# 57. ročník matematické olympiády na středních školách

Kategorie P

In: Karel Horák (editor); Daniel Kráľ (editor); Martin Mareš (editor); Peter Novotný (editor); Jaromír Šimša (editor); Jaroslav Švrček (editor); Pavel Töpfer (editor): 57. ročník matematické olympiády na středních školách. Zpráva o řešení úloh ze soutěže konané ve školním roce 2007/2008. 49. mezinárodní matematická olympiáda. 20. mezinárodní olympiáda v informatice. (Czech). Praha: Jednota českých matematiků a fyziků, 2010. pp. 93–113.

Persistent URL: http://dml.cz/dmlcz/405152

## P – I – 1

### O zdánlivém kopci

Franta a Pepík se chystali na výlet. Vymysleli si trasu, kudy spolu půjdou, potom každý vytáhnul svoji mapu a začal si v ní trasu prohlížet. Některé body, kterými trasa vedla, měly v mapách vyznačenu nadmořskou výšku.

„To je zajímavé," řekl po chvíli Franta. „Když se dívám jen na nadmořské výšky bodů, kterými budeme procházet, vypadá to, že půjdeme jenom přes jeden kopec — nejprve nahoru a potom dolů."

„Ale kdepak, to není pravda," podivil se Pepík.

Za chvíli zjistili, kde vzniknul problém: měli různě přesné mapy. Některé nadmořské výšky bodů, které byly na Pepíkově mapě vyznačeny, na Frantově mapě chyběly.

**Soutěžní úloha.** Napište program, který dostane na vstupu seznam nadmořských výšek z Pepíkovy mapy a zjistí, kolik nejvýše z nich může být vyznačeno i na Frantově mapě.

*Formát vstupu*: První řádek vstupu obsahuje jedno celé číslo $N$ ($1 \leqq N \leqq 100\,000$) — počet těch bodů na trase výletu, které mají na Pepíkově mapě vyznačenou nadmořskou výšku. Další řádky obsahují celkem $N$ celých čísel z rozsahu 1 až 1 000 000 000 — nadmořské výšky těchto bodů (v nějakých vám neznámých jednotkách). Výšky jsou uvedeny v pořadí, jak po sobě následují na trase výletu.

*Formát výstupu*: Výstupem bude jediné celé číslo $K$ — největší počet nadmořských výšek z Pepíkovy mapy, které mohly být vyznačeny i na Frantově mapě.

Jestliže $a_1, \ldots, a_K$ jsou výšky vyznačené na Frantově mapě, pak musí splňovat následující podmínku: pro nějaké $i$ z množiny $\{1, \ldots, K\}$ musí platit $\forall j \in \{1, \ldots, i-1\}$: $a_j < a_{j+1}$ a zároveň $\forall j \in \{i, \ldots, K-1\}$: $a_j > a_{j+1}$.

*Příklad:*

| *vstup:* | *výstup:* |
|---|---|
| 12<br>112 247 211 209 244<br>350 470 510 312 215<br>117 217 | 9<br><br>*Jedna možnost, jak mohly vypadat nadmořské výšky na Frantově mapě:*<br>112, 211, 244, 350, 470, 510, 312, 215, 117 |

## P – I – 2

### Rezervace místenek

„V horách se našlo zlato!“ šeptali si lidé v širokém okolí už několik týdnů. Ti s dobrodružnější povahou si už balili věci a mířili přímo do hor, do legendami opředené oblasti zvané Horní Klondík.

Když ale Horní Klondík zaplavila vlna nových zlatokopů, nastal nečekaný problém. Jediný místní vláček, který v Klondíku měli, byl najednou tak přecpaný, že se do něj polovina zájemců ani nevešla. A nedivte se, že zlatokopa, který se žene za co nejvýhodnější parcelou, něco takového pořádně rozzlobí.

Když už se zdálo, že zanedlouho dojde na každé železniční stanici ke krveprolití, dostali železničáři spásonosný nápad. Budou na vláček prodávat místenky.

**Soutěžní úloha.** Napište program, který bude zpracovávat rezervace míst na jednu jízdu vlaku. Trať vlaku má $N+1$ stanic, které si očíslujeme od 0 do $N$ v pořadí, jak na trati leží. Ve vlaku je $M$ míst, takže mezi každou dvojicí po sobě následujících stanic může jet vlakem nejvýše $M$ zlatokopů.

Váš program musí postupně zpracovat několik požadavků na rezervaci míst. Každý požadavek je tvaru „$x$ lidí chce jet ze stanice $y$ do stanice $z$.“ Pokud je ještě na každém úseku trati mezi stanicemi $y$ a $z$ ve vlaku aspoň $x$ míst volných, váš program takovýto požadavek přijme, v opačném případě ho odmítne.

*Formát vstupu*: První řádek vstupu obsahuje tři celá čísla $N$, $M$ a $P$ ($1 \leqq N, M, P \leqq 100\,000$) — počet úseků trati, počet míst ve vlaku a počet požadavků na rezervaci.

Následuje $P$ řádků, každý z nich popisuje jeden požadavek na rezervaci v pořadí, v jakém byly tyto požadavky zadány. Přesněji, $i$-tý z těchto řádků obsahuje tři celá čísla $x_i$, $y_i$, $z_i$ oddělená mezerami ($1 \leqq x_i \leqq M$, $0 \leqq y_i < z_i \leqq N$). Význam těchto hodnot byl popsán výše.

*Formát výstupu*: Pro každý požadavek vypište na výstup jeden řádek a na něm buď řetězec `prijat`, jestliže příslušný požadavek bylo ještě možné splnit, nebo řetězec `odmitnut`, pokud už ve vlaku nebylo dost volných míst.

*Příklad*:

| *vstup:* | *výstup:* |
|---|---|
| 4 6 6 | `prijat` |
| 2 1 4 | `prijat` |
| 2 1 3 | `odmitnut` |
| 3 2 4 | `odmitnut` |
| 3 1 2 | `prijat` |
| 6 0 1 | `prijat` |
| 4 3 4 | |

Po přijetí prvních dvou požadavků víme, že v úsecích mezi stanicemi 1 a 2 a mezi stanicemi 2 a 3 zůstanou už jen dvě volná místa. Proto musíme odmítnout následující dvě trojčlenné skupiny, které chtějí cestovat i na těchto úsecích tratě. Poslední dva požadavky opět můžeme splnit. U prvního z nich je to zřejmé, u druhého nám ve stanici 3 vystoupí dostatek lidí na to, aby se na poslední úsek trati uvolnila přesně potřebná čtyři místa.

