51. ročník matematické olympiády na středních školách

14. Mezinárodní olympiáda v informatice

In: Leo Boček (editor); Karel Horák (editor); Tomáš Pitner (editor); Jaromír Šimša (editor); Jaroslav Švrček (editor); Pavel Töpfer (editor); Jaroslav Zhouf (editor): 51. ročník matematické olympiády na středních školách. Zpráva o řešení úloh ze soutěže konané ve školním roce 2001/2002. 43. mezinárodní matematická olympiáda. 14. mezinárodní olympiáda v informatice. (Czech). Praha: Jednota českých matematiků a fyziků, 2003. pp. 166–181.

Persistent URL: http://dml.cz/dmlcz/405051

Terms of use:

Institute of Mathematics of the Czech Academy of Sciences provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these *Terms of use*.

This document has been digitized, optimized for electronic delivery and stamped with digital signature within the project *DML-CZ: The Czech Digital Mathematics Library* http://dml.cz

14. mezinárodní olympiáda v informatice

Ve dnech 18.–25. 8. 2002 se v Korejské republice konal 14. ročník mezinárodní olympiády v informatice IOI 2002 (International Olympiad in Informatics). Soutěž IOI je pořádána pod záštitou UNESCO a patří mezi mezinárodní předmětové olympiády středoškoláků. Její ohlas ve světě v posledních letech stále roste a každoročně se zvyšuje počet zúčastněných zemí. Letošního ročníku se zúčastnilo 276 soutěžících ze 72 zemí celého světa, z dalších pěti zemí přijeli pozorovatelé seznámit se s průběhem olympiády a s úmyslem zúčastnit se dalšího ročníku soutěže.

Mezinárodní olympiády v informatice se pravidelně účastní i reprezentační družstvo České republiky tvořené nejlepšími řešiteli celostátního kola Matematické olympiády — kategorie P (programování). Letošní reprezentační družstvo pro IOI 2002 bylo vybráno na základě výsledků celostátního kola kategorie P 51. ročníku MO. Družstvo mělo následující složení: *Josef Cibulka* (absolvent Gymnázia v Praze 1, Štěpánská), *Pavel Čížek* (student Gymnázia v Kralupech nad Vltavou), *Milan Straka* (student Gymnázia ve Strakonicích) a *Jiří Štěpánek* (student Gymnázia na tř. Kpt. Jaroše v Brně). Vedením družstva byli pověřeni doc. *Pavel Töpfer* a *Jan Kára*, oba z Matematicko-fyzikální fakulty Univerzity Karlovy v Praze.

Naše šestičlenná delegace odlétala z Prahy v sobotu 17. 8. 2002 v odpoledních hodinách přes Amsterdam do Soulu, kde jsme přistáli v neděli 18. 8. krátce před polednem na mezinárodním letišti Incheon. Tam nás již očekávali místní pořadatelé a pomocí autobusů nás přepravili na místo konání olympiády — do areálu Kyung Hee University ve městě Yong-In. Tamní areál univerzity je poměrně rozsáhlý a bylo v něm zajištěno vše potřebné pro průběh soutěže. Byli jsme ubytováni na studentských kolejích, stravování zajišťovala místní studentská jídelna, v sálech univerzitní knihovny byly připraveny prostory pro vlastní soutěž i pro práci jury a organizátorů. Součástí univerzitního celku byla celá řada sportovišť, která byla ve volném čase k dispozici i účastníkům informatické olympiády.

Místní multimediální centrum zase všem účastníkům nabízelo nepřetržitý bezplatný přístup k počítačům a k Internetu.

Stejně jako v minulých letech byla vlastní soutěž rozdělena do dvou soutěžních dnů, v každém z nich řešili soutěžící na počítačích tři úlohy. Úlohy zadané v letošním ročníku byly algoritmicky velmi obtížné, v soutěži se objevily také některé netradiční formy úloh (interaktivní úloha, úloha s otevřeným vstupem). Podle loni přijatého modelu měli studenti pro svou práci k dispozici alternativně operační systémy Windows a Linux, takže každý mohl pracovat v takovém prostředí, které lépe zná a na které je zvyklý. Výsledné programy bylo možné psát v programovacích jazycích Pascal, C nebo C++, používaly se překladače Free Pascal a GNU C/C++, vývojové prostředí RHIDE. Svá řešení úloh soutěžící odevzdávali k vyhodnocení prostřednictvím nově vyvinutého webového rozhraní. Vyhodnocení se provádělo plně automaticky pomocí předem připravených sad testovacích dat. Různá kvalita navržených algoritmů byla bodově odlišena na základě zvolených časových limitů.

