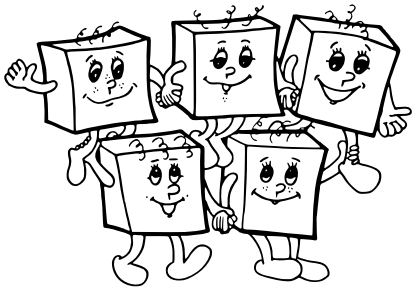
OLYMPIÁDA V INFORMATIKE

NA STREDNÝCH ŠKOLÁCH

http://oi.sk/



dvadsiaty deviaty ročník

školský rok 2013/2014

zadania celoštátneho kola, deň 1
kategória A

Priebeh celoštátneho kola

Celoštátne kolo 29. ročníka Olympiády v informatike, kategórie A, sa koná v dňoch 26.–29. marca 2014. Na riešenie úloh prvého, teoretického dňa majú súťažiaci 4,5 hodiny čistého času. Rôzne úlohy riešia súťažiaci na samostatné listy papiera. Akékoľvek pomôcky okrem písacích potrieb (napr. knihy, výpisy programov, kalkulačky) sú zakázané.

Čo má obsahovať riešenie úlohy?

- Slovne popíšte algoritmus.
  Slovný popis riešenia musí byť jasný a zrozumiteľný i bez nahliadnutia do samotného algoritmu/programu.
- Zdôvodnite správnosť vášho algoritmu.
- Uveďte a zdôvodnite jeho časovú a pamäťovú zložitosť.
- Podrobne uveďte dôležité časti algoritmu, ideálne vo forme programu v Pascale alebo C/C++.
- V prípade, že používate vo svojom programovacom jazyku knižnice, ktoré obsahujú implementované dátové štruktúry a algoritmy (napr. STL pre C++), v popise algoritmu stručne vysvetlite, ako by ste napísali program s rovnakou časovou zložitosťou bez použitia knižnice.

Hodnotenie riešení prvého (teoretického) dňa

Za každú úlohu môžete získať od 0 do 10 bodov.

Pokiaľ nie je v zadaní povedané ináč, najdôležitejšie dve kritériá hodnotenia sú v prvom rade **správnosť** a v druhom rade **efektívnosť** navrhnutého algoritmu. Na výslednom počte bodov sa môže prejaviť aj kvalita popisu riešenia a zdôvodnenie tvrdení o jeho správnosti a efektívnosti.

Efektívnosť algoritmu posudzujeme vypočítaním jeho časovej zložitosti – funkcie, ktorá hovorí, ako dlho vykonanie algoritmu trvá v závislosti od veľkosti vstupných parametrov. Nezávisí pri tom na konštantných faktoroch, len na rádovej rýchlosti rastu tejto funkcie.

V zadaní úlohy môžu byť uvedené limity na veľkosť premenných. Tieto môžete použiť na odhad toho, ako dobré vaše riešenie je. Na počítači, ktorý vykoná miliardu inštrukcií za sekundu, vyrieši vzorové riešenie ľubovoľný povolený vstup nanajvýš za niekoľko sekúnd.

A-III-1 Pán Buridan a kaviarne

Buridanov osol je známy filozofický myšlienkový experiment. Predstavte si, že hladného osla postavíte presne do stredu medzi dve kôpky sena. Osol by si síce dal seno, ale keďže je situácia dokonale symetrická, nemá sa podľa čoho rozhodnúť, či ísť doľava alebo doprava. A tak ostane na mieste, až chudák zdochne od hladu.

Dávidka už poznáte z domáceho kola. Stále si chudák hľadá bývanie v Manhattane. Teraz si však uvedomil, že mu hrozí rovnaký problém ako Buridanovmu oslovi: ak by existovali dve kaviarne, ktoré sú od jeho bytu rovnako ďaleko, mohlo by sa mu stať, že sa medzi nimi nebude vedieť rozhodnúť a nedopadne to dobre.

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 mapa Manhattanu. Pre každú križovatku zistite, či tam Dávidko môže bývať – teda či neexistujú žiadne dve kaviarne, ktoré by boli od danej križovatky rovnako ďaleko.

Formát vstupu

V prvom 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.

Formát výstupu

Vypíšte n riadkov a v každom z nich n znakov: pre každú križovatku buď '`A`' ak tam Dávidko bývať môže, alebo '`N`' ak nie.