## P – I – 3

### Fibonacciho soustava

Fibonacciho posloupnost vypadá následovně:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$$

Sestrojíme ji tak, že prvními dvěma členy posloupnosti jsou 0 a 1 a každý následující člen posloupnosti je součtem dvou předcházejících. Matematicky zapsáno:

$$\begin{aligned} F_0 &= 0, \\ F_1 &= 1, \\ F_{n+2} &= F_{n+1} + F_n \qquad (\forall n \geqq 0). \end{aligned}$$

Podívejme se nyní na to, jak fungují poziční číselné soustavy. Poziční číselná soustava je určena dvěma údaji: množinou používaných cifer a posloupností, která pro každou pozici v zápisu čísla udává hodnotu, jíž se příslušná cifra násobí.

Například naše desítková soustava používá množinu cifer $\{0, 1, 2, \ldots, 9\}$ a jednotlivé pozice v čísle mají hodnoty (zprava doleva) 1, 10, 100, 1 000, ...

Fibonacciho číselná soustava je poziční číselná soustava, která používá pouze cifry 0 a 1 a v níž jsou hodnoty jednotlivých pozic postupně rovny členům Fibonacciho posloupnosti (počínaje číslem $F_2 = 1$). Tedy například zápis 10011 ve Fibonacciho soustavě představuje číslo $1 \cdot 8 + 0 \cdot 5 + 0 \cdot 3 + 1 \cdot 2 + 1 \cdot 1 = 11$.

Na rozdíl od běžných číselných soustav ve Fibonacciho soustavě nemají některá čísla jednoznačný zápis. Například také zápis 10100 představuje číslo 11.

**Soutěžní úloha.**

a) *(3 body)* Zápis čísla ve Fibonacciho soustavě nazveme *pěkný*, jestliže se v něm nikde nevyskytují dvě po sobě jdoucí jedničky. Dokažte, že každé přirozené číslo má ve Fibonacciho soustavě právě jeden pěkný zápis. Napište program, který ze vstupu přečte přirozené číslo $N$ a vypíše jeho pěkný zápis.

b) *(7 bodů)* Napište program, který pro daná $k$, $A$ a $B$ spočítá, kolik čísel z množiny $\{A, A + 1, \ldots, B\}$ má ve svém pěkném zápisu právě $k$ jedniček.

*Formát vstupu*: V části a) je na vstupu jediné přirozené číslo $N$ ($1 \leqq N \leqq 1\,000\,000\,000$).

V části b) jsou na vstupu tři přirozená čísla $k$, $A$ a $B$ ($1 \leqq k \leqq 30$, $1 \leqq A < B \leqq 1\,000\,000\,000$).

*Formát výstupu*: V části a) vypište na jeden řádek řetězec nul a jedniček, který představuje pěkný zápis čísla $N$.

V části b) vypište jedno celé číslo — počet čísel ze zadaného intervalu, která mají ve svém pěkném zápisu právě $k$ jedniček.

*Příklady pro část a)*:

| *vstup 1:* | *výstup 1:* |
|---|---|
| 11 | 10100 |
| *vstup 2:* | *výstup 2:* |
| 174591 | 10100010000000000100010010 |

*Příklady pro část b)*:

| *vstup 1:* | *výstup 1:* |
|---|---|
| 1 4 13 | 3 |

(Pro $k = 1$ je výstupem počet Fibonacciho čísel v daném rozsahu. V zadaném rozsahu leží Fibonacciho čísla 5, 8 a 13.)

| *vstup 2:* | *výstup 2:* |
|---|---|
| 2 4 13 | 6 |
| *vstup 3:* | *výstup 3:* |
| 10 102000 103000 | 86 |

## P – I – 4

### Překládací stroje

V tomto ročníku olympiády budeme pracovat s překládacími stroji. Ve studijním textu uvedeném za zadáním úlohy jsou tyto stroje popsány.

**Soutěžní úloha.**

a) *(1 bod)* Všimněte si překládacích strojů $B$ a $C$ z příkladů ve studijním textu. Tyto dva stroje zjevně provádějí překlad „opačnými směry". Mohli bychom proto očekávat, že platí následující tvrzení:
Nechť $M$ je libovolná (třeba i nekonečná) množina, jejímiž prvky jsou nějaké řetězce písmen $a$, $e$ a $i$. Potom $C(B(M)) = M$. Slovně popsáno: Když vezmeme množinu $M$, přeložíme ji pomocí stroje $B$ a výsledek přeložíme pomocí $C$, dostaneme původní množinu.
Pokud toto tvrzení skutečně platí, dokažte ho. Pokud ne, najděte protipříklad.

b) *(1 bod)* Totéž jako v předcházející části, jen $M$ obsahuje řetězce tvořené znaky ▪ a • a zajímá nás, zda musí platit $B(C(M)) = M$.

c) *(3 body)* Řekneme, že řetězec je zajímavý, jestliže obsahuje pouze písmena $a$ a $b$, přičemž písmen $a$ je dvojnásobný počet než písmen $b$. Nechť $X$ je množina všech zajímavých řetězců. Tedy například $aaabab \in X$, ale $baba \notin X$.
Nechť $Y$ je množina všech řetězců, které obsahují nejprve několik písmen $a$ a za nimi trojnásobné množství písmen $b$. Tedy například $abbb \in Y$, ale $aaabab \notin Y$.
Sestrojte překládací stroj, který přeloží $X$ na $Y$.

d) *(5 bodů)* Na začátku máme množinu $M_1$, jež obsahuje všechny takové řetězce tvořené z písmen $a$, $b$, které obsahují stejný počet písmen $a$ jako $b$. Tedy například $abbbaa \in M_1$, ale $aabab \notin M_1$.
Nové množiny můžeme sestrojovat následujícími operacemi:
- ▷ překladem nějaké již sestrojené množiny nějakým strojem (můžeme použít pokaždé jiný stroj),
- ▷ sjednocením dvou již sestrojených množin,
- ▷ průnikem dvou již sestrojených množin.

Provedením co nejmenšího počtu operací sestrojte množinu $G$, která obsahuje právě všechny řetězce, v nichž je nejprve několik písmen $a$, potom stejný počet písemen $b$, a nakonec stejný počet písmen $c$. Tedy například $aabbcc \in G$, ale $abcc \notin G$, a ani $bac \notin G$.

*Překládací stroj* přijímá na vstupu řetězec znaků. Tento řetězec postupně čte a podle předem zvolené soustavy pravidel (tedy podle svého programu) občas nějaké znaky zapíše na výstup. Když stroj zpracuje celý vstupní řetězec a úspěšně ukončí svůj výpočet, vezmeme řetězec znaků zapsaný na výstup a nazveme ho *překladem* vstupního řetězce.

Výpočet stroje nemusí být jednoznačně určen. Jinými slovy, soustava pravidel může někdy stroji umožnit, aby se rozhodl o dalším postupu výpočtu. V takovém případě se může stát, že některý řetězec bude mít více různých překladů.