Za řešení každé úlohy bylo možné získat maximálně 100 bodů, skutečně dosažený průměrný bodový zisk z jedné úlohy byl však mnohem nižší a pohyboval se jenom kolem 20 bodů. I to svědčí o mimořádné náročnosti letošní soutěže. Mezi 276 účastníků bylo rozděleno celkem 23 zlatých medailí (mají obdobný význam jako I. cena v MMO a jsou určeny pro nejlepší přibližně 1/12 účastníků), 47 stříbrných medailí (obdoba II. ceny v MMO, určeny pro další 1/6 účastníků) a 68 bronzových medailí (odpovídají III. ceně na MMO, získá ji přibližně 1/4 účastníků). Některou z medailí má podle regulí IOI dostat nejvýše polovina soutěžících. Letos bylo uděleno celkem 138 medailí, tzn. medaili obdržela přesně polovina ze studentů soutěžících v IOI.

Výsledky našich studentů:

20.	Josef Cibulka	307 bodů	zlatá medaile
119.	Pavel Čížek	160 bodů	bronzová medaile
	Milan Straka	120 bodů	
	Jiří Štěpánek	84 bodů	

Výkony našich studentů v soutěži byly poměrně dobré a jejich výsledky nás řadí kolem 20. místa v celkovém pořadí. Olympiáda IOI je ovšem výhradně soutěží jednotlivců a oficiální výsledky družstev se zde vůbec nevyhlašují (neexistuje ani žádná metodika, jak je určovat — zda

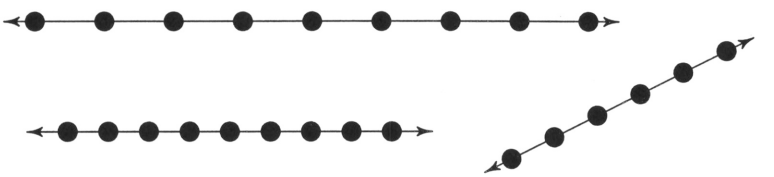
podle počtu získaných bodů, medailí nebo třeba pořadí). Velkým úspěchem našeho družstva je zejména zisk zlaté medaile *Josefa Cibulky*, druhým českým úspěšným řešitelem byl *Pavel Čížek* s bronzovou medailí. Zbývající dva naši studenti zůstali tentokrát bez medailového ocenění, pořadatelé nevydali ani kompletní výsledkovou listinu s pořadím neúspěšných řešitelů.

Příští, jubilejní 15. ročník mezinárodní olympiády v informatice IOI 2003 se bude konat ve dnech 16.–23. 8. 2003 v USA na univerzitě Wisconsin-Parkside. Američtí pořadatelé příštího ročníku olympiády během jednání probíhajících v Koreji pozvali všechny zúčastněné země k účasti na IOI 2003. Mezinárodní výbor IOI na svém jednání také potvrdil, že další budoucí ročníky IOI budou hostit po řadě Řecko (2004), Polsko (2005) a Mexiko (2006).

Texty soutěžních úloh

1. Zlobivá žába

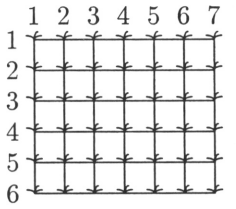
V Koreji je žába *cheonggaeguri* pověstná svou zlobivostí. Svou pověst si zcela zaslouží, protože v noci skáče přes rýžová políčka a ušlapává rostlinky. Když ráno uvidíte, které rostlinky jsou pošlapané, rádi byste zjistili, která žába vám způsobila nejvíce škody. Žába vždy skáče přes rýžové políčko po přímce a všechny její skoky jsou stejně dlouhé (obr. 45). Různé žáby mohou skákat různě dlouhými skoky a v různých směrech.



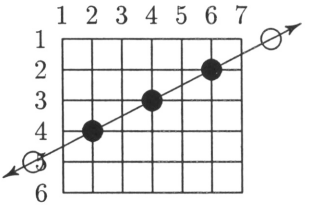
Obr. 45

Na vašem rýžovém políčku jsou rostliny vysázeny na průsečících čtvercové sítě, jak ukazuje obr. 46. Každá žába skáče přes celé políčko, tj. začíná i končí vně políčka, jak ukazuje obr. 47.

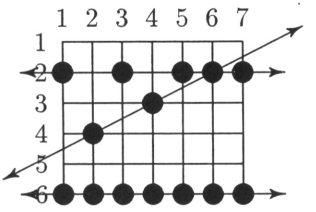
Přes políčko může skákat hodně žab. Každá žába při každém svém skoku dopadne na nějakou rostlinku, kterou tím ušlápne (obr. 48). Uvědomte si, že na jednu rostlinku může během noci doskočit několik žab. Čáry představující cestu žáby, které vidíte na obr. 48, samozřejmě na políčku nejsou vidět. Situaci, kterou ráno uvidíte, ukazuje obr. 49.



