Informácie a pravidlá

Pre koho je súťaž určená?

Do **kategórie B** sa smú zapojiť len tí žiaci základných a stredných škôl, ktorí ešte ani v tomto, ani v nasledujúcom školskom roku nebudú končiť strednú školu.
Do **kategórie A** sa môžu zapojiť všetci žiaci (základných aj) stredných škôl.

Odovzdávanie riešení domáceho kola

Riešitelia domáceho kola odovzdávajú riešenia sami, v elektronickej podobe, a to priamo na stránke olympiády: `http://oi.sk/`. Odovzdávanie riešení bude spustené niekedy v septembri 2013.

Riešenia kategórie A je potrebné odovzdať najneskôr **15. novembra 2013**.
Riešenia kategórie B je potrebné odovzdať najneskôr **30. novembra 2013**.

Priebeh súťaže

Za každú úlohu domáceho kola sa dá získať od 0 do 10 bodov. Na základe bodov domáceho kola stanoví Slovenská komisia OI (SK OI) pre každú kategóriu bodovú hranicu potrebnú na postup do **krajského kola**. Očakávame, že táto hranica bude približne rovná **tretine maximálneho počtu bodov**.

V krajskom kole riešitelia riešia štyri teoretické úlohy, ktoré môžu tematicky nadväzovať na úlohy domáceho kola. V kategórii B súťaž týmto kolom končí.

V kategórii A je približne najlepších 30 riešiteľov krajského kola (podľa počtu bodov, bez ohľadu na kraj, v ktorom súťažili) pozvaných do **celoštátneho kola**. V celoštátnom kole účastníci prvý deň riešia teoretické a druhý deň praktické úlohy. Najlepší riešitelia sú vyhlásení za víťazov. Približne desať najlepších riešiteľov následne SK OI pozve na týždňové výberové sústredenie. Podľa jeho výsledkov SK OI vyberie družstvá pre Medzinárodnú olympiádu v informatike (IOI) a Stredoeurópsku olympiádu v informatike (CEOI).

Ako majú vyzerať riešenia úloh?

V praktických úlohách je vašou úlohou vytvoriť program, ktorý bude riešiť zadanú úlohu. Program musí byť v prvom rade korektný a funkčný, v druhom rade sa snažte aby bol čo najefektívnejší.

V kategórii B môžete použiť ľubovoľný programovací jazyk.

V kategórii A musíte riešenia praktických úloh písať v jazyku Pascal, C, alebo C++, Odovzdaný program bude automaticky otestovaný na viacerých vopred pripravených testovacích vstupoch. Podľa toho, na koľko z nich dá správnu odpoveď, vám budú pridelené body. Výsledok testovania sa dozviete krátko po odovzdaní. Ak váš program nezíska plný počet bodov, budete ho môcť vylepšiť a odovzdať znova, až do uplynutia termínu na odovzdávanie.

Presný popis, ako majú vyzerať riešenia praktických úloh (napr. realizáciu vstupu a výstupu), nájdete na webstránke, kde ich budete odovzdávať.

Ak nie je v zadaní povedané ináč, riešenia teoretických úloh musia v prvom rade obsahovať **podrobný slovný popis použitého algoritmu, zdôvodnenie jeho správnosti** a diskusiu o efektivite zvoleného riešenia (t. j. posúdenie časových a pamäťových nárokov programu). Na záver riešenia uveďte program. Ak používate v programe netriviálne algoritmy alebo dátové štruktúry (napr. rôzne súčasti STL v C++), súčasťou popisu algoritmu musí byť dostatočný popis ich implementácie.

Usporiadateľ súťaže

Olympiádu v informatike (OI) vyhlasuje *Ministerstvo školstva SR* v spolupráci so *Slovenskou informatickou spoločnosťou* (odborným garantom súťaže) a *Slovenskou komisiou Olympiády v informatike*. Súťaž organizuje *Slovenská komisia OI* a v jednotlivých krajoch ju riadia *krajské komisie OI*. Na jednotlivých školách ju zaisťujú učitelia informatiky. Celoštátne kolo OI, tlač materiálov a ich distribúciu po organizačnej stránke zabezpečuje IUVENTA v tesnej súčinnosti so Slovenskou komisiou OI.

A-I-1 Taká zima...

