18. Mezinárodní olympiáda v informatice

In: Karel Horák (editor); Martin Mareš (editor); Peter Novotný (editor); Jaromír Šimša (editor); Jaroslav Švrček (editor); Pavel Töpfer (editor); Jaroslav Zhouf (editor): 55. ročník matematické olympiády na středních školách. Zpráva o řešení úloh ze soutěže konané ve školním roce 2005/2006. 47. mezinárodní matematická olympiáda. 18. mezinárodní olympiáda v informatice. (Czech). Praha: Jednota českých matematiků a fyziků, 2007. pp. 176–191.

Persistent URL: http://dml.cz/dmlcz/405121

18. mezinárodní olympiáda v informatice

Ve dnech 13.–20. 8. 2006 se konala v Mexiku 18. mezinárodní olympiáda v informatice IOI 2006 (IOI — International Olympiad in Informatics). Soutěž proběhla ve městě Mérida v provincii Yucatán, tedy na stejném místě, kde se o rok dříve konala i mezinárodní matematická olympiáda.

Olympiády se zúčastnilo 284 soutěžících ze 76 zemí celého světa. Každé družstvo je tvořeno nejvýše čtyřmi soutěžícími studenty a je doprovázeno dvěma pedagogickými pracovníky jako vedoucími. Českou republiku letos reprezentovalo družstvo ve složení:

Jan Hrnčíř, absolvent gymnázia F. X. Šaldy v Liberci,
Daniel Marek, absolvent gymnázia Ch. Dopplera v Praze 5,
Josef Pihera, student gymnázia ve Strakonicích,
Michal Vaner, absolvent gymnázia v Turnově.

Naši soutěžící byli vybráni na základě výsledků dosažených v celostátním kole 55. ročníku Matematické olympiády — kategorie P. Jako příprava vybraných reprezentantů na soutěž posloužilo zejména tradiční týdenní česko-polsko-slovenské přípravné soustředění, které se tentokrát konalo v červnu na Slovensku. Vedoucími české delegace byli jmenováni Mgr. *Martin Mareš* a doc. RNDr. *Pavel Töpfer*, CSc., oba pracovníci Matematicko-fyzikální fakulty Univerzity Karlovy v Praze.

Ubytování všech účastníků bylo zajištěno v několika hotelích nedaleko centra Méridy, ve stejném místě probíhala i všechna jednání spojená s organizací soutěže a přípravou soutěžních úloh. Vlastní soutěž studentů u počítačů byla uspořádána v moderním konferenčním středisku Yucatan SIGLO XXI na severním okraji města.

Soutěž IOI probíhá vždy ve dvou soutěžních dnech, v každém z nich soutěžící řeší po dobu pěti hodin tři zadané úlohy. Každý účastník má pro svoji práci přidělen osobní počítač s nainstalovanými překladači programovacích jazyků Pascal, C a C++ a s interaktivním webovým rozhraním pro komunikaci soutěžícího s řídicím a vyhodnocovacím systémem soutěže. To umožňuje zálohovat data, tisknout výpisy programů, ověřovat správnost chování programu a zejména pak předávat vytvořené programy

na vyhodnocení. Všechny soutěžní úlohy jsou algoritmického charakteru a je nutné dovést je až do podoby kompletního odladěného programu, podobně jako je tomu třeba i v praktické části celostátního kola naší Matematické olympiády — kategorie P. Odevzdané programy jsou vždy po skončení soutěžního dne automaticky testovány pomocí předem připravené sady testovacích dat, aby se ověřila jejich správnost. Důležitou součástí těchto testů jsou časové limity. Je pevně stanoveno, jak nejdéle může program počítat pro každá vstupní data. Tímto způsobem se mezi správně fungujícími programy rozliší, na jak dobrém algoritmu je který program založen. Některá vstupní data zadávaná při testování jsou malá, takže výpočet s nimi stihne v časovém limitu i pomalejší algoritmus, naopak jiná vstupní data jsou rozsáhlá a včas je zvládne zpracovat jedině program využívající dostatečně efektivní algoritmus.

Pro všechny účastníky IOI 2006 byl připraven i zajímavý doprovodný program. Ve volném dnu oddělujícím oba soutěžní dny nám organizátoři nabídli celodenní výlet do známé rekreační oblasti Progresso spojený s koupáním v moři a řadou dalších sportovních činností. Po ukončení soutěže jsme na závěr našeho pobytu v Mexiku měli možnost navštívit jedno z archeologických nalezišť se zachovalými památkami staré mayské kultury — asi 120 km vzdálené středisko Chichén Itzá.