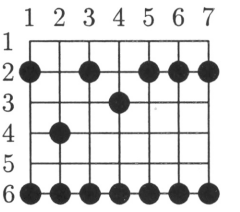
Obr. 46



Obr. 47



Obr. 48



Obr. 49

Z obr. 49 pak můžete zrekonstruovat všechny možné cesty, po kterých mohly žáby skákat přes vaše políčko. Zajímají vás pouze cesty těch žab, které vám pošlapaly alespoň tři rostlinky. Takové cesty nazveme *žabí*. V našem příkladě na obr. 48 jsou žabí všechny vyznačené cesty a také některé další cesty. Svislá cesta prvním sloupcem by mohla být cestou nějaké žáby se skokem délky čtyři, ale obsahuje pouze dvě ušlápnuté rostlinky, a proto se o ni nezajímáme. Diagonální cesta obsahující rostlinky v řadě 2 sloupci 3, řadě 3 sloupci 4 a řadě 6 sloupci 7 má sice tři rostlinky, ale ne v pravidelných vzdálenostech od sebe, a proto to nemůže být žabí cesta. Uvědomte si, že na přímce určené žabí cestou mohou ležet i nějaké další ušlápnuté rostlinky (například rostlinka na souřadnicích $(2, 6)$ na vodorovné cestě po řádce 2 v obrázku 4). Navíc ne každá ušlápnutá rostlinka musí být součástí nějaké žabí cesty.

Napište program, který ze všech možných žabích cest vybere tu, na níž žába ušlápla nejvíce rostlinek (tj. způsobila největší škodu na úrodě rýže). Na obr. 49 by to byla cesta po řadě 6 a výsledkem by bylo číslo 7.

Vstup. Váš program musí číst data ze standardního vstupu. První řádek obsahuje dvě celá čísla R a C určující počet řádků a sloupců na vašem rýžovém políčku, $1 \leqq R, C \leqq 5\,000$. Druhý řádek obsahuje jediné celé číslo N — počet ušlápnutých rostlinek, $3 \leqq N \leqq 5\,000$. Každý ze zbývajících N řádků obsahuje dvě celá čísla oddělená jednou mezerou, která představují číslo řádku a číslo sloupce ušlápnuté rostlinky (obě

čísla jsou alespoň 1 a nejvýše R, resp. C). Každá ušlápnutá rostlinka je uvedena na vstupu právě jednou.

Výstup. Váš program musí vypisovat výsledek na standardní výstup. Výstup obsahuje jeden řádek s jediným celým číslem. Toto číslo určuje počet ušlápnutých rostlinek na té žabí cestě, která způsobila největší škody. Pokud neexistuje žádná žabí cesta, výstupem bude číslo 0.

Příklady vstupů a výstupů.

Příklad 1: Vstup (příklad z obr. 49):

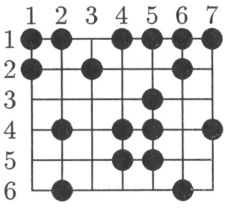
```
6 7
14
2 1
6 6
4 2
2 5
2 6
2 7
3 4
6 1
6 2
2 3
6 3
6 4
6 5
6 7
```

Výstup: 7

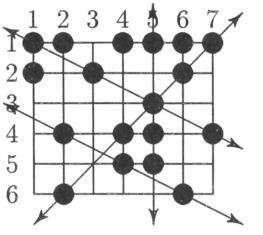
Příklad 2: Vstup (příklad z obr. 50):

```
6 7
18
1 1
6 2
3 5
1 5
4 7
1 2
1 4
1 6
1 7
2 1
2 3
2 6
4 2
4 4
4 5
5 4
5 5
6 6
```

Výstup: 4



Obr. 50

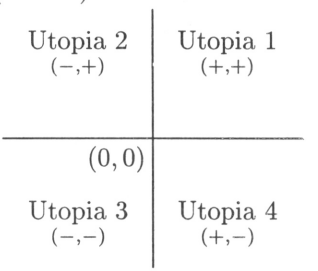


Obr. 51

Hodnocení. Pokud váš program odpoví správně v časovém limitu, získáte za daný testovací vstup plný počet bodů, jinak za tento vstup získáte 0 bodů.

2. Rozdělená Utopie

Nádherná země Utopie byla kdysi zasažena válkou. Po válce byla země rozdělena na čtyři oblasti podle poledníku (severo-jižně) a podle rovnoběžky (východo-západně). Průsečík těchto hraničních čar byl označen souřadnicemi $(0,0)$. Všechny čtyři oblasti se pojmenovaly Utopie, ale časem se zavedlo označení Utopie 1 (severovýchodní oblast), 2 (severozápadní), 3 (jihozápadní), 4 (jihovýchodní). Bod v každé z oblastí má jednoznačné souřadnice, jimiž jsou vzdálenost na východ a na sever od bodu $(0,0)$. Tyto souřadnice mohou být záporné. Body v Utopii 2 se tedy označují dvojicí souřadnic (*záporná, kladná*), v Utopii 3 dvojicí souřadnic (*záporná, záporná*), v Utopii 4 (*kladná, záporná*) a v Utopii 1 dvojicí kladných čísel (obr. 51).



Velkým problémem bylo, že obyvatelé nesměli přejíždět hranice. Naštěstí Pat a Mat (bývalí účastníci IOI) sestrojili teleport. Stroj se spouští zadáním kódových čísel, přičemž každé z nich může být použito pouze jednou. Vaším úkolem je řídit teleport z výchozí pozice $(0,0)$ po oblastech Utopie v daném pořadí. Nezáleží na tom, v kterém místě dané oblasti přistanete.