Naopak, může se stát, že v určité situaci se podle daných pravidel nebude moci v překladu pokračovat vůbec. V takovém případě se může stát, že některý řetězec nebude mít vůbec žádný překlad.

**Formálnější definice překládacího stroje.** Každý překládací stroj pracuje nad nějakou předem zvolenou konečnou množinou znaků. Tuto množinu znaků budeme nazývat *abeceda* a značit $\Sigma$. V soutěžních úlohách bude vždy $\Sigma$ známa ze zadání úlohy. Abeceda nebude nikdy obsahovat znak \$, ten budeme používat k označení konce vstupního řetězce.

Stroj si může během překladu řetězce pamatovat informaci konečné velikosti. Formálně tuto skutečnost definujeme tak, že stroj se v každém okamžiku překladu nachází v jednom z *konečně mnoha* stavů. Nutnou součástí programu překládacího stroje je tedy nějaká konečná *množina stavů*, v nichž se stroj může nacházet. Tuto množinu označíme $K$. Kromě samotné množiny stavů je také třeba uvést, ve kterém stavu se stroj nachází na začátku každého překladu. Tento stav nazveme *počáteční stav*.

Program stroje se skládá z konečného počtu překladových pravidel. Každé pravidlo má tvar čtveřice $(p, u, v, q)$, kde $p, q \in K$ jsou nějaké dva stavy a $u, v$ jsou nějaké dva řetězce znaků z abecedy $\Sigma$.

Stavy $p$ a $q$ mohou být i stejné. Řetězec $u$ může být tvořen jediným znakem \$. Řetězce $u$ a $v$ mohou být i stejné. Některý z těchto řetězců může být případně prázdný. Aby se program lépe četl, budeme místo prázdného řetězce psát symbol $\varepsilon$.

Překladové pravidlo má následující význam: „Když je stroj právě ve stavu $p$ a dosud nepřečtená část vstupu začíná řetězcem $u$, může stroj tento řetězec ze vstupu přečíst, na výstup zapsat řetězec $v$ a změnit

svůj stav na $q$." Všimněte si, že pravidlo tvaru $(p, \varepsilon, v, q)$ můžeme použít vždy, když se stroj nachází ve stavu $p$, bez ohledu na to, jaké znaky ještě zůstávají na vstupu.

Ještě potřebujeme stanovit, kdy překlad úspěšně skončil. V první řadě budeme požadovat, aby překládací stroj přečetl celý vstupný řetězec. Kromě toho umožníme stroji „odpovědět", zda se mu překlad podařil nebo ne. To zařídíme tak, že některé stavy stroje označíme jako *koncové stavy*. Množinu všech koncových stavů budeme značit $F$.

**Formální definice překládacího stroje.** Překládací stroj je uspořádaná pětice $(K, \Sigma, P, q_0, F)$, kde $\Sigma$ a $K$ jsou *konečné* množiny, $q_0 \in K$, $F \subseteq K$ a $P$ je *konečná* množina překladových pravidel popsaných výše. Přesněji, nechť $\Sigma^*$ je množina všech řetězců tvořených znaky ze $\Sigma$, potom $P$ je *konečná* podmnožina množiny $K \times (\Sigma^* \cup \{\$\}) \times \Sigma^* \times K$.

(Pro každé $q \in K$ budeme množinu pravidel, jejichž první složkou je $q$, nazývat „překladová pravidla ze stavu $q$".)

Chceme-li definovat konkrétní překládací stroj, musíme uvést všech pět výše uvedených objektů.

Když už máme definován konkrétní stroj $A = (K, \Sigma, P, q_0, F)$, můžeme určit, jak tento stroj překládá konkrétní řetězec. Uvedeme nejprve formální definici a potom ji slovně vysvětlíme.

Množina *platných překladů* řetězce $u$ překládacím strojem $A$ je:

$$A(u) = \Big\{ v \;\Big|\; \exists n \geqq 0 \exists (p_1, u_1, v_1, r_1), \ldots, (p_n, u_n, v_n, r_n) \in P : (\forall i \in \{1, \ldots, n-1\} : r_i = p_{i+1}) \wedge p_1 = q_0 \wedge r_n \in F \wedge \wedge \exists k \geqq 0 \colon u_1 u_2 \ldots u_n = u \underbrace{\$ \ldots \$}_{k} \wedge v_1 v_2 \ldots v_n = v \Big\}.$$

Definice stanoví, kdy je řetězec $v$ překladem řetězce $u$. Vysvětlíme si slovně význam jednotlivých řádků definice:

- První řádek říká, že aby se dalo $u$ přeložit na $v$, musí existovat nějaká posloupnost překladových pravidel, kterou při tomto překladu použijeme. Další dva řádky popisují, jak tato posloupnost musí vypadat.
- Druhý řádek zabezpečuje, aby stavy v použitých pravidlech byly správné: První pravidlo musí být pravidlem z počátečního stavu, každé další pravidlo musí být pravidlem z toho stavu, do něhož se stroj dostal použitím předcházejícího pravidla.
  Navíc stav, v němž se bude stroj nacházet po skončení výpočtu, musí být koncový.

▷ Poslední řádek popisuje řetězce, které stroj při použití dotyčných překladových pravidel čte a zapisuje.
Řetězec, který při použití těchto pravidel stroj přečte ze vstupu, musí být skutečně zadaným řetězcem $u$, případné může být zprava doplněn vhodným počtem znaků \$.
Řetězec, který stroj zapíše na výstup, musí být přesně řetězcem $v$.

**K čemu budeme používat překládací stroje?** Překládací stroje nám budou sloužit k získání překladu jedné množiny řetězců na jinou množinu řetězců. Jestliže $A$ je překládací stroj a $M \subseteq \Sigma^*$ nějaká množina řetězců, potom překlad množiny $M$ strojem $A$ je množina

$$A(M) = \bigcup_{u \in M} A(u).$$

Jinými slovy, výslednou množinu $A(M)$ sestrojíme tak, že vezmeme všechny řetězce z $M$ a pro každý z nich přidáme do $A(M)$ všechny jeho platné překlady.

*Příklad 1*: Mějme abecedu $\Sigma = \{0, \ldots, 9\}$. Nechť $M$ je množina všech řetězců, které představují zápisy kladných celých čísel v desítkové soustavě. Sestrojíme překládací stroj $A$, pro který bude platit, že překladem této množiny $M$ bude množina zápisů všech kladných celých čísel, která jsou dělitelná třemi.

Řešení. Nejjednodušší bude prostě vybrat z $M$ ta čísla, která jsou dělitelná třemi. Náš překládací stroj bude kopírovat cifry ze vstupu na výstup, přičemž si bude pomocí stavu pamatovat, jaký zbytek po dělení třemi dává dosud přečtené (a zapsané) číslo. Nachází-li se po dočtení vstupu ve stavu odpovídajícím zbytku 0, přejde do koncového stavu.