Soutěžní úlohy letošního ročníku olympiády byly algoritmicky zajímavé a dobře připravené, jejich náročnost byla vcelku přiměřená této soutěži. Za každou úlohu bylo možné získat maximálně 100 bodů, tj. celkově v soutěži 600 bodů. To se ovšem nikomu ze studentů nepodařilo, celkový vítěz získal 480 bodů. Na základě dosažených výsledků se na IOI udělují mediale tak, že polovina účastníků obdrží některou z medailí, přičemž počet zlatých, stříbrných a bronzových medialí je v rámci možností přibližně v poměru 1 : 2 : 3. Letos bylo uděleno 24 zlatých medailí (soutěžícím, kteří dosáhli alespoň 385 bodů), 51 stříbrných medailí (za zisk alespoň 314 bodů) a 70 bronzových medailí (pro ty, kdo v soutěži získali minimálně 219 bodů).

Reprezentanti ČR si vedli v soutěži dobře, získali jednu stříbrnou a dvě bronzové medaile. Výsledky našich studentů:

57.	Josef Pihera	339 bodů	stříbrná medaile
85.	Daniel Marek	296 bodů	bronzová medaile
144.	Michal Vaner	219 bodů	bronzová medaile
255.	Jan Hrnčíř	41 bodů	

Mezinárodní olympiáda v informatice je soutěží jednotlivců, takže žádné oficiální pořadí národních družstev se v ní nevyhlašuje. Není ani stanoveno, zda by se mělo určovat podle součtu dosažených bodů, součtu umístění nebo třeba podle počtu získaných medailí. Nejúspěšnějšími zeměmi letošního ročníku IOI byly Čína (4 zlaté medaile), Polsko a Rusko (po 3 zlatých medailích), rovněž celkový vítěz IOI byl z Polska. Slovensko získalo na IOI 2006 tři stříbrné medaile.

19. mezinárodní olympiáda v informatice se bude konat v chorvatském Zagrebu.

Texty soutěžních úloh

1. Mayské písmo

Luštění písma starých Mayů se ukázalo být poněkud tvrdším oříškem, než se zprvu zdálo. Po dvou staletích vesměs marného zkoumání teprve posledních 30 let přineslo jakési výsledky.

Mayské písmo je založeno na malých obrázcích neboli *glyfech*, které odpovídají jednotlivým zvukům. Mayská slova se obvykle zapisují několika glyfy poskládanými k sobě.

Jedním z mnoha problémů při čtení mayských nápisů je určit správné pořadí glyfů. Staří písaři se totiž pro vzájemnou polohu glyfů tvořících slovo často nerozhodovali podle nějakých pravidel, ale spíš podle svého uměleckého cítění. I když výslovnost mnohých glyfů je již známa, často nevíme, jak vyslovit jimi zapsané slovo.

Badatelé zrovna v nápisech hledají určité slovo W. Vědí, z jakých je složeno glyfů, ale doposud neobjevili všechny možnosti, jak je k sobě Mayové skládali. A jelikož se doslechli o vašem věhlasu, požádali vás o pomoc. Dostanete od nich g glyfů tvořících slovo W a posloupnost glyfů S; tyto glyfy tvoří právě studovaný nápis (v pořadí, jak jsou namalovány). Pomozte jim spočítat všechny možné výskyty slova W v nápisu S.

Úloha: Napište program, který dostane glyfy slova W a posloupnost glyfů S a vypíše všechny možné výskyty slova W v posloupnosti S, čili všechny posloupnosti g po sobě jdoucích glyfů v posloupnosti S, které jsou permutací glyfů tvořících slovo W.

Omezení: $1 \leqq g \leqq 3\,000$ — počet glyfů slova W,
$g \leqq |S| \leqq 3\,000\,000$, kde $|S|$ je počet glyfů v posloupnosti S.

Vstup: Program bude číst vstupní data ze souboru `writing.in`:

`writing.in`	*Popis*:
`4 11`	*1. řádek*: obsahuje dvě celá čísla oddělená mezerou: g a $\|S\|$.
`cAda`	
`AbrAcadAbRa`	*2. řádek*: obsahuje g po sobě jdoucích znaků popisujících glyfy slova W. Jsou povoleny znaky `a`–`z` a `A`–`Z`; velká a malá písmena považujeme za různá.
	3. řádek: obsahuje $\|S\|$ po sobě jdoucích znaků popisujících glyfy v posloupnosti S. Opět jsou povoleny znaky `a`–`z` a `A`–`Z` a rozlišuje se velikost písmen.

Výstup: Program zapíše do výstupního souboru `writing.out` následující údaje:

`writing.out`	*Popis*:
`2`	*Jediný řádek*: obsahuje počet možných výskytů W v S.

Hodnocení: Část testovacích vstupů (dohromady za 50 bodů) bude splňovat podmínku $g \leqq 10$.

Upozornění pro Pascalisty: Freepascalský typ `string` je implicitně omezen na 255 znaků. Pokud byste chtěli používat delší stringy, připište si těsně za řádek `program` ... direktivu `{$H+}`.

