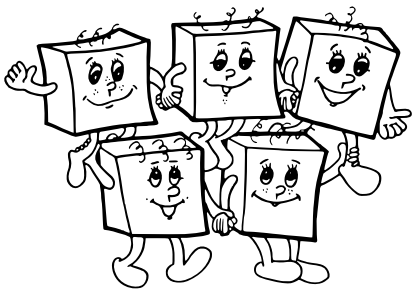
OLYMPIÁDA V INFORMATIKE

NA STREDNÝCH ŠKOLÁCH

http://oi.sk/



dvadsiaty deviaty ročník
školský rok 2013/2014

zadania celoštátneho kola, deň 2
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 druhého, praktického dňa majú súťažiaci 4,5 hodiny čistého času. Akékoľvek pomôcky okrem písacích potrieb (napr. knihy, výpisy programov, kalkulačky) sú zakázané.

Čo má obsahovať riešenie úlohy?

- Skompilovateľný program v jazyku Pascal, C, alebo C++. Ak sa váš program nepodarí na našom testovacom počítači skompilovať, bude automaticky hodnotený 0 bodmi.

Hodnotenie riešení druhého (praktického) dňa

Za každú úlohu môžete získať od 0 do 10 bodov.

Po ukončení súťaže zoberieme pre každú úlohu váš posledný odovzdaný program a ten automaticky otestujeme na vopred pripravených testovacích vstupoch.

Testovanie na každom vstupe prebieha samostatne. Spustíme váš program a na štandardný vstup mu dáme konkrétne vstupné údaje. Hovoríme, že váš program daný vstup vyriešil, ak splní nasledujúce kritériá:

- Skončí skôr ako uplynie stanovený časový limit.
- Neprekročí stanovený pamäťový limit.
- Skončí korektne, nie chybou počas behu.
- Dáta, ktoré vypíše na štandardný výstup, tvoria korektný výstup, zodpovedajúci danému vstupu.
- Nebude používať žiadne funkcie zakázané kvôli bezpečnosti testovacieho systému.

Ku každej úlohe máme pripravených 10 sád testovacích vstupov. Sada vstupov pozostáva z jedného alebo viacerých testovacích vstupov. Za každú sadu vstupov, ktorej všetky vstupy (každý zvlášť) váš program správne vyrieši, získate jeden bod.

Sady vstupov sú navrhované tak, aby každé korektné riešenie získalo nejaké body, bez ohľadu na to, ako pomalé je. Bližšie informácie o testovacích dátach nájdete na konci zadania každej úlohy.

A-III-4 Hore a dole

Baška mala postupnosť n celých čísel, každé z rozsahu od 0 po m. Pre každé dva po sebe idúce členy postupnosti si zapísala znak `<`, `=`, alebo `>` podľa toho, či bol skorší z nich menší, rovnaký, alebo väčší ako neskorší. Napr. pre postupnosť $(3, 1, 7, 7, 4)$ platí $3 > 1 < 7 = 7 > 4$, takže Baška by zapísala reťazec `><=>`.

Olívia Baškinu postupnosť nepozná. Pozná len n, m a postupnosť znakov, ktoré si Baška zapísala. Olívia by teraz chcela Baškinu postupnosť (v rámci možností) zrekonštruovať.

Časť bodov za túto úlohu môžete získať tak, že napíšete program, ktorý zrekonštruuje ľubovoľnú takú postupnosť (ak nejaká existuje). Plný počet bodov dostanete za program, ktorý navyše vždy nájde to riešenie, v ktorom je *súčet členov Baškinej postupnosti najmenší možný*.

Formát vstupu a výstupu

Vstup má dva riadky. V prvom z nich sú čísla n a m. V druhom z nich je reťazec $n - 1$ znakov, ktoré si Baška zapísala. Na výstup vypíšte jeden riadok a v ňom n medzerami oddelených celých čísel: zrekonštruovanú postupnosť. Ak žiadna postupnosť zodpovedajúca vstupu neexistuje, vypíšte n-krát hodnotu -1.

Obmedzenia a hodnotenie

Pripravených je 10 sád vstupov, očíslovaných od 01 do 10. Za správne vyriešenie každej sady získate 1 bod. V sadách 01 až 05 akceptujeme ľubovoľnú korektnú postupnosť, v sadách 06 až 10 len postupnosť s najmenším možným súčtom členov.