Formálně $A$ bude pětice $(K, \Sigma, P, 0, F)$, kde $K = \{0, 1, 2, end\}$, $F = \{end\}$ a překladová pravidla vypadají následovně:

$$P = \{(x, y, y, z) \mid x \in \{0, 1, 2\} \ \wedge \ y \in \Sigma \ \wedge \ z = (10x + y) \bmod 3\} \cup \\ \cup \{(0, \$, \varepsilon, end)\}.$$

*Příklad 2*: Mějme abecedu $\Sigma = \{a, e, i, \bullet, -\}$. Sestrojíme překládací stroj $B$, pro který bude platit, že překladem libovolné množiny $M$, která obsahuje pouze řetězce z písmen $a$, $e$ a $i$, bude množina stejných řetězců zapsaných v morseovce (bez oddělovačů mezi znaky). Zápisy našich písmen v morseovce vypadají následovně: $a$ je $\bullet -$, $e$ je $\bullet$ a $i$ je $\bullet\bullet$.

Například množinu $M = \{ae, eea, ia\}$ by náš stroj měl přeložit na množinu $\{\bullet - \bullet, \bullet\bullet\bullet -\}$. (Všimněte si, že řetězce *eea* a *ia* mají v morseovce bez oddělovačů stejný zápis.)

Řešení. Překládací stroj $B$ bude číst vstupní řetězec po znacích a vždy zapíše na výstup kód přečteného znaku.

Formálně $B$ bude pětice $(K, \Sigma, P, \heartsuit, F)$, kde $K = \{\heartsuit\}$, $F = \{\heartsuit\}$ a překladová pravidla vypadají takto:

$$P = \{(\heartsuit, a, \bullet -, \heartsuit), (\heartsuit, e, \bullet, \heartsuit), (\heartsuit, i, \bullet\bullet, \heartsuit)\}.$$

Všimněte si, že nepotřebujeme nijak zvlášť kontrolovat, zda jsme na konci vstupu. Během celého překladu je totiž stroj v koncovém stavu, takže jakmile přečte poslední znak ze vstupu, bude vytvořený překlad platný.

*Příklad 3*: Mějme abecedu $\Sigma = \{a, e, i, \bullet, -\}$. Sestrojíme překládací stroj $C$, pro který bude platit, že překladem libovolné množiny $M$, která obsahuje pouze řetězce tvořené znaky $\bullet$ a $-$, bude množina *všech* řetězců z písmen $a$, $e$ a $i$, jejichž zápisy v morseovce (bez oddělovačů mezi znaky) jsou obsaženy v množině $M$. Například množinu $M = \{\bullet - \bullet, \bullet\bullet\bullet -\}$ by náš stroj měl přeložit na $\{ae, eea, ia\}$.

Řešení. Našemu překládacímu stroji dáme možnost rozhodnout se v každém okamžiku překladu, že bude číst kód nějakého písmena a zapíše na výstup toto písmeno. Potom každé možnosti, jak lze rozdělit vstupní řetězec na kódy písmen, bude odpovídat jeden platný překlad.

Formálně $C$ bude pětice $(K, \Sigma, P, \diamondsuit, F)$, kde $K = \{\diamondsuit\}$, $F = \{\diamondsuit\}$ a překladová pravidla vypadají následovně:

$$P = \{(\diamondsuit, \bullet -, a, \diamondsuit),\ (\diamondsuit, \bullet, e, \diamondsuit),\ (\diamondsuit, \bullet\bullet, i, \diamondsuit)\}.$$

Ukážeme si, jak mohl probíhat překlad řetězců z výše uvedené množiny $M$. Existují tyto tři možnosti:

- $(\diamondsuit, \bullet -, a, \diamondsuit), (\diamondsuit, \bullet, e, \diamondsuit)$
- $(\diamondsuit, \bullet\bullet, i, \diamondsuit), (\diamondsuit, \bullet -, a, \diamondsuit)$
- $(\diamondsuit, \bullet, e, \diamondsuit), (\diamondsuit, \bullet, e, \diamondsuit), (\diamondsuit, \bullet -, a, \diamondsuit)$

Kdybychom zkusili pro libovolný vstupní řetězec z $M$ použít překladová pravidla v jiném pořadí – např. pro vstup $\bullet\bullet\bullet -$ použít třikrát pravidlo $(\diamondsuit, \bullet, e, \diamondsuit)$ – nepodaří se nám dočíst vstupní řetězec až do konce.

*Příklad 4*: Mějme abecedu $\Sigma = \{a, b, c\}$. Nechť množina $X$ obsahuje právě všechny řetězce, v nichž je obsažen stejný počet znaků $a$ a $b$. Tedy například $abbccac \in X$, ale $cbaa \notin X$.

Nechť množina $Y$ obsahuje právě všechny řetězce, které neobsahují žádné $a$, neobsahují podřetězec $bc$, a písmen $c$ je dvakrát více než písmen $b$. Tedy například $ccccbb \in Y$, ale $cccbcb \notin Y$ a $acacba \notin Y$.

Sestrojíme překládací stroj $D$, který přeloží $X$ na $Y$.

ŘEŠENÍ. Budeme překládat jenom některé vhodné řetězce z množiny $X$. Budou to ty řetězce, které neobsahují žádné $c$ a všechna $a$ v nich předcházejí všem znakům $b$. Takovýto řetězec přeložíme tak, že nejprve každé $a$ přepíšeme na $cc$, a potom zkopírujeme na výstup všechna $b$. Tedy například překladem slova $aabb$ bude slovo $ccccbb$.

Formálně $D$ bude pětice $(K, \Sigma, P, \textbf{čti-a}, F)$, kde $K = \{\textbf{čti-a}, \textbf{čti-b}\}$, $F = \{\textbf{čti-b}\}$ a překladová pravidla vypadají takto:

$$P = \{(\textbf{čti-a}, a, cc, \textbf{čti-a}),\ (\textbf{čti-a}, \varepsilon, \varepsilon, \textbf{čti-b}),\ (\textbf{čti-b}, b, b, \textbf{čti-b})\}.$$

Proč tento překládací stroj funguje? Když vstupní řetězec obsahuje nějaké písmeno $c$, při jeho překládání se u prvního výskytu $c$ náš stroj zastaví. Proto takové řetězce nemají žádný platný překlad. Podobně nemají platný překlad řetězce, v nichž není dodrženo pořadí písmen $a$ a $b$. Po přečtení nějaké posloupnosti písmen $a$ přejde stroj pomocí druhého pravidla do stavu **čti-b**, a pokud se poté ještě objeví na vstupu $a$, stroj se zastaví.

