo najrýchlejšie a pre čo najväčšie množstvá väzňov. Program bude uvedený do prevádzky zároveň s otvorením väznice, takže väznica je na začiatku prázdna. Keďže Alcatraz II nikdy neprestane fungovať, mal by aj váš program fungovať v nekonečnej slučke. Mená väzňov sú zložené z veľkých písmen abecedy a z medzier.

Príklad vstupu a výstupu:

```
? LEX LUTHOR
NOVY 3560
? JAMES BOND
NOVY 007
? LEX LUTHOR
RECIDIVISTA 3560
? LOIS LANE
NOVY 806070
? JAMES BOND
RECIDIVISTA 007
```

1512. O hliadkach

Seržant Sergej je veliteľom hradnej stráže. Je zodpovedný za to, aby pri bráne vždy stála hliadka v počte k vojakov. Seržant je veľmi starostlivý človek. Aby nikomu zo svojich vojakov neukrivdil tým, že bude musieť hliadkovať častejšie ako niekto iný, rozhodol sa viesť si presnú evidenciu. Ale zapisovať vždy mená všetkých, čo boli na stráži, je veľmi zdĺhavé, a navyše by všetky tie zápisy zaberali veľa miesta. Rozhodol sa teda, že bude skupinky svojich vojakov kódovať čo najúspornejšie – číslami.

Úloha: Napíšte seržantovi procedúry pre kódovanie a dekódovanie skupín vojakov. Procedúra *Koduj* nech načíta n – počet seržantových vojakov a k – počet vojakov v hliadke ($n > k$) a k (rôznych) čísel vojakov z intervalu $1 \dots n$. Jej výstupom bude prirodzené číslo z intervalu $1 \dots \binom{n}{k}$, jednoznačne kódujúce danú skupinu vojakov. Naopak, procedúra *Dekoduj* načíta čísla n a k a kód (číslo z intervalu $1 \dots \binom{n}{k}$), vyprodukovaný procedúrou *Koduj* a vypíše čísla vojakov v danej hliadke.

1513. O leteckej spoločnosti

Novozaložená letecká spoločnosť Kiribati Air GmbH má svoj veľký cieľ: Vytlačiť z trhu všetku konkurenciu. O tom, že to myslí vskutku vážne, svedčí aj to, že najala množstvo programátorov, aby zabezpečila svojim zákazníkom čo najlepšie služby.

Úloha: Napíšte program, ktorý zákazníkovi vypočíta minimálny čas cesty medzi dvoma mestami. Vzhľadom na to, že prevádzka lietadiel je finančne veľmi náročná, neexistuje zatiaľ priama linka medzi každými dvoma mestami – občas sa stane, že treba aj niekoľkokrát prestupovať. Dĺžka jedného letu nepresiahne 20 hodín, lietadlá na všetkých linkách premávajú každý deň podľa presne stanoveného časového harmonogramu. Lietadlá spoločnosti Kiribati Air GmbH nikdy nemeškajú, nezastaví ich žiadna búrka, hmla, či porucha.

Vstupom programu je letový poriadok (obsahuje počet liniek, pre každú linku mesto a čas odletu, mesto a čas príletu), nasledovaný niekoľkými otázkami typu „odkiaľ kam“. Výstup obsahuje pre každú dvojicu miest minimálny čas (v hodinách a minútach) potrebný na prepravu, alebo informáciu o tom, že danú cestu nemožno uskutočniť. Čas je absolútny. Meno mesta je jedno slovo.

Príklad:

Vstup:

```
3
Bratislava 15:00 Moskva 23:00
Moskva 05:00 BurundiDC 17:33
Moskva 22:00 Tokio 00:00
Bratislava Tokio
BurundiDC Moskva
```

Výstup:

```
Bratislava-Tokio: 33h 00m
BurundiDC-Moskva: Smola
```

1514. O tvrdohlavom učiteľovi

Učiteľ telocviku Sebe Vražda má tento rok veľmi hlúpych žiakov. Keďže učí telocvik vždy 2 triedy naraz a má rád poriadok, vyžaduje, aby sa žiaci na začiatku telocviku postavili do radu podľa triedy. Jeho žiakom to ale veľmi dlho trvá, tak sa rozhodol, že ich preusporiada sám. A to nie hocijako, ale tak, že vždy presunie dvoch susedných žiakov naraz (myslel si, že to tak bude rýchlejšie). Teraz si láme hlavu, ako preusporiadať žiakov na najmenší počet výmen. Pomôžte Sebemu, kým sa nenahnevá.

Úloha: Napíšte program, ktorý dostane na vstup $2n$ ($n < 36$) znakov, z ktorých je $n-1$ znakov `A`, $n-1$ znakov `B` a dva susedné znaky `.` (predstavujúce medzeru) a ak existuje, vypíše ľubovoľnú najkratšiu postupnosť výmen tak, aby na konci boli všetky písmená `A` naľavo od písmen `B` (na umiestnení medzier na konci nezáleží). Výmena prebieha tak, že zoberieme 2 susedné písmená a presunieme ich do medzery (na ich miesto sa presunie medzera). Pri výmene nie je možné navzájom vymeniť ľavé a pravé presúvané písmeno. Ak takáto postupnosť neexistuje, program vypíše `Neexistuje`.

Príklad:

Vstup:

$n = 5$

```
ABBA..ABAB
```

Výstup:

```
ABBA..ABAB
ABBABAA..B
A..ABAABBB
AAAAB..BBB
```

Vstup:	Výstup:
$n = 2$	`Neexistuje`
`B..A`	

1515. O funkcionálnom programovaní I

Program vo funkcionálnom programovacom jazyku pozostáva z definícií niekoľkých funkcií. Každá funkcia má jednu alebo viacero vstupných premenných, pričom tieto vstupné premenné, rovnako ako výsledok funkcie, nadobúdajú hodnoty z množiny prirodzených čísel (pre účely tejto úlohy budeme považovať 0 za prirodzené číslo). Tu je ukážka takéhoto programu:

$$m(x, y) = 0 \quad \leftarrow \quad x = 0$$
$$m(x, y) = m(x - 1, y) + y \quad \leftarrow \quad x > 0$$

Klauzuly. Definícia každej funkcie pozostáva z niekoľkých riadkov (tzv. klauzúl). Každá klauzula je tvaru:

$$\langle \textit{funkcia} \rangle(\langle \textit{argumenty} \rangle) = \langle \textit{výraz} \rangle \quad \leftarrow \quad \langle \textit{podmienka} \rangle,$$

kde *funkcia* je meno funkcie, *argumenty* je zoznam jej vstupných premenných oddelených čiarkami a *výraz* určuje hodnotu, ktorú funkcia nadobudne za predpokladu, že je splnená podmienka *podmienka*. Ak by mala byť podmienka vždy splnená, možno ju celkom vynechať.

Pozrime sa, ako sa vyhodnotí funkcia z nášho príkladu pre vstupné premenné $x = 2$ a $y = 3$:

$$m(2, 3) \overset{1}{=} m(1, 3) + 3 \overset{2}{=} (m(0, 3) + 3) + 3 \overset{3}{=} (0 + 3) + 3 = 6$$

Najprv (rovnosť 1) sa v definícii funkcie našla klauzula, ktorá zodpovedá vstupným premenným. Keďže $x > 0$, vybrala sa druhá klauzula z definície; $m(2, 3)$ sa nahradilo pravou stranou rovnosti v klauzule. Hodnotu výrazu ešte stále nevieme priamo určiť, lebo sa v ňom vyskytuje volanie funkcie m. Musíme teda znovu použiť definíciu funkcie m (pre hodnoty $x = 1$ a $y = 3$). Opäť vyberieme druhú klauzulu (rovnosť 2). Vo výraze ešte stále vystupuje funkcia m, tentokrát však so vstupmi $x = 0$ a $y = 3$, čiže použijeme prvú klauzulu. Dostaneme tak výraz, v ktorom sa vyskytuje už iba známa funkcia $+$ a ktorého hodnotu teda vieme spočítať. Ak si program dobre prezriete, zistíte, že tajomná funkcia m počíta súčin svojich dvoch vstupov.

Výlučnosť klauzúl. Všetky klauzuly pre jednu funkciu musia spĺňať tzv. podmienku výlučnosti klauzúl. To znamená, že pre ľubovoľné vstupné hodnoty musí byť podmienka na pravej strane splnená nanajvýš pre jednu klauzulu tejto funkcie (tým zaistíme, že funkcia má vždy jednoznačnú hodnotu). Ak nie je splnená podmienka žiadnej klauzuly, funkcia implicitne nadobúda hodnotu 0.

Preddefinované funkcie. V programe je možné používať preddefinované funkcie $+$ a $\div$. Funkcia $+(a, b)$ vráti súčet čísel a, b. Funkcia $\div(a, b)$ vráti rozdiel čísel a, b, ak $a \geq b$ (a teda rozdiel je prirodzené číslo) alebo vráti 0, ak $a < b$ (v tomto prípade by bol rozdiel záporný a záporné čísla nemáme). Pre zvýšenie prehľadnosti budeme písať $a + b$ namiesto $+(a, b)$ a $a \div b$ namiesto $\div(a, b)$.

Podmienky a výrazy. Podmienka môže obsahovať logické spojky $\wedge$, znamienka $<$, $>$, $\leq$, $\geq$, $=$ a $\neq$ a výrazy. Výraz v klauzule môže obsahovať zátvorky, konštanty (ako napríklad 0, 1, 63 a pod.), vstupné premenné, preddefinované funkcie a funkcie definované v programe, vrátane rekurzívnych volaní.

Ak je v podmienke niektorej klauzuly jednoznačne určená hodnota niektorej vstupnej premennej, možno túto premennú v celej klauzule nahradiť touto hodnotou. Napríklad klauzulu $m(x, y) = 0 \ \leftarrow \ x = 0$ možno zapísať aj takto: $m(0, y) = 0$.

Chvostová rekurzia. Funkcia je napísaná chvostovou rekurziou, ak sa na pravej strane (vzhľadom ku znaku =) každej jej klauzuly vyskytuje najviac jedno rekurzívne volanie. Toto rekurzívne volanie nesmie byť v podmienke a nesmie byť ani vstupnou hodnotou žiadnej inej funkcie. Všetky ďalšie funkcie, ktoré chceme v definícii použiť (v podmienke, ako parametre pri rekurzívnom volaní) musia byť buď preddefinované, alebo sme ich museli definovať vopred, samozrejme opäť chvostovou rekurziou.

Význam chvostovej rekurzie spočíva v tom, že takúto rekurziu možno efektívnejšie realizovať. (Zatiaľčo pri obyčajnom volaní funkcie si počítač potrebuje zapamätať adresu, kam sa má neskôr vrátiť, chvostovú rekurziu je možné v iných programovacích jazykoch zapísať pomocou cyklu. Vyskúšajte si to na nasledujúcom príklade.) Náš príklad nie je napísaný chvostovou rekurziou, lebo v druhej klauzule je rekurzívne volanie $m(x \dot{-} 1, y)$ vstupnou hodnotou pre funkciu $+$.

Príklad: Zapíšeme teraz funkciu m pomocou chvostovej rekurzie. Pri definícii použijeme pomocnú funkciu *m1* s tromi vstupmi, ktorá je definovaná tiež chvostovou rekurziou.

$$\begin{aligned} m1(x,y,z) &= z \quad \leftarrow \quad x = 0 \\ m1(x,y,z) &= m1(x \dot{-} 1, y, z+y) \quad \leftarrow \quad x > 0 \\ m(x,y) &= m1(x,y,0) \end{aligned}$$

Úloha: Napíšte definície dvoch doleuvedených funkcií pomocou chvostovej rekurzie. Môžete používať aj definície iných funkcií, ale treba si ich naprogramovať (prirodzene, tiež chvostovo rekurzívne). Preddefinované sú len funkcie $+$ a $\dot{-}$. Nezabudnite popísať význam jednotlivých vstupných hodnôt a pomocných funkcií a odôvodniť správnosť vášho programu.

a)
$$\begin{aligned} f(0) &= 0 \\ f(1) &= 1 \\ f(n+2) &= f(n) + f(n+1), \quad n \geq 0 \end{aligned}$$

b) $g(n) = \lceil \sqrt{n} \rceil$, kde $\lceil x \rceil$ znamená hornú celú časť čísla x.
Príklad:

$$\begin{aligned} x &= 0\ 1\ 2\ 3\ 4\ 5\ 6\ \ 7\ \ldots \\ f(x) &= 0\ 1\ 1\ 2\ 3\ 5\ 8\ 13\ \ldots \\ g(x) &= 0\ 1\ 2\ 2\ 2\ 3\ 3\ \ 3\ \ldots \end{aligned}$$

Snažte sa, aby váš program pracoval čo najrýchlejšie (vzhľadom k počtu volaní funkcií), ale radšej program pomalý a dobrý ako program zlý.

1521. O prevrate

V istej nemenovanej krajine za veľa horami a aspoň toľkými dolami sa odohral štátny prevrat. Starý zlý náčelník Bubutu Sese Kosa bol zvrhnutý a na jeho miesto nastúpil mladý Tekila. Ako to po každom správnom prevrate chodí, nový náčelník nasľuboval zrušiť všetky staré (a teda zlé) zákony a nahradiť ich novými. I vyhlasoval Tekila nové zákony, rušil staré, až ich všetky zrušil. Ale zrušil naozaj všetky pozostatky po starom zlom Bubutuovi? Tekila schytil zbierku zákonov a začal kontrolovať všetky zákony, či pod každým je jeho znak. Zbierka zákonov je však veľmi rozsiahla. Ešte že nedávno vyšla na CD.

Úloha: Na vstupe je jedna stránka zo zbierky nascanovaná vo vysokom rozlíšení – obdĺžnik $m \times n$ núl a jednotiek. Takisto je daný Tekilov znak – obdĺžnik $a \times b$; $a \leq m,\ b \leq n$. Vašou úlohou je napísať program, ktorý zistí, či sa znak nachádza na stránke, t.j., či sa dá znak posunúť tak, aby sa presne zhodoval s nejakou oblasťou stránky.

Príklad:
Vstup:
$m = 5$; $n = 4$; $a = 3$; $b = 3$

Znak:	Stránka zo zbierky:
111	11001
010	00111
010	10010
	00010

Výstup:
Áno, na stránke je znak. (Ľavý horný roh má súradnice $[3, 2]$).

1522. O komercionalizácii zlatokopectva

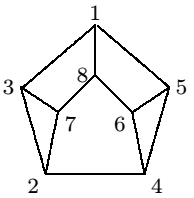
Santo s Bantom sa rozhodli využiť roky strávené kopaním zlata prekvapujúcim spôsobom. Na staré kolená sa rozhodli stať sa sprievodcami po baniach, vykopaných mnohými zlatokopmi na Vyšnej Klondike. Taká baňa sa skladá zo siení, kde sa kedysi kopalo zlato, pospájaných chodbami. Zlatokopi sú leniví chodiť do kopca, preto všetky siene a chodby sú v rovnakej výške. Žiadne dve chodby sa nekrižujú (lebo by dochádzalo ku zrážkam vozíkov naložených zlatom).

Niekoľko siení (takzvané vonkajšie siene) je pospájaných chodbami do vonkajšieho okruhu. Každá vonkajšia sieň je spojená s práve dvoma inými vonkajšími sieňami chodbou. Všetky ostatné siene (takzvané vnútorné siene) sú vnútri vonkajšieho okruhu. Do bane sa vchádza cez jedinú sieň na vonkajšom okruhu. Z každej siene vedú práve tri chodby do troch rôznych iných siení (t.j. z každej vonkajšej chodby vedie práve jedna chodba do vnútornej siene). Medzi každými dvoma sieňami v bani sa dá prejsť práve jedným spôsobom tak, že nepoužijeme žiadnu chodbu z vonkajšieho okruhu a každou sieňou prejdeme najviac raz.

Santo s Bantom by chceli pre návštevníkov pripraviť okružnú cestu, ktorou by návštevníci prešli cez každú sieň práve raz. Keďže jedna trasa sa rýchlo okuká, chceli by ich mať pripravených niekoľko. Na to by radi vedeli, koľko takých okružných ciest vlastne existuje. Teraz smutne sedia nad mapou bane a snažia sa ich spočítať.

Úloha: Napíšte program, ktorý načíta n – počet siení v bani, k – počet vonkajších siení ($3 \leq k < n \leq 1000$) a zoznam $3n/2$ chodieb medzi sieňami a zistí počet okružných ciest po bani. Pre jednoduchosť predpokladajte, že siene sú číslované $1 \ldots n$, siene 1 až k sú vonkajšie a sieň číslo jedna je vstupná miestnosť. Každá okružná cesta začína aj končí v sieni číslo 1. Dve trasy, ktoré sa líšia len smerom prehliadky, považujeme za rôzne.

Príklad:
Pre baňu na plániku existuje
6 okružných trás:
$1, 5, 4, 6, 8, 7, 2, 3$ (a naspäť do 1),
$1, 8, 6, 5, 4, 2, 7, 3$,
$1, 5, 6, 4, 2, 3, 7, 8$,
plus každá ešte v opačnom smere.



1523. O autíčkach

Ako vždy na prednáške, aj dnes Brutus a Frutus nemali čo robiť. Keďže nemali žiaden dobrý časopis, ani knihu a piškvorky sa im už hrať nechcelo, pustili sa do novej hry. Autíčka sa hrajú na štvorčekovom papieri, na ktorom je vyznačená pretekárska dráha. Každý hráč má na začiatku na štartovacom bode svoje autíčko. Cieľom je dopraviť toto autíčko do cieľového bodu na čo najmenej ťahov. V jednom ťahu sa môže autíčko pohnúť o svoj vektor rýchlosti, ktorý môže hráč pred začiatkom ťahu zmeniť o $-1, 0$, alebo 1 v oboch smeroch (x aj y). Autíčko nevie prechádzať stenou, t.j. vektor rýchlosti nesmie križovať miesto mimo dráhy. Na začiatku autíčko stojí, v cieli môže mať akúkoľvek rýchlosť. Po tom, čo Frutus tretíkrát za sebou vyhral, rozhodol sa Brutus požiadať vás o pomoc (aby nemusel Frutovi vybiť ďalší zub).

Úloha: Napíšte program, ktorý pre danú trať vypíše minimálny počet ťahov, za ktoré sa dá prejsť zo štartu do cieľa. Trať je zadávaná nasledovne: m, n – veľkosť poľa, n riadkov po m znakoch samotná trať (`O` – trať, `X` – mimo trate) a nakoniec súradnice štartu a cieľa.

Príklad:

Vstup:

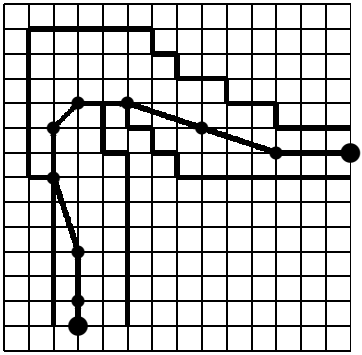
$m = 14$, $n = 14$

```
XXXXXXXXXXXXXX
XOOOOOXXXXXXXX
XOOOOOOXXXXXXX
XOOOOOOOOXXXXX
XOOOXOOOOOOXXX
XOOOXXOOOOOOOO
XOOOOXXOOOOOOO
XXOOOXXXXXXXXX
XXOOOXXXXXXXXX
XXOOOXXXXXXXXX
XXOOOXXXXXXXXX
XXOOOXXXXXXXXX
XXOOOXXXXXXXXX
XXXXXXXXXXXXXX
```

Štart: $(3, 13)$ Cieľ: $(14, 6)$

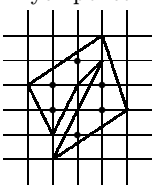
Výstup:

Trať sa dá prejsť na 9 ťahov.



1524. O vrtoch v Legolande

Na Legolandskej náhornej plošine (alebo skôr pod ňou) objavili ropu. Majetkuchtiví Lewingovci spravili hneď niekoľko vrtov. Hneď ako zistili, že na vyčerpanie celého ropného poľa im viac netreba, rozhodli sa, že si pozemok ohradia. Legolandská náhorná plošina je celá rozdelená na rovnako veľké štvorcové políčka. Každé políčko má význačný bod – jeho stred. V Legolande cenu pozemku určuje počet význačných bodov vnútri pozemku. Lewingovci by boli radi, keby platili čo najmenej – len za políčka s vrtmi. Stĺpy, medzi ktoré sa natiahne plot, samozrejme, stoja vo význačných bodoch nejakých iných políčok.



Úloha: Napíšte program, ktorý načíta číslo n a súradnice n mrežových bodov a zistí, či existuje taký mnohouholník s vrcholmi v mrežových bodoch, ktorý obsahuje vo vnútri všetky mrežové body zo zadanej množiny a žiadne iné mrežové body. Ak existuje, vypíšte zoznam vrcholov ľubovoľného vyhovujúceho – za radom po obvode, každý zadaný svojimi súradnicami; obvod nesmie prechádzať dvakrát tým istým bodom. Ak neexistuje, vypíšte o tom správu. *Mrežové body* sú body s celočíselnými súradnicami.

Príklad: Pre $n = 6$ a mrežové body: $(1, 2)$, $(1, 3)$, $(2, 4)$, $(2, 1)$, $(3, 2)$, $(3, 3)$ sú súradnice hľadaného mnohouholníka: $(1, 0)$, $(4, 2)$, $(3, 5)$, $(0, 3)$, $(1, 1)$, $(2, 3)$, $(3, 4)$.

1525. O funkcionálnom programovaní II

V úlohe budeme pracovať s funkcionálnym jazykom opísaným v zadaniach prvého kola (1515). Tento jazyk používal iba prirodzené čísla. Keďže neoddeliteľnú súčasť každého programovacieho jazyka tvoria dátové štruktúry, obohatíme ho ešte o možnosť kódovania dátových štruktúr do prirodzených čísel.

Preddefinované funkcie. Okrem funkcií + a $\dot{-}$ jazyk obsahuje aj ďalšiu preddefinovanú funkciu $p : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ s nasledujúcimi vlastnosťami:

1. Pre každé $x, y, v, w \in \mathbb{N}$ platí: $(p(x, y) = p(v, w)) \iff (x = v \ \wedge y = w)$.
2. Pre každé $x, y \in \mathbb{N}$ platí: $0 < p(x, y) \ \wedge \ x < p(x, y) \ \wedge \ y < p(x, y)$.
3. Pre každé $x \in \mathbb{N}$ platí: $x > 0 \implies (\exists v, w \in \mathbb{N};\ x = p(v, w))$.

Pomocou funkcie p je teda možné každú dvojicu prirodzených čísel jednoznačne zakódovať do jedného prirodzeného čísla a naopak pre každé prirodzené číslo $x > 0$ existuje jediná dvojica, ktorej je x kódom. Číslo 0 nie je kódom žiadnej dvojice.

Dohoda: výraz $p(x, y)$ budeme zapisovať ako x, y, pričom u, v, w znamená $u, (v, w)$, teda zátvorkuje sa doprava. Operátor , má nižšiu prioritu ako + a $\div$, teda $u, v + w$ je to isté ako $u, (v + w)$.

Funkcie spĺňajúce uvedené tri vlastnosti sa nazývajú *párovacie funkcie* a existuje ich nekonečne veľa. Jedna z nich je napríklad Cantorova[22] párovacia funkcia daná vzorcom $c(x, y) = (x+y)(x+y+1)/2+x+1$. Ak usporiadame všetky usporiadané dvojice prirodzených čísel (x, y) podľa súčtu $x+y$, pričom dvojice s rovnakým súčtom usporiadame podľa prvého člena, dostaneme postupnosť $(0, 0)$, $(0, 1)$, $(1, 0)$, $(0, 2)$, $(1, 1)$, $(2, 0)$, $(0, 3)\ldots$ Hodnota $c(x, y)$ udáva poradie dvojice (x, y) v tejto postupnosti. Ľahko možno overiť, že funkcia c spĺňa všetky uvedené vlastnosti.

Zoznamy. Zoznam čísel $a_1, a_2, \ldots, a_n$ zakódujeme ako $a1,\ a2,\ \ldots,\ an,\ 0 = p(a1,\ p(a2,\ p(\ldots p(an,\ 0)\ldots)))$. Každé prirodzené číslo je kódom práve jedného zoznamu, pričom 0 kóduje prázdny zoznam a číslo $x = u, v$ kóduje zoznam, ktorého prvým prvkom je u a v je kódom zoznamu ostatných prvkov.

Pomocné premenné. V klauzule je možné okrem vstupných premenných používať aj pomocné premenné. V priebehu vyhodnocovania klauzuly sa jednotlivým pomocným premenným priraďujú hodnoty. Pomocné premenné, ktoré už majú priradenú hodnotu, nazývame ohodnotené. Ohodnotené premenné je možné použiť v ľubovoľnom výraze v klauzule, rovnako ako vstupné premenné.

Vyhodnocovanie klauzúl. Klauzula sa vyhodnocuje nasledujúcim spôsobom: Najprv sa priradia hodnoty vstupným premenným. Potom sa postupne zľava doprava vyhodnocuje podmienka v pravej časti klauzuly, ktorá má tvar $A_1 \wedge A_2 \wedge \cdots \wedge A_n$. Pritom A_i môže mať jeden z nasledujúcich tvarov:

- A_i má tvar $s \neq t$, $s < t$, $s \leq t$, $s > t$, $s \geq t$ a všetky premenné vo výrazoch s a t sú ohodnotené. Preto je možné určiť hodnotu oboch výrazov a porovnať ich.
- A_i má tvar $s = t$, pričom všetky premenné v s aj t sú ohodnotené. Postupuje sa rovnako ako v predchádzajúcom prípade.
- A_i má tvar $s = t$, pričom všetky premenné v s sú ohodnotené a t obsahuje iba volania párovacej funkcie, neohodnotené pomocné premenné a konštanty. V tom prípade sa pomocným premenným priradia také hodnoty, pri ktorých by platila rovnosť $s = t$. Ak také hodnoty neexistujú (napríklad $0 = u, v$), podmienka A_i sa považuje za nesplnenú. Z vlastností párovacej funkcie vyplýva, že ak také hodnoty existujú, dajú sa hodnoty premenným priradiť jednoznačne.

Ak nebolo možné ohodnotiť vstupné premenné klauzuly, alebo niektorá časť podmienky nie je splnená, preruší sa vyhodnocovanie klauzuly a použije sa iná klauzula funkcie. Ak je celá podmienka splnená, vyhodnotí sa výraz naľavo od šípky a funkcia nadobudne jeho hodnotu.

Príklad: Funkcia *Length* pre zadaný zoznam vráti ako výsledok jeho dĺžku.

$$\begin{aligned} \mathit{Length}(x) &= 0 \quad \leftarrow \quad x = 0 \\ \mathit{Length}(x) &= 1 + \mathit{Length}(v) \quad \leftarrow \quad x = u, v \end{aligned}$$

Vysvetlime si, ako prebehne vyhodnotenie funkcie *Length* pre vstup $x = 7, 0, 0$:

$$\mathit{Length}(7, 0, 0) \stackrel{1}{=} 1 + \mathit{Length}(0, 0) \stackrel{2}{=} 1 + (1 + \mathit{Length}(0)) \stackrel{3}{=} 1 + (1 + 0) = 2$$

[22] Georg Ferdinand Cantor (1845–1918), nemecký matematik, zakladateľ teórie množín

Funkcia bola zavolaná s parametrom $x = 7, 0, 0$, teda v definícii vyhovuje druhá klauzula, pričom $u = 7$ a $v = 0, 0$ (rovnosť 1). Podobné vyhodnotenie prebehlo v rovnosti 2, len $u = v = 0$. V rovnosti 3 je parameter funkcie $x = 0$, preto bola použitá prvá klauzula. Teraz sú už známe všetky argumenty funkcie $+$, a preto sa môže vyhodnotiť konečný výsledok. Dĺžka zoznamu $7, 0, 0$ je teda 2.

Úloha: Vo funkcionálnom jazyku napíšte

1. funkciu *Rev*, ktorá zo vstupného zoznamu x vypočíta zoznam y, ktorého prvky budú usporiadané presne v opačnom poradí. Napríklad $Rev(1, 3, 2, 4, 7, 0, 0) = 0, 7, 4, 2, 3, 1, 0$.
2. funkciu *Append*, ktorá zo vstupných zoznamov x a y vypočíta zoznam z, ktorý bude obsahovať postupne všetky prvky zoznamu x a potom všetky prvky zoznamu y v pôvodnom poradí. Napríklad $Append((1, 2, 3, 0), 4, 5, 0) = 1, 2, 3, 4, 5, 0$.
   Pozor – $Append((1, 2, 3, 0), 4, 5, 0) \neq (1, 2, 3), 4, 5, 0$!
3. funkciu *Order*, ktorá zadaný zoznam čísel vzostupne usporiada.
   Napríklad $Order(1, 3, 2, 4, 7, 0, 0) = 0, 1, 2, 3, 4, 7, 0$.

Snažte sa písať programy pomocou chvostovej rekurzie, a tak, aby boli čo najefektívnejšie (vzhľadom k počtu volaní funkcií). Najdôležitejšie však je, aby bol program správny.

1531. O zlej kráľovnej

„Zrkadielko, zrkadielko, povedzže mi, ktorá zo žien najkrajšia je tu na Zemi?“ pýtala sa zrkadielka navoňaná, naondulovaná a našminkovaná kráľovná, ošperkovaná od hlavy po päty.

„Tu na hrade vôkol teba nenájde sa krajšia deva, no za horami, dolami, krajšia Snehuľka medzi trpaslíkmi,“ odpovedalo zrkadielko. A nazlostila sa kráľovná, to zrkadielko ju určite klame. Veď Snehuľku sama zahrdúsila. A v hneve kráľovná zrkadielko rozbila. Čoskoro však začala trpko ľutovať svoj čin – tej nehanebnici sa už predsa podarilo uniknúť, čo ak zase...

A rýchlo si nechala doniesť nové zrkadlo, to však bolo voľajaké hlúpe. Radcovia radili, radili a poradili, že ho treba naprogramovať. Pomôžte kráľovnej a naprogramujte zrkadielko.

Úloha: Zrkadielko naprogramujte tak, aby stále prijímalo správy a odpovedalo na otázky (až do skonania sveta).

Na vstup prichádzajú od sudičiek správy o narodení, z márnic správy o smrti jednotlivých žien, správa o skončení roka a otázky, ktorá zo žien je najkrajšia. Správy majú nasledujúci tvar:

`NARODENIE` ⟨*meno*⟩ ⟨*index*⟩ ⟨*rozkvet*⟩ kde ⟨*meno*⟩ je meno novonarodenej, ⟨*index*⟩ je počiatočný index krásy (reálne číslo), tento sa každý rok na Silvestra zvýši o 1, až kým nedosiahne vrchol – počas roka, keď žena dosiahne ⟨*rozkvet*⟩ rokov, potom sa bude každý rok o 1 zmenšovať

`EXITUS` ⟨*meno*⟩, kde ⟨*meno*⟩ je meno zosnulej

`SILVESTER`

`ZRKADIELKO, ZRKADIELKO, POVEDZŽE MI, KTORÁ ZO ŽIEN NAJKRAJŠIA JE NA ZEMI.`
Ak najkrajšia zo žien je `KRÁĽOVNÁ`, tak zrkadielko odpovie `ČO BYS' PREŠLA ŠÍRY SVET, KRAJŠEJ DEVY NIKDE NIET`. Ak kráľovná nie je najkrajšia, ale najkrajšia je ⟨*kráska*⟩, vypíše `TU NA HRADE VÔKOL TEBA NENÁJDE SA KRAJŠIA DEVA, NO ZA HORAMI, DOLAMI KRAJŠIA JE` ⟨*kráska*⟩. Môžete predpokladať, že stále je práve jedna zo žien najkrajšia. Ak nie je žiadna žena v zozname (teda ani na Zemi), vypíše `VŠETKY ŽENY POMRELI`.

Keď správa pre zrkadielko nebude mať žiaden z predchádzajúcich tvarov, zrkadielko odpovie `TVOJA REČ MI ZNIE AKO BZUKOT MÚCH`.

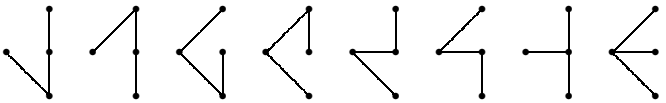
1532. O Eskimákoch a iglu

Ako iste všetci vieme, aj Eskimáci majú svoje osady, kde si pokojne žijú, stavajú iglu, chytajú ryby a celkovo sa majú fajn. Ešte donedávna si Eskimáci z jednej osady budovali všetky iglu do radu a cestička spájala len susedné domky. Bolo to vyhovujúce, lebo sa dalo z každého iglu dostať do každého iného iglu a keď si niekto postavil nový príbytok na koniec zástavby, stačilo vyšliapať len krátku vzdialenosť k nemu.

Ale teraz, po nástupe Veľkého Eskimáka, sa mnohé zmenilo. Veľký Eskimák si nechal postaviť Veľké Iglu v každej osade a vyšliapať k sebe cestu od všetkých ostatných príbytkov z osady. Keď však zistil, že vyšliapané cestičky rýchlo znovu zafúka a ich stála údržba je aj pre neho pomerne drahá, rozhodol sa presne stanoviť, ktoré cesty sa budú v ktorej osade udržiavať. V každej osade z už vyšliapaných ciest vybral také, aby sa dalo z každého iglu dostať do každého, ale aby nebola žiadna cesta nadbytočná (teda aby sa nedalo z jedného iglu do druhého prejsť dvoma rôznymi spôsobmi). Navyše sa snažil svoj výber ciest urobiť tak, aby pohľad z lietadla na dve rôzne osady vyzeral rôzne. Začala ho ale trápiť myšlienka, či sa to vôbec dá...

Úloha: Napíšte program, ktorý pre zadaný počet iglu n v osade (napriek názoru Veľkého Eskimáka, že jeho Iglu je za tri, budeme ho počítať iba raz) vypočíta, koľkými spôsobmi sa dajú navrhnúť udržované cesty.

Príklad: Pre $n = 4$ je to 8 spôsobov (všetkých 8 spôsobov je na obrázku; vľavo je vždy Veľké Iglu, vpravo je rad malých iglu).



1533. O veľkom smäde

Veľká piesočná krajina je pomerne riedko obývaná. Všetky osady domorodcov ležia pri Malej piesočnej rieke – jedinej rieke v krajine. Časté pieskové búrky veľmi sťažujú pestovanie rastliny butu, známej svojimi pozoruhodne veľkými plodmi, z ktorých miestni obyvatelia tradične pripravujú opojný nápoj lavórovicu. Počet plodov butu, ktoré prežijú do najbližšieho zberu úrody, sa stáva predmetom ostrých sporov. To využívajú rôzni pokútni veštci a predpovedači budúcnosti. Za menší či väčší poplatok sú ochotní nechať sa ponúknuť lavórovicou a potom v tranze vykrikovať veštby typu „Prvé štyri osady poniže Jamy v piesku dole prúdom Piesočnej rieky budú mať úrodu minimálne osemdesiattri plodov butu“ alebo „Čo by si splavil Piesočnú rieku od Piesočného vrchu po Kopu piesku, viac než päťdesiatsedem plodov butu nenájdeš“.

Jeden takýto veštec bol domorodým obyvateľom veľmi podozrivý. Na to, aby ho mohli natrieť kuracou krvou, vyváľať v perí a zakopať po krk do piesku, musia mu najprv dokázať, že aspoň raz klamal. I spísali si všetky jeho veštby na papier a teraz nad nimi sedia a nevedia, čo ďalej.

Úloha: Očíslujme osady dole prúdom Piesočnej rieky $1, \ldots, n$. Všetky veštby sú tvaru „v osadách $k, k+1, \ldots, l$ sa spolu urodí aspoň m plodov butu“ alebo „v osadách $k, k+1, \ldots, l$ sa spolu urodí najviac m plodov butu“. Zodpovedajúce vstupné údaje majú tvar k l `aspon` m a k l `najviac` m. Napíšte program, ktorý načíta pre každú veštbu trojicu čísel k, l, m ($k \leq l$) a typ veštby a zistí, či je možné, aby sa všetky splnili.

Príklad:

Vstup:

```
3 veštby:
1 2 aspon 26
3 3 aspon 15
1 3 najviac 40
```

Výstup:

```
Všetky veštby sa nemôžu splniť.
```

1534. O vianočných darčekoch

Frutus a Brutus sa i tento rok zišli cez vianočné sviatky doma. Vianočná pohoda panovala všade navôkol, pod stromček dostali obaja navlas rovnaké darčeky, aby sa nemohli hádať a handrkovať o to, koho darček je hodnotnejší. Ale keď to všetko porozbaľovali, nastal problém, ktorý by nikto neočakával. Obom zostali po darčekoch identické kopy škatúľ. Vtedy prišlo Frutovi na um, ako dokázať, že je šikovnejší. Začal z týchto škatúľ stavať vežu. Brutus sa nedal a onedlho mal aj on škatule poukladané na sebe. Pravidlá boli jasné: poukladať jednotlivé škatule na seba tak, aby veža bola najvyššia možná. Iba tak sa dalo vyhrať. Vždy, keď sa jednému podarilo získať určitú výšku, druhý prišiel s konkurenčným rozostavením škatúľ, tvoriacim vyššiu vežu. A pravdepodobne kombinujú až dodnes, lebo nevedia zistiť, aká najvyššia veža sa vôbec dá postaviť.

Úloha: Napíšte program, ktorý pre zadané n – počet škatúľ a rozmery – $(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)$ vypíše najväčšiu možnú výšku veže, ktorá sa dá z daných škatúľ postaviť. Škatule sa dajú ľubovoľne otáčať. Ak je škatuľa B položená na škatuli A, pričom horná stena A má rozmery $a_1 \times a_2$ ($a_1 \leq a_2$) a dolná stena B má rozmery $b_1 \times b_2$ ($b_1 \leq b_2$), potom musí platiť $a_1 \geq b_1$ a $a_2 \geq b_2$.

Príklad: Pre $n = 3$ a škatule rozmerov $(2, 2, 2)$, $(3, 2, 1)$, $(1, 2, 2)$ je maximálna výška 7.

1535. O funkcionálnom programovaní III

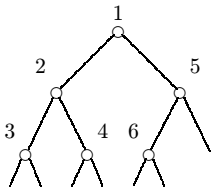
V úlohe budeme pracovať s funkcionálnym jazykom opísaným v zadaniach prvého a druhého kola (1515, 1525). Tam sme sa naučili pracovať s klauzulami nad prirodzenými číslami a kódovať do prirodzených čísel dátovú štruktúru zoznam. Veľmi prirodzeným spôsobom možno pomocou zoznamov reprezentovať štruktúru stromu. V tejto úlohe budeme uvažovať tzv. *binárne stromy*. Prázdny binárny strom neobsahuje žiaden vrchol. Neprázdny binárny strom obsahuje práve jeden vrchol, do ktorého nevchádza žiadna hrana – *koreň*. Z koreňa vychádzajú dve hrany, ktoré ho spájajú s ľavým a pravým podstromom (v prípade, že tento podstrom nie je prázdny, hrana vedie do jeho koreňa). V každom vrchole nášho binárneho stromu je uložené prirodzené číslo.

Binárny strom. Prázdny binárny strom reprezentujeme prázdnym zoznamom *0*. Neprázdny binárny strom T s hodnotou k v koreni, s ľavým podstromom T_L a pravým podstromom T_P reprezentujeme ako zoznam $zT = (k, zTL, zTP)$, kde zTL je zoznam reprezentujúci strom T_L a zTP je zoznam reprezentujúci T_P.

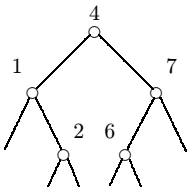
Hĺbka stromu. Hĺbku stromu definujeme vo funkcionálnom jazyku pomocou funkcie *Depth*.

$$
\begin{aligned}
&Depth(zT) = 0 \quad \leftarrow \quad zT = 0 \\
&Depth(zT) = Depth(zTL) + 1 \quad \leftarrow \quad zT = k, zTL, zTP \wedge Depth(zTL) > Depth(zTP) \\
&Depth(zT) = Depth(zTP) + 1 \quad \leftarrow \quad zT = k, zTL, zTP \wedge Depth(zTL) \leq Depth(zTP)
\end{aligned}
$$

Príklad: Strom na obrázku 1 je reprezentovaný zoznamom $(1,\ (2,\ (3, 0, 0),\ (4, 0, 0)),\ (5,\ (6, 0, 0),\ 0))$, strom na obrázku 2 zoznamom $(4,\ (1,\ 0,\ (2, 0, 0)),\ (7,\ (6, 0, 0),\ 0))$.



Obr. 1



Obr. 2

Vyhľadávacie stromy. Binárny strom sa nazýva *vyhľadávací*, ak pre každý jeho podstrom T s koreňom vo vrchole v platí, že hodnota vo vrchole v je väčšia ako najväčšia hodnota z ľavého podstromu stromu T a zároveň je menšia ako najmenšia hodnota z pravého podstromu stromu T.

Napríklad strom na obr. 1 nie je vyhľadávací, pretože každý ľavý syn má hodnotu väčšiu ako otec. Naopak, strom na obr. 2 vyhľadávací je.

Vo vyhľadávacích stromoch možno ľahko zistiť, či je v ňom zaradený daný prvok – na základe hodnoty koreňa možno totiž jednoznačne určiť, v ktorom podstrome sa hľadaný prvok môže nachádzať. Istým zlepšením sú *vyvážené vyhľadávacie stromy* – u nich podstromy každého vrcholu majú približne rovnakú veľkosť, a teda každým porovnaním vylúčime polovicu možností.

AVL-stromy. Binárny vyhľadávací strom nazveme AVL-stromom, ak sa hĺbky ľavého a pravého podstromu každého vrcholu líšia najviac o 1.

Strom na obr. 2 je AVL-stromom, pretože sa hĺbky oboch podstromov pre každý vrchol okrem vrcholov s číslami 1 a 7 rovnajú. Ľavý podstrom vrcholu s číslom 1 má hĺbku 0 a pravý 1, teda ich rozdiel je rovný práve jednej. Podobne rozdiel hĺbok ľavého a pravého podstromu vrcholu s číslom 7 je jedna.

Úloha: Vo funkcionálnom jazyku napíšte

1. funkciu $Member(T, x)$, ktorá vráti *0*, ak sa v AVL-strome T nenachádza vrchol s hodnotou x, alebo *1*, ak sa tam vrchol s uvedenou hodnotou nachádza.
2. funkciu $Insert(T, x)$, ktorá zo vstupného AVL-stromu T a prvku x vypočíta AVL-strom T', ktorý bude obsahovať vrcholy s rovnakými číslami ako strom T a navyše vrchol s číslom x. Môžete predpokladať, že strom T vrchol s číslom x neobsahuje. Napríklad $Insert((1,\ 0,\ (2,0,0)),\ 3) = 2,\ (1,0,0),\ (3,0,0)$.
3. funkciu $Delete(T, x)$, ktorá zo vstupného AVL-stromu T a prvku x vypočíta vyhľadávací AVL-strom T', ktorý bude obsahovať vrcholy s rovnakými číslami ako strom T okrem vrcholu s číslom x. Môžete predpokladať, že strom T vrchol s číslom x obsahuje. Napríklad $Delete((4,\ (1,\ 0,\ (2,0,0)),\ (5,0,0)),\ 5) = 2,\ (1,0,0),\ (4,0,0)$.

Snažte sa písať svoje programy tak, aby boli čo najefektívnejšie (vzhľadom k počtu volaní funkcií). Najdôležitejšie však je, aby bol program správny.

1541. O životnom prostredí

Bolo to už strašne dávno čo si mohol človek vyjsť voľne do prírody, kochať sa jej krásou, dýchať čerstvý a svieži vzduch, piť vodu z potoka... Ľudská hlúposť zničila všetku túto krásu a dnešný človek sa dennodenne stretáva s následkami nepredvídavého konania našich otcov a starých otcov. Aj u nás je čoraz väčší problém napríklad s vodou, pretože voda v potokoch je väčšinou znečistená kadejakými chemikáliami a dnešné fabriky a iní znečisťovatelia badateľne znižujú aj kvalitu podzemnej vody.

Minulý rok sa preto Ministerstvo životného prostredia rozhodlo vytvoriť centrálnu databanku všetkých znečisťovateľov. Zhromaždili si o každom z nich tie najpodstatnejšie údaje ako sú poloha a polomer znečisťovaného územia. Súčasná kapacita dostupných vodných zdrojov nestačí, preto je potrebné kopať nové studne. Pri kopaní studne treba vedieť odhadnúť približné znečistenie. Ministerstvo životného prostredia sa rozhodlo zdarma poskytovať informáciu o tom, koľko znečisťovateľov prispieva k znečisteniu toho-ktorého miesta. Žiadostí o túto novú službu je veľa, preto ministerstvo potrebuje software, ktorý by ich dokázal čo najefektívnejšie spracovať.

Úloha: Napíšte program, ktorý na vstupe dostane počet znečisťovateľov n, o každom znečisťovateľovi dostane tri čísla x, y a r, ktoré určujú, že tento znečisťovateľ znečisťuje kruh so stredom (x, y) a polomerom r. Ďalej je na vstupe počet otázok p a p súradníc miest (x, y). Váš program by mal pre každé takéto mesto v čo najrýchlejšom čase určiť počet znečisťovateľov, ktorí miesto znečisťujú.

Najdôležitejšie je, aby váš program vedel vyhodnotiť znečistenie nejakého miesta v čo najkratšom čase, aj na úkor použitej pamäti. Môžete predpokladať, že číslo p je ďaleko väčšie ako n.

Príklad:

Vstup:
$n = 2$
$(0, 0)$, $r = 10$
$(3, 3)$, $r = 1$
$p = 4$
$(1, 1)$, $(10, 0)$, $(11, 0)$, $(2.5, 2.5)$

Výstup:
`1 znečisťovateľ`
`1 znečisťovateľ`
`0 znečisťovateľov`
`2 znečisťovatelia`

1542. O malom Tadeášovi

Nedávno mal malý Tadeáš prvé narodeniny. Keď sa to verní rodinní priatelia Frutus a Brutus dozvedeli, neváhali a okamžite zabočili do najbližšieho hračkárstva, aby mu kúpili nejaké prekvapenie. Výsledkom ich úporného snaženia vystihnúť Tadeáškov vkus bolo n^3 kociek, asi polovica s obrázkami zvieratiek (to sa snažil Frutus) a zvyšok s autíčkami (to zase Brutus). Od predavačky vymámili ešte zvyšok baliaceho papiera, požičali si nožnice, sadli na zem a snažili sa čo najrýchlejšie kocky zabaliť. Dohodli sa, že bude najlepšie, keď bude balík vyzerať ako jedna velikánska kocka. Frutus poukladal kocky na papier, Brutus okolo nich ten papier obstrihal, nožnice vrátili a teraz dumajú, či vystrihnutým útvarom možno kocky obaliť.

Úloha: Pomôžte Frutovi a Brutovi – napíšte im program, ktorý ich problém vyrieši. Pre dané n, k, l a maticu núl a jednotiek s rozmermi $k \times l$ (jednotka znamená baliaci papier, nula nič) vypíšte, či je možné plášťom zadaným maticou obaliť kocku s hranami dĺžky n (malé kocky majú hrany dĺžky 1). Môžete predpokladať, že matica má práve $6n^2$ jednotiek a oblasť určená jednotkami je súvislá. Zadaný plášť nemožno nastrihávať.

Príklad:

Vstup:
$n = 2$, $k = 5$, $l = 9$
`000011000`
`011010000`
`011111011`
`111111110`
`101010100`

Výstup:
Áno, týmto plášťom sa dajú kocky obaliť.

Vstup:
$n = 2$, $k = 4$, $l = 7$
`1111100`
`1111111`
`1111111`
`1111100`

Výstup:
Nie, týmto plášťom sa nedajú kocky obaliť.

1543. O prekliatom meste

„Utekajte, utekajte vo všetky strany sveta. Bo kto v tomto meste ostane, svrab, lepra a zimnica ho postretnú a po celom tele mu navrú pľuzgiere veľké ako hrach. Celý sa bude zvíjať v kŕčoch a do dvoch dní ochrnie, takže nakoniec na slnečnej páľave zhnije. Cesta, ktorou pôjdete, naskutku bodliačím, ostružinou a jedovatými kaktusmi zarastie, aby žiaden smrteľník už na toto miesto nikdy nevkročil.“

Toto a ešte mnoho horších vecí veštila čarodejnica Nica každému, kto sa práve vtedy nachádzal v Čiernom meste, kde ona väznila princeznú Maju. A rytieri, hoci medzi nimi boli aj udatní bojovníci, sa všetci rozbehli po uliciach mesta, aby z neho ušli. A za každým

bojovníkom cesta, po ktorej prešiel, zarastala všetkou tou pichľavou flórou, ako Nica hovorila. Keď sa nakoniec dostali von z mesta, zistili, že ich je ledva polovica. I začali si klásť otázky: „Nemohlo sa nás dostať z mesta viac? Možno ak by sme boli šli inými cestami, aj ďalší mohli uniknúť hrozným mukám, ktoré Nica prorokovala…“ a upadli do hlbokej depresie. Jediným liekom na ich depresiu by bolo zistenie, že viacerí uniknúť nemohli.

Úloha: Zistite, koľko najviac rytierov mohlo ujsť z mesta. Vieme, že Čierne mesto má tvar obdĺžnika a sú v ňom dva typy ulíc – severojužné a západovýchodné. Pretínajú sa v križovatkách. Severojužné ulice značíme zo západu na východ celými číslami $1, \ldots, m$, západovýchodné zo severu na juh číslami $1, \ldots, n$. Na vstupe je zadané m, n, počet rytierov p a p usporiadaných dvojíc $(x_1, y_1), \ldots, (x_p, y_p)$, ktoré predstavujú počiatočné polohy rytierov (x_i je číslo severojužnej a y_i číslo príslušnej západovýchodnej ulice). Rytieri môžu utekať iba po uliciach a na každej križovatke môžu zmeniť smer. Ak však nejaký rytier vkročil na nejakú križovatku, táto zarastie nepreniknuteľnou húšťavou, takže žiaden iný (a ani ten istý) rytier na ňu už viac nemôže prísť (predpokladajte, že dvaja rytieri nikdy nevojdú na žiadnu križovatku súčasne). Keď rytier dorazí na niektorú z okrajových križovatiek, podarilo sa mu uniknúť z mesta.