Veľkosti a typ testovacích dát sú popísané v nasledujúcej tabuľke:

číslo sady	obmedzenia		
01, 06	$2 \leq n \leq 10$,	$m = 10^9$,	niektorý jeden znak je `>` a všetky ostatné sú `<`
02, 07	$2 \leq n \leq 20$,	$m = 10^9$	
03, 08	$2 \leq n \leq 1000$,	$m = 998$	
04, 09	$2 \leq n \leq 75\,000$,	$0 \leq m \leq 10^9$,	v reťazci nie je znak `=`
05, 10	$2 \leq n \leq 250\,000$,	$0 \leq m \leq 10^9$	

Príklady

vstup

```
5 1000000000
><=>
```

výstup

```
1 0 1 1 0
```

Tento vstup by mohol patriť do sady 02 alebo sady 07. Ak by patril do sady 02, akceptovali by sme aj odpoveď „3 1 7 7 4“, ale v sade 07 je jedinou správnou odpoveďou práve „1 0 1 1 0“.

vstup

```
4 2
>>>
```

výstup

```
-1 -1 -1 -1
```

Reťazec `>>>` popisuje štvorprvkovú klesajúcu postupnosť. Keďže ale $m = 2$, my môžeme použiť len hodnoty 0, 1 a 2, a teda takúto postupnosť vyrobiť nevieme.

A-III-5 Vyvážené reťazce

Reťazec je *vyvážený* vtedy, keď sa v ňom všetky jeho písmená vyskytujú rovnako veľa ráz. Príklady vyvážených reťazcov: `aaaaa`, `badcx`, `bbaaab`. Reťazec `abacb` vyvážený nie je – napríklad preto, že sú v ňom dve `a` ale len jedno `c`.

V danom reťazci nájdite najdlhší súvislý podreťazec, ktorý je vyvážený.

Hľadaný podreťazec nemusí obsahovať všetky druhy písmen, ktoré sa vyskytujú v pôvodnom reťazci. Teda napr. pre reťazec `cbababac` je správnym riešením podreťazec `bababa`.

Formát vstupu a výstupu

Vstup má jediný riadok a v ňom reťazec tvorený malými písmenami anglickej abecedy. Dĺžku reťazca si označíme n, znaky reťazca očíslujeme zľava doprava od 0 po $n-1$.

Na výstup vypíšte jeden riadok a v ňom dve celé čísla: index znaku, ktorým hľadaný podreťazec začína, a index znaku, ktorým tento podreťazec končí. Ak existuje viacero optimálnych riešení, nájdite to, ktoré začína najskôr.

Obmedzenia a hodnotenie

Pripravených je 10 sád vstupov, očíslovaných od 01 do 10. Za správne vyriešenie každej sady získate 1 bod. Veľkosti a typ testovacích dát sú popísané v nasledujúcej tabuľke:

číslo sady	obmedzenia
01	$1 \leq n \leq 20$, reťazec obsahuje len písmená `ab`
02	$1 \leq n \leq 1000$, reťazec obsahuje len písmená `ab`
03	$1 \leq n \leq 1000$, reťazec obsahuje len písmená `abcdefgh`
04	$1 \leq n \leq 1000$, reťazec obsahuje len písmená `abcdefgh`
05	$1 \leq n \leq 100\,000$, reťazec obsahuje len písmená `ab`
06	$1 \leq n \leq 100\,000$, reťazec obsahuje len písmená `abc`, v opt. riešení nie sú použité všetky tri
07	$1 \leq n \leq 100\,000$, reťazec obsahuje len písmená `abc`
08	$1 \leq n \leq 50\,000$, reťazec obsahuje len písmená `abcde`
09	$1 \leq n \leq 50\,000$ reťazec obsahuje len písmená `abcde`
10	$1 \leq n \leq 50\,000$ reťazec obsahuje len písmená `abcde`

Príklady

vstup

```
baab
```

výstup

```
0 3
```

Optimálnym podreťazcom je celý reťazec.

vstup

```
baaab
```

výstup

```
1 3
```

Optimálnym podreťazcom je reťazec `aaa`.

vstup

```
cbababac
```

výstup

```
1 6
```

Príklad zo zadania.

vstup

```
cbabadbabae
```

výstup

```
1 4
```

Skorší výskyt je ten správny.

A-III-6 ACGT

Usámec sa snaží z fragmentov DNA skladať genóm. Fragment DNA je postupnosť znakov `ACGT`. Usámec už našiel n fragmentov a označil ich $F_1, \ldots, F_n$. Navyše zistil, ktoré dvojice fragmentov môžu nasledovať bezprostredne za sebou.

Poskladanie DNA. Poskladaním DNA nazývame postupnosť fragmentov $a_1, a_2, \ldots, a_k$ ($1 \leq a_i \leq n$, niektoré fragmenty sa môžu aj opakovať) takú, že každé dva po sebe idúce fragmenty po sebe smú nasledovať. Poskladaniu DNA zodpovedá reťazec znakov `ACGT`, ktorý dostaneme pospájaním príslušných fragmentov.

Príklad: Nech $n = 3$, $F_1 =$ `CG`, $F_2 =$ `AT` a $F_3 =$ `TCC`. Po sebe nech môžu nasledovať dvojice fragmentov (1,2), (1,3) a (2,1). Korektným poskladaním DNA pre tento prípad je napr. postupnosť 2,1 (ktorej zodpovedá reťazec `ATCG`) a tiež postupnosť 1,2,1,3 (reťazec `CGATCGTCC`). Postupnosť 3,1 ani postupnosť 1,1,2 nie sú korektné.