Platné překlady tedy existují skutečně pouze pro slova výše popsaného tvaru. Je zjevné, že překladem každého z nich získáme nějaký řetězec z $Y$, takže $D(X) \subseteq Y$. Naopak, vybereme-li si libovolné slovo z $Y$, snadno najdeme slovo z $X$, které se na něj přeloží. Proto také $Y \subseteq D(X)$, takže $Y = D(X)$.

## P – II – 1

### Parkování kočárů

Král Kazimír vdává dceru. U takové slávy (a tolika jídla zdarma) nemůže chybět žádný šlechtic z okolí. A jak tak šlechtici jedou ve svých kočárech na svatbu, vůbec netuší, kolik starostí sluhům krále Kazimíra způsobí při řešení následujícího problému.

Všechny kočáry je třeba zaparkovat, a to ne jen tak nahodile. Kočáry musí stát v řadách za sebou a těchto řad musí být co nejméně, aby královi neponičily trávník.

Dvorní etiketa káže, že až se budou hosté rozjíždět ze svatby domů, musí odjíždět seřazeni podle důležitosti, nejdůležitější host jako poslední.

To ovšem ještě není všechno. Kočáry, které jsou zaparkovány v jedné řadě, musí samozřejmě odjíždět v pořadí, ve kterém stojí. Aby se předešlo srážkám kočárů, sluhové je navíc chtějí zaparkovat tak, aby po skončení svatby odjela vždy celá jedna řada kočárů, až potom začala odjíždět další řada atd.

**Soutěžní úloha.** Sluhové přesně znají pořadí, v němž budou přijíždět hosté na svatbu, a znají také důležitost každého nich. Když parkují kočár, mohou ho zaparkovat *na začátek nebo na konec* libovolné již existující řady kočárů, případně ho mohou postavit do nové řady. Sluhové musí zaparkovat jednotlivé kočáry v tom pořadí, jak hosté přijíždějí.

Vaším úkolem je určit nejmenší počet řad, který stačí k tomu, aby po správném zaparkování mohly kočáry odjíždět ve stanoveném pořadí.

*Formát vstupu*: Vstup začíná celým číslem $N$ ($1 \leqq N \leqq 100\,000$), které určuje počet hostů. Následuje $N$ *různých* kladných celých čísel $d_i$ ($1 \leqq d_i \leqq 10^9$), která určují důležitost hostů v pořadí, v jakém přijíždějí (větší číslo představuje důležitějšího hosta).

*Formát výstupu*: Výstup je tvořen jediným řádkem, který obsahuje jedno celé číslo představující nejmenší možný počet řad pro zaparkování kočárů.

*Příklady*:

| *Vstup*: | *Výstup*: |
|---|---|
| 10 | 1 |
| 10 9 11 12 13 8 14 7 6 100 | |

*V tomto případě stačí držet se pravidla „kočáry méně důležité než 10 jdou na začátek řady, ostatní na konec“.*

| *Vstup*: | *Výstup*: |
|---|---|
| 6 | 2 |
| 12 17 9 23 16 14 | |

*Jedním možným řešením je zaparkovat kočáry 12 a 17 do různých řad, a pak kočár 9 postavit před kočár 12, kočár 23 za kočár 17, kočár 16 před kočár 17 a nakonec kočár 14 před kočár 16.*

| *Vstup*: | *Výstup*: |
|---|---|
| 12 | 6 |
| 1 3 2 5 4 7 6 9 8 11 10 12 | |

*V jediném optimálním řešení budou po zaparkování řady: (1 2), (3 4), (5 6), (7 8), (9 10) a (11 12).*

# P – II – 2

## Dluhopisy

Kleofáš nedávno zdědil po své bohaté tetičce Anastázii hromadu peněz. Nevěděl však co s nimi, a proto se rozhodl investovat je do dluhopisů. Od vás si chce nechat poradit, jak by měl svou investici optimálně spravovat.

Pro jednoduchost budeme předpokládat následující skutečnosti:

- ▷ Každý typ dluhopisu má svoji pevnou cenu, stejnou při koupi i prodeji.
- ▷ Každý typ dluhopisu má pevně daný roční výnos, který se vyplácí vždy na konci roku.
- ▷ Je možné nakoupit libovolné množství každého typu dluhopisu.

Uvažujme například následující situaci: Banka nabízí dva typy dluhopisů: Dluhopisy za 4 000 korun s ročním výnosem 400 a dluhopisy za 3 000 korun s ročním výnosem 250. Má-li Kleofáš 10 000 korun, nejlepší, co s nimi může udělat, je koupit dva dluhopisy po 3 000 a jeden za 4 000, čímž získá roční výnos 900 korun. Po dvou letech obdrží Kleofáš dvakrát výnosy a bude mít celkově kapitál 11 800 korun. V tomto okamžiku se mu vyplatí jeden dluhopis za 3 000 korun prodat a místo něj si koupit dluhopis za 4 000. Po třetím roce bude jeho kapitál roven 12 850 korunám.

**Soutěžní úloha.** Napište program, který přečte ze vstupu Kleofášův počáteční kapitál, ceny a výnosy nabízených dluhopisů a počet roků, na které chce Kleofáš investovat, a spočítá, kolik nejvíce peněz může Kleofáš mít po uplynutí daného počtu roků.

*Formát vstupu*: Na prvním řádku vstupu je jedno celé číslo $K$ ($1 \leqq K \leqq 1\,000\,000$), které udává Kleofášův počáteční kapitál. Na druhém řádku je uveden počet typů nabízených dluhopisů $D$ ($1 \leqq D \leqq 100$). Na třetím řádku je $D$ dvojic celých čísel $c_i$ a $v_i$, které představují ceny a výnosy jednotlivých dluhopisů ($0 < c_i \leqq 10^9$, $0 \leqq v_i \leqq c_i/10$, $c_i$ je vždy násobkem $T = 1000$). Na posledním řádku vstupu je uveden počet roků $R$ ($1 \leqq R \leqq 40$).

Časovou složitost svého algoritmu vyjádřete pomocí $K$, $D$, $T$ a $R$. Navrhněte algoritmus, který pro hodnoty $K$, $D$, $T$ a $R$ z výše uvedených rozsahů bude co nejrychlejší.

*Formát výstupu*: Výstupem programu je jediné číslo, které určuje maximální hodnotu Kleofášova kapitálu po $R$ letech obchodování s dluho-

pisy. Můžete předpokládat, že se tato hodnota vejde do běžné celočíselné proměnné.