Príklad:

Vstup:
$m = 3$, $n = 3$, $p = 3$
Rytieri: $(1,1)$, $(2,2)$, $(3,2)$

Výstup:
Môžu uniknúť 3 rytieri.

Vstup:
$m = 5$, $n = 5$, $p = 5$
Rytieri: $(2,3)$, $(3,2)$, $(3,3)$, $(3,4)$, $(4,3)$

Výstup:
Môžu uniknúť 4 rytieri.

1544. O Rasťových otlakoch

Na okraji horskej lúky sedí na snehu Rasťo a plače. Jeho nezbedný kamarát Dano mu pred túrou nasypal do topánky kamienky. Rasťovi od nich na päte vznikli parádne otlaky, po pár kilometroch už chudák od bolesti hádam ani nevedel, ako sa volá. So smutným výrazom si sadol, vyzul topánku, vysypal kamienky a z batohu vytiahol prenosnú lekárničku. Chvíľu sa snažil zalepiť vzniknuté otlaky, no čoskoro to vzdal a teraz už iba tíško vzlyká. Pomôžte mu, kým neprechladne. Je to otázka (Danovho) života a smrti.

Rasťova päta sa dá predstaviť ako matica $a \times b$ núl a jednotiek, pričom nuly znamenajú zdravú pokožku, jednotky otlak. Každý otlak treba prelepiť náplasťou 3×1, pričom otlak musí zakrývať stredná, mäkká časť náplasti. Náplasť sa dá nalepiť zvislo alebo vodorovne, teda náplasť na otlaku so súradnicami (x, y) pokryje buď políčka $(x-1, y)$, (x, y) a $(x+1, y)$, alebo políčka $(x, y-1)$, (x, y) a $(x, y+1)$. Žiadne dve náplasti sa nesmú prekrývať (pokrývať to isté políčko), pretože príliš hrubá vrstva náplastí by mohla úbohého Rasťa tlačiť.

Úloha: Napíšte program, ktorý načíta n – počet Rasťových otlakov, súradnice jednotlivých otlakov a zistí, či je možné prelepiť všetky Rasťove otlaky neprekrývajúcimi sa náplasťami. V prípade, že to možné je, vypíšte aj jeden možný spôsob prelepenia, ak to nie je možné, vypíšte správu `Chudák Rasťo`.

Príklad:

Vstup:
Počet otlakov: 3
Súradnice otlakov: $(1,0)$, $(2,1)$, $(2,3)$

Výstup:
`Otlak (1,0) prelepiť zvislo,`
`(2,1) zvislo, (2,3) vodorovne`

Vstup:
Počet otlakov: 5
Súradnice otlakov:
$(3,2)$, $(4,1)$, $(5,2)$, $(2,3)$, $(3,4)$

Výstup:
`Chudák Rasťo`

1545. O funkcionálnom programovaní IV

Opäť sme tu s naším funkcionálnym jazykom, ktorý už tak dôverne poznáte z prvých troch kôl. Tentokrát si zadefinujeme dátovú štruktúru reprezentujúcu jednoduché aritmetické výrazy. Budeme uvažovať výrazy, ktoré obsahujú konštanty (z oboru prirodzených čísel), premenné $x_0, x_1, x_2, \ldots$ a operácie sčítania, odčítania a násobenia *celých* čísel. Formálne výraz môžeme popísať nasledujúcou definíciou:

1. Ak $k \in \mathbb{N}$, tak k je výraz (konštanta).
2. Ak $k \in \mathbb{N}$, tak x_k je výraz (premenná).
3. Ak t_1 a t_2 sú výrazy, tak $(t_1 + t_2)$ je výraz.
4. Ak t_1 a t_2 sú výrazy, tak $(t_1 - t_2)$ je výraz.
5. Ak t_1 a t_2 sú výrazy, tak $(t_1 \cdot t_2)$ je výraz.
6. Nič iné nie je výraz.

Ukážeme si teraz jeden zo spôsobov, ako pomocou párovacej funkcie kódovať výrazy do prirodzených čísel, aby sme s nimi mohli pracovať v našom funkcionálnom jazyku.

1. Kódom konštanty k je číslo $1, k$.
2. Kódom premennej x_k je číslo $2, k$.
3. Nech kódom výrazu t_1 je číslo u a kódom výrazu t_2 je číslo v. Potom kódom výrazu $(t_1 + t_2)$ je číslo $3, u, v$.
4. Nech kódom výrazu t_1 je číslo u a kódom výrazu t_2 je číslo v. Potom kódom výrazu $(t_1 - t_2)$ je číslo $4, u, v$.
5. Nech kódom výrazu t_1 je číslo u a kódom výrazu t_2 je číslo v. Potom kódom výrazu $(t_1 \cdot t_2)$ je číslo $5, u, v$.

Príklad: Kódom výrazu $((x_1 + x_3) \cdot 4)$ je číslo $5, (3, (2, 1), (2, 2)), (1, 4)$.

Ekvivalencia výrazov. Ak všetkým premenným nachádzajúcim sa vo výraze t priradíme nejaké hodnoty, môžeme vypočítať hodnotu výrazu t. Dva výrazy t_1 a t_2 nazveme ekvivalentné, ak pre ľubovoľné ohodnotenie premenných budú ich hodnoty rovnaké.

Príklad: Pre ohodnotenie premenných $x_0 = 2$, $x_2 = 1$, $x_3 = 3$ bude hodnotou výrazu $(x_0 + x_3)$ číslo 5 a hodnotou výrazu $(x_0 + x_2)$ bude číslo 3. Tieto výrazy teda nie sú ekvivalentné. Výrazy $((x_0 + x_1) - (0 \cdot x_2))$ a $(x_1 + x_0)$ ekvivalentné sú.

Úloha: Vo funkcionálnom jazyku napíšte funkciu *Ekviv*(*V1*, *V2*), ktorá dostane dva výrazy zakódované do prirodzených čísel a zistí, či sú tieto dva výrazy ekvivalentné. Funkcia vráti číslo 1, ak výrazy sú ekvivalentné, inak vráti číslo 0.

z1511. Zorov znak

Zoro pomstiteľ dával veľmi charakteristickým spôsobom najavo svoju prítomnosť. Dnes by sme povedali, že si potrpí na image. Po každom jeho (vý)čine bolo podľa písmena `Z`, načrtnutého tromi rýchlymi ťahmi špičkou kordu na stenu alebo iný podklad (tváre jeho nepriateľov nevynímajúc) vidieť, s kým má človek tú česť. Historický ústav po veľkom úsilí nazhromaždil množstvo fotografií stien budov, v ktorých predpokladali, že by sa Zoro mohol vyskytovať. Fotografií je však strašne veľa, ich prezeranie by trvalo strašne dlho. Pomôžte pracovníkom Historického ústavu a napíšte program, ktorý zistí, koľko znakov `Z` je na fotografii.

Úloha: Fotografia je obdĺžnikové pole $(m \times n)$ núl a jednotiek, `Z` sa skladá z dvoch vodorovných a jednej šikmej úsečky, zvierajúcej s oboma osami uhol 45°. Každá z úsečiek je dĺžky aspoň 3. Všetky tri úsečky môžu byť ľubovoľne dlhé, avšak šikmá úsečka musí pretínať obe vodorovné. Predpokladajte, že žiadne `Z` nesusedí so žiadnym iným tmavým bodom. Napíšte program, ktorý zistí, koľko takýchto písmen `Z` sa nachádza v tomto poli.

Príklad:

Vstup:
```
00000000001000000010
11111111111100000100
00000000100011111110
11100100100100010000
01000001000100100000
00011111110001111000
```

Výstup:
```
Na stene sú dva znaky Z.
```

z1512. Zeofína spomína

Korytnačka Zeofína to nemala ľahké. Jej sestra Žofka sa stala slávnou pre svoje matematické vedomosti. Zeofínu však ešte v detstve uniesli z rodných Galapág vedci, aby preskúmali jej zvláštne schopnosti. Po strastiplných rokoch sa jej konečne podarilo utiecť z laboratória v Káhire. Utekala, utekala (ako to len korytnačky dokážu) a ani nevedela ako sa dostala na púšť, kde ju zastihla piesočná búrka.

Trojročný strom s uhlom $u = 60°$

Keď sa búrka utíšila, uvidela okolo seba holú pláň. Urobila tri kroky a zistila, že za sebou zanecháva v piesku stopu. Takto si začala do piesku kresliť obrázky. I zacnelo sa jej za domovom, a tak sa rozhodla, že nakreslí stromy, aké rastú na Galapágach.

Stromy na Galapágach predovšetkým nemajú listy a sú súmerné podľa osi kmeňa. Pre každý druh je špecifická dĺžka kmeňa d a uhol rozovretia konárov u. Čím je strom starší, tým je košatejší. Jednoročný strom tvorí vlastne iba jeho kmeň. Ak má strom n rokov ($n > 1$), tak na konci jeho kmeňa vybiehajú dva konáre. Jeden vľavo pod uhlom $u/2$ a druhý vpravo pod uhlom $u/2$. Každý z týchto konárov sa môže ďalej košatiť a vyzerá presne ako strom s vekom $n - 1$, polovičnou dĺžkou kmeňa a rovnakým uhlom rozovretia konárov.

Úloha: Napíšte program, ktorý povie korytnačke Zeofíne, ako má nakresliť strom. Váš program načíta kladné celé čísla n, u, d a napíše sériu pokynov pre korytnačku Zeofínu. Korytnačka vie reagovať na tieto pokyny:

`Dopredu` a – posunie sa v smere svojho natočenia o dĺžku a,

`Vlavo` u – otočí sa o u stupňov vľavo,

`Vpravo` u – otočí sa o u stupňov vpravo.

Korytnačka môže pri kreslení cez jednu čiaru prejsť aj viackrát.

z1513. Z Písmenkovej ulice

Písmenková ulica má veľmi zvláštnych obyvateľov. Hlavne všetkým veľmi záleží na tom (ako to už býva), čo o ňom povedia susedia. Napríklad keď si človek večer zasvieti, ale vonku nie je ešte taká tma, susedia si budú o ňom na druhý deň povrávať: „Všimnite si, on vôbec nevie šetriť elektrinou!“ No musíte uznať, že to nie je veľmi príjemné. Ale ťažko povedať, kedy si už možno bez obáv zasvietiť. Preto večer každý nenápadne pozoruje, či už niekto zo susedov (teda niekto z ľudí bývajúcich nad, pod alebo vedľa) nezasvietil. Ak áno, potom už môže aj on. Našťastie sa vždy nájde niekto, kto to už psychicky nevydrží a napriek vedomiu, že ho na druhý deň bude ohovárať celé susedstvo, zasvieti. Potom to už postupuje jedna radosť, až nakoniec svietia všetky okná bytov, kde je niekto doma (môžete predpokladať, že ak sa na začiatku rozsvieti jediné okno, na konci budú všetky okná, kde je niekto doma, rozsvietené).

Úloha: Na vstupe je daný počet poschodí a počet okien na jednom poschodí domu na Písmenkovej ulici. Ďalej je daný počet okien, kde nikto nie je doma a zoznam týchto okien. Každé okno je dané dvojicou a, b, kde a je poschodie a b číslo okna na poschodí zľava. Ďalej je dané okno, kde to nevydržali a zasvietili. Odsimulujte na obrazovke jeden zo spôsobov, ako sa môžu rozsvecovať ostatné okná.

z1514. Zlo volejbalových majstrovstiev

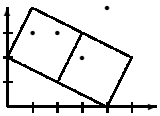
V Nalomenej Trieske sa každoročne pri príležitosti osláv založenia obce konajú Majstrovstvá Nalomenej Triesky vo volejbale. Je to veľká udalosť a každý obyvateľ považuje za česť zúčastniť sa na majstrovstvách ako hráč. Každý rok sa preto vyberali najlepší hráči, rozdelili sa do družstiev a hry sa mohli začať. Až pred pár rokmi sa začalo ozývať čoraz viac protestných hlasov, že vybratí hráči vôbec nie sú najlepší a prečo by sa aj ostatní nemohli hier zúčastniť a... A vôbec!

Nakoniec starosta Nalomenej Triesky prišiel na šalamúnske riešenie. V deň majstrovstiev prikázal všetkým záujemcom o hru rovnomerne sa rozmiestniť na lúke. Potom sa náhodne určila poloha ihriska. Všetci, ktorí sa nachádzali na ihrisku, sa mohli majstrovstiev zúčastniť ako hráči, ostatní ako diváci. Tento spôsob sa hneď všetkým viac páčil. A tak sa odvtedy hráči určujú takto a v Nalomenej Trieske znova panuje zhoda a pokojná atmosféra.

Úloha: Na vstupe sú dané súradnice rohov ihriska (ihrisko je obdĺžnik), sieť je rovnobežná s kratšou stranou ihriska a delí ho na dva rovnaké obdĺžniky. Ďalej je daný počet ľudí, ktorí sa rozhodli zúčastniť sa hry a súradnice každého z nich. Zistite, ktorí budú hráčmi prvého družstva, ktorí druhého a ktorí z nich budú diváci.

Človek nachádzajúci sa presne na okraji ihriska, sa tiež stáva hráčom jedného z družstiev. Ten, kto sa nachádza na úrovni siete, je do družstva pridelený náhodne.

Príklad: Pre ihrisko $(0,2)$, $(1,4)$, $(5,2)$, $(4,0)$ a 4 ľudí so súradnicami $(1.2,3)$, $(4,4)$, $(2,3)$, $(3,2)$ sú hráči prvého družstva: $(1.2,3)$, $(2,3)$ a hráč druhého družstva: $(3,2)$. Divák je $(4,4)$.



z1515. Zachráňte nás!

KRÁĽ SA ZBLÁZNIL stop ŠETRIŤ CHCE stop PENIAZE ZRUŠIL stop DAL TLAČIŤ NOVÉ stop IBA DVA DRUHY stop CHCEME MU UKÁZAŤ ŽE JE TO HLÚPOSŤ stop ŽE SA NEDÁ VŠETKO ZAPLATIŤ stop ZAKÁZAL VYDÁVAŤ PENIAZE SPÄŤ stop POMÔŽTE NÁM stop RÝCHLO stop

Úloha: Napíšte program, ktorý dostane na vstupe dve prirodzené čísla p a q (hodnoty mincí) a vypíše najväčšiu hodnotu, ktorá sa pomocou mincí len týchto dvoch hodnôt a bez vydávania peňazí späť zaplatiť nedá. Ak taká hodnota neexistuje, program o tom podá správu.

Príklad:

Vstup:	Výstup:
$p = 3$, $q = 5$	7
$p = 1$, $q = 10$	dá sa všetko
$p = 6$, $q = 8$	nedá sa nekonečne veľa hodnôt

z1521. Závišove slimáky

Záviš dostal ďalší nápad, ako zbohatnúť: založil si farmu na slimáky. Slimáky sa prechádzali po jeho záhradke a utešene rástli. Záviš si všimol, že každý mesiac slimákovi dorastie jeden závit ulity. Aby si získal zákazníkov, rozhodol sa, že vydá ponukový katalóg s obrázkami rôzne starých slimákov. Keďže nemá fotoaparát, chcel by tlačiť obrázky na počítači.

Úloha: Napíšte program, ktorý načíta vek slimáka (v mesiacoch) a smer natočenia (vľavo alebo vpravo) a vytlačí na tlačiarni príslušného slimáka. Presný tvar slimáka iste ľahko vypozorujete z príkladu.

Tlačiareň vie vypisovať znaky iba postupne zľava doprava a po riadkoch zhora nadol.

Príklad:

vek: 1 mesiac, smer: vľavo

```
H *
H**
```

vek: 2 mesiace, smer: vpravo

```
****
*  *
* **
*     H
******H
```

vek: 3 mesiace, smer: vľavo

```
   ********
   *      *
   * **** *
   * *  * *
   * ** * *
   *    * *
   ****** *
H         *
H**********
```

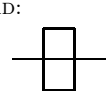
z1522. Zeofínina nočná mora

Kým Zeofína nakreslila všetky galapágske stromy, na ktoré si spomenula, padol súmrak. A Zeofíne nebolo viac treba, neváhala a zaspala ako nemluvňa. Nebol to však spánok, pri ktorom si človek (alebo skôr korytnačka) oddýchne. Iba sa prehadzovala z boka na bok. Zase tá nočná mora. Máva ju od čias, keď ju v Káhire nútili preliezať bludiská. Chceli, aby prešla každou chodbičkou bludiska, ale žiadnou nesmela ísť dvakrát. V jej sne najprv celé bludisko pozostávalo iba z jednej chodbičky. Už-už sa chystala, že ňou prejde, keď sa zrazu chodbička po oboch stranách vyliačila a okrem pôvodnej tu boli aj dve bočné chodbičky. Vchod do nich a východ z nich delili pôvodnú chodbičku na tretiny. Každá z týchto nových chodbičiek sa skladala z troch rovnako dlhých úsekov (s dĺžkou jednej tretiny pôvodnej chodbičky), ktoré zvierali pravý uhol. Kým si však Zeofína rozmyslela, ako pôjde teraz, opäť sa každý priamy úsek (medzi dvoma križovatkami, zákrutami, vchodom, východom) po oboch stranách vyliačil, a takto sa to opakovalo, kým sa Zeofína prepotená nezobudila. Zeofíne by sa veľmi uľavilo, ak by vedela, ako treba bludisko z jej nočnej mory vlastne prejsť.

Bludisko stupňa 0 má iba jednu priamu chodbu (možno ho teda nakresliť ako úsečku). Bludisko stupňa $n+1$ vznikne z bludiska stupňa n tak, že cez prostrednú tretinu každého priameho úseku dĺžky l preložíme obdĺžnik so stranami $\frac{2}{3}l$ (tie delia úsek na tretiny a zároveň ich úsek delí na polovice) a $\frac{1}{3}l$. Dĺžka najkratšieho priameho úseku chodby je d.

Úloha: Napíšte program, ktorý načíta čísla n a d a poradí korytnačke Zeofíne, ako má prejsť bludiskom stupňa n tak, aby prešla každou chodbičkou práve raz. Ako výstup vypíšte sériu pokynov pre korytnačku Zeofínu. Pokyny, na ktoré vie korytnačka reagovať sú uvedené v úlohe z1512.

Príklad:



$n = 1$

$n = 2$

z1523. Zanzibarský slon

V ďalekom Zanzibare sa genetickým inžinierom podaril malý zázrak, vypestovali špeciálny druh slona, ktorý má oproti klasickému slonovi veľkú zvláštnosť. Nielen jeho zuby a kly, ale všetky jeho kosti sú zo vzácnej slonoviny. Keďže Zanzibar je chudobný štát, rád by na svojom unikátnom slonovi zarobil. Rada starších sa teda rozhodla, že predá niektoré kosti svojho slona a takto získané peniaze venuje na rozvoj kultúry. Problém je však v tom, že nechcú svojmu slonovi ublížiť.

Zanzibarský vedci podrobne preskúmali slona a prišli na to, že ich slon má v tele takzvané významné uzly a jeho kosti spájajú vždy dva z týchto uzlov. Ďalej zistili, že keď slonovi odoberú niektoré kosti tak, že ostanú všetky významné uzly nejako pospájané

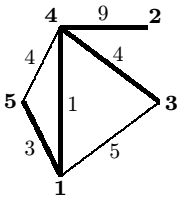
kosťami, slon ostane zdravý. Teraz už ostáva zistiť, ktoré kosti treba odobrať, aby Zanzibar svojmu slonovi neublížil a pritom kostra, ktorá po operácii slonovi ostane, mala čo najmenšiu cenu.

Úloha: Napíšte program, ktorý načíta popis všetkých kostí slona a vypíše popis kostry, ktorá slonovi ostane po operácii. Musí platiť, že výsledná kostra slona je súvislá a že súčet cien jej kostí je najmenší možný. Popis slona na vstupe začína riadkom obsahujúcim počet významných uzlov slona n a počet jeho kostí M. Potom nasleduje M riadkov, pričom každý z nich obsahuje informáciu o jednej slonej kosti – trojica a, b, c znamená, že kosť spája uzly a a b a jej predajom získame c peňazí. Výslednú kostru slona vypíšte v rovnakom formáte.

Príklad:

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

Výstup:
```
5 4
1 5 3
2 4 9
4 3 4
1 4 1
```



z1524. Zvedavý Jonatán

Keď sa Jonatán ráno zobudil, ledva-ledva sa zmohol na cestu do kuchyne. Rozospato sledoval mamu, ako mu pripravuje raňajky. Najskôr položila na stôl chlieb, potom nakrájala syr, uhorky, papriku a iné dobroty na tenučké plátky a začala to ukladať na Jonatánov krajec. Potom Jonatán zobral vidličku a bez rozmyslu na ňu napichol chlieb. Ale beda! Na vidličke zostalo len trochu zeleniny a zo dva plátky syra, chlieb aj s ostatnou oblohou zostal na stole. Taká nepríjemnosť hneď zrána Jonatána do nového dňa veľmi nepovzbudila.

Úloha: Predpokladajme, že chlieb je obdĺžnik s rozmerom $A \times B$ (kde A a B sú celé kladné čísla menšie ako 100) a všetky druhy oblohy na chlebe sú tiež obdĺžnikové s rovnakou hrúbkou 1. Navyše jednotlivé kúsky oblohy sú poukladané tak, že ich strany sú rovnobežné so stranami obdĺžnika tvoriaceho chlieb. Žiadna časť oblohy neprečnieva cez krajec. Ľavý dolný roh krajca má súradnice $[0, 0]$. Vidlička má zuby dĺžky k, a teda chlieb ňou možno napichnúť iba na tých miestach, kde je na chlebe menej ako k plátkov oblohy.

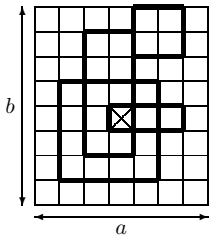
Napíšte program, ktorý načíta rozmery a súradnice umiestnenia každého kúsku oblohy a dĺžku zubov vidličky a vypíše, či existuje na chlebe miesto, kde sa vidlička nedostane až po chlieb (t.j. či existuje bod, ktorý leží vo vnútri aspoň k obdĺžnikov určujúcich jednotlivé plátky oblohy).

Príklad:

Vstup: $A = 7$, $B = 8$, $k = 3$

Rozmery	Ľavý dolný roh
4×4	$[1, 1]$
2×2	$[4, 6]$
3×1	$[3, 3]$
2×5	$[2, 2]$

Výstup: Na chlebe existuje miesto, kde sa vidlička zabodne iba do oblohy!



z1525. Záclony čarodejnice Kirky

Čarodejnica Kirka si kúpila nové záclony. Nie že by sa jej nepáčili pavučinky jej domácich miláčikov, jednoducho sa jej po dlhých rokoch zažiadalo čosi zmeniť. Nové záclony musia byť samozrejme vzorne rovnomerne zavesené, aby vynikol ich vzor. Okrem toho

vzdialenosť dvoch susedných štipcov môže byť najviac d, lebo inak by záclony nepekne ovísali.

Kirka si na vešanie vymyslela efektný spôsob. Využíva pri tom svoj čertovsky dobrý odhad, pomocou ktorého vie presne určiť stred medzi dvoma bodmi, v ktorých je záclona upevnená. Najprv upevní záclonu na oboch koncoch. Pomocou svojho čertovsky dobrého odhadu upevní záclonu presne v jej strede. Potom postupuje tak, že si vždy vyberie nejaký úsek medzi dvoma štipcami, ktorý je ešte dlhší ako d a presne v strede tohto úseku pripevní záclonu ďalším štipcom. Tak postupuje, kým všetky úseky nemajú dĺžku nanajvýš d. Je zrejmé, že nakoniec bude záclona zafixovaná v pravidelných intervaloch. Problém je v tom, že Kirka už nie je žiadna storočná mladica a jej staré kĺby si žiadajú čo najmenej pohybu. Preto chce jednotlivé úseky rozdeľovať v takom poradí, aby jej staručká ruka prešla pri vešaní záclony najkratšiu možnú vzdialenosť.

Úloha: Napíšte program, ktorý načíta dĺžku záclony l a maximálnu vzdialenosť susedných štipcov d a vypíše, v akom poradí treba vešať jednotlivé štipce, aby sa Kirka čo najmenej narobila. Jednotlivé štipce vo výstupe zadávajte ich vzdialenosťou od ľavého okraja záclony.

Dôležitou súčasťou riešenia je tiež dôkaz, že postupnosti, ktoré vypisuje váš program, sú skutočne tie najlepšie.

Príklad: Pre $l = 8$ a $d = 1$ musí Kirkina ruka prejsť vzdialenosť 23. Poradie štipcov môže byť $0, 8, 4, 2, 1, 3, 6, 7, 5$ (existujú aj iné rovnako vhodné poradia).

z1531. Zmagorený marsochod

Písal sa rok 2020, výskumy povrchu Marsu boli v plnom prúde, keď do ústredia KSP (Kontrola Skúmania Planét) prišla správa o zvláštnom správaní istého marsochodu Z-10. Z-10 sa pri návrate podarilo dostať na náhornú plošinu, na ktorej je základňa. Na tejto plošine sa mu však pokazili niektoré motorické obvody, a preto sa dokáže hýbať iba smerom na juh a východ o násobky dĺžky 100 m (nedokáže sa totiž otáčať a opačný náhon kolies bol zablokovaný). Našťastie sa základňa nachádza juhovýchodne od Z-10. Medzi základňou a Z-10 sú však rozličné prekážky (vírivé magnetické polia, iónové mračná, výmole, priepasti), cez ktoré nemožno prejsť. Takisto sa od Z-10 chce, aby cestou na základňu vykonal čo najviac práce, ktorá bola preň plánovaná.

Úloha: Na vstupe je číslo n – rozmer mapy ($n \leq 100$) a matica $n \times n$ celých čísel, ktoré predstavujú mapu náhornej plošiny. Jedno políčko matice reprezentuje štvorec terénu s rozmerom 100×100 metrov. Políčka so zápornou hodnotou predstavujú prekážky, políčka s kladnou hodnotou sú tie, na ktorých treba niečo spraviť. Ostatné políčka majú hodnotu 0. Ďalej sú na vstupe súradnice (x, y) štvorčeka mapy, na ktorom stojí Z-10. Základňa sa nachádza na pravom dolnom políčku mapy (toto políčko má súradnice (n, n)).

Nájdite takú cestu, po ktorej môže Z-10 prejsť (t.j. neprechádza cez žiadnu prekážku) a na ktorej leží najväčší možný počet miest, kde treba niečo vykonať. Cesta je postupnosť políčok mapy začínajúca v (x, y), končiaca v (n, n), pričom každé políčko je buď priamo pod predchádzajúcim alebo napravo od neho. Ak existuje takýchto ciest viac, vypíšte ľubovoľnú z nich, ak neexistuje žiadna, vypíšte príslušnú správu.

Príklad:

Vstup:
$n = 3$, $x = 1$, $y = 1$
Mapa:
```
0 1 1
0 -1 -1
0 3 0
```

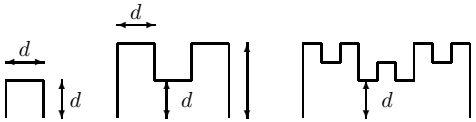
Výstup:
Najlepšia cesta:
$(1, 1)$, $(1, 2)$, $(1, 3)$, $(2, 3)$, $(3, 3)$

z1532. Zeofínine veže

Po mnohých nástrahách sa Zeofíne podarilo takmer nemožné – vrátila sa na rodné Galapágy. Predtým však prešla hodný kus sveta. A všetky užovky, volavky, ba aj korytnačie družky sa začali Zeofíny vypytovať, ako v tom veľkom svete bolo. Zaujímali sa najmä o Európu. Tu Zeofínu obzvlášť zaujali hrady a zámky. Chcela im nejaké nakresliť. Zistila však, že ak chce, aby pochopili, ako hrady vyzerajú, musí ich najskôr oboznámiť s niečím jednoduchším – napríklad s vežami.

Veža stupňa 1, resp. 2 veľkosti d je na obr. 1, resp. obr. 2. Vežu stupňa väčšieho ako 2 dostaneme tak, že nakreslíme ľavú stenu dĺžky d, vrch veže a pravú stenu dĺžky d. Vrch veže sa skladá z troch do seba zasadených vežičiek o 1 nižšieho stupňa veľkosti $d/2$, pričom stredná veža je naopak (obr. 3).

Úloha: Napíšte program, ktorý pre zadané s a d vypíše postupnosť príkazov pre Zeofínu tak, aby nakreslila vežu stupňa s a veľkosti d. V úlohe 1512 nájdete pokyny, na ktoré vie korytnačka reagovať.



Obr. 1 ($s = 1$) Obr. 2 ($s = 2$) Obr. 3 ($s = 3$)

z1533. Znak krásy

Kde bolo, tam bolo, za siedmimi horami a siedmimi dolami žil raz jeden kráľ a ten kráľ mal jednu-jedinkú dcéru. A tá bola taká pekná, aká bola hlúpa. A veru, väčšiu špatu nepoznalo ani Zrkadielko zo susedného Snehulienkinho kráľovstva. Ani hlúpy Jano by si ju nevzal za ženu, hoc aj s pol kráľovstvom. Ale všetci si ju vážili pre jej múdrosť a zdalo sa, že princeznej krása nechýba. Avšak len do dňa, keď sa jej kráľovský otec rozhodol spraviť z nej svoju nástupkyňu, keď už mu Boh nepožehnal syna a zať bol v nedohľadne. Princezná bola síce bystrá a múdra, ale ako môže svojím vzhľadom reprezentovať vlastnú krajinu v zahraničí? A tak princezná putovala od jedného plastického chirurga k druhému, z jedného salónu krásy do druhého, ale výsledok nebol omnoho lepší. Jediná nádej bola v starej dobrej čarodejnici Kirke. Kirka usadila princeznú do zaprášeného kresla a čosi si mumlala popod nos, zatiaľ čo v kotlíku bublala tajomná tekutina. A čuduj sa svete, kúzlo sa podarilo! Z voňavých pár vystupujúcich z kotlíka sa vyformoval akýsi obrazec a ľahko ako pavučinka pristál na nových Kirkiných záclonách. Bol to univerzálny znak krásy. Skôr než odletí, princezná si ho musí odkresliť na papier. Nesmie však zdvihnúť pero z papiera skôr, ako bude celý obrazec nakreslený. Princezná to samozrejme zvládne, ale skúste presvedčiť Kirku, že v tom nie sú žiadne čary, a možno vám aj uverí.

Úloha: Vstupom vášho programu bude popis znaku. Znak krásy sa skladá z niekoľkých uzlov, pričom medzi niektorými z nich vedú čiary. Prvý riadok vstupu obsahuje čísla n a m, kde n je počet uzlov a m je počet čiar. Uzly sú očíslované číslami $1, 2, \ldots, n$. Potom nasleduje m dvojíc čísel uzlov, každá dvojica obsahuje koncové body jednej čiary. Váš program má vypísať postupnosť čísel uzlov udávajúcu, ako sa bude daný obrazec kresliť. Každá čiara musí byť vykreslená práve raz.

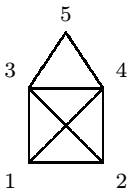
Príklad:

Vstup:

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

Výstup:

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



z1534. Zúfalý Santo

Santo sa otočil a uvidel, že sa za ním objavil stôl plný jedla. Podišiel k nemu, načiahol sa a CŔŔŔŔŔŔŔŔŔŔŔŔN. Santova ruka vyletela a zmietla budík zo stola. Keď sa konečne vyhrabal spod perín, zbadal na zemi rozkotúľané kolieska. Vzdychol, ale keďže Santo je šikovný chlapík, nakoniec sa mu podarilo budík opraviť. No keď ho natiahol, zistil, že sa ručičky točia opačným smerom. Pokúšal sa ho znovu opraviť, ale úplne sa poplietol, a už nevie, ktorým smerom sa má ktoré koliesko točiť. Teraz skúša otáčať rôznymi kolieskami a pozerá, ktoré sa kam točí.

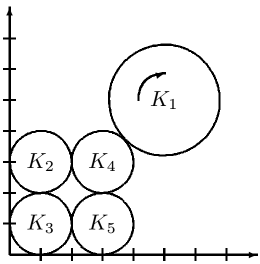
ÚLOHA: Je daný počet koliesok n, súradnice stredov a polomery všetkých koliesok a smer otáčania prvého kolieska zo vstupu. Žiadne dve kolieska sa v rovine nepretínajú, niektoré dvojice sa však môžu dotýkať. Ak sa dve kolieska dotýkajú, musia sa otáčať navzájom rôznymi smermi. Majme dve dotýkajúce sa kolieska K_1 a K_2. Nech koliesko K_1 má polomer r_1 a koliesko K_2 polomer r_2. Ich rýchlosti otáčania (fyzik by povedal uhlové rýchlosti) ω_1 resp. ω_2 musia byť v opačnom pomere ako polomery, t.j. $\omega_1/\omega_2 = r_2/r_1$. Rýchlosť otáčania prvého kolieska je 1 otáčka za minútu. Ak nie je možné tieto podmienky splniť, budík sa zasekne. Naopak, môže sa stať, že nejaká skupinka koliesok nemá žiadne spojenie s prvým kolieskom, a preto stojí.

Zistite, ktorým smerom a akou rýchlosťou sa otáča každé z koliesok. Ak sa budík zasekne, vypíšte o tom správu `Budik sa zasekol`. Kolieska, ktoré nie sú v spojení s prvým kolieskom majú nulovú rýchlosť otáčania. Ak takéto kolieska existujú, vypíšte o tom správu `Budik ma nepohanane kolieska`.

Polomery a stredy sú zadávané ako reálne čísla. Počítajte s tým, že presnosť vstupov aj výpočtov bude ovplyvnená zaokrúhľovacími chybami (pozor na porovnanie).

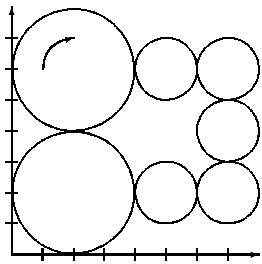
PRÍKLAD:

1. budik



	Rýchlosť	Smer
K_1	1	záporný
K_2	1.828	záporný
K_3	1.828	kladný
K_4	1.828	kladný
K_5	1.828	záporný

2. budik



`Budik sa zasekol`

z1535. Zakrývanie koberca

Na Kiribati vládne zhon. Čoskoro do ich krajiny príde na návštevu náčelník susedného súostrovia a Kiribatčania chystajú slávnostné privítanie. Voľakde zohnali dokonca aj dlhočizný červený koberec a rozhodli sa rozprestrieť ho od letiska až do stredu ich hlavnej osady. Ako ho tak rozprestierajú, zrazu zistili, že na mnohých miestach je koberec prehryzený od molí!

Aby v hanbe nezostali, musia s tým niečo spraviť. V mnohých kiribatských domácnostiach používajú malé červené koberčeky, ktoré sú síce rovnako široké ako dlhočizný

koberec, ale dlhé sú iba jeden meter. Niekoho napadlo, že by sa týmito koberčekmi dali zakryť úseky koberca, ktoré sú prehryzené od molí. Chceli by však použiť čo najmenej koberčekov, aby v domácnostiach zbytočne nechýbali.

Úloha: Daná je dĺžka koberca d, počet poškodených úsekov koberca n a pre každý poškodený úsek sú dané vzdialenosti jeho začiatku a konca od letiska (v metroch). Vypíšte najmenší počet koberčekov, aký stačí na zakrytie, ak každý poškodený úsek musí byť celý zakrytý koberčekmi, koberčeky nemožno strihať (po odchode návštevy ich treba vrátiť pôvodným majiteľom), ale môžu sa navzájom prekrývať. Ďalej vypíšte, ako treba jednotlivé koberčeky poukladať, aby sa zakryli všetky poškodené úseky.

Dôležitou súčasťou riešenia je dôkaz, že váš algoritmus je správny, t.j., že vždy pozakrýva diery pomocou najmenšieho možného počtu koberčekov.

Príklad:

Vstup:	Výstup:
$d = 10$, $n = 3$	Stačia 3 koberčeky.
Poškodené úseky:	Rozmiestnenie koberčekov:
1.2–1.3, 5.4–6.5, 1.95–2.1	1.15–2.15, 5.4–6.4, 5.5–6.5

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.

111. Stačí postupne deliť 2 a nepárnymi číslami menšími ako $\sqrt{n}$. Ak nájdeme deliteľa p_i, je to istotne prvočíselný deliteľ n. Delíme ním n, koľkokrát sa dá, čím dostaneme príslušný exponent e_i. Ďalej hľadáme ďalších (väčších) deliteľov čísla $n' = n/p_i^{e_i}$.

112. Riešenie, v ktorom sa prepisujú nenulové prvky do druhého, pomocného poľa, čím sa aj zhusťujú, považujeme za nešikovné. V riešení s jedným poľom sa postupne prezerajú prvky od najnižšieho indexu. Ak $A[i] \neq 0$, posunie sa na miesto $A[i-z]$, kde z je doteraz objavený počet núl. Keď prídeme na koniec poľa, vieme, že posledný nenulový prvok je $A[n-z]$. Vypíšeme teda prvých $n-z$ prvkov upraveného poľa.

113. Keďže $\left|\hat{\phi}^k/\sqrt{5}\right|$ je pre každé k menšie ako $1/2$, stačí vypočítať $\phi^k/\sqrt{5}$ a zaokrúhliť. Tento spôsob je však kvôli obmedzenej presnosti reálnych čísel pre väčšie k nevhodný. Fibonacciho čísla sa však dajú spočítať ešte inak – použitím matíc. Ako sa ľahko ukáže indukciou, platí

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^k = \begin{pmatrix} F_{k+1} & F_k \\ F_k & F_{k-1} \end{pmatrix}.$$

Použitím vzťahu $M^k = (M^{k \operatorname{div} 2})^2 \cdot M^{k \bmod 2}$ vieme k-tu mocninu vyššie uvedenej matice vypočítať v čase $O(\log k)$.

114. Pozor, ak generátor dáva rovnomerne rozložené náhodné čísla, musíme generovať všetky hody kockami a nie len náhodné čísla z intervalu $\langle 6, 36\rangle$, lebo napríklad súčet 7 môžeme dostať 6 spôsobmi, teda zákonite je súčet 7 pravdepodobnejší než súčet 36.

115. Prvky generujeme vo vzostupnom poradí podľa ich hodnoty. Položíme $q[0] = 1$, ďalšie $q[i]$ musia spĺňať podmienku ii) zadania. Teda $q[i]$ je najmenšie číslo väčšie ako $q[i-1]$, ktoré má tvar $2q[j] + 1$, alebo $3q[j] + 1$ pre nejaké vhodné $j < i$. Zavedieme si pomocné premenné j_2 a j_3; j_2 označuje pozíciu najmenšieho prvku $q[j_2]$ takého, že $2q[j_2]+1$ sme ešte nepridali do poľa, podobne $q[j_3]$ je najmenší prvok taký, že $3q[j_3]+1$ sme ešte nepridali do poľa. Ako ďalší prvok vždy pridáme menšieho z oboch „kandidátov“. Riešenie v [17. kapitole v 10, v 11, v 19.2 v 18].

121. Priemerný počet vylomených pokladníc je 137/60. Vo všeobecnosti, keď máme n pokladníc, je priemerný počet vylomených pokladníc rovný harmonickému číslu $H_n = \sum_{k=1}^{n} 1/k$.

Všimnite si, že rozdelenie n kľúčov do n pokladníc vieme popísať permutáciou n prvkov. Počet pokladníc, ktoré bude treba vylomiť, je rovný počtu cyklov v tejto permutácii. Podrobnejšie napríklad [3.2.1 v 22, 1.2.10 a 1.3.3 v 23, 3.5 v 30].

122. Celú časť podielu vypočítame ľahko, v ďalšom budeme uvažovať len desatinnú časť. Uvedieme viacero možných riešení.

a) Nech $z(x) = m \cdot 10^x \bmod n$ je zvyšok, ktorý dostaneme po vypočítaní x-tého desatinného miesta zlomku m/n. Keď vypočítame j-ty zvyšok, porovnáme ho postupne so všetkými predchádzajúcimi zvyškami. Ak $z(j) = z(i)$ pre $0 \leq i < j$, dĺžka predperiódy je i a periódy $j - i$.

b) Toto riešenie vieme zrýchliť, ak si v poli pre každý zvyšok zapamätáme číslo kroku delenia, kedy sa prvý raz vyskytol.

c) Metóda dvoch bežcov. Predstavme si, že si zoberieme dva papiere a na každom z nich pôjdeme počítať podiel zo zadania. Striedavo vypočítame jednu cifru výsledku na prvom papieri a dve cifry výsledku na druhom. Prestaneme, keď sa po niektorom kroku aktuálne zvyšky[23] v oboch výpočtoch rovnajú. Toto vždy nastane – skôr či neskôr sa oba výpočty dostanú do periodickej časti, a tam rýchlejší výpočet „o kolo predbehne“ ten pomalší. (Takýmto postupom sme našli nejakú predperiódu a periódu, nie nutne najkratšiu.)

d) Bez ujmy na všeobecnosti môžeme predpokladať, že čitateľ zlomku m a jeho menovateľ n sú nesúdeliteľné. Zapíšme n v tvare $n = u \cdot v$, kde $u = 2^\alpha 5^\beta$ a $\mathrm{nsd}(10, v) = 1$. Tvrdíme, že dĺžka predperiódy je $d = \max\{\alpha, \beta\}$. Dôkaz. Všimnime si, že pre $x \geq d$ je číslo $m \cdot 10^x$ násobkom u. A keďže aj n je násobkom u, musí byť aj $z(x) = m \cdot 10^x \bmod n$ násobkom u. Naopak, pre $x < d$ číslo $m \cdot 10^x$, a teda ani číslo $z(x)$ násobkom u nie je. Keď sa teda pozrieme na (nekonečnú) postupnosť zvyškov, ktoré dostávame pri delení, vidíme, že od určitého miesta (d) sú všetky deliteľné u; zvyšky pred týmto miestom sa preto neopakujú – predperióda má určite dĺžku *aspoň* d. Ukážeme teraz, že zvyšok $z(d)$ sa nám v postupnosti zvyškov zopakuje, čím bude dôkaz hotový. Vieme už, že $z(d)$ je nejaký násobok u, označme teda $z(d) = u \cdot c$. Platí $z(d+k) = (uc \cdot 10^k) \bmod n = (uc \cdot 10^k) \bmod uv = u(c \cdot 10^k \bmod v)$. Keďže 10 a v sú nesúdeliteľné, z Eulerovej vety $10^{\varphi(v)} \equiv 1 \pmod{v}$, a teda $z(d) = z(d+\varphi(v))$. Dokázali sme teda, že dĺžka predperiódy je práve d. Z vyššie uvedeného dôkazu sa dá navyše odvodiť aj to, že dĺžka najkratšej periódy delí $\varphi(v)$.

123. Ukážeme si niekoľko rôzne dobrých riešení.

a) Pole $|k|$-krát posunieme o jeden prvok, na čo treba $O(|k| \cdot n)$ posunov. Prípadne ak $|k| > n/2$, posúvame $(n - |k|)$-krát opačným smerom.

b) Využijeme pomocné pole. Tomuto riešeniu stačí $2n$ presunov prvku, ale potrebuje veľa pomocnej pamäte.

c) Zrkadlovo otočíme celé pole (to vieme ľahko aj bez pomocného poľa), potom stačí otočiť jednotlivé časti. Treba rádovo n výmen prvkov.

$$a_0a_1\ldots a_{n-k-1}b_{n-k}b_{n-k+1}\ldots b_{n-1} \longrightarrow b_{n-1}\ldots b_{n-k+1}b_{n-k}a_{n-k-1}\ldots a_1a_0 \longrightarrow$$
$$b_{n-k}b_{n-k+1}\ldots b_{n-1}a_0a_1\ldots a_{n-k-1}$$

d) [s. 73–75 v 11] Nech má počiatočná postupnosť $p = p_0$ tvar AB, chceme dostať $p' = BA$ (posun o $|B|$ vpravo). Nech $|A| > |B|$, pre $|A| < |B|$ by sa postupovalo analogicky. Označme si $A = A_0A_1$ tak, že $|A_0| = |B|$. Teraz môžeme v A_0A_1B vymeniť navzájom A_0 s B a dostaneme $p_1 = BA_1A_0$. Aby sme dostali p' musíme vymeniť A_1 s A_0, čo je riešenie pôvodnej úlohy, ale menšieho rozsahu, lebo $|A| < |p|$. Všimnite si, že pri každej vzájomnej výmene dvoch prvkov príde vždy aspoň jeden na svoje miesto, teda potrebujeme najviac n výmen.

e) Kam sa posunie a_0? Na a_k. A kam sa posunie pôvodné a_k? Na a_{2k}, atď. Kedy sa vrátime k prvku, kde sme už boli? Keďže $ik \bmod n = 0$, pre najmenšie kladné i, platí $ik = \mathrm{nsn}(n,k)$. Navyše $\mathrm{nsn}(n,k) = nk/\mathrm{nsd}(n,k)$, teda cyklus bude mať dĺžku $n/\mathrm{nsd}(n,k)$ a počet cyklov bude $\mathrm{nsd}(n,k)$.

124. Skúste vypísať štvorec po riadkoch, bez použitia pomocného poľa. „Šírka štvorca“, teda počet čísel v prvom riadku, bude $\lfloor \frac{n+5}{4} \rfloor$. „Výška štvorca“, teda počet riadkov, bude $\lfloor \frac{n+7}{4} \rfloor$.

125. Pri riešení podúlohy a) určite, koľko je permutácií pred permutáciou začínajúcou $a_0, a_1, \ldots, a_i$, ak viete, koľko ich je pred takou, čo začína $a_0, a_1, \ldots, a_{i-1}$. Podobne funguje

[23] Pozor, porovnávať treba zvyšky, nie posledné vypočítané cifry!

aj riešenie polúlohy b). Treba určiť, akou cifrou začína permutácia s kódom k. Kód k treba zmenšiť o počet permutácií začínajúcich menšími ciframi a uvedomiť si, že daná cifra sa už vo zvyšku permutácie nemôže vyskytovať.

211. Backtracking. Medzi jednotlivé cifry vkladáme znamienka $+$, $-$, alebo nič. Všetkých možností je zrejme 3^8 (prečo?). Pre každý výraz spočítame jeho hodnotu. Uvedomte si, že vo výraze nemôže byť číslo väčšie než 234 prípadne 123, takže netreba skúšať úplne všetky možnosti. Úloha má 11 riešení.

212. Pozor! Rekurzívny popis dobre uzátvorkovaných výrazov v zadaní úlohy zvádza k priamočiaremu prepisu do rekurzívneho programu. Takýto program vypíše niektoré dobré výrazy viackrát (vďaka pravidlu 3 v zadaní).

Najjednoduchšie, ale najmenej šikovné, je vygenerovať všetky $2n$ ciferné binárne čísla, 0 a 1 kódujú) a (, a vybrať z nich len tie, ktoré reprezentujú dobré výrazy.

Lepšie riešenie: Snažíme sa vytvárať dobre uzátvorkované výrazy postupným pridávaním zátvoriek zľava doprava. Zátvorky budeme vypisovať tak, aby sme každý neúplný výraz vedeli vždy doplniť na dobre uzátvorkovaný. Označme si l počet ľavých zátvoriek, ktoré ešte môžeme použiť, a p počet pravých, ktoré musíme vypísať, aby bol doteraz vypísaný výraz dobre uzátvorkovaný. Teraz stačí rozobrať, akú zátvorku môžeme pridať na koniec v prípadoch, keď $p > 0 \wedge l > 0$; $p = 0 \wedge l > 0$; $p > 0 \wedge l = 0$. Skúste riešiť bez použitia rekurzie. Počet rôznych dobre uzátvorkovaných výrazov zostavených z n párov zátvoriek je Catalanove číslo[24] $\frac{1}{n+1}\binom{2n}{n}$.

213. Niekoľko možných riešení:

a) Skúšame všetky možné začiatky a konce podpostupnosti a pre každú dvojicu v $O(n)$ spočítame súčet. Časová zložitosť $O(n^3)$.

b) Všimneme si, že ak vieme súčet úseku $x_i, \ldots, x_{j-1}$, na súčet $x_i, \ldots, x_j$ stačí pripočítať x_j, nemusíme spočítavať od začiatku. Takto vieme predchádzajúce riešenie zlepšiť na $O(n^2)$

c) Majme nájdenú podpostupnosť s najväčším súčtom, ktorá končí na i-tej pozícií. Presvedčte sa, že podpostupnosť s najväčším súčtom, ktorá končí na $(i+1)$-vej pozícií, buď dostaneme predĺžením predchádzajúcej postupnosti, alebo obsahuje len prvok na $(i+1)$-vej pozícii. Z týchto dvoch možností si zakaždým vyberieme tú lepšiu.

Riešenia sú prevzaté z kapitoly 7 v [3].

214. Netreba tisícprvkové pole. Zistite, koľko rôznych súčtov môžeme pre trojciferné čísla dostať. Riešenie by sa malo dať ľahko zovšeobecniť aj na viac než trojciferné čísla. Pokúste sa dokázať, že postup pre ľubovoľnú počiatočnú hodnotu vždy skončí číslom 1 alebo 4. Viac v [21].

215. Pri vytváraní tabuľky početností je šikovné využiť priamo kódy znakov zo vstupu – početnosť znaku bude uložená v poli na indexe rovnom kódu daného znaku. Percentuálne zastúpenie je dobré vypočítať až pri výpise a nevyskytujúce sa znaky nevypisovať.

