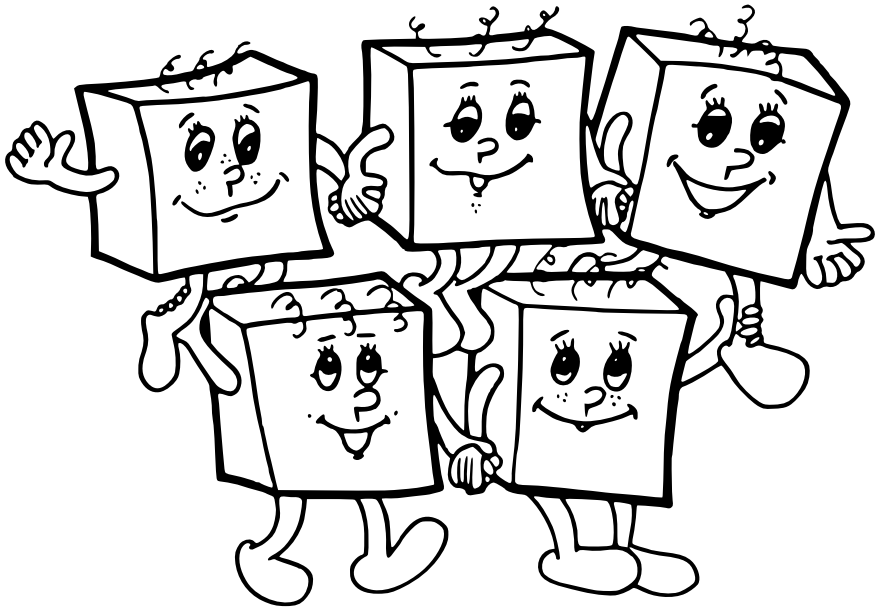
OLYMPIÁDA V INFORMATIKE
NA STREDNÝCH ŠKOLÁCH



dvadsiaty druhý ročník
školský rok 2006/07

zadania celoštátneho kola
kategória A

1. súťažný deň

- **Olympiáda v informatike** je od školského roku 2006/07 samostatnou súťažou. Predchádzajúcich 21 ročníkov tejto súťaže prebiehalo pod názvom **Matematická olympiáda, kategória P** (programovanie).
- Oficiálnu **webstránku** súťaže nájdete na `http://www.ksp.sk/oi/`.

Informácie a pravidlá

Priebeh celoštátneho kola

Celoštátne kolo 22. ročníka Olympiády v informatike, kategórie A, sa koná v dňoch 21. – 24. 4. 2007. 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?

Pokiaľ nie je v zadaní povedané ináč, základným kritériom hodnotenia riešenia je správnosť a efektívnosť navrhnutého algoritmu. Hodnotí sa aj kvalita popisu riešenia a zdôvodnenie tvrdení o jeho správnosti a efektívnosti.

Súčasťou riešenia by mali teda byť:

- **Popis riešenia,** to znamená slovný popis použitého algoritmu, argumenty zdôvodňujúce jeho správnosť (prípadne dôkaz správnosti algoritmu), diskusiu o efektivite vášho riešenia (časová a pamäťová zložitosť). Slovný popis riešenia musí byť jasný a zrozumiteľný i bez nahliadnutia do samotného zápisu algoritmu (do programu).
- **Program.** V úlohách **A-III-1** a **A-III-2** treba uviesť dostatočne podrobný zápis algoritmu, najlepšie v tvare zdrojového textu najdôležitejších častí programu v jazyku Pascal alebo C/C++. Zo zápisu môžete vynechať jednoduché operácie ako vstupy, výstupy, implementáciu jednoduchých matematických vzťahov a pod.

  V prípade, že používate vo svojom programovacom jazyku knižnice, ktoré obsahujú implementované dátové štruktúry a algoritmy (napr. STL pre C++), v popise algoritmu vysvetlite, ako sa časti programu, ktoré obsahujú volania knižnice, dajú implementovať bez nej bez zhoršenia časovej zložitosti.

  V úlohe **A-III-3** je namiesto klasického programu potrebné uviesť program pre grafomat.