2. Pyramida

Poté, co král Jaguár zvítězil ve veliké bitvě, rozhodl se postavit pyramidu, která poslouží nejen jako památník jeho slavného vítězství, ale také coby hrobka vojáků v bitvě padlých. Pyramidu postaví na bojišti. Bojiště má tvar obdélníka, který si můžeme představit jako čtvercovou síť $m \times n$ políček (m sloupců v n řádcích) a jsou v něm četné terénní nerovnosti. Královští stavitelé pro každé políčko sítě změřili jeho výšku.

Základnou pyramidy bude obdélník o velikosti $a \times b$ políček (sloupce × × řádky). Uvnitř bude na úrovni terénu menší obdélníková komora velikosti $c \times d$ políček (opět sloupce × řádky), v níž spočinou těla a zbraně padlých bojovníků.

Jelikož obzvláštní přízni krále se těší celá čísla, bude jak pyramida, tak komora mít půdorys složený z políček zmíněné čtvercové sítě. Zatímco terén políček, na nichž je umístěna pohřební komora, zůstane v původní úrovni, zbývající políčka tvořící základnu pyramidy budou srovnána do stejné výšky přesouváním zeminy z vyšších políček na nižší. Výsledná *výška základny* bude rovna aritmetickému průměru původních výšek všech políček tvořících základnu kromě políček pod komorou.

Stavitelé mohou komoru umístit kdekoliv uvnitř pyramidy, pokud bude ze všech stran obklopena zdmi širokými alespoň 1 políčko.

Pomozte stavitelům vybrat nejlepší umístění pyramidy a pohřební komory. To je takové, při němž bude základna v největší možné výšce. Přitom je nutné dodržet požadované rozměry pyramidy i komory.

Obrázek 59 ukazuje příklad bojiště. Čísla v jednotlivých políčkách udávají jejich výšky. Šedá políčka představují základnu pyramidy, zatímco jimi obklopený bílý obdélník značí možné umístění komory. Na obrázku je jedno z optimálních řešení.

	1	2	3	4	5	6	7	8
1	1	5	10	3	7	1	2	5
2	6	12	4	4	3	3	1	5
3	2	4	3	1	6	6	19	8
4	1	1	1	3	4	2	4	5
5	6	6	3	3	3	2	2	2

Obr. 59

Úloha: Napište program, který pro dané rozměry bojiště, pyramidy a komory a pro dané výšky všech políček bojiště najde umístění pyramidy na bojišti a komory v pyramidě takové, že výška základny bude největší možná.

Pokud existuje více optimálních řešení, vypište libovolné jedno z nich.

Omezení: $3 \leqq m \leqq 1\,000$, $3 \leqq n \leqq 1\,000$,
$3 \leqq a \leqq m$, $3 \leqq b \leqq n$,
$1 \leqq c \leqq a - 2$, $1 \leqq d \leqq b - 2$.
Všechny výšky jsou celá čísla od 1 do 100.

Vstup: Program bude číst vstupní data ze souboru `pyramid.in`:

`pyramid.in`

```
8 5 5 3 2 1
1 5 10 3 7 1 2 5
6 12 4 4 3 3 1 5
2 4 3 1 6 6 19 8
1 1 1 3 4 2 4 5
6 6 3 3 3 2 2 2
```

Popis:

1. řádek: obsahuje 6 celých čísel oddělených mezerami: m, n, a, b, c, d.

Následujících n řádků: Tyto řádky popisují jednotlivé řádky čtvercové sítě v pořadí od prvního k n-tému. Každý z nich obsahuje m celých čísel oddělených mezerami. Tato čísla popisují výšky políček v příslušném řádku sítě v pořadí od prvního sloupce k m-tému.

Výstup: Program zapíše do výstupního souboru `pyramid.out` následující údaje:

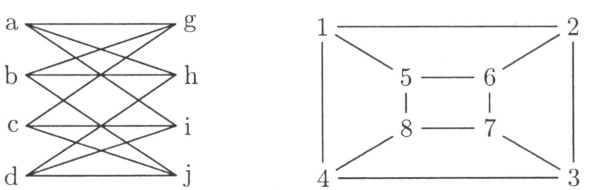
`pyramid.out`	*Popis*:
`4 1` `6 2`	*1. řádek*: obsahuje dvě celá čísla oddělená mezerou. Tato čísla udávají souřadnice levého horního rohu základny pyramidy (v pořadí sloupec, řádek). *2. řádek*: obsahuje dvě celá čísla oddělená mezerou. Tato čísla udávají souřadnice levého horního rohu komory (v pořadí sloupec, řádek).

Hodnocení: Část testovacích vstupů (dohromady za 30 bodů) bude splňovat podmínky $3 \leqq m, n \leqq 10$.

