12. mezinárodní olympiáda v informatice

In: Leo Boček (editor); Karel Horák (editor); Jaromír Šimša (editor); Jaroslav Švrček (editor); Pavel Töpfer (editor): 49. ročník matematické olympiády na středních školách. Zpráva o řešení úloh ze soutěže konané ve školním roce 1999/2000. 41. mezinárodní matematická olympiáda. 12. mezinárodní olympiáda v informatice. (Czech). Praha: Jednota českých matematiků a fyziků, 2005. pp. 179–191.

Persistent URL: http://dml.cz/dmlcz/405021

12. mezinárodní olympiáda v informatice

Tento ročník mezinárodní olympiády v informatice (The International Olympiad in Informatics) se konal ve dnech 23.–30. 9. 2000 v Pekingu v Číně. Olympiády se zúčastnilo 276 soutěžících ze 70 zemí. Soutěž IOI získává v celém světě stále větší zájem a popularitu, počet účastníků se každoročně zvyšuje. Ještě před deseti lety v ní soutěžili studenti jenom z asi dvaceti zemí, pro srovnání loni bylo na IOI 250 soutěžících z 65 zemí.

Olympiáda se konala v konferenčním středisku umístěném v severní části Pekingu v těsném sousedství hotelů Grand Continental a Catic, v nichž bylo zajištěno ubytování a stravování všech účastníků. Pro zajímavost můžeme uvést, že nedaleko se nachází rozsáhlý sportovní komplex a že s těmito hotely, konferenčním střediskem a sportovišti se počítá i pro případ, že bude Číně svěřeno uspořádání letních olympijských her v roce 2008. Celá akce byla organizačně i technicky výborně připravena, vlastní soutěž i vyhodnocování proběhlo zcela bez závad a bez problémů. Pro všechny účastníky byl připraven i velmi zajímavý doprovodný program. Měli jsme možnost navštívit nejvýznamnější historické památky nacházející se v Pekingu a jeho okolí — náměstí Tian'anmen, areály historických císařských paláců Forbidden City a Summer Palace, park s chrámy Temple of Heaven a také slavnou velkou čínskou zeď.

Soutěž byla jako vždy rozdělena do dvou soutěžních dnů, v každém z nich řešili studenti tři úlohy. Řešení úloh v IOI probíhá u počítačů podobným způsobem jako v praktické části celostátního kola kategorie P naší matematické olympiády. Každý soutěžící má přidělen svůj osobní počítač a na práci v každém soutěžním dnu má k dispozici omezený čas 5 hodin. Soutěžní den je vždy zakončen testováním odevzdaných programů, při kterém se sleduje nejen správnost výpočtu, ale pomocí časových limitů také kvalita vytvořených programů (tzn. jejich časové a paměťové nároky). Pro každou úlohu bylo stanoveno maximální možné ohodnocení 100 bodů, každý soutěžící navíc obdržel v každém ze soutěžních dnů

bonus 50 bodů. Celkové pořadí bylo dáno součtem hodnocení všech šesti úloh a obou bonusů, maximálně tedy bylo možné získat 700 bodů.

Slavnostního zakončení olympiády spojeného s vyhlášením výsledků a předáním medailí nejlepším řešitelům se zúčastnila celá řada významných státních představitelů, zástupci čínských odborných informatických společností a organizací a zástupci sponzorů. Mezinárodní olympiáda v informatice je výhradně soutěží jednotlivců a žádné oficiální pořadí družstev v ní podle pravidel IOI není vyhlašováno. Celkem bylo v soutěži uděleno 23 zlatých medailí, 47 stříbrných a 69 bronzových medailí. Počet udělených medailí se stanoví v IOI podle pravidla, že medaili získá přibližně polovina soutěžících, přičemž zlaté, stříbrné a bronzové medaile se rozdělují v poměru 1 : 2 : 3. Drobné odchylky od tohoto pravidla jsou způsobeny tím, že více soutěžících může dosáhnout stejného bodového zisku (udělených medailí pak může být o něco více).