Zadania prvého súťažného dňa

A-III-1 Pizza vracia úder

Marec, mesiac daňových priznaní, zbrzdil rozmach Marcovej firmy. Počas rozvážania pizze a zakladania nových pobočiek nemal Marco čas zaoberať sa účtovníctvom, a teraz s hrôzou zistil, že si do svojich poznámok zabudol zapísať, čo sú príjmy a čo výdaje. Napriek tomu nespanikáril, lebo si spomenul na príbehy, ktoré mu rozprával jeho dedo (bývalá hlava talianskej mafie) a rozhodol sa doplniť (čítaj: sfalšovať) účtovné záznamy.

Aby výsledok vyzeral čo najdôveryhodnejšie, rozhodol sa použiť čísla zo svojich poznámok a vybrať si pri každom čísle, či je to výdaj alebo príjem. Naviac sa rozhodol, že keď už falšovať, tak poriadne. Rád by vyzeral ako úspešný obchodník, a teda neskončil v strate. Na druhej strane by chcel platiť čo najmenšie dane, teda jeho zisk by mal byť čo najmenší. Pri riešení tejto neľahkej úlohy by mu mohlo pomôcť to, že veľkosť čísel v jeho poznámkach je podstatne menšia ako ich počet.

Súťažná úloha

Je daných N prirodzených čísel $a_1, \ldots, a_N$ z rozsahu 1 až K. Hodnota K je väčšinou rádovo menšia ako N. Vašou úlohou je nájsť čísla $s_i \in \{-1, +1\}$ také, že súčet $z = s_1a_1 + s_2a_2 + \ldots + s_Na_N$ je nezáporný a zároveň najmenší možný.

Formát vstupu

Prvý riadok obsahuje jediné celé číslo N udávajúce počet Marcovych účtovných záznamov.

Druhý riadok obsahuje N hodnôt $a_1, \ldots, a_N$ – jednotlivé účtovné záznamy.

Formát výstupu

Výstup obsahuje N riadkov. i-ty riadok obsahuje `+` ak má brať Marco i-tu položku ako príjem a `-` ak ju má brať ako výdaj.

Príklady

vstup

```
4
1 2 3 4
```

výstup

```
+
-
-
+
```

Platí $+a_1 - a_2 - a_3 + a_4 = 0$ *a lepšie to zjavne nejde.*

vstup

```
1000
2 3 3 3 3 3 ... 3 3 3 3 3 3
```

výstup

```
-
-
-
...
+
+
+
```

Veľkú časť vstupu aj výstupu sme pre prehľadnosť vynechali.
Vo výstupe je najskôr 500 *znakov* -, *potom* 500 *znakov* +. *Výsledný zisk je 1. Zisk 0 sa zjavne dosiahnuť nedá, preto je toto riešenie optimálne.*

A-III-2 Obdĺžnik

Súťažná úloha

V rovine je daných N navzájom rôznych bodov $B_1, \ldots, B_N$.

Vašou úlohou je navrhnúť algoritmus, ktorý nájde obdĺžnik $A_1A_2A_3A_4$, ktorého vrcholy sú niektoré zo zadaných bodov (t. j. $A_1 = B_i$ pre nejaké i, $1 \leq i \leq N$, analogicky pre A_2, A_3 a A_4). Hrany obdĺžnika $A_1A_2A_3A_4$ musia byť rovnobežné s osami, teda x-ové súradnice vrcholov A_1 a A_4 a vrcholov A_2 a A_3 musia byť rovnaké a rovnako y-ové súradnice vrcholov A_1 a A_2 a vrcholov A_3 a A_4 musia byť rovnaké.

Navyše počet bodov B_i, ktoré ležia vo vnútri alebo na obvode vami nájdeného obdĺžnika $A_1A_2A_3A_4$, musí byť maximálny možný.

Formát vstupu

Prvý riadok vstupu obsahuje jediné celé číslo N udávajúce počet zadaných bodov B_i. Nasleduje N riadkov, pričom i-ty riadok obsahuje súradnice i-teho bodu.