221. Treba si uvedomiť, že stačí sledovať len vzdialenosť častice od vnútornej steny reaktora, označme ju y. Na začiatku častica vletí do steny pod uhlom α_1, bude teda $y_1 = 25 \sin \alpha_1$. Ďalej platí, $y_{i+1} = y_i + 25 \sin(\alpha_{i+1} + \gamma_i)$, kde $\gamma_i = \gamma_{i-1} + \alpha_i$, $\gamma_0 = 0$ a $1 \leq i \leq 20$.

Ak pre niektoré i dostaneme $y_i < 0$, častica sa vrátila späť do reaktora, ak dostaneme $y_i > H$ (kde H je hrúbka steny v centimetroch), častica stenou preletí. Ak aj po vypočítaní y_{20} nenastal ani jeden z týchto prípadov, častica sa zastavila v stene.

222. Napr. v [44] sa dočítame, že každé racionálne číslo sa dá zapísať ako *reťazový zlomok*, t.j. v tvare $q_1 + 1/(q_2 + 1/(q_3 \cdots 1/q_n))$; skrátene zapisujeme $[q_1, q_2, \ldots, q_n]$, kde q_k je neúplný podiel v k-tom kroku výpočtu najmenšieho spoločného deliteľa čísel p a q. Zlomky

[24] podľa belgického matematika Eugèna Charlesa Catalana (1814–1894)

P_k/Q_k také, že $P_k/Q_k = [q_1, q_2, \ldots, q_k]$ voláme *zblížené zlomky*. Zblížené zlomky sa dajú ľahko vypočítať podľa rekurentného vzťahu $P_k = q_k P_{k-1} + P_{k-2}$, $Q_k = q_k Q_{k-1} + Q_{k-2}$, pričom $P_{-1} = 0$, $P_0 = 1$, $Q_{-1} = 1$, $Q_0 = 0$. Platí, že medzi zlomkami, ktorých menovatele sú menšie alebo rovné Q_k aproximuje zlomok P_k/Q_k číslo p/q najlepšie. Iné riešenie založené na Sternovom-Brocotovom[25] strome je v [14], jeho vylepšená verzia sa dá nájsť vo vzorovom riešení jednej úlohy IPSC 2005.

223. Označme b_i počet tých prvkov hľadanej permutácie, ktoré sú naľavo od i a sú od neho väčšie. (Hodnoty b_i teda tvoria zadanú tabuľku inverzií.) Predstavte si, že vezmeme čistý pásik papiera s n políčkami, do ktorých chceme vpísať čísla od 1 do n tak, aby tvorili hľadanú permutáciu. Čísla budeme vpisovať v poradí od 1 do n. Pritom i-te číslo v poradí musíme vpísať práve do $(b_i + 1)$. voľného políčka zľava, lebo naľavo od neho si musíme nechať miesto na práve b_i spomedzi nasledujúcich čísel. Výsledná permutácia je teda jednoznačne určená, a navyše vyššie popísaný postup vieme triviálne odsimulovať v čase $O(n^2)$. Využitím intervalového stromu alebo vyvážených stromov sa dá vytvoriť algoritmus s časovou zložitosťou $O(n \log n)$: Napríklad si môžeme vo vrcholoch intervalového stromu pamätať, koľko voľných políčok sa ešte v danom intervale nachádza. Takto vieme každému číslu nájsť správne políčko v čase $O(\log n)$, a v rovnakom čase vieme aj upraviť pamätané hodnoty v strome.) Podrobnejšie v [5.1.1.6 v 25].

224. Backtracking. Binárny reťazec s hľadanými vlastnosťami sa nazýva de Bruijnovým[26] cyklom dĺžky 2^n. De Bruijnove cykly súvisia s eulerovskými ťahmi (úloha 643) vo vhodne zostrojenom orientovanom grafe (vrcholy budú binárne reťazce dĺžky 2^{n-1}, hrana pôjde z u do v, ak $u = s_1 s_2 \ldots s_{n-1}$ a $v = s_2 \ldots s_{n-1} s_n$, [10.1 v 8]). Nájsť jeden de Bruijnov cyklus sa teda dá efektívne. Ak ich ale máme vymenovať všetky, neexistuje polynomiálne riešenie, keďže počet rôznych de Bruijnových cyklov (v závislosti od n) rastie rýchlejšie ako exponenciálne (je ich $2^{2^{n-1}-n}$).

225. Využite výsledok úlohy 215. V slovenčine sa najčastejšie vyskytujú písmená A, O, E, I, N a S (teda najčastejšie sa vyskytujúce písmeno v texte bude pravdepodobne A). Urobte interaktívny program a postupným vylepšovaním dekódujte text.

Existujú aj lepšie riešenia, napríklad s využitím štatistiky výskytov *tetragramov* (štvoríc po sebe idúcich písmen) v slovenskom texte sa dá napísať riešenie, ktoré tento text dokáže dekódovať úplne automaticky – postupne „vylepšuje“ šifrovací kľúč tak, aby mu dešifrovaný text „znel čo najviac slovensky“. Inou možnosťou je vybaviť program slovníkom a snažiť sa nájsť takú permutáciu písmen, pri ktorej sa bude v našom slovníku nachádzať veľa dešifrovaných slov.

231. Najlepšia je možnosť ii). Očakávané výsledky: pri prvej možnosti ročne vyťažíme v priemere 25 000, pri druhej 35 000 a pri tretej 27 000 ton ropy. Pri druhej možnosti teda môžeme očakávať, že 100 000 ton ropy sa nám podarí vyťažiť za približne tri roky.

232. 13. deň v mesiaci padol na pondelok 173, utorok 169, stredu 173, štvrtok 171, piatok 171, sobotu 172 a nedeľu 171-krát.

Nech $d.m.r$ je zadaný dátum, pričom platí, že rok je z rozmedzia $1582 \le r \le 4902$. Označme $y = r \bmod 100$ a $c = r \operatorname{div} 100$. Ďalej m' je upravené číslo mesiaca: 1 – marec, 2 – apríl, ..., 11 – január, 12 – február. Potom platí:

$$\text{deň v týždni} = \left(\lfloor 2.6m' - 0.2 \rfloor + d + y + \left\lfloor \frac{y}{4} \right\rfloor + \left\lfloor \frac{c}{4} \right\rfloor - 2c \right) \bmod 7$$

Výsledok 0 je nedeľa, 1 je pondelok, atď., až 6 je sobota.

---

[25] podľa francúzskeho hodinára Achilla Brocota (1817–1878) a nemeckého matematika Moritza Abrahama Sterna (1807–1894)

[26] Nicolaas Govert de Bruijn (1918–), holandský matematik

Pri odvádzaní horeuvedeného vzorca si treba uvedomiť, ktoré roky sú priestupné[27]. Predpokladajte, že viete, ktorý deň v týždni bol 1. marec niektorého roku (napríklad tohto) a určite, ktorý deň v týždni je 1. marec roku r. Potom treba ešte pripočítať počet dní od 2. marca po daný dátum $d.m$, vrátane.

233. Riešenia, v ktorých sa kontrolujú všetky postupnosti, alebo riešenia využívajúce dvojkovú sústavu, nepovažujeme za dobré. Lepšia úvaha: Ako vyzerá *dobrá* postupnosť dĺžky n (kde $n > 2$)? Končí sa buď jednotkou, alebo nulou. Ak sa končí jednotkou, pred ňou môže byť ľubovoľná *dobrá* postupnosť dĺžky $n-1$. Ak sa končí nulou, pred touto nulou musí byť jednotka. No a pred touto jednotkou môže byť ľubovoľná *dobrá* postupnosť dĺžky $n-2$. Ak teda označíme $P(n)$ počet *dobrých* postupností dĺžky n, dostávame vzťah: $P(n) = P(n-1)+P(n-2)$. Začiatočné hodnoty $P(1) = 2$ a $P(2) = 3$ ľahko spočítame ručne. Teraz už len postupne nájdeným vzorcom spočítame hodnoty $P(3)$ až $P(n)$. Všimnite si, že $P(n)$ je rovné Fibonacciho číslu F_{n+2}.

234. Tento algoritmus sa volá pivotizácia podľa nuly. Pivotizácia (podľa ľubovoľnej hodnoty) je hlavnou časťou triedenia QuickSort – rozdelí nám pole na dve časti, ktoré môžeme triediť každú zvlášť. Dobrá realizácia si vystačí s pár premennými. Pôjdeme naraz z oboch koncov poľa. Vždy, keď nájdeme „naľavo“ nejaké kladné číslo a „napravo“ nejaké záporné, vymeníme ich. Rozmyslite si, ako ošetriť nulu.

235. Počiatok súradnicovej sústavy dajte do bodu $[\frac{n}{2}, \frac{n}{2}]$, aby sa mriežka ľahko otáčala.

311. Uvedomte si, kedy už pre niektorý prvok istotne viete, že nebude vo všetkých riadkoch. Veľmi pomôže to, že riadky sú usporiadané. Možných riešení je veľa, existujú dokonca riešenia lineárne od počtu čísel v tabuľke. Uvedieme dve takéto riešenia.

a) Pre každý riadok si udržiavame index prvku, po ktorý sme už v ňom pri kontrole prišli. Tieto prvky volajme *práve skúmané*. Na začiatku budú práve skúmané prvky z prvého stĺpca tabuľky. Hodnotu najväčšieho spomedzi práve skúmaných prvkov nazvime *kandidát*. Práve skúmané prvky budeme mať rozdelené na dve kôpky – tie, ktoré sa rovnajú kandidátovi a tie, ktoré sú od neho menšie. Ak sa všetky rovnajú kandidátovi, máme riešenie. Ak nie, každý práve skúmaný prvok, ktorý sa mu nerovná, nahradíme jeho „pravým susedom“. Akonáhle sa nám niektorý riadok minie, môžeme skončiť. Takto každý prvok tabuľky spracujeme najviac raz.

b) Tabuľku budeme spracúvať po riadkoch, pričom si budeme udržiavať vzostupne usporiadaný zoznam kandidátov na spoločný prvok. Na začiatku dáme do zoznamu všetky prvky prvého riadku. Pri spracovaní každého ďalšieho riadku vyrobíme nový zoznam kandidátov, do ktorého dáme len tých, ktorí sa vyskytujú aj v práve spracúvanom riadku. (Keďže riadok aj zoznam kandidátov sú usporiadané, vieme to spraviť v lineárnom čase.) Výhodou tohto riešenia sú jeho malé pamäťové nároky.

312. Skupina sa dá zapísať práve vtedy, ak (voľné miesto v poli) + (dĺžka skupiny s rovnakou hlavičkou) − (dĺžka danej skupiny) ≥ 0.

313. Na odsimulovanie jedného kroku stačí vygenerovať tri náhodné čísla: súradnice voliča a následne jeden z ôsmich smerov (teda vlastne jedného z jeho ôsmich susedov). Priebeh „diskusie“ bude vyzerať nasledovne: začnú sa vytvárať čoraz väčšie skupiny rovnako hlasujúcich, pokým neostanú len dve, a po čase napokon zostane len jedna. Skúste sa zamyslieť, ako by sa zmenil výsledok úlohy, ak by sme neuvažovali susedov „na druhom kraji“, teda ak by voliči sediaci na okrajoch štvorca mali menej ako osem susedov.

---

[27] Priestupné roky boli zavedené v Juliánskom kalendári. Pri jeho tvorbe sa odhadlo, že rok má 365 a pol dňa, čo je oproti astronomickému roku o niečo viac – ten má 365.2422 dňa. Preto pápež Gregor XIII prijal roku 1582 reformu kalendára. Vypustili desať dní, 11. marec sa stal 21. marcom. Reformu ovplyvnil významný posun kalendára vzhľadom na začiatky ročných období. V Anglicku bola reforma prijatá r. 1752, 3. september sa stal 14. septembrom a v Rusku prijali reformu r. 1918, 1. február sa stal 14. februárom.

314. Úlohou je previesť číslo do vyváženej trojkovej sústavy (balanced ternary) – sústava so základom tri, ktorá má namiesto 0, 1, 2 cifry -1, 0 a $+1$. Cifru -1 značíme $\overline{1}$; napr. $(10\overline{1})_{bal.\,3} = 1 \cdot 3^2 + 0 \cdot 3^1 + (-1) \cdot 3^0 = 8$. Možné prístupy:

a) Nech $c(n) = (3^n - 1)/2$, potom pomocou n cifier vieme zapísať číslo $-c(n) \le x \le c(n)$. To znamená, že ak $-c(n-1) \le x \le c(n-1)$, stačí $(n-1)$ cifier, teda prvá cifra je nulová; ak $x > c(n-1)$, prvá cifra je 1 a ostáva zapísať číslo $x - c(n)$; ak $x < -c(n-1)$, prvá cifra je $\overline{1}$ a ostáva zapísať číslo $x + c(n)$.

b) Dá sa na to ísť aj od opačného konca. Podľa zvyšku, aký dáva hmotnosť predmetu po delení tromi, vieme povedať, či a ako treba použiť 1kg závažie. Po jeho správnom umiestnení bude rozdiel hmotností na miskách deliteľný tromi. Aj všetky závažia, ktoré nám ostali, majú hmotnosť deliteľnú tromi, môžeme teda všetko vydeliť tromi... a máme pôvodnú úlohu, len závaží je o jedno menej. V reči číselných sústav, cifry určujeme od najnižšieho rádu podľa aktuálneho zvyšku, aký dáva n po delení tromi.

c) Trikový postup z [24] je previesť číslo do trojkovej sústavy, pripočítať nekonečné (stačí dostatočne veľké) číslo tvaru $(\ldots 111)_3$ a zmenšiť každú cifru výsledku o 1. Napr. $208 = (21201)_3$; keď pričítame $(\ldots 111)_3$, dostaneme $(\ldots 111210012)_3$ a po znížení všetkých cifier o 1 dostávame $208 = (\ldots 00010\overline{11}01)_{bal.\,3}$.

315. Po prečítaní celého vstupu skonštruujte triedy ekvivalencie danej relácie R. (Ľubovoľnú reláciu si môžeme znázorniť grafom, kde vrcholy budú prvky a hrana vedie z x do y, ak (x, y) je v relácii R. Symetrickú reláciu si môžeme znázorniť neorientovaným grafom, ktorý dostaneme z vyššie definovaného orientovaného tak, že zabudneme na orientáciu hrán. Triedy ekvivalencie zodpovedajú komponentom súvislosti v takomto grafe.)

321. Existujú efektívne spôsoby generovania Pytagorejských trojíc. Jedna možnosť: Pre každé prirodzené u, v je trojuholník so stranami $v^2 - u^2$, $2uv$ a $u^2 + v^2$ pravouhlý. Dá sa dokázať, že každý Pytagorejský trojuholník vieme dostať ako „násobok“ nejakého takto nájdeného trojuholníka. Dá sa dokonca dokázať, že ak použijeme len nesúdeliteľné u, v, z ktorých práve jedno je párne, tak dostaneme práve všetky *primitívne* Pytagorejské trojuholníky (t.j. také, kde $\mathrm{nsd}(a, b, c) = 1$), každý práve raz.

322. Stačí vedieť pripočítavať 1 v dvojkovej sústave, generovať čísla od 0 po $2^h - 1$ (h je počet hviezdičiek) a jednotlivé bity použiť namiesto hviezdičiek. Iná jednoduchá implementácia je pomocou rekurzie.

323. Rovnica úsečky s koncovými bodmi (x_1, y_1), (x_2, y_2), kde $x_1 < x_2$, je

$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1, \quad x_1 \le x \le x_2.$$

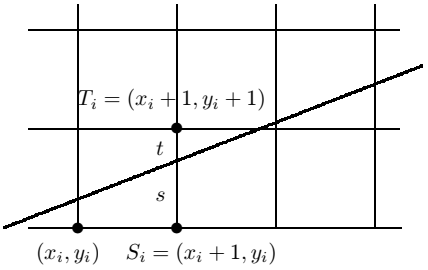
Keď do nej budeme postupne dosadzovať celočíselné hodnoty x, budeme dostávať body z tejto úsečky. Nevýhodou je, že pre „strmé“ úsečky dostaneme jednotlivé body „ďaleko“ od seba a pre zvislé dokonca delíme nulou. Toto sa dá odstrániť tak, že pre úsečky, ktoré sú „viac zvislé ako vodorovné“, použijeme duálny vzťah

$$x = \frac{x_2 - x_1}{y_2 - y_1}(y - y_1) + x_1, \quad y_1 < y_2, \quad y_1 \le y \le y_2.$$

Bresenhamov algoritmus predstavuje vylepšenú verziu tohto postupu. Vystačíme si dokonca s použitím celých čísel a jednoduchých matematických operácii.

Uvažujme úsečku ležiacu na priamke $y = kx + l$, kde $k = \Delta y/\Delta x \in \langle 0, 1\rangle$. Úsečku budeme kresliť postupne; dajme tomu, že sme práve nakreslili bod $[x_i, y_i]$. Ďalší možný pixel je $S_i = [x_i + 1, y_i]$ alebo $T_i = [x_i + 1, y_i + 1]$. Označme s a t vzdialenosť skutočného bodu úsečky od bodov S_i a T_i. Platí $s = k(x_i + 1) + l - y_i$ a $t = y_i + 1 - k(x_i + 1) - l$. O tom, ktorý z pixlov S_i a T_i si zvolíme, rozhodne rozdiel $s - t$. Ak $s - t \ge 0$, potom $s \ge t$,

teda bod T_i je bližšie; naopak, ak $s - t < 0$, zvolíme bod S_i. Výraz ešte upravíme tak, aby sme používali iba celočíselnú aritmetiku: $s - t = 2k(x_i + 1) + 2l - 2y_i - 1$ prenásobíme Δx; dostaneme $d_i = \Delta x(s - t) = 2\Delta y(x_i + 1) - 2\Delta x y_i + \Delta x(2l - 1)$, kde $2\Delta y + \Delta x(2l - 1)$ je konštanta. Preto odčítaním $d_{i+1} - d_i = 2\Delta y(x_{i+1} - x_i) - 2\Delta x(y_{i+1} - y_i)$ dostaneme $d_{i+1} = d_i + 2\Delta y - 2\Delta x(y_{i+1} - y_i)$.



Počiatočné $d_1 = 2\Delta y - \Delta x$, $x_1 = x$ a $y_1 = y$. Podľa znamienka d_{i+1} sa rozhodneme, ktorý z pixlov S_{i+1} ($y_{i+1} = y_i$) a T_{i+1} ($y_{i+1} = y_i + 1$) si vyberieme; hodnotu d_{i+1} vieme vypočítať z d_i pomocou celočíselnej aritmetiky.

Úsečky zobrazujeme vo všetkých oktantoch, každá sa dá vhodnou transformáciou previesť na náš prípad. Stačí vhodne zvoliť prírastky s_x, s_y, d_x, d_y, prípadne vzájomne vymeniť x-ovú a y-ovú súradnicu. Riešenie je z [16].

```
procedure ciara(x, y, x2, y2 : integer);
var i, sx, sy, Δx, Δy, d : integer;
    strma : boolean;
begin
  Δx := |x2 − x|; sx := sgn(x2 − x);
  Δy := |y2 − y|; sy := sgn(y2 − y);
  strma := (Δx < Δy);
  if strma then begin vymen(x, y); vymen(sx, sy); vymen(Δx, Δy); end;
  d := 2 · Δx − Δy;
  for i := 1 to Δx do begin
    if strma then putpixel(y, x) else putpixel(x, y);
    while (0 ≤ d) do begin y := y + sy; d := d − 2 · Δx; end;
    x := x + sx; d := d + 2 · Δy;
  end;
  putpixel(x2, y2);
end;
```

324. Prehľadávanie do šírky. (Nie backtracking!)

325. Pozri 115.

331. Orezávanie. Rovinu rozdelíme na 9 častí. Bodu (x, y) priradíme $s(x)$ a $s(y)$, kde

$$s(x) = \begin{cases} -1 & \text{ak } x < x_{min} \\ 0 & \text{ak } x_{min} \le x \le x_{max} \\ 1 & \text{ak } x_{max} < x \end{cases} \qquad s(y) = \begin{cases} -1 & \text{ak } y < y_{min} \\ 0 & \text{ak } y_{min} \le y \le y_{max} \\ 1 & \text{ak } y_{max} < y \end{cases}$$

Hodnoty $s(x)$ a $s(y)$ stačia na to, aby sme vedeli priamo určiť, ktoré úsečky ležia celé v zobrazovacom okne a takisto odhalili väčšinu z tých, ktoré sú celé mimo. Vo zvyšných prípadoch treba vypočítať prienik úsečky s hranicami okna. Podrobnejšie v [15, 17].

332. Priamočiara realizácia. Najjednoduchšie je odchody domorodcov simulovať po jednom.

V prípade, že máme domorodcov na ostrove veľmi veľa, môžeme sa uspokojiť s približným, ale omnoho efektívnejším riešením: náhodná premenná „počet domorodcov, ktorí

vyplávajú v daný deň z daného ostrova“ má približne *normálne rozdelenie pravdepodobnosti* so strednou hodnotou np a štandardnou odchýlkou $\sqrt{np(1-p)}$. Na odsimulovanie jedného odchodu domorodcov nám potom stačí vygenerovať (s dostatočnou presnosťou) jedno náhodné reálne číslo z intervalu $\langle 0,1)$, a vypočítať, akému počtu presúvajúcich sa domorodcov zodpovedá.

333. Označme m_n číslo, ktoré hľadáme a P množinu všetkých prvočísel. Platí, že

$$m_n = \prod_{p\in P} p^{\max\{l \mid p^l \le n\}}.$$

Teraz už stačí len vedieť násobiť veľké číslo malým.

334. Ukážeme niekoľko možných prístupov.

a) Priamočiare riešenie: vygenerujeme všetky zlomky s čitateľom a menovateľom z daného rozsahu, pre každý zistíme, či je čitateľ s menovateľom nesúdeliteľný a tie, čo zostanú, utriedime.

b) Lepšie riešenia sú založené na Sternovom-Brocotovom strome: Napíšme si vľavo zlomok $0/1$, vpravo $1/1$ a vždy medzi dva zlomky m/n a m'/n' pridajme zlomok $(m+m')/(n+n')$ (kým nie je menovateľ príliš veľký). Dá sa jednoducho implementovať pomocou rekurzie.

c) Pre tri po sebe idúce členy m'/n', m''/n'' a m'''/n''' platí

$$m''' = \left\lfloor \frac{n'+n}{n''} \right\rfloor m'' - m', \qquad n''' = \left\lfloor \frac{n'+n}{n''} \right\rfloor n'' - n'.$$

Podrobnejšie riešenia nájdete napr. v [23, 14].

335. Rozmyslite si, kam má zmysel dávať ‘(’ a ‘)’ a že napr. $-(a_i - (a_{i+1} + a_{i+2})) = -(a_i - a_{i+1}) + a_{i+2}$.

411. Pomerne zjavne má každé takéto číslo aspoň dve cifry (lebo nezačína nulou, a teda má aspoň jednu nulu) a najviac 10 cifier (lebo väčšie cifry ako 9 nemáme). Majme teda číslo $c = \overline{a_0 \dots a_{n-1}}$, kde a_i je zároveň počet cifier i v c. V prvom rade si všimnime ciferný súčet čísla c. Keďže a_i je počet cifier i, súčet všetkých a_i je celkový počet cifier, teda n. Iný spôsob, ako spočítať ciferný súčet čísla c: Máme a_0 núl, a_1 jednotiek, atď. Preto ciferný súčet čísla c je $a_0 \cdot 0 + 1 \cdot a_1 + \cdots + (n-1) \cdot a_{n-1}$. Na to, aby sme našli všetky riešenia, stačí už len postupne vyskúšať všetky možné hodnoty c. Dve podmienky, ktoré sme si odvodili, nám už dostatočne zmenšia priestor možných čísel. Prípadne môžeme pokračovať v ďalších úvahách: ak $n > 6$, súčet hodnôt a_6 až a_{n-1} je najviac jedna. Ak by bol totiž väčší, mali by sme v čísle c aspoň dve cifry s hodnotou väčšou ako 5, a teda by jeho ciferný súčet bol väčší ako 10. Pokračovaním v podobných úvahách sa celá úloha dá vyriešiť aj ručne.

412. Označme si prvky v k-prvkovej podmnožine vo vzostupnom poradí $a_1, a_2, \dots, a_k$ (pre $i < j$ je $a_i < a_j$). Na určenie poradového čísla množiny nám stačí zistiť, koľko k-tic je v lexikografickom poradí za danou množinou. Za danou množinou sú jednak množiny, ktorých všetky prvky sú väčšie ako a_1 (tých je $\binom{n-a_1}{k}$), ďalej také, ktoré obsahujú a_1 a všetky ostatné prvky väčšie ako a_2 (tých je $\binom{n-a_2}{k-1}$), atď. Spolu to je

$$V = \binom{n-a_1}{k} + \binom{n-a_2}{k-1} + \cdots + \binom{n-a_{k-1}}{2} + \binom{n-a_k}{1}$$

množín ($\binom{x}{y}$ je 0 pre $x < y$). Kód $\{a_1, a_2, \dots, a_k\}$ je potom $\binom{n}{k} - V$.

413. Existuje veľa správnych algoritmov riešiacich túto úlohu, stručne popíšeme aspoň niektoré z nich.

a) $O(n^3)$: Budeme hľadať hrany konvexného obalu (KO). Úsečka určená dvoma bodmi je hrana KOpráve vtedy, keď všetky ostatné body množiny ležia v jednej ňou určenej polrovine.

b) $O(n^2)$: Jarvisov algoritmus, tiež známy ako package wrapping (balenie balíčka); v predchádzajúcom spôsobe netreba skúmať všetky dvojice bodov. Keď máme hranu KO pq a chceme nájsť nasledujúcu, stačí skúmať úsečky s koncovým bodom q. Najskôr nájdeme bod s najmenšou y-ovou súradnicou; ak je takých viac, ten s najmenšou x-ovou; označme ho p_1. Bod p_2 vyberieme tak, aby orientovaný uhol od kladnej poloosi osi y po polpriamku p_1p_2 bol čo najmenší. Vo všeobecnosti bod p_k bude taký bod, pri ceste do ktorého sa musíme v p_{k-1} čo najmenej otočiť od smeru, ktorým sme tam prišli. Každý nasledovný vrchol KO vieme nájsť v lineárnom čase vzhľadom na počet bodov. Ak je veľa bodov vo vnútri KO, je tento algoritmus výhodný, pretože vyžaduje len $O(hn)$ operácií, kde h je počet vrcholov KO. Ďalšou výhodou je, že sa dá zovšeobecniť pre body v priestore.

c) $O(n \log n)$: Grahamov[28] algoritmus, resp. Andrewova úprava. Body utriedime podľa x-ovej súradnice (Graham ich pôvodne triedil podľa polárnych súradníc vzhľadom na nejaký bod z vnútra konvexného obalu).

Ukážeme si, ako nájdeme vrchnú časť KO (spodnú nájdeme analogicky). Začneme v najľavejšom bode p_1 a postupujeme smerom doprava. Keď pôjdeme po KO v tomto smere, budeme sa stále točiť do pravej strany. (Inými slovami, každé tri po sebe idúce body KO sú vpravoorientované, pozri nižšie.) Ako postupujeme doprava, v zásobníku si udržiavame vrchnú časť KO tých bodov, ktoré sme zatiaľ prešli. Keď prídeme k novému bodu, pozrieme sa, či spolu s vrchnými dvoma vrcholmi KO v zásobníku tvorí trojicu orientovanú správnym smerom. Kým toto neplatí, vyhadzujeme vrchný vrchol zo zásobníka. Následne vložíme do zásobníka práve spracovaný bod. Na konci máme hotovú vrchnú časť KO všetkých bodov. Pri implementácii je vhodné použiť zarážku $p_0 = [x_1, y_1 - 1]$.

Orientáciu trojice bodov vieme zistiť nasledovne:

$$\begin{array}{l} p_1 = [x_1, y_1] \\ p_2 = [x_2, y_2] \\ p_3 = [x_3, y_3] \end{array} \qquad P = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} > 0 \quad \Longleftrightarrow \text{ keď } p_1, p_2, p_3 \text{ sú vpravoorientované.}$$

Keďže každý bod najviac raz vložíme a raz vyberieme zo zásobníka, hľadanie hornej časti KO vieme spraviť v lineárnom čase; najviac času teda zaberie triedenie.

Podrobnejšie popísané algoritmy nájdete napr. v [33.3 v 6, 8.4 v 34, VIII.2 v 35, 39, 25 v 41].

414. Viacero možných prístupov:

a) Riešenie využívajúce smerníky. Každý človek v kruhu si pamätá svojho aktuálneho nasledovníka; $O(nm)$.

b) Použitím šikovnejšej dátovej štruktúry, napríklad vyváženého stromu alebo skip-listu, vieme dosiahnuť zložitosť $O(n \log n)$.

c) Prekvapujúci súvis s úlohou 223 [5.1.1.2 v 25]. Označme p_i číslo, ktoré hovorí, koľký v poradí zomrel človek s číslom i. Zjavne $p = (p_1, \ldots, p_n)$ je permutácia čísel od 1 do n. Jej tabuľku inverzií môžeme vypočítať nasledovne: $b_1 = (m-1) \bmod n$, $b_{j+1} = (b_j + m - 1) \bmod (n - j)$.

Iné riešenie je v [14], riešenie pre $m = 3$ je v [29 v 2.5.10].

415. Binárne vyhľadávanie: Nech $l = 1$ a $r = n$ – index i hľadáme v intervale $\langle l, r \rangle$. Pozrime sa na stredný prvok $A[m]$, kde $m = \lfloor (l + r)/2 \rfloor$. Ak $A[m] < x$, potom index i sa

[28] Ronald Lewis Graham (1935–), americký matematik

určite nachádza niekde v druhej polovici intervalu, t.j. v $(m, r\rangle$. Naopak, ak $x \le A[m]$, index i sa nachádza niekde v prvej polovici intervalu, t.j. v $(l, m\rangle$. Dostávame teda menší interval, v ktorom hľadáme index i rovnakým spôsobom. Keďže každým krokom sa interval zmenší na polovicu, stačí $\lfloor \log_2 n \rfloor$ takýchto krokov.

Pozri napríklad [4. kapitolu v 3, alebo 1.6 v 46].

421. Predstavme si, že cestujeme po zadanej lomenej čiare. Tá predstavuje konvexný n-uholník práve vtedy, ak sa pri ceste po nej točíme stále tým istým smerom (stále doľava, alebo stále doprava) a do okamihu, keď sa vrátime na miesto, odkiaľ sme vyšli, sa otočíme presne o 360 stupňov.

422. Vďaka symetrii stačí uvažovať uhly $\alpha \in \langle 0, \pi/2)$. (Pozor, treba generovať náhodný uhol α a nie priamo $\sin \alpha$.) Ak máme ihlu dĺžky $\ell \le d$ a padne pod uhlom α, jej výška (dĺžka pozdĺž osi y) je $\ell \sin \alpha$. Teda pravdepodobnosť toho, že ihla pretne nejakú čiaru (pre daný uhol α) je $\ell \sin \alpha / d$. Stačí teda spočítať priemer cez všetky hodnoty α:

$$p = \int_0^{\pi/2} \frac{\ell \sin \alpha}{d} \,\mathrm{d}\alpha \Big/ \frac{\pi}{2} = \frac{2}{\pi} \int_0^{\pi/2} \frac{\ell \sin \alpha}{d} \,\mathrm{d}\alpha = \frac{2}{\pi} \frac{\ell}{d} \left[-\cos \alpha \right]_0^{\pi/2} = \frac{2}{\pi} \frac{\ell}{d},$$

čo je v našom prípade $\ell = d$ rovné $2/\pi$.

423. Backtracking. Takto zafarbené šachovnice sa volajú *latinské štvorce*.

424. Urobte načítanie celého čísla so znamienkom. Potom stačí prečítať číslo pred desatinnou bodkou, za desatinnou bodkou a exponent a z nich reálne číslo spočítať. Pozri [1.11.3 v 46].

425. Pre nenulové prvky vieme vypočítať, kde majú byť po preusporiadaní. Číslo i má byť na mieste $i + (i - 1) \operatorname{div} (n/r - 1)$.

431. Pole utriedime (v čase $O(n \log n)$). Tým sa nám rovnaké prvky dostanú vedľa seba a dajú sa spočítať na jeden prechod poľom v lineárnom čase. Dá sa dokázať, že ani jednoduchšia úloha – zistiť, či dané pole obsahuje dva rovnaké prvky – sa vo všeobecnosti nedá vyriešiť efektívnejšie.

432. Nie je nám známy efektívny algoritmus, ktorý by vedel nájsť riešenie s minimálnym počtom otočení. Vzorovým riešením je teda backtracking.

Existujú však efektívne postupy, ktoré si vystačia síce nie s minimálnym, ale určite s lineárnym počtom otočení. Jedna možnosť: V i-tom kroku dáme na miesto kartičku s číslom i. Na jeden krok treba najviac dve otočenia. Rozmyslite si, ako na to.

Iné riešenie: Políčko k rozdelí kartičky na dve časti, označme ich A a B. Najprv vymeníme navzájom prvky nepatriace do časti A s tými, čo nepatria do časti B. Ďalej ako v predchádzajúcom riešení.

433. Úlohu prevedieme na úlohu teórie grafov: nech vrcholy sú políčka miestnosti a hranami sú spojené tie vrcholy, ktoré sú spojené kachličkou vo výslednom uložení. Dostávame takzvaný *bipartitný graf* a našou úlohou je zistiť, či v ňom existuje párovanie. Najjednoduchší efektívny algoritmus je založený na hľadaní *zlepšujúcich ciest*. Podrobnejšie je uvedený napr. v [8 v 38, 9 v 8, 19 v 27].

434. Čísla menšie než 2^k majú najviac k cifier, pričom prvá a posledná musia byť jednotky. Okrem `1`, `11`, `101` a `111` teda stačí generovať všetky čísla $x < 2^{\lfloor k/2 \rfloor - 1}$ a *ak sa to dá*, z každého vyrobiť tri symetrické čísla $\mathtt{1}x x^R\mathtt{1}$, $\mathtt{1}x\mathtt{0}x^R\mathtt{1}$ a $\mathtt{1}x\mathtt{1}x^R\mathtt{1}$; x^R je zrkadlový obraz x, t.j. x napísané odzadu.

435. „Odzadu“ stačí vypisovať do buffera cifry čísla $\mathit{round}(10^m r)$ a po m cifrách vypísať desatinnú bodku a zvyšné cifry. V prípade, že je dĺžka p vypísaného čísla menšia alebo

rovná n, vypíšeme ešte $n - p$ medzier, inak buffer naplníme '$*$', čo znamená, že sa číslo r v danom formáte nedá vypísať. Pozri [1.11.3 v 46].

511. Vypočítajte, koľko medzier treba pridať medzi slová v riadku, aby bol výstupný riadok zarovnaný. Pozor na dlhé slová. Vyriešte otázku viacnásobných medzier vo vstupnom texte. Text čítajte po riadkoch. Riešenie v [B.2 v 32, 18].

512. Všimnite si, že i-ty prvok v Q_i je i-ty aj v Q. Nech $Q_{i,j}$ je j-te číslo v postupnosti Q_i (číslujeme od 0). To isté číslo sa nachádza v postupnosti Q_{i-1}, akurát je pred ním ešte $v = (j \operatorname{div} i) \cdot i = j - (j \bmod i)$ prvkov, ktoré sme vyškrtli, teda $Q_{i,j} = Q_{i-1,j+v}$; pričom vieme, že $Q_{0,i} = i + 1$.

513. Na reprezentáciu rytierov použite frontu. Dá sa ju implementovať ako spájaný zoznam, prípadne ako „cyklické“ (t.j. dokola stále znova prepisované) pole.

514. Vpíšte do kruhu najväčší možný štvorec a spočítajte len plôšky, ktoré ostali. Iné riešenie: Myšlienka Bresenhamovho algoritmu (pozri úlohu 323) sa dá upraviť aj na kreslenie časti oblúku kružnice. Nám teraz stačí prejsť po obvode kružnice a pre každý riadok papiera si zaznačiť najpravejší štvorček, ktorý ešte leží v kruhu. Drobné vylepšenie: stačí počítať plochu jednej osminy kruhu.

515. Možné riešenia:

a) Oblasti vyfarbujeme postupne po jednej, vždy celé. Z každého políčka súvislej oblasti sa snažíme rozšíriť na jeho štyri susedné políčka (prehľadávanie do hĺbky/šírky).

b) Stačí si pamätať dva susedné vodorovné pásiky poľa. Každú súvislú oblasť vyfarbujeme rôznou farbou, ak zistíme, že sa nám dve súvislé oblasti spojili, farby zjednotíme. Na dosiahnutie dobrej zložitosti $O(mn \log^* n)$ sa dá pri spájaní farieb použiť algoritmus union-find.

c) Existuje aj riešenie s časovou zložitosťou $O(mn)$ a pamäťovou zložitosťou $O(n)$. Postupujeme rovnako ako v predchádzajúcom riešení. Nový riadok pridávame nasledovne: Ofarbíme súvislé úseky v ňom novými farbami, potom zostrojíme graf, ktorého vrcholy sú staré a nové farby, nájdeme komponenty súvislosti, a každý prefarbíme na jednu farbu.

521. Rozdeľte k na tisícky, stovky a desiatky s jednotkami.

522. Treba použiť dynamické programovanie. Dve možnosti:

a) Podľa [18]: Pre každý prvok a_i postupnosti určíme číslo b_i – dĺžku *najdlhšej podpostupnosti končiacej prvkom* a_i. Pri počítaní b_i treba nájsť najdlhšiu podpostupnosť naľavo od a_i, na ktorú môžeme a_i napojiť. Hľadáme teda maximálne b_j také, že $j < i$ a $a_j < a_i$; platí $b_i = 1 + \max\{b_j \mid 1 \le j < i \wedge a_j < a_i\}$. Toto riešenie vyžaduje $O(n^2)$ operácií.

b) Podľa [6.11.1 v 34, 20.2 v 18, 11]: Označme P_k najmenší zo všetkých posledných prvkov „zatiaľ nájdených“ podpostupností dĺžky k. To znamená, že ak $P_k = x$, tak existuje podpostupnosť dĺžky k, ktorá končí číslom x, ale neexistuje žiadna podpostupnosť dĺžky k, ktorá končí číslom menším ako x. Zjavne je postupnosť čísel P_k rastúca (ak $P_{k+1} = x$, existuje rastúca postupnosť dĺžky $k+1$, ktorá končí číslom x; keď toto číslo odstránime, dostaneme postupnosť dĺžky k, ktorá končí číslom $y < x$ a $P_k \le y$, lebo P_k je najmenšie také číslo; preto $P_k < P_{k+1}$). Na začiatku je $P_0 = -\infty$ a $P_i = \infty$ pre $i > 0$. Postupne, ako načítavame vstup, postupnosť P_k upravujeme.

   Načítajme teda a_i; koľko údajov sa zmení? Aby sme zmenili P_{k+1}, musí byť $a_i < P_{k+1}$. Avšak musí tiež platiť $P_k < a_i$, aby sme mohli číslo napojiť na nejakú postupnosť dĺžky k. Zmení sa teda iba jedna hodnota P_{k+1} a to taká, že $P_k < a_i < P_{k+1}$. Keďže P_k je rastúca postupnosť, vhodné k môžeme hľadať binárne (úloha 415), takže celkovo potrebujeme len $O(n \log n)$ operácií. (Mimochodom, postupnosť P_k tvorí aj výsledok.)

523. Úloha je z teórie grafov: mestá predstavujú vrcholy, cesty medzi mestami sú hrany; úlohou je nájsť najkratšie cesty z vrcholu u do všetkých ostatných vrcholov, pričom predpokladáme, že dĺžky hrán sú nezáporné. Túto úlohu rieši Dijkstrov[29] algoritmus.

V každom vrchole si budeme pamätať dĺžku doteraz nájdenej najkratšej cesty z počiatočného vrcholu u. Vrcholy budeme mať rozdelené do dvoch skupín: H budú „hotové" vrcholy, do ktorých už vieme dĺžku najkratšej cesty. Ostatné vrcholy budú v skupine S, treba ich ešte spracovať. Pre ne poznáme dĺžku $d(x)$ najkratšej cesty z u, ktorá ide *iba cez hotové mestá* (ak taká neexistuje, $d(x) = \infty$). Na začiatku je $H = \emptyset$, $d(u) = 0$ a $d(x) = \infty$ pre $x \neq u$.

V každom kroku vyberieme z S vrchol x, ktorý má hodnotu $d(x)$ (vzdialenosť cez hotové mestá) najmenšiu. Tvrdíme, že táto vzdialenosť je skutočná: Zoberme ľubovoľnú cestu, ktorá nevedie len cez vrcholy z H. Nech y je prvý vrchol zo skupiny S na tejto ceste. Keďže dĺžky hrán sú nezáporné a $d(x) \leq d(y)$, žiadna takáto cesta nie je kratšia ako tá, ktorú sme už našli. Hodnota $d(x)$ je teda definitívna a vrchol x môžeme presunúť do H. Tým sa však mohli zmeniť vzdialenosti tých miest z S, do ktorých vedie kratšia cesta cez x. Takže susedom v vrcholu x upravíme $d(v) = \min\{d(v), d(x) + l(x,v)\}$, kde $l(x,v)$ je dĺžka hrany z x do v.

V každom kroku spracujeme jedno mesto z S v lineárnom čase. Celkovo teda treba $O(n^2)$ operácií. Pozri [6.1 v 28, 4.13 v 38, 6 v 8].

524. Pozri 515.

525. Určite uhly otočenia jednotlivých kĺbov pre body P_1 a P_2, $(\alpha_1, \beta_1, \gamma_1)$ a $(\alpha_2, \beta_2, \gamma_2)$. Uhly α_1, α_2 vieme vypočítať pomocou arctg, uhly β_i a γ_i pomocou kosínusovej a sínusovej vety z trojuholníka $K_2P_iK_3$. Počet operácií *otoč* potrebných na presun ramena je $\max\{|\alpha_1 - \alpha_2|, |\beta_1 - \beta_2|, |\gamma_1 - \gamma_2|\}$. Je treba vybrať najbližší mrežový bod k bodu P_2.

531. Nájdite výskyt slova tisíc a sto. Pozor na výnimky: desať, jedenásť, štrnásť, dvesto, dvetisíc, a pod.

532. Pozri 122.

533. Zostrojme graf, ktorého vrcholy sú cifry a premenné. Hranami spojíme tie dvojice, ktoré sa majú rovnať. (Napr. pre vstup $0XY1$, $0YZZ$ budeme mať hrany X–Y, Y–Z a 1–Z.) Každému komponentu súvislosti tohto grafu musí zodpovedať v riešení práve jedna cifra. Ak teda v niektorom komponente leží viac ako jedna cifra, úloha nemá riešenie. V opačnom prípade nech k je počet komponentov, ktoré obsahujú len premenné. Každému z týchto k komponentov musíme priradiť jednu cifru. A je úplne jedno, ktorému akú, preto riešení je 10^k.

534. Podľa zadania začíname s postupnosťou čísel väčších ako 2. Naopak, pri riešení si najskôr nebudeme všímať čísla menšie ako k, t.j. začneme s postupnosťou $k, k+1, k+2, \ldots$. V tejto postupnosti je k prvý raz vedúcim už v 1. kroku. Teraz budeme postupne uvažovať aj menšie čísla (resp. postupnosti začínajúce $k-1$, $k-2$, až 3). Ak sa číslo k stane vedúcim prvý raz v p-tom kroku, pričom uvažujeme postupnosť $i+1, i+2, \ldots, k, \ldots$, koľko krokov s vedúcim členom i sme vynechali? Ak si nevšímame čísla menšie ako i, číslo i bude vedúcim v každom i-tom kroku. Takže sme vynechali $1 + \lfloor (p-1)/i \rfloor$ krokov.

535. Nájsť jedno riešenie sa dá efektívne. Jeden možný postup: rozdelíme šachovnicu na štyri menšie. Tú, ktorá obsahuje dieru, vydláždime zopakovaním tohto postupu. (Prípadne ak už je len rozmeru 2×2, jednoducho tam položíme jedno trimino.) Čo teraz so zvyšnými tromi štvrtinami šachovnice? Každú z nich vieme (rovnakým postupom) vydláždiť tak, aby nám zostalo voľné nami zvolené políčko... už si len dobre zvoliť.

541. Majme mince s hodnotami $m_1, \ldots, m_k$ a sumu na zaplatenie S (všetky ceny premeníme na haliere). Vo všeobecnosti sa dá úloha riešiť metódou dynamického programovania:

[29] Edsger Wybe Dijkstra (1930–2002), holandský matematik, informatik a fyzik

Postupne budeme vypĺňať tabuľku p rozmerov $k \times S$, kde $p_{i,j}$ bude počet spôsobov, ktorými vieme zaplatiť sumu j, ak budeme používať iba mince $m_1, \ldots, m_i$. Ako vypočítame hodnotu $p_{i,j}$? Pri platení máme dve možnosti: buď mincu hodnoty m_i použijeme, alebo nie. Ak nie, použijeme iba prvých $i-1$ mincí – počet takýchto spôsobov je $p_{i-1,j}$. Ak mincu použijeme, ostáva nám zaplatiť sumu $j - m_i$ a je $p_{i,j-m_i}$ takých spôsobov. Teda $p_{i,j} = p_{i-1,j} + p_{i,j-m_i}$. Keďže v našej úlohe je k konštanta, časová zložitosť je $O(S)$.

Existuje však aj riešenie v konštantnom čase. Vydeľme všetky hodnoty piatimi (majme mince 1, 2, 4, 10, 20, 40, 100). Ľahko zistíme, že $p_{1,S} = 1$ a $p_{2,S} = 1 + \lfloor S/2 \rfloor$. S trochou námahy sa nám podarí odvodiť $p_{3,S} = (k+1)(k+1+\lfloor r/2 \rfloor)$, kde $S = 4k+r$ a $0 \le r < 4$. Viac so stredoškolskou matematikou asi nevypočítame. Pomocou generujúcich funkcií [7. kapitola v 14] však vieme odvodiť vzorec riešiaci celú túto úlohu. Pre zaujímavosť, spomínaný vzorec má $\sum_i m_i = 177$ členov `:-)`.

542. Použijeme techniku zametania. Budeme prechádzať zľava doprava akousi pomyselnou priamkou (sweepline; rovnobežná s osou y), pričom si budeme pamätať, v akom poradí sa úsečky na tejto priamke nachádzajú. Zrejme tá, ktorú vidíme, je najnižšia. Kedy sa situácia na sweepline zmení? Jedna možnosť je, že začne alebo skončí nejaká úsečka; druhá možnosť je, že sa pretnú dve úsečky – na sweepline sa tým pádom vymenia. Tieto udalosti budeme uchovávať v halde – potrebujeme vedieť rýchlo zistiť, ktorá udalosť je najskôr. Úsečky na sweepline je dobré uchovávať vo vyváženom strome; keď začne nová úsečka, pridáme ju do stromu a pozrieme sa na jej bezprostredných susedov (ak sa s niektorým pretína, pridáme túto udalosť do haldy); keď sa dve úsečky pretnú, treba ich v strome vymeniť a skontrolovať priesečníky s ich bezprostrednými susedmi.

Tento algoritmus sa dá implementovať v čase $O(n \log n)$. Pozri [2.1 v 4 alebo 33.2 v 6] pre podobný algoritmus na hľadanie priesečníkov úsečiek.

543. Treba si zvoliť také poradie permutácií, ktoré sa dá efektívne popísať. Niektoré možné prístupy:

a) Využite, že ak viete vygenerovať všetky permutácie z i prvkov, potom viete vygenerovať ľahko aj z $i+1$ prvkov. Stačí prvok $i+1$ umiestniť na všetky možné pozície v každej permutácii z i prvkov.

b) Generujte lexikograficky po sebe nasledujúce permutácie.

c) Existuje riešenie, v ktorom každú ďalšiu permutáciu dostaneme z predchádzajúcej jedinou výmenou.

544. Záhadné reťazce majú tvar uvu^Rv^R, kde w^R označuje zrkadlový obraz (*reverz*) reťazca w; $s = u^R$, $t = v^R$. Uvedomte si, že $|uv| = |st|$. Teda u stačí voliť dĺžky $1, \ldots, |uvst|/2$. Tým sme určili aj v a ostáva overiť, či sme zvolili správne. Časová zložitosť tohto riešenia je $O(n^2)$, kde n je dĺžka celého reťazca.

Šikovnejšie riešenie od Maja Dvorského je založené na hľadaní výskytu $v^Ru^R = ts = (uv)^R$ v reťazci $stst$. Dokážte, že sa tam nachádza práve vtedy, keď má slovo $uvst$ požadovaný tvar. Vyhľadávanie podreťazca sa dá realizovať lineárne od súčtu dĺžky vzorky a textu, teda máme časovú zložitosť $O(n)$.

545. Ak $m > k$, stačí nájsť takú postupnosť prevozov, ktorých výsledkom klesne o jedného počet misionárov aj kanibalov na jednom brehu. Ak $m = k$, tak pre $m < 3$ je to ľahké, prípad $m = 3$ je v zadaní a pre $m > 3$ sa to nedá. Dokázať to môžete sporom.

611. Pozri úlohu 233. Výsledok sa dá zovšeobecniť. Nech $P(n)$ je počet *dobrých* postupností dĺžky n. Pre $0 \le n < k$ je $P(n) = 2^n$ – všetky postupnosti sú *dobré*. Vo všeobecnosti sa *dobrá* postupnosť končí 0 až $k-1$ nulami, pred ktorými je jednotka a pred touto jednotkou je ľubovoľná *dobrá* postupnosť, teda $P(n) = P(n-1) + P(n-2) + \cdots + P(n-k)$.