3. Zakázaný podgraf

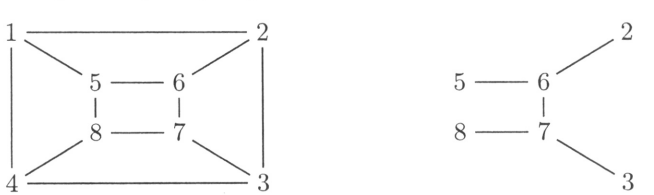
Dva grafy G a H nazveme *isomorfní*, pokud mají stejný počet vrcholů a vrcholům grafu G lze přiřadit navzájem různé vrcholy grafu H tak, aby pro každou dvojici vrcholů v grafu G platilo, že mezi nimi vede hrana právě tehdy, vede-li hrana také mezi jim odpovídající dvojicí vrcholů v grafu H.

Kupříkladu oba grafy na obr. 60 jsou isomorfní, přestože na první pohled je každý úplně jiný. Můžeme totiž jejich vrcholy přiřadit například takto: a–1, b–6, c–8, d–3, g–5, h–2, i–4, j–7.



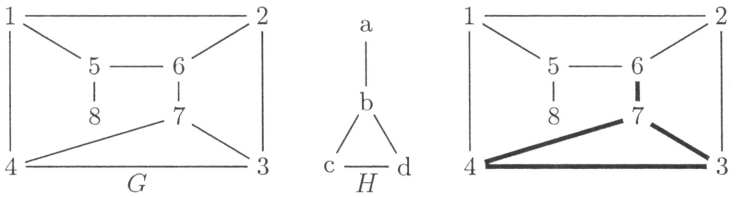
Obr. 60

Podgrafem grafu G nazveme libovolný graf, jehož množiny vrcholů a hran jsou podmnožinami množin vrcholů a hran grafu G. Obr. 61 ukazuje příklad grafu a jeho podgrafu.



Obr. 61

Řekneme, že graf G *obsahuje* graf H, existuje-li v grafu G alespoň jeden podgraf H', který je isomorfní s grafem H (obr. 62).



Obr. 62

Úloha: Pro zadané neorientované grafy G a H najděte podgraf G' grafu G takový, aby G a G' měly stejný počet vrcholů a graf G' neobsahoval graf H.

Podgrafů s žádanými vlastnostmi přirozeně může být mnoho. Úkolem je nalézt jeden z těch, které mají co možná největší počet hran.

Základní algoritmus: Podgrafy neobsahující graf H je možné hledat například následujícím primitivním způsobem: Budeme procházet hrany grafu G v pořadí, v jakém jsou popsány vstupním souborem, a postupně je přidávat do G'. V každém kroku přitom budeme ověřovat, zda H není obsažen v G'. Správná implementace tohoto hladového algoritmu nějaké body získá, ovšem existují i mnohem lepší strategie.

Omezení: $3 \leqq m \leqq 4$ — počet vrcholů grafu H,
$3 \leqq n \leqq 1\,000$ — počet vrcholů grafu G.

Vstup: Dostanete deset vstupních souborů `forbidden1.in` až `forbidden10.in`, každý z nich v následujícím tvaru:

`forbiddenK.in`	*Popis*:
`3 5`	*1. řádek*: obsahuje 2 celá čísla oddělená mezerou: m a n.
`0 1 0`	*Následujících m řádků*: Každý z těchto řádků obsahuje m celých čísel oddělených mezerami a popisuje jeden vrchol grafu H. Vrcholy jsou očíslovány od 1 do m. Na j-tém řádku tohoto bloku vstupu je i-té číslo rovno 1, pokud vrcholy i a j jsou spojeny hranou, jinak je rovno 0.
`1 0 1`	
`0 1 0`	
`0 1 0 0 0`	
`1 0 1 0 0`	
`0 1 0 1 0`	
`0 0 1 0 1`	
`0 0 0 1 0`	*Následujících n řádků*: Každý z těchto řádků obsahuje n celých čísel oddělených mezerami. Tato čísla obdobným způsobem popisují graf G.

Povšimněte si, že s výjimkou prvního řádku obsahuje vstup matice sousednosti grafů H a G.

Výstup: Odevzdejte 10 souborů, pro každý ze vstupních souborů jeden. Každý soubor nechť obsahuje následující:

`forbidden`K`.out`	*Popis*:
`#FILE forbidden` K	*1. řádek*: hlavička souboru. Musí obsahovat `#FILE forbidden` K kde K je číslo od 1 do 10 identifikující vstup, ke kterému tento výstup patří.
`5`	*2. řádek*: obsahuje jediné celé číslo n.
`0 1 0 0 0`	*Následujících* n *řádků*: Každý řádek obsahuje n mezerami oddělených celých čísel, která popisují graf G' výše uvedeným způsobem.
`1 0 0 0 0`	
`0 0 0 0 0`	
`0 0 0 0 0`	
`0 0 0 0 0`	

Povšimněte si, že kromě řádků 1 a 2 výstup odpovídá matici sousednosti grafu G' a že výše uvedený ukázkový výstup je správný, ale ne optimální.

