15. mezinárodní olympiáda v informatice

In: 52. ročník matematické olympiády na středních školách. Zpráva o řešení úloh ze soutěže konané ve školním roce 2002/2003. 44. mezinárodní matematická olympiáda. 15. mezinárodní olympiáda v informatice. (Czech). Praha: Jednota českých matematiků a fyziků, 2004. pp. 173–186.

Persistent URL: http://dml.cz/dmlcz/405066

15. mezinárodní olympiáda v informatice

Hostitelem 15. mezinárodní olympiády v informatice IOI 2003 byly Spojené státy americké. Soutěž se uskutečnila ve dnech 16.–23. 8. 2003 v univerzitním kampusu University of Wisconsin-Parkside nedaleko města Kenosha. Organizátoři zvolili termín konání v době univerzitních prázdnin, takže pro potřeby olympiády mohli využít dočasně vyprázdněné studentské ubytovny a jídelny, univerzitní posluchárny pro vlastní soutěž i pro různá probíhající jednání, aulu pro slavnostní zahájení a zakončení soutěže, ale také rozsáhlá sportoviště pro aktivní využití volného času všech účastníků.

Olympiáda byla výborně připravena po stránce organizační, po stránce počítačového vybavení i z hlediska kvality přípravy soutěžních úloh. V každém ze dvou soutěžních dnů řešili soutěžící studenti u počítačů tři poměrně náročné příklady. Se svými počítači i jejich softwarovým vybavením se přitom všichni mohli podrobně seznámit den před vlastní soutěží, kdy probíhalo neoficiální tréninkové předkolo. Při soutěži bylo možné programovat v některém z programovacích jazyků Pascal, C nebo C++, každý si mohl zvolit podle svých předchozích zkušeností pracovní prostředí operačních systémů Windows nebo Linux. O oba uvedené systémy byl mezi účastníky přibližně stejný zájem.

K testování a hodnocení vytvořených programů se na IOI již řadu let používá automatické vyhodnocování pomocí připravené sady vstupních dat. Všechny prováděné testy mají dobu výpočtu omezenu předem známým časovým limitem a jednotlivá testovací vstupní data mají různou velikost a různou složitost, což dohromady umožňuje bodově rozlišit programy podle kvality použitého algoritmu. Za každou úlohu lze získat maximálně 100 bodů, nejčastěji bývá při hodnocení zadáno 20 sad testovacích dat 5 bodech. U některých soutěžních úloh se hodnotí také dosažení optimality nalezeného řešení. Za nalezení nejlepšího možného řešení soutěžící dostane pro daná testovací data plný počet bodů, zatímco o něco horší řešení je hodnoceno méně body podle předem známého klíče (v hodnocení některých úloh se tak objevují i desetiny bodů).

Patnácté mezinárodní olympiády v informatice se zúčastnilo 265 soutěžících studentů ze 69 zemí celého světa. Počtem zúčastněných zemí tak IOI již dostihla ostatní mezinárodní předmětové olympiády středoškoláků a její velikost a význam rok od roku stále narůstá. Českou republiku reprezentovalo čtyřčlenné družstvo ve složení *Pavel Čížek* (absolvent Dvořákova gymnázia v Kralupech nad Vltavou), *Tomáš Gavenčiak* (student gymnázia M. Koperníka v Bílovci), *Jan Kadlec* (absolvent gymnázia Ch. Dopplera v Praze 5) a *Milan Straka* (absolvent gymnázia ve Strakonicích). Naši soutěžící byli vybráni na základě výsledků dosažených v celostátním kole kategorie P (programování) 52. ročníku Matematické olympiády. Vedením družstva byli pověřeni doc. *Pavel Töpfer* a RNDr. *Daniel Kráľ*, oba z Matematicko-fyzikální fakulty Univerzity Karlovy v Praze.

Na mezinárodní olympiádě v informatice se udělují ocenění podle podobného klíče, jaký se používá například také na mezinárodní matematické olympiádě. Nejvýše polovina soutěžících obdrží některou z medailí, přičemž zlaté, stříbrné a bronzové medaile se dělí přibližně v poměru 1 : 2 : 3. Na letošní IOI 2004 bylo rozděleno celkem 132 medailí, z toho 24 zlatých, 45 stříbrných a 63 bronzových. O velké vyrovnanosti světové špičky svědčí skutečnost, že o udělených 24 zlatých medailí se podělili reprezentanti z 19 zemí, mezi nimi i naši. Pouze pět letos nejúspěšnějších států získalo po dvou zlatých medailích (Korea, USA, Rumunsko, Švédsko a Polsko), žádná země nemá tři zlaté medaile.