Máte tedy dáno N čísel oblastí, v nichž musíte postupně přistát. Může být požadováno, abyste přistáli vícekrát za sebou v téže oblasti. Po opuštění výchozího bodu $(0,0)$ nesmíte nikdy přistát na hranici.

Dále máte dáno $2N$ kódových čísel, která můžete použít pro řízení teleportu. Úkolem je vytvořit z nich N párů a před každé číslo umístit znaménko $+$ nebo $-$. Nacházíte-li se v bodě o souřadnicích (x, y) a použijete kódový pár $(+u, -v)$, budete teleportováni do bodu o souřadnicích $(x + u, y - v)$. Zadaných $2N$ čísel můžete použít v libovolném pořadí a každé z nich buď se znaménkem $+$, nebo $-$.

Máte-li například dána kódová čísla $7, 5, 6, 1, 3, 2, 4, 8$ a máte řídit teleport postupně do oblastí číslo $4, 1, 2, 1$, výsledná posloupnost kódových párů může obsahovat $(+7, -1)$, $(-5, +2)$, $(-4, +3)$, $(+8, +6)$. Tato posloupnost vás postupně přenese do bodů $(7, -1)$, $(2, 1)$, $(-2, 4)$ a $(6, 10)$. Tyto body se nacházejí po řadě v oblastech Utopie 4, Utopie 1, Utopie 2 a Utopie 1.

Úkol. Je dáno $2N$ navzájem různých kódových čísel a posloupnost N čísel oblastí, ve kterých máte přistát. Sestrojte z daných čísel takovou posloupnost kódových dvojic, která řídí teleport po oblastech dle zadaného pořadí.

Vstup. Váš program musí číst vstupní data ze standardního vstupu. První řádek obsahuje kladné celé číslo N, $1 \leqq N \leqq 10\,000$. Druhý řádek obsahuje $2N$ různých celých kódových čísel ($1 \leqq$ *kódové číslo* $\leqq 100\,000$) oddělených jednou mezerou. Poslední řádek obsahuje posloupnost N čísel oblastí, z nichž každé je 1, 2, 3 nebo 4.

Výstup. Váš program musí vypsat výsledek na standardní výstup. Výstup se skládá z N řádků. Každý z nich obsahuje dvojici kódových čísel, každé kódové číslo je opatřeno znaménkem. Mezi znaménkem a číslem nesmí být mezera, ale čísla na řádku musí být oddělena jednou mezerou.

Pokud existuje více řešení, váš program může vypsat jedno libovolné z nich. Jestliže žádné řešení neexistuje, program vypíše jediné číslo 0.

Příklady vstupů a výstupů.

Příklad 1: Vstup:

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

Výstup:

```
+7 -1
-5 +2
-4 +3
+8 +6
```

Příklad 2: Vstup:

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

Výstup:

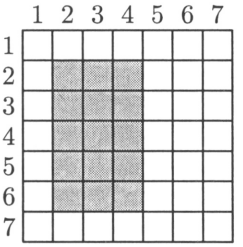
```
+3 -2
-4 +5
-6 +1
+8 +7
```

3. XOR-ová komprese

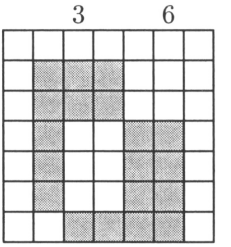
Vytváříte aplikaci pro mobilní telefon s černobílou obrazovkou. Na obrazovce se x-ové souřadnice bodů číslují od 1 zleva doprava a y-ové

souřadnice od 1 shora dolů, jak ukazuje obr. 52. Ve vaší aplikaci potřebujete na obrazovku kreslit různé obrazce různých velikostí. Abyste obrazce nemuseli ukládat, chcete je vytvářet pomocí grafické knihovny v telefonu. Můžete předpokládat, že na počátku kreslení jsou všechny body na obrazovce bílé. Jediná operace, kterou grafická knihovna podporuje, je `XOR(L,R,T,B)`. Tato operace invertuje všechny body v obdélníku s levým horním rohem na souřadnicích `(L,T)` a pravým dolním rohem `(R,B)`. `L` zde znamená levý okraj, `R` pravý, `T` horní a `B` dolní.

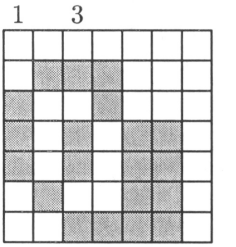
Například aplikováním operace `XOR(2,4,2,6)` na prázdnou obrazovku získáme obrazec z obr. 53. Aplikováním `XOR(3,6,4,7)` na obrazec z obr 1 dostaneme obrazec z obr. 54 a aplikováním `XOR(1,3,3,5)` na obrazec z obr 2 získáme obrazec z obr. 54.