Toto je praktická úloha. Pomocou webového rozhrania odovzdajte ***funkčný, odladený program****.*

„Teda, taká zima ako dnes bola naposledy na Štedrý deň...“ povzdychol si nad pivom dedo Bonifác. „To nič nie je, včera bola taká zima, že podobná bola naposledy druhého decembra!“ chcel ho tromfnúť dedo Ignác. No na Bonifáca nemal. „Pche, to zabúdaš na zimu čo bola piateho januára, tá bola od tej včera menšia, ale od druhého decembra väčšia!“

Ignác celú noc nemohol zaspať, spomínal na teploty a snažil sa prísť na výrok, ktorý už Bonifác nedokáže vyvrátiť ani prekonať. Chcel by povedať výrok nasledovného typu: „pred x dňami bolo tak teplo, že podobne bolo naposledy predtým pred y dňami“.

Nech $T[x]$ a $T[y]$ sú teploty v dotyčné dva dni. Aby Bonifác Ignácov výrok nevedel vyvrátiť, musí platiť, že v žiaden z dní medzi x a y nemohla byť teplota medzi $T[x]$ a $T[y]$, vrátane. No a aby Ignácov výrok bol čo najohurúcejší, musí byť $y - x$ (teda počet dní, ktoré medzi spomínanými udalosťami uplynuli) čo najväčšie.

Súťažná úloha

Dané je pole $T[0..n-1]$, pričom $T[i]$ je priemerná teplota pred i dňami. Nájdite indexy $x < y$ také, že platí:

- Pre každé i také, že $x < i < y$, platí buď $T[i] < \min(T[x], T[y])$ alebo $T[i] > \max(T[x], T[y])$.
- Rozdiel $y - x$ je najväčší možný.
- Ak je stále takých dvojíc (x, y) viac, tak chceme tú, kde je y najväčšie možné.

Formát vstupu a výstupu

V prvom riadku vstupu je počet dní n. V druhom riadku vstupu je n medzerami oddelených celých čísel $T[0]$ až $T[n-1]$. Na výstup vypíšte jediný riadok a v ňom hľadané dve celé čísla x a y, oddelené medzerou.

Obmedzenia a hodnotenie

Vždy bude platiť $n \geq 2$ a pre všetky i bude $-10^9 \leq T[i] \leq 10^9$.
Riešenia budú testované na desiatich sadách testovacích vstupov; za každú z nich sa dá získať 1 bod. Maximálnu hodnotu n v jednotlivých sadách udávame v nasledujúcej tabuľke.

sada	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
max. n	10	20	100	1500	6000	7000	50 000	65 000	85 000	100 000

Navyše bude platiť:
V sadách #1, #4 a #7 obsahuje pole T práve raz každú z hodnôt od 1 po n.
V sadách #2, #5 a #8 obsahuje pole T len hodnoty od 1 po n (ale každú ľubovoľne veľakrát).

Príklady

vstup

```
8
-5 10 32 17 24 -12 13 19
```

výstup

```
1 6
```

Pred $x = 1$ dňom bola teplota 10 stupňov, pred $y = 6$ dňami to bolo 13 stupňov. Žiadna z teplôt medzi nimi (32, 17, 24 ani -12) neleží v intervale $\langle 10, 13 \rangle$, takže tento výrok Bonifác skutočne nevie vyvrátiť.

No a žiaden dlhší interval už nevyhovuje – napr. nemôže byť $(x, y) = (1, 7)$, lebo $T[3] = 17$ leží medzi $T[1] = 10$ a $T[7] = 19$. Na záver si všimnite, že dvojica $(x, y) = (0, 5)$ síce vyhovuje a má rovnakú dĺžku, ale nami vypísaná dvojica má väčšiu hodnotu y.

vstup

```
5
7 7 7 7 7
```

výstup

```
3 4
```

Tu sa nedá nič robiť, musíme zvoliť $y = x + 1$. Následne zvolíme najväčšie možné y.

A-I-2 Dva ploty

Toto je praktická úloha. Pomocou webového rozhrania odovzdajte ***funkčný, odladený program****.*

Marika má na dedine babku. Marikina babka má sad a v ňom tie najlepšie ovocné stromy: hrušky maslovky. Lenže na hrušky sa naučila chodiť celá dedina, a tak keď Marika príde babku pozrieť cez jesenné prázdniny, po hruškách už ani chýru, ani slychu.
Peťko jej chce pomôcť. Zohnal si nejaké koly, laty, klince a kladivo a ide okolo stromov postaviť plot. Už-už sa chcel pustiť do práce, keď sa zháčil. Čo tak postaviť tie ploty dva? Nebolo by na ne treba menej materiálov?