*Příklady*:

| *Vstup*: | *Výstup*: |
|---|---|
| 10000<br>2<br>4000 400 3000 250<br>4 | 14050 |

*Příklad ze zadání úlohy. Ve čtvrtém roce bude Kleofáš vlastnit 3 dluhopisy po 4 000, čímž vydělá dalších 1 200.*

| *Vstup*: | *Výstup*: |
|---|---|
| 100000<br>3<br>103000 9001 47000 7 83000 100<br>31 | 112001 |

*Kleofáš koupí jeden dluhopis za 83 000. Tím za 30 let získá 3 000 korun. Na poslední rok si konečně může koupit první dluhopis za 103 000. V posledním roce tedy vydělá dalších 9 001.*

| *Vstup*: | *Výstup*: |
|---|---|
| 100000<br>3<br>103000 9001 47000 7 83000 100<br>37 | 166014 |

*Pokračování z předchozího příkladu. Po roce 36 Kleofáš dokoupí dluhopis za 47 000, takže v posledním roce získá o 7 korun více.*

## P – II – 3

### Piškvorkový turnaj

Silvestr se rozhodl, že uspořádá programátorský turnaj v piškvorkách. Požádal tedy své přátele, aby vytvořili programy pro hraní piškvorek, které se turnaje zúčastní. Silvestrovi přátelé na jeho výzvu rychle zareagovali, a tak velký turnaj může začít.

Pravidla turnaje určil Silvestr velmi jednoduše: v každém kole se náhodně vylosují dva programy, které vzájemně sehrají partii, a program, který prohraje, z turnaje nadobro vypadne.

Den před turnajem však Silvestra přemohla zvědavost a zkusil pustit některé dvojice programů proti sobě. Poté si však uvědomil, že se tímto

svým počínáním připravil o velkou část překvapení spojenou s turnajem. Protože všechny programy jsou deterministické (tj. nepoužívají náhodnost), dopadne souboj každých dvou programů vždy stejně. Silvestr už tedy ví, že některé programy turnaj nemohou vyhrát. Pro jednoduchost předpokládáme, že to, který program v dvojici začne souboj, výsledek souboje těchto dvou programů neovlivní.

**Soutěžní úloha.** Pro dané výsledky soubojů některých dvojic programů určete, které programy mohou v turnaji ještě zvítězit.

*Formát vstupu*: Na prvním řádku vstupu je jedno celé číslo $N$ ($1 \leqq \leqq N \leqq 100\,000$), které určuje počet programů přihlášených do turnaje. Programy jsou očíslovány od 1 do $N$.

Následuje $N$ řádků, které popisují výsledky zápasů, které Silvestr již zná; $i$-tý z těchto řádků začíná číslem $d_i$, které určuje počet programů poražených $i$-tým programem ve vzájemných soubojích. Tento řádek pak obsahuje $d_i$ čísel programů, které $i$-tý program porazil. Těchto $d_i$ čísel je seřazeno podle velikosti od nejmenšího po největší.

Označme počet zápasů $d_1 + \ldots + d_N$, které Silvestr den před turnajem spustil, jako $M$. Můžete předpokládat, že platí $0 \leqq M \leqq 1\,000\,000$. Také můžete předpokládat korektnost vstupu, tedy speciálně, že pokud program $x$ porazil program $y$, pak program $y$ neporazil program $x$.

*Formát výstupu*: Výstupem programu je jediný řádek, na kterém budou uvedena čísla všech programů, které mohou v turnaji zvítězit.

*Příklady*:

| *Vstup*: | *Výstup*: |
|---|---|
| 4<br>2 2 3<br>0<br>1 2<br>1 2 | 1 3 4 |

*Do turnaje jsou přihlášeny 4 programy. Program 1 v souboji porazí programy 2 a 3, programy 3 a 4 porazí program 2. Program 2 tedy prohraje souboj s libovolným jiným programem, a tak určitě nemůže turnaj vyhrát.*

*Ostatní programy v turnaji však zvítězit mohou. Jako příklad si předveďme, jak může v turnaji zvítězit program 3: nejdříve program 3 vyřadí program 2, pak program 4 vyřadí program 1 a nakonec program 3 vyřadí program 4.*

| *Vstup*: | *Výstup*: |
|---|---|
| 5 | 1 2 3 4 5 |
| 2 2 3 | |
| 0 | |
| 1 2 | |
| 1 2 | |
| 0 | |

*Tentokrát může zvítězit libovolný z programů 1, 2, 3, 4 a 5. Např. program 2 může zvítězit následovně: nejdříve program 3 porazí program 4, pak program 5 postupně vyřadí programy 3 a 1 a nakonec program 2 vyřadí program 5.*

## P – II – 4

### Překládací stroje

Studijní text je stejný jako v úloze P–I–4.

**Soutěžní úloha.**

a) *(6 bodů)* Nechť $M_1$ je množina tvořená všemi řetězci písmen $a$ a $b$, které obsahují stejný počet písmen $a$ a $b$. Tedy např. $abbbaa \in M_1$, avšak $aabab \notin M_1$.

Nové množiny můžeme sestrojit následujícími operacemi:

- ▷ překladem již sestrojené množiny pomocí překládacího stroje (lze použít jiné překládací stroje při různých překladech),
- ▷ sjednocením dvou již sestrojených množin, nebo
- ▷ průnikem dvou již sestrojených množin.

Pomocí co nejmenšího počtu výše popsaných operaci sestrojte množinu $G$, která obsahuje právě všechny řetězce písmen $a$, $b$ a $c$, které obsahují stejné množství písmen $a$ jako písmen $b$ a také jako písmen $c$. Tedy například $aabbcc \in G$, $bac \in G$, ale $abcc \notin G$.

b) *(4 body)* Množina $X$ obsahuje zápisy kladných celých čísel v desítkové soustavě, v nichž se vyskytuje stejný počet číslic 1 a 2. Tedy například $1122 \in X$, $21231 \in X$, $47 \in X$, ale $112 \notin X$ a $031221 \notin X$ (zápis kladného čísla nemůže začínat číslicí 0).

Množina $Y$ obsahuje zápisy v desítkové soustavě těch kladných čísel, která jsou dělitelná 7. Tedy například $140 \in Y$, $7707 \in Y$, ale $47 \notin Y$ a $07 \notin Y$.

Sestrojte překládací stroj, který přeloží množinu $X$ na množinu $Y$, anebo dokažte, že takový překládací stroj neexistuje.

# P – III – 1

## Karpaty

Už zanedlouho si splníte svůj velký cestovatelský sen a vydáte se na vytouženou pouť přes celé Vnější Karpaty. Vše máte dobře naplánováno — začnete v Beskydech a budete pokračovat přes Slovensko, Polsko a Ukrajinu až do Rumunska, odkud už se chcete vrátit skrze civilizaci. Cestou máte v úmyslu vystoupit na co nejvíce hor, kolem kterých pojedete, ale jak se znáte, tak jakmile vystoupíte na nějakou horu, už se vám nebude chtít na žádnou stejně vysokou nebo dokonce nižší.

