MATEMATICKÁ OLYMPIÁDA NA STREDNÝCH ŠKOLÁCH

50. ROČNÍK, školský rok 2000/2001

Kategória P

Matematická olympiáda je súťaž pre žiakov stredných škôl našej republiky. **Kategória P** je zameraná na programovanie a je určená žiakom všetkých ročníkov.

Organizácia súťaže v kategórii P

Kategória P matematickej olympiády má tri postupové kolá – domáce, krajské a celoštátne.

V **I. kole** účastníci riešia štyri úlohy uvedené v tomto letáku. Riešenia odovzdajú svojmu učiteľovi informatiky do **13. novembra 2000**. Učitelia informatiky odošlú riešenia v tomto termíne zo školy na príslušnú adresu podľa krajov takto:

Košický, Prešovský:
RNDr. Gabriela Andrejková, CSc., KMI PF UPJŠ, Jesenná 5, 041 54 Košice

Žilinský:
RNDr. Peter Varša, KI FR VŠDS, Moyzesova 20, 010 26 Žilina

Banskobystrický:
Mgr. Vladimír Siládi, KI FPV UMB, Tajovského 40, 974 01 Banská Bystrica

Trnavský, Trenčiansky, Nitriansky:
Doc. RNDr. V. Štoffová, CSc., KI FPV UKF, tr. A. Hlinku 1, 949 74 Nitra

Bratislavský:
RNDr. Dana Pardubská, CSc., KVI MFF UK, Mlynská Dolina, 842 48 Bratislava

Najúspešnejší riešitelia domáceho kola sú pozvaní do **II. kola** (krajského), kde riešia štyri teoretické úlohy. Do **III. kola** (celoštátneho) sú pozývaní najúspešnejší riešitelia všetkých krajských kôl, pričom riešia tri teoretické úlohy a dve praktické úlohy pri počítači. Z najlepších riešiteľov tohto kola SK MO vyberie družstvá pre Medzinárodnú informatickú olympiádu a Stredoeurópsku informatickú olympiádu.

Predbežné termíny 50. ročníka MO, kategória P

I. Domáce kolo	13. novembra 2000
II. Krajské kolo	9. januára 2001
III. Celoštátne kolo	4.–7. apríla 2001

Usporiadateľ súťaže

MO vyhlasuje *Ministerstvo školstva SR*. Súťaž organizuje *Slovenská komisia MO* v spolupráci s *Jednotou slovenských matematikov a fyzikov* a *Matematickým ústavom Slovenskej akadémie vied* a v jednotlivých krajoch ju riadia *krajské komisie MO* pri pobočkách JSMF. Na jednotlivých školách ju zaisťujú učitelia matematiky a informatiky.

Formálna úprava riešenia

Riešenia súťažných úloh domáceho kola kategórie P pozostávajú z dvoch častí:

Popis riešenia. Riešenia musia obsahovať podrobný popis použitého algoritmu, zdôvodnenie jeho správnosti a diskusiu o efektivite zvoleného riešenia (t.j. posúdenie časových a pamäťových nárokov programu). Algoritmus by mal byť jasný už z popisu riešenia bez toho, aby bolo potrebné nahliadnuť do programu.

Program. V praktických úlohách **P–I–1**, **P–I–2** a **P–I–3** je potrebné k riešeniu pripojiť odladený program napísaný v jazyku Pascal, C alebo C++.[1] Program sa odovzdáva v písomnej forme (jeho výpis je teda súčasťou riešenia) i na diskete, aby bolo možné otestovať jeho funkčnosť.
Súbory na diskete pomenujte `p1`x`.pas`, `p1`x`.cpp`, kde x je číslo súťažnej úlohy. Disketu označte menom riešiteľa. Z jednej školy možno poslať všetky riešenia na jednej diskete. V tomto prípade pre každého riešiteľa vytvorte podadresár označený jeho priezviskom a disketu označte adresou školy.
V úlohe **P–I–4** je potrebné navrhnúť a zapísať príslušné dlaždicové programy.