Súťažná úloha

Daných je $n \geq 3$ bodov v rovine. Hľadáme jeden alebo dva mnohouholníky také, že:

- každý z n daných bodov leží vo vnútri alebo na obvode aspoň jedného z mnohouholníkov,
- každý vrchol každého mnohouholníka je v niektorom z daných bodov,
- žiadny mnohouholník nemá nulový obsah.

Vypočítajte najmenšiu možnú hodnotu celkového obvodu nájdených mnohouholníkov.

Formát vstupu a výstupu

V prvom riadku vstupu je číslo n. V každom z nasledujúcich n riadkov sú dve celé čísla x_i, y_i: súradnice jedného z bodov. Platí $1 \leq x_i, y_i \leq 10^6$.

Na výstup vypíšte jeden riadok a v ňom jedno reálne číslo: najmenšiu celkovú dĺžku plotu. Vypíšte aspoň 6 miest za desatinnou bodkou. Odpovede s relatívnou chybou nanajvýš 10^{-6} budú považované za správne.

Obmedzenia a hodnotenie

Riešenia budú testované na desiatich sadách testovacích vstupov; za každú z nich sa dá získať 1 bod.
Maximálne n v jednotlivých sadách vstupov bude nasledovné: $(3, 5, 8, 18, 18, 50, 50, 200, 300, 300)$.
V sadách #1, #2, #4, #6 a #9 bude platiť, že žiadne tri body neležia na jednej priamke.
V sadách #1, #6 a #8 budú použité len vstupy, pre ktoré je optimálnym riešením postaviť jeden plot.

Príklady

vstup

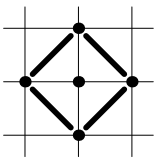
```
5
2 3
2 1
3 2
1 2
2 2
```

výstup

```
5.65685425
```

Oplatí sa postaviť štvorcový plot. Jeho presná dĺžka obvodu je $4\sqrt{2}$.

Za správne považujeme odpovede cca. z rozsahu $[5.6568486, 5.6568599]$.

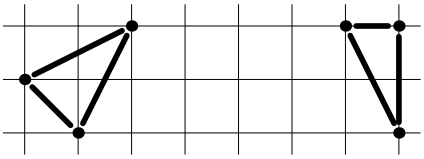


vstup

```
6
2 1
8 1
7 3
1 2
3 3
8 3
```

výstup

```
11.12241749487
```



A-I-3 Kaviarne

Toto je teoretická úloha. Pomocou webového rozhrania odovzdajte súbor vo formáte PDF, obsahujúci riešenie, spĺňajúce požiadavky uvedené v pravidlách.

Keď sa Dávidko sťahoval do New Yorku, potreboval si nájsť niekde na Manhattane bývanie. Niektorí ľudia sa snažia hľadať bývanie blízko pri práci, iní v peknom prostredí, no Dávidkova priorita bola jasná: *káva*. Plánuje pravidelne navštevovať aspoň k kaviarní v okolí svojho bytu.
Pre jednoduchosť si Manhattan predstavíme ako štvorcovú sieť, po ktorej sa dá pohybovať len v štyroch základných smeroch. Na niektorých mrežových bodoch (križovatkách) sú kaviarne; na každom najviac jedna.

Súťažná úloha

Na vstupe je číslo k a mapa Manhattanu. Pre každú križovatku spočítajte najmenšie d, pre ktoré platí: ak by Dávidko býval na tejto križovatke a bol ochotný prejsť vzdialenosť d, mal by na výber aspoň k kaviarní, do ktorých sa vie dostať.

Formát vstupu

V prvom riadku je číslo k.
V druhom riadku je rozmer n štvorcovej siete predstavujúcej Manhattan: tvorí ho n vodorovných a n zvislých ciest. Máme teda presne n^2 križovatiek.
Zvyšok vstupu tvorí n riadkov, v každom z nich je n čísel: 1 predstavuje križovatku s kaviarňou, 0 križovatku bez kaviarne. Dokopy je v Manhattane aspoň k kaviarní.

Formát výstupu

Vypíšte n riadkov a v každom z nich n čísel: pre každú križovatku vzdialenosť d takú, že do vzdialenosti d vrátane od dotyčnej križovatky leží aspoň k kaviarní.

