64. ročník Matematické olympiády – 2014/2015

Úlohy ústředního kola kategorie P – 1. soutěžní den

Na řešení úloh máte 4,5 hodiny čistého času. Řešení každé úlohy pište na samostatný list papíru. Při soutěži je zakázáno používat jakékoliv pomůcky kromě psacích potřeb (tzn. knihy, kalkulačky, mobily, apod.).

Řešení každého příkladu musí obsahovat:

- **Popis řešení,** to znamená slovní popis principu zvoleného algoritmu, argumenty zdůvodňující jeho správnost, diskusi o efektivitě vašeho řešení (časová a paměťová složitost). Slovní popis řešení musí být jasný a srozumitelný i bez nahlédnutí do samotného zápisu algoritmu (do programu). Není povoleno odkazovat se na Vaše řešení předchozích kol, opravovatelé je nemají k dispozici; na autorská řešení se odkazovat můžete.
- **Zápis algoritmu.** V úlohách **P-III-1** a **P-III-2** je třeba uvést zápis algoritmu, a to buď ve tvaru zdrojového textu nejdůležitějších částí programu v jazyce Pascal nebo C/C++, nebo v nějakém dostatečně srozumitelném pseudokódu. Nemusíte detailně popisovat jednoduché operace jako vstupy, výstupy, implementaci jednoduchých matematických vztahů, vyhledávání v poli, třídění apod. V úloze **P-III-3** je potřeba popsat požadovanou magickou síť a dokázat, že splňuje zadané vlastnosti; pokud požadovaná síť neexistuje, je nutné její neexistenci zdůvodnit.

Za každou úlohu můžete získat maximálně 10 bodů. Hodnotí se nejen správnost řešení, ale také kvalita jeho popisu (včetně zdůvodnění správnosti) a efektivita zvoleného algoritmu. Algoritmy posuzujeme zejména podle jejich časové složitosti, tzn. podle závislosti doby výpočtu na velikosti vstupních dat. Záleží přitom pouze na řádové rychlosti růstu této funkce.

P-III-1 Oříšková čokoláda

Štaflík a Špagetka s Bobem a Bobkem se záhadně dostali ze svých Večerníčků. Všichni se sešli u veliké oříškové čokolády a domluvili se, že si zahrají následující hru.

Štaflík a Špagetka budou hrát proti Bobovi a Bobkovi, oba týmy se budou pravidelně střídat v tazích. Během hry čokoládou neotáčejí a oba týmy se na ni dívají ze stejné strany. Tým, který je na tahu, si vybere buď celý poslední řádek (nejvíce dole), nebo celý poslední sloupec (nejvíce napravo). Vybraný řádek, resp. sloupec z čokolády odlomí a sní. Aby se mohli v týmu spravedlivě rozdělit, musí čokoláda odlomená v každém tahu celkem obsahovat sudý počet oříšků. Tým, který nemůže táhnout, prohrává a musí se okamžitě vrátit do své pohádky. Vítězný tým může spokojeně spořádat zbytek čokolády.

Zjistěte, která dvojice vyhraje, pokud budou oba týmy hrát optimálně. Pozor, postavičkám nezáleží na množství zkonzumované čokolády ani oříšků.

Soutěžní úloha

Obdélníková čokoláda má R řádků a S sloupců ($1 \leq R, S \leq 1\,000$). Políčka čokolády indexujeme dvojicemi přirozených čísel (i, j), kde $1 \leq i \leq R$, $1 \leq j \leq S$. Políčko $(1, 1)$ značí levý horní roh čokolády. Pro každé políčko je dán počet oříšků $B_{i,j}$, který se v něm nachází. Navíc platí, že $0 \leq B_{i,j} \leq 10^6$. První jsou na tahu Bob a Bobek.