Členové reprezentačního družstva z České republiky byli vybráni na základě výsledků celostátního kola kategorie P 49. ročníku Matematické olympiády. Na IOI 2000 nás reprezentovalo družstvo ve složení *Jakub Bystroň* (absolvent Gymnázia Karviná), *Pavel Charvát* (absolvent Gymnázia Ohradní v Praze 4), *Ondřej Rucký* (absolvent Gymnázia na Mikulášském nám. v Plzni) a *Jiří Svoboda* (student Gymnázia Ch. Dopplera v Praze 5). Vzhledem k podzimnímu termínu konání soutěže tři z našich reprezentantů již ukončili své studium na gymnáziu a v současné době jsou Jakub Bystroň a Pavel Charvát posluchači 1. ročníku oboru informatika na Matematicko-fyzikální fakultě Univerzity Karlovy v Praze, Ondřej Rucký nyní studuje v 1. ročníku na Západočeské univerzitě v Plzni a pouze Jiří Svoboda je nadále studentem gymnázia. Vedením české delegace byl pověřen místopředseda Ústředního výboru Matematické olympiády pro kategorii P doc. RNDr. *Pavel Töpfer*, CSc., z Matematicko-fyzikální fakulty Univerzity Karlovy v Praze, druhým vedoucím byl *Daniel Kráľ* z téže fakulty.

V rámci přípravy na soutěž se naši studenti zúčastnili na konci srpna 2000 týdenního přípravného soustředění v Polsku, které bylo společné pro týmy vybrané na IOI ze Slovenska, z Polska a z České republiky. Soustředění organizačně a finančně zajistili kolegové pečující o programátorskou olympiádu v Polsku, na odborném programu se podíleli rovným dílem vedoucí ze všech tří zúčastněných zemí. Naši reprezentanti nebyli letos tak úspěšní, jako tomu bylo dosud snad ve všech předchozích ročnících této soutěže. Dosažené výsledky však přesto stále patří v celosvětovém měřítku mezi nadprůměrné. Pavel Charvát, Jakub Bystroň a Jiří Svo-

boda získali bronzové medaile, na Ondřeje Ruckého tentokrát medaile nezbyla.

Výsledky našich studentů:

79.	Pavel Charvát	380 bodů	bronzová
102.	Jakub Bystroň	320 bodů	bronzová
108.	Jiří Svoboda	300 bodů	bronzová
	Ondřej Rucký	210 bodů	–

Texty soutěžních úloh

1. Parkoviště

Parkoviště u Velké zdi je tvořeno dlouhou řadou parkovacích míst. Jeden konec řady budeme považovat za její levý konec a druhý za její pravý konec. Všechna parkovací místa jsou obsazena. Každý zaparkovaný vůz je určitého typu, několik různých vozů může být stejného typu. Typy vozů jsou označeny celými čísly. Několik nudících se dělníků se rozhodlo přeuspořádat zaparkované vozy podle jejich typů, a to v rostoucím pořadí zleva doprava. Dělníci chtějí použít následující postup složený z několika po sobě následujících kroků: v každém kroku vyjedou současně několika vozy z jejich parkovacích míst a zaparkují je na místa, která se takto uvolnila. Každý dělník může přeparkovat v jednom kroku nejvýše jeden vůz. V jednom kroku lze tedy přeparkovat nejvýše tolik vozů, kolik je dělníků. Aby se příliš nenadřeli, chtějí dělníci svůj záměr uskutečnit během co nejmenšího počtu kroků.

Nechť N je počet vozů a W je počet dělníků. Vaším úkolem je vytvořit program, který na vstupu obdrží popis typů zaparkovaných vozů a nalezne způsob, jak vozy přeparkovat během nejvýše $\lceil N/(W-1) \rceil$ kroků, kde $\lceil N/(W-1) \rceil$ je číslo $N/(W-1)$ zaokrouhlené nahoru na nejbližší celé číslo. Počet kroků, které jsou potřeba k přeparkování vozů, je vždy nejvýše $\lceil N/(W-1) \rceil$.