Hodnotenie

Plných 10 bodov dostanete za riešenie s asymptoticky optimálnou časovou zložitosťou. Pomalšie korektné riešenia môžu dostať 3 až 9 bodov podľa konkrétnej časovej zložitosti.

Príklady

vstup

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

výstup

```
A A A A A
A A A A N
N N N N A
A A A A A
A A A A A
```

vstup

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

výstup

```
N N N
N N N
N N N
```

vstup

```
2
0 1
0 0
```

výstup

```
A A
A A
```

A-III-2 Preťahovanie lanom

V Dolebrežnej Vieske sa koná tradičný turnaj v preťahovaní lanom. Keďže v tejto obci nie je nikde žiadna rovina, hrá sa turnaj na úbočí kopca. Sklonu kopca zodpovedá koeficient $q > 1$. Do turnaja sa prihlásilo n hráčov. V poradí od najmladšieho po najstaršieho dostali čísla 1 až n. Silu hráča i označíme s_i.

Na začiatku turnaja sú všetci hráči aktívni. Turnaj pozostáva z viacerých zápasov. V každom zápase hrajú všetci aktívni hráči. Na zápas sa aktívni hráči rozdelia do dvoch tímov. Horný tím (na vrchu kopca) tvorí k najstarších aktívnych hráčov, dolný (na jeho spodku) tvoria všetci ostatní. Tím ťahajúci dohora má zjavne nevýhodu: vyhrajú práve vtedy, ak ich súčet síl je ostro väčší ako q-násobok súčtu síl tímu ťahajúceho dodola. Turnaj končí, akonáhle v niektorom zápase vyhrá horné družstvo.

Po každom zápase jeden z hráčov prestane byť aktívny. Domorodci si všimli, že je to vždy buď najmladší aktívny hráč (ktorý už musí ísť domov), alebo najstarší aktívny hráč (ktorý odíde do bufetu). Hodnota k sa nemení, teda ak po zápase odišiel najstarší hráč, v nasledujúcom zápase bude za horný tím hrať aj najstarší z hráčov, ktorí doteraz hrali za dolný tím.

Súťažná úloha

Na vstupe dostanete počet hráčov n, ich sily $s_1, \ldots, s_n$, koeficient strmosti kopca q a veľkosť horného tímu k. Napíšte program, ktorý vypočíta, koľko najviac zápasov sa môže v danom turnaji odohrať. Inými slovami, váš program by mal nájsť také poradie odchádzajúcich hráčov, pri ktorom prvá výhra horného tímu nastane čo najneskôr.

Formát vstupu a výstupu

V prvom riadku vstupu sú celé čísla n a k a racionálne číslo q. Môžete predpokladať, že $1 \leq k \leq n$, že $q > 1$ a že všetky vaše výpočty s číslom q dávajú presné výsledky.

V riadku druhom sú sily $s_1, \ldots, s_n$. Môžete predpokladať, že sú to kladné celé čísla, ktorých súčet sa zmestí do bežnej číselnej premennej.

Na výstup vypíšte jediný riadok a v ňom jedno celé číslo: maximálny možný počet zápasov.

Hodnotenie

Riešenia s časovou zložitosťou exponenciálnou od n získajú nanajvýš 3 body. Ľubovoľné riešenie s časovou zložitosťou polynomiálnou od n môže (v prípade, že je kompletné a korektné) získať aspoň 5 bodov. Časovú zložitosť optimálneho riešenia vám úmyselne neprezradíme. Nezabudnite zdôvodniť správnosť vášho algoritmu!

Príklad

vstup	výstup
5 2 1.01 40 70 1000 30 20	4

Tento kopec je skoro rovina, horný tím takmer nemá výhodu. Hore začínajú hráči so silou 30+20 = 50 a dole hráči so silou 40+70+1000 = 1110. Keďže $1110 \times 1.01 > 50$*, dolný tím prvý zápas poľahky vyhrá.*

Turnaj bude najdlhšie trvať vtedy, keď po prvom zápase odíde najmladší hráč (so silou 40) a po druhom zápase odíde opäť najmladší hráč (so silou 70). Po treťom zápase je jedno kto odíde – na štvrtý zápas tak či onak ostanú hore dvaja hráči a dole nikto, a teda horný tím poľahky vyhrá.