Formát výstupu

Výstup obsahuje jediné číslo – najvyšší možný počet bodov v obdĺžniku spĺňajúcom podmienky zo zadania. Ak sa zo zadaných bodov nedá vytvoriť žiadny vhodný obdĺžnik, vypíšte namiesto toho správu „`Neexistuje vhodný obdĺžnik`“.

Príklady

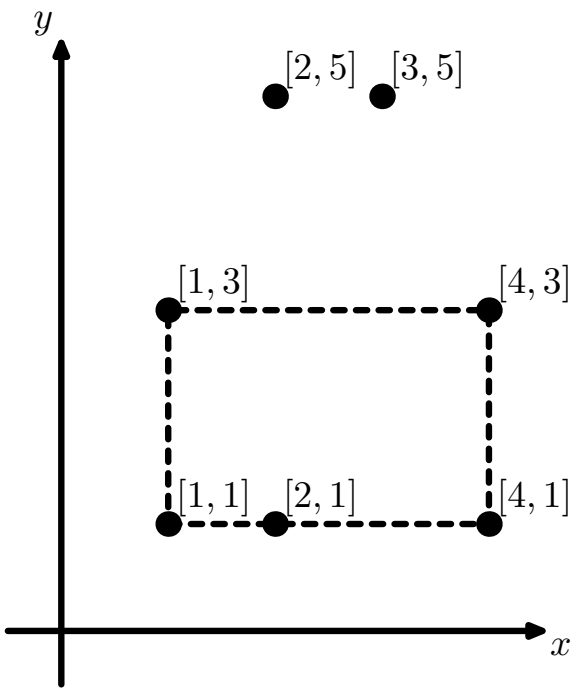
vstup

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

výstup

```
5
```

Takto vyzerá situácia popisovaná týmto vstupom a jediné optimálne riešenie:



vstup

```
4
1 1
2 1
1 2
4 7
```

výstup

```
Neexistuje vhodný obdĺžnik
```

A-III-3 Grafomat a šamani

V osadách indiánskeho kmeňa Apačov-Grafomatérov sa zajtra koná najväčšia slávnosť roka, pri ktorej majú šamani kmeňu zabezpečiť náklonnosť veľkého Manitú pri tohtoročnom love bizónov.

Rituál vyžaduje, aby sa indiáni v niektorých osadách pomaľovali na červeno a v ostatných osadách na modro. No a šamani vedia, že rituál bude úspešný len vtedy, ak bude „červených“ a „modrých“ osád rovnako. A tak teraz v každej osade šaman stojí pri svojom signálnom ohni a snaží sa so susednými osadami dohodnúť, akou farbou sa vlastne kde majú maľovať.

Súťažná úloha

Napíšte program pre grafomat, ktorý v zadanom 3-grafe s jedným označeným vrcholom označí presne polovicu vrcholov grafu a skončí.

Presnejšie, váš program sa má správať nasledovne: Na začiatku je v jednom vrchole grafu $x = 1$ a vo všetkých ostatných $x = 0$. Na konci výpočtu má byť v práve polovici vrcholov $y = 1$ (tieto vrcholy predstavujú „červené“ dediny) a v ostatných vrcholoch má byť $y = 0$.

Môžete predpokladať, že graf je súvislý a že má párny počet vrcholov, takže hľadané rozdelenie vždy existuje. (Dá sa dokázať, že dokonca každý 3-graf má párny počet vrcholov, ale pre riešenie úlohy to nie je podstatné.)

Rady a upozornenia

Ak si s úlohou neviete poradiť, rozmyslite si najskôr, ako by ste ju riešili, ak by použitým grafom nebol 3-graf ale strom (súvislý graf bez cyklov).