Hodnocení: Počet bodů, které získáte, bude záviset na počtu hran vašeho grafu G' a bude stanoven takto: Nenulový počet bodů můžete získat pouze tehdy, bude-li výstup splňovat zadání. Pokud splňuje, označíme si E_y počet hran ve vašem řešení, E_b počet hran v řešení nalezeném základním algoritmem, E_m počet hran v nejlepším ze všech odevzdaných řešení a použijeme následující vzorec:

$$\text{body} = \begin{cases} 30\,\dfrac{E_y}{E_b}, & \text{pokud } E_y \leqq E_b, \\ 30 + 70\,\dfrac{E_y - E_b}{E_m - E_b}, & \text{pokud } E_y > E_b. \end{cases}$$

4. Mexická dolina

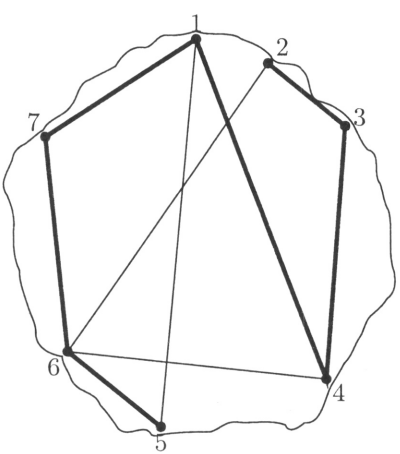
Mexico City stojí v malebném údolí známém jako Mexická dolina. V dávných časech bylo na jeho místě jezero, ale okolo roku 1300 se aztéčtí velekněží usnesli, že ve středu jezera vytvoří ostrov, na kterém vybudují centrum celé říše. Dnes už z jezera nezbylo nic.

Před příchodem Aztéků stálo na břehu jezera c měst. Některé dvojice měst uzavřely obchodní dohody, podle nichž převážely zboží na lodích mezi těmito městy. Trasa lodí vedla vždy po úsečce.

Jak už to bývá, králové měst se rozhodli obchod zorganizovat. Navrhli obchodní cestu spojující všechna města stojící okolo jezera. Cesta měla následující vlastnosti:

- ▷ začínala v některém městě, navštívila každé z ostatních měst a skončila v městě různém od počátečního;
- ▷ cesta každé město navštívila právě jednou;
- ▷ každá dvojice měst jdoucích na cestě po sobě měla spolu uzavřenu obchodní dohodu;
- ▷ každá taková dvojice měst byla spojena lodní trasou vedoucí po úsečce;
- ▷ aby se lodě nesrážely, žádné dvě lodní trasy se nekřížily.

Obr. 63 ilustruje jezero a města vůkol. Úsečky odpovídají obchodním dohodám, tučně je vyznačena obchodní cesta, která začíná v městě 2 a končí v městě 5. Tato cesta sama sebe nikde nekříží. Nebylo by například možné, aby cesta z města 2 vedla do města 6 a pak do měst 5 a 1, protože tehdy by se křížila.



Obr. 63

Města jsou očíslována od 1 do c ve směru hodinových ručiček.

Úloha: Napište program, jenž pro zadaný počet měst a seznam obchodních dohod mezi nimi sestrojí obchodní cestu splňující výše uvedené podmínky.

Omezení: $3 \leqq c \leqq 1\,000$ počet měst okolo jezera.

Vstup: Program bude číst vstupní data ze souboru `mexico.in`:

mexico.in	*Popis*:
7	*1. řádek*: obsahuje jediné celé číslo c.
9	*2. řádek*: obsahuje jediné celé číslo n — počet obchodních
1 4	dohod.
5 1	*Následujících n řádků*: Tyto řádky popisují jednotlivé ob-
1 7	chodní dohody. Každý z nich obsahuje dvě celá čísla od-
5 6	dělená mezerou — čísla měst, která uzavřela dohodu.
2 3	
3 4	
2 6	
4 6	
6 7	

Výstup: Program zapíše do výstupního souboru `mexico.out` následující údaje:

mexico.out	*Popis*:
2	Pokud je možné sestrojit obchodní cestu, soubor obsahuje
3	c řádků, na nichž jsou uvedena čísla všech měst v pořadí,
4	jak jdou po sobě na obchodní cestě. Pokud požadovaná
1	cesta neexistuje, soubor obsahuje jediný řádek s číslem `-1`.
7	
6	
5	

Poznámka: Pokud existuje více cest splňující zadané podmínky, vypište libovolnou z nich.