Formát vstupu

První řádek vstupu obsahuje dvě mezerou oddělená celá čísla R a S. Následuje R řádků, na každém z nich se nachází S nezáporných celých čísel oddělených mezerami; j-té číslo na i-tém řádku vyjadřuje počet oříšků $B_{i,j}$ na políčku (i, j).

Formát výstupu

Na výstup vypište jediný řádek obsahující jedno písmeno. Pokud hru vyhrají Bob a Bobek, vypište písmeno `B`. Vyhrají-li Štaflík a Špagetka, vypište písmeno `S`.

Příklady

Vstup:	*Výstup:*

`3 4`
`0 1 2 3`
`4 5 6 7`
`8 9 10 11`
	`B`

Bob s Bobkem odlomí poslední řádek. Štaflík se Špagetkou poté mohou vzít buď poslední řádek, nebo poslední sloupec. Kdyby ovšem vzali řádek, Bob s Bobkem vezmou také řádek (poslední zbývající) a vyhrají. Po prvním tahu se tedy Štaflík se Špagetkou pokusí vzít poslední sloupec (políčka obsahující 3 a 7 oříšků). Poté už ale budou moci oba týmy až do posledního tahu ulamovat pouze sloupce a ten poslední si vezme Bob s Bobkem, čímž vyhrají.

Vstup:	*Výstup:*
2 2 3 6 4 7	S

Bob s Bobkem nemohou již na začátku nijak táhnout.

P-III-2 Ztracené pakety

Při přenosu velkých souborů po síti je nutné rozdělit je na menší kusy (pakety), které jsou doručovány samostatně. Při doručování není zaručeno zachování pořadí paketů. Aby je přijímající počítač dokázal správně spojit, pakety jsou očíslovány od 1 do n dle pořadí, v jakém po sobě následují v souboru.

Na přijímající počítač zatím dorazilo $n - k$ z n odeslaných paketů, k se jich tedy ztratilo. Abychom mohli požádat odesílající počítač o jejich znovuzaslání, potřebujeme určit čísla těchto ztracených paketů. Přijaté pakety jsme dosud neměli čas nijak zpracovat, máme je pouze uloženy na disku v pořadí, v jakém dorazily. Paketů je mnoho, nemůžeme si proto všechna jejich čísla uložit do paměti (oproti tomu k je poměrně malé). Může proto být nutné používat pomocné soubory. Čtení dat z disku je pomalé a chceme ho minimalizovat.

Formát vstupu

Na prvním řádku vstupu jsou uvedena dvě přirozená čísla n ($1 \leq n \leq 10^9$) a k ($1 \leq k \leq 100$). Na každém z $n - k$ následujících řádků je jedno přirozené číslo v rozsahu 1 až n včetně, udávající číslo paketu. Každé číslo paketu se v souboru vyskytuje nejvýše jednou, pořadí čísel paketů ve vstupním souboru může být libovolné.

Formát výstupu

Výstupem je k navzájem různých přirozených čísel v rozsahu 1 až n včetně, která se nevyskytují mezi čísly paketů ve vstupním souboru. Tato čísla můžete vypsat v libovolném pořadí.

Příklad

Vstup:	*Výstup:*

`6 2`
`2`
`1`
`6`
`5`
	`3 4`

Bodování

Přijatelná jsou pouze řešení, jejichž paměťová složitost závisí pouze na k, nikoliv na n, a která používají dohromady nejvýše $2k + 2$ pomocných souborů.

Dále bude brán ohled na celkový počet čísel zapsaných či čtených ze souborů (vstupního i pomocných). Plných 10 bodů získá algoritmus, který úlohu vyřeší pro zadaná omezení na n a k s použitím pomocné paměti nejvýše 10 kB a s méně než $6 \cdot 10^9$ čísly zapsanými a čtenými ze souborů. Až 6 bodů může získat libovolný algoritmus, schopný úlohu pro zadaná omezení na n a k vyřešit s použitím pomocné paměti nejvýše 10 kB a s méně než 10^{11} čísly zapsanými a čtenými ze souborů.