Vzdialenosť reťazcov. Pre dané dva reťazce p a q definujeme ich vzdialenosť ako najmenší počet zmien, ktoré treba postupne spraviť, aby sme z p vyrobili q (resp. naopak). Povolené zmeny sú troch typov: pridanie znaku (kamkoľvek do reťazca), zmazanie znaku a zmena jedného znaku na iný.

Príklad: Nech $p =$ `ATTA` a $q =$ `AGA`. Zjavne nevieme p na q prerobiť jednou zmenou, ide to však dvomi: najskôr zmažeme jedno `T` a potom zmeníme druhé `T` na `G`. Preto majú tieto dva reťazce vzdialenosť 2. Ľubovoľný reťazec má sám od seba vzdialenosť 0.

Súťažná úloha

Na vstupe je popis Usámcovych fragmentov a toho ako môžu na seba nadväzovať. Ďalej je na vstupe reťazec R (opäť tvorený znakmi `ACGT`), *veľmi malé* nezáporné celé číslo d a dva indexy fragmentov u a v (pričom $u \neq v$). Vašou úlohou je nájsť (jedno ľubovoľné) poskladanie DNA, ktoré začína fragmentom u, končí fragmentom v, a reťazec, ktorý mu zodpovedá, má od reťazca r vzdialenosť nanajvýš d.

Môžete predpokladať, že žiaden fragment nesmie nasledovať sám po sebe – teda že v zozname prípustných dvojíc nebudú záznamy tvaru (x, x). Tiež môžete predpokladať, že ak po nejakom fragmente x môže nasledovať aj fragment y aj fragment z, tak sa prvé písmená F_y a F_z líšia.

Formát vstupu a výstupu

V prvom riadku vstupu je počet fragmentov n a počet prípustných dvojíc m. V nasledujúcich n riadkoch sú jednotlivé fragmenty F_1 až F_n. V ďalších m riadkoch sú dvojice čísel a_i, b_i, ktoré vyjadrujú, že po fragmente a_i môže nasledovať fragment b_i. Predposledný riadok obsahuje čísla d, u a v; posledný riadok obsahuje reťazec R.

Na výstup vypíšte jeden riadok obsahujúci -1, pokiaľ žiadne vhodné poskladanie DNA neexistuje. Ináč vypíšte jeden riadok, ktorý obsahuje jedno vhodné poskladanie DNA – teda postupnosť indexov fragmentov oddelených medzerami. (Nie je potrebné mať najmenšiu možnú vzdialenosť od R.)

Hodnotenie

Pripravených je 10 sád vstupov, očíslovaných od 01 do 10. Na sadách 01 až 08 bude vaše riešenie otestované na testovači. Vstupy 09 a 10 si stiahnite z testovača, spustite na nich váš program a odovzdajte iba výstup.

Za správne vyriešenie každej sady získate 1 bod. V nasledujúcej tabuľke f označuje súčet dĺžok fragmentov DNA a r dĺžku reťazca R. Pokiaľ nie je uvedené inak, tak v sadách 01 až 08 platí $n \leq 1000$, $m \leq 4000$, $f \leq 50\,000$, $r \leq 50\,000$ a $d \leq 5$. V sadách 09 a 10 platí $n \leq 1000$, $m \leq 1000$, $f \leq 1\,000\,000$, $r \leq 1\,000\,000$ a $d \leq 5$. Navyše tieto vstupy sú z reálnych dát, čo napríklad znamená, že medzi fragmentmi na vstupe nenájdete veľa podobných. Dodatočné obmedzenia pre jednotlivé sady sú nasledovné:

číslo sady	obmedzenia	komentár
01	$d = 0$	fragmenty musia naskladať presne R
02, 03	$n = 2$, $m = 1$, $f \leq 1000$, $r \leq 1000$	len dva fragmenty a jeden spôsob ich zreťazenia
04, 05	$n = 2$, $m = 1$	
06	$d = 1$	smieme mať nanajvýš jednu chybu
07, 08	$f \leq 1000$, $r \leq 1000$	

Príklady

vstup

```
3 3
CG
AT
TCC
1 2
2 1
1 3
0 1 3
CGATCGTCC
```

výstup

```
1 2 1 3
```

Toto je príklad zo zadania, hľadaný reťazec vieme naskladať presne.

vstup

```
4 4
CCC
A
T
GGG
1 2
2 3
3 2
2 4
1 1 4
CCCATTGGG
```

výstup

```
1 2 3 2 4
```

Toto poskladanie nám dá reťazec `CCCATAGGG`*, ktorý sa od reťazca na vstupe líši jednou zmenou.*

vstup

```
4 4
CCC
A
T
GGG
1 2
2 3
3 2
2 4
1 1 4
CCCAAAAGGG
```

výstup

```
-1
```

Treba aspoň dve zmeny, povolenú však máme nanajvýš jednu.

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