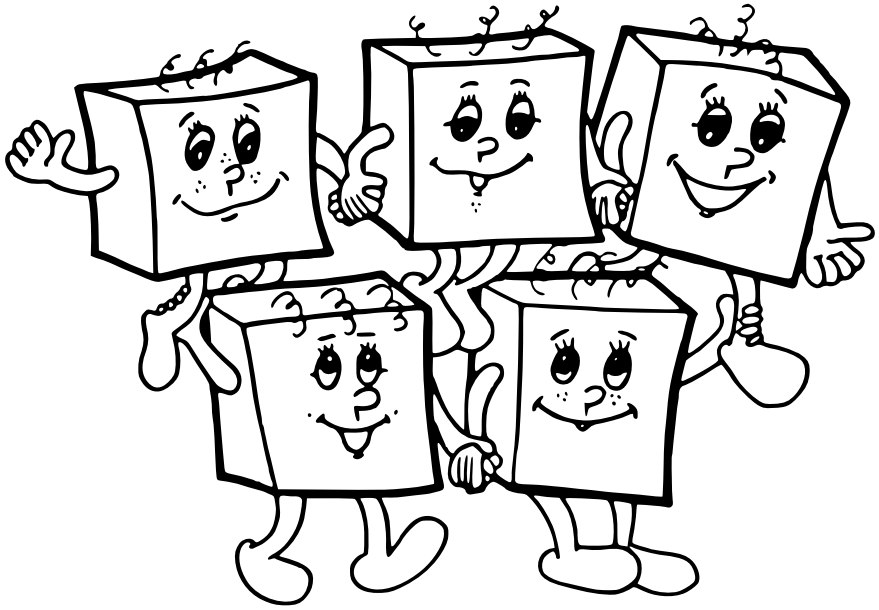
OLYMPIÁDA V INFORMATIKE

NA STREDNÝCH ŠKOLÁCH



dvadsiaty tretí ročník

školský rok 2007/08

zadania krajského kola
kategória B

- **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/`.

Priebeh krajského kola

Krajské kolo 23. ročníka Olympiády v informatike, kategórie B, sa koná 15. januára 2007 v dopoludňajších hodinách. Na riešenie úloh majú súťažiaci 4 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 zápisu algoritmu (do 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

Za každú úlohu môžete získať od 0 do 10 bodov.

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.

Efektívnosť algoritmu posudzujeme vypočítaním jeho časovej zložitosti – funkcie, ktorá hovorí, ako dlho vykonanie algoritmu trvá v závislosti od veľkosti vstupných parametrov.

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, zrieši vzorové riešenie ľubovoľný povolený vstup do niekoľko sekúnd.

Zadania kategórie B

B-II-1 Zberateľky

Danka a Janka sú dvojičky. Obe zbierajú cudzokrajné známky. Ešte donedávna mali obe presne rovnaké zbierky, každá tú svoju pekne spôsobne utriedenú vo svojom albume.

Lenže včera cestou zo školy našla Janka na zemi obálku s dvoma nádhernými známkami, ktoré ešte vo zbierke nemali. Danke ich však odmietla ukázať, a doma si ich rýchlo založila do svojho albumu.

Danku teraz zožiera zvedavosť – akéže to známky Janka má a ona nie? A tak, keď Janka odišla vyniesť smeti, Danka schytila oba albumy a začala nové známky hľadať.

Súťažná úloha

Dané je celé číslo N a dve polia celých čísel $A[1 \ldots N]$ a $B[1 \ldots N+2]$. Polia A aj B sú utriedené v rastúcom poradí. Pole B obsahuje presne to isté ako pole A, a navyše dva prvky x, y (zaradené na správnom mieste). Všetky čísla v poli B (a teda aj v poli A) sú navzájom rôzne.

Napíšte procedúru/funkciu, ktorá prvky x a y čo najrýchlejšie nájde.

(Polia už existujú v pamäti pri zavolaní vašej procedúry, teda čas potrebný na ich načítanie nerátame do časovej zložitosti procedúry.)

Formát vstupu

Vaša procedúra dostane ako parametre číslo N a polia A a B. (Prípadne, ak je vám to pohodlnejšie, považujte N, A a B za globálne premenné.)

Formát výstupu

Vaša procedúra má vrátiť dve celé čísla – hodnoty prvkov x a y, ktoré sú v poli B navyše.

