63. ročník Matematické olympiády – 2013/2014

Úlohy ústředního kola kategorie P – 2. soutěžní den

Na řešení úloh máte 4,5 hodiny čistého času. Při soutěži je zakázáno používat jakékoliv pomůcky kromě psacích potřeb a přiděleného počítače (tzn. knihy, kalkulačky, mobily, apod.).

Řešením každého příkladu je zdrojový kód programu zapsaný v programovacím jazyce Pascal, C nebo C++. Po skončení soutěže bude pro každou úlohu váš poslední odevzdaný program automaticky otestován pomocí předem připravených 10 sad testovacích vstupních dat. Každá sada je tvořena jedním nebo několika testovacími vstupy. Jeden bod získáte za každou sadu vstupů, kterou váš program celou správně vyřeší – tzn. pro každý testovací vstup z této sady dá program správný výsledek ve stanoveném časovém a paměťovém limitu. Za každou úlohu tedy můžete získat 0 až 10 bodů.

Sady vstupů jsou navrženy tak, aby každé korektní řešení získalo nějaké body, i když je založeno na neefektivním algoritmu. Časové a paměťové limity zajišťují, že plný počet bodů získá pouze efektivní řešení. Podrobnější informace o testovacích datech najdete na konci zadání každé úlohy.

P-III-4 Nahoru a dolů

Program:	`nahorudolu.pas` / `nahorudolu.c` / `nahorudolu.cpp`
Vstup a výstup:	standardní vstup a výstup
Časový limit:	řádově sekundy
Paměťový limit:	1 GB

Jeníček měl posloupnost n celých čísel, každé z rozsahu od 0 do m. Pro každé dva po sobě jdoucí členy posloupnosti si zapsal znak `<`, `=`, nebo `>` podle toho, zda byl dřívější člen posloupnosti menší, stejný, nebo větší než pozdější člen posloupnosti. Například pro posloupnost $(3, 1, 7, 7, 4)$ platí $3 > 1 < 7 = 7 > 4$, takže Jeníček by si zapsal řetězec `><=>`.

Mařenka nezná Jeníčkovu posloupnost čísel. Zná pouze n, m a posloupnost znaků, které si Jeníček zapsal. Chtěla by ale původní posloupnost čísel (v rámci možností) zrekonstruovat.

Část bodů za tuto úlohu můžete získat tím, že napíšete program, který najde libovolnou vyhovující posloupnost čísel (pokud vůbec nějaká existuje). Plný počet bodů dostanete za program, který navíc vždy najde takové řešení, v němž je součet členů posloupnosti nejmenší možný.

Popis vstupu a výstupu

Vstup je tvořen dvěma řádky. Na prvním z nich jsou dvě celá čísla n a m. Na druhém řádku je řetězec $n - 1$ znaků, které si Jeníček zapsal. Na výstup vypište

jeden řádek a na něm n mezerami oddělených celých čísel – zrekonstruovanou posloupnost čísel. Pokud žádná posloupnost odpovídající vstupním údajům neexistuje, vypište n-krát hodnotu -1.

Omezení a hodnocení

Je připraveno 10 testovacích sad vstupních dat, očíslovaných od 01 do 10. Za správné vyřešení každé sady získáte 1 bod. V sadách 01 až 05 akceptujeme libovolnou korektní posloupnost, v sadách 06 až 10 pouze posloupnost s nejmenším možným součtem členů. Velikosti a typ jednotlivých sad popisuje následující tabulka:

sada	*omezení*		
01, 06	$2 \leq n \leq 10$	$m = 10^9$	jeden znak je `>` a všechny ostatní `<`
02, 07	$2 \leq n \leq 20$	$m = 10^9$	
03, 08	$2 \leq n \leq 1\,000$	$m = 998$	
04, 09	$2 \leq n \leq 75\,000$	$0 \leq m \leq 10^9$	v řetězci není znak `=`
05, 10	$2 \leq n \leq 250\,000$	$0 \leq m \leq 10^9$	

Příklady

Vstup:	*Výstup:*
`5 1000000000` `><=>`	`1 0 1 1 0`

Tento vstup by mimo jiné mohl patřit do sady 02 nebo do sady 07. Pokud by patřil do sady 02, akceptovali bychom třeba i odpověď „3 1 7 7 4“, ale v sadě 07 je jedinou správnou odpovědí právě „1 0 1 1 0“.