Podívejme se na následující příklad: Na parkovišti je 10 vozů; jejich typy jsou označeny čísly 1, 2, 3 a 4. Čtyři dělníci chtějí tyto vozy přeparkovat. Původní rozmístění vozů je následující:

2 3 3 4 4 2 1 1 3 1

Minimální možný počet kroků je tři a rozmístění vozů po jednotlivých krocích může být například následující:

2 1 1 4 4 2 3 3 3 1 — po prvním kroku,
2 1 1 2 4 3 3 3 4 1 — po druhém kroku,
1 1 1 2 2 3 3 3 4 4 — po třetím kroku.

Vstup: Vstupní soubor se jmenuje `CAR.IN`. Jeho první řádek obsahuje tři celá čísla. První z nich, N, je počet vozů na parkovišti ($2 \leqq N \leqq$ $\leqq 20\,000$) a druhé z nich, M, je počet typů vozů ($2 \leqq M \leqq 50$). Jednotlivé typy vozů jsou označeny čísly 1 až M. Mezi zaparkovanými vozy se nachází od každého z těchto typů alespoň jeden. Třetí číslo, W, určuje počet dělníků, kteří chtějí vozy přeparkovat ($2 \leqq W \leqq M$). Na druhém řádku se nachází N čísel, která určují počáteční rozmístění vozů — i-té číslo představuje typ i-tého vozu v řadě počítáno zleva doprava.

Výstup: Výstupní soubor se jmenuje `CAR.OUT`. První řádek výstupního souboru obsahuje jedno celé číslo R, představující počet kroků nalezeného řešení úlohy. Následujících R řádků popisuje po řadě jednotlivé kroky. První číslo, C, na každém z těchto řádků určuje počet vozů přeparkovávaných v příslušném kroku. Za tímto číslem následuje dalších $2C$ celých čísel, která popisují pozice vozů. Parkovací místa jsou očíslována zleva doprava od 1 do N. První dvojice těchto čísel popisuje přesun jednoho z vozů: první číslo z dvojice udává pozici vozu před tímto krokem a druhé po tomto kroku. Další dvojice čísel určuje, jak bude přeparkován další z vozů, atd. Pokud existuje více možných řešení, vaším úkolem je nalézt a vypsat jedno libovolné z nich.

Příklady vstupů a výstupů:

`CAR.IN`	`CAR.OUT`
`10 4 4`	`3`
`2 3 3 4 4 2 1 1 3 1`	`4 2 7 3 8 7 2 8 3`
	`3 4 9 9 6 6 4`
	`3 1 5 5 10 10 1`

Hodnocení. Předpokládejme, že váš program nalezl způsob, jak vozy přeparkovat v R krocích. Označme si $\lceil N/(W-1) \rceil$ jako Q. Jestliže nalezený postup přesunu vozů nesplňuje zadání úlohy, je příslušný test hodnocen 0 body. Jinak je ohodnocen body podle následujících pravidel:

$R \leqq Q$	100 % bodů
$R = Q + 1$	50 % bodů
$R = Q + 2$	20 % bodů
$R \geqq Q + 3$	0 % bodů

2. Palindromy

Palindromem nazýváme takový znakový řetězec, který je symetrický, tj. který se čte stejně zleva i zprava. Vytvořte program, jenž pro zadaný řetězec určí minimální počet znaků, které je nutné do tohoto řetězce vložit tak, aby se z něj stal palindrom.

Například vložením dvou znaků do řetězce „`Ab3bd`" můžeme získat palindrom „`dAb3bAd`" nebo „`Adb3bdA`". Vložením méně než dvou znaků však z tohoto řetězce palindrom nelze vytvořit.

Vstup: Vstupní soubor se jmenuje `PALIN.IN`. První řádek vstupního souboru obsahuje jedno celé číslo N, délku vstupního řetězce, $3 \leqq N \leqq 5\,000$. Druhý řádek obsahuje vstupní řetězec. Řetězec se skládá z velkých písmen od »`A`« do »`Z`«, z malých písmen od »`a`« do »`z`« a z číslic od »`0`« do »`9`«. Velká a malá písmena v řetězci představují odlišné znaky.