Hodnocení: Část testovacích vstupů (dohromady za 40 bodů) bude splňovat podmínky $3 \leqq c \leqq 20$.

5. Spojovaná

Spojovaná (Joining Points) je hra pro jednoho hráče. Začíná tím, že si zvolíte dvě celá čísla $g \geqq 2$ a $r \geqq 2$. Poté nakreslíte čtyři body ve vrcholech čtverce: horní dva zelené, dolní dva červené. Pokračujete v kreslení zelených a červených bodů uvnitř čtverce tak, aby žádné tři body (včetně prvních čtyř) neležely v jedné přímce. Takto nakreslíte celkem g zelených a r červených bodů.

Po nakreslení všech bodů je začnete spojovat úsečkami. Dva body můžete spojit, pokud:

▷ oba body jsou téže barvy, a zároveň

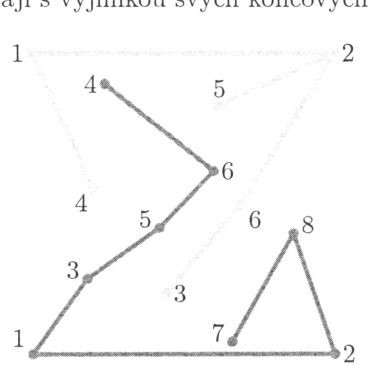
▷ nová úsečka neprotíná žádnou již nakreslenou úsečku jinde než v krajních bodech.

O dvou bodech u, v řekneme, že leží v téže *komponentě*, pokud je možné dojít z bodu u do bodu v po nakreslených úsečkách.

Hru vyhrajete, pokud se vám podaří všechny zelené body spojit do jedné komponenty pomocí právě $g - 1$ úseček a také všechny červené body do jedné komponenty pomocí právě $r - 1$ úseček. Lze dokázat, že ať už jsou body zadány jakkoliv, *hru lze vždy vyhrát.*

Dostanete čtvercový hrací plán o rozměrech $s \times s$ obsahující g zelených a r červených bodů, nakreslených na souřadnicích (x_i, y_i), kde x_i a y_i jsou celá čísla. Zelené body očíslujeme od 1 do g, přičemž bod v levém horním rohu čtverce na souřadnicích $(0, s)$ dostane číslo 1, bod v pravém horním rohu na souřadnicích (s, s) číslo 2 a body uvnitř čtverce čísla 3 až g. Červené body očíslujeme od 1 do r, levý dolní na souřadnicích $(0, 0)$ bude mít číslo 1, pravý dolní na $(s, 0)$ číslo 2, vnitřní body čísla 3 až r.

Obr. 64 ilustruje jeden příklad zadání a vyhrávajícího řešení. Všechny zelené body jsou spojeny do jedné komponenty a všechny červené do druhé. Povšimněte si, že žádné tři body neleží v téže přímce a že se žádné dvě úsečky neprotínají s výjimkou svých koncových bodů.



Obr. 64

Úloha: Napište program, který pro zadané souřadnice g zelených a r červených bodů určí, jak nakreslit $g - 1$ zelených a $r - 1$ červených úseček tak, aby byly všechny zelené body spojeny do jedné komponenty, všechny červené do druhé a žádné dvě úsečky se neprotínaly.

Omezení:	$3 \leqq g \leqq 50\,000$	počet zelených bodů,
	$3 \leqq r \leqq 50\,000$	počet červených bodů,
	$0 \leqq s \leqq 200\,000\,000$	velikost hracího plánu.

Vstup: Program bude číst vstupní data ze souboru `points.in`:

`points.in`

```
6
0 1000
1000 1000
203 601
449 212
620 837
708 537
8
0 0
1000 0
185 300
314 888
416 458
614 622
683 95
838 400
```

Popis:

1. řádek: obsahuje jediné celé číslo g.

Následujících g řádků: Tyto řádky popisují jednotlivé zelené body v pořadí od bodu 1 do bodu g. Na každém řádku se nacházejí dvě mezerou oddělená celá čísla x_i a y_i, souřadnice i-tého zeleného bodu.

$(g+2)$-hý řádek: obsahuje jediné celé číslo r.

Následujících r řádků: Tyto řádky popisují jednotlivé červené body v pořadí od bodu 1 do bodu r. Na každém řádku se nacházejí dvě mezerou oddělená celá čísla x_i a y_i, souřadnice i-tého červeného bodu.

Výstup: Program zapíše do výstupního souboru následující údaje:

`points.out`

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

Popis:

Výstupní soubor obsahuje $(g-1)+(r-1)$ řádků, z nichž každý popisuje jednu úsečku spojující dva body.