Písomnú časť riešenia vypracujte čitateľne na listy formátu A4. **Každú úlohu začnite na novom liste** a v záhlaví uveďte vaše meno, triedu, adresu školy a označenie príkladu podľa tohto letáku. Zadania úloh nemusíte opisovať. Ak sa vám riešenie nezmestí na jeden list, uveďte na ďalších listoch vľavo hore svoje meno a označenie úlohy a očíslujte strany.

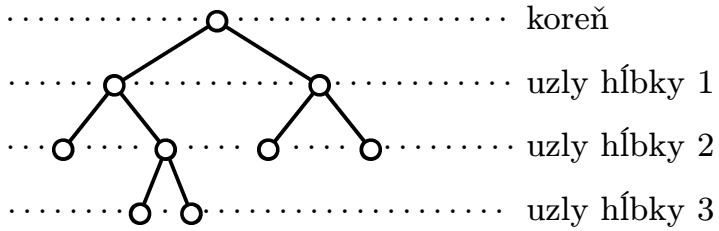
[1] Použite dialekty Borland Pascal v. 7.0 a Borland C++ v. 3.1.

KATEGÓRIA P

P-I-1

(Jan Kára)

Binárny strom je štruktúra tvorená jednotlivými uzlami. Jeden z uzlov je významný – hovoríme mu koreň. Každý z uzlov buď nemá žiadneho následníka (potom sa nazýva list), alebo má práve dvoch následníkov (ďalšie uzly stromu). Hĺbkou uzla rozumieme jeho vzdialenosť od koreňa stromu. Uvedomte si, že koreň môže byť i listom – potom je binárny strom tvorený jediným vrcholom hĺbky 0. Príklad binárneho stromu si môžete prezrieť na nasledujúcom obrázku:



Aby sme mohli binárne stromy jednoducho popisovať, zavedieme si nasledujúce kódovanie: k-ty riadok kódovania (pre $k = 0, 1, 2 \ldots$) popisuje práve uzly hĺbky k v poradí zľava doprava. Uzol binárneho stromu, ktorý nie je listom, budeme v našom kódovaní zobrazovať znakom `U`, listy budeme označovať znakom `L`. Binárny strom z predchádzajúceho obrázku teda bude zakódovaný ako:

```
U
UU
LULL
LL
```

Súťažná úloha:

Je daný počet listov N ($N \leq 10\,000$) a ich hĺbky v binárnom strome (nejakých N prirodzených čísel). Napíšte program, ktorý zostaví binárny strom so zadanými hĺbkami listov a ten vypíše v našom kódovaní. Ak vstupným dátam vyhovuje viac binárnych stromov, program vypíše ľubovoľný jeden z nich. Pokiaľ vyhovujúci strom neexistuje, program o tom vypíše správu.

Formát vstupu: Prvý riadok vstupného súboru `stromy.in` obsahuje jediné číslo N (počet listov). Druhý riadok obsahuje N čísel – hĺbky listov hľadaného stromu.

Formát výstupu: Výstupný súbor `stromy.out` bude obsahovať kódovanie nájdeného stromu vo vyššie uvedenom formáte, prípadne správu „`Zodpovedajuci strom neexistuje.`“.

Príklad:

Vstup:

```
stromy.in
4
2 3 1 3
```

Výstup:

```
stromy.out
U
UL
LU
LL
```

Vstup:
`stromy.in`
`3`
`1 1 2`

Výstup:
`stromy.out`
`Zodpovedajuci strom neexistuje.`

P-I-2

(Jan Kára)

Na kráľovstvo kráľa Mieromila III. zaútočili nepriateľské vojská a podarilo sa im obsadiť niekoľko miest. Kráľ teraz potrebuje dať svojmu generálovi príkaz k protiútoku (bez príkazu predsa generál nemôže bojovať). Generál však momentálne prevádza inšpekciu vojsk v inom meste. Je preto treba vyslať posla, ktorý príkaz čo najrýchlejšie doručí. Príkaz však v žiadnom prípade nesmie padnúť do rúk nepriateľa! Preto sa posol musí neustále držať čo najďalej od nepriateľom obsadených miest. Vašou úlohou je navrhnúť pre posla čo najlepšiu trasu.