Priamočiarou implementáciou dostávame algoritmus so zložitosťou $O(nk)$. Ľahko si však môžeme všimnúť, že

$$\begin{aligned} P(n) &= \quad P(n-1) + P(n-2) + \cdots + P(n-k+1) + P(n-k) \\ P(n+1) &= P(n) + P(n-1) + P(n-2) + \cdots + P(n-k+1), \end{aligned}$$

teda $P(n+1) - P(n) = P(n) - P(n-k)$; $P(n+1) = 2P(n) - P(n-k)$.

612. Zakódujte vpravohľadiacich vojakov 1 a vľavohľadiacich 0. Teraz stačí v každom kroku nahradiť všetky dvojice 10 dvojicou 01. Pozor, otáčania treba vykonať „paralelne“. To, že otáčanie vždy skončí, je najľahšie nahliadnuť nasledovne: Predstavte si, že dvaja vojaci, ktorí hľadia na seba, sa neotočia, ale vymenia si miesta. Ak sa na vojakov dívame len ako na postupnosť núl a jednotiek, bude po každej sekunde vyzerať presne rovnako ako v pôvodnej úlohe. Teraz ale ľahko vidíme, že vojaci, ktorí sa na začiatku otočili doprava, postupujú doprava, až kým sa už ďalej ísť nedá, a ich opačne otočení kolegovia zase postupujú doľava. Preto určite v konečnom čase skončíme v situácii, kde najskôr budú stáť všetci hľadiaci doľava a za nimi všetci hľadiaci doprava. Skúste si dokázať, že otáčanie sa skončí vždy po najviac $n-1$ krokoch.

613. Huffmanov[30] kód. Predstavme si písmenkový strom, v ktorom sú uložené kódy jednotlivých znakov. Podmienka c) zo zadania hovorí, že každému znaku zodpovedá jeden list tohto stromu. *Cenou stromu* nazvime priemerný počet bitov potrebný na zakódovanie jedného znaku zo vstupu, ak použijeme kód zodpovedajúci danému stromu.

Ako spočítať cenu konkrétneho stromu? Pre každý znak vieme pravdepodobnosť jeho výskytu v texte, touto vynásobíme dĺžku zodpovedajúceho kódového slova (teda hĺbku zodpovedajúceho listu) a výsledky pre všetky znaky sčítame.

Algoritmus na nájdenie stromu T s minimálnou cenou je založený na nasledujúcich dvoch tvrdeniach.

Tvrdenie 1. Ak p_1 a p_2 sú dve minimálne pravdepodobnosti výskytu znaku spomedzi $p_1, p_2, \ldots, p_n$ (ak je minimálnych viac, ľubovoľné z nich), existuje optimálny kódovací strom T, v ktorom sú listy l_1 a l_2 zodpovedajúce týmto znakom bratia. (Bratia sú listy, ktoré majú spoločného otca – „čerešničky“.)

Tvrdenie 2. Ak nájdeme optimálny strom S pre pravdepodobnosti $p_1 + p_2, p_3, \ldots, p_n$, pričom p_1 a p_2 sú dve najmenšie, potom pre $p_1, p_2, \ldots p_n$ je optimálny strom T, ktorý dostaneme z S nahradením listu zodpovedajúcemu znaku s početnosťou $p_1 + p_2$ čerešničkami, bratmi zodpovedajúcimi znakom s početnosťami p_1 a p_2.

Priamočiara implementácia tohto algoritmu má časovú zložitosť $O(n^2)$. Časová zložitosť $O(n \log n)$ sa dá dosiahnuť použitím haldy, v ktorej si pamätáme aktuálne pravdepodobnosti. Iný, trikový algoritmus s rovnakou zložitosťou spomíname v riešení úlohy 724.

614. Uvedieme viacero rôzne dobrých riešení.

a) Asi najhoršie riešenie je použiť backtracking: postupne predlžovať cestu a keď sa už nedá, snažiť sa posledné predĺženie nejako zmeniť. Malé vylepšenie dosiahneme, keď cestu prestaneme predlžovať v prípade, že by bola dlhšia než doteraz minimálna, alebo ak by sme ju chceli predĺžiť na políčko, kadiaľ už vedie.

b) Zaujímavejšie je riešenie, v ktorom stále prechádzame cez všetky políčka. V každom políčku je uložená dĺžka najkratšej cesty od políčka $[1,1]$. Každému políčku skontrolujeme susedov a ak sa dá, zmenšíme dĺžku doteraz najkratšej cesty, ktorá doň vedie. Skončíme, ak sa nám nepodarí zmeniť ani jednu hodnotu. Časová zložitosť tohto riešenia je $O(n^4)$.

[30] David Albert Huffman (1925–1999), americký informatik

c) Namiesto políčiek si predstavíme neorientovaný graf, v ktorom budú vrcholmi políčka pospájané podľa susednosti. Teraz stačí vedieť hľadať najkratšiu cestu v neorientovanom grafe (Dijkstrov algoritmus, pozri riešenie úlohy 523). Aj toto riešenie má časovú zložitosť $O(n^4)$.

d) Ešte lepšie riešenie je uvedomiť si, že zadaný graf je riedky (má málo hrán). V takomto prípade sa oplatí množinu S z riešenia úlohy 523 implementovať pomocou haldy. Takto dostávame riešenie s časovou zložitosťou $O(n^2 \log n)$.

615. Aký by mal byť dobrý labyrint? Nemal by mať príliš dlhé rovné chodby, mal by mať dosť križovatiek a ideálne je, keď k pokladu vedie od vchodu len jedna cesta.

Existuje viacero tradičných prístupov, používaných pri generovaní bludísk. Väčšina z nich je založená na prístupoch z teórie grafov. Predstavme si graf, ktorého vrcholy sú políčka štvorcovej siete, hranou je spojená každá dvojica susedných políčok. Bludiská, v ktorých medzi každou dvojicou políčok existuje práve jedna cesta, zodpovedajú kostrám tohto grafu. Zostáva nejako zvoliť vhodnú, dostatočne náhodnú kostru.

a) Náhodné prehľadávanie do hĺbky. Začneme v mieste, kde je vchod do bludiska. Vždy, keď máme viac susedov, v ktorých sme ešte neboli, vyberieme si z nich náhodného. Pre každé políčko si zapamätáme hranu, ktorou sme naň prvýkrát prišli.

b) Zvolíme každej hrane náhodnú váhu a spustíme algoritmus na hľadanie najlacnejšej kostry. Pri Kruskalovom algoritme dokonca stačí v náhodnom poradí spracúvať dvojice susedných políčok. (Pozri riešenie úlohy 813.)

c) Na začiatku nech žiadne políčko nemá ani jedny „dvere“. Niektoré políčka vyhlásime za aktívne. Počas výpočtu udržujeme množinu aktívnych políčok. V každom kroku vyberieme jedno aktívne políčko. Ak už každý jeho sused má aspoň jedny dvere, políčko len prestane byť aktívne. V opačnom prípade vyberieme náhodného suseda bez dvier, spravíme k nemu dvere z tohto políčka, a aj sused bude odteraz aktívny.
Dá sa experimentovať s tým, ako voliť, ktoré aktívne políčko vyberieme. Zaujímavé výsledky dostaneme napríklad vtedy, ak ho zakaždým vyberieme náhodne spomedzi niekoľkých najkratšie aktívnych políčok.

621. Optimálne riešenie je zjavne lomená čiara, ktorej body zlomu sú niektoré konce múrikov, a navyše ide „zľava doprava“. Konce múrikov, štart a cieľ budeme volať *významné body*. Pre jednoduchosť dodefinujme $y_{1,n+1} = y_{2,n+1} = 0$. Označme $c_{l,k}$ (kde $l \in \{1, 2\}$, $0 < k \leq n + 1$) minimálnu dĺžku cesty z bodu $[x_0, 0]$ do bodu $[x_k, y_{l,k}]$. Hodnota $c_{1,n+1}$ je hľadaným riešením zadanej úlohy. Ako tieto hodnoty spočítať? Najkratšia cesta do bodu $B = [x_k, y_{l,k}]$ určite vyzerá tak, že najkratšou cestou prídeme do nejakého iného významného bodu naľavo od B a odtiaľ pôjdeme priamo do B.

Priamočiare riešenie teda vyzerá nasledovne: Pre každý významný bod prejdeme všetky, ktoré od neho ležia naľavo. Pre každý z nich zistíme, či sa odtiaľ dalo prísť priamo (teda v ceste nie sú žiadne múriky). Ak áno, pozrieme sa, či sme týmto spôsobom našli kratšiu cestu ako máme doteraz nájdenú. Toto riešenie má časovú zložitosť $O(n^3)$.

Lepšie riešenie: Dá sa efektívnejšie zisťovať, či sa dva významné body „vidia“, teda či sa medzi nimi dá priamo prejsť. Všimnime si nejaký významný bod B. Pôjdeme postupne spracúvať múriky od neho doľava a zisťovať, ktoré významné body sú z neho viditeľné. Keď sme už spracovali niekoľko múrikov, tie nám zakrývajú niekoľko intervalov smerov z B. Tieto intervaly si môžeme pamätať napr. vo vyváženom strome. Keď ideme spracovať nový múrik, vypočítame si pre každý jeho koncový bod uhol smeru z B doň, pozrieme, či už sú tieto smery zakryté, a následne upravíme pamätanú množinu zakrytých intervalov. Takéto riešenie má časovú zložitosť $O(n^2 \log n)$. Pravdepodobne existuje ešte lepšie riešenie tejto úlohy, to už ale presahuje rámec tohto textu.

622. Pre menej ako tri čísla je problém triviálny. V ďalšom texte predpokladáme, že čísla sú aspoň tri. Označme si čísla po obvode kruhu $v_0, v_1, \ldots, v_{n-1}$. Nech $s_{i,j}$ je súčet

čísel od v_i po v_j vrátane, idúc po obvode kruhu v tom istom smere, ako sme ich označili. (Napríklad teda $s_{n-1,1} = v_{n-1} + v_0 + v_1$.) Pre jednoduchosť budeme občas používať aj iné indexy čísel na obvode kruhu ako 0 až $n-1$, tieto indexy budeme chápať mod n. Teda napríklad $s_{n-2,n+1}$ bude to isté ako $s_{n-2,1}$.

Zoberme papier a napíšme si naň všetkých n^2 hodnôt $s_{i,j}$ pre aktuálnu pozíciu. Teraz spravme jednu povolenú operáciu, zoberme nový papier a napíšme naň všetky nové hodnoty $s_{i,j}$. Ako sa budú naše dva papiere líšiť? Ak pôvodná postupnosť čísel vyzerala:

$$\ldots,\ v_{i-1} = x,\ v_i = -y,\ v_{i+1} = z,\ \ldots$$

nová postupnosť bude vyzerať:

$$\ldots,\ x - y,\ y,\ z - y,\ \ldots$$

Súčet nových troch hodnôt je rovnaký ako súčet pôvodných troch hodnôt. Preto sa súčty $s_{a,b}$, ktoré tento interval neobsahujú, alebo ho obsahujú celý, nezmenia. Zmenia sa súčty, ktoré končia na pozícii $i-1$ a i. Ako? Nech pôvodne bolo $s_{a,i-2} = S$, $s_{a,i-1} = S + x$, $s_{a,i} = S + x - y$. Po našej operácii bude $s_{a,i-2} = S$, $s_{a,i-1} = S + x - y$ a $s_{a,i} = S + (x - y) + y = S + x$. Takže zodpovedajúca časť našich papierov bude obsahovať tie isté čísla, len v inom poradí. Podobne dostaneme tie isté čísla, len v inom poradí, pre úseky začínajúce na pozícii i a $i+1$. Jediné dva súčty, ktoré sa naozaj zmenia, sú $s_{i,i}$ a $s_{i+1,i-1}$, teda súčet „len i-teho čísla“ a „všetkých ostatných čísel“. Pôvodne bol $s_{i,i}$ záporný a $s_{i+1,i-1}$ kladný a v absolútnej hodnote väčší. Ľahko nahliadneme, že hodnota $s_{i,i}^2 + s_{i+1,i-1}^2$ sa vykonaním našej operácie zmenší. Každá povolená operácia teda zmenší hodnotu $\sum_{0 \le i,j < n} s_{i,j}^2$. Preto ľubovoľná postupnosť povolených operácii v konečnom čase skončí v požadovanom cieľovom stave.

623. Určite súradnice ľavého horného a pravého dolného vrcholu prieniku obdĺžnikov. Označme X_{ALD}, Y_{ALD}, X_{APH}, Y_{APH} súradnice ľavého dolného a pravého horného vrcholu obdĺžnika A, analogicky pre B. Ak majú A a B neprázdny prienik, je to obdĺžnik so súradnicami vrcholov: $X_{LD} = \max\{X_{ALD}, X_{BLD}\}$, $Y_{LD} = \max\{Y_{ALD}, Y_{BLD}\}$, $X_{PH} = \min\{X_{APH}, X_{BPH}\}$, $Y_{PH} = \min\{Y_{APH}, Y_{BPH}\}$. Ich prienik je prázdny práve vtedy, keď vyjde $X_{LD} > X_{PH}$ a/alebo $Y_{LD} > Y_{PH}$.

624. Existuje veľa rôzne efektívnych riešení, uvedieme niektoré z nich:

a) Deliteľov zistíme ako pri testovaní prvočíselnosti: preveríme všetky čísla z intervalu $\langle 1, \sqrt{x} \rangle$.

b) Nech je prvočíselný rozklad $x = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$. Súčet všetkých deliteľov čísla x označme $S(x)$. Platí $s(x) = S(x) - x$ a $S(x) = (1 + p_1 + p_1^2 + \cdots + p_1^{a_1})(1 + p_2 + p_2^2 + \cdots + p_2^{a_2}) \cdots (1 + p_k + p_k^2 + \cdots + p_k^{a_k}) = (p_1^{a_1+1} - 1)/(p_1 - 1) \cdot (p_2^{a_2+1} - 1)/(p_2 - 1) \cdots (p_k^{a_k+1} - 1)/(p_k - 1)$.

c) Podobne ako Eratostenovo[31] sito. Vypočítame pole S, pričom $S[i]$ bude súčet deliteľov čísla x. Na začiatku toto pole obsahuje nuly, potom postupne pre každé číslo x zväčšíme o x každú z hodnôt $S[kx]$.

Desať najmenších spriatelených dvojíc: $(220, 284)$, $(1184, 1210)$, $(2620, 2924)$, $(5020, 5564)$, $(6232, 6368)$, $(10744, 10856)$, $(12285, 14595)$, $(17296, 18416)$, $(63020, 76084)$, $(66928, 66992)$.

625. Je šikovné najprv z analyzovaného riadku vymazať všetky medzery už pri jeho čítaní. V programe môžeme čítať buď číslo, alebo oddeľovač, alebo identifikátor. Problematický je len identifikátor, lebo ten môže byť ešte kľúčovým slovom. Uvedomte si, že `AORC` sa musí rozložiť na `A`, `OR` a `C`. Keď začneme čítať identifikátor, musíme kontrolovať, či jeho nejaká podčasť nie je kľúčovým slovom a ak je, načítali sme identifikátor a kľúčové slovo. Ak sa pri čítaní identifikátora vyskytne číslica, potom si všímame iba číslice a sú

[31] Eratostenes Cyrénsky (276–194 p.n.l.), grécky matematik, geograf a astronóm

súčasťou identifikátora. Číslo môže začínať iba po kľúčovom slove alebo oddeľovači. Aby sme kľúčové slová ľahko našli, je výhodné si ich pamätať v tabuľke a v pomocnom poli indexovanom písmenami si pamätať počiatočný a koncový index kľúčových slov začínajúcich daným písmenom. Pozorne treba vyriešiť ešte viacznakové oddeľovače.

Existujú automatické nástroje (*lex*, *yacc*, *flex*, *bison*), ktoré v „reálnom živote“ programátorovi s takouto úlohou značne pomôžu. Ak vieme popísať jednotlivé typy tokenov pomocou regulárnych výrazov, *flex* nám vygeneruje program v jazyku C, ktorý svoj vstup rozdelí na postupnosť nami definovaných tokenov. Ak teraz popíšeme vhodnou bezkontextovou gramatikou syntax korektných programov v našom jazyku, *bison* nám z nej vygeneruje program v jazyku C, ktorý k vstupnej postupnosti tokenov nájde jej hierarchickú štruktúru (teda to, ako sa tieto tokeny spájajú do príkazov, príkazy do blokov a bloky do programu).

631. Postupujeme podobne ako pri delení veľkých čísel. Nech $a(x) = a_n x^n + \cdots + a_0$, $b(x) = b_m x^m + \cdots + b_0$, kde $a_n, b_m \neq 0$. Ak $n < m$, tak $a(x) = 0 \cdot b(x) + a(x)$, teda $a(x) \operatorname{div} b(x) = 0$ a $a(x) \bmod b(x) = a(x)$. Ďalej predpokladajme, že $n \geq m \geq 0$. Budeme postupovať indukciou: ak $n = 0$, potom $m = 0$, teda $a(x)/b(x) = r(x) = a_0/b_0$ a zvyšok $q(x) = 0$. Predpokladajme, že vieme deliť polynóm stupňa menšieho ako n. Nech $a'(x) = a(x) - a_n/b_m x^{n-m} \cdot b(x)$. Všimnite si, že $(a_n/b_m x^{n-m})$-násobok sme zvolili tak, aby sme vynulovali najvyšší koeficient $a(x)$, takže $a'(x)$ má stupeň menší ako n. Podľa indukčného predpokladu vieme vypočítať $q'(x) = a'(x) \operatorname{div} b(x)$ a $r'(x) = a'(x) \bmod b(x)$. Potom $q(x) = q'(x) + a_n/b_m x^{n-m}$ a $r(x) = r'(x)$.

632. Jedna žaba zjavne nepotrebuje skákať nikam. Pre dve až desať žiab môžeme nájsť najlepšie riešenie prehľadaním priestoru možných stavov do šírky. Dostaneme, že potrebujeme nasledujúce počty skokov: 3, 5, 10, 16, 21, 31, 36, 50, 55. Z tejto postupnosti môžeme uhádnuť, ako vyzerá optimálne riešenie pre väčšie počty žiab: Ak je ich počet (označme ho n) nepárny a väčší ako 1, potrebujeme $n(n+3)/2 - 4$ skokov. Ak je párny, potrebujeme $n(n+1)/2$ skokov. Nasledujúcim krokom by malo byť, že zoberieme optimálne riešenia pre 9 a 10 žiab a skúsime ich zovšeobecniť.

633. Obrázok sa vždy dá nakresliť jedným ťahom, bez toho, aby sme šli dvakrát po tej istej čiare. Takýto postup kreslenia je zjavne optimálny.

Existujú všeobecné algoritmy, ktoré zistia, ako daný obrázok nakresliť jedným ťahom (ak sa to dá), stručný popis uvádzame v riešení úlohy 643. V tomto prípade však stačilo nájsť konkrétny postup v závislosti od n.

Predstavme si, že začneme kresliť v ľubovoľnom vrchole, v každom kroku nakreslíme uhlopriečku do vrcholu, ktorý je o k v smere hodinových ručičiek ďalej, a prestaneme, keď sa vrátime do vrcholu, kde sme začínali. Obrázok, ktorý takto nakreslíme, nazveme *cik-cak s krokom* k.

Zjavne hľadaný obrázok vieme rozdeliť na cik-caky s krokom 1 až $(n-1) \operatorname{div} 2$. Pritom cik-cakov s konkrétnym krokom k bude práve $\operatorname{nsd}(n, k)$. Ostáva prísť na to, v akom poradí ich kresliť.

Povedzme si, že pre konkrétne $k > 1$ budú cik-caky s krokom k začínať vo vrcholoch s číslami 1 až $\operatorname{nsd}(n, k)$. Teraz začneme kresliť obvod mnohouholníka (čo je vlastne jediný cik-cak s krokom 1), pričom vždy, keď prídeme do nejakého vrcholu, nakreslíme všetky cik-caky s krokom väčším ako 1, ktoré v ňom začínajú.

Pozor! Nasledovný algoritmus je *zlý*: Začneme s krokom $k = 2$. Z prvého vrcholu nakreslíme cik-cak s krokom k, potom sa posunieme do susedného vrcholu. Ak sme odtiaľto už kreslili cik-cak s krokom k, k o jedna zväčšíme a pokračujeme z tohto vrcholu rovnakým spôsobom ďalej, až pokým k neprekročí $(n-1) \operatorname{div} 2$. Vtedy prejdeme zvyšné vrcholy s krokom $k = 1$. Najmenší počet vrcholov, keď tento algoritmus zlyhá, je 24. Problém je

v tom, že ak je $\sum_{k=2}^{(n-1)\operatorname{div}2} \operatorname{nsd}(n,k) > n$, začneme znovu chodiť po už nakreslenej časti obvodu n-uholníka.

634. Označme $g_n(i,k)$ počet takých k-prvkových podmnožín množiny $\{i, i+1, \ldots, n\}$, ktoré obsahujú prvok i. Zrejme $g_n(i,k) = \binom{n-i}{k-1}$. Algoritmus môžeme zapísať takto

1. $s := x$; $i := 1$;
2. ak $g_n(i,k) \geq s$, ďalší prvok podmnožiny je i; $i := i+1$; $k := k-1$;
3. ak $g_n(i,k) < s$, tak $s := s - g_n(i,k)$; $i := i+1$;
4. ak $k = 0$, sme hotoví, inak pokračujeme bodom 2.

Hodnotu $g_n(1,k)$ vieme vypočítať v čase $O(k)$ podľa definície kombinačného čísla. Po každej zmene i alebo k vieme novú hodnotu $g_n(i,k)$ vypočítať zo starej v konštantnom čase. Takto dostávame riešenie s časovou zložitosťou $O(n)$.

635. Riešenie v čase $O(n^3)$: Predĺžime úsečky, ktoré ohraničujú zadané obdĺžniky, na priamky. Takto rozdelíme rovinu na $O(n^2)$ obdĺžnikových častí. V dvojrozmernom poli si pre každý zadaný obdĺžnik zaznačíme, ktoré z častí obsahuje.

Existujú efektívnejšie riešenia, založené na *zametaní*. Rozdelíme si rovinu na vodorovné pásy rovnako ako v predchádzajúcom riešení. Budeme teraz pásy spracúvať v poradí zhora nadol, pričom si udržiavame množinu obdĺžnikov, ktoré majú s aktuálnym pásom neprázdny prienik. Pomocou vyvážených alebo intervalových stromov sa dá dosiahnuť časová zložitosť $O(n \log n)$. Podrobnejšie napr. v [VIII.5.1.3 v 35].

641. Určite, aké veľké pole v závislosti od počiatočnej konfigurácie isto stačí na simulovanie. Prefarbovanie vždy skončí po maximálne n krokoch.

642. Jedno možné riešenie: Rozdeľte M na M_1 a M_2 také, že M_1 obsahuje len párne (majú posledný bit `0`) a M_2 len nepárne prvky (majú posledný bit `1`). Takto máme zaručené, že medzi prvkom z M_1 a prvkom z M_2 nebude ich aritmetický priemer. Takže stačí rekurzívne preusporiadať M_1 a M_2 (obe skupiny rovnako rozdelíme podľa predposledného bitu, potom podľa predpredposledného, atď.).

643. Úlohou je nájsť tzv. *eulerovský*[32] *ťah* – uzavretý ťah, ktorý prechádza každou hranou v grafe.

Ak graf obsahuje eulerovský ťah, musí byť súvislý. Navyše, keďže sme z každého vrcholu vyšli toľkokrát, koľkokrát sme doňho vošli, musí mať každý vrchol párny stupeň (toto skontrolujeme na začiatku).

Naopak, v súvislom grafe, ktorého vrcholy majú párny stupeň, vieme nájsť eulerovský ťah vždy: Začneme v ľubovoľnom vrchole v_0 a ľubovoľným smerom sa vydáme po hranách grafu (po žiadnej však nesmieme ísť dvakrát). Takto pokračujeme, kým môžeme. Keďže vrcholy majú párny stupeň, ak do nejakého vrcholu prídeme, existuje hrana, ktorou môžeme odísť. Preto tento ťah (označme ho t_0) skončíme tam, kde sme ho začali – vo vrchole v_0. Ak sme v tomto okamihu prešli celý graf, môžeme skončiť.

V opačnom prípade budeme ťah t_0 predlžovať. Keďže je graf súvislý, na t_0 existuje vrchol v_1, z ktorého vychádza ešte nepoužitá hrana. Začnime teraz nový ťah t_1 z vrcholu v_1. Ak pôjdeme iba po nepoužitých hranách, musíme sa opäť vrátiť do v_1. Výsledný ťah vytvoríme takto: začneme vo v_0, pôjdeme po hranách t_0 až do v_1. Odtiaľ pôjdeme po hranách t_1, vrátime sa späť do v_1 a po hranách t_0 do v_0. Takto sme dostali *dlhší* ťah. Tento postup môžeme opakovať, kým nedostaneme ťah, ktorý prechádza cez všetky hrany.

Algoritmus s časovou a pamäťovou zložitosťou $O(n+m)$ sa dá elegantne implementovať upraveným prehľadávaním do hĺbky: Vrchol spracujeme tak, že najskôr rekurzívne spracujeme všetky vrcholy, do ktorých vedie ešte nepoužitá hrana (hranu označíme ako použitú) a nakoniec vrchol vypíšeme.

Viac sa o eulerovských ťahoch dočítame v [9.A v 38, 10 v 8].

[32] Leonhard Euler (1707–1783), švajčiarsky matematik, fyzik, mechanik a astronóm

644. Úloha sa dá riešiť podobnou technikou ako pivotizácia. Pole počas riešenia môžeme udržiavať v tvare biele, modré, červené a zatiaľ neupratané, alebo v tvare biele, modré, zatiaľ neupratané a červené, pričom si pamätáme konce, alebo začiatky skupín. Šikovnejší je druhý tvar. Skúste úlohu vyriešiť jedným cyklom, pričom si budete všímať prvý prvok zo zatiaľ neusporiadaných škatúľ a podľa jeho farby ho dávať na jeho miesto. Keď to urobíte šikovne, cyklus prebehne najviac m-krát a v každom cykle urobíte najviac jednu výmenu škatúľ. Nech j a k sú indexy prvej a poslednej zatiaľ neusporiadanej škatule, všimnite si rozdiel $k - j$.

645. Ak je výraz úplne uzátvorkovaný, úlohu vieme riešiť jednoduchým rekurzívnym programom – výraz je totiž buď ⟨konštanta⟩, alebo ⟨premenná⟩, má tvar `-`⟨výraz⟩ alebo `(` ⟨výraz⟩ ⟨operátor⟩ ⟨výraz⟩ `)`.

Iné riešenie je využiť zásobník, kam si budeme ukladať načítané symboly. Keď príde pravá zátvorka, urobíme *redukciu* – časť výrazu na zásobníku po najbližšiu ľavú zátvorku preložíme do kódu kalkulačky. Napr. $A+B$ preložíme ako $\$:= A,\ \$+B,\ p_i := \$$ a namiesto $(A + B)$ do zásobníka vložíme p_i (p_i je prvé voľné pamäťové miesto).

Podúloha b) sa tiež dá riešiť pomocou rekurzie, ale nie celkom tak priamočiaro ako a). Na druhej strane, riešenie pomocou zásobníka je podobné ako v a). Treba si len rozmyslieť, kedy robiť redukciu (je to vtedy, keď príde pravá zátvorka alebo operátor s nižšou prioritou ako predošlý). Pozri tiež úlohu 732.

711. Jednotlivé body utriedime, pričom si zapamätáme, či sa jedná o začiatok alebo o koniec intervalu. (Uvedomte si, že nezáleží na tom, ktorému intervalu bod patrí. Dôležité je len, či je počiatočný, alebo koncový.) Teraz stačí body postupne prechádzať a počítať si, v koľkých intervaloch sa práve nachádzame. Treba si dať pozor na situáciu, ak v jednom bode nejaké intervaly končia a iné začínajú. Treba najskôr spracovať začiatky intervalov, až potom konce.

712. Prehľadávame do šírky alebo do hĺbky. Z každého voľného políčka sa pohneme na štyri strany, samozrejme, ak je to možné. Začneme v $[1, 1]$, ak sa dostaneme do $[n, n]$, sme hotoví. Toto riešenie sa ľahko programuje využitím rekurzie.

Iný pohľad: Spojenie existuje práve vtedy, keď neexistuje cesta po zničených uzloch z horného alebo pravého okraja k dolnému alebo ľavému. To overíme rovnako ako v predchádzajúcom riešení. Susednosť políčok v tomto prípade musíme rozšíriť aj na políčka po uhlopriečkach.

713. Číslo „rozbalíme“ tak, aby žiadna menšia cifra nepredchádzala väčšiu. Pri sčítavaní rovnaké cifry dávame spolu. Na koniec treba ešte upraviť tie cifry, ktoré sa dajú zapísať špeciálne (IV, IX, XL, XC, CD, CM). Pri odčítavaní postupujeme podobne, len cifry vyškrtávame a keď treba, „rozmeníme“ väčšiu cifru na menšie.

Lepšie riešenie je rozložiť číslo do poľa $C[1\,..\,7]$, $C[i]$ určuje počet výskytov jednotlivých cifier. Ak sa menšia cifra vyskytuje pred väčšou, je v poli reprezentovaná -1. Pri sčitovaní a odčítavaní stačí sčítať prípadne odčítať príslušné prvky poľa a výsledok upraviť tak, aby sa vo výsledku nevyskytovali symboly V, L a D viac než raz a symboly I, X a C viac než tri razy.

714. Netreba si pamätať celý text, stačí pole $p[1\,..\,k, 1\,..\,26]$ (dĺžka správy $\times$ počet písmen abecedy), kde $p[i, z]$ určuje počet výskytov znaku z na pozícii i. Potom i-te písmeno najpravdepodobnejšej správy je to, ktoré sa najčastejšie nachádza na i-tej pozícií. Pre text v zadaní boli správy `NIC` a `SEMINAR`.

715. V prvej podúlohe stačí kocky utriediť podľa väčšej cifry na domine a striedavo dávať najväčšie cifry vľavo dolu a vpravo hore. Nie je známe efektívne riešenie druhej podúlohy, vzorovým riešením je backtracking.

721. Od každého členu postupnosti odčítame p. Úseky, ktoré mali v pôvodnej postupnosti priemer väčší ako p, sú práve tie, ktoré majú v upravenej postupnosti kladný súčet.

Spočítame čiastočné súčty $s_0 = 0$, $s_i = s_{i-1} + a_i$, pre $0 < i \leq n$ (s_i je súčet prvých i zadaných čísel). Súčet čísel od i-teho po j-te teraz zistíme ľahko ako $s_j - s_{i-1}$. Hľadáme teda také dve čísla i, j, že platí $s_j - s_{i-1} > 0$ a $j - i$ je maximálne. Netreba skúšať všetkých $O(n^2)$ možností, lepšie je utriediť dvojice $[s_i, i]$ vzostupne podľa s_i. Utriedený zoznam už len prejdeme a počas toho si pamätáme, aké najmenšie i sme doteraz videli. Toto riešenie má časovú zložitosť $O(n \log n)$.

722. Ak $\mathrm{nsd}(a_i, b_j) \neq 1$ pre niektoré $1 \leq i \leq n$ a $1 \leq j \leq m$, vydelíme ním a_i aj b_j.

Iné riešenie: Využijeme, že $a_i, b_j \leq 500$. Čitateľ aj menovateľ stačí rozložiť na prvočinitele, na to stačí deliť a_i, b_j prvočíslami $\leq \sqrt{500}$, t.j. $2, 3, 5, 7, 11, 13, 17, 19$. Pri rozklade čitateľa pripočítavame k výskytu príslušného prvočísla 1, pri rozklade menovateľa zasa 1 odčítavame.

723. Možných riešení je viac.

a) Priamočiara simulácia: počítame hodnoty $x_i^{(t)}$ (koľko lajstier má úradník i v čase t) postupne s rastúcim t. Platí:

$$x_i^{(t+1)} = x_1^{(t)} U_{i,1}/100 + \ldots + x_n^{(t)} U_{i,n}/100$$

Keď sa nám prestanú vypočítané hodnoty výrazne meniť, máme približné riešenie. Časová zložitosť takéhoto prístupu je $O(tn^2)$ na odsimulovanie t krokov.

b) Označme U maticu, v ktorej v i-tom riadku a j-tom stĺpci je hodnota $U_{i,j}/100$ – teda desatinné číslo, zodpovedajúce tomu, akú časť hromádky si berie i-ty úradník od j-teho. Označme ďalej $X^{(t)}$ stĺpcový vektor obsahujúci hodnoty $x_i^{(t)}$.
Teraz odsimulovanie jednej hodiny vieme skrátene zapísať nasledovne:

$$X^{(t+1)} = U \cdot X^{(t)}$$

Odsimulovanie troch po sebe idúcich hodín teda vyzerá nasledovne:

$$X^{(t+3)} = U \cdot (U \cdot (U \cdot X^{(t)}))$$

Tu však môžeme využiť, že násobenie matíc je asociatívne, a teda platí

$$X^{(t+3)} = U^3 \cdot X^{(t)}$$

Vo všeobecnosti teda môžeme písať

$$X^{(t)} = U^t \cdot X^{(0)}$$

V čom nám takýto zápis pomôže? Keď poznáme maticu U^t, tým, že ju vynásobíme samú sebou, dostaneme maticu U^{2t}. Začneme teda s maticou U, pomocou t násobení matíc vypočítame maticu U^{2^t}, a tou následne vynásobíme $X^{(0)}$.
Časová zložitosť takéhoto prístupu je teda $O(tn^3)$ na odsimulovanie 2^t krokov. (Porovnajte si to s predchádzajúcim postupom.)

c) Využijeme, že zo zadania vieme, že existuje stabilné rozdelenie papierov. (Uvedomte si, že aj keď to platí, dokázať to nie je vôbec ľahké!)
Nech X je toto stabilné rozdelenie. Túto skutočnosť vieme matematicky zapísať: $X = U \cdot X$. Slovne: po odsimulovaní jedného kroku sa rozdelenie nezmení.
Keď si rovnosť $X = U \cdot X$ rozpíšeme, dostaneme sústavu n lineárnych rovníc. Tieto rovnice však určite nie sú lineárne nezávislé! Ich sčítaním dostaneme totiž rovnosť 0=0. Jednu ľubovoľnú z nich teda zahodíme. Ešte sme ale nepoužili informáciu, koľko je vlastne lajstier. Táto nám dá potrebnú n-tú rovnicu: $x_1 + \cdots + x_n = 10^8$.

Na riešenie takejto sústavy rovníc slúži *Gaussova eliminačná metóda*. Jej použitím vypočítame presne stabilné rozdelenie lajstier v čase $O(n^3)$.

724. Táto úloha veľmi silne súvisí s Huffmanovým kódovaním (pozri úlohu 613). Postup jej riešenia je presne rovnaký ako zostrojovanie tohto kódu. Totiž platí, že ak v každom kroku zjednotíme dve v tej dobe najmenšie množiny, dostaneme optimálne riešenie.

Trikové riešenie bez použitia zložitých dátových štruktúr: Množiny na začiatku utriedime podľa veľkosti a vložíme ich do jednej fronty. Do druhej fronty budeme vkladať novovytvorené množiny. Všimnite si, že nové množiny sú v poradí, v akom ich vytvárame, usporiadané podľa veľkosti. V každom okamihu je teda najmenšia množina na začiatku jednej z našich dvoch front.

725. Vstupný reťazec vždy vieme upraviť do podoby $a_0{*}a_1{*}a_2{*}\cdots{*}a_i{*}\cdots{*}a_j$, kde všetky reťazce a_k obsahujú iba písmená abecedy a znaky ?. Reťazce a_0 a a_j majú v slove pevné miesto a ľahko overíme, či sa v názve súboru nachádzajú. Takže nám ostane riešiť prípad, keď hľadaný reťazec začína aj končí znakom $*$. Rozmyslite si, že stačí nájsť najľavejšiu časť názvu súboru, ktorá pasuje na reťazec a_1, za ňou najľavejšiu časť, ktorá pasuje na a_2, a tak ďalej. Existuje riešenie, ktorého časová zložitosť je lineárna od veľkosti vstupných údajov, teda súčtu dĺžky zadaného reťazca a dĺžok názvov súborov.

731. Backtracking. Je dobré využiť, že ak nie je súčet políčok patriacich miestnosti deliteľný štyrmi, nemá zmysel sa o nič pokúšať. Miestnosť môžeme začať parketovať hocikde a je výhodné použiť okolo miestnosti „plôtik“ z nulových políčok (zarážku), aby sme nemuseli zvlášť kontrolovať okraje. Je šikovné postupovať z horného ľavého rohu do pravého dolného po ešte voľných políčkach zľava doprava a zapamätať si všetkých 19 rôznych polôh tetramín, prípadne ich relatívne súradnice vzhľadom na ich ľavý horný roh.

732. Pre jednoduchosť si celý výraz vložíme do jednej zátvorky. Operátory a ľavé zátvorky je potrebné ukladať do zásobníka, operandy rovno vypisujeme na výstup. Ak je pri vkladaní operátora na vrchu zásobníka operátor s väčšou alebo rovnakou prioritou, zo zásobníka ho vyhodíme (výnimku tvorí len operátor mocniny, kde sa rovnaké operátory zo zásobníka nevytláčajú). Pri načítaní pravej zátvorky vypíšeme obsah zásobníka až po najbližšiu ľavú zátvorku a tú zo zásobníka vyhodíme. Špeciálne treba ošetriť unárne mínus.

733. Znázorníme si situáciu ako graf. Dôležité je uvedomiť si, že nemusí byť orientovaný, lebo ak má maniak a v zozname maniaka b, akonáhle mu zavolá, dozvie sa maniak b číslo maniaka a. Potom je už úloha jednoduchá: Prehľadávaním do šírky alebo do hĺbky nájdeme v našom grafe komponenty súvislosti.

734. Vyriešime každý typ cólštoku samostatne. Nech má cólštok prvého typu r ramien. Potom ak máme k dispozícii k kusov nejakého písmena, môžeme ho na každé rameno dať $\lfloor k/r \rfloor$-krát. Pre každý počet ramien takto spočítame, koľko najviac písmen môžeme dať na cólštok. Lepšie riešenie je skúšať za r iba prvočísla.

Cólštok druhého typu je vlastne to isté, ako cólštok prvého typu, len na horný aj na dolný koniec môžeme pridať po jednom písmene. Ak máme r ramien, horných písmen je $a = \lfloor r/2 \rfloor + 1$, dolných je $b = \lceil r/2 \rceil$. Podobne ako v predchádzajúcom prípade spočítame, koľkokrát môžeme každé písmeno použiť v strednej časti ramena. Roztrieďme si teraz písmená podľa toho, koľko ich zvýšilo. Na prvú kôpku dáme tie, ktorých zvýšilo aspoň a, na druhú tie, ktorých zvýšilo menej ako b, a ak $a \neq b$, tak na tretiu tie, ktorých zvýšilo práve b. Zjavne nezáleží na tom, ktoré konkrétne písmeno dáme na ktorý koniec, dôležité je len, na ktorej kôpke je. Takto dostávame najviac 9 možností, z ktorých vyberieme tú najlepšiu.

Poznámky: Ak na niektorý koniec vyberieme písmeno z druhej kôpky, znamená to vlastne, že ho do strednej časti dáme o jedno menej krát, ako sme pôvodne chceli. Netreba zabudnúť na možnosť, že na oba konce pôjde to isté písmeno. Pre cólštoky druhého typu už trik s prvočíselným počtom ramien priamočiaro použiť nejde.

735. Pre každú dvojicu bodov zistíme, či ich spojnica prechádza tučnou úsečkou. Rovnica priamky prechádzajúcej cez body $[x_1, y_1], [x_2, y_2]$ je $(y_2 - y_1)x + (x_1 - x_2)y + x_2y_1 - x_1y_2 = 0$. Ak do ľavej strany za x, y dosadíme dva body a dostaneme čísla s opačnými znamienkami, tak tieto dva body ležia v opačných polrovinách vzhľadom na túto priamku. Túto myšlienku môžeme využiť pri zisťovaní, či sa dve úsečky pretínajú. Je potrebné ošetriť pomerne veľa okrajových prípadov, ako napríklad body ležiace na tučnej úsečke alebo spojnica bodov, ktorá sa úsečky dotýka.

811. Ak je súčet k najmenších prvkov menší než t, tak `ANO`, inak `NIE`. Šikovné je nájsť k najmenších prvkov tak, že odhadneme, aké by mali byť veľké. Napríklad najprv skúsime sčítať k prvkov menších ako t/k. Ak sme nenašli žiadne, odpoveď je `NIE`, ak sme ich našli dosť, odpoveď je `ANO`. Ak sme ich našli $k_1 < k$ a ich súčet bol s_1, skúsime nájsť zvyšok, t.j. $k - k_1$ prvkov menších než $(t - s_1)/(k - k_1)$, ale väčších ako t/k, atď.

812. Ide o zovšeobecnenie úlohy z IMO 1977. Elegantné riešenie našiel na spomínanej IMO súťažiaci z Československa Martin Čadek. Uvedieme jeho riešenie pôvodnej úlohy pre $m = 7$ a $n = 11$. Označme členy hľadanej postupnosti x_i. Nech teraz s_i je súčet prvých i členov našej postupnosti. Podmienky zo zadania sa potom dajú prepísať nasledovne: $\forall k;\ s_{k+7} < s_k$ a $\forall k;\ s_{k+11} > s_k$. Ak by mala hľadaná postupnosť 17 a viac členov, muselo by platiť:

$$0 = s_0 < s_{11} < s_4 < s_{15} < s_8 < s_1 < s_{12} < s_5 < s_{16} < s_9 < \\ < s_2 < s_{13} < s_6 < s_{17} < s_{10} < s_3 < s_{14} < s_7 < s_0 = 0$$

a to zjavne nejde. Postupnosť so 16 členmi zostrojíme tak, že zvolíme ľubovoľné hodnoty s_i tak, aby platilo:

$$s_{10} < s_3 < s_{14} < s_7 < \ s_0 = 0 \ < s_{11} < s_4 < s_{15} < \\ < s_8 < s_1 < s_{12} < s_5 < s_{16} < s_9 < s_2 < s_{13} < s_6$$

Následne dopočítame jednotlivé hodnoty $x_i = s_i - s_{i-1}$.

Tento postup sa dá zovšeobecniť pre ľubovoľné m a n (dá sa pomocou neho dokázať, že hľadaná postupnosť má vždy dĺžku $m + n - 1 - \mathrm{nsd}(m, n)$), a navyše podľa tohto postupu ľahko napíšeme program, ktorý bude hľadanú postupnosť generovať – stačí postupne zvyšovať počet prvkov, až kým nedostaneme spor.

Uvedieme ešte jeden elegantný dôkaz, že ak sú m a n nesúdeliteľné, postupnosť musí byť kratšia ako $m + n - 1$. Sporom, nech táto postupnosť existuje. Uvažujme tabuľku s m riadkami a n stĺpcami, v ktorej sú napísané prvky našej postupnosti, pričom prvok v i-tom riadku a j-tom stĺpci je x_{i+j-1}. (V poslednom riadku a stĺpci je teda práve x_{m+n-1}.) Každý riadok tvorí n po sebe idúcich prvkov postupnosti, teda súčet každého riadku je kladný. Preto aj súčet celej tabuľky je kladný. Ale podobnou úvahou pre stĺpce dostávame, že súčet celej tabuľky musí byť záporný, čo je spor.

813. Táto úloha je klasická úloha z teórie grafov a nazýva sa úloha o najlacnejšej kostre. Existuje viacero efektívnych algoritmov, popíšeme najznámejšie z nich.

a) Kruskalov algoritmus. Utriedime podľa veľkosti vzdialenosti miest a postupne spájame najbližšie mestá tak, aby sme nevytvorili spojeniami cyklus. Na triedenie potrebujeme $O(n^2 \log n)$ a na kontrolu, či sme nevytvorili kružnicu $O(n^2 \log n)$ alebo $O(n^2 \log^* n)$ (pozri union-find, úlohu 942) operácií, teda prevláži triedenie.

b) Jarníkov-Primov algoritmus. Mestá rozdelíme na už spojené a ešte nespojené. Na začiatku je spojené iba jedno mesto (ľubovoľné, samo so sebou). Do skupiny spojených pridávame vždy to mesto, ktorého vzdialenosť od niektorého už spojeného mesta je minimálna. Pre každé pridané mesto musíme aktualizovať minimálne vzdialenosti do

ešte nespojených miest. Nájdenie najlacnejšieho a aktualizácia nás stojí $O(n)$ operácií a musíme prejsť všetky mestá, celkovo $O(n^2)$ operácií.

c) Borůvkov algoritmus. Ku každému mestu nájdeme to, ktoré je k nemu najbližšie. Naraz každú takúto dvojicu spojíme. Na spojené oblasti sa odteraz dívame ako na nové „veľké mestá". Celý postup opakujeme, až kým nepospájame všetko. (Vzdialenosť dvoch veľkých miest je samozrejme najkratšia zo vzdialeností miest, ktoré ich tvoria.) Priamočiara implementácia má časovú zložitosť $O(n^2 \log n)$, ide to ale naprogramovať aj šikovnejšie.

Podrobnejšie popisy týchto algoritmov sú napr. v [5.B v 38, 6.2 v 28, 23 v 6, 7.6 v 34, 5.1 v 1, 4.3 v 8].

814. Tento problém sa volá *topologické triedenie*.

a) Riešenie hrubou silou: Pre každého vojaka pomocou prehľadávania spočítame, koľko vojakov je pod ním v hierarchii. Ak u nejakého vojaka zistíme, že medzi jeho podriadenými je aj on sám, úloha nemá riešenie. Inak vypíšeme vojakov utriedených podľa počtu podriadených. (Rozmyslite si, že takéto poradie naozaj vyhovuje zadaniu!)

b) Pre každého vojaka si budeme počas celého výpočtu pamätať, koľko spomedzi jeho priamych podriadených sme ešte nevypísali. Zjavne v danom okamihu môžeme vypísať práve tých vojakov, pre ktorých si pamätáme nulu. Takýchto vojakov si teda budeme pamätať v samostatnej dátovej štruktúre (napr. fronte alebo zásobníku). Vždy, keď vypíšeme vojaka, prejdeme jeho nadriadených a každému zmenšíme pamätanú hodnotu. Ak sa niektorým z nich zmenšila na nulu, zaradíme ich medzi kandidátov na vypísanie. Takto postupujeme, pokým nevypíšeme všetkých vojakov alebo nezistíme, že úloha nemá riešenie.

c) Ešte šikovnejšie je použiť obyčajné prehľadávanie do hĺbky. Vojakov si znázorníme ako orientovaný graf s hranami od nadriadeného k podriadenému. Postupne prechádzame vrcholy grafu od 1 do N, pričom vždy, keď narazíme na ešte neofarbený vrchol, pustíme z neho prehľadávanie do hĺbky. Vždy, keď končíme spracúvanie nejakého vrcholu, vypíšeme jeho číslo. Platí, že ak má zadaná úloha riešenie, tak týmto postupom získaná postupnosť je riešením – netreba však zabudnúť na „skúšku správnosti".

815. Riešenie je priamočiare. Skontrolujeme, kto má viac celých boxov. Vhodné je použiť také zakódovanie farieb, aby sme ľahko vedeli zistiť, či sa nedoplnený box dá doplniť na jednofarebný. Napríklad biely bude 1, čierny bude 5 a nezafarbené políčko bude 0.

821. Je nešikovné najprv vyrátať celú krivku (prípadne ju uložiť do poľa) a potom vypísať len želanú časť. Celá krivka môže byť omnoho väčšia než hľadaná časť. Lepšie je pre každý bod (x, y) patriaci do skúmaného obdĺžnika určiť, či obsahuje $*$ alebo medzeru. Uvedomte si, že krivka stupňa n sa skladá zo štyroch kriviek stupňa $n-1$ a z troch $*$, ktoré ich spájajú. Stačí teda rekurzívne určovať, do ktorej zo štyroch kriviek menšieho stupňa padne bod (x, y), prípadne, či nepadne do oblastí medzi krivkami. Treba pritom dávať pozor, lebo, ako vidno zo zadania, krivky menšieho stupňa môžu byť otočené. Krivka stupňa 1 je $*$. Celý postup sa dá ešte zefektívniť, keď si všimneme, že body, ktorých obe súradnice sú párne, sú medzery a body s oboma súradnicami nepárnymi sú $*$. Skúste si dokázať prečo.

Iné riešenie: napíšeme si rekurzívnu procedúru na vykresľovanie celej Hilbertovej krivky, potom do nej prirobíme kontrolu, že ak je celá práve vykresľovaná časť krivky mimo, tak nič nekreslíme a ani nerobíme ďalšie rekurzívne volania.

822. Ukážeme si viacero rôzne efektívnych riešení.

a) $O(n^2)$; priamočiare riešenie je postupne kontrolovať body a zistiť, či všetky ostatné ležia v jednej polrovine.

b) $O(n \log n)$; body utriedime podľa *polárneho uhlu* – uhlu, ktorý zviera polpriamka z bodu $[0,0]$ do daného bodu s kladnou poloosou osi x. Ak je medzi niektorými dvoma po sebe idúcimi bodmi uhol väčší než 180°, hľadaná priamka existuje, inak nie.

c) $O(n)$; uhly z predchádzajúceho riešenia nemusíme triediť, stačí si napr. pamätať najmenší a najväčší uhol v každom kvadrante. Z tejto informácie už ľahko zistíme riešenie našej úlohy. (Zadanie síce hovorí, že bodu ležia na kružnici, toto riešenie však dokonca funguje pre ľubovoľne rozmiestnené body.)

d) Ak nie je žiaden bod nad osou x, alebo nie je žiaden bod pod ňou, máme riešenie. Inak vieme, že hľadaná polrovina musí byť „šikmá“. Keďže vieme, že v našom prípade body ležia na kružnici, vieme, že napr. keď body nad osou x usporiadame podľa x-ovej súradnice, budú usporiadané aj podľa uhlu. Nad aj pod osou x nás teda budú zaujímať len po dva body: ten s najväčšou a ten s najmenšou x-ovou súradnicou. Porovnaním týchto súradníc vieme vždy povedať, či hľadaná polrovina existuje. Rozmyslite si, ako na to.