Na každém řádku jsou tři mezerou oddělené údaje: dvě celá čísla a znak reprezentující barvu úsečky. Čísla označují body spojené úsečkou, znak má hodnotu `g`, pokud jsou oba body zelené, nebo `r`, jsou-li červené.

Nezáleží na pořadí, ve kterém úsečky uvedete, ani na pořadí bodů v popisu úsečky.

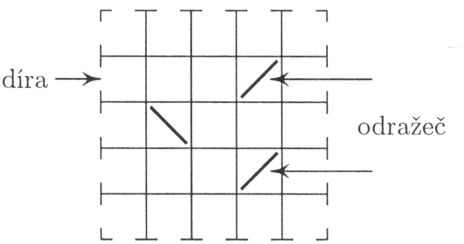
Hodnocení: Část testovacích vstupů (dohromady za 35 bodů) bude splňovat podmínky $3 \leqq g, r \leqq 20$.

6. Černá skříňka

Pojďte, zahrajeme si hru s černou skříňkou. Na stole leží skříňka ve tvaru čtverce. Na každé straně skříňky je n otvorů (celkem v ní je tedy

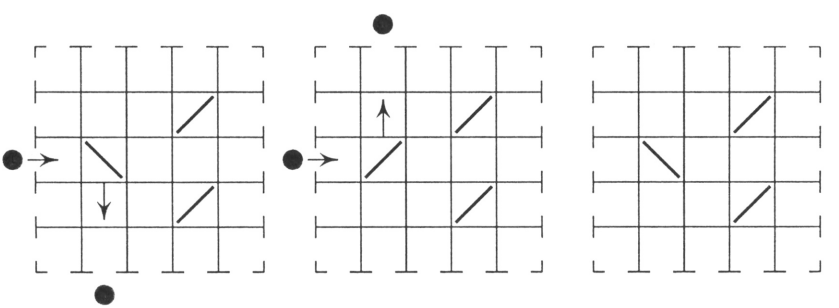
$4n$ otvorů), do nichž lze házet kuličky. Každá vhozená kulička po čase vypadne některým z otvorů ven, možná i otvorem, jímž jsme ji vhodili.

Vnitřek skříňky si můžeme představit (a také nakreslit, obr. 65) jako mřížku $n \times n$ políček. Otvory jsou umístěny na obou koncích všech řádků a sloupců mřížky. Každé políčko mřížky je buďto prázdné, nebo obsahuje *odražeč*. To je zařízení, které mění směr pohybu kuličky o 90°.



Obr. 65

Vhozená kulička se uvnitř skříňky pohybuje přímo, než buďto narazí do odražeče, nebo ze skříňky vypadne některým z otvorů. Pokud narazí do odražeče, kulička změní svůj směr pohybu a odražeč se otočí o 90°. Obr. 66 ukazuje činnost odražeče:



Obr. 66

a) Kuličku jsme vhodili otvorem. Kulička naráží do odražeče a mění směr pohybu.
b) Po průletu první kuličky změnil odražeč svůj směr. Vhodili jsme další kuličku do stejného otvoru, ta se od téhož odražeče odrazí v opačném směru než první kulička.
c) Odražeč se otáčí pokaždé, když je zasažen.

Při každém nárazu kuličky do odražeče se ozve pípnutí. Počet nárazů vhozené kuličky do odražečů při průletu skříňkou lze tedy snadno zjistit spočítáním pípnutí. Lze dokázat, že kulička vždy ze skříňky vypadne. Skříňka je vybavena tlačítkem, které všechny odražeče otočí do počátečního stavu, a druhým tlačítkem, jímž můžeme všechny odražeče otočit o 90°.

Úloha: Dostanete 15 černých skříněk, které můžete ovládat pomocí knihovny funkcí. Zjistěte vnitřní uspořádání všech skříněk co nejpřesněji a odevzdejte soubory popisující jednotlivé skříňky. Rovněž budete mít k dispozici prostředky pro definování svých vlastních testovacích černých skříněk.

Omezení: $1 \leqq n \leqq 30$.

Výstup: Odevzdejte 15 souborů, pro každou černou skříňku jeden. Každý soubor nechť obsahuje následující:

`blackbox`K`.out` *Popis*:

`#FILE blackbox` K *1. řádek*: hlavička souboru. Musí obsahovat

```
.....
.../.
.\...
.../.
.?.?.
```

`#FILE blackbox` K

kde K je číslo od 1 do 15 identifikující černou skříňku, ke které tento výstup patří.

Následujících n řádků: Každý řádek popisuje jeden řádek černé skříňky, v pořadí od horního řádku po spodní. Na každém řádku je přesně n znaků; každý znak odpovídá jednomu políčku řádku (v pořadí zleva doprava):

- ▷ '`.`' značí, že políčko je prázdné,
- ▷ '`/`' značí, že políčko obsahuje odražeč s počátečním stavem '`/`',
- ▷ '`\`' značí, že políčko obsahuje odražeč s počátečním stavem '`\`',
- ▷ '`?`' značí, že jste nezjistili obsah políčka.