P-III-3 Magická síť

K této úloze se vztahuje studijní text uvedený na následujících stranách. Studijní text je identický se studijním textem z domácího a krajského kola. V zadání úloh se vyskytují následující omezení:

- ONE-IN-THREE(x, y, z) předepisuje, že právě jedna z proměnných x, y a z má hodnotu 1.
- CH(x, y, z) předepisuje, že alespoň jedna z proměnných x a y má hodnotu 1 a alespoň jedna z proměnných y a z má hodnotu 0.

Úkol 1: *(3 body)* Nalezněte síť používající pouze omezení typu ONE-IN-THREE, která simuluje CH.

Úkol 2: *(3 body)* Ukažte, že pomocí CH nelze simulovat ONE-IN-THREE.

Úkol 3: *(4 body)* Ukažte, že pro libovolné omezení $O(x_1, \ldots, x_n)$ existuje síť používající pouze omezení typu ONE-IN-THREE, která simuluje O.

Studijní text

Magická síť se skládá z *omezení* a *proměnných*. Každá proměnná může nabývat hodnot 0 nebo 1. Omezení pak předepisují podmínky, které ohodnocení proměnných musí splňovat. Například omezení XOR(x, y, z) předepisuje, že z proměnných x, y a z jich lichý počet musí mít hodnotu 1, omezení OR(x, y, z) předepisuje, že alespoň jedna z x, y a z musí mít hodnotu 1, omezení EQ(x, y) předepisuje, že x a y musí mít stejnou hodnotu, a podobně. Proměnné se v jednom omezení mohou opakovat.

Některé z proměnných jsou *vstupní* a můžeme jim nastavit konkrétní hodnotu. Po seslání příslušného zaklínadla se pak ostatním proměnným nastaví takové hodnoty, aby všechna omezení byla splněna. Pokud žádná taková volba hodnot neexistuje, zaklínadlo nás na to upozorní. V prvním případě říkáme, že magická síť zadaný vstup *přijímá*, ve druhém ho *odmítá*. Magická síť *simuluje omezení* O, jestliže přijímá právě stejné hodnoty proměnných jako omezení O.

Příklad 1: Magickou síť zapisujme jako seznam typů omezení, k nimž do závorek budeme připisovat proměnné, na které jsou aplikovány. Síť XOR(a, b, c), XOR(b, c, d) tedy vynucuje, že lichý počet z proměnných a, b a c má hodnotu 1 a že lichý počet z proměnných b, c a d má hodnotu 1.

Nechť a a d jsou vstupní proměnné této sítě. Jestliže a má hodnotu 0, pak právě jedna z proměnných b a c musí mít hodnotu 1, a proto d musí mít hodnotu 0. Naopak, má-li a hodnotu 1, pak hodnota proměnné b musí být stejná jako hodnota proměnné c, a proto d musí mít hodnotu 1.

Tato magická síť tedy přijímá právě ty vstupy, kde a a d mají stejnou hodnotu, a simuluje tedy omezení EQ(a, d).

Obecněji: Typicky nás bude zajímat, která omezení jdou vyjádřit pomocí jiných. Říkáme, že množina typů omezení $\{O_1, O_2, \ldots, O_n\}$ *simuluje* omezení O, jestliže existuje magická síť používající pouze omezení typu O_1, O_2, …, O_n, která simuluje O. Příklad 1 tedy ukazuje, že XOR simuluje EQ.

Omezení $O(x_1, \ldots, x_n)$ nazveme *slabé*, jestliže zakazuje právě jednu kombinaci hodnot proměnných $x_1, \ldots, x_n$. Slabé omezení budeme zapisovat jako $S_{h_1 h_2 \ldots h_n}$, kde $h_1, \ldots, h_n$ jsou hodnoty proměnných, které zakazuje. Třeba omezení $S_{001}(x, y, z)$ je splněno, jestliže $x = 1$ nebo $y = 1$ nebo $z = 0$, a omezení S_{000} je stejné jako OR. Nechť SAT_n označuje množinu všech slabých omezení s právě n proměnnými.