Výstup: Výstupní soubor se jmenuje `PALIN.OUT`. První řádek obsahuje jedno číslo — nalezený minimální počet znaků, které je do řetězce třeba vložit.

Příklady vstupů a výstupů:

`PALIN.IN`	`PALIN.OUT`
`5` `Ab3b`	`2`

3. Střední tvrdost

V rámci jednoho z výzkumných projektů je třeba otestovat N vzorků materiálů. Vzorky jsou označeny čísly od 1 do N, kde N je liché číslo. Tvrdost každého ze vzorků lze vyjádřit celým číslem Y ($1 \leqq Y \leqq N$). Žádné dva vzorky nemají stejnou tvrdost. Vzorek X je vzorek se střední tvrdostí, jestliže stejný počet vzorků má tvrdost menší než vzorek X a stejný počet vzorků má tvrdost větší. Vaším úkolem je vytvořit program, který určí vzorek se střední tvrdostí.

Jediný přístroj, který je v laboratoři k dispozici, je schopný určit ze tří předložených vzorků ten, který má mezi těmito vzorky střední tvrdost.

Popis knihovny. Obdrželi jste knihovnu pojmenovanou `device`, která obsahuje následující tři rutiny:

- Rutina `GetN` musí být zavolána právě jednou, a to na začátku výpočtu; volá se bez parametrů a její návratovou hodnotou je číslo N.
- Rutina `Med3` je volána se třemi parametry, jimiž jsou čísla tří navzájem různých vzorků vložených do přístroje. Její návratová hodnota je číslo vzorku, který má střední tvrdost mezi předloženými vzorky.

▷ Rutina `Answer` musí být zavolána právě jednou, a to na konci výpočtu; jejím jediným parametrem je vaším programem určené číslo vzorku X, tj. číslo vzorku se střední tvrdostí. Zavolání této rutiny zároveň korektně ukončí výpočet programu.

Knihovna `device` vytvoří dva soubory: `MEDIAN.OUT` a `MEDIAN.LOG`. První řádek souboru `MEDIAN.OUT` obsahuje jedno číslo, které je rovno číslu vzorku předanému vaším programem rutině `Answer`. Druhý řádek obsahuje jedno celé číslo představující počet volání rutiny `Med3` vaším programem. Komunikace mezi vaším programem a knihovnou je zaznamenána v souboru `MEDIAN.LOG`.

Pokyny pro programátory v jazyce Pascal: Vložte do vašeho programu následující řádek:

```
uses device;
```

Pokyny pro programátory v jazyce C/C++: Vložte do vašeho programu následující řádek:

```
#include "device.h"
```

Dále vytvořte projekt `MEDIAN.PRJ` a vložte do tohoto projektu soubory `MEDIAN.C` (`MEDIAN.CPP`) a `DEVICE.OBJ`.

Ladění. Komunikaci programu s poskytnutou knihovnou lze ladit následujícím způsobem: Vytvoříte textový soubor `DEVICE.IN`. Tento soubor by měl obsahovat dva řádky. Na prvním z nich bude uvedeno jedno celé číslo — počet vzorků N. Druhý řádek obsahuje N celých čísel od 1 do N ve vámi zvoleném pořadí; i-té z nich představuje tvrdost i-tého vzorku.

Příklad vstupu: `DEVICE.IN`

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

Tento soubor `DEVICE.IN` obsahuje popis následujících pěti vzorků:

Vzorek	1	2	3	4	5
Tvrdost	2	5	4	3	1

Korektní posloupnost komunikace vašeho programu s knihovnou by mohla vypadat následovně:

1. `GetN` (v Pascalu) nebo `GetN()` (v C/C++) — návratová hodnota je 5.
2. `Med3(1,2,3)` — návratová hodnota je 3.
3. `Med3(3,4,1)` — návratová hodnota je 4.
4. `Med3(4,2,5)` — návratová hodnota je 4.
5. `Answer(4)`