823. Najskôr zrátame minimálne vzdialenosti medzi všetkými dvojicami miest. Jedna z možností je spustiť Dijkstrov algoritmus (pozri úlohu 523) z každého vrcholu. Jednoduchšie je však použiť dynamické programovanie – algoritmus Floyda a Warshalla: Nech $d^k_{i,j}$ je dĺžka najkratšej cesty z i do j, ak cesta prechádza iba cez vrcholy $\{1, 2, \ldots, k\}$ (ak neexistuje žiadna taká cesta, dĺžka je ∞). Pre $k = 0$ je táto množina prázdna – uvažujeme teda iba hrany medzi vrcholmi, naopak, pre $k = n$ uvažujeme všetky vrcholy – túto hodnotu chceme vypočítať. Keď chceme vypočítať $d^k_{i,j}$, sú dve možnosti: buď najkratšia cesta neprechádza cez vrchol k (teda prechádza iba cez vrcholy $\{1, 2, \ldots, k-1\}$) – vtedy je jej dĺžka rovná $d^{k-1}_{i,j}$; alebo cez vrchol k prechádza. Keďže hľadáme najkratšiu cestu (a v grafe nie je cyklus zápornej dĺžky), cez k prejdeme *iba* raz. Preto dĺžka najkratšej cesty medzi i a j je súčet dĺžok najkratších ciest medzi i, k a k, j. Tieto cesty už však neprechádzajú cez k, teda majú dĺžku $d^{k-1}_{i,k}$ a $d^{k-1}_{k,j}$. Dĺžky ciest teda počítame postupne podľa vzorca $d^k_{i,j} = \min\{d^{k-1}_{i,j}, d^{k-1}_{i,k} + d^{k-1}_{k,j}\}$. Oba postupy vyžadujú $O(n^3)$ operácií a stačí $O(n^2)$ pamäte.

Teraz stačí zrátať pre každé mesto súčet vzdialeností do ostatných a vybrať mesto, kde je tento súčet minimálny. Takže celkovo potrebujeme na realizáciu $O(n^3)$ operácií.

Podrobnejší popis Floydovho-Warshallovho algoritmu nájdete napríklad v [38, 8].

824. Je dobré začať baliť od najvyšších hmotností. Osemkilové výrobky sú jasné. Ku každému sedemkilovému môžeme (ak máme) pribaliť kilový. Ku každému šesťkilovému môžeme pribaliť jeden dvojkilový, alebo ak už dvojkilové nemáme, tak aj (najviac) dva kilové. Podobne to funguje aj pre päťkilové výrobky. Štvorkilové najskôr balíme po dvojiciach. Ak nám jeden zvýši, sú dve možnosti: buď s ním pôjde do balíka jeden trojkilový (a ak máme, tak aj jeden kilový), alebo nie – teda zvyšné miesto doplníme dvojkilovými a kilovými výrobkami. Vyskúšame obe možnosti. V oboch prípadoch už malé výrobky ľahko dobalíme. Vyberieme si lepšie z týchto dvoch riešení. (Všimnite si napríklad vstupy „4kg, 2×3kg, 3×2kg“ a „4kg, 5×3kg, 2×2kg“. V prvom prípade je výhodnejšie pribaliť k štvorkilovému výrobku dva dvojkilové, v druhom je to naopak.)

825. Pozor na mnoho okrajových podmienok. V prípade, že nebol vytvorený box, je situácia korektná, ak sa počet ťahov bieleho a čierneho líši najviac o jedna. Ak boli ale vytvorené boxy, podľa pravidiel ten, kto svojou čiarou vytvoril box, dostal ťah navyše. Ale pozor, ak sa mu podarilo vytvoriť jednou čiarou boxy dva, rovnako dostal iba jeden extra ťah. Podľa pravidiel hra končí aj vtedy, keď sa už nedá vytvoriť žiaden box. Ak v takejto situácii vykonáme ťah, dostaneme nekorektnú situáciu. No a teraz už neostáva nič iné, iba metódou spätných návratov postupne generovať jednotlivé priebehy hry.

831. Krajina tvorí graf, izolované časti sú tzv. *komponenty súvislosti*. Vieme ich nájsť jednoduchou úpravou prehľadávania do šírky alebo do hĺbky: Postupne kontrolujeme všet-

ky vrcholy a vždy, keď nájdeme neofarbený, zvýšime počet komponentov a spustíme z tohto vrcholu prehľadávanie, ktoré celý komponent ofarbí. Algoritmus má optimálnu časovú aj pamäťovú zložitosť $O(m+n)$.

832. Zjavne obdĺžnik musí byť z každej strany ohraničený dierou alebo okrajom dosky, inak by sa dal zväčšiť. Najjednoduchšie a najmenej šikovné je pre každý okraj vyskúšať všetky možnosti, a pre každý obdĺžnik, ktorý takto dostaneme, skontrolovať, či neobsahuje ďalšie diery. Toto riešenie vyžaduje $O(k^5)$ operácií.

Lepšie riešenie vyžaduje len $O(k^2)$ operácií. Najprv utriedime diery podľa rastúcej x-ovej súradnice. Postupne pre každú dieru v tomto poradí skontrolujeme všetky nezväčšiteľné obdĺžniky, ktoré majú na tejto diere ľavú stranu. Vtip je v tom, že takýchto obdĺžnikov je maximálne k. Na začiatku majme obdĺžnik šírky 0, nastavme jeho hornú a dolnú stranu (h a d) na horný a dolný okraj dosky. Teraz postupne spracúvajme ostatné diery smerom doprava. Náš obdĺžnik vždy natiahneme po x-ovú súradnicu nasledujúcej diery, a následne (ak treba) upravíme h alebo d tak, aby pri ďalšom zväčšovaní náš obdĺžnik túto dieru neobsahoval. Samozrejme, z obdĺžnikov, ktoré vyššie popísaným postupom vyrobíme, vyberieme ten najväčší.

Aby sme nemuseli zvlášť ošetrovať ľavý okraj, stačí na začiatku pridať $k+1$ nových dier, ktoré ležia na ľavom okraji: prvá úplne hore, posledná úplne dole, ostatným dáme y-ové súradnice také, aby sa po zoradení podľa y-ovej súradnice striedali pôvodné a nové diery. Takto vlastne každá nová diera zodpovedá jednému vodorovnému pásu medzi starými dierami.

V súčasnosti asi najlepší známy algoritmus riešiaci túto úlohu vymysleli Aggarwal a Suri v roku 1987. Jeho časová zložitosť je $O(k \log^2 k)$.

833. Vtipné riešenie je použiť jednorozmerné pomocné pole veľkosti mn. Pre každé písmeno návrhu A si doň uložíme v klesajúcom poradí indexy jeho výskytov v návrhu B. Potom nájdeme v tomto poli najdlhšiu rastúcu podpostupnosť (pozri úlohu 522) – to budú indexy tých písmen návrhu B, ktoré tvoria hľadanú spoločnú podpostupnosť.

„Klasické“ riešenie: Označme $L_{i,j}$ dĺžku najdlhšieho spoločného návrhu z častí $A[1\,..\,i]$ a $B[1\,..\,j]$. Chceme nájsť $L_{m,n}$. Ako určiť hodnoty $L_{i,j}$? Pre $0 \le i \le n$ a $0 \le j \le m$ je $L_{i,0} = L_{0,j} = 0$. $L_{i,j}$ určíme na základe $L_{i-1,j}$, $L_{i,j-1}$ a $L_{i-1,j-1}$: Ak $A[i] = B[j]$, tak $L_{i,j} = L_{i-1,j-1} + 1$; v opačnom prípade je $L_{i,j} = \max(L_{i-1,j}, L_{i,j-1})$. Znaky najdlhšieho spoločného návrhu určíme od posledného tak, že prezeráme hodnoty poľa L a všímame si, kde hodnota L klesá.

834. Vstup môžeme reprezentovať bipartitným grafom. Jednu partíciu tvoria možné mená, druhú prápory, hrana znamená, že v danom prápore je (aspoň jeden) vojak s daným menom. Úlohou je zistiť, či v tomto grafe existuje úplné párovanie. Efektívny algoritmus riešiaci túto úlohu sa dá nájsť napríklad v [8 v 38, 9 v 8].

835. Zostrojme nový graf G, ktorého vrcholy budú usporiadané dvojice štátov. Zmluvy budú zodpovedať hranám v tomto grafe. Vznik vojny zodpovedá výberu vrcholu v v G. Vojny nastanú medzi tými dvojicami štátov, ktorých vrcholy sú dosiahnuteľné z v.

Môžeme postupne skúšať za v zvoliť všetky vrcholy G a zakaždým prehľadávaním do šírky zistiť všetky bojujúce dvojice štátov. Nakoniec vždy overíme, či bojuje celý svet alebo nie. Toto riešenie má časovú zložitosť $O(N^4 + N^2K)$.

Efektívnejšie riešenie: V grafe G nájdeme silne súvislé komponenty. Toto vieme v čase $O(N^2 + K)$. Každý z nich nahradíme jedným vrcholom obsahujúcim množinu „zúčastnených“ štátov. Toto ide v $O(N^3)$. Takto sme dostali acyklický graf. Spracujeme jeho vrcholy v topologickom usporiadaní. Pre každý z nich spočítame zjednotenie „jeho“ množiny štátov a množín pre vrcholy, do ktorých z neho vedie hrana. (Takto sme dostali množinu štátov, ktoré by boli vo vojne, ak by sa tá začala v danom silne súvislom komponente.) Túto fázu

vieme spraviť v čase $O(NK)$ – pre každú hranu robíme jedno zjednotenie. Celková časová zložitosť tohto riešenia je teda $O(N^3 + NK)$.

841. Zrejme trasa bude prechádzať všetkými stanovišťami, lebo inak by sme ju vedeli predĺžiť. Z rovnakého dôvodu bude cieľ na tom istom stanovišti ako štart. Všimnite si, že najvzdialenejšie stanoviština nemusia nasledovať po sebe na najdlhšej trase. (Napr. ak máme štyri stanoviština na súradniciach $[-100, 0]$, $[100, 0]$, $[0, 100]$ a $[0, 90]$.) Problém je NP-ťažký. Nie je nám známe efektívnejšie riešenie ako postupné prezretie všetkých možných trás, ktorých je $n!$.

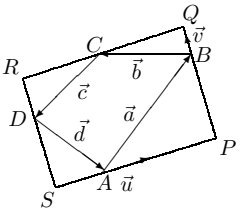
842. Minimálny počet presunov je $2^{n+2} - 5$. Najprv vyriešte jednoduchšiu úlohu, keď sú rovnaké disky nerozlíšiteľné. Vtedy treba $2^{n+1} - 2$ presunov (je to dvakrát viac ako v originálnom hlavolame ale len s n diskami). Uvedomte si, že keď presuniete vežu s nerozlíšiteľnými rovnakými diskami, tieto sú v opačnom poradí. Takže po dvoch presunutiach sú opäť v pôvodnom usporiadaní.

843. Vzhľadom na to, že body máme zadané proti smeru chodu hodinových ručičiek, môžeme použiť napríklad časť Grahamovho algoritmu na vylúčenie bodov, ktoré do konvexného obalu nepatria, pozri úlohu 413.

844. Na úvod treba určiť jazyk, v ktorom je zadaný vstup. Najjednoduchšie je asi vyhľadať v ňom číslovky od 0 do 9 v oboch jazykoch.

Pre angličtinu začnite vyhľadaním slova `thousand`. Ak sa tam nevyskytuje, máme číslo menšie ako tisíc, ak áno, rozdelí nám vstup na dve časti a každá z nich je opäť číslo menšie ako tisíc, spracujeme každú z nich samostatne. Pokračujeme podobne, hľadaním výskytu slova `hundred`.

Niektoré ďalšie časti tejto úlohy boli zadané ako úlohy 521 a 531, pozri aj ich riešenia.



845. Riešenie od Ladislava Kisa. Obdĺžnik s minimálnym obsahom sa dá nájsť, len treba použiť trochu matematiky. Každý opísaný štvoruholník, ktorý je kandidátom na minimálny, sa iste bude dotýkať na každej svojej strane aspoň jedného z daných vrcholov (inak by sa dal zmenšiť). Pre jednoduchosť si predstavme, že máme zadané iba štyri vrcholy A, B, C, D. Zrátame obsah S pravouholníka $PQRS$ v závislosti od vektora $\vec{u}$.

Platí, že $S = |SP| \cdot |PQ|$, kde $|SP| = |AS| + |AP|$ a $|AP| = (\vec{a} \cdot \vec{u})/\|\vec{u}\|$ a analogicky aj $|AS|$, $|BQ|$ a $|BP|$. Prípad $\vec{u} = (0, z)$ skontrolujeme osobitne. Keď položíme $\vec{u} = (1, z)$, $\vec{r} = \vec{a} + \vec{d} = (r_1, r_2)$ a $\vec{s} = \vec{a} + \vec{b} = (s_1, s_2)$, dosadíme do vzťahu pre S a roznásobíme, dostaneme $S(z)$ vzhľadom na parametre $\vec{s}$ a $\vec{r}$, ktoré sú však konštantné. Zaujíma nás, kde má $S(z)$ maximum. Zrátame $S'(z) = 0$ a dostaneme $z_1 = -q + \sqrt{q^2 + 1}$ a $z_2 = -q - \sqrt{q^2 + 1}$, kde $q = (r_1 s_2 + r_2 s_1)/(r_2 s_2 - r_1 s_1)$. Dve riešenia z_1 a z_2 predstavujú navzájom kolmé vektory a v jednom z nich nadobúda $S(z)$ maximum.

Ako toto všetko využijeme, keď je bodov viac než 4? Zoberme niektorý vrchol v_i, nech leží na jednej zo strán nejakého opísaného obdĺžnika. Je zrejmé, že zvyšné tri strany k nemu vieme ľahko určiť, keď si vyberieme smer tejto strany. Ten nemôže byť ľubovoľný, ale iba z intervalu, ktorý určujú strany k susedným vrcholom v_{i-1} a v_{i+1}. V obidvoch krajných prípadoch sú naše štyri body známe. Interval určený týmito krajnými polohami môže byť ešte rozdelený na podintervaly, kde v každom sú už iba tie isté štyri vrcholy ležiace na stranách opísaného obdĺžnika (intervaly sa dajú nájsť na $O(n)$ operácií). Teraz môžeme použiť metódu pre štyri body a zrátať, ako by mala byť strana otočená, aby bol opísaný obdĺžnik minimálny, ak vyjde smer mimo aktuálny interval, vďaka tomu, že funkcia je medzi oboma extrémami monotónna, stačí preveriť hraničné smery intervalu a zobrať menší obdĺžnik. Toto musíme urobiť pre všetky intervaly.

911. Uvedomte si, že $2^m - 1 = 2^0 + 2^1 + \cdots + 2^{m-1}$. Teda v prípade $m = n$ je to ľahké. Ak $m < n$, treba zobrať vhodný násobok $2^m - 1$ a, ak treba, niektoré sčítance rozpísať podľa vzťahu $2^{x+1} = 2^x + 2^x$, aby sme dostali práve n sčítancov. Najelegantnejšie je asi zobrať násobok $2^{n-m}(2^m - 1) = 2^{n-m} + 2^{n-m+1} + \cdots + 2^{n-1}$ a rozpísať prvý sčítanec 2^{n-m}. Dostaneme tak práve n sčítancov: $\underbrace{2^0 + (2^0 + 2^1 + \cdots + 2^{n-m-1})}_{=2^{n-m}} + 2^{n-m+1} + \cdots + 2^{n-1}$.

912. K tejto úlohe sa dá pristupovať rôznymi spôsobmi:

a) Prehľadávanie do šírky. Každý rok obalíme každý ostrov novou vrstvou políčok. (Vrstvu budú tvoriť tie políčka mora, ktoré práve susedia stranou s nejakým políčkom ostrova.) Jednotlivé vrstvy značíme číslom ostrova, ku ktorému patria, a číslom roka, v ktorom sme ich vytvorili. Prestaneme po roku, v ktorom sa nám dva zväčšované ostrovy pretnú. Vyberieme ten priesečník, kde je súčet čísiel vrstiev najmenší. Aby sme vyznačili samotný most, stačí sa z miesta priesečníka pohybovať „po vrstvách“, až kým neprídeme na pôvodný ostrov. Aby bolo toto riešenie efektívne, musíte pri obaľovaní prezerať len políčka pobrežia, nie celú mapu. Políčka pobrežia si ukladajte do fronty.

b) Vždy existuje najkratší most, ktorý má najviac jednu zákrutu. (Totiž ľubovoľný najkratší most vieme upraviť na takýto. Ak by sa to nedalo, mohla nám prekážať iba pevnina, ale to je spor s tým, že most je najkratší.) Tento poznatok sa dá využiť tak, že pre každé políčko mora spočítame jeho vzdialenosť od pevniny na severe, juhu, východe a západe. Z toho už ľahko určíme najkratší most. Zamyslite sa, ako tieto vzdialenosti spočítať efektívne.

c) Predchádzajúce riešenie sa dá ešte trochu zlepšiť. Všimnime si, že stačí uvažovať mosty troch typov: „vodorovný“, „najskôr dole a potom doľava“ a „najskôr dole a potom doprava“. Vstup budeme spracúvať po riadkoch, pre každé políčko budeme počítať vzdialenosť od pevniny na severe, východe a západe (a samozrejme aj to, o ktorý ostrov ide). Toto riešenie sa dá naprogramovať tak, aby bežalo v čase $O(mn)$ a používalo len $O(m + n)$ pamäte.

913. Plocha plechu P sa rovná $\sum_{i=1}^{k} x_i y_i$. Pozor, nestačí určiť m a n také, že $P = mn$, ale treba aj overiť, či sa na takomto plechu koláčiky mohli piecť. Kontrola rozloženia koláčikov na plechu sa dá vykonať postupným prezretím všetkých prípustných rozložení (backtracking). Pri rozkladaní si stačí pamätať iba obrys už rozložených koláčikov.

914. Z každej dvojice po sebe idúcich slov sa dozvieme jednu informáciu o usporiadaní abecedy. Napr. ak `ghrb` je v abecede pred `ghcgg`, znamená to, že `r` musí byť pred `c`. Ak si tieto informácie znázorníme v orientovanom grafe, úlohou je nájsť jedno jeho topologické utriedenie. Pozri riešenie úlohy 814.

915. Robiť úplnú syntaktickú analýzu je zbytočné, pretože program na vstupe je syntakticky správny. Lepšie je realizovať iba príkazy *NEXT*, *PUT*, *GET* procedúrami a premenné *TOP* a *INP* funkciami. Ak sa uspokojíme s Turbo Pascalom, môžeme ich umiestniť do unitu. Všeobecnejšie riešenie ich vsunie medzi hlavičku programu a prvý **begin**. Fronta sa dá realizovať poľom pevnej dĺžky alebo spájaným zoznamom. Zamyslite sa nad výhodami a nevýhodami takýchto implementácií.

921. Niekoľko možných riešení:

a) Prvky poľa usporiadame nerastúco; zložitosť závisí od použitého algoritmu triedenia (najlepšie však $O(n \log n)$).

b) Na pole, ktoré spĺňa podmienku zo zadania, sa môžeme dívať ako na k-árnu haldu. (To je halda, kde každý vrchol má najviac k synov. Presvedčte sa, že pre $k = 2$ dostávame klasickú haldu.) Ak prvky od k po n „bubleme nahor“, potrebujeme zhruba $(n - k) \log_k n$ operácií.

c) Lepšie je postupne od $n/k+1$ po 1 „bublať prvky nadol“ – t.j. vždy vyberieme z k synov najväčšieho a vymieňame ho za otca, kým neplatí nerovnosť, alebo kým sa nedostaneme k listom; takéto riešenie vyžaduje

$$\sum_{i=0}^{\infty} \frac{n}{k^i} \cdot ki = n \cdot \sum_{i=0}^{\infty} \frac{i}{k^{i-1}} = n \cdot O(1) = O(n)$$

operácií (keďže najviac n/k^i vrcholov má výšku i v kompletnom k-árnom strome s n vrcholmi a na jedno prebublanie z výšky i potrebujeme zhruba ki operácií).

922. Prejdeme postupne celú mapu a kontrolujeme každé políčko pevniny, či nesusedí s morom.

Inou možnosťou je obísť ostrov po mori alebo po pobreží. Pozor na ostrov, ktorý má na mape políčka pevniny susediace navzájom iba rohom. Je dobré okolo mapy použiť zarážku pozostávajúcu z políčok predstavujúcich more, aby sme nemuseli testovať, či sme nevyšli z mapy (ostrov môže byť až po hranice mapy). Ak by sme už mapu mali načítanú v pamäti a vedeli súradnice aspoň jedného políčka ostrova, bolo by toto riešenie efektívnejšie ako to predchádzajúce.

923. Riešenie „hrubou silou“: Prejdeme celú trasu, počítame si Bantove súradnice. Zapamätáme si najmenšiu aj najväčšiu x-ovú aj y-ovú súradnicu. Teraz alokujeme pole zodpovedajúcej veľkosti, „zakreslíme“ doň Bantovu cestu a spočítame políčka vo vnútri.

Lepšie riešenie: Nakreslime vodorovnú priamku cez každý bod, v ktorom Banto menil smer. Takto sa nám nakreslený obrázok rozpadne na niekoľko pásov. Pozrime si ľubovoľný z nich, označme jeho výšku δy. Nie je ťažké prísť na to, že ak si pozrieme všetky Bantove prechody cez tento pás „zoradené zľava doprava“, idú striedavo dohora a dodola. Pritom do vyznačenej parcely patria časti pásu medzi každým „nepárnym“ a „párnym“ prechodom. Ak máme Bantov prechod na súradnici x, potom $\delta y \times x$ je plocha obdĺžnika od osi x po daný prechod. Rozmyslite si, že celkovú plochu parcely pre tento pás vieme spočítať tak, že sčítame plochy obdĺžnikov pre „párne“ prechody a od toho odčítame plochy obdĺžnikov pre „nepárne“ prechody.

Trik je v tom, že delenie na pásy vôbec nepotrebujeme robiť. Stačí postupne spracovať všetky Bantove pohyby v smere osi y. Za každý, ktorý ide dohora, pripočítame k celkovej ploche plochu jemu zodpovedajúceho obdĺžnika, za každý, ktorý ide dodola, plochu jemu zodpovedajúceho obdĺžnika odčítame.

Posledné detaily: Ak Banto obchádzal parcelu v smere hodinových ručičiek, vyjde nám veľkosť správna, ale záporná. Rozmyslite si, že tento postup funguje aj pre záporné súradnice vrcholov mnohouholníka.

924. Pozri 813.

925. Stručne popíšeme niekoľko rôzne efektívnych riešení:

a) $O(m^2n^2(m+n))$; priamočiare riešenie, skontrolujeme všetky obdĺžniky tak, že pre všetky prípustné pravé horné rohy vyberieme všetky ľavé dolné rohy a preveríme, či ich obvod je nakreslený.

b) $O(m^2n^2)$; zlepšenie dosiahneme za cenu pomocnej pamäti, v ktorej si budeme pamätať pre každé pole hracej dosky, aká súvislá čiara je z neho nakreslená smerom hore a vľavo, čo vieme zistiť pri načítaní; tak ušetríme kontrolu obvodu obdĺžnikov.

c) $O(nm^2)$ – riešenie od Maťa Ondrušku: Pre každý riadok pre každú dvojicu stĺpcov si zapamätáme hodnotu $s[i, j_1, j_2]$, či sú spojené body $[i, j_1]$ a $[i, j_2]$. Teraz sa pozrime na dva stĺpce (nech sú to j a k) – počet obdĺžnikov, ktoré majú ľavú stranu v j-tom a pravú stranu v k-tom stĺpci vieme spočítať v lineárnom čase. (Stačí ísť po oboch stĺpcoch a kontrolovať, kedy sa prerušia – ak je v nejakom neprerušenom úseku p priečok, je tam $\binom{p}{2} = p(p-1)/2$ obdĺžnikov.) Algoritmus sa dá ešte zlepšiť, aby nepotreboval

toľko pamäte – pôjdeme postupne po riadkoch a počítame pre každé dva stĺpce počet „priečok“; keď sa nejaké dva stĺpce prerušia, pripočítame počet obdĺžnikov a počet priečok vynulujeme (hodnoty $s[i, j_1, j_2]$ stačí počítať za behu).

931. Riešenie je náročné, pozri [43].

Myšlienka nie síce najefektívnejšieho, ale predsa len polynomiálneho riešenia: Oblasť, kam sa vieme dostať koncom nanajvýš k-krát zalomenej rúry, je pre každé k mnohouholník, ležiaci v páse. Postupne s rastúcim k budeme zostrojovať tieto mnohouholníky.

Keď už máme zostrojenú oblasť pre nejaké k, oblasť pre $k+1$ vieme zostrojiť nasledovne: Rúra s $k+1$ zlomami má svoj posledný zlom niekde v práve zostrojenej oblasti, a odtiaľ ide rovno. Predstavme si, že do každého bodu práve zostrojenej oblasti umiestnime malinkú žiarovku. Potom nová oblasť je tvorená práve miestami, kam tieto žiarovky (vnútrom pásu) dosvietia.

Zjavne stačí uvažovať žiarovky na obvode oblasti pre k. Tých je síce ešte stále nekonečne veľa, ale aj s tým si poradíme.

Pre jednoduchosť predpokladajme, že žiadne tri vrcholy pásu neležia na priamke. Všimnime si konkrétnu žiarovku a zapíšme si poradie, v akom okolo nej (polárne utriedené) ležia vrcholy obvodu pásu. Posúvajme sa teraz po obvode oblasti a sledujme, kedy sa táto informácia zmení. Zjavne jediná možná zmena je, že si dva dovtedy „susedné“ vrcholy vymenia miesto. Toto môže nastať len keď práve prechádzame cez priamku idúcu cez tieto dva vrcholy (a nejdeme pomedzi ne). Cyklické poradie vrcholov sa teda mení len na konečne veľa miestach. Presnejšie, na polynomiálne veľa miestach.

Tieto význačné miesta, kde nastáva nejaká zmena, nám teda rozdelia obvod našej oblasti na polynomiálne veľa úsečiek. Každú úsečku budeme riešiť samostatne. Začnime tým, že vypočítame oblasť, ktorú osvetlí žiarovka umiestnená (takmer) v jednom koncovom bode. Keď teraz budeme žiarovku posúvať po úsečke, budú sa len niektoré vrcholy osvetlenej oblasti posúvať, tiež po úsečkách. Stačí teda vedieť obe „koncové“ oblasti, a z nich už vieme spočítať celú plochu osvetlenú touto úsečkou. Takto dostaneme pre každú úsečku jeden mnohouholník, všetky ich zjednotíme a máme zostrojenú ďalšiu oblasť.

932. Skúste viac metód a vyberte najvhodnejšiu pre daný vstup. Jednoduché prístupy: Použite bity na uloženie hodnoty 1 a 0; zapamätajte si len pozície 1; uložte iba počty po sebe idúcich 1 a 0 (vyberte najvhodnejší smer); uložte si iba počty po sebe idúcich 0 a 1 „skopírujte“.

Existujú aj zložitejšie ale účinnejšie postupy, vrátane tých, ktoré používajú bežné kompresné programy. Za zmienku stojí Huffmanovo kódovanie (pozri úlohu 613) vstupu rozdeleného na k-tice znakov.

933. Riešenie nemajú len tie tabuľky, ktoré majú práve jeden riadok a v ňom čísla 132, 213 alebo 321.

Rozmyslite si, ako poskladať ľubovoľnú dvojriadkovú tabuľku. Viacriadkové tabuľky potom skladáme po riadkoch odhora, až kým neostanú už len dva riadky. Skúste vymyslieť všeobecný postup skladania, pri ktorom navyše nikdy nepotrebujete rotovať riadok, kde je práve diera.

934. Prečo vojna vždy skončí, a dokonca rovnako? Každým odrezaním vlastne z hranice len odstránime niekoľko zlomových bodov. A toto môžeme spraviť len konečne veľakrát, pričom na konci dostaneme spojnicu začiatku a konca pôvodnej hranice.

Vyskúšanie všetkých možností, ako by mohla vojna prebiehať, má časovú zložitosť $O(N^3 N!)$ – vyskúšame všetky dvojice bodov, pre každú skontrolujeme, či môžeme prislúchajúci úsek odrezať, a ak áno, odrežeme ho a rekurzívne vyriešime vzniknutú situáciu.

Za cenu veľkých pamäťových nárokov vieme toto riešenie trochu zrýchliť. Všimnime si, že aktuálna situácia je jednoznačne určená množinou pôvodných zlomových bodov, ktoré ešte stále ležia na hranici. Takýchto množín je $O(2^N)$. Postupne pre každú množinu bodov

spočítame optimálne riešenie, takto dostaneme riešenie s pamäťovou zložitosťou $O(2^N)$ a časovou $O(N^3 2^N)$. Nie je nám známe efektívne riešenie tejto úlohy.

935. Označme si dané konvexné obaly (KO) A a B.

a) Ak využijeme to, že v hornej časti a dolnej časti KO sú body usporiadané podľa x-ovej súradnice, môžeme použiť Grahamov algoritmus (je popísaný v riešení úlohy 413). Dostávame riešenie v lineárnom čase (keďže zotriediť všetky body vieme v lineárnom čase).

Priamku, ktorá prechádza niektorým vrcholom KO budeme volať *dotyčnicou*, ak všetky body KO ležia od priamky len na jednej strane (priamka KO nepretína). Iné riešenia môžu využiť fakt, že stačí nájsť dve spoločné dotyčnice ku A a B, ktoré pridáme do výsledného KO a odstránime body medzi nimi. Lineárne riešenie môžeme dostať úpravou Jarvisovho algoritmu (pozri 413).

b) Ak budeme navyše predpokladať, že A je vľavo od B (t.j. najpravejší bod A má menšiu x-ovú súradnicu ako najľavejší bod B), vieme nájsť spoločné dotyčnice oveľa ľahšie. Ukážeme si, ako nájdeme spodnú dotyčnicu v lineárnom čase (pozor, nemusí to byť spojnica najnižšieho bodu A a B!):

$a :=$ najpravejší bod A;
$b :=$ najľavejší bod B;
while priamka ab nie je spoločná dotyčnica A a B **do begin**
  while priamka ab nie je dotyčnica ku A **do**
    $a :=$ nasledujúci bod A v smere hodinových ručičiek;
  while priamka ab nie je dotyčnica ku B **do**
    $b :=$ nasledujúci bod B proti smeru hodinových ručičiek;
  end;

Na zistenie, či je priamka prechádzajúca bodom q dotyčnica ku KO, stačí otestovať, či dvaja susedia bodu q v KO ležia vzhľadom na túto priamku na jednej strane. Hornú spoločnú dotyčnicu nájdeme podobne.

Tento algoritmus sa dá využiť na hľadanie KO: Body utriedime podľa x-ovej súradnice a použijeme metódou *rozdeľuj a panuj*: rekurzívne zostrojíme konvexný obal prvej a druhej polovice bodov a tieto KO spojíme. Ide o tzv. *MergeHull* algoritmus (dvojrozmerná analógia MergeSort-u); má časovú zložitosť $O(n \log n)$.

941. Aby sa tanec podaril musí byť tanečníkov párny počet a viac než 7. Vtedy stačí uberať z niektorého kraja dvoch, až kým nám nezostane osem tanečníkov. Pre tých už je postup uvedený v zadaní.

942. Problém union-find. Z každej skupiny krajín, ktoré sú medzi sebou nejako spojené sľubom spolupatričnosti si vyberieme jedného bezvýznamného reprezentanta (alebo kratšie: šéfa). Krajiny sú teraz spojené vtedy, keď majú rovnakého šéfa.

Najjednoduchšie je pamätať si v poli pre krajinu i jej šéfa (na začiatku nie je nikto spojený – každý je sám sebe šéfom). Keď chceme spojiť krajiny x a y, stačí pole prejsť a všetky hodnoty šéf(y) prepísať na šéf(x). V najhoršom prípade budeme na spracovanie k dvojíc krajín potrebovať $O(kn)$ operácií.

Lepšie je pamätať si spájaný zozname krajín, ktoré sú spojené (t.j. každá krajina si pamätá svojho nasledovníka v zozname a navyše svojho šéfa). Keď teraz spájame dve územia, nemusíme príslušné krajiny hľadať – stačí len príslušné zoznamy spojiť a prepísať číslo šéfa pripojeným krajinám. Aby toto bolo vylepšenie, treba si pamätať ešte dĺžku zoznamov a pripájať vždy kratší zoznam k dlhšiemu. Koľkokrát sa potom môže jednej krajine zmeniť šéf? (Aká veľká bude skupina po spojení?) Na k spojení treba $O(k + n \log n)$ operácií.

Ešte lepšie riešenie sa dá nájsť v [3.8 v 28, 4.6–8 v 1, 4.5 v 34, 21 v 6, 10 v 27].

943. Metóda rozdeľuj a panuj: Utriedime si brlôžky napríklad podľa x-ových súradníc. Rozdeľme ich na ľavú a pravú polovicu. V každej polovici zistime najmenšiu vzdialenosť dvojice brlôžkov. Minimum z týchto dvoch čísel nech je m. Potom buď m je vzdialenosť dvoch najbližších brlôžkov, alebo je najbližšia dvojica rozdelená, jeden brlôžok je v ľavej, druhý v pravej polovici. Majme v obidvoch poloviciach utriedené brlôžky podľa y-ovej súradnice. Teraz nám stačí pre každý brloh z ľavej polovice, ktorý je od deliacej čiary vzdialený menej ako m nájsť najbližší brloh z pravej polovice. Stačí sa zaujímať iba o brlohy, ktoré sú bližšie ako m, a tých bude pre každý ľavý brloh max. 4. Preto to vieme spraviť v lineárnom čase. Ešte k triedeniu podľa y: hlavná rekurzívna funkcia, počítajúca z miním oboch polovíc minimum celku, môže mať podobný vedľajší efekt ako triedenie zlučovaním (MergeSort): spojí ľavú a pravú polovicu, ktoré už boli utriedené podľa y do jedného utriedeného poľa. Tak budeme mať tento úsek poľa utriedený pre ďalšie spájanie.

944. Stačí testovať len vzdialenosti v každom ľavom ostrom rohu (takom, ktorý vyčnieva do chodby) od protiľahlej zvislej a protiľahlej vodorovnej steny a od všetkých ostrých rohov pravej steny, ležiacich medzi týmito stenami. Ako ostré rohy treba uvažovať aj koniec a začiatok steny.

945. Body štvorky, v ktorých sa vodorovná palička napája na zvislé, nazvime koleno (ľavý bod) a uzol (pravý bod). Počet všetkých štvoriek, ktoré majú spoločné koleno a uzol je zrejme $h_{k,y} \times h_{u,y} \times d_{u,y}$, kde $h_{k,y}$ a $h_{u,y}$ označujú dĺžky paličiek nad daným kolenom a uzlom a $d_{u,y}$ je dĺžka paličky pod uzlom (prirodzene, za predpokladu, že koleno a uzol sú spojené vodorovnou čiarou). Počet štvoriek, ktoré majú spoločný uzol (môžu mať rôzne kolená) je suma $\sum_k h_{k,y} \times h_{u,y} \times d_{u,y} = h_{u,y} \times d_{u,y} \times \sum_k h_{k,y}$ cez všetky kolená prislúchajúce danému uzlu. Skúsme na to ísť „dynamicky“, postupne. Pre každý bod vieme ľahko vypočítať $h_{x,y}$: $h_{x,y} = 0$, ak nad bodom nie je zvislá čiara, v opačnom prípade je $h_{x,y} = h_{x,y-1} + 1$. Tiež vieme ľahko pre každý bod (ako potenciálny uzol) vypočítať $s_{x,y} = \sum_k h_{k,y}$ pre tie body $[k, y]$, ktoré sú s $[x, y]$ spojené vodorovnou paličkou: ak je vľavo palička, $s_{x,y} = s_{x-1,y} + h_{x,y}$, v opačnom prípade je $s_{x,y} = h_{x,y}$. Z týchto hodnôt už vieme vypočítať $S_{x,y}$ – počet štvoriek, ktorých nožička končí v danom bode. Ak totiž nad bodom nie je zvislá palička, v tomto bode nekončí žiadna štvorka ($S_{x,y} = 0$). Naopak, ak nad bodom je palička, sú dve možnosti: buď v danom bode končí štvorka, ktorej nožička má dĺžku 1 – takých je $s_{x,y-1} \times h_{x,y-1}$, alebo má dlhšiu nožičku, t.j. palička predlžuje všetky štvorky, ktoré končia o bod vyššie – takých je $S_{x,y-1}$. Teda $S_{x,y} = S_{x,y-1} + s_{x,y-1} \times h_{x,y-1}$. Výsledok je suma $\sum_x \sum_y S_{x,y}$.

Všetky fázy vieme robiť v čase lineárnom od veľkosti vstupu. Úlohu sme kvôli názornosti riešili postupne – v skutočnosti sa všetky „fázy“ budú robiť naraz, už pri načítaní. Keďže všetky hodnoty závisia iba od hodnôt najviac o riadok vyššie, stačí si pamätať iba dva riadky.

1011. Existuje lepšie riešenie ako backtracking, a to dynamické programovanie. Budeme postupne vyberať fľaše zľava doprava. To, či môžeme nasledujúcu fľašu zobrať, závisí od výberu dvoch predchádzajúcich. Môžeme mať teda štyri stavy: NN, NV, VN a VV (V – vybral, N – nevybral). Pre každý stav si budeme pamätať najväčší možný objem, ktorý možno vypiť z radu prvých k fliaš, pričom skončíme v danom stave (označme ho $o[s, k]$). Ak vieme hodnoty pre k fliaš, nie je problém vypočítať ich pre $k + 1$ fliaš: napríklad do stavu VN sa môžeme dostať buď z NV alebo z VV (pre k fliaš), pričom $(k + 1)$-vú fľašu nevypijeme ($o[VN, k + 1] = \max(o[NV, k], o[VV, k])$); do stavu VV sa môžeme dostať iba z NV vypitím poslednej ($o[VV, k + 1] = o[NV, k] + v_{k+1}$), atď. Maximálny objem vypitých fliaš bude v poslednom riadku. Teraz ešte stačí od konca zistiť, ktoré fľaše treba vypiť. Toto vieme spraviť tak, že ideme cez vypočítané hodnoty od konca. Ak nám ešte ostáva k fliaš, pozrieme sa, ktorá z hodnôt $o[?, k]$ je najväčšia, podľa toho vieme, či k-tu fľašu vypiť alebo nie.

1012. Nech je hodnota políčka $A_{i,j}$ rovná veľkosti najväčšieho chrámu, ktorý by tu mal svoj najsevernejší bod. Vtedy už úlohu ľahko vyriešime. Ako určiť hodnotu $A_{i,j}$? Veľkosť chrámu s najsevernejším bodom (i,j) priamo závisí od veľkostí najväčších chrámov, ktoré majú najsevernejšie body v políčkach $(i-1,j-1)$, $(i,j-1)$, $(i+1,j-1)$. Teda $A_{i,j} = \min\{A_{i-1,j-1}, A_{i,j-1}, A_{i+1,j-1}\} + 1$.

1013. Lineárne riešenie: postupujeme ako pri zlučovaní utriedených postupností a počítame si, koľkých sme už zlúčili.

Lepšie riešenie. Všimnime si prvky $x = A[(n+1) \operatorname{div} 2]$ a $y = B[(n+2) \operatorname{div} 2]$, teda výšky približne prostredných žiakov v oboch radoch. Ak $x = y$, je od nich menších žiakov rovnako veľa ako väčších, a teda medián je x. Nech teda je (bez ujmy na všeobecnosti) $x > y$. Potom všetci žiaci, ktorí sú menší od prostredného žiaka v druhom rade, sú menší aj od prostredného žiaka v prvom rade, a teda aj od všetkých žiakov za ním. Žiaden z týchto malých žiakov teda nemôže byť medián – vieme o aspoň n vyšších od neho. Podobne žiaci za prostredným žiakom prvého radu sú na medián privysokí. Všimnite si, že sme „vyhodili“ rovnako veľa primalých a priveľkých žiakov. Ostáva nájsť medián výšok tých žiakov, ktorí nám ostali. To je opäť naša pôvodná úloha, len počet žiakov je už len polovičný.

Použijeme teda ideu binárneho vyhľadávania. V každej postupnosti si pamätáme rovnako dlhé úseky, v ktorých sa môže nachádzať hľadaný žiak. Vždy porovnáme „stredných“ z oboch úsekov. Ak sú rovnako vysokí, skončili sme. Ak je napríklad ten z prvého úseku menší, v nasledujúcom kroku berieme prvú polovicu druhého úseku a druhú polovicu prvého úseku. Dostávame algoritmus so zložitosťou $O(\log n)$.

1014. Táto úloha je veľmi komplikovaná, obsahuje vo svojom riešení mnoho podúloh, ktoré by mohli tvoriť samostatné zadania. Každú ulicu treba prejsť. Ak by sa nám podarilo každú prejsť iba raz, máme riešenie. To vieme, ak sa vo všetkých križovatkách zbieha párny počet ciest (úloha 643). Ak sú v meste aj nepárne križovatky, pokúsime sa medzi nimi doplniť pomocné cesty (chceme úlohu previesť na úlohu 643). Cesty musíme doplniť tak, aby sme celkovú dĺžku ulíc mesta zväčšili minimálne. Preto si pre každú dvojicu nepárnych križovatiek vypočítame najkratšiu cestu medzi nimi (úloha 823). Teraz ide o to vybrať, ktorú križovatku s ktorou spojiť tak, aby súčet dĺžok ciest bol čo najmenší – potrebujeme teda križovatky popárovať. Ak si spravíme kompletný graf, kde vrcholy budú nepárne križovatky a hrany budú ich vzdialenosti, v tomto grafe treba nájsť najlacnejšie úplné párovanie. Hrany, ktoré sa nachádzajú v párovaní, pridáme do pôvodného grafu (ako pomocné cesty). Teraz sa vo všetkých križovatkách zbieha párny počet ciest.

Riešenie celej úlohy je v [9.B v 38, 10.3 v 8]; úloha sa nazýva úloha čínskeho poštára.

1015. Riešenie založené na backtrackingu považujeme za veľmi nešikovné. Všimnime si najskôr, že ľubovoľná okružná cesta, ktorá prechádza prvých k miest (k je najvýchodnejšie mesto), musí priamo prechádzať medzi k a $k-1$; navyše, ak pri ceste na východ prejdeme z i do k, potom pri návrate musíme prejsť postupne všetky mestá $k-1, \ldots, i+1$. Použijeme metódu dynamického programovania. Predpokladajme, že poznáme dĺžky najkratších okružných ciest pre všetky j, $2 \le j \le k$; ako z týchto údajov vypočítame dĺžku najkratšej okružnej cesty pre $k+1$ miest? Táto cesta začína v 1, pri ceste na východ príde do niektorého mesta i ($1 \le i < k$) a odtiaľ rovno do $k+1$. Pri ceste späť musí prejsť postupne cez $k, \ldots, i+1$ a odtiaľ čo najlepšie do 1. Súčet vzdialeností $|i,k+1|+|k+1,k|+\cdots+|i+2,i+1|$ je pre pevné i konštanta – treba teda minimalizovať súčet dĺžok úsekov $1, \ldots, i$ a $i+1, \ldots, 1$. Všimnite si, že keď pridáme hranu $|i,i+1|$, dostaneme okružnú cestu cez $i+1$ a dĺžku najkratšej takejto cesty už máme vypočítanú. Teda pre dané i je D_{k+1}, dĺžka okružnej cesty cez $k+1$, rovná $D_{i+1} - |i,i+1| + |i,k+1| + \sum_{j=i+1}^{k} |j,j+1|$ (teda dĺžka najkratšej okružnej cesty cez $i+1$, pričom hranu $(i,i+1)$ nahradíme cestou cez $k+1$). Zo všetkých $1 \le i < k$ vyberieme minimum – tak dostávame algoritmus so zložitosťou $O(n^2)$. Rozmyslite si, ako sa dá určiť, ktoré mestá ležia na ceste tam a ktoré na ceste späť a v akom poradí.

Asi na to budete potrebovať informáciu, pre ktoré $i+1$ sa dosiahlo minimum pre každé $1 < k+1 \leq n$.

1021. Priamočiare riešenie: Odpovede si zapisujeme do matice A rozmerov $n \times n$, $A_{i,j} = 1$, ak i pozná j. Hľadáme teda také k, aby k-ty stĺpec obsahoval samé jednotky a k-ty riadok obsahoval samé nuly, okrem $A_{k,k} = 1$. Časová zložitosť $O(n^2)$.

Lepšie riešenie: Všimneme si, že môžeme využiť obe z možných odpovedí. Pri každej vieme vylúčiť jedného kandidáta na dôležitú osobu. Teda po $n-1$ otázkach máme jediného kandidáta. Stačí skontrolovať, či nikoho nepozná a či jeho všetci poznajú. Na toto overenie stačí položiť ešte $n+k-3+[k=1]$ otázok, kde k je číslo kandidáta. Toto riešenie má teda časovú zložitosť $O(n)$.

1022. Zistenie priemeru binárneho stromu. *Priemer grafu* je dĺžka najdlhšej z najkratších ciest medzi nejakými dvoma vrcholmi. V strome medzi každými dvoma vrcholmi existuje práve jedna cesta, teda treba nájsť najvzdialenejšie vrcholy, resp. najdlhšiu cestu. Tá sa buď nachádza v ľavom podstrome, alebo v pravom podstrome, alebo prechádza cez koreň – vtedy má dĺžku rovnú súčtu hĺbok oboch podfjordov (podstromov). Dostávame tak jednoduché rekurzívne riešenie: Aby sme určili priemer fjordu zloženého z dvoch podfjordov, zoberieme maximum zo súčtu hĺbok oboch podfjordov a priemerov oboch podfjordov. Hĺbka fjordu je maximum z hĺbok jeho podfjordov zväčšené o dĺžku fjordu.

Iné, trikové riešenie. Začneme v ľubovoľnom vrchole a, napríklad v koreni. Nech b je vrchol najvzdialenejší od a (v strome ho ľahko nájdeme v lineárnom čase). Nech teraz c je najvzdialenejší vrchol od b. Potom cesta z b do c je najdlhšou cestou v celom strome. Skúste si to dokázať.

1023. Jednou z možností je použiť princíp zapojenia a vypojenia. Spočítame plochy všetkých kruhov, odpočítame od nich plochy prienikov všetkých dvojíc, pripočítame prieniky trojíc atď. Nevýhoda je, že ak sú skoro všetky kruhy na kope, takýto algoritmus môže byť až exponenciálny.

Lepšia možnosť je spočítať všetky priesečníky kružníc, každým viesť jednu vodorovnú priamku. Navyše pre každú kružnicu uvažujme jej dve vodorovné dotyčnice. Tieto priamky nám rozdelia rovinu na niekoľko pásov, v ktorých sa kružnice nepretínajú. V každom páse ostane niekoľko kruhových odsekov (pozor, niektoré ležia vnútri iných), ktorých plochu nie je problém zrátať.

Existujú ešte lepšie riešenia, ktoré sú však už veľmi náročné na implementáciu. Môžeme napríklad spracúvať vyššie spomenuté pásy v poradí zhora dole, pričom nové priesečníky (a teda polohu nasledujúcich vodorovných priamok) hľadáme až „za behu“ ako priesečníky niektorých dvoch častí kružníc, ktoré v aktuálnom páse susedia. Kružnicové oblúky, ktoré zasahujú do aktuálneho pásu, si je dobré pamätať v stromovitej dátovej štruktúre.

1024. Ihrisko si, pochopiteľne, môžeme predstaviť ako graf. Rozmyslite si, že „osmička“ taká, ako sme ju definovali v zadaní, na ihrisku existuje práve vtedy, ak niektorý komponent súvislosti ihriska obsahuje viac než jednu kružnicu. A toto je zase ekvivalentné s tvrdením, že počet hrán v danom komponente je väčší ako počet jeho vrcholov.

1025. V reči teórie grafov táto úloha hovorí: treba si udržiavať v pamäti meniaci sa graf a v každom okamihu vedieť povedať, či je súvislý.

Efektívne riešenie je veľmi náročné. Spomeňme preto dve pomalé, ale funkčné riešenia: Prvou možnosťou je pamätať si graf ako maticu susednosti. Po každej zmene spustíme prehľadávanie do šírky, aby sme zistili, či je súvislý. Takto teda potrebujeme na každú operáciu čas $O(M+N)$.

O niečo lepšie riešenie: Pre každý vrchol si pamätáme číslo komponentu, v ktorom leží. Pre každý komponent si navyše budeme pamätať jednu jeho kostru. Presnejšie, pre každú hranu si pamätáme, v kostre ktorého komponentu leží (alebo že neleží v žiadnej z kostier). Jedinou náročnou operáciou bude zmazanie hrany, ktorú máme v niektorej

z kostier, vtedy príslušnú kostru rozdelíme na dve časti a pozrieme, či existuje doteraz nekostrová hrana, ktorá ich spája. V najhoršom prípade bude aj toto riešenie potrebovať $O(M+N)$ na operáciu, dá sa ale dokázať, že pre náhodné vstupy to bude $O(N)$.

Efektívne algoritmy si vystačia s polylogaritmickým časom na každú z operácií.