vstup	výstup
12 7 2.0 2 4 9 5 3 3 4 8 8 6 1 6	4

Jeden možný optimálny priebeh zápasov: po prvom zápase odíde najmladší (sila 2), po druhom najstarší (sila 6), po treťom zase najmladší (sila 4). Na štvrtý zápas teraz na vrchu kopca nastúpia hráči so silou $3+3+4+8+8+6+1 = 33$*, zatiaľ čo na spodku kopca už budú len hráči so silou* $9+5 = 14$*. No a keďže* $14 \times 2.0 < 33$*, štvrtý zápas už vyhrá horný tím a turnaj končí.*

A-III-3 Mimozemské počítače

K tejto úlohe patrí aj študijný text, ktorý nájdete na nasledujúcich stranách. Študijný text sa zhoduje so študijným textom z domáceho a krajského kola. Nedefinujeme v ňom žiadne nové problémy.

Podúloha A (1 bod)

Mimozemšťania nám dodali sálový KSP, ktorý rozhoduje problém existencie Hamiltonovskej cesty obsahujúcej predpísanú hranu. Tento KSP má teda funkciu `cesta_s_hranou(n,E,u,v)`. Táto funkcia čaká ako parametre počet n vrcholov grafu, zoznam E jeho hrán, a dve čísla vrcholov u a v. Ak v danom grafe existuje Hamiltonovská cesta, na ktorej u a v nasledujú bezprostredne po sebe, KSP rozsvieti zelené svetlo, inak rozsvieti červené. Túto funkciu môžete použiť **len raz** a jej volaním výpočet vášho programu končí.

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 (5 bodov)

Mimozemšťania nám dodali kufríkový KSP, ktorý rozhoduje problém existencie Hamiltonovskej cesty. Tento KSP má teda funkciu `cesta(n,E)`. Táto funkcia čaká ako parametre počet n vrcholov grafu a zoznam E jeho hrán. Na výstupe táto funkcia vracia `True` alebo `False` podľa toho, či v dotyčnom grafe existuje aspoň jedna Hamiltonovská cesta. Funkciu `cesta` môžete volať aj viackrát.

Na vstupe dostanete jednoduchý neorientovaný graf G ktorý obsahuje Hamiltonovskú cestu. Napíšte pomocou tohto kufríkového KSP program, ktorý v polynomiálnom čase (nerátajúc volania funkcie `cesta`) v G jednu Hamiltonovskú cestu nájde.

Pozor! Počet bodov, ktoré dostanete, bude závisieť od toho, rádovo koľko volaní funkcie `cesta` vaše riešenie v najhoršom prípade potrebuje.

Podúloha C (4 body)

Mimozemšťania nám dodali sálový KSP, ktorý rozhoduje problém existencie Hamiltonovskej cesty. Tento KSP má teda funkciu `cesta(n,E)`. Táto funkcia čaká ako parametre počet n vrcholov grafu a zoznam E jeho hrán. Ak v danom grafe existuje Hamiltonovská cesta, KSP rozsvieti zelené svetlo, inak rozsvieti červené. Túto funkciu môžete použiť **len raz** a jej volaním výpočet vášho programu končí.

Na vstupe dostanete *orientovaný* graf G. Každá hrana tohto grafu má teda určený jeden smer, v ktorom sa po nej smie ísť. Napíšte program s polynomiálnou časovou zložitosťou, ktorý rozsvieti zelené alebo červené svetlo podľa toho, či G obsahuje aspoň jednu *orientovanú* Hamiltonovskú cestu. (Teda Hamiltonovskú cestu, ktorá každú svoju hranu použije v správnom smere.)

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 cesta_s_hranou(n : longint; m : longint; E : array of hrana; u,v : longint);
```

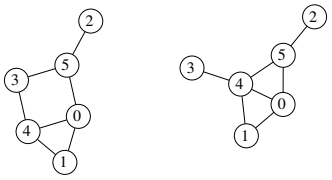
a v C++ môžeme použiť buď nízkoúrovňové polia, alebo trebárs aj vektor dvojíc:

```
void cesta_s_hranou(int n, int m, int E[][2], int u, int v);          // prva moznost
void cesta_s_hranou(int n, vector< pair<int,int> > E, int u, int v);  // druha moznost
```

Študijný text: zoznam 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 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.

Študijný text: konkrétne semiorganické počítače

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 Anderle, Vladimír Boža, Michal Forišek, Peter Fulla
Recenzia: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: IUVENTA – Slovenský inštitút mládeže, Bratislava 2014