Naši studenti si vedli v soutěži velmi dobře, získali jednu zlatou, jednu stříbrnou a jednu bronzovou medaili. Podobného výsledku, tj. jedné zlaté, jedné stříbrné a jedné bronzové medaile, dosáhla také reprezentace Slovenska. Mezinárodní olympiáda v informatice je podle svých stanov soutěží jednotlivců, žádné oficiální pořadí zúčastněných zemí není vyhlašováno a není ani stanoveno, podle jakého kritéria by se mělo takové pořadí určovat (zda podle počtu získaných medailí, součtu bodů všech reprezentantů příslušné země či například podle součtu pořadí ve výsledkové listině). Při jakémkoliv způsobu počítání by se naše výprava umístila kolem 12.–13. místa, což je velmi dobrý výsledek. Následující tabulka shrnuje výsledky všech českých studentů v soutěži:

14.	Milan Straka	375,5 bodů	zlatá
43.	Pavel Čížek	306,1 bodů	stříbrná
111.	Tomáš Gavenčiak	195,0 bodů	bronzová
146.	Jan Kadlec	162,5 bodů	–

Příští, v pořadí šestnáctá mezinárodní olympiáda v informatice IOI 2004 se uskuteční v Athénách v první polovině září 2004. Pořadatelé z Řecka již nyní pozvali všechny země zúčastněné na IOI 2003, aby se zúčastnily i příštího ročníku soutěže. V roce 2004 se bude konat také 11. ročník Středoevropské olympiády v informatice CEOI 2004, a to v první polovině července ve Varšavě. Rovněž od polských pořadatelů jsme obdrželi pozvání k účasti.

Texty soutěžních úloh

1. Wisconsinské krávy (interaktivní úloha)

Krávy farmáře Johna se volně pohybují mezi N ($1 \leqq N \leqq 200$) pastvinami, které jsou očíslovány od 1 do N. Pastviny jsou navzájem odděleny lesem. Krávy udržují systém cest mezi pastvinami tak, aby kdykoliv bylo možné přejít po udržovaných cestách mezi libovolnými dvěma pastvinami. Po všech cestách je možné chodit oběma směry.

Krávy ve skutečnosti cesty samy nevytvářejí. Místo toho používají stezky lesní zvěře. Pro každý týden si vyberou některé ze stezek, které znají, a ty pak udržují jako cesty mezi pastvinami.

Krávy jsou zvířata od přírody velmi zvědavá. Na začátku každého týdne objeví jednu novou stezku. Ze stezek, které znaly z dřívějška, a z nově objevené stezky, pak vyberou množinu stezek, které budou v nadcházejícím týdnu udržovat jako cesty. Výběry stezek pro jednotlivé týdny jsou navzájem zcela nezávislé.

Stezku, která je udržována, mohou krávy ihned používat jako cestu mezi pastvinami. Pokud stezka přestane být udržována, není ji již nadále možné používat jako cestu. Krávy chtějí udržovat vždy takovou soustavu cest, aby součet délek udržovaných cest byl nejmenší možný.

Stezky lesní zvěře bývají klikaté. Proto může existovat více stezek různých délek, které spojují stejnou dvojici pastvin. I když se dvě cesty v lese kříží, krávy vždy pokračují v chůzi po původně zvolené cestě.

Vaším úkolem je pro každý týden určit nejmenší možný součet délek udržovaných cest. Váš program bude postupně dostávat informace o nových stezkách objevených krávami. Po načtení popisu každé nové stezky musí váš program vypsat optimální součet délek udržovaných cest.

Vstup: standardní vstup (standard input)

▷ První řádek obsahuje dvě celá čísla N a W oddělená jednou mezerou. N určuje počet pastvin ($1 \leqq N \leqq 200$) a W udává počet týdnů, v nichž činnost krav sledujeme ($1 \leqq W \leqq 6\,000$).