V rámci pečlivého plánování jste si vypsali výšky všech hor v tom pořadí, v jakém kolem nich budete projíždět a přemýšlíte, na které z hor budete chtít vystoupit, abyste vystoupili během své cesty na co největší počet hor.

Po krátkém zamyšlení jste si uvědomili, že těch nejdelších posloupností hor takových, že jejich výšky striktně rostou, může být i více. Místo výčtu všech možností byste si raději u každé hory poznačili, zda ji navštívíte určitě (tedy pokud je v každé nejdelší posloupnosti hor, kde výšky hor rostou), zda ji můžete navštívit, nebo ji určitě nenavštívíte, pokud chcete vystoupat na co největší počet hor.

**Soutěžní úloha.** Pro zadanou posloupnost celých kladných čísel určete, která ze zadaných čísel leží v každé nejdelší rostoucí podposloupnosti, v nějaké nejdelší rostoucí podposloupnosti a v žádné nejdelší rostoucí podposloupnosti.

*Formát vstupu*: Vstup je tvořen dvěma řádky. Na prvním řádku je jedno celé číslo $N$, které určuje počet hor na vaší cestě. Druhý řádek pak obsahuje $N$ celých kladných čísel $v_1 \ldots v_n$, která udávají výšku hor.

*Formát výstupu*: Výstup je tvořen jedním řádkem s $N$ slovy oddělenými jednou mezerou. Slova ve výstupu jsou '`musím`', '`mohu`' a '`nemohu`' podle toho, zda na danou horu musíte, můžete nebo nemůžete vystoupat, pokud chcete během své cesty zdolat co největší počet hor tak, aby jejich výšky postupně rostly.

*Příklad*:
*Vstup*

```
9
3 2 1 4 5 6 1 8 7
```

*Výstup*

```
mohu mohu mohu musím musím musím nemohu mohu mohu
```

Nejdelší posloupnost rostoucích hor zde má délku 5 a z první trojice a poslední dvojice vždy obsahuje právě jeden prvek.

## P – III – 2

### Velechrám

Oslavy 75. narozenin královny Šeherezády se blíží a v paláci se to jen hemží. Vyměňují se tapisérie, pokládají se nové koberce a přemalovávají se obrazy. Radní se rozhodli, že královnu překvapí novou duhovou výzdobou velechrámu. Chtějí nechat natřít sloupy v chrámu různými barvami, aby z toho jen oči přecházely. Aby byl výsledný dojem co nejlepší, je třeba, aby v žádné řadě sloupů neměly dva sloupy stejnou barvu. Královská pokladna však není bezedná, a proto je třeba nakoupit co nejméně druhů barev. Proto si vás zavolali na pomoc.

**Soutěžní úloha.** Velechrám si lze představit jako šachovnici o velikosti $N \times N$. Na čtvercových polích stojí $M$ sloupů, tj. ne na všech polích stojí sloup. Žádné dva sloupy nestojí na stejném poli. Vaším úkolem je obarvit tyto sloupy co nejmenším počtem $K$ barev tak, aby žádné dva sloupy ve stejném řádku nebo sloupci neměly stejnou barvu.

*Formát vstupu*: Na prvním řádku dostanete dvě přirozená čísla $N$ a $M$, velikost šachovnice $1 \leqq N \leqq 500$ a počet sloupů $0 \leqq M \leqq N^2$.

Následuje $M$ řádků popisujících polohu sloupů. Na $i$-tém z nich najdete dvě celá čísla $R_i$ a $S_i$, řádek $1 \leqq R_i \leqq N$ a sloupec $1 \leqq S_i \leqq N$ pole, na kterém stojí $i$-tý sloup.

*Formát výstupu*: První řádek výstupu obsahuje minimální počet $K$ barev. Na následujících $M$ řádků vypište barvy sloupů. Na $i$-tý z těchto řádků napište číslo barvy $B_i$, $1 \leqq B_i \leqq K$, kterou má být obarven $i$-tý sloup. Optimálních obarvení bude zřejmě více (barvy lze například mezi sebou libovolně permutovat), vypište proto libovolné z nich.

*Příklady*:

| *Vstup*: | *Výstup*: |
|---|---|
| 3 4 | 2 |
| 1 1 | 1 |
| 1 3 | 2 |
| 3 3 | 1 |
| 3 1 | 2 |

*Sloupy leží ve vrcholech čtverce. Protilehlé sloupy dostanou stejnou barvu.*

| *Vstup*: | *Výstup*: |
|---|---|
| 4 8 | 3 |
| 1 1 | 1 |
| 1 3 | 2 |
| 1 4 | 3 |
| 2 1 | 2 |
| 3 3 | 3 |
| 4 1 | 3 |
| 4 2 | 2 |
| 4 3 | 1 |

## P – III – 3

### Překládací stroje

Studijní text je stejný jako v úloze P–I–4.

Uvažme následující množiny $A$, $B$, $C$ a $D$ řetězců. Množina $A$ obsahuje všechny dobře uzávorkované řetězce znaků '(', ')', '[', ']', '{' a '}'. Množinu $A$ lze rekurzivně nadefinovat následujícím způsobem: $A$ obsahuje prázdný řetězec $\varepsilon$ a všechny řetězce tvaru $(\alpha_1 \ldots \alpha_k)$, $[\alpha_1 \ldots \alpha_k]$ a $\{\alpha_1 \ldots \alpha_k\}$, kde $\alpha_1, \ldots, \alpha_k$ jsou nějaké řetězce v $A$ již obsažené. Množina $A$ tedy obsahuje řetězce `(([{}]))` a `()()(){}`, ale neobsahuje ani jeden z řetězců `([)]` a `())((`.

Množiny $B$ a $C$ jsou množiny řetězců nad dvojpísmennou abecedou $\{a, b\}$. Množina $B$ obsahuje řetězce nejprve tvořené $n \geqq 0$ písmeny $a$ a pak alespoň $2n$ písmeny $b$. Množina $C$ obsahuje ty řetězce, které jsou tvořeny stejným počtem písmen $a$ a $b$. Např. $aabbbbb \in B$ a $abba \in C$, ale $aaabbbb \notin B$, $bbba \notin B$ a $ababb \notin C$.

Množina $D$ je tvořena těmi řetězci, které nejprve obsahují $n \geqq 0$ písmen $a$, pak $n$ písmen $b$ a nakonec $n$ písmen $c$. Tedy, $aaabbbccc \in D$, ale $aababbbccc \notin D$ a $bbbaaaccc \notin D$.