Příklad 2: SAT_3 simuluje XOR, jelikož síť

$$S_{000}(x, y, z), S_{011}(x, y, z), S_{101}(x, y, z), S_{110}(x, y, z)$$

zakazuje všechny kombinace hodnot proměnných x, y a z, v nichž se hodnota 1 vyskytuje suděkrát.

Příklad 3: Množina omezení SAT_2 nesimuluje OR. Abychom si to dokázali, zaveďme si nejprve funkci maj se třemi vstupy. Ta bude vracet ten vstup, který se vyskytuje nejčastěji (proto se jí také někdy říká *majorita*). Tedy třeba $maj(1, 1, 1) = maj(1, 0, 1) = 1$ a $maj(0, 0, 1) = 0$.

Uvažujme nyní libovolnou síť s omezeními z množiny SAT_2. Nechť $x_1, \ldots, x_n$ jsou proměnné této sítě. Řekněme, že by tato síť simulovala $\text{OR}(x_1, x_2, x_3)$. Jelikož omezení OR je splněno pro hodnoty 1, 0 a 0, existuje nějaké přiřazení hodnot proměnným, které splňuje všechna omezení, x_1 má hodnotu 1 a x_2 a x_3 mají hodnoty 0. Nechť a_i označuje hodnotu proměnné x_i v tomto přiřazení, pro $i = 1, \ldots, n$. Obdobně existuje přiřazení s hodnotami b_i splňující všechna omezení takové, že $b_2 = 1$ a $b_1 = b_3 = 0$, a přiřazení s hodnotami c_i splňující všechna omezení takové, že $c_3 = 1$ a $c_1 = c_2 = 0$.

Uvažme ohodnocení s hodnotami $d_i = maj(a_i, b_i, c_i)$. Tvrdíme, že d_i také splňuje všechna omezení: Mějme nějaké omezení $S_{h_1 h_2}(x_i, x_j)$ ze sítě. Pokud $a_i = b_i$ a $a_j = b_j$, pak $d_i = maj(a_i, a_i, c_i) = a_i$ a $d_j = maj(a_j, a_j, c_j) = a_j$ splňuje podmínku $S_{h_1 h_2}$, protože ji splňuje a_i a a_j. Proto předpokládejme, že $a_i \neq b_i$ nebo $a_j \neq b_j$, a obdobně $a_i \neq c_i$ nebo $a_j \neq c_j$ a stejně tak $b_i \neq c_i$ nebo $b_j \neq c_j$. Ze symetrie mezi i a j a mezi ohodnoceními a, b a c stačí uvažovat případ, že $a_i \neq b_i$ a $a_i \neq c_i$. Z toho odvodíme, že $b_i = c_i$, a proto $b_j \neq c_j$. Díky symetrii mezi b a c pak stačí uvažovat případ, že $a_j = b_j$ a $a_j \neq c_j$. Pak ale $\text{maj}(a_i, b_i, c_i) = maj(a_i, b_i, b_i) = b_i$ a $\text{maj}(a_j, b_j, c_j) = maj(b_j, b_j, c_j) = b_j$, a proto $d_i = b_i$ a $d_j = b_j$ splňuje podmínku $S_{h_1 h_2}$.

Povšimněme si, že $d_1 = maj(1, 0, 0) = 0$ a obdobně $d_2 = d_3 = 0$. Ale omezení $\text{OR}(x_1, x_2, x_3)$ předepisuje, že alespoň jedna z proměnných x_1, x_2 a x_3 má hodnotu 1, a proto v každé magické síti simulující $\text{OR}(x_1, x_2, x_3)$ musí přiřazení hodnot d_i proměnným porušovat nějaké omezení. Uvažovaná síť tedy nesimuluje $\text{OR}(x_1, x_2, x_3)$.