▷ Pro každý týden pak následuje samostatný řádek, který popisuje nově objevenou stezku. Tento řádek je tvořen třemi celými čísly oddělenými jednou mezerou, která udávají čísla pastvin spojených novou stezkou a její délku (1 ... 10 000). Každá stezka spojuje dvě různé pastviny.

Výstup: standardní výstup (standard output)

Popis další stezky nelze načíst, dokud váš program nevypíše řešení pro současnou množinu stezek. Pro každý týden, váš program vypíše jedno celé číslo na samostatném řádku: Toto číslo udává nejmenší možný součet délek cest, které je třeba udržovat, aby byla zachována propojenost všech pastvin. Pokud takový systém cest neexistuje, program vypíše číslo `-1`.

Program musí skončit po vypsání řešení pro poslední sledovaný týden.

Příklad komunikace:

Vstup	*Výstup*	*Vysvětlení*
4 6		
1 2 10		
	-1	Žádná stezka nespojuje pastvinu č. 4 s ostatními pastvinami.
1 3 8		
	-1	Žádná stezka nespojuje pastvinu č. 4 s ostatními pastvinami.
3 2 3		
	-1	Žádná stezka nespojuje pastvinu č. 4 s ostatními pastvinami.
1 4 3		
	14	Systém cest je tvořen stezkami 1 4 3, 1 3 8 a 3 2 3.
1 3 6		
	12	Systém cest je tvořen stezkami 1 4 3, 1 3 6 a 3 2 3.
2 1 2		
	8	Systém cest je tvořen stezkami 1 4 3, 2 1 2 a 3 2 3.
konec programu		

Omezení:

Časový limit	1 s CPU
Paměťový limit	64 MB

Hodnocení. Za každý testovací vstup obdržíte plný počet bodů, pokud váš program vypíše správný výstup. V opačném případě je testovací vstup hodnocen 0 body.

2. Ukradený kód

Společnost Racine Business Networks (RBN) se rozhodla zažalovat společnost Heuristic Algorithm Languages (HAL). RBN tvrdí, že HAL

vykradla část jejího zdrojového kódu RBN UNIX™ a začlenila ho do svého operačního systému HALnix.

Obě společnosti RBN a HAL používají stejný programovací jazyk. Každá instrukce je uvedena na samostatném řádku a všechny mají jednotný formát: `STOREA = STOREB + STOREC`, kde `STOREA`, `STOREB` a `STOREC` jsou jména proměnných. Jméno první proměnné začíná v prvním sloupci, pak následuje jedna mezera, rovnítko a další mezera. Poté je uvedeno jméno druhé proměnné následované jednou mezerou, plusem a další mezerou. Řádek je ukončen jménem třetí proměnné. Jedna proměnná se může na témže řádku vyskytovat několikrát. Jména proměnných jsou tvořena 1 až 8 velkými písmeny anglické abecedy (A, ..., Z).

RBN tvrdí, že programátoři společnosti HAL okopírovali souvislé kusy jejího kódu a provedli pouze následující změny, aby zamaskovali svůj odporný zločin: Programátoři HAL vždy vzali několik po sobě následujících řádků z kódu firmy RBN a v něm změnili jména některých proměnných. Nikdy se však nestalo, že by dvě různé proměnné měly po přejmenování totéž jméno. Programátoři HAL také občas zaměnili pořadí sčítanců na pravé straně instrukce. Tedy instrukci `STOREA = STOREB + STOREC` nahradili instrukcí `STOREA = STOREC + STOREB`. Pořadí jednotlivých instrukcí však zůstalo zachováno.

Vaším úkolem je v programu společnosti HAL najít nejdelší souvislý kus kódu, který mohl být vykraden z programu společnosti RBN výše popsaným způsobem. Odpovídající si kusy kódu mohou v každém z programů začínat na různých řádcích.

Vstupní soubor: `code.in`

- První řádek obsahuje dvě celá čísla R a H oddělená jednou mezerou ($1 \leqq R \leqq 1\,000$; $1 \leqq H \leqq 1\,000$). R udává počet řádků programu společnosti RBN a H programu společnosti HAL.
- Následujících R řádků obsahuje kód programu společnosti RBN.
- Dalších H řádků pak obsauje kód programu společnosti HAL.