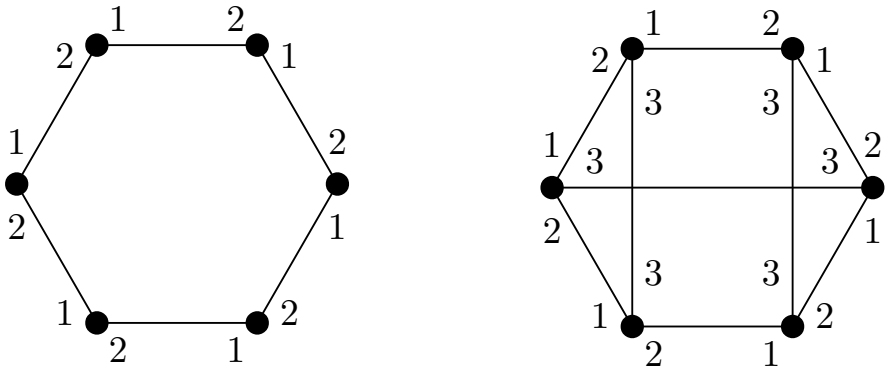
Dajte si pozor na to, že počet stavov každého automatu musí byť konečný, teda **nemôžete** používať premenné, ktorých veľkosť by závisela od veľkosti grafu.

Študijný text

Študijný text je identický s tým z krajského kola až na definíciu ukončenia výpočtu a príslušnú úpravu riešenia príkladov.

Grafom nazývame ľubovoľnú konečnú množinu V *vrcholov* grafu spolu s množinou E *hrán*, čo sú neusporiadané dvojice vrcholov. Každá hrana teda predstavuje spojenie medzi dvoma vrcholmi. Žiadne dva vrcholy nie sú spojené viac ako jednou hranou, žiadna hrana nespája vrchol sám so sebou. Počet hrán grafu budeme označovať N a počet jeho hrán M.

K-graf budeme hovoriť takému grafu, v ktorom z každého vrcholu vedie práve K hrán a konce týchto hrán sú očíslované prirodzenými číslami od 1 po K. Konce jednej hrany môžu byť očíslované rôzne. Pokiaľ budeme hovoriť o hranách vychádzajúcich z nejakého vrcholu v, budeme spomínať *miestne* čísla hrán (čísla toho konca, ktorým je v) a *vzdialené* čísla (to sú tie na opačných koncoch hrán). Nasledujúci obrázok ukazuje príklad 2-grafu a 3-grafu.



Ohodnotením grafu nazveme priradenie prvkov nejakej konečnej množiny vrcholom grafu - teda napríklad rozdelenie vrcholov na čierne a biele, alebo označenie vrcholov číslami od 1 po 5.

Grafomat je zariadenie na automatické riešenie grafových úloh. Jeho vstupom je ľubovoľný K-graf G spolu s jeho ohodnotením. Výstupom je nejaké ďalšie ohodnotenie toho istého grafu. Samotný výpočet je vykonávaný *automatmi* umiestnenými v jednotlivých vrcholoch grafu. Každý automat má svoju pamäť a riadi sa programom. Programy všetkých automatov sú identické. Okrem toho,

že automat môže ľubovoľne narábať so svojou pamäťou, môže aj nahliadať do pamäte svojich susedov.

Pamäť automatu si môžeme predstaviť ako pascalovské premenné typu interval. Teda každá premenná obsahuje jedno prirodzené číslo, ktoré je z nejakého pevne zvoleného rozsahu, ktorý nezávisí na veľkosti vstupu. Okrem toho je tiež možné používať pole takýchto premenných. Opäť, rozmery poľa musia byť dopredu známe a nesmú závisieť od veľkosti vstupu. Žiadne iné typy premenných (neobmedzene veľké čísla, smerníky, ...) sa nedajú používať.

Zvláštnu rolu hrajú premenné x a y. Premenná x na začiatku výpočtu obsahuje vstupné ohodnotenie toho vrcholu grafu, v ktorom program beží, hodnota premennej y na konci výpočtu určí výstupné ohodnotenie tohto vrcholu. Všetky premenné s výnimkou premennej x majú svoju počiatočnú hodnotu pevne určenú. Deklarácia premenných vyzerá napríklad takto:

```
var x: 1..5;          { číslo od 1 do 5, na začiatku je to vstup }
    y: 1..5 = 3;           { y je na začiatku 3, na konci výstup }
    z: array [1..2] of 3..4 = (3, 4);         { dvojprvkové pole }
```

Riadiaci program automatu si môžeme predstavit ako pascalovský program, v ktorom zakážeme používanie rekurzie a dovolíme manipulovať len s premennými v pamäti automatu a s premennými v pamäti susedných automatov. Na svoje vlastné premenné sa automat odkazuje ich menami, ako by to boli obyčajné pascalovské globálne premenné. Na to, aby sme vedeli odkazovať na premenné susedov, využijeme miestne čísla hrán. Presnejšie, nech `i` je celočíselný výraz s hodnotou z rozsahu $1 \ldots K$. Potom `S[i].p` je výraz predstavujúci premennú `p` u suseda, do ktorého od nás vedie hrana s miestnym číslom `i`. (Samozrejme, `i` môže byť ľubovoľný výraz a `p` ľubovoľné meno premennej.) Premenné susedov sa dajú len čítať.