Súťažná úloha:

Program dostane na vstupe zadaný počet miest N ($1 \leq N \leq 100$). Jednotlivé mestá budeme označovať číslami $1 \ldots N$. Ďalej je na vstupe uvedený počet ciest M ($1 \leq M \leq 10\,000$) a zoznam týchto ciest vedúcich medzi mestami. Každá cesta je určená dvojicou čísel miest, ktoré spája. Cesty sa krížia len v mestách a je možné sa po nich dostať z ľubovoľného mesta do ľubovoľného (prípadne cez iné mestá). Ďalší údaj K zadaný na vstupe určuje počet miest obsadených nepriateľom, nasleduje zoznam obsadených miest. Nakoniec program dostane číslo mesta, odkiaľ vyráža posol, a číslo mesta, kde sa zdržuje generál. Váš program má nájsť trasu, ktorej vzdialenosť od miest obsadených nepriateľom je maximálna. Pokiaľ existuje takých trás viac, program určí ľubovoľnú najkratšiu z nich. Vzdialenosť miest $\mathcal{A}$ a $\mathcal{B}$ počítame ako minimálny počet ciest, po ktorých musíme prejsť, aby sme sa dostali z mesta $\mathcal{A}$ do mesta $\mathcal{B}$. Vzdialenosť trasy od mesta $\mathcal{A}$ je potom najmenšia zo vzdialeností mesta $\mathcal{A}$ od jednotlivých miest ležiacich na uvažovanej trase. Vzdialenosťou trasy od obsadených miest rozumieme najmenšiu zo vzdialeností medzi trasou a niektorým z obsadených miest alebo nulu pokiaľ niektoré mesto na trase samotnej je obsadené.

Formát vstupu: Prvý riadok vstupného súboru `posol.in` obsahuje čísla N (počet miest) a M (počet ciest). Po ňom nasleduje M riadkov, z ktorých každý obsahuje popis jednej cesty. Cesta je popísaná dvojicou čísel koncových miest. Nasleduje riadok s číslom K (počet obsadených miest) a za ním K riadkov s číslami obsadených miest. Posledný riadok vstupného súboru obsahuje číslo mesta, odkiaľ vychádza posol, a číslo mesta, kde sa nachádza generál.

Formát výstupu: Výstupom programu v súbore `posol.out` sú čísla miest na najlepšej nájdenej trase uvedené v poradí, v akom nimi má posol prechádzať. Všetky čísla miest sú zapísané v jedinom riadku výstupného súboru a sú oddelené medzerami.

Príklad:

Vstup:
posol.in
```
10 12
1 2
2 3
3 4
4 5
2 5
1 6
6 7
7 8
8 5
1 9
9 10
10 5
1
3
1 5
```

Výstup:
posol.out
```
1 9 10 5
```

P-I-3 *(Daniel Kráľ)*

Skupina priateľov sa rozhodla, že v lete podniknú spoločný výlet na bicykloch. Väčšina zvolenej trasy však vedie prírodnou rezerváciou, a preto môžu nocovať len v kempoch. Kempy, v ktorých na svojom výlete prespia, ešte nevybrali.

Celková dĺžka naplánovanej trasy je L ($1 \leq L \leq 1\,000\,000\,000$). Maximálna vzdialenosť, ktorú naši priatelia môžu prejsť za jeden deň, je K, t.j. počas dvoch po sebe nasledujúcich dňoch musia prespať v kempoch vzdialených nanajvýš o K. Na naplánovanej trase sa nachádza celkom N kempov ($0 \leq N \leq 10\,000$); i-ty kemp je vo vzdialenosti l_i od začiatku ich výletu a cena za prespanie v ňom je c_i ($1 \leq c_i \leq 20\,000$). Čísla L, K, l_i a c_i sú celé kladné; všetky l_i sú navzájom rôzne a platí $0 < l_1 < l_2 < \ldots < l_N < L$.