Příklad vstupního souboru:

```
4 3
RA = RB + RC
RC = D + RE
RF = RF + RJ
RE = RF + RF
HD = HE + HF
HM = HN + D
HN = HA + HB
```

Výstupní soubor: `code.out`

Výstupní soubor musí být tvořen jediným řádkem obsahujícím jedno celé číslo. Toto číslo udává počet řádků nejdelšího souvislého kusu kódu, který mohl být vykraden z programu společnosti RBN.

Příklad výstupního souboru: 2

Řádky 1 a 2 programu společnosti RBN odpovídají řádkům 2 a 3 programu společnosti HAL (RAHM, RB → D, RC → HN, D → HA, RE → HB). Žádné tři po sobě jdoucí řádky programu společnosti RBN neodpovídají třem řádkům programu společnosti HAL.

Omezení:

Časový limit	2 s CPU
Paměťový limit	64 MB

Hodnocení. Za každý testovací vstup obdržíte plný počet bodů, pokud váš program vytvoří správný výstupní soubor. V opačném případě je testovací vstup hodnocen 0 body.

3. Posloupnost (open-data úloha)

Vaším úkolem je vytvořit program pro počítač TOM. Počítač TOM má 9 paměťových registrů, jejichž hodnoty lze nastavit na začátku výpočtu. Registry jsou očíslovány čísly od 1 do 9 a každý z nich může uchovávat jedno celé číslo z intervalu 0 ... 1000. Počítač má implementovány pouze následující dvě instrukce:

`S i j`	Do registru j přiřadí hodnotu registru i zvýšenou o 1. Čísla i a j mohou být stejná.
`P i`	Vytiskne na výstup hodnotu uloženou v registru i.

Program počítače TOM je tedy tvořen počátečním nastavením hodnot registrů a posloupností instrukcí. Vaším úkolem je vytvořit pro zadané celé číslo N ($0 \leqq N \leqq 255$) program, který vypíše posloupnost čísel $N, N-1, N-2, \ldots, 0$. Snažte se, aby počet po sobě následujících S-instrukcí ve výsledném programu byl nejmenší možný.

Příklad programu pro počítač TOM pro $N = 2$ a průběh jeho výpočtu:

Instrukce	Obsah registrů									Výstup
	1	2	3	4	5	6	7	8	9	
Počáteční hodnoty	0	2	0	0	0	0	0	0	0	
`P 2`	0	2	0	0	0	0	0	0	0	2
`S 1 3`	0	2	1	0	0	0	0	0	0	
`P 3`	0	2	1	0	0	0	0	0	0	1
`P 1`	0	2	1	0	0	0	0	0	0	0

Testovací data jsou očíslována od 1 do 16. Vstupní soubory, které je obsahují, si můžete stáhnout ze soutěžního serveru.

Formát vstupních souborů:

- První řádek obsahuje jedno celé číslo K, které udává pořadové číslo testovacích dat.
- Druhý řádek obsahuje číslo N.

Příklad vstupního souboru:

```
1
2
```

Formát výstupních souborů:

První řádek výstupního souboru musí obsahovat řetězec „`FILE reverse K`“, kde K je pořadové číslo testovacích dat.

Druhý řádek má obsahovat 9 celých čísel navzájem oddělených mezerami. Tato čísla představují počáteční hodnoty uložené v registrech (první v registru 1, druhé v registru v 2, atd.).

Zbytek výstupního souboru obsahuje kód programu pro počítač TOM. Každý řádek obsahuje právě jednu instrukci. Instrukce na posledním řádku programu by měla vytisknout na výstup číslo 0. Kód programu by měl být ve tvaru jako v následujících příkladech.

Příklad výstupu #1 (část bodů):

```
FILE reverse 1
0 2 0 0 0 0 0 0 0
P 2
S 1 3
P 3
P 1
```

Příklad výstupu #2 (plný počet bodů):

```
FILE reverse 1
0 2 1 0 0 0 0 0 0
P 2
P 3
P 1
```

Hodnocení. Počet bodů, které obdržíte za testovací data, závisí na správnosti a optimalitě odevzdaného řešení.