Hodnotenie

Plných 10 bodov dostanete za riešenie s asymptoticky optimálnou časovou zložitosťou. Pomalšie korektné riešenia môžu dostať 4 až 8 bodov podľa konkrétnej časovej zložitosti.
Ak úlohu neviete riešiť, môžete dostať 3 body za vyriešenie špeciálneho prípadu $k = 1$.

Príklady

vstup

```
1
4
0 0 0 0
0 0 0 1
1 0 0 0
1 0 0 0
```

výstup

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

Pre $k = 1$ hľadáme vzdialenosť do najbližšej kaviarne.

vstup

```
3
5
1 0 0 0 0
0 0 0 1 0
1 0 0 0 1
1 0 0 1 0
0 0 0 1 0
```

výstup

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

A-I-4 Mimozemské počítače

Toto je teoretická úloha. Pomocou webového rozhrania odovzdajte súbor vo formáte PDF, obsahujúci riešenie, spĺňajúce požiadavky uvedené v pravidlách.
K tejto úlohe patrí študijný text uvedený na nasledujúcich stranách. Odporúčame najskôr prečítať ten a až potom sa vrátiť k samotným súťažným úlohám.
Jednotlivé podúlohy spolu nesúvisia, môžete ich riešiť v ľubovoľnom poradí.
Na časovej zložitosti vašich algoritmov pri hodnotení nezáleží – len musí byť polynomiálna.

Podúloha A (3 body)

Mimozemšťania nám dodali sálový KSP, ktorý rozhoduje problém existencie Hamiltonovskej kružnice v danom jednoduchom neorientovanom grafe.

Tento KSP má teda funkciu `kruznica(n,E)`. Táto funkcia čaká ako parametre počet n vrcholov grafu a zoznam E jeho hrán. Ak v danom grafe existuje Hamiltonovská kružnica, KSP rozsvieti zelené svetlo, inak rozsvieti červené.

Na vstupe dostanete jednoduchý neorientovaný graf G. Napíšte program s polynomiálnou časovou zložitosťou, ktorý rozsvieti zelené alebo červené svetlo podľa toho, či G obsahuje aspoň jednu Hamiltonovskú cestu.

Podúloha B (3 body)

A teraz naopak. Mimozemšťania nám dodali sálový KSP, ktorý rozhoduje problém existencie aspoň jednej Hamiltonovskej cesty v danom jednoduchom neorientovanom grafe.

Tento KSP má teda funkciu `cesta(n,E)`. Táto funkcia čaká ako parametre počet n vrcholov grafu a zoznam E jeho hrán (každá hrana je dvojica čísel od 0 po $n-1$). Ak v danom grafe pre nejaké u a v existuje Hamiltonovská cesta z u do v, KSP rozsvieti zelené svetlo, inak rozsvieti červené.

Na vstupe dostanete jednoduchý neorientovaný graf G. Napíšte program s polynomiálnou časovou zložitosťou, ktorý rozsvieti zelené alebo červené svetlo podľa toho, či G obsahuje Hamiltonovskú kružnicu.

Podúloha C (4 body)

Mimozemšťania nám dodali kufríkový KSP, ktorý rozhoduje problém existencie 3-farbenia v zadanom grafe.

Tento KSP má teda funkciu `je_trojfarbitelny(n,E)`. Táto funkcia čaká ako parametre počet n vrcholov grafu a zoznam E jeho hrán. Na výstupe táto funkcia vracia `True` ak sa dá tento graf ofarbiť tromi farbami a `False`, ak sa ofarbiť nedá. Túto funkciu môžeme v našom programe volať ľubovoľne veľakrát pre ľubovoľné grafy.

Na vstupe dostanete jednoduchý neorientovaný graf G, ktorý sa dá ofarbiť tromi farbami. Napíšte program s polynomiálnou časovou zložitosťou, ktorý jedno takéto ofarbenie nájde.

Programovacie jazyky

Vo svojich riešeniach môžete používať ľubovoľný štrukturovaný programovací jazyk. Vhodne si zvoľte potrebné dátové štruktúry. Napr. v Pascale by funkcia z podúlohy A mohla vyzerať nasledovne:

```
type hrana = array[0..1] of longint;
procedure kruznica(n : longint; m : longint; E : array of hrana);
```

a v C++ môžeme použiť buď nízkoúrovňové polia:

```
void kruznica(int n, int m, int E[][2]);
```

alebo trebárs aj vektor dvojíc:

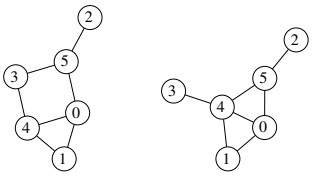
```
void kruznica(int n, vector< pair<int,int> > E);
```

Študijný text: zoznam problémov

Hlavný študijný text, uvedený na ďalších stranách, sa odkazuje na niekoľko problémov, ako napr. „existencia Hamiltonovskej kružnice“. Na tejto strane uvádzame stručné definície týchto problémov.