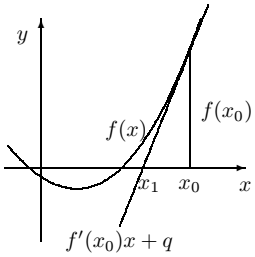
1031. Po každom presadnutí zmenšíme počet nesprávne sediacich. Teda na voľnú stoličku si presadne vždy niekto, kto nesedí na dobrom mieste, a môže skončiť na mieste, ktoré je práve voľné. (Ak je na začiatku voľné miesto v strede, môžeme tam posadiť hociktorého z nesprávne sediacich. Inak človeka, ktorý sedí v strede, považujeme za správne sediaceho.)

Skúste si dokázať, že takýto algoritmus je správny. Viete to spraviť tak, že každého čakajúceho skontrolujete iba raz?

1032. Existuje veľa možných prístupov, popíšeme niektoré z nich.

a) Koľko je vojakov v prvých n štvorcoch? $S(n) = \sum_{i=1}^{n} i^2$. Takže vojak s číslom id patrí do štvorca veľkosti $x = \min_n\{S(n) - id \geq 0\}$. Hodnota x je rádovo rovná $\sqrt[3]{n}$, preto má algoritmus, ktorý postupne počíta hodnoty $S(i)$, zložitosť $O(\sqrt[3]{n})$. V programe stačí použiť len sčítanie a porovnanie!

b) Dá sa spočítať, že $S(n) = n(n+1)(2n+1)/6$. Teraz treba nájsť kladné riešenie kubickej rovnice $f(x) = S(x) - id$. Ak $\widehat{x}$ je riešenie, potom $\lceil \widehat{x} \rceil$ je strana hľadaného štvorca. Jedným možným postupom hľadania približného riešenia takejto rovnice je Newtonova[33] iteračná metóda [2. kap. v 40]: Prvé priblíženie $\widehat{x}$ položíme $x_0 = id$, nasledujúce počítame podľa vzťahu $x_{i+1} = x_i - f(x)/f'(x)$, čo nás vždy vďaka vhodnej voľbe x_0 a konvexnosti $f(x)$ dovedie k riešeniu. Na obrázku vpravo je znázornený výpočet hodnoty x_1 z nami zvolenej hodnoty x_0.



c) Korene kubickej rovnice vieme nájsť pomocou Cardanových[34] vzorcov.

1033. Existuje veľa rôzne efektívnych prístupov:

a) Slová utriedime; použiť treba algoritmus, ktorý vykoná približne $n \log n$ operácií. Spomedzi utriedených slov už ľahko nájdeme najčastejšie sa vyskytujúce slovo.

b) Hašovanie, pozri [45, 46, 25]. Každému slovu nájdeme pomocou hašovacej funkcie jednoznačne miesto v tabuľke slov, $T[0\,..\,k-1]$. Hašovacia funkcia môže mať napríklad tvar $C \bmod k$, kde C je číslo, ktorého zápis v (trebárs) 147-kovej sústave je tvorený práve kódmi slabík zadaného slova. V takomto prípade odporúčame za k zvoliť dostatočne veľké prvočíslo.

   Rovnaké slová budú v tabuľke na tom istom mieste, takže si stačí pamätať, koľko ich je. Môže sa stať, že na rovnaké miesto sa zobrazí viac rôznych slov – tieto je najjednoduchšie ukladať do spájaného zoznamu. Pri vhodnej hašovacej funkcii treba v priemernom prípade na nájdenie slova len konštantne veľa pokusov, dokonca i keď je počet vložených prvkov rovný veľkosti tabuľky. Podrobnejšie v [6.4 v 25].

c) Binárny vyhľadávací strom, pozri [4.4.3 v 46]. V každom vrchole je uložené slovo a počet jeho výskytov. V ľavom podstrome sú všetky slová menšie a v pravom zasa väčšie než je slovo uložené v koreni.

d) „Písmenkový“ strom: Vytvoríme v pamäti zakorenený strom, v ktorom každá hrana zodpovedá kódu nejakej slabiky, a každá cesta z koreňa do listu predstavuje jedno slovo zo vstupu. V každom liste si pamätáme počet výskytov daného slova.

1034. Zo zadania nie je úplne jasné, či sa môžu nájdené rovnaké úseky čiastočne prekrývať. Jednoduchšie je uvažovať, že áno. Možné efektívne riešenia:

---

[33] Sir Isaac Newton (1643–1727), anglický matematik, fyzik a astronóm

[34] Girolamo Cardano (1501–1576), taliansky matematik, doktor, astrológ a spisovateľ

a) Hľadáme zhodné úseky, ktorých začiatky sú od seba vzdialené o k znakov (postupne pre všetky rozumné k). S rastúcim i postupne overujeme, či $T[i] = T[i+k]$, a zapamätáme si najdlhší súvislý úsek, kde to platí.

b) Pre prácu s reťazcami existujú pokročilejšie dátové štruktúry. Vieme napríklad efektívne utriediť všetky sufixy (koncové podreťazce) zadaného reťazca a pre každé dva sufixy, ktoré po sebe nasledujú v utriedenom poradí, spočítať, v koľkých znakoch sa zhodujú. Maximum z týchto čísel je dĺžka hľadaného úseku, zodpovedajúce dva sufixy hovoria, kde v pôvodnom texte hľadané úseky ležia.

c) Vyššie popísanej dátovej štruktúre sa hovorí sufixové pole. Ešte silnejšou zbraňou sú sufixové stromy. Sufixový strom pre reťazec je vlastne „komprimovaná“ verzia písmenkového stromu, ktorý obsahuje všetky sufixy daného reťazca. Hľadaný úsek v tomto strome zodpovedá najhlbšiemu vrcholu, ktorý ešte nie je listom. Sufixový strom sa dá z reťazca zostrojiť v lineárnom čase, známe algoritmy sú však veľmi komplikované.

1035. Máme vlastne nájsť kružnicu s minimálnym polomerom, vnútri ktorej budú ležať všetky vrcholy n-uholníka. Jej stred bude zrejme hľadaný bod. Vyskúšame metódu fučiaceho balónika. Nájdeme nejakú kružnicu, vnútri ktorej sa nachádzajú všetky vrcholy (napríklad opíšeme n-uholníku obdĺžnik so stranami rovnobežnými so súradnicovými osami a jemu opíšeme kružnicu) a postupne ju budeme zmenšovať, tak, aby vždy boli všetky vrcholy vnútri. Necháme jej stred S_1 na mieste a polomer zmenšíme tak, aby prechádzala bodom X najvzdialenejším od stredu. Ďalej ju trochu vyfučíme tak, aby stále prechádzala X, jej nový stred S_2 bol na spojnici S_1 a X a aby prechádzala ďalším vrcholom Y. Máme všetko pripravené, môžeme použiť vyfučovaciu procedúru: Nájdeme stred novej kružnice, ktorý leží na osi $\overline{XY}$ a má tú vlastnosť, že všetky body ležia vnútri kružnice a jej polomer je minimálny. X a Y iste budú ležať na novej kružnici. Buď budú ležať na priemere, alebo na nej bude ležať ešte jeden bod Z. Ak X a Y tvoria priemer, skončili sme, lebo zrejme menšiu kružnicu nenájdeme. Ak $\triangle XYZ$ je ostrouhlý, tiež sme zrejme našli najmenšiu možnú kružnicu. Ak je tupouhlý, iste existuje menšia, prechádzajúca tými dvoma vrcholmi, pri ktorých nie je tupý uhol. Na tieto dva vrcholy opäť našijeme vyfučovaciu procedúru. Je zrejmé, že ten „tupý“ vrchol už nikdy nebude na obvode našej kružnice, preto ho môžeme úplne vyhodiť zo zoznamu vrcholov. Tým je dokázané, že algoritmus po najviac $n-2$ vyhodeniach vrcholu skončí, a preto je kvadratický.

1041. Riešenie bez delenia: Využijeme, že kombinačné číslo je celé. Každý činiteľ v čitateli aj menovateli rozložíme na prvočinitele, potom stačí len pripočítavať alebo odpočítavať exponenty k príslušným exponentom prvočísel vo výsledku. Nakoniec prvočísla vynásobíme – na to stačí vedieť násobiť veľké číslo malým. Uvedomte si, že netreba každý činiteľ rozkladať na prvočinitele osobitne. Urobte si pomocné pole, v ktorom budete mať pre každý činiteľ informáciu, akým najmenším prvočíslom je deliteľný.

Riešenie s delením: Pre $k > 1$ môžeme využiť vzťah $\binom{n}{k} = n\binom{n-1}{k-1}/k$. Pri delení číslom k vopred vieme, že výsledok bude celé číslo. Podrobnejšie v [4.3.1 v 24].

1042. Komplikované (ale lineárne) riešenie je založené na prehľadávaní do šírky (vo fronte si udržujeme naposledy potopené políčka a z nich potápame susedné).

Jednoduchšie riešenie je pre každý ostrov metódou dynamického programovania nájsť najväčší štvorec otočený o 45° ($\diamond$) iba z jednotiek. Nech $A_{i,j}$ je veľkosť najväčšieho štvorca, ktorého spodný roh má súradnice (i,j). Platí $A_{i,j} = \min\{A_{i-1,j-1}, A_{i,j-1}, A_{i+1,j-1}, A_{i,j-2}\} + 1$ (nakreslite si to, prípadne si pozrite tiež úlohu 1012). Ostrov, ktorý obsahuje štvorec so stranou r, sa potopí najskôr za r rokov.

Prečo je tomu tak? Predstavme si úlohu bez políčok, spojito. Každý bod ostrova sa potopí za čas úmerný vzdialenosti od pobrežia. Ak teda nájdeme najväčšiu vpísanú kružnicu, nájdeme zároveň bod, ktorý sa najneskôr potopí (jej stred; pre každý iný bod

ostrova je nejaká časť pobrežia bližšie). Jediný rozdiel je meranie vzdialenosti – naša úloha je v tzv. manhattanovskej alebo L_1 metrike (kde kružnica vyzerá ako otočený štvorec).

1043. Úloha je viacnásobným hľadaním konvexného obalu (KO). Na určenie KO možno použiť napríklad Jarvisov algoritmus, ktorý je popísaný v riešení b) úlohy 413. Na určenie KO n bodov potrebujeme $O(kn)$ operácií, kde k je počet vrcholov KO. V tejto úlohe je každý spomedzi n bodov najviac v jednom KO, teda celková zložitosť algoritmu bude $O(n^2)$.

1044. Úloha z teórie grafov: meny sú vrcholy, (orientovaná) hrana medzi dvoma vrcholmi má váhu najlepšieho kurzu, za aký sa dá prvá mena vymeniť za druhú. Na riešenie možno použiť jednoduchú modifikáciu Floydovho-Warshallovho algoritmu (pozri riešenie úlohy 823), keď sčítanie nahradíme násobením.

1045. Úlohou je nájsť tranzitívny uzáver zadaného grafu. Najjednoduchším riešením je použiť Floydov-Warshallov algoritmus, ktorý nie len zistí pre každú dvojicu vrcholov, či z prvého do druhého existuje cesta, ale aj nájde dĺžku najkratšej z týchto ciest. Pozri riešenie úlohy 823.

Existujú aj efektívnejšie (ale komplikovanejšie) algoritmy. Presnejšie, dá sa ukázať, že táto úloha je rovnako ťažká ako úloha „vynásobte dve matice rozmerov $n \times n$“ [5.2 v 27]. V súčasnosti najlepší známy algoritmus na násobenie matíc má časovú zložitosť $O(n^{2.376})$.

z1011. Priamo zo zadania vyplýva, že Čiapočky si môžu čiapočky vymeniť práve vtedy, keď $C[C[i]] = i$ pre všetky i $(1 \le i \le n)$.

z1012. Označme $S_{i,j}$ súčet $C_i + C_{i+1} + \cdots + C_j$. Prvá myšlienka: pre každý možný začiatok úseku skúšame všetky možné konce, t.j. postupne s rastúcim j počítame hodnoty $S_{i,j}$. Toto sa dá spraviť v čase $O(n^2)$. Rozmyslite si, ktoré konce sa už neoplatí skúšať.

Predstavme si teraz, že sme pre začiatok i vyskúšali všetky konce a zistili sme, že $S_{i,j-1} < k$, ale $S_{i,j} > k$. Keďže všetky C_x sú kladné, pripočítavať ďalšie čísla je zbytočné. Skúsme teda začiatok $i+1$. Kontrolovať súčty s koncom menším ako j je tiež zbytočné (už $S_{i,j-1} < k$). Preto ďalší úsek, ktorý budeme kontrolovať, bude od $i+1$ po j, vrátane.

Začneme teda s podpostupnosťou C_1 (jeden prvok; súčet je $S_{1,1} = C_1$). Vždy, keď práve máme $S_{i,j} < k$, na pravom konci podpostupnosť predĺžime o 1 (teda vypočítame $S_{i,j+1} = S_{i,j} + C_{j+1}$). Naopak, ak máme $S_{i,j} > k$, na ľavom konci podpostupnosť skrátime (t.j. vypočítame $S_{i+1,j} = S_{i,j} - C_i$). Ak nájdeme také i, j, že $S_{i,j} = k$, Mamojkov strach bol opodstatnený, inak nie. Všimnite si, že obe premenné, i a j, resp. začiatok a koniec práve skúmaného úseku, pritom iba prejdú pole zľava doprava; máme teda lineárny algoritmus.

z1013. Na poradí prvkov nezáleží, sú to maskované kombinácie k prvkov z n, takže Pišta môže batoh naplniť $\binom{n}{k}$ spôsobmi. Ako tieto spôsoby šikovne generovať?

a) Jedna možnosť je generovať kombinácie postupne v lexikografickom usporiadaní. Ak si na papier napíšeme pár po sebe idúcich kombinácií, ľahko prídeme na spôsob, ako vypočítať pre ľubovoľnú kombináciu tú nasledujúcu v lineárnom čase.

b) Šikovnejšie je rekurzívne riešenie: Máme vybrať k tovarov z n. Pozrime sa na posledný, n-tý prvok. Sú dve možnosti: buď tento tovar vyberieme, alebo ho nevyberieme, resp., sú kombinácie, ktoré tento prvok obsahujú a tiež také, ktoré ho neobsahujú. Prvú skupinu kombinácií vygenerujeme tak, že si n-tý prvok zapamätáme a rekurzívne k nemu dogenerujeme všetky $(k-1)$-tice zo zvyšných $n-1$ tovarov. Druhú skupinu vygenerujeme tak, že rekurzívne vygerenujeme všetky k-tice zo zvyšných $n-1$ tovarov.

z1014. Najjednoduchšie je nechať zohrať 1. zápas borcov č. 1 a č. 2 a víťaza 1. zápasu zohrať 2. zápas s borcom č. 3, atď., víťaza $(n-2)$-ého zápasu zohrať $(n-1)$-vý zápas s borcom č. n. Analogicky určíme spomedzi ostatných najslabšieho. Celkovo potrebujeme $2n-3$ zápasov.

Lepší spôsob je nechať zohrať zápasy najprv $\lfloor n/2 \rfloor$ rôznych dvojíc, a potom spomedzi neporazených rovnakým spôsobom ako vyššie vybrať najsilnejšieho a spomedzi porazených

zasa najslabšieho. Ak je n párne, potrebujeme $n/2+(n/2-1)+(n/2-1) = (3/2)n-2$ zápasov, ak je n nepárne, $(n-1)/2+((n-1)/2-1)+((n-1)/2-1)+2 = (3/2)n-3/2$ zápasov. Toto vieme v oboch prípadoch zapísať ako $\lceil (3/2)n-2 \rceil$.

z1021. Stačí si vybrať ešte neskontrolovanú Čiapočku, postupne prejsť cez všetky, ktoré sa takto držia a označiť ich ako skontrolované. Uvedomte si, že nakoniec musíme prísť opäť k Čiapočke, s ktorou sme začali (prečo?). Ak sme skontrolovali už všetky, sme hotoví, ak nie, nájdeme prvú neskontrolovanú a začneme odtiaľ. Novú neskontrolovanú Čiapočku stačí hľadať medzi tými Čiapočkami, ktoré majú väčšie číslo ako naposledy vybraná neskontrolovaná Čiapočka.

z1022. Nešikovné je počítať vzdialenosti pre každé dve mestá osobitne. Spočítajte si vzdialenosti s_i z 1. mesta do mesta i. Vzdialenosť miest j a k je potom $s_k - s_j$, pre $k > j$.

z1023. Treba si uvedomiť, že stačí brať maškrty usporiadané podľa jednotkovej ceny (c/v) od najväčšej. Akonáhle už nemôžeme zobrať celú maškrtu, zoberieme takú jej časť, aby sme zapratali celý zvyšok batohu. Takto možno dosiahnuť časovú zložitosť rovnú zložitosti triedenia, t. j. prinajlepšom $O(n \log n)$.

Existuje však aj lepšie, lineárne riešenie, podobné hľadaniu mediánu (pozri riešenie úlohy 1333): Maškrty rozdelíme na dve počtom podobné časti tak, aby každá maškrta v jednej časti mala jednotkovú cenu vyššiu alebo rovnú ako každá maškrta z druhej časti. Potom sa pozrieme, či možno zobrať všetky drahé maškrty. Ak áno, všetky vezmeme a zaoberáme sa lacnými maškrtami (dostávame rovnakú úlohu: ktoré z lacných potravín treba zobrať, pričom sa musia zmestiť do zvyšku batohu?). Ak sa nám drahé maškrty nezmestia všetky, lacnými sa vôbec nemusíme zaoberať, úlohu riešime iba s drahými maškrtami. Daný postup opakujeme, až kým nezostane len jedna maškrta, ktorú podľa potreby rozkrojíme.

z1024. Vždy treba mať akcie len jednej spoločnosti. Výhodnejšia je tá, ktorej pomer cien q_{i+1}/q_i je väčší (skúste dokázať), q_i je cena v i-ty deň. Majetok po n dňoch vzrastie najviac $\prod_{i=1}^{n-1} \max\{w_{i+1}/w_i, t_{i+1}/t_i\}$ násobne. Pozor, vybrať lepšiu spoločnosť na základe $q_{i+1} - q_i$ je zlé riešenie. Protipríkladom sú napríklad hodnoty $t = 1,\ 2$ a $w = 0.1,\ 1$.

z1031. Hľadaná postupnosť sa volá *Grayov kód*[35]. Ako ju nájsť? Ak vieme úlohu vyriešiť pre n a riešením sú vektory $v_1, v_2, \ldots, v_{2^n}$, riešenie pre $n+1$ dostaneme napríklad takto: Všetky vektory napíšeme dvakrát pod seba, raz v poradí 1 až 2^n, a potom v zrkadlovom poradí 2^n až 1. K prvej polovici pripíšeme na začiatok 0 a k druhej 1. Dostaneme teda $0v_1, 0v_2, \ldots, 0v_{2^n}, 1v_{2^n}, \ldots, 1v_2, 1v_1$.

a) Dá sa napísať rekurzívny program, ktorý túto postupnosť generuje. Vystačíme si s jedným poľom **array**$[1\,..\,n]$ **of** $0\,..\,1$.

b) Očíslujme si riadky riešenia od 1 po 2^n a stĺpce od 1 po n (zhora nadol a zľava doprava). Keď si pozornejšie všimneme vyššie popísané riešenie, prídeme na to, že vieme priamo určiť, aká cifra je v r-tom riadku a s-tom stĺpci: s-tý stĺpec začína 2^{n-s} nulami, a potom sa striedajú skupiny jednotiek a núl dĺžky 2^{n-s+1}. Teda v riadku r a stĺpci s je 1 práve vtedy, keď $\lfloor (r-1-2^{n-s})/2^{n-s+1} \rfloor \bmod 2 = 0$.

c) Na každý riadok sa môžeme dívať ako na zápis nejakého n-bitového čísla v dvojkovej sústave. Dá sa dokázať, že v $(i+1)$-vom riadku je zápis čísla i xor $(i \operatorname{div} 2)$.

z1032. Treba preskúšať všetky možné obsahy batohu. Najšikovnejšie je usporiadať tovary podľa objemu. Keď volíme jednotlivé naplnenia, vieme, či má zmysel ešte do batohu pridávať. Takýmto spôsobom podstatne zmenšíme počet zbytočne kontrolovaných naplnení batohu. Pozor! Riešenie, v ktorom sa tovary do batohu vyberajú na základe maximálnosti ceny alebo podielu cena/objem alebo minimálnosti objemu, prípadne kombináciou predchádzajúcich spôsobov, nie je dobré. Nájdite protipríklady na jednotlivé spôsoby.

[35] Gray Frank, kód vynašiel r. 1953 ako výskumník v Bell Labs. Jeho snahou bolo minimalizovať dopad chyby pri prenose digitálneho signálu.

z1033. V skupine, ktorá kandiduje na najdlhšiu, musia byť najprv znaky A, potom znaky B a na koniec znaky C a pre ich počet musí platiť $počet(A) \geq počet(B) \leq počet(C)$. Teda rozhodujúci je počet znakov B v skupine.

z1034. Ak si označíme $f[s,n]$ farbu Čiapočky v skupine $s \in \{1,2\}$, ktorú drží Čiapočka číslo n ľavou rukou, po prechytení bude držať Čiapočku farby $f[3-s, f[s,n]]$ (všimnite si, že $3-s$ je číslo druhej skupiny). Teraz stačí „ísť“ postupne po Čiapočkách, kým neprídeme k Čiapočke, od ktorej sme začali. Ak sme už prešli cez všetkých n Čiapočiek, sme hotoví, inak treba nájsť Čiapočku, cez ktorú sme ešte nešli. Takto postupne prejdeme cez všetky kruhy, ktoré po prechytení vznikli.

Hľadaný vstup z podúlohy a) má $n = 3$. V podúlohe b) pre ľubovoľné n stačí, aby sa aspoň v jednej skupine točili všetky Čiapočky na mieste.

z1041. Táto úloha patrí medzi klasické programátorské úlohy – program, ktorý vypíše sám seba sa nazýva *quine*. V riešení treba prekonať iba jednu prekážku: ako vytlačiť pomocou štandardných funkcií alebo procedúr tie riadky, ktoré obsahujú príkazy na vytlačenie riadkov programu (trochu krkolomné, ale tento cyklus treba v riešení rozseknúť). Uvádzame jedno krátke riešenie v C:

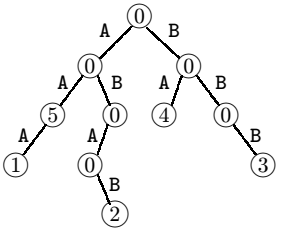
```
char *s = "char *s=%c%s%c;%cmain(){printf(s,34,s,34,10,10);}%c";
main(){printf(s,34,s,34,10,10);}
```

... a ešte jedno v Pascale (`#97` je znak `a`, `#91` a `#93` sú znaky `[`, `]`, atď.):

```
var a : array [1..7] of string; i : integer;
begin
  a[1] := 'var a : array [1..7] of string; i : integer;';
  a[2] := 'begin';
  a[3] := '  writeln(a[1]); writeln a[2]);';
  a[4] := '  for i := 1 to 7 do';
  a[5] := '    writeln(#97:3, #91, i, #93, #58, #61, #39, a[i], #39, #59);';
  a[6] := '  for i := 3 to 7 do writeln(a[i])';
  a[7] := 'end.';
  writeln(a[1]); writeln(a[2]);
  for i := 1 to 7 do
    writeln(#97:3, #91, i, #93, #58, #61, #39, a[i], #39, #59);
  for i := 3 to 7 do writeln(a[i]);
end.
```

z1042. Uvedené preusporiadanie sa nazýva pivotizácia[36]. Najšikovnejšie je hľadať prvok $A[j] \leq p$ a prvok $A[k] \geq p$ také, že $j < k$, a potom ich vymeniť. Ak také prvky už neexistujú, sme hotoví.

z1043. Nešikovné riešenie je zapamätať si všetky vstupné slová v poli a potom zadané slovo s nimi porovnávať. Lepšie je použiť písmenkový strom. V našej úlohe to bude vlastne len binárny strom, ale nič nebráni tomu, aby sme ho rozšírili na viac písmen.



Vstupné slová uložíme do stromu nasledovne. Začneme v koreni, ak je prvé písmeno v slove `A`, posunieme sa v strome vľavo, ak je prvé písmeno `B`, posunieme sa vpravo.

[36] tvorí základ triediaceho algoritmu QuickSort, ktorý vymyslel Charles Antony Richard Hoare, Computer Journal 5, 1962, 10–15

Prvé písmeno zo slova odstránime a z vrcholu, v ktorom sa nachádzame, pokračujeme so zvyškom vstupného slova. Keď vyčerpáme celé vstupné slovo, do vrcholu zapíšeme jeho poradové číslo. Je zrejmé, ako teraz kontrolujeme, či zadané slovo bolo vo vstupnej postupnosti – stačí sa podľa písmen slova posúvať. Keď vyčerpáme celé slovo a hodnota bude 0, alebo chceme ísť neexistujúcou vetvou, slovo sa vo vstupnej postupnosti nenachádzalo. Inak je vo vrchole jeho poradové číslo. Uvedomte si, že pri takejto reprezentácii vykonáme počet porovnaní úmerný dĺžke zadaného slova, bez ohľadu na to, koľko slov bolo na vstupe. Na obrázku je príklad stromu pre úvodnú postupnosť: `AAA`, `ABAB`, `BBB`, `BA`, `AA`.

z1044. Pri riešení tejto úlohy sa dajú vymyslieť rôzne prístupy.

a) Zo zadania vieme, že riešením má byť polynóm $P_k(n) = a_{k+1}n^{k+1} + a_k n^k + \cdots + a_1 n + a_0$. Navyše, keďže súčet nula k-tych mocnín je rovný nule, $P_k(0) = a_0 = 0$. Potrebujeme teda iba zistiť príslušné koeficienty $a_1, a_2, \ldots, a_{k+1}$. Vypočítajme $P_k(n)$ pre hodnoty $n = 1, 2, \ldots, k+1$, t.j., sčítajme $1, 2, \ldots, k+1$ k-tych mocnín. Dostaneme sústavu $k+1$ rovníc s $k+1$ neznámymi.
Napríklad pre $k = 2$ vyzerá zostavenie sústavy takto:

$$\begin{array}{lll} P_2(1) = 1 & 1 = 1^3a_3 + 1^2a_2 + 1^1a_1 & 1 = a_3 + a_2 + a_1 \\ P_2(2) = 1 + 4 = 5 & 5 = 2^3a_3 + 2^2a_2 + 2^1a_1 & 5 = 8a_3 + 4a_2 + 2a_1 \\ P_2(3) = 1 + 4 + 9 = 14 & 14 = 3^3a_3 + 3^2a_2 + 3^1a_1 & 14 = 27a_3 + 9a_2 + 3a_1 \end{array}$$

Túto sústavu rovníc vieme vyriešiť *Gaussovou eliminačnou metódou* v čase $O(k^3)$. (Keďže vieme, že polynóm stupňa najviac k je jednoznačne určený svojimi hodnotami v ľubovoľných k bodoch, vieme, že naša sústava musí mať práve jedno riešenie. Toto nám môže trochu uľahčiť písanie programu.)

b) Platí $P_k(n) = \sum_{i=1}^{n} i^k$, teda máme $P_k(n) + (n+1)^k = P_k(n+1)$, odkiaľ dostaneme $(n+1)^k = P_k(n+1) - P_k(n)$. Teraz obe strany rozpíšeme a porovnáme koeficienty pri rovnakých mocninách n. Opäť dostaneme sústavu $k+1$ rovníc, tentokrát však tzv. trojuholníkovú, takže stačí postupne dosadzovať (to vieme v kvadratickom čase):
Pre $k = 2$ máme

$$\begin{array}{l} (n+1)^2 = P_2(n+1) - P_2(n) \\ n^2 + 2n + 1 = (3a_3)n^2 + (3a_3 + 2a_2)n + (a_3 + a_2 + a_1) \end{array} \qquad \begin{array}{l} 1 = 3a_3 \\ 2 = 3a_3 + 2a_2 \\ 1 = a_3 + a_2 + a_1 \end{array}$$

Vo všeobecnosti má sústava tvar $\binom{k}{k-j} = \sum_{i=j}^{k} a_{i+1}\binom{i+1}{i+1-j}$ pre $0 \le j \le k$.

1111. Veľmi pomôže jednoduché tvrdenie: Súčet arít všetkých operátorov P-výrazu je o 1 menší ako počet tokenov, (t.j. v našom prípade počet znakov). Jednoduchý dôkaz indukciou: pre `0...9` to platí, nech to platí pre $v_1 \ldots v_n$, potom to platí aj pre $Ov_1 \ldots v_n$, kde O je n-árny operátor. Najprv zistíme súčet arít všetkých operátorov a počet otáznikov. Musí platiť $n = A + p \times a_o + 1$, kde n je počet znakov, A je súčet arít známych operátorov, p počet otáznikov a a_o neznáma arita otáznika. Z toho vypočítame a_o. Ak nevyjde celočíselné, skončili sme. Inak treba (pri už známej arite **?**) skontrolovať korektnosť P-výrazu, čo spravíme rekurzívnou funkciou presne podľa definície P-výrazu.

1112. Ako ináč, pomôcť môže jedine rekurzia. Počet vnorení nejakej procedúry bude znamenať veľkosť súčtu. Procedúra *Nacitaj* načíta do globálnej premennej číslo a zavolá procedúru *Minus1*. Tá zníži premennú o 1 a zavolá sama seba. Ak je premenná nulová, zavolá opäť *Nacitaj*. Pri návrate *Minus1* zvýši opäť premennú o 1. *Nacitaj*, keď načíta 0, vynuluje premennú a skončí. Vylepšenia: procedúry pre zapamätanie väčších čísel (*Minus10*, *Minus100*, ...). Nezabudnite na záporné čísla na vstupe.

1113. Trochu prestrelené zadanie. Nie je nám známe nejaké exaktné riešenie. Testovanie trojuholníkových alebo n-uholníkových nerovností funguje len ako nutná podmienka. Nádej poskytujú iteratívne riešenia. Jednoduché fyzikálne iteratívne riešenie: Rozmiestnime plte náhodne, namiesto palíc medzi plte natiahnime pružiny s danou tuhosťou a dĺžkou rovnou dĺžke palice. Každá plť nech má svoju hmotnosť, odpor prostredia nech je priamo úmerný rýchlosti plte. Simulujme pohyb pltí. Ak sme vhodne zakomponovali odpor prostredia, po istom čase sa plte prestanú hýbať. Potom stačí skontrolovať, či tento rovnovážny stav vyhovuje zadaniu a či stačí vymeniť pružiny za palice.

1114. Usporiadajme si domy od najjužnejšieho po najsevernejší (podľa južnej steny). Východné a západné steny delia x-ovú os (rovnobežku) na pásiky. V každom tomto pásiku si stačí pamätať výšku doteraz najvyššieho domu. Postupne budeme pridávať domy od najjužnejšieho, t.j. v každom pásiku, do ktorého dom zasahuje, zistíme, či ho vidno, ak áno, zapamätáme si tento dom ako najvyšší v tomto pásiku a vieme, že tento dom určite vidno. Pásikov je úmerne n, preto zložitosť je $O(n^2)$.

Existuje aj komplikované riešenie s časovou zložitosťou $O(n \log n)$. Jeden možný algoritmus vyzerá nasledovne: Domy si utriedime podľa vzdialenosti od Santa, najbližší budeme spracúvať ako prvý. V okamihu, keď ideme spracovať nejaký dom, máme teda už spracované všetky, ktoré ho môžu „zacloniť“.

Horný obrys už spracovaných domov tvorí zľava doprava idúcu lomenú čiaru. Vodorovné úseky tejto čiary si budeme pamätať v listoch vyváženého stromu. (Napr. *splay strom* je vhodnou dátovou štruktúrou.) V logaritmickom čase vieme nájsť miesta, kde na tejto lomenej čiare začína a končí práve spracúvaný dom. Aby sme upravili lomenú čiaru, stačí efektívne odstrániť časť stromu medzi týmito miestami. (Splay stromy toto vedia.) Aby sme vedeli povedať, či je z práve pridávaného domu aspoň kúsok viditeľný, potrebujeme vedieť povedať, aké je najnižšie miesto na odstraňovanej časti lomenej čiary. Túto informáciu si preto budeme pamätať vo vnútorných vrcholoch stromu. (Presnejšie, vo vrchole stromu si pamätáme minimálnu výšku tej časti lomenej čiary, ktorá zodpovedá jeho podstromu.)

1115. Návrh a implementácia robustného „systému na organizovanie súťaže“ je práca na niekoľko mesiacov. Zadanie tejto úlohy sa snažilo sústrediť na jednu malú časť – skompilovanie, spustenie a otestovanie správnosti jedného riešenia. Najschodnejšia cesta je asi použiť *skript* (dávkový súbor), ktorý bude v správnom poradí volať ďalšie skripty pre jednotlivé kroky. Potom sa už len treba pohrať s detailmi, ako je napríklad meranie času bežiaceho riešenia a jeho zastavenie po uplynutí limitu.

1121. Najprv treba zistiť, či sa všetkým dá vyhovieť. Dá sa to, ak kamarát každého študenta chce bývať s ním a zároveň nie sú dvaja ľudia, ktorí chcú bývať s tým istým študentom. Sťahovanie: Najjednoduchšie je zobrať študenta a, ak nebýva s kamarátom, vyhodiť jeho spolubývajúceho a prisťahovať tam jeho kamaráta. Opakovať pre všetkých študentov. Aby sme nemuseli lineárne prezerať pole *Izba*, je vhodné spraviť si pole, v ktorom bude pre každú izbu zoznam obyvateľov a udržovať si ho počas výpočtu.

1122. Jedno z riešení je na tvrdo napísať procedúru na výpis čísla v dvojkovej sústave zhruba v štýle

if n **and** \$4000 > 0 **then** *write*(1);
if n **and** \$2000 > 0 **then** *write*(1) **else if** $n >$ \$2000 **then** *write*(0);

... alebo využiť silu rekurzie (pozri tiež úlohu 1112). Základom bude rekurzívna procedúra *Bin*. Ak je n párne, väčšie ako 1, vydelí ho 2, rekurzívne sa zavolá a vypíše 0. Pri návrate musí obnoviť pôvodný stav, t.j. vynásobiť n dvoma. Ak je nepárne, odpočíta 1, vydelí 2, rekurzívne sa zavolá, vypíše 1, vynásobí n dvoma a pripočíta 1. Triviálne prípady $n = 0, 1$

iba vypíšu 0 alebo 1. Po skončení *Bin* máme v n pôvodné číslo, na obrazovke je vypísané, môžeme ho vypísať normálne (v desiatkovej sústave) a skončiť.

1123. Backtracking. Lepšie riešenie neexistuje, lebo rôznych postupov, ako získať žiadanú pečiatku, môže byť veľmi veľa. Ku každej pečiatke si môžeme pamätať zoznam pečiatok, ktoré ju potrebujú, a počet pečiatok, ktoré treba na jej získanie. Ďalej máme zoznam pečiatok, ktoré vieme získať hneď. Z tohto zoznamu vyberieme jednu pečiatku, dáme ju do zoznamu riešenia, aktualizujeme zoznam hneď získateľných pečiatok a počty potrebných pečiatok pre každú pečiatku. Rekurzívne opakujeme, kým nezískame žiadanú pečiatku.

1124. Majme graf, v ktorom sú ilegáli vrcholmi a hrana medzi i a j je práve vtedy, ak sa i a j poznajú. *Artikuláciou* v grafe nazveme taký vrchol, po ktorého vyňatí sa graf rozpadne na dve časti (to je vlastne naša dôležitá osoba). Úlohou je zistiť, či sa v tomto grafe nachádzajú artikulácie (pozri [3.B v 38]).

Použijeme jednoduché prehľadávanie do hĺbky, pričom si pri každom vrchole budeme pamätať dve čísla: p_i značí poradie vrcholu i, v ktorom sme ho prehľadávaním v grafe objavili a najmenšie r_i také, že sa z i do vrcholu s poradím r_i môžeme dostať cestou pozostávajúcou z niekoľkých hrán prechádzaných v smere prehľadávania a nasledovaných jednou spätnou hranou (to je taká, ktorou sme pri prehľadávaní neprešli, lebo sme v jej koncovom vrchole už boli skôr). Ak po skončení algoritmu pre niektorý vrchol platí, že existuje aspoň jeden jeho syn j taký, že $r_i \geq p_j$, potom je vrchol i artikuláciou. Artikuláciou je aj vrchol s poradím 1, pokiaľ sme pri prehľadávaní z neho vyrazili aspoň dvoma rôznymi smermi.

1125. Treba si uvedomiť, že môžu nastať konflikty – môže sa stať, že chce viac písmeniek ísť na to isté políčko. V takomto prípade treba pre každé políčko, na ktoré chce ísť viac kandidátov, náhodne vybrať jedného. Ostatným treba oznámiť, že sa hýbať nebudú. Takto sa môže stať, že celé „reťaze“, ba až „stromy“ písmeniek sa nebudú môcť želaným smerom pohnúť, lebo posledné písmenko nemá kam ísť. Šikovným spôsobom implementácie je najskôr si v pomocnom poli pre každé políčko zaznačiť, z ktorých smerov naň chcú písmenká prísť. Vo fronte si budeme pamätať políčka s konfliktom. Potom až kým sa nám fronta nevyprázdni: Vyberieme políčko, vyberieme jedného „výhercu“ (ak je niekto z kandidátov už teraz na políčku, automaticky vyhráva, inak vyberieme náhodne), ostatným nastavíme, že chcú zostať tam, kde sú. Ak týmto vznikli nové konflikty, zaradíme aj ich políčka do fronty. Po vyprázdnení fronty zostane už len niekoľko „reťazí“ písmeniek, ktoré sa naozaj posunú.

1131. Najjednoduchšie je prejsť pre každé slovo všetky riadky, stĺpce aj uhlopriečky. Funkcia hľadajúca reťazec (vzorku) v inom reťazci sa dá napísať lineárne (napríklad KMP, t.j. Knuthov-Morrisov-Prattov algoritmus). Ešte rýchlejšie je použiť Ahov-Corasickov algoritmus a hľadať všetky vzorky naraz.

1132. Všetko treba schovať do rekurzie. Budeme mať procedúry `OblaZatvorka`, `SpicataZatvorka` a `BrckataZatvorka`. Všetky z nich, a aj hlavný program, budú vyzerať nasledovne: Načítame zo vstupu jeden znak. Pokiaľ je to niektorá otváracia zátvorka, zavoláme príslušnú procedúru a po jej skončení pokračujeme ďalej. Pokiaľ je to zatváracia zátvorka, tak skontrolujeme, či sme v procedúre s „rovnakým“ menom. Ak nie, vypíšeme `NIE` a skončíme celý program, ak áno, skončíme aktuálnu procedúru a pokračujeme ďalej. Ak sme po dočítaní vstupu v hlavnom programe, vypíšeme `ANO`, inak `NIE`.

1133. Stupeň čísla zrejme nezávisí od poradia cifier, preto najmenšie číslo s daným stupňom bude mať cifry vzostupne usporiadané. Čísla, obsahujúce párnu cifru a 5 majú stupeň max. 2. Cifry $5, 5$ svedčia o stupni najviac 3. Ďalej vieme, že niektoré dvojice čísel sa dajú vymeniť za iné cifry alebo dvojice cifier (napríklad $2, 2$ vymeníme za 4). Takto vieme veľmi obmedziť počet čísel, ktoré treba prezrieť.

1134. Ťažká úloha. Treba zistiť, či je zadaný graf rovinný.

Schodnou cestou k pomalému riešeniu je *Kuratowského*[37] *veta*: Graf je rovinný práve vtedy, keď neobsahuje ako podgraf delenie grafu $K_{3,3}$ alebo K_5 (K_5 má 5 vrcholov pospájaných každý s každým, $K_{3,3}$ má 6 vrcholov v 2 trojiciach, každý z jednej trojice je spojený s každým v druhej). *Delenie hrany* $\{u, v\}$ grafu G znamená pridanie nového vrcholu z a nahradenie $\{u, v\}$ hranami $\{u, z\}$ a $\{z, v\}$ (teda akoby sme na hranu prikreslili nový vrchol – zjavne bude takýto graf rovinný práve vtedy, keď bol pôvodný graf rovinný). *Delenie (subdivízia) grafu* je graf, ktorý môžeme dostať z pôvodného niekoľkými deleniami niektorých jeho hrán.

Existuje však dokonca lineárne riešenie, ktoré dokonca nájde pre rovinný graf aj jedno korektné nakreslenie. Autorom prvého takéhoto algoritmu je Tarjan, jeho algoritmus je však príliš komplikovaný na to, aby sme ho tu popisovali.

1135. Najjednoduchšie je asi každý výstupný formát natvrdo „zadrôtovať“ do kódu programu. Výber formátu potom môžeme spraviť parametrami na príkazovom riadku. Dôležitou súčasťou programu je parsovanie vstupu: treba vedieť rozpoznať aspoň to, čo je hlavička a čo telo programu, a vedieť rozdeliť blok kódu na jednotlivé príkazy.

1141. Vyriešme najskôr zadanú úlohu pre $k = 1$. Nech $random(n)$ generuje rovnomerne rozložené náhodné celé čísla od 0 do $n - 1$, vrátane. Pomocou volaní tejto funkcie teraz vygenerujeme jednu náhodnú permutáciu.

Majme pole $A[1\,..\,n]$, pričom na začiatku položíme $A[i] = i$. Permutáciu budeme generovať postupne. Po i krokoch bude $A[1\,..\,i]$ obsahovať prvých i prvkov hľadanej permutácie, no a na $A[i + 1\,..\,n]$ budú práve tie prvky, ktoré sme doteraz nepoužili. Ak teda chceme v $(i + 1)$-vom kroku vygenerovať ďalší prvok permutácie, jednoducho ho náhodne vyberieme spomedzi ešte nepoužitých.

```
for i := 1 to  n − 1 do begin
  t := i + random(n + 1 − i);
  v := a[i]; a[i] := a[t]; a[t] := v
end
```

Ak teraz máme vygenerovať niekoľko permutácií a nesmú byť žiadne dve rovnaké, budeme si musieť už vygenerované permutácie pamätať. Jednou možnosťou je vytvárať si analógiu písmenkového stromu – zakorenený strom, v ktorom z každého vrcholu vedie n hrán (očíslovaných od 1 do n), pričom každá cesta z koreňa do listu zodpovedá jednej už vygenerovanej permutácii.

Pre veľké k je lepším postupom kódovať permutácie do čísel od 1 do $n!$ (alebo jednoduchšie, do niektorých čísel od 0 do $n^n - 1$) a pamätať si v bitovom poli, ktoré už boli vygenerované.

Ak $k \leq n!/2$, je takéto generovanie permutácií optimálne. Pri generovaní každej z k hľadaných permutácií je pravdepodobnosť aspoň $1/2$, že bude nová, a teda v očakávanom prípade potrebujeme na nájdenie novej permutácie najviac dva pokusy.

Pre väčšie k si už môžeme dovoliť vygenerovať všetky permutácie (je ich nanajvýš dvakrát toľko ako máme vypísať) a potom v náhodnom poradí vypísať k z nich.

1142. Majme číslo $c = a_0 \cdot 10^0 + a_1 \cdot 10^1 + a_2 \cdot 10^2 + \ldots + a_n \cdot 10^n$. Na výpočet zvyšku $c \bmod 7$ potrebujeme vypočítať zvyšky $(a_i \cdot 10^i) \bmod 7$. Zvyšky $10^i \bmod 7$ nadobúdajú periodicky hodnoty 1, 3, 2, 6, 4, 5. Program sa nachádza v jednom z konečného počtu stavov určených doterajším zvyškom a zvyškom 10^i. Tieto stavy je možné si zapamätať vo vetvení programu. Premennú použijeme na načítanie ďalšej cifry.

1143. *Anagram* nejakého slova je slovo, ktoré vznikne z pôvodného slova utriedením jeho písmen. Slovo je prešmyčkou iného slova práve vtedy, ak majú rovnaké anagramy.

[37] Kazimierz Kuratowski (1896–1980), poľský matematik

Slová utriedime v prvom rade podľa anagramov, ale pri rovnosti anagramov podľa poradia na vstupe. Skupiny treba vypisovať v správnom poradí – udržiavajte si informáciu o tom, kde sa pri triedení nachádza prvok z pôvodnej i-tej pozície. Ešte efektívnejšie je pamätať si anagramy slov zo vstupu v písmenkovom strome, pričom pre každý anagram si pamätáme zoznam slov zo vstupu, ktorým prislúcha.

1144. Dáta je vhodné uchovávať v dynamických štruktúrach – vyhľadávacie stromy.

1145. Definované makrá si pamätáme v tvare dvoch reťazcov: pôvodného a nahradeného. Keď pri čítaní vstupu narazíme na lomítko nasledované nejakým výrazom, hľadáme vyhovujúce makro. Pri tom treba testovať, či vyhovuje tvar podvýrazov (predpokladáme jednoznačnosť priradenia podvýrazov). Pretože zadanie pripúšťa vnorenie makier, výsledok ukladáme do buffera a ďalej spracovávame.

1211. Pozri riešenie úlohy z1031.

1212. Program, ktorý je korektný vo viacerých programovacích programovacích jazykoch, sa volá *polyglot*. Tu je naše riešenie:

```
var x : string;
begin
  x := {{}'}'{'{};
  if x = '}' then writeln('NIE')
  else writeln('ÁNO');
end.
```

1213. Skúšaním všetkých dosiahneme časovú zložitosť $O(2^{(n^2)})$. Lepší čas vieme dosiahnuť metódou dynamického programovania, ak obetujeme trochu pamäte. *Stavom riadku* budeme označovať *nenápadné* vyjedenie kompótov v riadku. Skúste si dokázať, že počet rôznych stavov riadku je Fibonacciho číslo F_{n+2}, čo je $\Theta(\phi^n)$ (pozri úlohu 113). Pre každý stav v i-tom riadku vypočítame najväčší možný objem, ktorý vieme dosiahnuť – to vieme urobiť rýchlo, ak sme si už rovnaké údaje zapamätali pre predchádzajúci riadok. Takto dostávame algoritmus s pamäťovou zložitosťou $O(\phi^n)$ a časovou zložitosťou $O(n\phi^{2n})$.

1214. Riešením úlohy je jednoduchý backtracking. Pre jeho urýchlenie uvažujte symetriu v úlohe a ďalšie podmienky, ktoré môžu vylúčiť existenciu riešenia na začiatku výpočtu (parita počtu políčok a pod.)

1215. Najmenší počet *Download*ov dosiahneme vtedy, keď z pamäti vyhodíme vždy to písmeno, ktoré sa vo zvyšku súboru vyskytuje najneskôr.

1221. Riešenie založíme na tvrdení: Nech máme postupnosť všetkých permutácií n prvkov takú, že každé dve susedné permutácie sa líšia výmenou dvoch prvkov – nazvime túto postupnosť X. Potom postupnosť permutácií $n+1$ prvkov s požadovanou vlastnosťou môžeme z tejto vytvoriť tak, že z každej permutácie n prvkov vytvoríme $n+1$ permutácií $n+1$ prvkov vhodným vsunutím $(n+1)$-vého prvku.

1222. Zreťazíme dvakrát za sebou prvé slovo a dáme v ňom vyhľadávať druhé. Jediným problémom ostáva vyhľadávanie podreťazca v reťazci. (Pozri [9 v 1, 32 v 6, 6.7 v 34, 19 v 41, 1.12 v 47].)

1223. Nech pri poslednom spájaní spájame integráče ABCD a EFG. Potom je zrejmé, že je úplne jedno, či najprv vytvoríme ABCD a potom EFG alebo najprv EFG a potom ABCD, alebo chvíľu spájame jedno a chvíľu druhé. Aby však bolo výsledné spájanie optimálne, musia byť aj jednotlivé časti (ABCD a EFG) pospájané optimálne.

To nás navádza na riešenie pomocou dynamického programovania. Vypočítame si optimálne spájanie pre každý úsek integráčov a ukladáme si medzivýsledky do poľa. Ak postupujeme od najmenších úsekov po najväčšie, máme už všetky potrebné informácie o menších úsekoch v poli. Zložitosť takéhoto algoritmu je $O(n^3)$.

1224. Úloha preformulovaná do teórie grafov: zistite, či existuje jednoduchý súvislý graf, ak sú dané stupne jeho vrcholov. Využijeme dve tvrdenia:

Tvrdenie 1: Nech $n-1 \geq s_1 \geq s_2 \geq \cdots \geq s_n \geq 0$, potom n-vrcholový graf so stupňami vrcholov $s_1, \ldots, s_n$ existuje práve vtedy, keď existuje $(n-1)$-vrcholový graf so stupňami vrcholov $s_2 - 1, \ldots, s_{s_1+1} - 1,\ s_{s_1+2},\ \ldots,\ s_n$ (inými slovami, nič nepokazíme, keď zoberieme vrchol najväčšieho stupňa a spojíme ho s tými ďalšími vrcholmi najväčšieho stupňa, aký má mať stupeň).

Tvrdenie 2: Ak G je graf s n vrcholmi a h hranami, ktorý neobsahuje izolované vrcholy a platí $h \geq n-1$, potom existuje graf G' taký, že jeho vrcholy majú rovnaké stupne ako vrcholy G a navyše je súvislý.

Treba teda overiť, či $0 < s_i < n$, či $h \geq n-1$ (počet hrán vieme zo vzťahu $\sum s_i = 2h$) a prvé tvrdenie – to vieme overiť v $O(n^2)$ (triediť môžeme CountSortom, prípadne si pre každý stupeň zapamätáme, koľko vrcholov tohto stupňa máme).

Erdős a Gallai vymysleli ešte šikovnejšiu podmienku, pomocou ktorej vieme našu úlohu riešiť v lineárnom čase.