Vstup:	*Výstup:*
`4 2` `>>>`	`-1 -1 -1 -1`

Řetězec `>>>` *popisuje čtyřprvkovou klesající posloupnost. Protože ale $m = 2$, můžeme použít jenom hodnoty 0, 1 a 2, takže takovou posloupnost nelze vytvořit.*

P-III-5 Vyvážené řetězce

Program:	`vyvazene.pas` / `vyvazene.c` / `vyvazene.cpp`
Vstup a výstup:	standardní vstup a výstup
Časový limit:	cca 0,5 s
Paměťový limit:	1 GB

Řetězec nazveme *vyváženým* tehdy, když v něm mají všechna jeho písmena stejný počet výskytů. Vyvážené řetězce jsou například: `aaaaa`, `badcx`, `bbaaab`. Řetězec `abacb` vyvážený není – například proto, že obsahuje dvě `a`, ale jen jedno `c`.

V zadaném řetězci najděte nejdelší souvislý podřetězec, který je vyvážený.

Hledaný podřetězec nemusí obsahovat všechny druhy písmen, která se vyskytují v původním řetězci. Například pro řetězec `cbababac` je správným řešením podřetězec `bababa`.

Popis vstupu a výstupu

Vstupem programu je jediný řádek, který obsahuje řetězec tvořený malými písmeny anglické abecedy. Délku řetězce označíme n, znaky řetězce očíslujeme zleva doprava od 0 do $n - 1$.

Na výstup vypište jeden řádek a na něm dvě celá čísla: index znaku, kterým hledaný podřetězec začíná, a index znaku, kterým tento podřetězec končí. Pokud existuje více různých optimálních řešení, najděte a vypište to z nich, které v řetězci začíná co nejdříve.

Omezení a hodnocení

Je připraveno 10 testovacích sad vstupních dat, očíslovaných od 01 do 10. Za správné vyřešení každé sady získáte 1 bod. Velikosti a typ jednotlivých testovacích dat popisuje následující tabulka:

sada	*omezení*	
01	$1 \leq n \leq 20$	řetězec obsahuje jen písmena `ab`
02	$1 \leq n \leq 1\,000$	řetězec obsahuje jen písmena `ab`
03	$1 \leq n \leq 1\,000$	řetězec obsahuje jen písmena `abcdefgh`
04	$1 \leq n \leq 1\,000$	řetězec obsahuje jen písmena `abcdefgh`
05	$1 \leq n \leq 100\,000$	řetězec obsahuje jen písmena `ab`
06	$1 \leq n \leq 100\,000$	řetězec obsahuje jen písmena `abc`, v žádném optimálním řešení nejsou použita všechna tři
07	$1 \leq n \leq 100\,000$	řetězec obsahuje jen písmena `abc`
08	$1 \leq n \leq 50\,000$	řetězec obsahuje jen písmena `abcde`
09	$1 \leq n \leq 50\,000$	řetězec obsahuje jen písmena `abcde`
10	$1 \leq n \leq 50\,000$	řetězec obsahuje jen písmena `abcde`

Příklady

Vstup:	*Výstup:*
`baab`	`0 3`

Optimálním podřetězcem je celý řetězec.

Vstup:	*Výstup:*
`baaab`	`1 3`

Optimálním podřetězcem je řetězec `aaa`*.*

Vstup:	*Výstup:*
`cbababac`	`1 6`

Příklad ze zadání.

Vstup:	*Výstup:*
`cbabadbabae`	`1 4`

Dřívější výskyt je ten správný.

P-III-6 ACGT

Program:	`acgt.pas` / `acgt.c` / `acgt.cpp`
Vstup a výstup:	standardní vstup a výstup
Časový limit:	řádově sekundy, u sad 09 a 10 zvýšeno na 15 s
Paměťový limit:	1 GB

Sestavení DNA

Fragment DNA je posloupnost znaků z abecedy `A`, `C`, `G`, `T`. Získali jsme n fragmentů a označili je $F_1, \ldots, F_n$. Zjistili jsme také, které dvojice fragmentů mohou následovat bezprostředně po sobě.

Sestavením DNA nazveme takovou posloupnost čísel fragmentů $a_1, a_2, \ldots, a_k$ ($1 \leq a_i \leq n$, některá čísla se mohou v posloupnosti opakovat), že každé dva po sobě jdoucí fragmenty mohou v daném pořadí po sobě následovat. Každému sestavení DNA přímo odpovídá řetězec znaků `ACGT`, který dostaneme spojením příslušných fragmentů.

Příklad: Nechť $n = 3$, $F_1 =$ `CG`, $F_2 =$ `AT` a $F_3 =$ `TCC`. Po sobě mohu následovat dvojice fragmentů $(1, 2)$, $(1, 3)$ a $(2, 1)$. Korektním sestavením DNA pro tento případ je například posloupnost $2, 1$ (které odpovídá řetězec `ATCG`) a také posloupnost $1, 2, 1, 3$ (řetězec `CGATCGTCC`). Posloupnost $3, 1$ ani posloupnost $1, 1, 2$ nejsou korektní.