Príklady

vstup	výstup
$N = 5$ $A = (1, 3, 4, 7, 110)$ $B = (1, 2, 3, 4, 7, 47, 110)$	`2 47`

vstup	výstup
$N = 4$ $A = (2, 3, 7, 110)$ $B = (2, 3, 7, 110, 111, 123)$	`111 123`

Hodnotenie riešení

Celých 10 bodov môžu dostať len riešenia, ktoré majú lepšiu ako lineárnu časovú zložitosť, teda nájdu riešenie bez toho, aby videli väčšinu prvkov v poliach A a B.
Maximálne 6 bodov dostanú riešenia, ktoré vyriešia v lepšom ako lineárnom čase ľahšiu verziu úlohy: stačí, aby fungovali, ak čísla x a y nasledujú v poli B bezprostredne za sebou.
Maximálne 5 bodov dostanú riešenia (pôvodnej úlohy) s lineárnou časovou zložitosťou, teda také, ktorých čas behu je priamo úmerný číslu N.
Maximálne 3 body dostanú pomalšie riešenia.

B-II-2 Krtkovou norou tam a späť

Kdesi v okolí Brna žijú prúdoví krtkovia. Žijú pod zemou, majú tam svoje brlôžky a medzi nimi vyrazené chodbičky. Chodbičky sú rôzne dlhé, priamo úmerne tomu, koľko dní danú chodbičku krtkovia razili. A že sú krtkovia prúdoví, chodbičky razia rýchlo, a teda im to nikdy dlho netrvá.

Krtek Vítek býva v brlôžku 1. V brlôžku N býva jeho kamarátka, ku ktorej chodí na návštevu obzvlášť rád. A keďže sa k nej veľmi teší, ide k nej vždy

najkratšou možnou cestou. To však neznamená, že ide vždy presne rovnakou cestou – najkratších ciest môže byť viac a Vítek si vždy zvolí, ktorou z nich pôjde.

Súťažná úloha

Napíšte program, ktorý načíta popis siete brlôžkov, a nájde všetky brlôžky, ktoré môže Vítek cestou ku svojej kamarátke navštíviť.

Formát vstupu a výstupu

V prvom riadku vstupu sú dve celé čísla N a M ($1 \leq N \leq 10\,000$, $0 \leq M \leq 100\,000$), kde N je počet brlôžkov a M počet chodbičiek medzi nimi. Brlôžky sú očíslované od 1 do N. Krtek Vítek býva v brlôžku číslo 1, jeho kamarátka v brlôžku N.

Nasleduje M riadkov. V každom z nich sú tri kladné celé čísla a, b a d, kde a a b ($1 \leq a, b \leq N$) sú čísla brlôžkov spojených chodbičkou a d ($1 \leq d \leq 7$) udáva, koľko dní krtkovia chodbičku razili.

Na výstup vypíšte jeden riadok a v ňom medzerami oddelené čísla všetkých brlôžkov, ktoré ležia na niektorej najkratšej ceste z 1 do N.

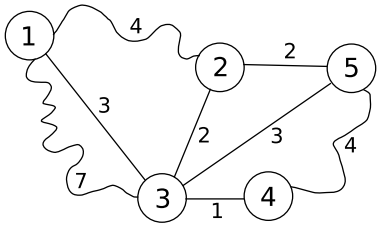
Príklad

vstup

```
5 8
1 2 4
1 3 7
1 3 3
2 3 2
2 5 2
3 4 1
3 5 3
4 5 4
```

výstup

```
1 2 3 5
```



Najkratšie cesty sú dve: 1-2-5 a 1-3-5.

Hodnotenie riešení

Plný počet bodov môžu získať riešenia, ktorých časová zložitosť závisí lineárne od veľkosti vstupu (od hodnoty $M + N$).
Riešenia, ktoré spravia počet krokov úmerný N^2, získajú najviac 8 bodov.
Riešenia, ktorých časová zložitosť rastie exponenciálne v závislosti od N, získajú najviac 4 body.

B-II-3 Opravovanie XML

Pripomeňme si najskôr, ako vyzerajú korektné XML dokumenty. Dokument sa skladá z tagov a textu medzi nimi. Za každým otváracím tagom musí nasledovať jemu zodpovedajúci zatvárací, pričom musí byť dodržané vnáranie tagov do seba. (Inými slovami, pre ľubovoľný otvárací tag musí platiť, že keď zoberieme všetky tagy a text za ním, až po jemu zodpovedajúci zatvárací tag, dostaneme opäť korektný XML dokument.)