Aby program mohol dávať do súvislosti svoje hrany s hranami svojich susedov, má k dispozící ešte premenne `P[1]` až `P[K]`, ktoré sú pevne nastavené tak, že `P[i]` obsahuje vzdialené číslo tej hrany, ktorá má miestne číslo `i`.

Použitie si ukážeme na výraze `S[i].S[P[i]].x`. Označme vrchol, v ktorom práve sme, písmenom v. Výrazy `S[i].volaco` sú premenné suseda, do ktorého sa dostaneme hranou s číslom `i`. Označme tohto suseda písmenom u. Premenná,

ktorú chceme, je `S[nieco].x`. Takže chceme premennú `x` od niektorého suseda vrcholu u. Ale ktorého? Toho, do ktorého sa dostaneme hranou s číslom `P[i]`. To je ale práve vzdialené číslo hrany, ktorou sme do u prišli. Inými slovami, v u je to jeho miestne číslo hrany, ktorá vedie späť do v. Takže výraz `S[i].S[P[i]].x` predstavuje to isté ako výraz `x`.

Ak sa vám to zdá zložité, povieme si to ešte raz v dvoch krokoch. Najskôr sa vyhodnotí `nieco=P[i]`. (`P[i]` je naša lokálna premenná.) Teraz sa pozrieme na premennú `S[i].S[nieco].x`. (To prvé `S` je naše lokálne pole s premennými susedov, to druhé je podobné pole u suseda.) No a tento sused v premennej `S[nieco].x` vidí to, čo máme v našej premennej `x`.

Výpočet automatu prebieha v taktoch, a to nasledovne: V nultom takte sa premenné všetkých automatov nastavia na počiatočnú hodnotu a premenné x na vstupné ohodnotenie jednotlivých vrcholov. V každom ďalšom takte sa vždy znova spustí program každého automatu, pričom premenné svojich susedov vidí program v stave, v akom boli na začiatku taktu. Aj keď jednotlivé automaty bežia súčasne, nemôže sa teda stať, že by jeden čítal z premennej, do ktorej práve druhý zapisuje.

ZMENA oproti krajskému kolu:
Výpočet pokračuje tak dlho, až kým po nejakom takte neplatí, že všetky premenné vo všetkých automatoch majú rovnakú hodnotu ako mali po predchádzajúcom takte. Inými slovami, výpočet končí, akonáhle sa v nejakom takte už nič nové nestane. Vtedy grafomat prečíta z premenných y výstupné ohodnotenie grafu.

Štruktúra grafu a obsahy premenných `P` zostávajú po celú dobu výpočtu nezmenené.

Za *časovú zložitosť* výpočtu budeme považovať počet taktov, ktoré ubehnú po zastavenie programu. Nijako teda nezávisí na rýchlosti jednotlivých automatov. Podobne ako je to u časovej zložitosti klasických algoritmov, nebudeme hľadieť na multiplikatívne konštanty a bude nás zaujímať len asymptotické správanie zložitosti – teda či je lineárna, kvadratická, atď. Prípady, kedy výpočet neskončí, nebudeme pripúšťať, pre úplnosť ale dodajme, že vtedy sa hodnoty premenných musia nutne opakovať.

Príklad 1

Je zadaný 3-graf a v ňom vyznačený jeden vrchol v, a to tak, že jeho premenná x bude inicializovaná jednotkou a ostatné vrcholy budú mať nulu. Napíšte program pre grafomat, ktorý označí všetky vrcholy, do ktorých sa dá dostať z vrcholu v po hranách, a to tak, že ich premenná y bude mať na konci výpočtu hodnotu jedna, pre ostatné vrcholy bude mať hodnotu nula.