Obr. 52



Obr. 53



Obr. 54

Máte danou sadu černobílých obrazců, které chceme vykreslovat na obrazovce telefonu. Vaším úkolem je vytvořit každý z těchto obrazců z počáteční bílé obrazovky pomocí co nejméně operací `XOR`. Obrazce dostanete popsány ve vstupních souborech. Odevzdat máte příslušné výstupní soubory s parametry pro operace `XOR`, nikoliv program!

Vstup. Dostanete deset vstupních textových souborů nazvaných `xor1.in` až `xor10.in`. Každý vstupní soubor má následující tvar. První řádek vstupního souboru obsahuje jedno celé číslo N, $5 \leqq N \leqq 2\,000$. Číslo N určuje počet řádků a sloupců obrazce. Zbývající řádky vstupního souboru reprezentují jednotlivé řady obrazce shora dolů. Řádek obsahuje vždy N celých čísel určujících zleva doprava body na obrazovce. Každé z těchto čísel je buď 0, nebo 1, přičemž 0 představuje bílý bod a 1 černý bod.

Výstup. Odevzdejte deset výstupních souborů odpovídajících daným vstupním souborům. Velikost každého výstupního souboru musí být menší než 1 megabyte.

První řádek výstupního souboru obsahuje text

`#FILE xor I`

kde I je číslo odpovídajícího vstupního souboru. Druhý řádek obsahuje celé číslo K — počet volání operace `XOR`. Následuje přesně K řádků popisujících jednotlivá volání operace `XOR` od prvního po poslední. Každý z těchto K řádků je tvořen čtyřmi celými čísly — parametry operace `XOR` v pořadí `L,R,T,B`.

Příklad vstupu a výstupu:

xor0.in	xor0.out
7	#FILE xor 0
0 0 0 0 0 0 0	3
0 1 1 1 0 0 0	2 4 2 6
1 0 0 1 0 0 0	3 6 4 7
1 0 1 0 1 1 0	1 3 3 5
1 0 1 0 1 1 0	
0 1 0 0 1 1 0	
0 0 1 1 1 1 0	

Hodnocení. Jestliže

- volání `XOR` uvedená ve výstupním souboru nevytvářejí požadovaný obrazec, nebo
- počet volání `XOR` ve výstupním souboru není roven K, nebo
- počet volání `XOR` K je větší než 40 000, nebo
- výstupní soubor obsahuje volání `XOR` s parametry $L > R$ nebo $T > B$, nebo
- výstupní soubor obsahuje volání `XOR` s nekladným parametrem, nebo
- výstupní soubor obsahuje volání `XOR` s parametrem větším než N,

pak dostanete za tento vstup 0 bodů. Jinak bude vaše hodnocení určeno vzorcem

$$1 + 9 \cdot \frac{NejmenšíDosaženýPočetVoláníZeVšechSoutěžících}{VášPočetVolání}.$$

Počet bodů za jeden vstup je zaokrouhlen na jedno desetinné místo. Celkové hodnocení úlohy je zaokrouhleno na celé číslo.

Pokud například odešlete řešení se 121 voláními operace `XOR` a vaše řešení pro tato vstupní data je nejlepší v soutěži, získáte za něj 10 bodů. Pokud by nejlepší odeslané řešení používalo pouze 98 volání `XOR`, za vaše řešení se 121 voláními dostanete $1 + 9 \cdot 98/121$ ($= 8{,}289\ldots$), což bude zaokrouhleno na 8,3 bodů.

4. Plánování dávek

Na počítači je třeba zpracovat posloupnost N úloh. Úlohy jsou očíslovány v pořadí od 1 do N. Úlohy je třeba rozdělit do jedné nebo více dávek, přičemž každá dávka se skládá z po sobě jdoucích úloh dané posloupnosti. Zpracování dávek začíná v čase 0. Dávky jsou zpracovávány po sobě v následujícím pořadí: Pokud dávka b obsahuje úlohy s menšími čísly než dávka c, je zpracována dříve. Úlohy jedné dávky jsou zpracovávány postupně, ale výstup úloh je zobrazen až po zpracování všech úloh v dávce. To znamená, že výstup úlohy j je zobrazen až v okamžiku ukončení zpracovávání dávky obsahující úlohu j.

Na začátku zpracování každé dávky potřebuje počítač čas S. Pro každou úlohu i je zadán cenový koeficient F_i a čas T_i potřebný ke zpracování úlohy. Pokud tedy dávka obsahuje úlohy $x, x+1, \ldots, y-1, y$ a její zpracování začíná v čase t, potom čas vypsání výstupu každé úlohy v dávce je $t + S + (T_x + T_{x+1} + \ldots + T_{y-1} + T_y)$. Nezapomeňte, že počítač vypisuje výsledky až v okamžiku zpracování všech úloh v dávce. Pokud čas vypsání výstupu úlohy i je O_i, pak cena zpracování této úlohy je $O_i \cdot F_i$. Předpokládejme, že máme 5 úloh, $S = 1$, $(T_1, T_2, T_3, T_4, T_5) = (1, 3, 4, 2, 1)$ a $(F_1, F_2, F_3, F_4, F_5) = (3, 2, 3, 3, 4)$. Pokud úlohy rozdělíme do tří dávek $\{1, 2\}$, $\{3\}$, $\{4, 5\}$, potom časy jejich výstupů budou $(O_1, O_2, O_3, O_4, O_5) = (5, 5, 10, 14, 14)$ a ceny zpracování jednotlivých úloh budou po řadě $(15, 10, 30, 42, 56)$. Celková cena je rovna součtu cen zpracování všech úloh. V našem příkladě při zvoleném rozdělení úloh do dávek je to 153.