Vašou úlohou je rozhodnúť, či skupina môže naplánovanú trasu prejsť. Ak možno trasu prejsť, potom určte, v ktorých kempoch majú prespať, aby:
a) ich výlet trval čo najmenší počet dní.
b) celková cena za prespanie v kempoch bola čo najmenšia.
Úlohy a) a b) riešte zvlášť; v prípade, že jednu z týchto úloh neviete vyriešiť, riešte len druhú z nich.

Formát vstupu: Vstupný súbor sa nazýva `vylet.in`. V prvom riadku sú čísla L, K a N oddelené medzerou. Na ďalších N riadkoch nasledujú dvojice čísel l_i a c_i oddelených medzerou, postupne pre $i = 1$ až $i = N$.

Formát výstupu: Výstupný súbor sa nazýva `vylet-a.out` pre úlohu a) a `vylet-b.out` pre úlohu b). Ak výlet nemožno uskutočniť tak, aby naši priatelia nikdy neprešli za deň vzdialenosť väčšiu ako K, výstupný súbor obsahuje jediný riadok s vetou „`Trasu nemozno prejst.`“. V opačnom prípade prvý riadok obsahuje dve čísla – M a C. Prvé z nich M ($0 \leq M$) je počet kempov, v ktorých skupina prespí, druhé z nich C je cena, ktorú za prespanie v týchto kempoch zaplatí.

Druhý riadok súboru obsahuje M medzerou oddelených čísel kempov, v ktorých naši priatelia budú nocovať. Kempy sú číslované od jednotky. Pokiaľ je $M = 0$, nemusí byť druhý riadok vôbec uvedený. V prípade, že existuje viac riešení spĺňajúcich podmienku a) alebo b), program má vypísať ľubovoľné jedno z nich.

Príklad:

Vstup:
```
vylet.in
25 5 9
4 2
5 8
8 2
10 8
12 2
15 8
16 2
20 8
24 2
```

Výstup:
```
vylet-a.out
4 32
2 4 6 8
```
```
vylet-b.out
5 16
1 3 5 7 8
```

Vstup:
```
vylet.in
15 10 2
2 11
4 12
```

Výstup:
```
vylet-a.out, vylet-b.out
Trasu nemožno prejsť.
```

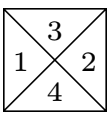
Vstup:
```
vylet.in
8 10 1
7 11
```

Výstup:
```
vylet-a.out, vylet-b.out
0 0
```

P-I-4

(Martin Mareš)

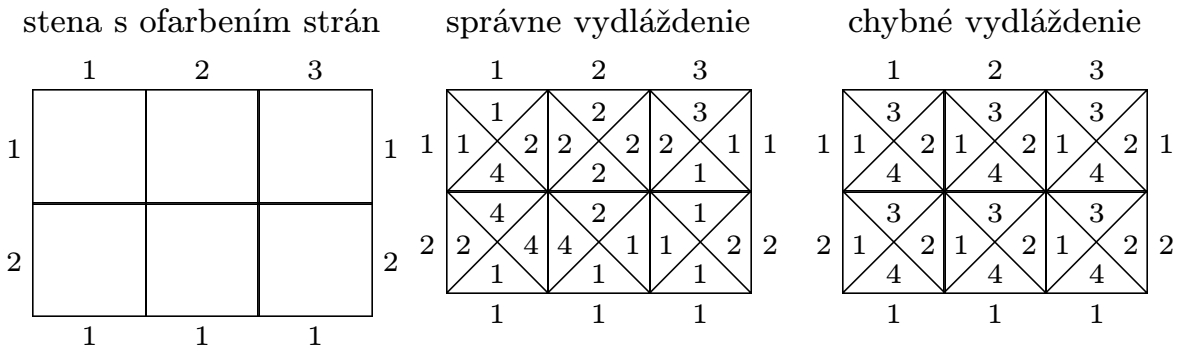
Najprv niekoľko de nícií: *Dlaždice* sú rovnako veľké štvorce s ofarbenými hranami. Konkrétnemu priradeniu farieb hranám dlaždice budeme hovoriť *typ* dlaždice a budeme ho zapisovať ako usporiadanú štvoricu (l, p, h, d) udávajúcu farbu v poradí ľavej, pravej, hornej a dolnej hrany. Aby sme si uľahčili prácu, budeme farby označovať rôznymi symbolmi — písmenami, číslami a podobne. Dlaždica typu $(1, 2, 3, 4)$ bude teda vyzerať nasledovne:



Priestor, ktorý budeme dláždiť (budeme mu hovoriť *stena*), má tvar obdĺžnika o veľkosti $m \times n$ (m aj n sú prirodzené čísla; jednotkou dĺžky nech je dĺžka hrany dlaždice). Strany obdĺžnika sú rozdelené na úseky jednotkovej dĺžky a každému úseku je opäť priradená farba. Našim cieľom je *vydláždiť* stenu dlaždicami tak, aby v každom z $m \cdot n$ jednotkových štvorcov steny bola umiestená práve jedna dlaždica, susedné dlaždice se dotýkali vždy hranami tej istej farby a rovnako

krajné dlaždice priliehali k okraju steny vždy hranou takej farby, akú má aj príslušný úsek okraja steny. Dlaždice nie je povolené otáčať.

Príklad:



Pomocou dláždenia môžeme ľahko riešiť úlohy, ktorých výsledkom je odpoveď ÁNO alebo NIE: zostavíme vhodnú množinu typov dlaždíc (tá je pre daný problém pevná – nezávisí na vstupe), vezmeme vhodne veľkú stenu, jej horný okraj ofarbíme podľa vstupu nášho problému, ostatné okraje ponecháme jednofarebné a budeme sa pýtať, či je túto stenu možné vydláždiť alebo nie. Pritom chceme, aby tento výsledok bol zhodný s riešením našej úlohy.

Aby sme sa nemuseli zaoberať tým, ako presne veľkú stenu máme zvoliť pre ten či onen vstup úlohy, budeme šírku steny voliť vždy rovnakú, ako je dĺžka vstupu (horný okraj teda bude celý zaplnený vstupom), zatiaľ čo výšku steny použijeme najmenšiu, pre ktorú existuje vydláždenie s použitím našej sady dlaždíc.

Keď tento spôsob počítania porovnáme s klasickým programovaním, zistíme, že zvolená sada dlaždíc tvorí v našom modeli niečo podobné programu a potrebná výška steny vzdialene odpovedá dobe behu výpočtu – budeme sa preto snažiť, aby v našich riešeniach bola čo najmenšia.

Formálne povedané, *dlaždicový program* je usporiadaná štvorica

$$D = (T, l_0, p_0, d_0),$$

kde T je konečná množina typov dlaždíc $\{(l_1, p_1, h_1, d_1), \ldots, (l_k, p_k, h_k, d_k)\}$ a l_0, p_0 a d_0 sú okrajové farby. *Rozhodovacou úlohou* $P(x)$ rozumieme úlohu zistiť, či vstup x (konečná postupnosť symbolov, resp. farieb z vopred určenej konečnej množiny) má požadovanú vlastnosť P. Hovoríme, že dlaždicový program *rieši rozhodovaciu úlohu* $P(x)$, ak platí, že $P(x) =$ ÁNO práve vtedy, keď existuje $v > 0$ také, že je možné vydláždiť dlaždicami typov obsiahnutých v množine T stenu veľkosti $|x| \times v$ s hornou hranou ofarbenou vstupom x a ľavou, pravou a dolnou hranou ofarbenou postupne farbami l_0, p_0 a d_0. Z každého typu je možné použiť ľubovoľne mnoho dlaždíc. *Zložitosťou* programu D pre daný vstup x nazveme najmenšie v, pre aké to je možné; pokiaľ také v neexistuje, a teda $P(x) =$ NIE, de nujeme zložitosť ako nulovú. Zložitosť programu je funkcia dĺžky vstupu n, ktorej hodnota udáva maximum zo zložitostí programov pre jednotlivé vstupy tejto dĺžky.