Riešenie: Inšpirujeme sa prehľadávaním do šírky. V každom takte sa každý vrchol pozrie, či je niektorý z jeho susedov už označený. Ak je, tak sa sám označí. Pokiaľ sa označenie vrcholu nezmení, vrchol tým vlastne hovorí, že súhlasí so zastavením. Priebeh výpočtu bude teda vyzerať tak, že po i-tom takte budú označené tie vrcholy, ktorých vzdialenosť od v je menšia alebo rovná i. Výpočet sa zastaví vtedy, keď sa hodnoty premenných prestanú meniť, čo znamená, že po najviac N taktoch. Preto je časová zložitosť nášho programu lineárna od počtu vrcholov (na rozdiel od klasického prehľadávania do šírky, ktoré závisí od počtu hrán).

Program vyzerá nasledovne:

```
var x: 0..1;                    { bol vrchol označený vo vstupe? }
    y: 0..1 = 0;                              { je označený teraz? }
    i: 1..3;
begin
   if x=1 then y := 1;          { prenesieme označenie zo vstupu }
   for i := 1 to 3 do          { pozrieme sa na všetkých susedov }
      if S[i].y <> 0 then            { ak je i-ty sused označený }
         y := 1;                                { označ aj sám seba }
end.
```

Príklad 2

Majme 2-graf zložený z jedného cyklu párnej dĺžky, teda z vrcholov očíslovaných $0 \ldots N-1$, pričom vrchol i je spojený hranou označenou 1 s vrcholom $(i+1) \bmod N$ a hranou označenou 2 s vrcholom $(i-1) \bmod N$. (Príklad takého grafu pre $N = 6$ nájdete na obrázku na začiatku tohto textu.) V tomto grafe je vyznačený jeden vrchol v, rovnako ako v predchádzajúcom príklade. Napíšte

program pre grafomat, ktorý označí vrchol protiľahlý k v, teda vrchol s číslom $(v + N/2) \bmod N$.

Riešenie: Vyšleme signál putujúci z vrcholu v v smere jednotkových hrán rýchlosťou 1 vrchol za takt. Zároveň vyšleme druhý signál putujúci rovnakou rýchlosťou opačným smerom. Akonáhle nejaký vrchol zistí, že do neho prišli oba signály, označí sa a signály už ďalej neposiela.

```
var x: 0..1;                                  { vstupná značka vrcholu }
    y: 0..1 = 0;                                      { výstupná značka }
    l, r: 0..1 = 0;           { už týmto vrcholom prešiel signál
begin                                                 doľava a doprava? }
   if x=1 then begin
      x := 0; l := 1; r := 1;                    { začíname posielať }
   end else if (S[2].l=1) and (S[1].r=1) then
      y := 1                                     { signály sa stretli }
   else if (S[2].l=1) and (l=0) then
      l := 1                                   { prišiel signál zľava }
   else if (S[1].r=1) and (r=0) then
      r := 1;                                  { prišiel signál sprava }
end.                  { inak sa v tomto vrchole teraz nič nedeje }
```

Zjavne výpočet tohto grafomatu skončí presne po $(N/2) + 2$ taktoch. Po $(N/2) + 1$ taktoch budeme v situácii, kedy:

- začiatočný vrchol bude mať $x = 0$, $l = r = 1$,
- $N - 1$ vrcholov naľavo od neho bude mať $l = 1$,
- $N - 1$ vrcholov napravo bude mať $r = 1$,
- no a vrchol oproti bude mať $l = r = 0$ a $y = 1$.

V nasledujúcom takte sa už žiadna premenná v žiadnom vrchole nezmení, a teda výpočet končí.

SLOVENSKÁ KOMISIA OLYMPIÁDY V INFORMATIKE

DVADSIATY DRUHÝ ROČNÍK OLYMPIÁDY V INFORMATIKE

Vydala IUVENTA s finančnou podporou Ministerstva školstva SR
Náklad: 40 výtlačkov
Zodpovedný redaktor: Michal Forišek
Sadzba programom LaTeX