Omezení specifická pro tuto úlohu

- ▷ Počet vzorků N splňuje tyto podmínky: $5 \leqq N \leqq 1\,499$ a N je liché.
- ▷ Číslo vzorku i splňuje $1 \leqq i \leqq N$.
- ▷ Tvrdost vzorku Y splňuje $1 \leqq Y \leqq N$; všechny vzorky mají navzájem různou tvrdost.
- ▷ Jméno knihovny v Pascalu: `device.tpu`
- ▷ Deklarace knihovních funkcí a procedur v Pascalu:

```
function GetN: integer;
function Med3(x,y,z:integer):integer;
procedure Answer(m:integer);
```

- ▷ Jméno knihovny v C/C++: `device.h`, `device.obj` (při kompilaci použijte LARGE memory model)
- ▷ Deklarace funkcí v C/C++:

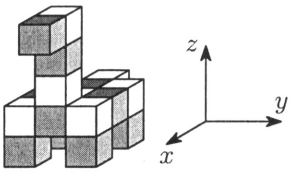
```
int GetN(void);
int Med3(int x, int y, int z);
void Answer(int m);
```

- ▷ Váš program smí provést nejvýše 7 777 volání funkce `Med3` během jednoho testu.
- ▷ Váš program nesmí číst ani zapisovat do žádných souborů.

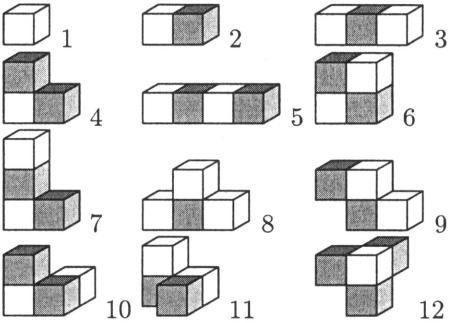
4. Stavby z kostek

Jednotková krychle je krychle o rozměrech $1 \times 1 \times 1$ taková, že všechny její rohy mají celočíselné souřadnice x, y a z. Dvě jednotkové krychle se dotýkají, pokud mají společnou jednu ze svých stěn. Dvě krychle jsou spojeny, pokud existuje posloupnost jednotkových krychlí taková, že každé dvě po sobě následující krychle se dotýkají a uvažované krychle jsou první a poslední v této posloupnosti. Stavbou z kostek rozumíme libovolnou množinu jednotkových krychlí takovou, že každé dvě jednotkové krychle v této množině jsou spojené (obr. 72). Objemem stavby z kostek nazýváme počet jednotkových krychlí, které tato stavba obsahuje. Kostkou rozumíme množinu nejvýše čtyř navzájem spojených jednotkových krychlí. Existuje 12 různých druhů kostek, které jsou znázorněny na obr. 73. Kostky kteréhokoliv druhu mohou být použity ve stavbě v libovolném množství a to jakkoliv posunuté a natočené (ne však zrcadlově převrácené). Barevné odstíny na obou obrázcích jsou použity pouze ke zvýšení jejich přehlednosti a nemají žádný další význam.

Množinu D kostek lze použít pro vytvoření stavby S, pokud lze všechny kostky z množiny D umístit a natočit tak, aby vytvořily přesně stavbu S a přitom byly navzájem disjunktní (mohou se však dotýkat).



Obr. 72. Kůň z kostek



Obr. 73. 12 různých druhů kostek

Vytvořte program, který obdrží na vstupu popis všech 12 druhů kostek a stavby S. Program poté určí nejmenší možný počet kostek, které jsou potřeba k vytvoření stavby S, a dále určí, jaké kostky budou ve stavbě použity.

Vstup: Ve vstupních souborech se na všechny jednotkové krychle odkazujeme pomocí trojice souřadnic x, y a z toho jejich rohu, pro který je $x + y + z$ nejmenší.