Tvrdenie 3: Nech $n-1 \geq s_1 \geq s_2 \geq \cdots \geq s_n \geq 0$, potom n-vrcholový graf so stupňami vrcholov $s_1, \ldots, s_n$ existuje práve vtedy, keď pre každé r platí:

$$\sum_{i=1}^{r} s_1 \leq r(r-1) + \sum_{i=r+1}^{n} \min(r, d_i)$$

1225. Hlavnou úlohou je vlastne napísať jednoduchý syntaktický analyzátor jazyka. Návod, ako na to, možno nájsť napríklad v [5.1-6 v 46].

1231. Označme si $P_{n,m}$ počet partícií čísla n so sčítancami neprevyšujúcimi m. Koľko je $P_{n,m}$? Nuž, buď m je jeden zo sčítancov (a potom ostáva vyriešiť, koľkými spôsobmi vieme napísať číslo $n-m$), alebo nie je (takže všetky sčítance sú nanajvýš $m-1$. Teda $P_{n,m}$ vieme ľahko spočítať pomocou rekurentného vzťahu $P_{n,m} = P_{n-m,m} + P_{n,m-1}$. Pripomeňme, že $P_{n,1} = 1$ a $P_{n,m} = P_{n,n}$, ak $m \geq n$; dodefinujeme $P_{n,0} = 0$.

Pomocou toho ale ľahko vieme určiť prvý sčítanec partície. Rovnakým spôsobom určíme aj ostatné sčítance rozkladu. Pri vhodnej realizácii má algoritmus zložitosť $O(n^2)$.

1232. Skúste využiť súčet všetkých kartičiek, prípadne operáciu xor.

1233. Použite Dijkstrov algoritmus (úloha 523 alebo [38]).

1234. Cez bod preložme priamku rovnobežnú s osou x a spočítajme počet tých priesečníkov s hranami mnohouholníka, ktoré ležia napravo od nášho bodu. Bod leží v mnohouholníku práve vtedy, keď je tento počet nepárny. Dajte si pozor na priesečníky s vrcholmi mnohouholníka. Zložitosť algoritmu je $O(n)$ pre jeden bod.

V prípade, že predpokladaný počet parašutistov je veľký, môžeme mnohouholník predspracovať tak, aby sa krajina, do ktorej parašutista dopadol, dala pomocou binárneho vyhľadávania určiť v čase $O(\log n)$.

1235. V zásade je možné použiť dva možné prístupy: každé okno si pamätá svoj obsah, alebo každé okno si pamätá obsah, ktorý prekrylo. Omnoho logickejší a prudko odporúčaný je prvý spôsob. Potom stačí, aby každé okno malo metódu „vykresli svoj obsah“. Aby obrazovka „neblikala“ pri prekresľovaní okien, je vhodné použiť double buffering: Okná sa „vykresľujú“ do poľa v pamäti. Až po tom, ako sa vykreslia všetky, vypíšeme obsah poľa na obrazovku. Stačí vždy vypísať tie znaky, ktoré sa od posledného zobrazenia zmenili.

1241. Najlepšia permutácia je taká, pri ktorej je najmenší spoločný násobok dĺžok nsn jej cyklov maximálny. Stačí uvažovať cykly s dĺžkou, ktorá je mocninou prvočísla alebo 1, pričom dĺžky každých dvoch cyklov sú nesúdeliteľné. Dá sa použiť dynamické programovanie: Nech $t[i]$ je maximálny čas trvania, ak uvažujeme iba i blšiek. Ak sme už

vypočítali $t[i]$ pre všetky $i < j$, potom $t[j]$ vypočítame ako $\max\{p^k \cdot t[j-p^k] \mid 1 \le p^k \le j\}$, kde p je 1 alebo prvočíslo.

Časová zložitosť je $O(n\pi(n)\log n)$, kde $\pi(n)$ je počet prvočísel menších ako n. Keďže $\pi(n) \sim n/\log n$, máme kvadratický algoritmus. Pamäťová zložitosť je lineárna.

1242. Pomocou zariadenia zo zadania vieme samostatne utriediť ceruzky na nepárnych pozíciách a ceruzky na párnych pozíciách. Ak teda chceme overiť, či sa dá usporiadať celá postupnosť ceruziek, utriedime v poli dĺžok čísla na párnych pozíciách, utriedime čísla na nepárnych pozíciách a skontrolujeme, či je výsledné pole utriedené.

1243. V reči teórie grafov (ľudia sú vrcholy a výstrely sú hrany) ide o zistenie, či je daný graf súvislý bipartitný. To je vtedy, keď vieme všetky vrcholy jednoznačne (až na výber farieb) ofarbiť dvoma farbami tak, aby žiadne dva susedné vrcholy nemali tú istú farbu. Úlohu vieme riešiť prehľadávaním do hĺbky alebo do šírky. Začneme v jednom (ľubovoľnom) vrchole a striedavo ofarbujeme vrcholy dvoma farbami. Ak na konci nie sú ofarbené všetky vrcholy, graf nie je súvislý – máme `MÁLO INFORMÁCIÍ`; ak počas ofarbovania zistíme, že dva susedné vrcholy majú rovnakú farbu, graf nie je bipartitný – máme `CHYBNÉ INFORMÁCIE`. Podrobnejšie riešenie je napr. v [38].

1244. Dieliky n-trisu si môžeme pamätať ako zoznam políčok usporiadaný najskôr podľa y-ovej, potom podľa x-ovej súradnice (po riadkoch) plus ten istý zoznam usporiadaný podľa x a potom podľa y *naopak* (po stĺpcoch *zdola nahor*) – to nám umožní dieliky ľahko otáčať a teda aj porovnávať.

Dieliky n-trisu je dobré generovať z už vytvorených dielikov $(n-1)$-trisu. Výhodné je tiež rozdeliť dieliky do kategórií, napr. podľa rozmerov najmenšieho obdĺžnika, v ktorom dielik leží. Dieliky n-trisu ležiace v obdĺžniku $A \times B$ môžu totiž vzniknúť iba z dielikov $(n-1)$-trisu ležiacich v obdĺžnikoch $(A-1) \times B$, $A \times (B-1)$, alebo $A \times B$, pričom v prvých dvoch prípadoch treba pridávať štvorček na vhodný kraj, v poslednom prípade treba skúsiť všetky možnosti vnútri obdĺžnika.

1245. Cifry čísla si budeme pamätať v poli (v 10-kovej sústave); navyše si budeme pamätať znamienko čísla a jeho dĺžku. Potrebné algoritmy poznáme zo základnej školy. Stačí vedieť počítať s kladnými číslami a znamienka riešiť samostatne, napr. $(-a)+(-b) = -(a+b)$, $(-a)-(-b) = b-a$, $(-a)(-b) = ab$ a pod.

1311. Použite Floydov-Warshallov algoritmus z riešenia úlohy 823.

1312. Podobne ako v úlohe 1132, všetko treba schovať do rekurzie. Môžeme mať napríklad procedúry A a B. V hlavnom programe iba v cykle voláme procedúry A a B podľa toho, aké písmeno načítame. Ak napríklad načítame `A`, zavoláme procedúru A. Ak procedúra A načíta ďalšie `A`, rekurzívne zavolá A, atď.; ak načíta B, skončí; procedúra B je podobná. Všimnite si, že počet rekurzívnych volaní procedúry bude vždy rovný rozdielu počtov písmeniek `A` a `B`. Teda ak narazíme na medzeru (koniec vstupu) v nejakej procedúre (A alebo B), treba vyhlásiť poplach.

1313. Trasa, ktorú má malý lenivý krtko II nájsť, sa volá *najkratšia Hamiltonovská kružnica*. Jej hľadanie je NP-ťažký problém. To znamená, že v súčasnosti nie je známy polynomiálny algoritmus, ktorý by túto úlohu riešil. Vzorové riešenie je založené na backtrackingu. Zamyslite sa, aké vylepšenia pri preberaní všetkých možností môžete použiť.

1314. Rozšírme polomer stĺpov o polomer valcočloveka a okolo stien „vybudujme“ nové steny takisto vo vzdialenosti polomeru valcočloveka. Cez pôvodnú miestnosť prejde valcočlovek práve vtedy, keď cez transformovanú miestnosť prejde (nekonečne malý) bod. A to sa dá práve vtedy, keď neexistuje množina upravených stĺpov, ktoré tvoria súvislý útvar a spájajú protiľahlé steny. Prehľadávaním do hĺbky alebo do šírky možno získať algoritmus so zložitosťou $O(p^2)$, kde p je počet stĺpov.

1315. Roznásobte výrazy a jednotlivé členy jednoznačne utrieďte. Potom možno výrazy porovnávať priamo.

V praxi sa však používa nasledujúci postup: Zoberme rozdiel oboch polynómov. Chceme overiť, či je ich rozdiel identicky rovný nulovej funkcii. Nech d je stupeň nášho polynómu. (Stupeň polynómu viacerých premenných je maximum stupňov jeho členov, stupeň členu je súčtom stupňov premenných v ňom.)

Zvoľme hodnoty premenných náhodne z množiny $\{0, \ldots, 2d-1\}$ a dosaďme ich do polynómu. Ak nám vyjde čokoľvek iné ako nula, máme istotu, že náš polynóm nulový nie je. Ak nám vyjde nula, budeme si zatiaľ myslieť, že je náš polynóm nulový, a celý experiment zopakujeme. Po vopred zvolenom počte opakovaní skončíme a prehlásime polynóm za nulový.

Úspech tohto algoritmu je založený na tvrdení: Ak náš polynóm nie je identicky rovný nule, tak šanca, že bude nulový pre vyššie popísaným spôsobom zvolené náhodné hodnoty, je najviac $1/2$. To znamená, že ak náhodný výber zopakujeme n-krát, dáme zlú odpoveď s pravdepodobnosťou nanajvýš 2^{-n}. V praxi bohato stačí zvoliť $n = 50$.

1321. Ukážeme niekoľko rôznych riešení.

a) Najjednoduchšie je pre každé kráľovstvo prejsť všetky ostrovy a zistiť, či doň patria. Výhodné, keď je málo ostrovov a kráľovstiev.

b) Predpočítame si počet ostrovov v štvorčekoch 1×1. Potom stačí prebehnúť tie štvorčeky, ktoré tvoria kráľovstvo. Treba vyriešiť zarátanie ostrovov, ktoré ležia na hraniciach štvorčekov.

c) Keď je ostrovov aj kráľovstiev veľa, je šikovné spočítať $p(x, y)$ – počet ostrovov v kráľovstvách s vrcholmi $[0, 0]$ a $[x, y]$. Kráľovstvo s vrcholmi $[x_1, y_1]$ a $[x_2, y_2]$ obsahuje aspoň $p(x_2, y_2) - p(x_1, y_2) - p(x_2, y_1) + p(x_1, y_1)$ ostrovov. Nezarátali sme ostrovy ležiace na ľavej a hornej strane kráľovstva. Tie sa dajú zarátať ľahko rovnakou technikou, stačí si spočítať počet ostrovov v pásikoch s celočíselnými súradnicami.

1322. Priamočiare, ale nevhodné, je použitie triediaceho algoritmu založeného na vzájomnom porovnávaní hodnôt prvkov. V najlepšom prípade totiž dostaneme algoritmus vyžadujúci $O(n \log n)$ operácií. Využijeme triedenie založené na inom princípe, napríklad RadixSort:

for $i := 1$ **to** d **do**
  utrieď stabilným algoritmom podľa i-tej najmenej významnej cifry

Tento algoritmus korektne utriedi čísla, ktoré majú najviac d cifier. Teraz si všimnite, že každé číslo z intervalu 0 až $n^2 - 1$ možno zapísať ako najviac dvojciferné číslo v číselnej sústave so základom n. Ak vieme n čísel utriediť použitím približne n operácií, sme hotoví. Vďaka rozsahu čísel to vieme napríklad využitím CountSortu. Pre každú hodnotu si spočítame, koľko je vo výslednom usporiadaní pred ňou prvkov, čo vieme zistiť na jeden prechod údajmi. Potom už iba stačí umiestniť prvky na ich správne miesto.

1323. Tento problém je NP-ťažký. To znamená, že v súčasnosti nie je známy algoritmus, ktorý by úlohu riešil v polynomiálnom čase (a ak P $\neq$ NP, taký algoritmus ani neexistuje). Preto treba dávať pozor na *chybné* heuristiky, ako napríklad výber možnej linky s najmenšou periódou. Vzorovým riešením je backtracking. Vždy pre prvý deň skúsime všetky možné intervaly, „pokryté“ dni vymažeme a rekurzívne hľadáme intervaly dostavníkov pre zvyšné dni.

1324. Máme dve rozumné možnosti, z nich musíme overiť, ktorá je najkratšia: 1. obídeme (skoro) celý kruh jedným smerom; 2. najskôr chvíľu točíme do jednej strany, potom už až do konca točíme do opačnej strany.

1325. Vhodnou metódou generovania prejavov je „zadrôtovať“ do svojho programu tzv. bezkontextovú gramatiku, ktorá popisuje štruktúru prejavu. Samotný prejav skladáme

z vopred vybraných slov a fráz, pričom gramatika popisuje, ako ich máme spájať dokopy, aby dávali (aký-taký) zmysel.

Uvedieme niekoľko najlepších ukážok z prejavov, ktoré poslali riešitelia tejto úlohy. Teda presnejšie, prejavov, ktoré vygenerovali ich programy. Citujúc jedného z riešiteľov, *„Autor nezodpovedá za politické názory tohto programu."*

Vážení súdruhovia a poslanci Národnej Rady. ⟨NSVPS⟩ *Situácia v našej vlasti je výborná, všetko funguje a nie som si vedomý žiadnych ťažkostí.* ⟨NSVPS⟩ ⟨NSVPS⟩ ⟨NSVPS⟩

(...) Podpredseda robí demokraciu. 8–(8–(8–(*Prečo ste mi nedali džús!!?* ⟨DP⟩ *Námestník robí demokraciu (...)*

Milé dámy a vážení páni! Sme vždy pripravení brániť našu vlasť a jej demokraciu. Naša strana vám zaručí skvelý zajtrajšok. Splníme všetky naše predsavzatia. ⟨NSVPS⟩ ⟨DP⟩ *Nech vás nemýli, že sa voláme ***. My sme vaši.* 8–(*A ak nás nebudete počúvať, bude vám veľmi zle. Tak zle ako vám ešte nikdy nebolo. A teraz vám poviem niečo z tej veselšej stránky vládnutia. Je to zložité povolanie. Vlastne to ani nie je povolanie, je to služba. Je dobre platená, hoci... aj tak sme len zamestnanci štátu. A vy viete, ako sú štátni zamestnanci platení. Ale za to môžu tí hore. Dovidenia a dopočutia.*

Vážení občania,

srdce mi zviera, keď vidím tú ľudskosť, tú nekonečnú oddanosť našim spoločným ideálom. Ľudia tejto krajiny sú pre nás prvoradí, dali nám svoje hlasy a my ich nesklameme. Oslovujem vás, lebo viem, že mi rozumiete.

Áno, svoje si nedáme, cudzie nechceme. Urobíme, čo bude v našich silách. My voličov nesklameme! Na Slovensku bude mať každý pracovitý občan prácu, máte moje čestné slovo! Tak ako tu pred vami stojím, svoju povinnosť voči občanom si splníme. Nikto nevie lepšie ako my, Slováci, čo je to zakúšať porobu po celé stáročia. Ale my sme sa poučili a ručíme, že história sa opakovať nebude! Koľko vody pretieklo, kým sa splnili naše odveké túžby, nebuďme ľahostajní. Ľuďom s ktorými som sa u nás stretol nechýba srdce, pevné ruky alebo odvaha, naši ľudia vedia, čo je to život. Neoddávajme sa malichernostiam, myslime na našu budúcnosť, myslime na budúcnosť našich detí. (...) Naša strana vám zabezpečí radostnú mladosť, činorodý život a pokojnú starobu. My sme tento boj nezačali, ale pevne veríme, že s vašou pomocou – občania – ho ukončíme.

1331. Riešenie je založené na tvrdení, že Tomáš sa pri optimálnom zápise môže zapísať na najskôr končiacu prednášku. Algoritmus je teda jasný: zotriedime prednášky podľa toho, kedy končia, vyberieme z nich prvú a zrušíme tie, ktoré sa s ňou prekrývajú, atď. Časová zložitosť pri dobrej realizácii je $O(n \log n)$.

1332. Najjednoduchšie je postupne skúšať, či daný vzťah platí v sústave so základom $2, 3, \ldots$. Nevýhodou je, že ak by sme nemali obmedzenie na veľkosť cifier, takýto program by v prípade, že úloha nemá riešenie neskončil.

Lepšie je takéto riešenie: Nech $A = a_n z^n + a_{n-1} z^{n-1} + \cdots + a_0$, $B = b_m z^m + b_{m-1} z^{m-1} + \cdots + b_0$ a $C = c_l z^l + c_{l-1} z^{l-1} + \cdots + c_0$. Ak pre všetky i je $a_i + b_i = c_i$, výraz platí vo všetkých sústavách so základom väčším ako je najväčšia cifra C (teda $z > \max\{c_k \mid 0 \leq k \leq l\}$). V opačnom prípade nech i je najmenšie také, že $a_i + b_i \neq c_i$. Potom musí byť $z = a_i + b_i - c_i$; treba iba overiť, či z vyhovuje.

1333. Najjednoduchší prípad je, ak majú všetky veže navzájom rôzne x-ové aj y-ové súradnice. Vtedy je riešením, teda y-ovou súradnicou veľkej rúry, medián (stredný prvok) y-ových súradníc veží. Totiž, ak by sme veľkú rúru posunuli trebárs vyššie, tak ju prisunieme bližšie nanajvýš k toľkým vežiam, od koľkých ju odsunieme ďalej.

Vo všeobecnosti to bude vyzerať takto: Pre každú x-ovú súradnicu, na ktorej leží aspoň jedna veža, si zjavne stačí pamätať vežu s najväčšou a vežu s najmenšou y-ovou súradnicou – ak pripojíme tieto dve, tie medzi nimi už určite budú pripojené.

Takto dostaneme $2k$ veží. (Pre x-ové súradnice, na ktorých je len jedna veža, zoberieme tú istú vežu dvakrát.) Skúste si dokázať, že hľadanou súradnicou veľkej rúry je medián y-ových súradníc týchto $2k$ veží.

Ako efektívne hľadať medián? Hoareho algoritmus podobný QuickSort-u hľadajúci k-ty najmenší prvok je založený na myšlienke preusporiadať pole tak, aby naľavo od zvoleného prvku (pivota) boli prvky od neho menšie a napravo od neho väčšie. Nech je po preusporiadaní pivot na i-tom mieste. Ak $i = k$, potom je pivot hľadaným prvkom. Ak $i > k$, potom je k-ty najmenší prvok medzi prvkami od 1 po $i - 1$. A ak $i < k$, potom hľadáme $(k - i)$-ty prvok medzi prvkami od $i + 1$ po n. Teda sme zmenšili prehľadávanú dĺžku poľa a riešime tú istú úlohu. Tento algoritmus má v priemernom prípade časovú zložitosť $O(n)$, v najhoršom ale $O(n^2)$.

Načrtneme tiež myšlienku algoritmu s časovou zložitosťou $O(n)$ aj v najhoršom prípade. Finta spočíva v lepšom výbere prvku, podľa ktorého rozdeľujeme pole na dva menšie úseky. Pole najskôr rozdelíme na päťprvkové postupnosti a za pivota zvolíme medián postupnosti mediánov päťprvkových postupností (medián päťprvkovej postupnosti vieme určiť v konštantnom čase, na určenie mediánu mediánov použijeme tento istý algoritmus, ale už iba na $n/5$ prvkov). Po preusporiadaní poľa podľa takto určeného pivota je ľahko vidno, že väčší z úsekov naľavo od pivota alebo napravo od pivota má najviac $3n/4$ prvkov. Ak $T(n)$ označíme čas výpočtu nášho algoritmu, potom podľa vyššieuvedeného platí $T(n) \leq c_1 n + T(n/5) + T(3n/4)$, z čoho vyplýva, že $T(n) = O(n)$.

Podrobnejšie riešenie nájdete napr. v [47].

1334. Nájsť optimálne riešenie, ktoré by v najhoršom možnom prípade potrebovalo minimálny počet otázok, je ťažký problém. Nevieme o principiálne lepšom riešení ako použitím backtrackingu skúšať všetky rozumné možnosti, čo vytočiť.

Iné riešenie: Ak v našom telefónnom zozname chýba viac ako jedna cifra, nebudeme úlohu vedieť vyriešiť. V opačnom prípade to určite ide.

Budeme postupne vytáčať čísla, pričom pri výbere čísla budeme chcieť maximalizovať pravdepodobnosť, že sa dovoláme (a niečo nové tým zistíme). Pre konkrétne vytáčané číslo túto pravdepodobnosť ľahko spočítame: Prejdeme cez zoznam a pre každé číslo, z ktorého ešte nevieme vytočiť všetky cifry, si spočítame, aká je šanca, že chýbajúce cifry uhádneme správne. Keďže sa nemôžeme dovolať na dve rôzne čísla naraz, jednotlivé pravdepodobnosti stačí sčítať.

Ešte lepším riešením by bolo maximalizovať očakávanú veľkosť informácie, ktorú vytočením daného čísla získame. Počítanie tejto hodnoty vyžaduje znalosť pojmu entropia a súvisiacej teórie, ktorá vysoko presahuje rámec stredoškolských osnov.

1335. Úlohu vieme riešiť v lineárnom čase – stačí si všimnúť, že celočíselných uhlov od 0 po 360 je konštantne veľa.

1341. Vystačíme si s tromi pomocnými súbormi, ktoré budeme prepisovať. Predpokladajme, že máme vstupný súbor, v ktorom je za sebou napísaných niekoľko postupností, z ktorých každá je utriedená a (možno až na poslednú) každá má dĺžku aspoň 2^k. (Na začiatku toto zjavne platí pre $k = 0$.) Teraz tento súbor budeme čítať a rozdelíme jeho prvky do dvoch nových súborov nasledovne: Prvé číslo zapíšeme do prvého nového súboru. Pre každé ďalšie číslo: Ak je väčšie alebo rovné ako predchádzajúce, zapíšeme ho do toho istého súboru, inak do opačného. Takto dosiahneme to, že striedavo rozhádžeme utriedené postupnosti do dvoch súborov. Teraz zmažeme pôvodný súbor a ideme spájať naše dva pomocné do jedného. Vždy spojíme jednu postupnosť z jedného a jednu z druhého pomocného súboru, podobne ako pri triedení MergeSort, do jednej utriedenej postupnosti. Takto dostaneme súbor, v ktorom už majú utriedené postupnosti dĺžku aspoň 2^{k+1}. Po $\lceil \log_2 n \rceil$ opakovaniach vyššie uvedeného postupu musí teda súbor byť utriedený. (Pozri napr. triedenie dvojcestným zlučovaním v [46].)

1342. Ukážeme niekoľko riešení tejto úlohy.

a) Jednoduché ale pomalé riešenie je použiť postupné presúvanie druhých dielov b_i, $i = 1, 2, \ldots, n$, na svoje miesto. Vyžaduje to $O(n^2)$ operácií. Rovnako náročné je aj riešenie využívajúce výmenu párnych prvých dielov s nepárnymi druhými dielmi, čím sa dostaneme prvé a druhé diely rovnakých kníh k sebe. Dvojice nebudú ešte všetky na svojich miestach. Zvyšné knihy vymieňame tak, aby sme zmenšovali počet dvojíc, ktoré nie sú na svojom mieste. Treba rozlišovať prípady, keď je dvojíc párny a nepárny počet.

b) Lepšie riešenie je využitím techniky rozdeľuj a panuj. Knihy $a_{p+1}, \ldots, a_n,\ b_1, \ldots, b_p$, kde $p = n$ div 2, posunieme cyklicky o p miest vpravo, pozri úlohu 123 (ak je n párne, obe časti stačí vzájomne vymeniť). Dostali sme dve časti, medzi ktorými sa už knihy vymieňať nebudú. Každú z častí rovnakým spôsobom usporiadame. Posunutie n kníh vieme realizovať $O(n)$ operáciami, takže celkovo vystačíme s $O(n \log n)$ operáciami.

c) Existuje ešte efektívnejšie riešenie, ktoré je založené na myšlienke: Budeme pole prechádzať zľava doprava a vždy vymeníme knihu, ktorá na dané miesto patrí, s knihou, ktorá tam práve je. Náročnou časťou je samozrejme určiť, kde sa práve nachádza kniha, ktorá na práve spracúvané miesto patrí.

1343. Možno riešiť pomocou dynamického programovania. Pre každé písmeno vety a pre každý jeho výskyt na náramku vieme z toho istého údaju pre predchádzajúce písmeno a všetky jeho umiestnenia na náramku spočítať minimálny počet otočení.

1344. Pozri riešenie úlohy 433.

1345. Jediným problémom pri riešení je uvádzanie zlomkov do základného tvaru – na to použite Euklidov algoritmus. Vnútorne v rámci operácií je potrebné zlomky uchovávať s dvojnásobnou presnosťou (napríklad ak sa bežne uchovávajú v premenných typu **integer**, treba použiť na medzivýsledok **longint**).

1411. Riešenie so zložitosťou $O(n^2)$ je pomerne jednoduché. Stačí si uvedomiť, že v každej triangulácii mnohouholníka $v_0, v_1, \ldots, v_{n-1}$ musí existovať taký vrchol v_i, že trojuholník v_{i-1}, v_i, v_{i+1} patrí do triangulácie (voláme ho *ucho*) – to je vtedy, keď uhol prislúchajúci v_i je konvexný (ostrý) a vnútri trojuholníka v_{i-1}, v_i, v_{i+1} neleží žiadny iný vrchol mnohouholníka. Toto vieme overiť lineárnym prebehnutím všetkých vrcholov. Ak vnútri niekto leží, pokračujeme v overovaní vrcholu v_{i+1}. Ak ho môžeme odrezať, urobíme tak. V tomto prípade pokračujeme vrcholom v_{i-1}, lebo by sa mohlo stať, že uhol pri ňom sa stal konvexným. Pri takejto stratégii máme zaručené, že budeme testovať najviac $2n$ vrcholov. V [4] je uvedený algoritmus zložitosti $O(n \log n)$; existuje dokonca lineárny algoritmus (Bernard Chazelle, roku 1991).

1412. Poskladaná dážďovka s k článkami sa dá popísať trojicou čísel a, b, c – súradnica najľavejšieho bodu, konca dážďovky a jej najpravejšieho bodu ($a \le 0 \le c$; $a \le b \le c$). Nech pre všetky a, b obsahuje $t[k, a, b]$ najmenšie c také, že existuje poskladanie k článkov dážďovky popísaným spôsobom. Potom nie je problém vypočítať $t[k+1, a, b]$ pre všetky a, b. Má zmysel uvažovať $|a|, |b| \le 2 \cdot d$, kde d je dĺžka najdlhšieho úseku dážďovky. Stačí si pamätať $t[k, a, b]$ len pre posledné dve hodnoty k.

Lepšie riešenie: Každú poskladanú dážďovku si navyše môžeme posunúť tak, aby súradnica jej najľavejšieho bodu bola 0. Takto dostávame riešenie s časovou zložitosťou $O(Nd)$.

1413. Preveďme si úlohu na problém z teórie grafov: stromy a rieky sú vrcholy, hrana medzi dvomi vrcholmi vedie práve vtedy, ak sa medzi nimi nedá pretlačiť klavír, (stromy, ktorých rozdiel x-ových súradníc je menší ako x-ový rozmer klavíra a zároveň rozdiel y-ových je menší ako y-ový rozmer klavíra; rieka a strom, ktorých rozdiel y-ových súradníc je menší ako y klavíra). Tomáš môže prejsť cez les práve vtedy, ak neexistuje v takomto grafe cesta od severnej rieky k južnej.

1414. Významné lávky, ktoré máme nájsť a nahradiť mostmi, sa aj v grafovej terminológii nazývajú *mosty*. Most je teda taká hrana, ktorej odstránením graf prestane byť súvislý.

Mosty vieme nájsť upraveným prehľadávaním do hĺbky. Hlavná myšlienka je nasledujúca: Pre každý vrchol si zapamätajme hranu, ktorou sme ho objavili. Tieto hrany tvoria kostru pôvodného grafu. Zjavne žiadna iná hrana nemôže byť most, lebo po jej odstránení nám ostane celá kostra, a teda graf bude stále súvislý. Ako teraz spoznať, ktoré kostrové hrany sú mosty? Všimnime si ten vrchol hrany, ktorý je ďalej od koreňa (t.j. od vrcholu, kde sme začali prehľadávanie), nazvime ho v. Naša hrana je most práve vtedy, ak žiadna z pôvodných (mimokostrových) hrán nevedie z podstromu s koreňom v von. To totiž znamená, že po odstránení našej hrany sa nebude dať z v dostať do koreňa.

Inou možnosťou je previesť túto úlohu na iný, podobný problém. Do stredu každej hrany dajme nový vrchol. Tým sa nám hrana rozpadne na dve menšie. Teraz nám vlastne stačí zistiť, či existuje nejaký nový vrchol, ktorého odobratím by sa nám graf rozpadol. Vrcholy, ktorých odobratím graf rozpadne, sa nazývajú *artikulácie*, dajú sa hľadať podobným postupom ako mosty. Viac informácií o oboch algoritmoch nájdete napríklad v [3.B v 38].

1415. Každý počítač akoby na vlastnú päsť prehľadával celú sieť do šírky. Zostavuje si postupnosť zoznamov počítačov, vzdialených l prerútovaní ($l = 0, 1, \ldots, \mathrm{diam}(G)$). Pre $l = 0$ je v ňom len sám. Vždy, keď má zoznam pre nejaké l hotový, pošle ho všetkým susedom a očakáva ich zoznamy. Z týchto zoznamov skonštruuje svoj zoznam pre $l + 1$. Opakuje, dokiaľ sa dozvedá nové mená, potom vie, že pozná všetkých v sieti a skončí. Pri implementácii treba dávať pozor na to, že kým od nejakého suseda ešte neprišiel zoznam pre l, od iného mohol prísť zoznam pre $l + 1$, ktorý treba odložiť pre neskoršie použitie.

1421. Sú dva možné prístupy, oba so zložitosťou $O(n^2)$. Prvý skúma graf popisujúci nakreslené čiary, na určenie počtu kusov koláča potrebujeme v tomto grafe (okrem iného) nájsť mosty a artikulácie. Nesmieme však zabudnúť na pôvodné zakreslenie v rovine – môže sa vyskytnúť napr. „diera v diere“.

Druhý prístup rozoberieme trochu podrobnejšie. Vytvoríme „obdĺžnikovú sieť“ tak, že každú úsečku predĺžime až po okraj koláča. Teraz nám vznikol nový graf, v ktorom vrcholy sú jednotlivé obdĺžniky siete a myslenou hranou spojíme každú dvojicu susediacich obdĺžnikov, ktoré nie sú oddelené nakreslenou čiarou. Potom stačí (napríklad prehľadávaním do šírky) vyfarbiť okrajové časti koláča. Ostanú nám neofarbené práve všetky diery. Teraz už dovolíme ísť aj cez nakreslenú čiaru, ale nie cez políčka, ktoré sme už ofarbili, a postupným ofarbovaním zvyšku koláča zistíme počet dier.

1422. Ak sa slovo x nachádza už v slove w, odpovieme `ano` a skončíme. V prípade, že je aj u aj v prázdne, odpovieme `nie` a skončíme.

Ak u je prázdne a v neprázdne, vieme opäť otázku ľahko zodpovedať rozobratím niekoľkých prípadov. Ak w neobsahuje dvojky, v ďalšej sekunde je po knihách. Ak w obsahuje k dvojok, pozrime sa na počet dvojok vo v, ten označme l. Rozmyslite si, že ak $l \leq 1$, stačí overiť, či sa x nachádza v l-krát za sebou napísanom v. Ak $l > 1$, bude sa postupnosť kníh na polici rozrastať, v tomto prípade stačí overiť, či sa x nachádza v dostatočne veľakrát za sebou napísanom v. Toto vieme spraviť nejakým klasickým algoritmom na vyhľadávanie v texte. Podobne prípad, ak je prázdne v. Ostalo nám to najzaujímavejšie, prípad s neprázdnymi u a v.

Všimnime si, že v žiadnom kroku sa počet kníh nezmenší. Nech n je počet písmen v x. Všimnime si prvý okamih, kedy sa niekde zjavilo x. Toto x muselo byť celé vytvorené prepísaním nejakých najviac n po sebe idúcich znakov z predchádzajúceho slova. A tieto znaky zase vznikli prepísaním predchádzajúcich najviac n znakov, atď.

Ak sa teda slovo x dá nejako vytvoriť, určite sa dá vytvoriť nasledujúcim postupom: Po každom prepísaní sa pozriem, či aktuálna postupnosť kníh nie je dlhšia ako n. Ak je, niektoré knihy zo začiatku, prípadne aj konca, zahodíme a necháme si ich len n.

Prehľadávaním do šírky nájdeme všetky n-tice písmen, ktoré vieme týmto postupom dosiahnuť, odpovieme podľa toho, či je medzi nimi x. Časová aj pamäťová zložitosť je $O(N2^N)$.

1423. Utriedením polôh vieme dosiahnuť zložitosť $O(n \log n)$. Lineárne riešenie je založené na Dirichletovom princípe (Pigeon-Hole Principle): Najväčšia vzdialenosť dvoch susedných vrabcov je aspoň $(max - min)/(n-1)$, kde min, resp. max je vzdialenosť najbližšieho, resp. najvzdialenejšieho vrabca od dediny. Ak by najväčšia vzdialenosť dvoch susedných vrabcov bola menšia, potom by boli najvzdialenejší a najbližší vrabec vzdialení menej ako $max - min$, čo je spor.

Drôt rozdelíme na neprekrývajúce sa úseky o veľkosti $(max - min)/n$, čo je určite menej ako hľadaná vzdialenosť. Zjavne teda hľadaní dvaja vrabci sedia v rôznych úsekoch. Postupne spracúvame vrabcov, pričom o každom úseku si pamätáme, či je prázdny, a ak nie, ktorý vrabec je v ňom „najviac vľavo“ a ktorý „najviac vpravo“. Po spracovaní všetkých vrabcov už len prejdeme „zľava doprava“ po úsekoch a nájdeme riešenie. Presnejšie, pre každý neprázdny úsek vypočítame vzdialenosť „najpravejšieho“ vrabca z neho a „najľavejšieho“ vrabca z nasledujúceho neprázdneho úseku smerom „napravo“.

1424. Možným riešením je Fordov-Fulkersonov algoritmus [6 v 38, 8 v 8, 16 v 27] na hľadanie maximálneho toku v sieti so zložitosťou $O(nm)$. Označme f_{ij} veľkosť toku medzi stanicami i a j, teda f_{ij} je 1, ak tam rum tečie, je 0, ak netečie a je -1, ak tečie v opačnom smere. Pre každú dvojicu staníc i, j, ktoré sú spojené rúrou spočítame *rezervu* rúry, t.j. o koľko rumu viac sme schopní prepraviť rúrou z i do j oproti súčasnému stavu. Rezerva je teda $r_{ij} = 1 - f_{ij}$.

Na začiatku pre každú rúru z i do j platí $f_{ij} = 0$. Začneme zo stanice s hľadať cestu do stanice t po rúrach s nenulovou rezervou. Ak nájdeme takúto *rezervnú cestu*, každej rúre na ceste zvýšime f_{ij} o 1, čím sa celkové množstvo prepraveného rumu zvýši o 1. Takto zväčšujeme tok dovtedy, kým ešte existuje nejaká rezervná cesta. Nájdenie rezervnej cesty trvá $O(m)$ a veľkosť toku je nanajvýš $n-1$, celková časová zložitosť je teda $O(nm)$.

Existujú aj efektívnejšie algoritmy na hľadanie maximálneho toku.

1425. Za šéfa budeme voliť počítač, ktorého meno je posledné v abecede. Ukážeme si dve riešenia.

a) Každý počítač pošle jedným smerom svoje meno. Každú správu, ktorá mu príde, posiela ďalej. Pritom ak obsahovala meno, ktoré je väčšie ako najväčšie jemu doteraz známe meno, zapamätá si ho. Keď mu príde jeho vlastné meno, vie, že už prečítal všetky mená, a skončí. Tento postup má (ak predpokladáme, že všetky linky sú približne rovnako rýchle) časovú zložitosť $O(n)$, ale rozpošle $\Theta(n^2)$ správ. Všimnite si, že nepotrebuje obojsmernú sieť – stačí vedieť posielať správy jedným smerom.

b) (Algoritmus Hirschberga a Sinclaira, 1980.) Každý počítač bude fungovať vo fázach. V k-tej fáze sa chce presvedčiť, že spomedzi 2^k počítačov naľavo a 2^k počítačov napravo od neho je on najväčší. Ak sa mu to podarí overiť, ide na ďalšiu fázu. Ak zistí, že v aktuálnom okolí najväčší nie je, od tohto okamihu už len pasívne posiela správy, ktoré idú cez neho. A posledná možnosť: akonáhle niekto zistí, že jeho k je už také veľké, že jeho okolie je celý kruh, vyhlási sa za šéfa a oznámi to celému kruhu.

   Presnejšie, algoritmus pre každý počítač vyzerá nasledovne: Na začiatku je aktívny a $k = 0$. Každú fázu začneme tým, že v oboch smeroch pošleme správu „volám sa ID a chcem byť šéf, pošli toto ďalším $2^k - 1$ počítačom“. Potom čakáme na odpovede z oboch smerov (a počas tohto čakania reagujeme na správy, ktoré cez nás posielajú ostatní). Ak dostaneme z niektorého smeru odpoveď „ID, už nie si šéf“, od tohto okamihu už

nič nerobíme, len preposielame cudzie správy. Ak dostaneme z oboch smerov odpoveď „ID, naďalej si šéf“, zväčšíme si k o 1 a začneme ďalšiu fázu.

Reakcia na správy vyzerá nasledovne: Nech nám prišla správa „volám sa ID_2 a chcem byť šéf, pošli toto ďalším d počítačom“.

Potom sú nasledujúce možnosti:

– Ak $ID_2 < ID$, teda naše ID je väčšie, autor správy celkovým šéfom nie je. Rovno mu túto smutnú správu oznámime – pošleme späť správu „ID_2, už nie si šéf“.

– V opačnom prípade môžeme už odteraz byť pasívni, keďže vieme, že šéfom nie sme. (Nemusíme to spraviť, on by nám to počítač ID_2 časom na našu otázku oznámil, ale prečo nevyužiť túto informáciu?)

Ak $d > 0$, pošleme správu ďalej v tom istom smere, len s o 1 menším d. V opačnom prípade potešíme autora správy, že v celom jeho okolí na tejto strane je najväčší – pošleme späť správu „ID_2, naďalej si šéf“.

Ak som globálnym šéfom, zistím to skôr či neskôr tak, že mi z niektorej strany príde moja vlastná správa „volám sa ID a chcem byť šéf, ...“.

Fázu k dosiahne nanajvýš $n/2^k$ počítačov, lebo každé dva počítače, ktoré dosiahli fázu k, musia mať medzi sebou aspoň 2^{k-1} iných. Akonáhle niekto dosiahne fázu (približne) $\log_2 n$, stáva sa globálnym šéfom a celý protokol končí.

Pozrime sa teraz na počítač, ktorý vykonáva svoju fázu k. V najhoršom prípade obe ním poslané správy spravia po 2^k krokov „tam“ a odpovede na ne po 2^k krokov „späť“. Všetky počítače, ktoré dosiahli k-tu fázu, v nej teda dokopy naposielajú najviac $(n/2^k) \cdot (4.2^k) = 4n = O(n)$ správ. Preto pri tomto algoritme je celkový počet správ $O(n \log n)$. Podobnou úvahou sa dá ukázať, že za predpokladu približne rovnako rýchlych liniek potrebuje tento algoritmus lineárny čas.

Burns dokázal, že ľubovoľný algoritmus na voľbu šéfa potrebuje poslať $\Omega(n \log n)$ správ, máme teda asymptoticky optimálne riešenie.

1431. Keďže sa zaujímame iba o priesečníky vnútri kruhu, vypočítame priesečníky priamok s kružnicou a ďalej sa budeme zaoberať iba vzniknutými úsečkami. Jednotlivé koncové body (bod na kružnici je jednoznačne určený uhlom) utriedime. Ak AB je úsečka, predpokladáme, že $A < B$; A voláme začiatok a B koniec úsečky. Úsečky AB a CD, kde $A < C$, sa pretínajú vtedy, keď $C < B < D$.

Pôjdeme postupne po obvode kruhu a v „čiernej krabičke“ si budeme pamätať konce úsečiek (tých, ktoré už začali, ale ešte neskončili). Keď narazíme na začiatok úsečky, pozrieme sa, koľko úsečiek v krabičke sa končí neskôr ako táto úsečka (tento počet pripočítame k počtu priesečníkov) a koniec úsečky pridáme do krabičky. Keď narazíme na koniec úsečky, vyberieme ho z krabičky. Keď prejdeme celý obvod kružnice, máme spočítaný počet všetkých priesečníkov. Ak budeme čiernu krabičku implementovať ako vyvážený binárny strom, jedna operácia sa vykoná v čase $O(\log n)$ a celková časová zložitosť bude $O(n \log n)$.

1432. Vzorový program po načítaní vstupu najprv skontroluje, či je riešenie zjavne nemožné, čo je v prípade, ak je trojuholníkov iný počet, ako počet vrcholov mínus 2, alebo ak je súčet obsahov trojuholníkov iný ako obsah mnohouholníka, alebo ak existuje trojuholník, pre ktorého jednu stranu neexistuje príslušná dĺžka uhlopriečky.

Ak riešenie nevylúčil, postupuje rekurzívne – na kraji mnohouholníka nájde trojuholník, aký má v zozname (že taký musí existovať, ak sa to dá, si iste dokážete sami). Skúsi ho teda odrezať a rekurzívne sa zavolá. Mnohouholník sa trojuholníkmi dá pokryť, ak nakoniec zostane z mnohouholníka trojuholník zhodný s tým, čo zostal v zozname. Ak sa tak nestane po vyskúšaní všetkých možností, trojuholníky sa na plech poukladať nedajú.

Vyššie popísané riešenie má časovú zložitosť $O(p(n) \cdot n!)$, kde $p(n)$ je nejaký vhodný polynóm (podľa toho, ako šikovne hľadáme trojuholník, ktorý odstrihnúť). Za cenu veľkých pamäťových nárokov sa toto riešenie dá zlepšiť na $O(p(n) \cdot 4^n)$. Predstavme si, že vyššie

popísané riešenie naprogramujeme ako rekurzívnu procedúru. Jej vstup môžeme zadať ako dve postupnosti bitov, ktoré hovoria, ktoré trojuholníky a ktoré vrcholy máme ešte k dispozícii. Ak si teda budeme pamätať, s ktorými z 4^n možných parametrov sme už našu procedúru volali, časť práce si ušetríme. (Nemá zmysel druhýkrát spúšťať procedúru s tými istými parametrami. Ak sme sa dostali k druhému jej volaniu, znamená to, že pri jej prvom volaní sme nič nenašli – a teda by sme nič nenašli ani teraz.)

1433. Ak požadujeme, aby sme nepoužili viac ako dvojnásobok minimálneho počtu škatúľ, stačí, aby program vymyslel riešenie, v ktorom budú všetky škatule naplnené aspoň do polovice.

Vzorové riešenie pracuje veľmi jednoducho: zo vstupu číta objemy vecí a postupne ich ukladá do škatule, kým sa tam vojdú. Keď nejaká vec do škatule nevojde, škatuľa sa uzavrie, otvorí sa nová a pokračuje sa v balení. Ak však objem veci, ktorá nevošla do škatule, bol väčší ako $s/2$, otvorená škatuľa sa neuzavrie, ale vec s nadpolovičným objemom sa zabalí do osobitnej škatule, tá sa uzavrie a pokračuje sa v balení do škatule, ktorá bola otvorená predtým.

Nech optimálne balenie potrebuje l škatúľ a nami nájdené riešenie potrebuje k škatúľ. V našom riešení sú všetky škatule (okrem poslednej) zaplnené viac ako z polovice, teda ich celkový objem je viac ako $s(k-1)/2$. V optimálnom balení sa do škatule zmestí najviac s, a preto $l > (k-1)/2$. Avšak l je celé číslo, čiže l je aspoň najmenšie celé číslo (ostro) väčšie ako $(k-1)/2$, čo v matematike zapíšeme ako $l \geq \lfloor (k-1)/2 \rfloor + 1$. Ak je k párne, dostávame $l \geq \lfloor (k-1)/2 \rfloor + 1 = (k-2)/2 + 1 = k/2$, ak je k nepárne, máme $l \geq \lfloor (k-1)/2 \rfloor + 1 = (k-1)/2 + 1 = (k+1)/2$. Teda v obidvoch prípadoch $k \leq 2l$, čo sme chceli dokázať.

Predpokladá sa, že neexistuje polynomiálny algoritmus, ktorý by vždy našiel optimálne rozloženie vecí do škatúľ. Z toho istého predpokladu (že P $\neq$ NP) dokonca vyplýva aj všeobecnejšie tvrdenie: neexistuje polynomiálny algoritmus, ktorý by vždy použil menej ako 3/2 optimálneho počtu škatúľ.

Algoritmus FFD (*first fit decreasing*): Veci utriedime podľa veľkosti. Teraz ich ideme baliť, začínajúc najväčšou. Každú vec zabalíme do prvej škatule, kde je ešte miesto. V roku 1991 Yue dokázal, že ak je optimálny počet škatúľ x, tak tento algoritmus použije najviac $(11/9)x + 1$ škatúľ. (Skúste s využitím tohto faktu dokázať, že algoritmus FFD nikdy nepoužije viac ako 3/2 optimálneho počtu škatúľ.)

Zdalo by sa teda, že algoritmus FFD je najlepší možný. Nie je tomu úplne tak. Ukazuje sa, že najťažším problémom je, ak máme veľa vecí, ktoré vyzerajú, že by sa mohli zmestiť do dvoch škatúľ. V situáciách, kde budeme potrebovať použiť veľa škatúľ, sa nám môže (a bude) dariť lepšie.

V roku 1982 Karmarkar a Karp publikovali algoritmus, ktorý pre ľubovoľný vstup, pre ktorý treba dostatočne veľa škatúľ, použije nanajvýš $(1+\varepsilon)$-násobok optimálneho počtu škatúľ. Časová zložitosť tohto algoritmu je $O(n \log n \varepsilon^{-8})$. Preložené do slov: čím presnejší výsledok chceme, tým dlhšie musíme počítať `:-)`.

Neskôr bol na základe podobných myšlienok vytvorený polynomiálny algoritmus, ktorý zaručuje, že oproti optimálnemu počtu škatúľ x použije len o $O(\log^2 x)$ škatúľ viac. Podrobnejšie informácie sa dajú nájsť v [37].

1434. Úlohou je nájsť ofarbenie planárneho grafu piatimi farbami tak, aby žiadne dva vrcholy spojené hranou nemali rovnakú farbu. Pri načítavaní si spravme zoznam všetkých vrcholov stupňa nepresahujúceho 5 (pre planárne grafy z Eulerovej vety vyplýva, že $e \leq 3v - 3$; ak by mali všetky vrcholy stupeň aspoň 6, graf by mal aspoň $3v$ hrán, čo je spor; preto je tento zoznam neprázdny). Tento zoznam budeme udržiavať tak, aby v ňom boli vždy všetky nespracované vrcholy stupňa najviac 5, avšak nebudeme požadovať, aby sa každý vrchol vyskytoval iba raz, alebo aby tam neboli iné vrcholy.

Spracovať vrchol stupňa menšieho ako 5 znamená: zapamätať si všetkých jeho susedov, vymazať ho z grafu (ak pri mazaní hrán klesne nejakému vrcholu stupeň na 5, treba ho

zaradiť do zoznamu), rekurzívne spracovať zvyšok grafu, vrátiť ho a ofarbiť ho farbou, ktorá sa nevyskytuje u jeho susedov.

Pri spracovávaní vrcholu v stupňa 5 musíme dosiahnuť, aby dvaja z jeho susedov mali rovnakú farbu. Iste existujú susedia i a j vrcholu v, ktorí nie sú spojení hranou (inak by graf nebol planárny – obsahoval by K_5). Zmažme vrchol v a vrcholy i, j skontrahujme. Takýto skontrahovaný graf bude opäť planárny. Rekurzívne ho ofarbíme, v_{ij} rozbijeme naspäť na i a j a vidíme, že vrchol v susedí s najviac štyrmi rôznymi farbami. Keď je zoznam nespracovaných vrcholov stupňa nanajvýš 5 prázdny, máme ofarbený celý graf.

1435. Oba krajné počítače začnú číslovanie. Sebe priradia jednotku, susedovi pošlú dvojku a každý ďalší počítač pošle susedovi, od ktorého číslo nedostal, číslo o jedna väčšie. Takto číslovania postupujú z oboch strán, až sa v istom okamihu stretnú. Napríklad pri siedmich počítačoch to bude vyzerať takto: $\begin{smallmatrix}1&2&3&4&5\\&&3&2&1\end{smallmatrix}$. Vidíme, že jedna strana od miesta stretnutia by si čísla mohla ponechať, ale druhú ešte musíme prečíslovať. To zariadime takto: Keď počítač dostane druhé číslo, porovná ho s prvým. Ak je menšie alebo rovné, pošle ďalej nulu a ako svoje definitívne číslo si priradí prvé číslo. Jedným smerom teda bude postupovať nula, ktorá je vždy menšia ako číslo daného počítača. Táto strana si takto ponechá pôvodné čísla. Ak bude druhé číslo väčšie ako prvé, počítač si nechá druhé číslo a ďalej pošle číslo o jedna väčšie ako druhé. Takto sa aj druhá strana postupne prečísluje.

1441. Rozdeľuj a panuj v čase $O(n \log n)$: Rozdelíme uhly na dve rovnako veľké množiny A a B, rekurzívne spočítame prienik uhlov v množine A (konvexný útvar K_A) a prienik uhlov v množine B (konvexný útvar K_B) a potom spočítame prienik K_A a K_B.