Príklad XML dokumentu:

```
<kviz>
  toto je text len tak
  <zaznam>
    <otazka>Kolky rocnik OI prave prebieha?</otazka>
    <odpoved>23</odpoved>
  </zaznam>
  <zaznam><otazka>2+2?</otazka><odpoved>4</odpoved></zaznam>
</kviz>
```

Všimnite si, že tagy sú v sebe vnorené – keď napríklad vo vnútri tagu `<zaznam>` otvoríme tag `<otazka>`, musíme najskôr použiť zatvárací tag `</otazka>`, až potom môžeme použiť tag `</zaznam>`.

V domácom kole sme si napísali program, ktorý o danom texte vedel povedať, či je to korektný XML dokument alebo nie. V praxi však často potrebujeme viac. Príkladom sú rôzne prehliadače www stránok. Mnohé www stránky sú napísané

nekorektne, prehliadač sa však napriek tomu snaží aj takúto stránku vám čo najlepšie zobraziť.

V tejto úlohe sa budeme zaoberať jednoduchým spôsobom opravy chýb, ktoré v XML dokumente objavíme. Ak vstupný dokument nie je korektný XML dokument, napravíme to tak, že niektoré tagy z neho vyškrtáme.

Tagy, ktoré vyškrtať, určíme nasledujúcim spôsobom. Tagy budeme spracúvať v poradí, v akom sa vyskytujú v dokumente. V každom okamihu si budeme pamätať postupnosť práve otvorených tagov.

Otvárací tag spracujeme vždy tak, že ho pridáme na koniec pamätanej postupnosti.

U zatváracieho tagu sú tri možnosti:

1. Pasuje k poslednému práve otvorenému tagu.
   V takomto prípade máme nájdený pár a jednoducho dotyčný otvárací tag vyhodíme z pamätanej postupnosti.
2. Pasuje k niektorému skoršiemu otvorenému tagu.
   V takomto prípade postupne (od konca) vyškrtávame otvorené otváracie tagy, až kým nedostaneme situáciu z bodu 1.
3. Nepasuje k žiadnemu otvorenému tagu.
   V takomto prípade vyškrtneme tento zatvárací tag.

Po spracovaní celého dokumentu ešte vyškrtáme tie otváracie tagy, ktoré ostali otvorené.

Súťažná úloha

Napíšte program, ktorý čo najefektívnejšie odsimuluje tento algoritmus a vypíše počet vyškrtnutých tagov.

Formát vstupu a výstupu

Vstup obsahuje niekoľko (nanajvýš 10 000) riadkov textu, každý z nich obsahuje najviac 100 znakov.

Môžete predpokladať, že znaky < a > sa vyskytujú len ako začiatky a konce tagov, a že názov každého tagu je tvorený 1 až 8 malými písmenami.

Na výstup vypíšte jedno celé číslo – počet vyškrtnutých tagov.

Príklady

vstup

```
<srnka><paroh>hnedy</paroh>
</srnka><jelen></jelen>
```

výstup

```
0
```

Korektný XML dokument.

vstup

```
<z> <a> <a> <b> </a>
<v> </v> </a>
```

výstup

```
2
```

Pri spracúvaní prvého `</a>` *vyšktrneme* `<b>`*. Po spracovaní celého dokumentu vyškrtneme* `<z>`*.*

vstup

```
<ahoj> <ako> </ako> </sa>
<mas>   </mas> </ahoj>
```

výstup

```
1
```

Tag `</sa>` *vyšktrneme, lebo nemá čo zavrieť.*

B-II-4 Rákosie sa vracia

V tomto ročníku olympiády v teoretickej úlohe pestujeme zvláštne druhy rákosia. V študijnom texte uvedenom za zadaním tejto úlohy je popísané, ako toto rákosie vyzerá. Študijný text je zhodný so študijným textom z domáceho kola.

Súťažná úloha

a) *(4 body)* Nájdite nejakú odrodu rákosia, ktorej mŕtve steblá obsahujú len bunky a, b a c, pričom buniek c je toľko isto ako buniek a a b dokopy. Samozrejme, chceme takú odrodu, kde môže narásť každé takéto steblo.

Vo vašej odrode rákosia teda musí existovať spôsob, ako narastú napr. steblá ε, *bcbcbc*, *cabc* a $a^{47}b^3c^{50}$, a naopak nesmú narásť napr. steblá a, *baca* ani *dedo*.

b) *(6 bodov)* Nájdite nejakú odrodu rákosia, ktorej mŕtve steblá sú tvorené len bunkami a, a navyše počet týchto buniek je ľubovoľné zložené číslo.