Vzdálenost řetězců

Pro dané dva řetězce P a Q definujeme jejich vzdálenost jako nejmenší počet změn, které je třeba provést, abychom z P získali Q (resp. naopak). Povolené změny jsou tří typů: přidání jednoho znaku (na libovolné místo v řetězci), smazání jednoho znaku (kteréhokoliv) a záměna jednoho znaku za jiný.

Příklad: Nechť $P =$ `ATTA` a $Q =$ `AGA`. Zjevně nedokážeme převést P na Q jedinou změnou, jde to však provést dvěma změnami: nejprve smažeme jedno `T` a potom změníme druhé `T` na `G`. Proto mají tyto dva řetězce vzdálenost 2. Libovolný řetězec má sám od sebe vzdálenost 0.

Soutěžní úloha

Na vstupu dostanete popis fragmentů DNA a informaci, jak po sobě mohou tyto fragmenty následovat. Dále je na vstupu zadán řetězec R (opět tvořený znaky `ACGT`), *velmi malé* nezáporné celé číslo d a dva indexy fragmentů u a v (přičemž $u \neq v$). Vaším úkolem je nalézt (jedno libovolné) sestavení DNA, které začíná fragmentem F_u, končí fragmentem F_v, a řetězec, který mu odpovídá, má od řetězce R vzdálenost nejvýše d.

Můžete předpokládat, že žádný fragment nesmí následovat sám po sobě – tedy že v seznamu přípustných dvojic nebudou záznamy tvaru (x, x). Můžete také předpokládat, že mohou-li po nějakém fragmentu F_x následovat fragment F_y i fragment F_z, tak se liší první písmena F_y a F_z.

Popis vstupu a výstupu

Na prvním řádku vstupu je zadán počet fragmentů n a počet přípustných dvojic m. Na následujících n řádcích jsou uvedeny jednotlivé fragmenty F_1 až F_n. Na dalších m řádcích jsou dvojice čísel a_i, b_i, které vyjadřují, že po fragmentu a_i může následovat fragment b_i. Předposlední řádek vstupu obsahuje čísla d, u a v. Na posledním řádku vstupu je řetězec R.

Na výstup vypište jeden řádek obsahující číslo -1, pokud žádné vhodné sestavení DNA neexistuje. Jinak vypište jeden řádek, který obsahuje jedno vhodné sestavení DNA – tedy posloupnost indexů fragmentů oddělených mezerami. (Není nutné dosáhnout nejmenší možné vzdálenosti od R.)

Omezení a hodnocení

Je připraveno 10 testovacích sad vstupních dat, očíslovaných od 01 do 10. Sady 09 a 10 jsou veřejné a můžete si je stáhnout ze soutěžního prostředí.

Za správné vyřešení každé sady získáte 1 bod. V následující tabulce f označuje součet délek fragmentů DNA a r délku řetězce R. Není-li uvedeno jinak, v sadách 01 až 08 platí $n \leq 1\,000$, $m \leq 4\,000$, $f \leq 50\,000$, $r \leq 50\,000$, $d \leq 5$. V sadách 09 a 10 platí $n \leq 1\,000$, $m \leq 1\,000$, $f \leq 1\,000\,000$, $r \leq 1\,000\,000$, $d \leq 5$. Navíc tyto vstupy obsahují reálná data, což například znamená, že mezi fragmenty na vstupu nenajdete mnoho podobných.

Dodatečná omezení pro jednotlivé sady jsou následující:

sada	*omezení*	*komentář*
01	$d = 0$	fragmenty musí sestavit přesně R
02, 03	$n = 2$, $m = 1$, $f \leq 1000$, $r \leq 1000$	jen dva fragmenty a jeden způsob jejich zřetězení
04, 05	$n = 2$, $m = 1$	
06	$d = 1$	povolena nejvýše jedna chyba
07, 08	$f \leq 1000$, $r \leq 1000$	
09, 10	bez dalších omezení	

Příklady

Vstup:	*Výstup:*
3 3 CG AT TCC 1 2 2 1 1 3 0 1 3 CGATCGTCC	1 2 1 3

Toto je příklad ze zadání, hledaný řetězec dokážeme sestavit přesně.

Vstup:	*Výstup:*
4 4 CCC A T GGG 1 2 2 3 3 2 2 4 1 1 4 CCCATTGGG	1 2 3 2 4

Toto sestavení nám dá řetězec `CCCATAGGG`*, který se od řetězce na vstupu liší jednou změnou.*

Vstup:	*Výstup:*
4 4 CCC A T GGG 1 2 2 3 3 2 2 4 1 1 4 CCCAAAAGGG	-1

Byly by zapotřebí aspoň dvě změny, povolenou ale máme nejvýše jednu.