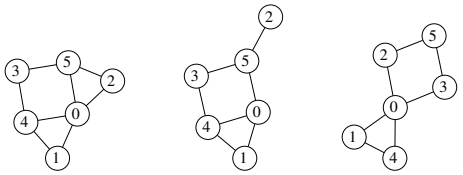
Jednoduchý neorientovaný graf je usporiadaná dvojica (V, E). V je konečná množina objektov nazývaných vrcholy. E je konečná množina dvojíc vrcholov; jej prvky voláme hrany. Takýto graf si môžeme predstaviť napríklad ako cestnú sieť: V je množina miest a E je množina dvojíc miest spojených cestami. Počet vrcholov budeme označovať n, počet hrán bude m. Jednotlivé vrcholy budeme číslovať od 0 po $n - 1$.

Hamiltonovská cesta z u do v:
Daný je jednoduchý neorientovaný graf G a jeho dva rôzne vrcholy u a v. Existuje v grafe G cesta, ktorá začína vo vrchole u, končí vo vrchole v a navštívi každý vrchol grafu G práve raz?



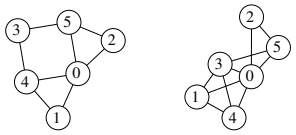
Graf vľavo obsahuje Hamiltonovskú cestu z 1 do 2: dá sa ísť postupne cez vrcholy 1, 0, 4, 3, 5 a 2.
Graf vpravo Hamiltonovskú cestu z 1 do 2 neobsahuje: ak chceme navštíviť vrchol 3, pôjdeme 2× cez vrchol 4.

Hamiltonovská kružnica:
Daný je jednoduchý neorientovaný graf G s $n \geq 3$ vrcholmi. Existuje v grafe G kružnica, ktorá navštívi každý vrchol grafu G práve raz?



Graf vľavo obsahuje Hamiltonovskú kružnicu: napr. začneme v 1 a postupne ideme do 0, 2, 5, 3, 4 a späť do 1.
Graf v strede ani graf vpravo Hamiltonovskú kružnicu neobsahujú.

Farbenie grafu k farbami:
Daný je jednoduchý neorientovaný graf G. Každému vrcholu grafu G chceme priradiť jednu z k možných farieb (pre naše pohodlie označených číslami od 0 po $k - 1$). Pritom ale musíme dodržať pravidlo, že žiadna hrana nesmie spájať dva vrcholy rovnakej farby.



Pre graf vľavo a $k = 2$ ofarbenie neexistuje. Napr. vrcholy 0, 1 a 4 musia zjavne mať navzájom rôzne farby.
Pre graf vľavo a $k = 3$ môžeme napr. dať vrcholom 1, 2, 3 farbu 0, vrcholom 4 a 5 farbu 1 a vrcholu 0 farbu 2.
Pre graf vpravo a $k = 3$ žiadne prípustné ofarbenie vrcholov neexistuje.

---

k-partícia postupnosti:
Daná je postupnosť n kladných celých čísel. Dá sa ich rozdeliť do k disjunktných skupín (partícií) tak, aby čísla v každej skupine mali presne rovnaký súčet?

Pre postupnosť $(3, 1, 4, 1, 5, 8)$ a $k = 2$ je odpoveď „áno“: jedno vyhovujúce delenie je na $3 + 8$ a $1 + 4 + 1 + 5$.
Pre postupnosť $(3, 1, 4, 1, 5, 9)$ a $k = 2$ je odpoveď „nie“.
Pre postupnosť $(3, 1, 4, 1, 5, 1)$ a $k = 3$ je odpoveď „áno“: jedno vyhovujúce delenie je na $3 + 1 + 1$, $1 + 4$ a 5.
Pre postupnosť $(2, 2, 2, 2, 2, 2, 12)$ a $k = 3$ je odpoveď „nie“.