Vo vašej odrode teda musia vedieť narásť napr. steblá a^4, a^6, a^8, a^9, ... Naopak, nesmú narásť napr. steblá ε, a, aa, aaa, a^5, a^{47} ani *dedo*.

Možno dobrá rada: Každé zložené číslo vieme zapísať v tvare $m \cdot n$, kde $m, n > 1$.

Študijný text

Programátorské rákosie je zvláštna rastlina. Každý exemplár sa skladá z niekoľkých buniek, ktoré rastú v rade za sebou. Každá bunka je buď živá, alebo mŕtva. Živé bunky budeme značiť veľkými písmenami abecedy, mŕtve bunky malými. Keďže bunky rastú v rade, každé steblo rákosia vieme jednoducho popísať tak, že vypíšeme zaradom (od koreňa) typy jeho buniek.

Každé steblo rákosia sa počas svojho života vyvíja, až do okamihu, keď už obsahuje samé mŕtve bunky. Počas tohto vývoja sa môže meniť počet aj druh buniek, ktoré obsahuje.

Existuje veľa rôznych odrôd rákosia. Každá odroda má v génoch „naprogramovanú“ svoju sadu pravidiel, podľa ktorých sa steblá danej odrody môžu meniť.

Mŕtve bunky sú pekné farebné. Pestovatelia preto často šľachtia rôzne odrody, ktoré by vytvárali z mŕtvych buniek rôzne zaujímavé vzory.

Formálnejšie definície

Steblo rákosia je ľubovoľná (aj prázdna) postupnosť veľkých a malých písmen, predstavujúcich živé a mŕtve bunky. Ak steblo neobsahuje žiadne živé bunky, hovoríme, že je to mŕtve steblo.

Aby sme prázdnu postupnosť buniek videli, budeme ju značiť ε. Ak ide v zápise stebla k rovnakých písmen x za sebou, môžeme to skrátene zapísať x^k.

Teda napríklad $baCDGk^3v^{47}XX$ je steblo rákosia, ktoré je tvorené 57 bunkami, z nich je 5 živých.

Všimnite si, že steblo vypisujeme vždy od koreňa nahor. Teda steblo *abc* je iné ako steblo *cba*.

Genetické pravidlo je pravidlo, ktoré určuje jednu možnú zmenu stebla. Každé pravidlo je tvaru: „ak sa na steble vyskytne postupnosť buniek *vstup*, môže sa zmeniť na postupnosť buniek *výstup*“. Postupnosti *vstup* aj *výstup* môžu obsahovať ľubovoľne veľa (aj nula) živých aj mŕtvych buniek. Jediné obmedzenie je, že *vstup* musí obsahovať aspoň jednu živú bunku.

Pravidlo zapisujeme *vstup* $\rightarrow$ *výstup*. Ak máme viac pravidiel, ktoré majú rovnaký vstup, môžeme ich skrátene zapísať ako
vstup $\rightarrow$ *výstup*$_1$ | *výstup*$_2$ | *výstup*$_3$.

Napríklad zo stebla $aBacaBd$ sa použitím pravidla $aB \rightarrow X$ môže stať buď steblo $XacaBd$ alebo steblo $aBacXd$.

Použitie pravidla budeme značiť $\Rightarrow$. (Všimnite si, že ide o dvojitú šípku, zatiaľ čo na zápis pravidla používame jednoduchú.) Teda napríklad druhú zmenu z predchádzajúceho príkladu by sme zapísali $aBacaBd \Rightarrow aBacXd$.

Keď vznikne nové steblo rákosia, je vždy tvorené práve jednou bunkou Z. **Odroda rákosia** je jednoznačne určená *konečnou* sadou genetických pravidiel. Tá jednoznačne popisuje, aké všetky druhy stebiel môžu zo začiatočnej bunky Z narásť.

Súbor vzorov, ktoré vie odroda rákosia vyrobiť, je množina *všetkých mŕtvych* stebiel, ktoré môžu v rákosí danej odrody narásť.

Príklad 1

Pestovateľ Ferko chcel, aby mu na rybníku vyrástlo rákosie, ktoré bude mať pekné strakaté steblá. Presnejšie, chcel, aby súbor vzorov jeho rákosia tvorili práve *všetky* steblá, na ktorých sa striedajú bunky *a* a *b*.

Ferko teda chce odrodu, v ktorej môžu narásť mŕtve steblá *ababa*, *abab* aj *babab*, ale nie *aab* ani *klmn*.