Správnost: 20 %

Program počítače TOM je korektní, pokud nevykoná více než 131 po sobě následujících S-instrukcí a zároveň na výstup vypíše postupně $N+1$ čísel v pořadí od N do 0. Pokud během výpočtu programu některá S-instrukce způsobí přetečení registru, pak je program považován za nekorektní.

Optimalita: 80 %

Úkolem je minimalizovat největší počet po sobě následujících S-instrukcí. Optimalita odevzdaného programu je porovnávána s nejlepším programem pro daný testovací vstup, který je k dispozici.

4. Není kráva jako kráva (interaktivní úloha)

Farmář John chce ustájit svých N ($1 \leqq N \leqq 50$) krav. Krávy jsou očíslovány celými čísly od 1 do N. Bohužel jsou si navzájem velmi podobné a není jednoduché rozlišit je mezi sebou. Protože je každá kráva zvyklá na své místo ve stáji, musí být John schopný rychle je rozeznávat.

Krávy lze rozeznávat podle P ($1 \leqq P \leqq 8$) rozlišovacích znaků, např. podle barvy visačky v uchu. Jednotlivé rozlišovací znaky si pro přehlednost očíslujeme čísly od 1 do P. Každý z rozlišovacích znaků nabývá u každé krávy jedné ze tří možných hodnot, které pro jednoduchost označme písmeny »X«, »Y« a »Z«. Můžete předpokládat, že každé dvě krávy se liší v aspoň jednom znaku.

Vaším úkolem je napsat program, který farmáři Johnovi pomůže rozpoznat zvolenou krávu z jeho stáda. Program může položit farmáři Johnovi nejvýše 100 otázek typu: „Patří hodnota rozlišovacího znaku s pořadovým číslem T do množiny S?“, kde S je podmnožina množiny $\{X, Y, Z\}$. Na základě odpovědí na položené otázky váš program pak určí číslo krávy. Snažte se, aby počet otázek potřebných pro určení krávy byl co nejmenší.

Vstupní soubor: `guess.in`

- První řádek obsahuje dvě celá čísla N a P oddělená jednou mezerou. N ($1 \leqq N \leqq 50$) je počet krav a P ($1 \leqq P \leqq 8$) je počet rozlišovacích znaků.
- Každý z následujících N řádků popisuje jednu krávu ze stáda farmáře Johna. Druhý řádek souboru popisuje krávu s pořadovým číslem 1, třetí řádek krávu s číslem 2, atd. Každý z těchto řádků obsahuje P písmen navzájem oddělených vždy jednou mezerou. První písmeno na řádku udává hodnotu rozlišovacího znaku s pořadovým číslem 1, druhé znaku s číslem 2, atd.

Příklad vstupního souboru:

```
4 2
X Z
X Y
Y X
Y Y
```

Interaktivní komunikace: standardní vstup a výstup (standard input and output)

Komunikace vašeho programu s farmářem Johnem probíhá přes standardní vstup a výstup.

Program položí otázku vypsáním řádku na standardní výstup v následujícím formátu: Prvním znakem řádku je velké písmeno `Q` následované jednou mezerou, pořadovým číslem rozlišovacího znaku, na který se ptá, a jednou nebo více jeho hodnotami. Všechny hodnoty jsou odděleny na řádku mezerami. Např. „`Q 1 Z Y`“ reprezentuje otázku „Je hodnota prvního rozlišovacího znaku dané krávy z množiny $\{Z, Y\}$?“ Číslo rozlišovacího znaku musí být z intervalu $1 \ldots P$. Každá z možných hodnot znaku může být uvedena v jedné otázce nejvýše jednou a musí být reprezentována jedním z písmen »`X`«, »`Y`« a »`Z`«.

Poté, co program položí otázku, načte odpověď ze standardního vstupu. Odpověď je reprezentována číslem 1 nebo 0: Číslo 1 znamená, že rozlišovací znak má jednu z hodnot uvedených v otázce, zatímco číslo 0 znamená opačnou odpověď.

Nakonec váš program vypíše řádek s pořadovým číslem krávy. Tento řádek musí začínat písmenem »`C`«, po kterém následuje jedna mezera a pořadové číslo krávy.