Študijný text: konkrétne semiorganické počítače

V tomto študijnom texte v listingoch programov používame Python. Na webe olympiády neskôr nájdete obšírnejšiu verziu s listingami aj v C++ a Pascale.

Píše sa rok 2113. Zem pred pár rokmi kontaktovala vyspelá mimozemská rasa z planéty Žblnk. Títo mimozemšťania nás zásobujú novými konkrétnymi semiorganickými počítačmi (KSP), ktoré vedia riešiť rôzne konkrétne výpočtové úlohy. Vy ste členmi elitného výskumného tímu, ktorý má jediný cieľ: integrovať tieto KSP do normálnych počítačov a prinútiť ich riešiť naše problémy.
Mimozemským počítačom však vôbec nerozumieme, všetky snahy o ich rozobranie sa stretajú s neúspechom. A teda jediný spôsob, ako ich vieme využiť, je zadať im vstup a počkať si na výstup. Tu je prvá zaujímavosť: u KSP nezáleží na veľkosti vstupu ani na probléme, ktorý daný KSP rieši. Keď ľubovoľný KSP spustíme na ľubovoľnom vstupe, odpoveď vždy dostaneme o presne 47 stotín sekundy. (Pri odhade časovej zložitosti programov toto považujeme za konštantu.)

Mimozemšťania nám dodávajú dva druhy KSP: *kufríkové* a *sálové*.

Kufríkový KSP má tvar kufríka, ktorý má na sebe dva porty. Do jedného vieme pripojiť kábel so vstupom, do druhého zas kábel, po ktorom nám KSP pošle výstup. Kufríkový KSP teda vieme používať v našom programe ľubovoľne veľakrát, s navzájom rôznymi vstupmi.

Ukážkové zadanie 1. Mimozemšťania nám dodali kufríkový KSP, ktorý rozhoduje problém existencie Hamiltonovskej cesty z u do v. Tento KSP teda počíta funkciu `cesta(n,E,u,v)`. Táto funkcia čaká ako parametre počet n vrcholov grafu, zoznam E jeho hrán a dve čísla vrcholov u a v. (E je zoznam m dvojíc čísel, každé z rozsahu od 0 po $n-1$.) Na výstupe funkcia `cesta` vráti `True` alebo `False` podľa toho, či dotyčný graf obsahuje dotyčnú Hamiltonovskú cestu.

Našou úlohou je napísať program, ktorý bude v polynomiálnom čase riešiť problém existencie Hamiltonovskej kružnice. Na vstupe teda dostaneme jednoduchý neorientovaný graf s $n \geq 3$ vrcholmi, o ktorom máme zistiť, či obsahuje Hamiltonovskú kružnicu.

Analýza zadania. Musíme teda napísať program, ktorý spraví nasledovné:

1. Na vstupe dostane n (počet vrcholov grafu) a E (zoznam hrán grafu)
2. Urobí nejaké výpočty, počas ktorých môže ľubovoľne veľakrát používať funkciu `cesta`.
3. Vráti na výstup `True` alebo `False` podľa toho, či vstupný graf obsahuje Hamiltonovskú kružnicu.

Riešenie.

```
def kruznica(n,E):
    for (x,y) in E:                                          # pre kazdu hranu (x,y) v zozname hran E:
        newE = [ (u,v)   for (u,v) in E   if (u,v) != (x,y) ]     # newE = E okrem hrany (x,y)
        if cesta(n,newE,x,y): return True
    return False
```

Popis riešenia. Postupne pre každú hranu (x, y) zadaného grafu si položíme otázku: „Existuje Hamiltonovská kružnica prechádzajúca hranou (x, y)?“ Kedy takáto kružnica existuje? Práve vtedy, keď vo zvyšku grafu existuje Hamiltonovská cesta z x do y. Vyrobíme si teda nový zoznam hrán `newE`, do ktorého dáme všetky hrany okrem (x, y), a na takýto graf zavoláme funkciu `cesta` nášho kufríkového KSP.
Ak v pôvodnom grafe nejaká Hamiltonovská kružnica existuje, náš program ju nájde a dá odpoveď `True`, len čo vyskúšame niektorú z jej hrán ako (x, y). A naopak, ak náš program niekedy dá odpoveď `True`, tak v pôvodnom grafe existuje Hamiltonovská cesta z nejakého x do nejakého y, no a tá spolu s hranou (x, y) tvorí hľadanú kružnicu. Náš program teda naozaj vždy robí to, čo má.
Časová zložitosť nášho programu je $\Theta(m^2)$, kde m je počet hrán zadaného grafu. Keďže v jednoduchom neorientovanom grafe platí $m \leq n(n-1)/2$, zhora môžeme našu časovú zložitosť odhadnúť ako $O(n^4)$.