Otázkou zostáva, ako spočítať prienik dvoch konvexných útvarov. Rozdelíme rovinu vodorovnými priamkami, ktoré prechádzajú vrcholmi konvexných útvarov, na pásy. V každom páse vieme spočítať prienik v konštantnom čase. Prienik dvoch konvexných útvarov (ak počet vrcholov je m) teda dokážeme spočítať v čase $O(m)$.

Šikovné je na začiatku namiesto každého uhlu uvažovať dve polroviny (ktorých prienikom je daný uhol). Aby sme sa vyhli priamkam a polpriamkam, resp. počítaniu prieniku nekonečných útvarov, je dobré nájsť obdĺžnik, v ktorom sa nachádzajú prieniky všetkých priamok – treba nájsť najvyšší, najnižší, najpravejší a najľavejší priesečník, a to v $O(n \log n)$, aby sa nám nezhoršila časová zložitosť (skúste priamky utriediť podľa uhla, ktorý zvierajú s osou x).

1442. Dynamické programovanie. Pre každý podreťazec zadaného reťazca nájdeme najlepší spôsob, ako ho spakovať. (Presnejšie, budeme si pamätať jeho dĺžku po spakovaní a prvý krok, ktorý pri jeho pakovaní spravíme.) Podreťazce spracúvame postupne od najkratších po najdlhšie. Ak chceme spakovať reťazec $w = s_i s_{i+1} \dots s_j$, máme tri možnosti: Prvá možnosť je nechať ho nespakovaný. Druhá možnosť je rozdeliť ho na dve časti $s_i \dots s_{k-1}$ a $s_k \dots s_j$ a spakovať každú samostatne. (To už vieme spraviť, lebo tieto podreťazce sme už spracúvali.) Pri tretej možnosti nájdeme všetky jeho periódy, zakaždým skúsime spakovať jednu periódu a pridať príslušné opakovacie znamienko. Zo všetkých týchto možností si, samozrejme, vyberieme tú najlepšiu.

Ako nájsť periódy reťazca? Ak má reťazec periódu dĺžky a aj dĺžky b, tak má aj periódu dĺžky $\mathrm{nsd}(a, b)$. Preto nám stačí nájsť dĺžku najkratšej periódy, ostatné budú jej násobky (tie, ktoré delia dĺžku stringu). Upraveným algoritmom KMP vieme v lineárnom čase nájsť najmenšie také d, že w „pasuje“ na svoju o d znakov posunutú kópiu. Ak d delí dĺžku w, je to dĺžka najkratšej periódy, inak w nie je periodické.

Časová zložitosť tohto algoritmu je $O(N^3)$, pamäťová $O(N^2)$. Rozmyslite si, ako by sa úloha sťažila, ak by sme chceli nájsť spomedzi všetkých optimálnych spakovaní to, ktoré je prvé v „abecednom“ poradí.

1443. Použijeme dynamické programovanie: Označme $p_{h,s}$ počet kľúčikov šírky s daných vlastností, ktorých hĺbka posledného výbrusu je h; $p_{h,s}$ je teda počet postupností $\{a_i\}_{i=0}^{s}$ dĺžky $s+1$ s uvedenými vlastnosťami, pričom $a_0 = k$, ale $a_s = h$. Tieto hodnoty budeme počítať postupne: $p_{k,0} = 1$, $p_{i,0} = 0$ pre $i \neq k$. Pre ďalšie hodnoty platí $p_{i,j+1} = p_{i-1,j} + p_{i,j} + p_{i+1,j}$, pretože hĺbka predchádzajúceho výbrusu sa líši najviac o 1. (Krajné hodnoty sa dajú ošetriť jednoducho, ak dodefinujeme $p_{-1,j} = p_{k+1,j} = 0$ pre všetky j.)

Na základe tohto vzťahu môžeme hodnotu $p_{0,n}$, čo je počet kľúčikov šírky n, ktoré končia v hĺbke 0 (tie chceme), vypočítať v čase $O(nk)$ s použitím $O(k)$ pamäte.

Iné riešenie: Dá sa previesť na umocňovanie matice rozmeru $(k+1) \times (k+1)$ na n. Tak dostávame zložitosť $O(M(k) \log n)$, kde $M(k)$ je zložitosť násobenie matíc rozmeru $k \times k$. Triviálne $M(k) = O(k^3)$, existujú však algoritmy so zložitosťou $M(k) = O(k^{2.376\ldots})$. Výhodné pre „dlhé“ kľúčiky, ale v praxi asi až tak nie `:-)`.

Kombinatorické riešenie: Kľúčikov je toľko, čo ciest (postupností $\searrow$, $\nearrow$ a $\rightarrow$) z bodu $(0,0)$ do bodu (n,k). Zabudnime najskôr na obmedzenie $0 \leq a_i \leq k$. Šípiek $\nearrow$ musí byť o k viac ako $\searrow$. Nech teda l je počet $\searrow$ ($0 \leq l \leq (n-k) \operatorname{div} 2$) – počet takýchto ciest je $\binom{n}{l}\binom{n-l}{l+k}$, lebo vyberáme l pozícií pre $\searrow$ a $l+k$ pozícií pre $\nearrow$ (zvyšné pozície doplníme $\rightarrow$). Počet všetkých ciest je teda

$$C(n,k) = \sum_{l=0}^{(n-k)\operatorname{div}2} \binom{n}{l}\binom{n-l}{l+k} = \sum_{l=0}^{(n-k)\operatorname{div}2} \frac{n!}{l!\,(l+k)!\,(n-k-2l)!}.$$

Pozrime sa teraz na zlé cesty, ktoré sme započítali. Nech $(i,-1)$ je prvý bod, kde sa cesta dostala pod nulu. Ak zvyšok cesty od $(i,-1)$ do (n,k) preklopíme zvislo, dostaneme cestu, ktorá ide do $(n,-k-2)$. Naopak, ľubovoľná cesta, ktorá ide do $(n,-k-2)$, musí prechádzať cez -1 a keď zvyšok cesty preklopíme, dostaneme cestu do (n,k). Počet ciest, ktoré idú do (n,k) cez -1 je preto $C(n,-k-2) = C(n,k+2)$. Podobne počet ciest, ktoré idú cez $k+1$ je $C(n,k+2)$. Výsledok je teda $C(n,k) - 2C(n,k+2)$; dá sa vypočítať v čase $O(n)$.

Všetky odhady zložitosti samozrejme pre jednoduchosť predpokladajú, že jednu operáciu s ľubovoľne veľkým číslom vieme spraviť v konštantnom čase. Uvedomte si ale, že s rastúcim n môže počet kľúčov rásť až exponenciálne, a teda ak by sme chceli počítať počet kľúčov pre veľké n, mali by sme brať do úvahy aj veľkosť čísel, s ktorými počítame.

1444. Ide o počet oblastí v rovinnom grafe. Stačí použiť známu Eulerovu vetu: $s + v = h + k + 1$, kde v je počet vrcholov, h je počet hrán, k je počet komponentov súvislosti a s je počet oblastí.

1445. Algoritmus bude v podstate sekvenčný. Ide o klasickú úlohu hľadania artikulácie v grafe. Pozri napríklad úlohu 1124.

z1411. Najjednoduchšie je napísať si procedúru *Otoc*$(i, j$: **integer**$)$, ktorá otočí úsek poľa od i po j, t.j. vymení prvky a_i a a_j, a_{i+1} a a_{j-1}, atď. Potom celý problém vyrieši postupnosť príkazov *Otoc*(i,j); *Otoc*$(j+1,k-1)$; *Otoc*(k,l); *Otoc*(i,l); (pozri tiež 123).

z1412. Najrozumnejšie je asi napísať funkciu, zisťujúcu počet dní napríklad od 1.1.1901. Pozri aj úlohu 232.

z1413. Treba vlastne zistiť, či je graf, ktorého vrcholy tvoria zámky a hrany cesty, súvislý. To môžeme robiť prehľadávaním do šírky či do hĺbky, pozri [28, 38]. Iný prístup je, že komponenty súvislosti budeme tvoriť postupne počas čítania vstupu. Využijeme algoritmus union-find. V prípade, že získame jeden komponent súvislosti, je odpoveď kladná a netreba ani dočítať celý vstup.

z1414. Treba prehľadávať všetky možné začiatky a diferencie. Niekoľko fint: postupnosť dĺžky n má diferenciu deliteľnú všetkými prvočíslami menšími ako n. Členy postupnosti sú (veľké) prvočísla, a teda nebudú deliteľné malými prvočíslami (napríklad ≤ 20). Z toho nám

vyplynú nejaké obmedzenia na možné prvé členy postupnosti. Testovanie prvočíselnosti sa dá robiť pravdepodobnostne (napríklad Millerov-Rabinov test) [31.8 v 6; 5 v 13; 10.6 v 28] a nakoniec, keď nájdeme kandidáta na postupnosť, overíme si, či sú všetky členy naozaj prvočísla. A najdlhšia nájdená postupnosť?

```
Zaciatok: 503213  pocet: 14  diferencia: 4504500
```

z1415. Užitočné finty: Nič sa nestane, ak k dx (dy) pripočítame násobok $2n$ ($2m$). Môžeme ich teda upraviť tak, aby $-n < dx < n$ ($-m < dy < m$). Potom počas jedného skoku narazíme najviac do jednej severojužnej a jednej východozápadnej steny. Najjednoduchšie bolo skočiť bez ohľadu na steny a potom spätne zobraziť žabu zrkadlovo podľa tých stien, ktoré „preskočila“. Tieto operácie sa dajú robiť nezávisle pre x-ovú a y-ovú súradnicu.

z1421. Najprv upravme pole B: pre všetky i priraďme $B[i] := B[i] - c$. V poli B je teraz celkový zisk/strata za deň – tu treba nájsť súvislý úsek s najväčším súčtom (pozri úlohu 213).

z1422. Keby nebolo jazier a všetky nuly by predstavovali more, stačilo by pre každé pevninové políčko zistiť, či hranou susedí s morom – potom by bolo určite na pobreží. Teda treba odlíšiť more od jazier. Na mape je more súvislý úsek z núl, ktorý obsahuje políčko $(1, 1)$. Jednoduchým rekurzívnym ofarbovacím algoritmom možno more označiť napríklad číslom 2, a to tak, že označíme políčko $(1, 1)$ a rekurzívne sa zavoláme na všetkých jeho ôsmych susedov.

z1423. Ide o hľadanie mediánu, pozri riešenie úlohy 1333.

z1424. Úloha sa nazýva aj Collatzov problém. Sú v podstate dva možné prístupy: backtracking alebo systematické „drevorubačské“ skúšanie možností. Pri backtrackingu začneme od 1 a snažíme sa postupne skúšať inverzné operácie k tým popísaným v zadaniach. Postupne získavame dlhšie a dlhšie postupnosti. Lepšie sa ale, prekvapujúco, javí postupné skúšanie všetkých nepárnych čísel, párne potom ľahko dostaneme – ak $2k+1$ má postupnosť dĺžky d, tak $(2k+1)\ 2^i$ má postupnosť dĺžky $d+i$. Najlepší doterajší výsledok: $n = 63728127$, postupnosť má dĺžku 950 členov a $p(n)/c(n) = 118.75$.

z1425. Jednoduchá simulácia. Nemá zmysel v každom kroku prekresľovať celé vláčiky, vždy len zmažeme koniec a nakreslíme nový začiatok (rušeň). Tak isto nemusíme posúvať celé pole, len si zaznačíme, na ktorom políčku trasy je práve rušeň. Pri posúvaní rušňa vždy otestujeme, či nenastala zrážka.

z1431. Existuje iba osem transformácii matice: Identita, otočenie o 90°, otočenie o 180°, otočenie o 270°, súmernosť podľa osi y, súmernosť podľa osi y a otočenie o 90°, súmernosť podľa osi y a otočenie o 180°, súmernosť podľa osi y a otočenie o 270°. Každá operácia zložená zo základných operácii zo zadania je ekvivalentná s niektorou z vymenovaných. Preto stačí skontrolovať iba každú z týchto ôsmich.

z1432. V hrách tohto typu uvažujeme spravidla nasledovne: Majme pole, do ktorého indexujeme jednojednoznačne podľa momentálneho stavu hry. Konkrétne – ak sú dve kôpky guľôčok veľkosti m a n, bude pole napríklad dvojrozmerné a indexovať sa bude hodnotami m, n: $pole[m, n]$. V tomto poli budeme mať uloženú jednobitovú informáciu – či existuje z tohto stavu pre hráča, ktorý je na ťahu vyhrávajúca stratégia. Pole môžeme vybudovať takto: Označíme za prehrávajúce tie stavy, ktoré sú určené zadaním hry. Za vyhrávajúce potom označíme tie stavy, z ktorých sa dá pre súpera dosiahnuť prehrávajúci stav. Ďalej analogicky: prehrávajúci stav je taký, že akýmkoľvek ťahom dosiahneme pre súpera vyhrávajúci stav, atď. Takto hravo zistíme, či je začiatočný stav vyhrávajúci. Vo väčšine prípadov však toto pole netreba; to, či je stav vyhrávajúci, sa dá zistiť aj jednoduchými vzorcami (ktoré možno z tohto poľa odkukať).

z1433. Predstavme si orientovaný graf – vrcholmi sú kone, hrany sú od koňa A ku B vtedy, ak sa niekto stavil, že A skončí pred B. Úlohou je nájsť také usporiadanie vrcholov, že zo žiadneho vrcholu nevedie šípka do nejakého vrcholu, ktorý je v poradí pred ním. Inými slovami: treba topologicky utriediť vrcholy orientovaného grafu, alebo povedať, že sa to nedá (pozri úlohu 814 alebo [38]).

z1434. Zadanie je jednoduchá tzv. monoalfabetická unilaterálna substitučná šifra, t.j. písmeno za písmeno podľa tabuľky. Šifra v druhej časti úlohy je posun o 25, 13, 22 a 24 znakov s periódou zámeny pochopiteľne 4. Na nájdenie týchto konštánt stačí použiť vhodnú heuristiku so zameraním na často opakujúce sa slová.

z1435. Jednoduchá simulácia. Na zistenie zacyklenia mohamedánov použijeme metódu dvoch bežcov: Námestie simulujeme v dvoch poliach, jedno sa za sekundu zmení raz, druhé dvakrát. Testujeme zhodnosť týchto polí. Ak sú po určitom čase obe polia rovnaké a stále sa niektorí otáčajú, znamená to, že sme našli cyklus. Zapamätáme si konfiguráciu námestia, aby sme pokračovali simuláciou do tohto istého stavu, označujúc si mohamedánov, ktorí sa aspoň raz pohnú. Potom sa už ľahko spočítajú označení, t.j. mohamedáni, ktorí sa nikdy neprestanú otáčať.

1511. Najlepšia je implementácia používajúca na uloženie mien a čísel recidivistov písmenkový strom, prípadne hašovaciu tabuľku. Popis hašovania nájdete napr. v [11 v 6, 16 v 41, v 45].

1512. Pozri úlohy 412 a 634.

1513. Zostrojíme vhodný graf, ktorý zobrazuje vstupné údaje. Náš stav počas cestovania vieme jednoznačne určiť tým, kde práve sme a koľko je doma hodín. Každému možnému stavu bude teda zodpovedať jeden vrchol grafu. Každému letu zjavne vieme priradiť jednu orientovanú hranu v takomto grafe. Dĺžkou takejto hrany bude samozrejme dĺžka letu. V každom meste ešte pridáme hrany zodpovedajúce čakaniu na najbližší odlet.

Vieme miesto, kde začíname cestu, čas jej začiatku si môžeme zvoliť. Hľadáme teda najkratšiu cestu z nejakého vrcholu (odkiaľ, t_1) do nejakého vrcholu (kam, t_2). Toto sa dá dosiahnuť napr. pomocou Dijkstrovho algoritmu. Keďže nami zostrojený graf je (aspoň pre reálne vstupy) veľmi riedky, oplatí sa použiť implementáciu s haldou.

S využitím vyššie uvedených myšlienok vieme napísať riešenie, ktoré každú otázku zodpovie v čase $O(l \log l)$, kde l je počet liniek. (Ak máme l liniek, máme najviac $2l$ zaujímavých bodov v časopriestore. Z každého z nich vedie práve jedna „čakacia“ hrana, takže aj hrán je len lineárne veľa od l.)

Ak by mal byť počet otázok veľmi veľký, t.j. pýtali by sme sa na (takmer) všetky dvojice miest, oplatí sa vypočítať si všetky odpovede naraz použitím Floydovho-Warshallovho algoritmu.

1514. Táto úloha je ťažká. Dá neefektívne riešiť prehľadávaním do šírky (každý vrchol grafu predstavuje jeden možný stav celého radu, t.j. vrcholov je exponenciálne veľa od dĺžky vstupu), prípadne backtrackingom. Pri backtrackingu sa treba poriadne zamyslieť nad tým, ako ohraničiť jeho hĺbku.

V praxi vieme lepšie riešenie dosiahnuť použitím algoritmu A^*. V princípe ide o to, že prehľadávanie do šírky doplníme o funkciu f, ktorá pre každú pozíciu vráti nejaký dolný odhad počtu krokov, ktoré treba na to, aby sme z nej vyrobili nejakú cieľovú pozíciu. Pri prehľadávaní teraz budeme vyberať na spracovanie tú pozíciu p, pre ktorú je súčet počtu krokov, na ktorý sme sa do nej dostali, a hodnoty funkcie f minimálny.

Všimnite si, že pre f identicky rovnú nule dostávame klasické prehľadávanie do šírky. Ak nájdeme lepšiu funkciu f (ako by mohla vyzerať pre túto úlohu?), môže sa nám síce stať, že niektoré pozície budeme musieť spracovať viackrát (prečo?), ale v praxi budeme na nájdenie riešenia potrebovať prezrieť oveľa menej pozícií.

1515. V podúlohe a) použijeme pomocnú funkciu $f(n, a, b)$, ktorá vráti n-té Fibonacciho číslo nasledujúce za a a b.

Podúloha b): Položme si $x = 0$ a to po jednej zvyšujeme, kým $x^2 < n$. Zlepšiť môžeme napríklad binárnym vyhľadávaním, prípadne lepším výpočtom druhej mocniny.

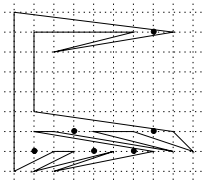
1521. Riešme najskôr situáciu, keď sú všetky riadky vyhľadávanej vzorky rôzne. Všimnime si, že keďže sú všetky riadky vyhľadávanej vzorky rovnako dlhé, na každom mieste veľkej matice začína nanajvýš jeden z vyhľadávaných riadkov. Očíslujeme si ich od 1 do N. Pre každý z nich si teraz jeho číslom označíme tie políčka veľkej matice, kde sa začína jeho výskyt. (Toto vieme spraviť klasickým algoritmom na vyhľadávanie v texte, napr. KMP, ešte šikovnejšie je použiť Ahov-Corasickov algoritmus a vyhľadávať všetky vzorky naraz.) Vzorka sa v matici nachádza, ak teraz v nejakom stĺpci nájdeme postupnosť čísel 1 až N.

Ak sú niektoré riadky rovnaké (čo pri Ahovom-Corasickovom algoritme zistíme, keď vyrábame písmenkový strom), jednoducho im pridelíme rovnaké číslo. Vzorke zo zadania by teda zodpovedala postupnosť 1, 2, 2. V stĺpcoch výslednej matice nebudeme hľadať postupnosť 1 až N, ale postupnosť, ktorá zodpovedá správnemu poradiu riadkov. Toto vieme opäť spraviť klasickým vyhľadávacím algoritmom KMP.

Na implementáciu jednoduchší prístup je zovšeobecnenie Rabinovho-Karpovho algoritmu pre viac rozmerov: Spočítame si hašovaciu hodnotu vzorky, ktorú hľadáme. Teraz postupne prechádzame po matici, v ktorej hľadáme, pre každú podmaticu správnych rozmerov spočítame jej hašovaciu hodnotu a porovnáme so vzorkou. Aby bolo toto riešenie efektívne, treba zvoliť vhodnú hašovaciu funkciu, aby sme vedeli rýchlo vypočítať novú hašovaciu hodnotu pri posunutí skúmanej podmatice o jeden stĺpec či riadok.

1522. Netradične, napíšeme celý vzorák. **begin** *writeln*(6); **end**. Skúste si spraviť dôkaz.

1523. Úlohu môžeme sformulovať v reči teórie grafov. Nech vrchol v grafe reprezentuje štvoricu čísel, a to políčko (x,y) a vektor rýchlosti (v_x,v_y), ktorou sme do políčka prišli. Ďalej nech vrchol i je spojený s vrcholom j práve vtedy, keď sa zo stavu zodpovedajúceho vrcholu i vieme jedným krokom dostať do stavu zodpovedajúceho vrcholu j. Úlohou je teraz nájdenie najkratšej cesty v grafe z vrcholu $(x_{start}, y_{start}, 0, 0)$ do ľubovoľného vrcholu s polohou cieľového políčka. Toto sa dá riešiť prehľadávaním do šírky.



1524. Ak by boli všetky body v jednom riadku, môžeme tie nepotrebné jednoducho „vyhryznúť“. Napríklad pre body (1,1), (4,1), (6,1) ich orámujeme obdĺžnikom (0,0), (7,0), (7,2), (0,2) a postupne „vyhryzávame“ nežiaduce body, čím vznikne tento zoznam: (0,0), (2,1), (3,1), (1,0), (5,1), (2,0), (7,1), (0,2). Keby sme tento algoritmus potrebovali rozšíriť pre viac riadkov, budeme z vrcholu (7,1) pokračovať vo „vyhryzávaní“ druhého riadku.

1525. 1. Napíšeme si pomocnú funkciu R s pomocným parametrom a, v ktorom sa vytvára obrátený zoznam: $R((u, v), a) = R(v, (u, a))$ (zoberieme zo začiatku a dáme na začiatok) a $R(0, a) = a$. Potom funkciu *Rev* dostaneme ľahko ako $Rev(x) = R(x, 0)$. 2. Využijeme *Rev* z predchádzajúcej úlohy: $Append(x, y) = R(Rev(x), y)$. 3. Môžeme použiť ľubovoľný triediaci algoritmus; ľahko implementovateľný je napríklad *MergeSort*. Potrebujeme urobiť funkcie *Msplit*, ktorá rozdelí zoznam na dve časti (pre jednoduchosť párne a nepárne pozície) a *Merge*, ktorá dva utriedené zoznamy spojí.

$$\begin{aligned} MergeSort(0) &= 0, \\ MergeSort(u, 0) &= u, 0, \\ MergeSort(u, v) &= Merge(MergeSort(a), MergeSort(b), 0) \leftarrow a, b = Msplit(0, 0, u, v) \end{aligned}$$

1531. Dá sa aj lepšie ako lineárne. Najefektívnejšie je použiť tri haldy. V jednej budú uložené ženy pred rozkvetom, v druhej po rozkvete (podľa krásy) a v tej tretej budú opäť

ženy pred rozkvetom, tentokrát usporiadané podľa toho, koľko im ostáva do rozkvetu. Takto vieme na Silvestra pomocou tretej haldy ľahko nájsť ženy, ktoré treba presunúť z prvej haldy do druhej. Na to, aj na operáciu `EXITUS` potrebujeme vedieť, kde v haldách sa ktorá žena nachádza. Preto si ich ešte raz uložíme do nejakej efektívnej štruktúry, napríklad písmenkového stromu alebo hašovacej tabuľky.

1532. Označme f_n hľadaný počet spôsobov, ako sa dajú udržiavať cesty a g_n počet tých spôsobov, kde je Veľký Eskimák spojený priamo cestou s posledným iglu. Indukciou dokážeme vzťahy $g_n = f_n - f_{n-1}$ a $f_n = 2f_{n-1} + g_{n-1} = 3f_{n-1} - f_{n-2}$ – n-té iglu môžeme totiž ku zvyšku pripojiť troma spôsobmi: k iglu „nad“ ním, ku veľkému iglu (pre oba prípady je f_{n-1} spôsobov), alebo k obom (potom Veľké a $(n-1)$-vé iglu nie sú spojené; presvedčte sa, že je g_{n-1} takých spôsobov). Toto už stačí na jednoduché lineárne riešenie – pamätáme si hodnoty dvoch predchádzajúcich výsledkov (f_{n-1} a f_{n-2}) a počítame ďalšie.

Indukciou sa dá dokázať, že $f_n = Fib_{2n-2}$, no a k-te Fibonacciho číslo sa dá vypočítať v logaritmickom čase pomocou známeho vzorca (pozri úlohu 113).

1533. Označme s_i celkovú úrodu v prvých i osadách. Otázky si prepíšeme do tvaru $s_{k-1} - s_l \leq -m$ alebo $s_l - s_{k-1} \leq m$ podľa toho, či hovoria **aspoň** alebo **najviac**. Keďže úroda všade musí byť nezáporná, máme navyše $s_k - s_l \leq 0$ pre všetky $k \leq l$. Vytvorme si graf, kde každú takúto nerovnicu premeníme na orientovanú hranu: $s_a - s_b \leq c$ prepíšeme na hranu z a do b ohodnotenú číslom c.

Rozmyslite si, že v našom grafe existuje cesta z x do y (pre $x > y$) s súčtom ohodnotení hrán d, znamená to, že z nerovníc zodpovedajúcich jednotlivým jej hranám vieme odvodiť tvrdenie: „v úseku od osady $y+1$ po osadu x vrátane sa urodilo najviac d plodov“. Podobne, cesty pre $x < y$ zodpovedajú tvrdeniam so slovom „najviac“. Zjavne ak náš graf obsahuje cyklus zápornej dĺžky, sústava nerovníc nemá riešenie, a teda sa nemôžu všetky veštby splniť. V opačnom prípade dĺžky najkratších ciest v našom grafe vlastne zodpovedajú „najprísnejším“ odvoditeľným podmienkam a dá sa z nich priamo zostrojiť jedno prípustné rozdelenie úrody.

Obe veci naraz (overenie existencie záporného cyklu aj nájdenie dĺžok najkratších ciest medzi vrcholmi) vieme spraviť použitím Floydovho-Warshallovho algoritmu (pozri riešenie úlohy 823).

1534. Priamočiare riešenie je preskúšanie všetkých možností, existujú však aj trochu lepšie riešenia.

Prvá možnosť: Vyskúšame postupne všetky natočenia darčekov (3^n možností) a pre každé natočenie polynomiálne zistíme, ako darčeky preusporiadať – najdlhšia nerastúca podpostupnosť, pozri 522. Takto získavame riešenie v čase $O(3^n n^2)$ alebo $O(3^n n \log n)$.

Druhá, lepšia možnosť: Keď staviame vežu, aktuálny stav je jednoznačne určený tým, ktorá stena ktorej krabice je na vrchu, a ktoré krabice sú ešte nepoužité. Takto sme priestor stavov zredukovali na $O(2^n n)$. Stavanie veže si môžeme predstaviť ako hľadanie najdlhšej cesty v (acyklickom) grafe, ktorého vrcholy sú stavy a hrany zodpovedajú pridaniu krabice. Túto úlohu vieme vyriešiť dynamickým programovaním (spracúvame vrcholy v topologickom usporiadaní), prípadne rekurzívnou funkciou s memoizáciou (t. j. pamätáme si, s akými parametrami sme sa už volali a čo sme vtedy vypočítali, aby sme neriešili tú istú úlohu viackrát).

Časová zložitosť tohto postupu je $O(2^n n^2)$, pamäťová $O(2^n n)$.

1535. Funkcia *Member* sa dala triviálne implementovať v logaritmickom čase, horšie to bolo s *Insert* a *Delete*. Keďže sme si pri stromoch nemohli pamätať žiadnu informáciu navyše, nedala sa dosiahnuť lepšia ako lineárna zložitosť. Najlepšie je strom skonvertovať na zoznam, na tom previesť samotnú operáciu, a potom zoznam skonvertovať naspäť na strom.

1541. Rovinu si narežeme na vodorovné pásy priamkami, ktoré prechádzajú cez všetky priesečníky kružníc. Takto nám vznikne nanajvýš n^2 pásov. Každý pás obsahuje nanajvýš $2n$ oblúkov, ktoré sa nepretínajú. Keď si pásy utriedime a takisto aj oblúky v pásoch, vieme použiť binárne vyhľadávanie, ktoré nájde správne políčko v čase $O(\log n)$. Predspracovanie urobíme použitím zametania, čo sa dá v časovej a pamäťovej zložitosti $O(n^3)$. Použitím prefíkanejších dátových štruktúr vieme toto riešenie zlepšiť na čas $O(n^2 \log n)$ a pamäť $O(n^2)$.

1542. Na začiatku určíme, ktoré políčko papiera bude v ľavom hornom rohu hornej steny kocky ($6n^2$ možností) a ako bude papier otočený a preklopený (8 možností). Zakaždým prehľadávaním do hĺbky postupne darček obaľujeme a zistíme, či sa dá obaliť alebo nie. Toto prehľadávanie má zložitosť $O(n^2)$, čiže celý program má zložitosť $O(n^4)$.

1543. Úloha je vlastne hľadanie maximálneho toku zo začiatočných políčok von z mesta. Použijeme Fordov-Fulkersonov algoritmus. Pozri 1424.

1544. Začnime tým, že vyriešime zjavné situácie: Ak niekde nie je prípustné prelepiť otlak jedným smerom, prelepíme ho druhým smerom. (Najjednoduchším prípadom sú dva otlaky susediace stranou.) Nech teraz už každý z otlakov, ktoré zostali zatiaľ nezalepené, možno zalepiť oboma smermi.

Pre každý otlak A nech a označuje výrok „A zalepíme zvislo“ a $\overline{a}$ výrok „A zalepíme vodorovne“. Ak otlaky A a B susedia rohom, tak obe náplasti musíme nalepiť rovnakým smerom – platí $a \iff b$. Ak A a B ležia v tom istom riadku a je medzi nimi práve jedno políčko, tak platí: ak jeden z nich nalepíme vodorovne, druhý musíme nalepiť zvislo. Teda neplatí $a \wedge b$. Toto môžeme zapísať ako $\overline{a} \Longrightarrow b$, resp. ekvivalentne ako $\overline{b} \Longrightarrow a$. Analogicky ak A a B ležia v tom istom stĺpci.

Situáciu na Rasťovej päte si teda môžeme znázorniť ako graf. Vrcholy budú predstavovať jednotlivé výroky (teda každému otlaku X priradíme dva vrcholy: x a $\overline{x}$; vrcholy x a $\overline{x}$ budeme volať *opačné*), hrany budú predstavovať jednotlivé implikácie. (Pre vodorovný aj zvislý prípad zaznačíme obe implikácie, aj napriek tomu, že sú ekvivalentné.)

Našou úlohou je ohodnotiť vrcholy grafu hodnotami 1 (pravda) a 0 (nepravda) tak, aby pre každé x práve jeden z vrcholov x a $\overline{x}$ bol ohodnotený 1, a navyše žiadna hrana neviedla z vrcholu ohodnoteného 1 do vrcholu ohodnoteného 0. (Spomeňte si, že každá hrana predstavuje jednu implikáciu, a všetky tieto implikácie musia byť splnené.)

Tvrdíme, že Rasťove otlaky sa dajú všetky prelepiť náplasťami práve vtedy, keď žiadne dva opačné vrcholy neležia v tom istom silne súvislom komponente (SSK). Ak totiž nejaké x a $\overline{x}$ ležia v rovnakom SSK, existuje cesta z x do $\overline{x}$, aj cesta z $\overline{x}$ do x – teda platí $x \iff \overline{x}$, čo sa zjavne nedá splniť.

Naopak, predpokladajme, že žiadne dva opačné vrcholy neležia v rovnakom SSK. Zjavne všetky vrcholy v jednom SSK musia dostať to isté ohodnotenie. Vytvoríme teda nový graf, v ktorom každý SSK nahradíme jedným „veľkým“ vrcholom. Dostaneme acyklický graf, ktorý topologicky usporiadame tak, aby všetky hrany z každého komponentu smerovali len do komponentov, ktoré sú pred ním.

Pre každé x si pozrime, ktorý z komponentov obsahujúcich x a $\overline{x}$ je v našom poradí skôr. Ten ohodnotíme 1, a druhý komponent ohodnotíme 0. Rozmyslite si, že týmto postupom zostrojíme korektné ohodnotenie vrcholov.

Implementácia v čase $O(n^2)$ je ľahká. Pre riešenie v čase $O(n)$ si pozrite [3.E v 38].

1545. Pozri riešenie úlohy 1315. Oba spomínané postupy sa dajú naprogramovať aj vo funkcionálnom programovacom jazyku.

z1511. Každú súvislú oblasť ofarbíme a otestujeme, či to je znak Z. Najlepšie je spraviť priemet na os y a spočítať, koľko znakov je v tom-ktorom riadku. Očakávame v každom riadku jednu jednotku a v dvoch riadkoch aspoň tri. Na záver ešte otestovať, či je to šikmá čiara. Program beží v čase úmernom počtu políčok.

z1512. Strom kreslíme rekurzívne. Nakreslíme rovnú čiaru, otočíme sa, menší podstrom, otočíme sa, menší podstrom a vrátime sa na začiatok (to je dôležité, pretože už pri kreslení menších podstromov sme predpokladali, že po nakreslení sa Zeofína vrátila na pôvodné miesto).

z1513. Existujú dva prístupy: Prehľadávanie do šírky, alebo do hĺbky. Oba postupy sú rovnocenné a majú časovú zložitosť rovnú počtu políčok. Pre optimalizáciu bolo potrebné neprekresľovať všetky okná v každom kroku.

z1514. Kratšiu stranu ihriska označíme ako $\vec{k}$ a dlhšiu ako $\vec{d}$. Každý bod roviny vieme teraz zapísať ako $A+x\vec{k}+y\vec{d}$, kde A je vrchol ihriska. Čísla x a y vypočítame tak, že si túto rovnicu rozložíme na x-ovú a y-ovú zložku a vyriešime sústavu dvoch lineárnych rovníc. Teraz podľa čísel x a y jednoducho určíme, kto je hráč a aký. Hráči majú x aj y medzi 0 a 1. Hráči na jednej polovici majú $x \leq 0.5$ a na druhej $x \geq 0.5$.

z1515. Existujú tri prípady podľa veľkosti mincí p a q: Ak máme jednokorunovú mincu, zjavne vieme zaplatiť všetky hodnoty. V opačnom prípade nech $d = \mathrm{nsd}(p, q)$. Ak $d > 1$, nedá sa zaplatiť nekonečne veľa hodnôt – všetky hodnoty, ktoré sa dajú zaplatiť, sú násobkom d.

Posledný prípad je, keď sú p a q nesúdeliteľné. Dá sa dokázať, že pre ľubovoľné dve celé čísla u, v sa $\mathrm{nsd}(u, v)$ dá zapísať v tvare $mu+nv$ pre vhodné celé čísla m a n. V našom prípade teda existujú m, n také, že $1 = mp + nq$. (Jedna takáto dvojica čísel m, n sa dá zistiť rozšíreným Euklidovým algoritmom.) Potom ľubovoľnú hodnotu t vieme zapísať ako $t = (tm)p + (tn)q$. Keby teda bolo povolené vydávanie, vedeli by sme zaplatiť každú sumu (jedno z čísel m, n je záporné).

Majme teraz nejakú sumu $t = mp + nq$ pre nám známe hodnoty m a n. Potom všetky dvojice m', n', pre ktoré platí $t = m'p + n'q$, majú tvar $m' = m + kq$ a $n' = n - kp$ pre ľubovoľné celočíselné k.

(Dosadením si overte, že ak $t = mp+nq$, tak aj $m'p+n'q = t$. Naopak, odčítaním rovníc dostaneme $(m - m')p = (n' - n)q$, a keďže p a q sú nesúdeliteľné, q musí deliť $(m - m')$.)

Pre ľubovoľné t teda vieme nájsť takú dvojicu m', n', že $t = m'p + n'q$ a $0 \leq m' < q$. Takáto dvojica je vždy práve jedna. Ak zakážeme vydávanie, vieme vyjadriť iba tie hodnoty, pre ktoré $n' \geq 0$, resp. nevieme vyjadriť práve tie sumy, kde $0 \leq m' < q$ a $n' < 0$. Najväčšie číslo, ktoré nevieme vyjadriť, dostaneme, ak zvolíme $m' = q - 1$ a $n' = -1$.

Výsledný algoritmus: Ak $p = 1$ alebo $q = 1$, vieme zaplatiť všetky hodnoty. Ak $\mathrm{nsd}(p, q) > 1$, nedá sa zaplatiť nekonečne veľa hodnôt. Inak je výsledok $pq - p - q$. Najväčšieho spoločného deliteľa zistíme Euklidovým algoritmom: $\mathrm{nsd}(a, b) = \mathrm{nsd}(b, a \bmod b)$.

Len pre zaujímavosť, ide o tzv. Frobéniov problém mincí. V roku 1884 ho pre dve mince vyriešil Sylvester. Dokázal tiež, že práve polovica hodnôt od 1 po $(p - 1)(q - 1)$ sa nedá vyjadriť. Pre viacero mincí je problém *podstatne* ťažší.

z1521. V prvom rade si treba uvedomiť, že stačí robiť iba slimákov otočených vpravo (vľavo otočení slimáci sú s nimi osovo súmerní). Taktiež oddelíme 1-mesačného slimáka ako osobitný prípad. Ostatní slimáci majú rozmery $4n - 1 \times 4n - 3$ a môžeme si ich po stĺpcoch rozdeliť na tri časti: stĺpce 1 až $2(n - 1) + 1$, stĺpce $2n$ až $4(n - 1)$ a zvyšok. Pre jednotlivé časti sa dajú, zvlášť pre párne, zvlášť pre nepárne stĺpce odvodiť vzorčeky pre výskyt $*$. Potom stačí v dvoch cykloch prebehnúť celého slimáka a tlačiť obrázok po znakoch.

z1522. Bludisko stupňa stupňa 0 prejdeme triviálne ako chodbičku dĺžky d – príkaz *Dopredu*(d). Ak postupnosť príkazov pre bludisko stupňa n nazveme $B(n)$, príkaz *Vlavo* nazveme L a *Vpravo* R, tak postupnosť pre bludisko stupňa $n + 1$ sa dá napísať takto: $B(n)$; R; $B(n)$; L; $B(n)$; L; $B(n)$; $B(n)$; L; $B(n)$; L; $B(n)$; L; $B(n)$; $B(n)$.

z1523. Ak sa na slonie kosti dívame ako na ohodnotený graf, potom úlohou je nájsť v ňom najlacnejšiu množinu hrán, ktorá pospája všetky vrcholy. Takúto množinu hrán voláme aj v grafovej terminológii *najlacnejšia kostra*. Pozri riešenie úlohy 813.

z1524. Triviálne riešenie je vyplniť si v poli rozmerov $A \times B$ pre každé políčko počet kusov oblohy, ktoré na ňom ležia; to vieme v čase $O(kAB)$.

Počet oblôh na políčku s ľavým dolným rohom $[x, y]$ vieme spočítať nasledovne: Všimnime si konkrétny kus oblohy. Kedy v ňom naše políčko leží? Ak sa spomedzi štyroch rohov oblohy len ľavý dolný roh nachádza v obdĺžniku s rohmi $[0, 0]$ a $[x, y]$.

Predstavme si, že teraz prejdeme všetky rohy oblôh, ktoré sa nachádzajú v obdĺžniku s rohmi $[0, 0]$ a $[x, y]$. Za každý ľavý dolný a pravý horný roh zarátame $+1$ a za každý ľavý horný aj pravý dolný roh zarátame -1. Zjavne každý kus oblohy, v ktorom sa naše políčko nachádza, prispeje do výsledného súčtu $+1$. Rozmysli si, že kus oblohy, v ktorom sa naše políčko nenachádza, do výsledného súčtu prispeje nulou. (V skúmanom obdĺžniku buď neleží ani jeden vrchol, alebo dva susedné, alebo všetky štyri, vždy dostaneme súčet nula.) Súčet, ktorý dostaneme, bude teda rovný počtu kusov oblohy na danom políčku.

Majme dvojrozmerné pole $p[A+1, B+1]$, ktoré je na začiatku plné núl. Pre každý kus oblohy upravíme o 1 štyri políčka p zodpovedajúce jeho rohom. Teraz postupne pre každé $[x, y]$ spočítame hodnotu $s[x, y]$ (súčet $p[i, j]$ pre $0 \leq i \leq x$ a $0 \leq j \leq y$). Toto celé vieme spraviť v čase $O(AB + k)$.

z1525. Program je veľmi jednoduchý, využíva rekurzívne volania. Interval (l, p) rozdelíme na polovicu a rovnakým algoritmom zaštipcujeme jeho ľavú časť $(l, (l+p)/2)$ a potom pravú $((l+p)/2, p)$.

z1531. Zostrojíme si tabuľku, pričom v každom políčku si pamätáme, koľko úloh môže marsochod vykonať, keď začína tu. Takúto tabuľku vyplníme zdola: Pre políčko, kde je stena tam položíme -1, pre prázdne políčko tam dáme maximum z políčok vpravo a dole a pre políčko s úlohou tam dáme $1 +$ maximum z políčok vpravo a dole. V začiatočnom políčku budeme mať nakoniec maximálny počet úloh, ktoré môže marsochod splniť. Pri rekonštrukcii cesty sa vyberieme vždy na to políčko, ktoré má väčšiu hodnotu.

z1532. Kreslíme rekurziou podľa zadania. Treba dávať pozor na to, že veža veľkosti 1 vyzerá trochu inak ako ostatné.

z1533. Znak sa dá nakresliť jedným ťahom, ak je súvislý (ďalej budeme predpokladať, že *je* súvislý) a ak vrcholov s nepárnym stupňom je buď 0 alebo 2. V riešení úlohy 643 sme ukázali, ako znak nakresliť jedným uzavretým ťahom, ak sú všetky vrcholy párneho stupňa.

Ak znak obsahuje dva vrcholy s nepárnym stupňom, vieme ho stále nakresliť jedným, avšak nie uzavretým ťahom. Použijeme nasledovný trik: Dva nepárne vrcholy spojíme *falošnou* hranou. Teraz majú všetky vrcholy párny stupeň a znak vieme podľa riešenia 643 nakresliť jedným uzavretým ťahom. Keď z tohto ťahu odstránime falošnú hranu, ostane nám neuzavretý ťah.

Tento poznatok sa dá zovšeobecniť: Ak znak obsahuje $2k > 0$ vrcholov s nepárnym stupňom (počet vrcholov s nepárnym stupňom je vždy párny), potom ho vieme nakresliť k ťahmi. Stačí pridať k falošných hrán a podľa riešenia 643 nájsť eulerovský ťah. Po odstránení falošných hrán sa tento ťah rozpadne na k neuzavretých ťahov.

z1534. Budík sa mohol zaseknúť iba tak, že dve dotýkajúce sa kolieska sa otáčajú opačne. Smer otáčania môžeme zistiť buď prehľadávaním do šírky alebo do hĺbky. Oba prístupy sú rovnako efektívne.

z1535. Diery si utriedime podľa vzdialenosti od začiatku. Potom ideme zľava doprava a položíme nový koberček na prvé nezakryté miesto. Takéto riešenie je optimálne. Triedenie je $O(n \log n)$ a samotný algoritmus je $O(n)$.

Literatúra

[1] Aho, V. A., Hopcroft, E. J., Ullman, D. J., *The Design and Analysis of Computer Algorithms*, Reading (Massachusetts) : Addison-Wesley, 1976.

[2] Barker, F., Ross-Macdonald, M., Castlereagh, D., *The Search Begins*, London : Aldus Books and Jupiter Books, 1973. Slovenský preklad: *Začiatky hľadania*, Bratislava : Mladé letá, 1981.

[3] Bentley, J., *Programming Pearls*, Reading (Massachusetts) : Addison-Wesley, 1986. Slovenský preklad: *Perly programovania*, Bratislava : Alfa, 1992.

[4] de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O., *Computational Geometry, Algorithms and Applications*, Berlin : Springer-Verlag, 1997.

[5] Carroll, L., *Alica v krajine zázrakov*, Bratislava : Jaspis, 1996.

[6] Cormen, H. T., Leiserson, E. C., Rivest, L. R., Stein, C., *Introduction to Algorithms*, druhé vydanie, Cambridge (Massachusetts) : The MIT Press, 2001.

[7] Demel, J., *Grafy*, Praha, 1989.

[8] ———, *Grafy a jejich aplikace*, Praha : Academia, 2002.

[9] ———, *Teorie grafů*, Praha, 1984.

[10] Dijkstra, W. E., *A Discipline of Programming*, Prentice-Hall, Inc., 1976.

[11] Dijkstra, W. E., Feijen, W. H. J., *A Method of Programming*, Reading (Massachusetts) : Addison-Wesley, 1988.

[12] Gonnet, H. G., Baeza-Yates, R., *Handbook of Algorithms and Data Structures, In Pascal and C*, Reading (Massachusetts) : Addison-Wesley, 1991.

[13] Giblin, P., *Primes and Programming, An Introduction to Number Theory with Computing*, Cambridge : Cambridge University Press, 1993.

[14] Graham, R. L., Knuth, D. E., Patashnik, O., *Concrete Mathematics: A Foundation for Computer Science*, Reading (Massachusetts) : Addison-Wesley, 1994.

[15] Granát, L., Sechovský, H., *Počítačová grafika*, Praha : KVT SNTL, 1980.

[16] ———, *Počítačová grafika – zobrazování úsečky*, *Rozhledy matematicko-fyzikální* 190–196, č. 5, roč. 66, 87/88.

[17] ———, *Počítačová grafika – ořezávání*, *Rozhledy matematicko-fyzikální* 270–274, č. 7, roč. 66, 87/88.

[18] Gries, D., *The Science of Programming*, Springer-Verlag, 1981.

[19] Gruska, J., *Foundation of Computing*, International Thomson Computer Press, 1997.

[20] Hopcroft, E. J., Ullman, D. J., *Formálne jazyky a automaty*, Bratislava : Alfa, 1978.

[21] Hvorecký, J., *Hra s algoritmom*, *Matematické Obzory 31*, 1–11, 1989.

[22] Kemp, R., *Fundamentals of the Average Case Analysis of Particular Algorithms*, (B. G. Teubner) John Wiley & Sons

[23] Knuth, D. E., *The Art of Computer Programming, Volume 1: Fundamental Algorithms*, Reading (Massachusetts) : Addison-Wesley, 1968.

[24] ———, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Reading (Massachusetts) : Addison-Wesley, 1981.

[25] ———, *The Art of Computer Programming, Volume 3: Sorting and Searching*, Reading (Massachusetts) : Addison-Wesley, 1973.

[26] ———, *The Stanford GraphBase : A Platform for Combinatorial Computing*, Reading (Massachusetts) : Addison-Wesley, 1993.

[27] Kozen, D. C., *The Design and Analysis of Algorithms*, New York : Springer-Verlag, 1992.

[28] Kučera, L., *Kombinatorické algoritmy*, MS 18, Praha : SNTL, 1983.
[29] Larson, C. L., *Metódy riešenia matematických problémov*, Bratislava : Alfa, 1990.
[30] Lovász, L., *Combinatorial Problems and Exercises*, Budapest : Akadémiai Kiadó, 1979.
[31] Lucas, É., *Récréations mathématiques*, Paris : Gauthier-Villars, 1891–1894.
[32] Machačka, I., Paulů, J., *Programování v jazyku BASIC*, Praha : SNTL, 1986.
[33] Maličký, P., *Zátvorkový problém*, *Matematické Obzory 33*, 61–67, 1989.
[34] Manber, U., *Introduction to Algorithms : A Creative Approach*, Reading (Massachusetts) : Addison-Wesley, 1989.
[35] Melhorn, K., *Data Structures and Algorithms 3: Multi-dimensional Searching and Computational Geometry*, Springer-Verlag, 1984.
[36] Molnár, Ľ., *Programovanie v jazyku Pascal*, Bratislava : Alfa, 1987.
[37] Motwani, R., *Approximation Algorithms*, Preliminary version, Stanford University Technical Report.
[38] Plesník, J., *Grafové Algoritmy*, Bratislava : VEDA, 1983.
[39] Preparata, F. P., Shamos, M. I., *Computational Geometry : An Introduction*, Text & Monographs in CS, New York : Springer-Verlag, 1985.
[40] Riečan, B., *Príbehy o integráloch*, SPN, 1988
[41] Sedgewick, R., *Algorithms*, druhé vydanie, Reading (Massachusetts) : Addison-Wesley, 1988.
[42] Sedgewick, R., Flajolet, P., *An Introduction to the Analysis of Algorithms*, Reading (Massachusetts) : Addison-Wesley, 1996.
[43] Suri, S., *A Linear Time Algorithm for Minimum Link Paths Inside a Simple Polygon*, Computer Vision, Graphics, and Image processing 35, 99–110, 1986.
[44] Vít, P., *Řetězové zlomky*, Škola mladých matematiků, sv. 49, Praha, 1982.
[45] Wiederman, J., *Vyhledávání*, MS 27, Praha : SNTL, 1991.
[46] Wirth, N., *Algorithms + Data Structures = Programs*, Prentice-Hall, Inc., 1975. Slovenský preklad: *Algoritmy a štruktúry údajov*, Bratislava : Alfa, 1987.
[47] ———, *Algorithms and Data Structures*, Prentice-Hall, Inc., 1986.

Index

Michal Forišek, Jakub Kováč
Zbierka riešených úloh Korešpondenčného seminára z programovania (1983–1998)

Pôvodný návrh obálky Dušan Bezák, Miroslav Dudík, Dano Štefankovič a Tomáš Vinař
Sádzané programom TEX

Vytlačila a vydala OKAT PLUS, spol. s r.o. Bratislava
12 + 183 strán, náklad 500 výtlačkov
ISBN 80-88720-09-5
EAN 9788088720096
Neprešlo jazykovou úpravou


9 788088 720096