Napište program, který pro daný čas S a posloupnost úloh s jejich dobami zpracování a cenovými koeficienty nalezne rozdělení úloh do dávek s minimální celkovou cenou.

Vstup. Váš program musí číst vstupní data ze standardního vstupu. První řádek obsahuje počet úloh N, $1 \leqq N \leqq 10\,000$. Druhý řádek obsahuje čas S (celé číslo, $0 \leqq S \leqq 50$). Následujících N řádků obsahuje informace o úlohách $1, 2, \ldots, N$ v daném pořadí. Na každém z těchto řádků je uvedeno celé číslo T_i, $1 \leqq T_i \leqq 100$ (doba zpracování úlohy) a za ním celé číslo F_i, $1 \leqq F_i \leqq 100$ (cenový koeficient úlohy).

Výstup. Váš program musí vypsat výsledek na standardní výstup. Výstup je tvořen jediným řádkem obsahujícím jedno celé číslo: minimální možnou celkovou cenu zpracování.

Příklady vstupů a výstupů.

Příklad 1: Vstup:

```
2
50
100 100
100 100
```

Výstup:

```
45000
```

Příklad 2: Vstup:

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

Výstup:

```
153
```

Příklad 2 je příkladem ze zadání úlohy.

Poznámka. Pro žádný testovací vstup celková cena pro libovolné rozdělení úloh do dávek nepřekročí $2^{31} - 1$.

Hodnocení. Pokud váš program vypíše správnou odpověď ve stanoveném časovém limitu, získáte za daná vstupní data plný počet bodů, jinak za tato vstupní data získáte 0 bodů.

5. Autobusové zastávky

Ve městě Yong-In se chystají vytvořit autobusovou síť s N zastávkami. Yong-In je moderní město, a tak jeho mapa vypadá jako čtvercová síť, v níž všechny čtverce mají stejnou velikost. Každá autobusová zastávka je umístěna v průsečíku dvou ulic. Dvě zastávky budou zvoleny jako přestupní (označme je H_1 a H_2). Přestupní zastávky budou navzájem propojeny expresní autobusovou linkou. Každá ze zbývajících $N - 2$ zastávek bude přímo spojena s právě jednou z přestupních zastávek, nebude ale propojena s žádnou nepřestupní zastávkou.

Vzdálenost mezi dvěma zastávkami je rovna délce nejkratší možné cesty po ulicích města. To znamená, že pokud souřadnice zastávky označíme (x, y), kde x je x-ová souřadnice a y je y-ová souřadnice, pak vzdálenost mezi dvěma zastávkami (x_1, y_1) a (x_2, y_2) je rovna $|x_1 - x_2| + |y_1 - y_2|$. Jsou-li zastávky A a B spojeny se stejnou přestupní zastávkou H_1, pak vzdálenost zastávek A a B je rovna součtu vzdáleností z A do H_1 a z H_1 do B. Jestliže jsou zastávky A a B spojeny s různými přestupními zastávkami, např. A je spojena s H_1 a B je spojena s H_2, potom vzdálenost zastávek A a B je rovna součtu vzdáleností z A do H_1, z H_1 do H_2 a z H_2 do B.

Plánovací oddělení Yong-Inské radnice chce zajistit, aby se každý občan města dostal do libovolného bodu ve městě co nejrychleji. Proto plánovači chtějí zvolit dvě přestupní zastávky a napojení ostatních zastávek na ně takovým způsobem, aby vzdálenost dvou nejvzdálenějších zastávek ve vzniklé autobusové síti byla co možná nejmenší.

Napište program, který spočte nejmenší možnou vzdálenost dvou nejvzdálenějších zastávek v Yong-Inu přes všechny možné volby dvou přestupních zastávek a napojení ostatních zastávek na ně.

Vstup. Váš program musí číst vstupní data ze standardního vstupu. První řádek obsahuje jedno kladné celé číslo N, $2 \leqq N \leqq 500$, což je počet všech autobusových zastávek. Každý ze zbylých N řádků vstupu obsahuje souřadnice jedné ze zastávek, nejprve x-ovou a poté y-ovou. Obě souřadnice jsou kladná celá čísla $\leqq 5\,000$. Žádné dvě autobusové zastávky neleží na stejné křižovatce.

Výstup. Váš program musí vypsat výsledek na standardní výstup. Výstup je tvořen jedním řádkem s jedním kladným celým číslem — minimální možnou vzdáleností dvou nejvzdálenějších zastávek.