Vstupní soubor, který popisuje jednotlivé druhy kostek, se jmenuje `TYPES.IN`. Obsah tohoto souboru je stejný pro všechny testy. Obsahuje popis 12 druhů kostek; všechny tyto kostky jsou znázorněny na obr. 73. Popis druhů kostek obsažený v tomto souboru je uspořádán podle čísel druhů jednotlivých kostek. Každý druh kostek je popsán na několika řádcích. První řádek popisu jednoho druhu kostky obsahuje jediné celé číslo I, které je jeho identifikátorem ($1 \leqq I \leqq 12$). Druhý řádek obsahuje jediné celé číslo V, které představuje objem kostky popisovaného druhu ($1 \leqq V \leqq 4$). Každý z následujících V řádků pak obsahuje trojici celých čísel x, y, z, která určují polohu jednotlivých jednotkových krychlí tvořících kostku ($1 \leqq x, y, z \leqq 4$).

Vstupní soubor, který popisuje zadanou stavbu z kostek, se jmenuje `BLOCK.IN`. Jeho první řádek obsahuje jediné celé číslo V, jež představuje celkový objem stavby ($1 \leqq V \leqq 50$). Každý z následujících V řádků pak obsahuje trojici celých čísel x, y, z, která určují polohu jednotlivých jednotkových krychlí tvořících stavbu ($1 \leqq x, y, z \leqq 7$).

Výstup: Výstupní soubor se jmenuje `BLOCK.OUT`. První řádek obsahuje jediné celé číslo M, které představuje minimální počet kostek, které jsou potřeba pro vytvoření zadané stavby. Druhý řádek obsahuje M iden-

tifikátorů druhů kostek, ze kterých je možné stavbu sestavit. Pokud existuje více optimálních řešení, váš program vypíše právě jedno z nich.

Příklady vstupů a výstupů:

TYPES.IN

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

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

```
10
4
2 1 1
1 2 1
2 2 1
2 1 2
11
4
1 1 1
1 2 1
2 2 1
1 1 2
12
4
2 2 1
2 1 2
1 2 2
2 2 2
```

BLOCK.IN

```
18
2 1 1
4 1 1
2 3 1
4 3 1
2 1 2
3 1 2
4 1 2
1 2 2
2 2 2
3 2 2
4 2 2
2 3 2
3 3 2
4 3 2
4 2 3
4 2 4
4 2 5
5 2 5
```

BLOCK.OUT

```
5
7 10 2 10 12
```

Poznámka.

1. Vstupní soubor BLOCK.IN popisuje stavbu koně z obr. 72.
2. Jiná přípustná řešení mohou mít ve výstupním souboru na druhém řádku následující identifikátory druhů kostek:

```
2 7 10 11 12
2 7 11 11 12
4 4  7 10 11
4 4  9 10 11
```

5. Poštovní úřady

V jednom nejmenovaném státě se nachází jedna dlouhá rovná dálnice s několika vesnicemi podél ní. Do každé vesnice vede jeden výjezd z dálnice; všechny tyto výjezdy jsou očíslovány celými kladnými čísly. Číslo výjezdu je rovno jeho vzdálenosti od začátku dálnice. U žádného z výjezdů neleží dvě různé vesnice. Vzdálenost mezi dvěma vesnicemi je rovna absolutní hodnotě rozdílu čísel výjezdů, které vedou do těchto vesnic (zanedbáváme tedy vzdálenost vesnic od dálnice).

V některých z těchto vesnic (ne nutně ve všech) mají být zřízeny poštovní úřady. Ředitelství pošt chce zřídit poštovní úřady tak, aby součet dojezdových vzdáleností všech vesnic byl co nejmenší. Dojezdovou vzdáleností vesnice rozumíme její vzdálenost od poštovního úřadu, který je k ní nejbližší.

Vytvořte program, který obdrží na vstupu zadána čísla jednotlivých výjezdů, u kterých leží vesnice, a počet poštovních úřadů, které plánuje ředitelství pošt zřídit. Program poté určí minimální možný součet dojezdových vzdáleností všech vesnic a nalezne rozmístění poštovních úřadů, které dosahuje tohoto součtu.