Příklad komunikace (pro výše uvedený vstupní soubor):

Vstup	*Výstup*	*Vysvětlení*
	`Q 1 X Z`	
`0`		Může být kráva 3 nebo 4.
	`Q 2 Y`	
`1`		Je to kráva 4!
	`C 4`	
program skončil		

Omezení:

Časový limit	1 s CPU
Paměťový limit	64 MB

Hodnocení. Správnost: 30 % bodů

Za daný testovací vstup získáte body za korektnost, pokud váš program položí nejvýše 100 otázek, správně určí pořadové číslo krávy a v okamžiku ukončení výpočtu existuje jediná kráva, jejíž rozlišovací znaky jsou konzistentní s odpověďmi na otázky programu.

Počet položených otázek: 70 % bodů

Zbývající body získáte podle počtu položených otázek. Rozhodující je počet položených otázek v nejhorším případě (podle zákona schválnosti všechny případy budou ty nejhorší možné). Část bodů bude přidělena i řešením, jejichž počet otázek bude blízký optimu.

5. Roboti

Stali jste se (ne)šťastnými majiteli dvou robotů, kteří jsou zrovna umístěni ve dvou bludištích. Každé z bludišť je obdélníkového tvaru. Představujme si každé z nich jako čtverečkovou síť tvořenou jednotlivými poli. Pole se souřadnicemi $(1, 1)$ je umístěno v levém horním rohu bludiště.

V i-tém bludišti ($i = 1, 2$) se nachází G_i strážců ($0 \leqq G_i \leqq 10$), kteří se snaží chytit roboty. Každý ze strážců se stále pohybuje po přímé trase tam a zpět. Trasa strážce je tvořena několika sousedními poli bludiště. Vaším úkolem je napsat program, který nalezne posloupnost příkazů, které vyvedou oba roboty z bludiště, aniž by byli chyceni některým ze strážců.

Na začátku každé minuty vyšlete stejný příkaz oběma robotům. Příkaz udává jeden ze čtyř směrů: nahoru, dolů, doprava, doleva (north, south, east, west). Robot se pak posune v zadaném směru o jedno pole, pokud mu v pohybu nebrání zeď bludiště. Jestliže je v daném směru zeď bludiště, robot žádný pohyb v následující minutě nevykoná, ale instrukce je považována za korektní. Robot opustí bludiště, pokud vykoná krok vedoucí mimo obdélníkovou síť popisující bludiště. Po opuštění bludiště robot další příkazy ignoruje.

Každý ze strážců se na začátku každé minuty posune o právě jedno pole. Výchozí pozice a natočení strážců jsou zadány na vstupu programu. Strážce nejprve vykoná tolik kroků, kolik je počet polí jeho trasy zmenšený o jedna. Na posledním poli se strážce otočí a začne se pohybovat zpět ke svému výchozímu poli, kde se opět otočí o 180° a takto hlídkuje, dokud oba roboti neopustí svá bludiště.

Trasa každého strážce je zvolena tak, že neprotíná zeď bludiště a ani nevychází z bludiště ven. Trasy různých strážců se mohou protínat, ale jejich pohyb je zvolen tak, že se nikdy nesrazí, tj. nikdy nebudou dva strážci na konci některé minuty stát na stejném poli a ani si během některé z minut nevymění vzájemně své pozice. Počáteční pozice strážců jsou zvoleny tak, že se žádný z nich nenachází na poli, kde stojí robot.

Strážce chytí robota, pokud se nachází na konci některé minuty na stejném poli jako robot nebo když si během některé z minut s robotem vymění svou pozici.

Váš program obdrží popis dvou bludišť, každé o rozměrech nejvýše 20×20 polí, spolu s počátečními pozicemi robotů a trasami jednotlivých strážců. Úkolem programu je najít posloupnost instrukcí, společnou pro oba roboty, podle které oba roboti opustí bludiště, aniž by byli chyceni některým ze strážců. Snažte se, aby čas, kdy poslední robot opustí své

bludiště, byl co nejmenší. Čas, kdy první z robotů opustí bludiště, je nepodstatný. Minimalizujte tedy čas, který uplyne od začátku do okamžiku, kdy se oba roboti nacházejí mimo svá bludiště.