Poznámka. Existuje aj efektívnejšie riešenie. Stačí si uvedomiť, že pred hľadaním Hamiltonovskej cesty z x do y vôbec netreba hranu (x, y) z grafu odstraňovať. Hamiltonovská cesta z x do y ju v grafe s $n \geq 3$ vrcholmi aj tak nemôže obsahovať. Stačí teda pre každú hranu (x, y) zistiť, či platí `cesta(n,E,x,y)`. Ďalej, Hamiltonovská kružnica musí prechádzať vrcholom 0, stačí teda namiesto každej hrany skúšať len hrany idúce z vrcholu 0.

Sálový KSP je obrovský a jeho jediným výstupom sú dve svetlá: červené a zelené. S týmto výstupom ďalej nepracujeme.

Ukážkové zadanie 2. Mimozemšťania nám dodali sálový KSP, ktorý rozhoduje problém 3-partície. Tento KSP má teda funkciu `tri_particia(X)`, ktorá rozsvieti zelené alebo červené svetlo podľa toho, či postupnosť X má 3-partíciu – teda či sa jej prvky dajú rozdeliť do *troch* skupín s navzájom rovnakým súčtom.

Na vstupe dostanete postupnosť kladných celých čísel A. Napíšte program s polynomiálnou časovou zložitosťou, ktorý rozsvieti zelené alebo červené svetlo podľa toho, či má postupnosť A 2-partíciu (teda podľa toho, či vieme prvky A rozdeliť do *dvoch* skupín s rovnakým súčtom).

Analýza zadania. Musíme teda napísať program, ktorý spraví nasledovné:

1. Na vstupe dostane postupnosť čísel A.
2. Urobí nejaké výpočty a vyrobí nejakú novú postupnosť čísel X.
3. Na konci (každej možnej vetvy) výpočtu raz zavolá funkciu `tri_particia(X)`, ktorá rozsvieti správne svetlo.

Riešenie.

```
def dva_particia(A):
    s = sum(A)
    if s%2 == 0:                # s%2 je zvysok cisla s po deleni 2
        X = A + [ s//2 ]        # // je celociselne delenie
    else:
        X = [1]                 # [1] je postupnost ktora urcite 3-particiu nema
    tri_particia(X)
```

Popis riešenia. Nech sme na vstupe dostali postupnosť $A = (a_1, \ldots, a_n)$. Spočítame si jej súčet s.
Ak je s párne, pridáme na koniec vstupnej postupnosti ešte jeden prvok s hodnotou $s/2$. Výslednú postupnosť pošleme do KSP. Ten nám odpovie, či má táto nová postupnosť 3-partíciu. Lenže táto nová postupnosť má 3-partíciu vtedy a len vtedy, keď mala naša pôvodná postupnosť 2-partíciu. KSP teda za nás práve vyriešil našu úlohu.
(Prečo platí vyššie spomenutá ekvivalencia? V novej postupnosti X je súčet všetkých prvkov rovný $3s/2$. Ak teda existuje 3-partícia X, tak každá zo skupín, na ktoré X rozdelíme, má súčet presne $s/2$. Ale potom zjavne jednu z týchto troch skupín tvorí samotný nami pridaný prvok s hodnotou $s/2$. A teda 3-partícia našej novej postupnosti existuje práve vtedy, keď sa dá ostatné prvky rozdeliť na dve skupiny s rovnakým súčtom – teda práve vtedy, keď pôvodná postupnosť A mala 2-partíciu.)
No a ak je s nepárne, vieme, že je odpoveď *nie*. Do KSP preto chceme poslať ľubovoľný vstup, pre ktorý vopred vieme, že KSP dá zápornú odpoveď. Takýmto vstupom je napr. $X = (1)$ alebo $X = (1, 1, 2)$.
Časová zložitosť tohto programu je lineárna od dĺžky postupnosti X.

DVADSIATY DEVIATY ROČNÍK OLYMPIÁDY V INFORMATIKE

Príprava úloh: Michal Forišek, Michal Anderle
Recenzia: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: IUVENTA – Slovenský inštitút mládeže, Bratislava 2013