**Soutěžní úloha.**

a) *(2 body)* Navrhněte překládací stroj, který přeloží množinu $A$ na množinu $B$.

b) *(3 body)* Nalezněte množinu řetězců $E$ nad abecedou $\{a, b\}$ a překládací stroje $M_1$ a $M_2$ takové, že stroj $M_1$ přeloží množinu $E$ na množinu $A$ a stroj $M_2$ přeloží množinu $E$ na množinu $D$.

c) *(5 bodů)* Sestrojte překládací stroj, který přeloží množinu $A$ na množinu $C$. Nezapomeňte, že nutnou součástí řešení je i důkaz, že vámi navržený překládací stroj přeloží množinu $A$ právě na množinu $C$.

# P – III – 4

## Sbírka čísel

*Program:* `sbirka.pas / sbirka.c / sbirka.cpp`
*Vstup:* `sbirka.in`
*Výstup:* `sbirka.out`

Pan Pterodaktylos je sběratelem. Vášnivým a takříkajíc univerzálním. Sbírá úplně všechno: známky, motýly, známky s obrázky motýlů, ba dokonce i všechna přirozená čísla. Těch má několik zvláště vydařených sbírek.

Každá sbírka čísel je určena parametry $B$ a $D$ a obsahuje všechna nezáporná celá čísla, která se dají zapsat v soustavě o základu $B$ pomocí nejvýše $D$ číslic.[1] Tato čísla jsou ve sbírce seřazena důmyslným způsobem: nejprve vzestupně podle svého ciferného součtu a uvnitř každé skupiny se stejným ciferným součtem pak vzestupně podle hodnoty čísel. Například sbírka označená parametry $B = 3$, $D = 2$ obsahuje nejvýše dvouciferná čísla v trojkové soustavě, a to v následujícím pořadí: 0; 1, 10; 2, 11, 20; 12, 21; 22 (středníkem jsou odděleny bloky čísel se stejným ciferným součtem).

Pro větší hodnoty $B$ a $D$ je taková sbírka dosti rozsáhlá. Pokud pan Pterodaktylos chce ze sbírky vytáhnout $k$-té číslo v pořadí, čeká ho obvykle dlouhá cesta mezi kilometry regálů, než se k němu dostane. Vás napadlo, že to vůbec není potřeba — když víte, že sbírka obsahuje opravdu všechna čísla, lze $k$-té číslo v pořadí snadno spočítat.

**Úloha.** Napište program, který dostane parametry $B$ a $D$ sbírky čísel a číslo $k$ a odpoví $k$-tým číslem ve sbírce.

*Vstup*: Vstupní soubor `sbirka.in` obsahuje jediný řádek, na němž jsou zapsána tři čísla oddělená mezerami: základ soustavy $B$ ($2 \leqq B \leqq 10$), délka čísla $D$ ($1 \leqq D \leqq 30$) a pořadí hledaného čísla ($1 \leqq k \leqq 10^{18}$). Můžete se spolehnout na to, že všechna čísla ve sbírce jsou menší než $10^{18}$.

*Výstup*: Výstupní soubor `sbirka.out` obsahuje jediný řádek a na něm $k$-té číslo sbírky zapsané v soustavě o základu $B$. Nevýznamné nuly na začátku čísla nevypisujte.

---

[1] Připomeňme, že zápis $x_{k-1} \ldots x_0$, $x_i \in \{0, \ldots, B-1\}$, v soustavě o základu $B$ představuje číslo $\sum_{i=0}^{k-1} x_i B^i$.

*Příklad 1*:

| sbirka.in | sbirka.out |
|---|---|
| 3 2 5 | 11 *(viz příklad sbírky výše)* |

*Příklad 2*:

| sbirka.in | sbirka.out |
|---|---|
| 10 15 999999999999998 | 999999999999989 |

## P – III – 5

### Strouhání mrkve

*Program:* mrkev.pas / mrkev.c / mrkev.cpp
*Vstup:* mrkev.in
*Výstup:* mrkev.out

Jak zajíček Hopsálek stárnul, začaly mu padat zuby. Dostal sice pěknou zubní protézu, ale už si nemohl pochutnávat na své oblíbené mrkvi. Rozhodl se tedy, že začne provozovat továrnu na strouhání mrkve. Do továrny pak může přijít zaječí důchodce a mrkev mu tam nastrouhají, aby si na ní mohl pochutnat i v pokročilém věku.

Hopsálek plánuje, že továrna bude fungovat na směny. Každá směna bude trvat přesně $M$ minut a směny budou následovat bezprostředně jedna za druhou. První směna musí začít v jedné z prvních $M$ minut provozu.

Továrna pracuje jenom tehdy, když do ní přijde nějaký (za)ječící důchodce. Pokud víme, že v průběhu celé směny nepřijde ani jeden, můžeme směnu zrušit a tím ušetřit.

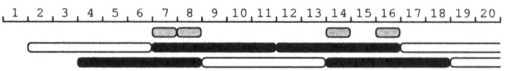
**Úloha.** Zjistili jste přesně, v jakých minutách budou do továrny chodit mrkvechtiví důchodci. Mrkev stihnou zaměstnanci továrny nastrouhat ještě v té minutě, kdy ji zákazník přinesl. Vaším úkolem je zjistit, kterou z prvních $M$ minut má začít první směna, aby bylo směn, ve kterých se v továrně pracuje, co nejméně.

*Vstup*: Na prvním řádku vstupního souboru mrkev.in jsou dvě mezerou oddělená přirozená čísla $M$ a $N$ ($1 \leqq M, N \leqq 1\,000\,000$). $M$ je délka trvání každé směny v minutách, $N$ je počet zaječích seniorů, kteří si přijdou nechat nastrouhat mrkev. Na druhém řádku je mezerou oddělených $N$ přirozených čísel $z_1 < z_2 < \ldots < z_N$ — čísla minut příchodů zaječích zákazníků. Všechna $z_i$ splňují $M < z_i < 2\,000\,000\,000$ a minuty jsou číslovány od jedné.

*Výstup*: Na první řádek výstupního souboru `mrkev.out` vypište jedno přirozené číslo $S$ — nejmenší počet směn, ve kterých se musí v továrně pracovat, tj. těch, které nebudou zrušeny. Na druhý řádek vypište, kterou minutou musí začínat první směna, aby bylo třeba pracovat jenom $S$ směn. Pokud je možností více, vypište všechny v *rostoucím* pořadí oddělené právě jednou mezerou.

*Příklad 1*:

| *Vstup* | *Výstup* |
|---|---|
| 5 4 | 2 |
| 7 8 14 16 | 2 4 |




Šedivou barvou jsou označeny příchody zaječích stařešinů. Spodní dva řádky znázorňují dvě optimální řešení, černé jsou směny, ve kterých se v továrně pracuje.

*Příklad 2*:

| *Vstup* | *Výstup* |
|---|---|
| 5 4 | 3 |
| 7 9 14 17 | 1 2 3 4 5 |

V této situaci jsou všechny možné začátky stejně dobré. Nezapomeňte na to, že je musíte vypsat setříděné.