Vstupní soubor: `robots.in`

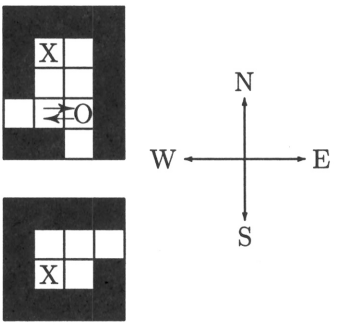
Vstupní soubor je tvořen dvěma částmi. První z nich popisuje první bludiště, pozici robota v něm a trasy strážců. Podobně druhá část souboru popisuje druhé bludiště.

- První řádek obsahuje dvě celá čísla R_1 a C_1 oddělená jednou mezerou. R_1 je počet řádků prvního bludiště a C_1 je počet jeho sloupců.
- Následuje R_1 řádků, z nichž každý obsahuje C_1 znaků. Tyto řádky tvoří mapu bludiště. Počáteční pozice robota je označena písmenem »`X`«, znak »`.`« představuje volné pole a znak »`#`« zeď. V popisu bludiště je právě jedno písmeno »`X`«.
- Další řádek obsahuje celé číslo G_1 ($0 \leqq G_1 \leqq 10$). Toto číslo udává počet strážců v prvním bludišti.
- Následuje G_1 řádků, z nichž každý popisuje počáteční pozici jednoho ze strážců. Každý z těchto řádků obsahuje tři celá čísla a jeden znak navzájem oddělené mezerami. První dvě čísla představují řádkovou a sloupcovou souřadnici strážce, třetí číslo počet polí jeho trasy a poslední znak počáteční nastavení směru pohybu strážce. Počet polí tvořících trasu strážce je 2, 3 nebo 4. Poslední znak na řádku je jedno z písmen »`N`«, »`S`«, »`E`«, »`W`« (north, south, east, west — nahoru, dolů, doprava, doleva).

Vstupní soubor poté obsahuje popis druhého bludiště ve stejném tvaru.

Příklad vstupního souboru:

```
5 4
####
#X.#
#..#
...#
##.#
1
4 3 2 W
4 4
####
#...
#X.#
####
0
```



Obr. 43

Výstupní soubor: `robots.out`

První řádek výstupního souboru musí být tvořen právě jedním celým číslem K ($K \leqq 10\,000$), které udává délku nalezené posloupnosti instrukcí. Je zaručeno, že pokud existuje posloupnost instrukcí, která vyvede oba roboty z bludišť, potom existuje i taková posloupnost s nejvýše 10 000 instrukcemi. Následujících K řádků bude obsahovat vámi nalezenou posloupnost instrukcí. Každý z těchto řádků je tvořen jedním znakem z množiny »`N`«, »`S`«, »`E`«, »`W`«. Pokud žádná taková posloupnost instrukcí neexistuje, výstupní soubor bude obsahovat jediný řádek s číslem `-1`.

Oba roboti se musí po provedení posloupnosti instrukcí uvedené ve výstupním souboru nacházet mimo bludiště. Poslední instrukce nalezené posloupnosti musí být právě ta, po níž poslední robot (nebo oba najednou) opustí bludiště.

Pokud existuje více optimálních řešení, můžete vypsat jedno libovolné z nich.

Příklad výstupního souboru:

```
8
E
N
E
S
S
S
E
S
```

Omezení:

Časový limit	2 s CPU
Paměťový limit	64 MB

Hodnocení. Za testovací vstup, pro který neexistuje řešení, lze získat pouze plný nebo nulový počet bodů. Ostatní testovací vstupy budou hodnoceny, jak je popsáno níže.

Správnost: 20 % bodů

Body za správnost získáte, pokud tvar výstupního souboru odpovídá popisu uvedenému v zadání této úlohy, nalezená posloupnost instrukcí má délku nejvýše 10 000, po jejím provedení se oba roboti nacházejí mimo svá bludiště a poslední instrukce nalezené posloupnosti způsobí, že alespoň jeden robot opustí své bludiště.

Optimalita: 80 % bodů

Body za optimalitu získáte, pokud lze výstup považovat za správný dle minulého odstavce a nalezená posloupnost instrukcí má nejmenší možnou délku. V opačném případě nezískáváte za optimalitu žádné body.