Knihovna: Dostanete knihovnu, která poskytuje následující funkce:

Pascal:

```
function Initialize(box: integer):integer;
```

Inicializuje knihovnu. Tuto funkci je třeba zavolat právě jednou na počátku vašeho programu. Funkce vrací počet otvorů na každé straně skříňky (n).

C/C++:

```
int Initialize(int box);
```

Parametr *box* obsahuje jedno celé číslo v rozsahu 1 až 15 určující skříňku, kterou chcete zkoumat, nebo nulu, pokud si obsah skříňky chcete zadat sami.

Pascal:

```
function throwBall(holeIn, sideIn: integer;
   var holeOut, sideOut: integer): longint;
```

C:

```
int throwBall(int holeIn, int sideIn,
   int *holeOut, int *sideOut);
```

C++:

```
int throwBall(int holeIn, int sideIn,
   int &holeOut, int &sideOut);
```

Vhodí do skříňky kuličku otvorem *holeIn* na straně *sideIn*. Strany jsou číslovány takto: 1 — horní, 2 — pravá, 3 — dolní, 4 — levá. Otvory jsou číslovány zleva doprava a shora dolů, na každé straně od jedničky. V *holeOut* a *sideOut* se dozvíte otvor a stranu, kudy kulička ze skříňky vypadla. Funkce *throwBall* vrací počet pípnutí, která se během pokusu ozvala.

Pascal:

```
procedure ResetBox;
```

C/C++:

```
void ResetBox();
```

Otočí všechny odražeče do počátečního stavu.

Pascal:

```
procedure ToggleDeflectors;
```

C/C++:

```
void ToggleDeflectors();
```

Otočí každý odražeč ve skříňce o 90°.

Pascal:

```
procedure Finalize;
```

C/C++:

```
void Finalize();
```

Ukončí komunikaci se skříňkou. Tuto funkci byste měli zavolat na konci svého programu.

Vynecháváme zde popis použití knihoven, protože nejsou k dispozici příslušné soubory...

Ukázka komunikace s knihovnou: S černou skříňkou z obr. 66a by bylo prostřednictvím knihovny možné komunikovat například takto:

`Initialize(0);` Pokud zadáme skříňku z obrázku, funkce vrátí hodnotu 5.

Pascal:

```
throwBall(3, 4, holeOut, sideOut);
```

C:

```
throwBall(3, 4, &holeOut, &sideOut);
```

C++:

```
throwBall(3, 4, holeOut, sideOut);
```

Kuličku vhazujeme do otvoru č. 3 (třetí shora) na levé straně. Funkce vrátí 1, což znamená, že kulička narazila do jednoho odražeče. Po návratu z funkce je *holeOut* = 2 a *sideOut* = 3, čili že kulička vypadla otvorem č. 2 (druhý zleva) na dolní straně skříňky.

Testování: Pokud funkci `Initialize` zavoláte s parametrem 0, knihovna si obsah skříňky přečte ze souboru `blackbox.in`. Takto můžete s knihovnou experimentovat. Soubor má následující formát:

`blackbox.in`	*Popis*:
`5`	*1. řádek*: obsahuje jediné celé číslo n — počet otvorů na každé straně skříňky.
`3`	
`2 3 \`	*2. řádek*: obsahuje jediné celé číslo d — počet odražečů uvnitř skříňky.
`4 2 /`	*Následujících d řádků*: Tyto řádky popisují jednotlivé odražeče. Každý z nich obsahuje dvě celá čísla oddělená mezerou — sloupec a řádek polohy odražeče. Za nimi následuje mezera a jeden znak, který popisuje počáteční stav odražeče a může mít hodnotu '/' nebo '\'.
`4 4 /`	

Poznámka: Uvedený příklad vstupu popisuje skříňku z obr. 66a.

Hodnocení: Pro každou skříňku odevzdejte textový soubor co nejlépe popisující vnitřek této skříňky. Hodnocen bude takto:

- Pokud váš soubor obsahuje znak '.', '/' nebo '\' na chybné pozici, dostanete za tuto skřínku 0 bodů.
- Označíme-li B_m maximální počet správně rozpoznaných políček mezi všemi správnými řešeními účastníků, a B_y počet políček správně rozpoznaných vaším řešením, dostanete za tuto skříňku

$$\frac{100 \cdot B_y}{B_m} \text{ procent maximálního počtu bodů.}$$

Poznámka: Vzorové řešení této úlohy dokáže automaticky určit 100 % počátečního obsahu všech zadaných skříněk za méně než 8 minut.