Vstup: Vstupní soubor se jmenuje `POST.IN`. Jeho první řádek obsahuje dvě celá kladná čísla: první z nich, V, představuje počet vesnic podél dálnice ($1 \leqq V \leqq 300$) a druhé z nich, P, představuje počet poštovních úřadů, které se plánují vybudovat ($1 \leqq P \leqq 30$, $P \leqq V$). Druhý řádek obsahuje rostoucí posloupnost V celých kladných čísel. Těchto V čísel představuje čísla výjezdů z dálnice u jednotlivých vesnic. Pro každé z těchto čísel, označme ho X, platí následující omezení: $1 \leqq X \leqq 10\,000$.

Výstup: Výstupní soubor se jmenuje `POST.OUT`. První řádek obsahuje jediné celé číslo, které představuje součet dojezdových vzdáleností všech vesnic pro vámi nalezené řešení. Druhý řádek obsahuje rostoucí posloupnost P kladných celých čísel. Tato čísla jsou čísla výjezdů, u kterých navrhujete zřídit poštovní úřad. Pokud existuje více optimálních řešení, váš program vypíše právě jedno z nich.

Příklad vstupu:

`POST.IN`	`POST.OUT`
`10 5`	`9`
`1 2 3 6 7 9 11 22 44 50`	`2 7 22 44 50`

Hodnocení. Pokud výstupní soubor nemá formát odpovídající zadání nebo pokud součet uvedený na prvním řádku neodpovídá řešení uvede-

ném na druhém řádku, váš program bude v příslušném testu hodnocen 0 body. V opačném případě se počet vámi získaných bodů spočte podle níže uvedené tabulky následujícím postupem: Označme S součet dojezdových vzdáleností všech vesnic pro vámi nalezené řešení a $S_{\min}$ tento součet pro optimální řešení; váš program bude hodnocen tolika body, kolik je uvedeno pro odpovídajíc poměr $q = S/S_{\min}$ v druhém řádku tabulky.

$q = S/S_{\min}$	Počet bodů
$q = 1{,}0$	10
$1{,}0 < q \leqq 1{,}1$	5
$1{,}1 < q \leqq 1{,}15$	4
$1{,}15 < q \leqq 1{,}2$	3
$1{,}2 < q \leqq 1{,}25$	2
$1{,}25 < q \leqq 1{,}3$	1
$1{,}3 < q$	0

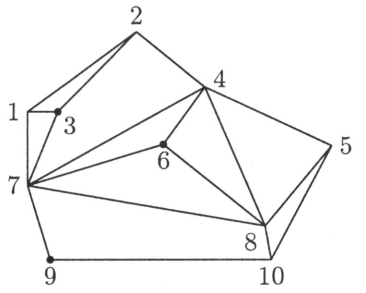
6. Velké zdi

V jednom nejmenovaném státě bylo zbudováno několik „Velkých zdí". Každá z těchto zdí spojuje právě dvě města, žádné dvě zdi se nekříží a mezi každými dvěma městy vede nejvýše jedna zeď. Stát je takto rozdělen na několik oblastí takových, že k přesunu z jedné oblasti do druhé je třeba projít některým z měst nebo přelézt zeď.

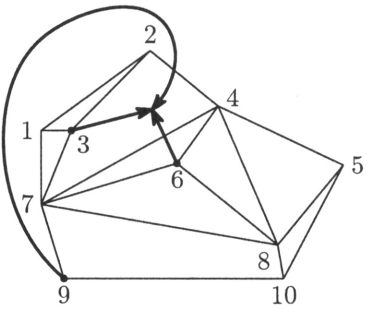
Demonstranti proti ekonomické globalizaci žijí v různých městech, v každém městě však žije nejvýše jeden z demonstrantů. Občas chtějí demonstranti uspořádat party nebo demonstraci. Za tímto účelem se potřebují všichni sejít v některé z oblastí ohraničených zdmi; z pochopitelných důvodů se však nemohou sejít v žádném z měst. Při své cestě na kolech na místo srazu nechtějí projet žádným městem, aby je nezadržela policie, a navíc chtějí přelézat co nejmenší počet zdí (s kolem to není zrovna nejsnazší).