6. Ohrada

Farmář Dan se rozhlíží po ohradě kolem svého čtvercového pole o rozměrech $N \times N$ metrů ($2 \leqq N \leqq 500\,000$), jehož mapu máte k dispozici. Protilehlé rohy ohrady mají na mapě souřadnice $(0, 0)$ a (N, N) a hranice pole jsou rovnoběžné s X-ovou a Y-ovou osou.

Pletivo ohrady je upevněno na mnoha kůlech. V každém rohu je umístěn jeden kůl a podél každé strany pole jsou kůly umístěny vždy v rozestupech po 1 metru. Celkem tedy ohrada obsahuje $4N$ kůlů. Kůly jsou svislé a jejich průměr považujeme za nulový. Dan chce určit, kolik kůlů uvidí ze zvolené pozice uvnitř svého pole.

Pole si můžeme představovat jako rovinu, na které se nachází R ($1 \leqq R \leqq 30\,000$) skal, které Danovi omezují výhled. Skály mají podobu kolmých hranolů, jejichž podstavy mají tvar konvexních mnohoúhelníků. Skály jsou na poli umístěny tak, že stojí na své podstavě. Skály se navzájem neprotínají ani nedotýkají a ani se nedotýkají ohrady. Pozice farmáře Dana je zvolena tak, že se nedotýká ani jedné ze skal nebo ohrady a Dan ani na žádné skále nestojí.

Váš program obdrží rozměry pole, pozici farmáře Dana, umístění a tvar jednotlivých skal. Vaším úkolem je spočítat, kolik kůlů lze ze zadané pozice vidět. Farmář se rozhlíží do všech stran, tj. „vidí“ celý interval 360°. Nevidí však ty kůly, mezi nimiž a jím se nachází některá ze skal. Pokud vrchol podstavy skály leží přesně na spojnici kůlu a pozice farmáře Dana, předpokládáme, že Dan tento kůl také nevidí.

Vstupní soubor: `boundary.in`

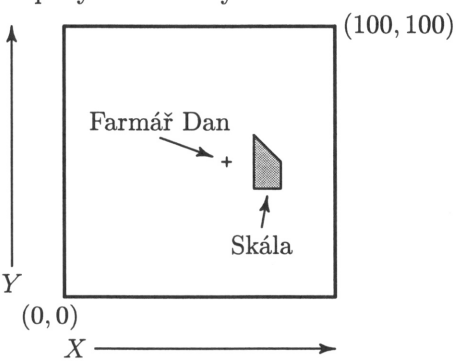
▷ První řádek souboru obsahuje dvě celá čísla N a R oddělená mezerou.

N ($2 \leqq N \leqq 500\,000$) je délka jedné strany pole a R ($1 \leqq R \leqq 30\,000$) je celkový počet skal, které se nacházejí na poli.

- ▷ Další řádek vstupního souboru obsahuje dvě celá čísla X a Y oddělená jednou mezerou. X a Y udávají souřadnice pozice farmáře Dana uvnitř jeho pole.
- ▷ Zbytek vstupního souboru tvoří popis R skal, které se nacházejí na poli:
  - ▷▷ Popis i-té skály začíná řádkem obsahujícím číslo p_i ($3 \leqq p_i \leqq 20$), které představuje počet vrcholů podstavy této skály.
  - ▷▷ Následuje p_i řádků, z nichž každý obsahuje dvě celá čísla X a Y oddělená jednou mezerou. Tato čísla udávají souřadnice vrcholů podstavy skály proti směru pohybu hodinových ručiček.

Příklad vstupního souboru:

```
100 1
60 50
5
70 40
75 40
80 40
80 50
70 60
```



Obr. 44

Povšimněte si, že podstava skály obsahuje tři kolineární vrcholy: $(70, 40)$, $(75, 40)$ a $(80, 40)$.

Výstupní soubor: `boundary.out`

Výstupní soubor musí být tvořen jedním řádkem obsahujícím jedno celé číslo. Toto číslo udává počet kůlů, které farmář vidí ze své pozice.

Příklad výstupního souboru: `319`

Omezení:

Časový limit	1 s CPU
Paměťový limit	64 MB

Hodnocení. Za každý testovací vstup obdržíte plný počet bodů, pokud váš program vytvoří správný výstupní soubor. V opačném případě je testovací vstup hodnocen 0 body.