Příklady vstupů a výstupů.

Příklad 1: Vstup:

```
6
1 7
16 6
12 4
4 4
1 1
11 1
```

Výstup:

```
20
```

Příklad 2: Vstup:

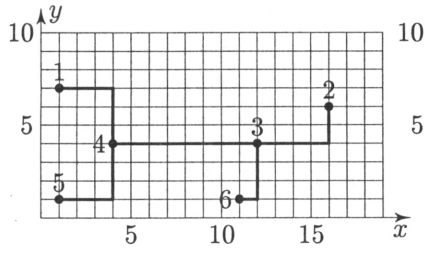
```
7
7 9
10 9
5 3
1 1
7 2
15 6
17 7
```

Výstup:

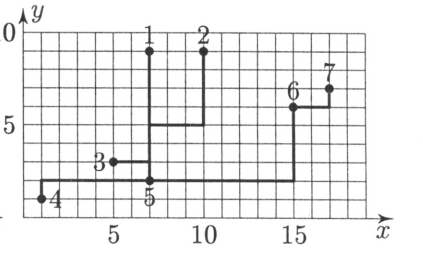
```
25
```

Obrázek 55 ukazuje autobusovou síť pro vstupní data z příkladu 1. Jsou-li v příkladu 1 zastávky 3 a 4 zvoleny jako přestupní, pak největší vzdálenost je buď mezi zastávkami 2 a 5, nebo mezi zastávkami 2 a 1. Lepší volba přestupních zastávek není možná, takže správná odpověď je 20.

Obrázek 56 ukazuje situaci pro data z příkladu 2. Jsou-li zastávky 5 a 6 zvoleny jako přestupní, potom jsou nejvzdálenějšími zastávkami



Obr. 55



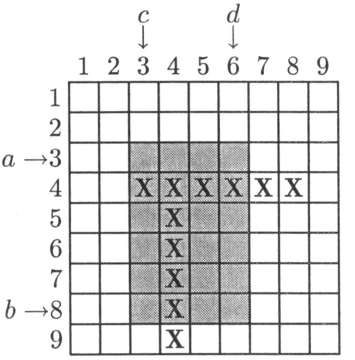
Obr. 56

zastávky 2 a 7. Neexistuje lepší volba přestupních stanic, a proto 25 je výsledkem úlohy.

Hodnocení. Pokud váš program vypíše správnou odpověď ve stanoveném časovém limitu, získáte za daná vstupní data plný počet bodů, jinak za tato vstupní data získáte 0 bodů.

6. Dvě tyče

Tyčí nazveme buď vodorovnou, nebo svislou řadu alespoň dvou políček čtvercové sítě. Dvě tyče (jedna vodorovná a jedna svislá) jsou umístěny v síti $N \times N$. Na obr. 57 jsou tyto tyče vyznačeny znaky **X**. Tyče mohou ale nemusí mít stejnou délku, mohou také mít společné políčko. Pokud nějaké políčko může ležet v jedné nebo obou tyčích (jako je tomu na obr. 57 u políčka $(4, 4)$), tak situaci chápeme vždy tak, že leží v obou tyčích. Svislá tyč na obr. 57 tedy končí na políčku $(4, 4)$, a ne na políčku $(5, 4)$. Políčko $(4, 4)$ je koncem svislé tyče a zároveň vnitřním bodem vodorovné tyče.



Obr. 57

Na počátku nevíte, kde tyče jsou, a tak máte napsat program, který je nalezne. Vodorovnou tyč označme ROD1 a svislou ROD2. Každé políčko je reprezentováno dvojicí řádek/sloupec (r, c). Políčko v horním levém rohu čtvercové sítě má souřadnice $(1, 1)$. Každá tyč je popsána dvojicí svých koncových políček $\langle (r_1, c_1), (r_2, c_2) \rangle$. Na obr. 57 popíšeme tyč ROD1 dvojicí $\langle (4, 3), (4, 8) \rangle$ a tyč ROD2 dvojicí $\langle (4, 4), (9, 4) \rangle$.

V této úloze musíte pro vstup a výstup použít speciální knihovnu. Rozměry čtvercové sítě (číslo N) získáte voláním knihovní funkce `gridsize`. Tuto funkci musí váš program zavolat na začátku výpočtu pro každá testovací data. Pro nalezení polohy tyčí můžete používat pouze knihovní funkci `rect(a,b,c,d)`, která zkoumá obdélníkovou oblast sítě vymezenou souřadnicemi $[a, b] \times [c, d]$ (dejte pozor na význam jednotlivých parametrů — viz šedivou oblast na obr. 57), kde $a \leqq b$ a $c \leqq d$. [Nenechte se zmást tím, že grafická knihovna jednoho nejmenovaného mobilního telefonu používá odlišný souřadný systém.] Pokud se ve zkoumaném obdélníku nachází alespoň jeden znak **X** (tj. část nějaké tyče), funkce `rect` vrátí 1; jinak vrátí 0. V příkladu na obr. 57 by volání `rect(3,8,3,6)` vrátilo hodnotu 1. Vaším úkolem je napsat program, který určí přesnou pozici obou tyčí pomocí omezeného počtu volání funkce `rect`.