Riešenie príkladu 1

Existuje veľa rôznych odrôd, ktoré majú túto vlastnosť. Ferko vyšľachtil odrodu, ktorá mala nasledujúce genetické pravidlá:

$$Z \to X, \quad Z \to bX, \quad X \to abX, \quad X \to \varepsilon \quad \text{a} \quad X \to a.$$

Ukážeme, ako môže narásť niekoľko dobrých stebiel. Ku každému použitiu pravidla $\Rightarrow$ napíšeme pre názornosť poradové číslo pravidla, ktoré sme použili. Teda napríklad $\Rightarrow_3$ znamená, že bolo použité pravidlo $X \to abX$.

$$\begin{array}{llllllll} Z & \Rightarrow_1 & X & \Rightarrow_3 & abX & \Rightarrow_3 & ababX & \Rightarrow_3 & ababa \\ Z & \Rightarrow_1 & X & \Rightarrow_3 & abX & \Rightarrow_3 & ababX & \Rightarrow_4 & abab \\ Z & \Rightarrow_2 & bX & \Rightarrow_3 & babX & \Rightarrow_3 & bababX & \Rightarrow_4 & babab \end{array}$$

Všimnite si, že použitie pravidla 4 zmaže zo stebla bunku X.

Príklad 2

Pestovatelia Jožko a Mirko chceli, aby ich rákosie „vyrábalo“ mŕtve steblá, ktoré budú mať rovnako veľa buniek a aj b (a žiadne iné), ale na rozdiel od Ferka nech tieto bunky môžu byť ľubovoľne rozmiestnené.

Chcú teda odrodu, kde môžu narásť steblá ab, $abba$ aj $ba^{47}b^{46}$, ale nie $ababa$ ani bac.

Riešenie príkladu 2

Jožko vyšľachtil rákosie s pravidlami:

$$Z \to aZbZ \mid bZaZ \mid \varepsilon.$$

Mirko vyšľachtil rákosie s pravidlami:

$$Z \to ABZ \mid \varepsilon, \quad AB \to BA, \quad BA \to AB, \quad A \to a, \quad B \to b.$$

Hlavný rozdiel medzi týmito dvoma odrodami je v tom, že Mirkovi sa bude ľahšie presviedčať návštevy, že jeho rákosie naozaj „funguje“.

V prípade Mirkovho rákosia je to naozaj jednoduché. Použitím prvého pravidla k-krát dostaneme steblo, na ktorom je k živých buniek A a k živých buniek B. Použitím pravidiel 2 a 3 sa tieto bunky môžu ľubovoľne premiešať a keď následne použijeme pravidlá 4 a 5, vyhrali sme.

U Jožkovho rákosia je zjavné, že počty a a b budú rovnaké, ale vieme naozaj vyrobiť všetky možné dobré steblá? Matematickou indukciou podľa dĺžky stebla dokážeme, že áno.

Steblá dĺžky 0 (ε) aj 2 (ab a ba) vyrobiť vieme. Nech teraz vieme vyrobiť všetky dobré steblá dĺžky $\leq 2n$. Zoberme si ľubovoľné dobré steblo dĺžky $2n+2$. Bez ujmy na všeobecnosti nech začína bunkou a. Budeme postupne čítať steblo a počítať si, o koľko viac a ako b sme videli. Na začiatku je táto hodnota 1 (začíname bunkou a), na konci bude 0 (lebo a a b v celom steble je rovnako).

Všimnime si okamih, kedy prvýkrát klesne táto hodnota na 0. Muselo sa to stať tak, že sme práve prečítali nejaké b. Napríklad pre steblo $aabaabbabbbaab$ sa tak stane po prečítaní desiateho znaku, pre $aabb$ po štvrtom.

Teraz naše slovo môžeme schematicky zapísať ako $a\beta b\gamma$, kde b je to b, ktoré sme si práve našli. Vieme, že v časti $a\beta b$ je rovnako veľa a aj b. Preto musí byť rovnako veľa a aj b v časti β, a teda aj v časti γ. Podľa indukčného predpokladu vieme teda zo Z vyrobiť aj β, aj γ. No a celé naše steblo teda vyrobíme tak, že najskôr použijeme pravidlo $Z \to aZbZ$, potom z prvého Z vyrobíme β, a nakoniec z druhého Z vyrobíme γ.

SLOVENSKÁ KOMISIA OLYMPIÁDY V INFORMATIKE

DVADSIATY TRETÍ ROČNÍK OLYMPIÁDY V INFORMATIKE

Zodpovedný redaktor: Michal Forišek
Sadzba programom LaTeX