Během své cesty na místo srazu musí každý z demonstrantů přelézt několik zdí (možná i žádnou). Demonstranti proto chtějí zvolit takovou oblast pro setkání, aby součet počtů přelezení zdí uskutečněných všemi demonstranty byl co nejmenší.

Města jsou označena celými čísly od 1 do N, kde N je počet všech měst. Očíslované uzly na obr. 74 představují města a úsečky spojující uzly odpovídají zdem. Předpokládejme, že ve státě žijí tři demonstranti a ti bydlí ve městech 3, 6 a 9. Nejvýhodnější oblast pro jejich setkání a příslušné trasy pro jednotlivé demonstranty jsou vyznačeny na obr. 75.



Obr. 74



Obr. 75

Demonstranti budou muset přelézt dvě zdi: demonstrant z města 9 musí přelézt zeď vedoucí mezi městy 2 a 4, demonstrant z města 6 musí přelézt zeď vedoucí mezi městy 4 a 7.

Vytvořte program, který obdrží na vstupu počet měst, popis oblastí ohraničených zdmi a seznam měst, kde bydlí demonstranti. Program poté určí optimální oblast pro setkání demonstrantů a spočítá minimální počet přelezení zdí uskutečněných dohromady všemi demonstranty při jejich cestě na místo srazu.

Vstup: Vstupní soubor se jmenuje `WALLS.IN`. Jeho první řádek obsahuje jedno celá kladné číslo M — počet oblastí ($2 \leqq M \leqq 200$). Druhý řádek obsahuje jedno celé kladné číslo N úpočet měst ($3 \leqq N \leqq 250$). Třetí řádek obsahuje jedno celé kladné číslo L — počet demonstrantů ($1 \leqq L \leqq 30$, $L \leqq N$). Čtvrtý řádek vstupního souboru obsahuje rostoucí posloupnost L různých celých kladných čísel — seznam čísel měst, v nichž žijí demonstranti.

Poté ve vstupním souboru následuje $2M$ řádků. Každá dvojice po sobě následujících řádků popisuje jednu z oblastí, tzn. první dva z těchto řádků popisují první oblast, následující dva řádky popisují druhou oblast, atd. První řádek v každé dvojici udává počet měst, I, která leží na hranici příslušné oblasti. Druhý řádek dvojice pak obsahuje posloupnost I celých kladných čísel, která jsou čísly měst ležících na hranici příslušné oblasti a která jsou zde uvedena seřazená po směru hodinových ručiček (začátek této posloupnosti může být zvolen libovolně). Poslední oblast uvedená v souboru je vnější oblast; v popisu této oblasti jsou města na její hranici uspořádána proti směru hodinových ručiček (to je jediná výjimka proti pravidlu o pořadí výpisu měst na hranici oblasti). Pořadí, v jakém jsou jednotlivé oblasti uvedeny ve vstupním souboru, určuje jejich očíslování,

tzn. oblast uvedená jako první má číslo 1, oblast uvedená jako druhá má číslo 2 atd. Povšimněte si, že ve vstupním souboru je uveden popis všech oblastí včetně „vnější" oblasti.

Výstup: Výstupní soubor se jmenuje `WALLS.OUT`. První řádek obsahuje jediné celé číslo, které představuje minimální celkový počet přelezení zdí na cestě demonstrantů na místo srazu. Druhý řádek obsahuje jediné celé číslo, kterým je pořadové číslo oblasti vybrané k uskuteční srazu demonstrantů. Pokud existuje více optimálních řešení, váš program vypíše právě jedno z nich.

Příklad vstupu: Následující vstupní a výstupní soubor odpovídají příkladu uvedenému v zadání.

```
WALLS.IN
10
10
3
3 6 9
3
1 2 3
3
1 3 7
4
2 4 7 3
3
4 6 7
```

```
3
4 8 6
3
6 8 7
3
4 5 8
4
7 8 10 9
3
5 10 8
7
7 9 10 5 4 2 1
```

```
WALLS.OUT
2
3
```