Príklad:

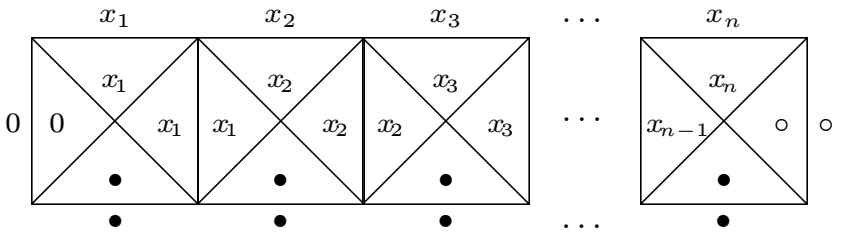
Skúsme teraz skonštruovať dlaždicový program, ktorý bude overovať, či je daná postupnosť tvorená prirodzenými číslami $x_1, \ldots, x_n$ $(0 \leq x_i \leq 9)$ neklesa-

júca. Použijeme dlaždice nasledujúcich typov:

$$T = \left\{ \begin{array}{|c|}\hline x \\ i \diagdown\!\diagup x \\ \bullet \\ \hline\end{array}\ , \begin{array}{|c|}\hline x \\ i \diagdown\!\diagup \circ \\ \bullet \\ \hline\end{array}\ 0 \le i \le x \le 9 \right\},$$

ľavý okraj ofarbíme farbou 0, pravý ∘, dolný farbou • a tvrdíme, že tento program rieši danú úlohu so zložitosťou $O(1)$. To je však treba dokázať.

Predovšetkým si overíme, že každá stena, ktorú je možné vydláždiť dlaždicami typov z množiny T, má jednotkovú výšku. To jasne vyplýva z toho, že spodná hrana každej dlaždice má farbu •, ktorá sa nevyskytuje na žiadnej hornej hrane. Z toho istého dôvodu sa dlaždice majúce na svojej pravej hrane farbu ∘ musia vyskytovať tesne pri pravom okraji steny a nikde inde. Každé korektné dláždenie preto musí vyzerať takto:



čo je však možné práve vtedy, keď $0 \le x_1 \le x_2 \le \ldots \le x_{n-1} \le x_n$, teda keď postupnosť na vstupe je neklesajúca.

Súťažné úlohy:

a) Zostrojte dlaždicový program, ktorý o danej postupnosti núl a jednotiek zistí, či je dvojkovým zápisom nejakého prirodzeného čísla deliteľného piatimi.

b) Zostrojte dlaždicový program, ktorý o danej postupnosti prirodzených čísel $x_1, \ldots, x_n$ $(0 \le x_i \le 9)$ rozhodne, či je nekonštantná (t.j. vydláždenie existuje práve vtedy, keď existujú indexy i, j také, že $x_i \ne x_j$).

SLOVENSKÁ KOMISIA MATEMATICKEJ OLYMPIÁDY
50. ROČNÍK MATEMATICKEJ OLYMPIÁDY
Leták kategórie P

Vydala IUVENTA – zariadenie pre voľný čas detí, mládeže i dospelých MŠ SR
pre vnútornú potrebu Ministerstva školstva SR
Náklad: 350 výtlačkov
Programom TEX sadzbu pripravil Richard Kráľovič
Autori príkladov: Jan Kára
Daniel Kráľ
Martin Mareš
© Slovenská komisia matematickej olympiády, 2000