Výsledek předáte voláním další knihovní funkce `report`$(r_1, c_1, r_2, c_2, p_1, q_1, p_2, q_2)$, kde tyč ROD1 je reprezentována dvojicí $\langle (r_1, c_1), (r_2, c_2) \rangle$ a tyč ROD2 je reprezentována dvojicí $\langle (p_1, q_1), (p_2, q_2) \rangle$. Volání funkce `report` ukončí váš program. Uvědomte si, že ROD1 je vodorovná tyč a ROD2 svislá tyč, (r_1, c_1) je levý konec vodorovné tyče ROD1, (p_1, q_1) je horní konec svislé tyče ROD2. Platí tedy $r_1 = r_2$, $c_1 < c_2$, $p_1 < p_2$ a $q_1 = q_2$. Pokud parametry volání funkce `report` neodpovídají těmto podmínkám, bude na standardní výstup vypsáno chybové hlášení.

Omezení.

- Vstup můžete obdržet pouze pomocí knihovních funkcí `gridsize` a `rect`.
- N splňuje podmínky $5 \leqq N \leqq 10\,000$.
- Pro každá testovací data můžete funkci `rect` zavolat nejvýše 400krát. Pokud váš program zavolá funkci `rect` vícekrát, bude ukončen.
- Váš program musí volat funkci `rect` více než jednou a funkci `report` právě jednou.
- Když je funkce `rect` volána s nekorektními parametry (např. zkoumaný obdélník přesahuje rozměry čtvercové sítě), je váš program ukončen.

▷ Váš program nesmí přistupovat k žádným souborům a nesmí používat standardní vstup a výstup.

Knihovny. Máte k dispozici následující knihovny:

FreePascal (`prectlib.ppu`, `prectlib.o`)

```
function gridsize : LongInt;
function rect(a,b,c,d : LongInt) : LongInt;
procedure report(r1, c1, r2, c2, p1, q1, p2, q2 :
                 LongInt);
```

Do vašeho programu `rods.pas` vložte příkaz

```
uses prectlib;
```

Program přeložte pomocí

```
fpc -So -O2 -XS rods.pas
```

Program `prodstool.pas` ukazuje příklad použití této knihovny.

GNU C/C++ (`crectlib.h`, `crectlib.o`)

```
int gridsize();
int rect(int a, int b, int c, int d);
void report(int r1, int c1, int r2, int c2,
            int p1, int q1, int p2, int q2);
```

Do svého programu `rods.c` vložte direktivu

```
#include "crectlib.h"
```

Program přeložte pomocí

```
gcc -O2 -static rods.c crectlib.o -lm
g++ -O2 -static rods.cpp crectlib.o -lm
```

Program `crodstool.c` ukazuje příklad použití této knihovny.

C/C++ v RHIDE

Nezapomeňte nastavit Option→Linker na `crectlib.o`.

Testování. Abyste mohli testovat svůj program s knihovnou, musíte vytvořit textový soubor `rods.in`. Soubor musí obsahovat tři řádky. První řádek obsahuje jedno celé číslo N — rozměr čtvercové sítě. Druhý řádek obsahuje souřadnice koncových bodů tyče ROD1, r_1, c_1, r_2, c_2, kde (r_1, c_1) je levý konec tyče. Třetí řádek obsahuje analogicky souřadnice koncových bodů tyče ROD2, p_1, q_1, p_2, q_2, kde (p_1, q_1) je horní konec tyče.

Po ukončení vašeho programu voláním funkce `report` dostanete výstupní soubor `rods.out`. Soubor obsahuje počet volání funkce `rect` a souřadnice konců tyče, které jste předali funkci `report`. Pokud během výpočtu nastala nějaká chyba při volání knihovních funkcí, soubor `rods.out` bude obsahovat příslušné chybové hlášení.

Dialog mezi vaším programem a knihovnou je zaznamenáván do souboru `rods.log`. Tento soubor obsahuje posloupnost volaní funkce `rect` ve tvaru „k : `rect`(a, b, c, d) = ans“. Záznam znamená, že při k-tém volání `rect`(a, b, c, d) byla vrácena hodnota ans.

Příklad souborů pro testování.

`rods.in`	`rods.out`
9	20
4 3 4 8	4 3 4 8
4 4 9 4	4 4 9 4

Hodnocení. Pokud váš program poruší některé z výše uvedených omezení (např. vykoná více než 400 volání funkce `rect`) nebo dá chybný výsledek, tak pro tato testovací data bude hodnocen 0 body.

Pokud je výsledek správný, hodnocení závisí na počtu provedených volání funkce `rect`. Pokud bylo pro tato vstupní data provedeno nejvýše 100 volání, získáte 5 bodů. Pokud program provedl 101 až 200 volání, dostanete 3 body. Je-li počet provedených volání mezi 201 a 400, dostanete 1 bod.