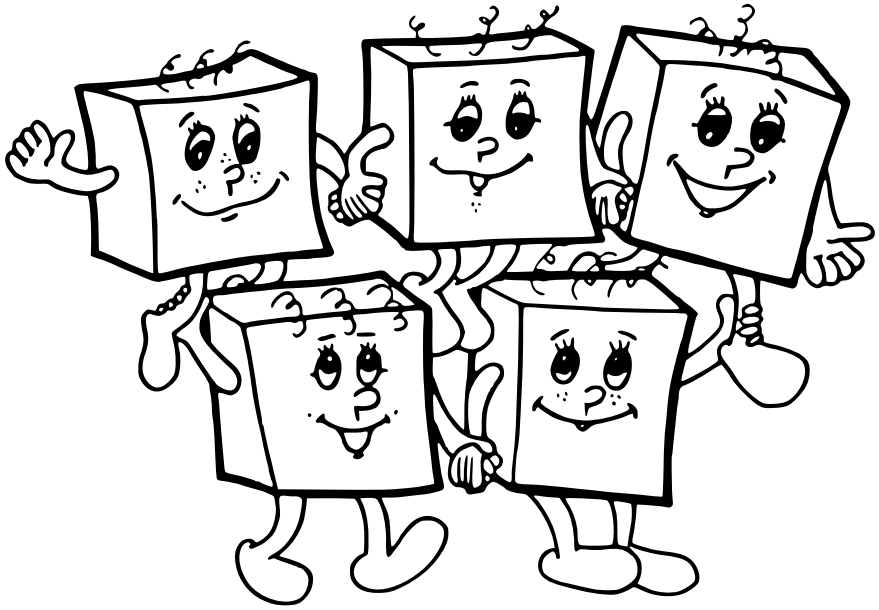
OLYMPIÁDA V INFORMATIKE

NA STREDNÝCH ŠKOLÁCH



dvadsiaty tretí ročník

školský rok 2007/08

zadania krajského kola
kategória A

- **Olympiáda v informatike** je od školského roku 2006/07 samostatnou súťažou. Predchádzajúcich 21 ročníkov tejto súťaže prebiehalo pod názvom **Matematická olympiáda, kategória P** (programovanie).
- Oficiálnu **webstránku** súťaže nájdete na `http://www.ksp.sk/oi/`.

Priebeh krajského kola

Krajské kolo 23. ročníka Olympiády v informatike, kategórie A, sa koná 15. januára 2007 v dopoludňajších hodinách. Na riešenie úloh majú súťažiaci 4 hodiny čistého času. Rôzne úlohy riešia súťažiaci na samostatné listy papiera. Akékoľvek pomôcky okrem písacích potrieb (napr. knihy, výpisy programov, kalkulačky) sú zakázané.

Čo má obsahovať riešenie úlohy?

- Slovne popíšte algoritmus.
  Slovný popis riešenia musí byť jasný a zrozumiteľný i bez nahliadnutia do samotného zápisu algoritmu (do programu).
- Zdôvodnite správnosť vášho algoritmu.
- Uveďte a zdôvodnite jeho časovú a pamäťovú zložitosť.
- Podrobne uveďte dôležité časti algoritmu, ideálne vo forme programu v Pascale alebo C/C++.
- V prípade, že používate vo svojom programovacom jazyku knižnice, ktoré obsahujú implementované dátové štruktúry a algoritmy (napr. STL pre C++), v popise algoritmu stručne vysvetlite, ako by ste napísali program s rovnakou časovou zložitosťou bez použitia knižnice.

Hodnotenie riešení

Za každú úlohu môžete získať od 0 do 10 bodov.

Pokiaľ nie je v zadaní povedané ináč, základným kritériom hodnotenia riešenia je **správnosť** a **efektívnosť** navrhnutého algoritmu. Hodnotí sa aj kvalita popisu riešenia a zdôvodnenie tvrdení o jeho správnosti a efektívnosti.

Efektívnosť algoritmu posudzujeme vypočítaním jeho časovej zložitosti – funkcie, ktorá hovorí, ako dlho vykonanie algoritmu trvá v závislosti od veľkosti vstupných parametrov.

V zadaní úlohy môžu byť uvedené limity na veľkosť premenných. Tieto môžete použiť na odhad toho, ako dobré vaše riešenie je. Na počítači, ktorý vykoná miliardu inštrukcií za sekundu, zrieši vzorové riešenie ľubovoľný povolený vstup do niekoľko sekúnd.

Zadania kategórie A

A-II-1 Parkovanie kočov

Kráľ Drozdia brada vydáva dcéru. Pri tej sláve (a jedle zdarma) nemôže chýbať nijaký šľachtic z okolia. A ako sa tak šľachtici na svojich kočoch vezú na svadbu, vôbec netušia, že sluhom kráľa Drozdej brady spôsobia zaujímavý problém.

Všetky koče totiž treba zaparkovať, a to nie len tak hocijako. Koče treba parkovať do radov. A aby kráľovi nezničili trávnik, musí tých radov byť čo najmenej.

Dvorná etiketa káže, že keď budú hostia odchádzať, musia odchádzať zoradení podľa dôležitosti, najdôležitejší hosť ako posledný.

A to ešte nie je všetko. Koče, ktoré stoja v tom istom rade, musia samozrejme odchádzať v poradí, v akom stoja. A aby predišli zrážkam kočov, chcú ich sluhovia zaparkovať tak, aby pri odchádzaní vždy najskôr odišiel celý jeden rad, až potom začal odchádzať ďalší, a tak ďalej.

Súťažná úloha

Sluhovia presne vedia poradie, v akom budú prichádzať hostia, aj dôležitosť každého z nich. Keď parkujú koč, môžu ho zaparkovať *na začiatok alebo na koniec* hociktorého z už stojacich radov, prípadne ho postaviť do nového radu. Koče musia sluhovia zaparkovať v poradí, v akom hostia prichádzajú.

Nájdite najmenší počet radov, ktorý stačí na to, aby po správnom zaparkovaní vedeli koče odísť v predpísanom poradí.

Formát vstupu

Vstup začína celým číslom N ($1 \leq N \leq 100\,000$) – počet hostí. Nasleduje N *rôznych* kladných celých čísel d_i ($1 \leq d_i \leq 10^9$), udávajúcich dôležitosť hostí v poradí, v akom prichádzajú (väčšie číslo je dôležitejší hosť).

Formát výstupu

Výstup má obsahovať jediný riadok a v ňom jediné celé číslo – potrebný počet radov.

Príklady

vstup

```
10
10 9 11 12 13 8 14 7 6 100
```

výstup

```
1
```

V tomto prípade sa stačí držať pravidla „koče menej dôležité ako 10 idú na začiatok, ostatné na koniec“.

vstup

```
6
12 17 9 23 16 14
```

výstup

```
2
```

Jedno možné riešenie: Koče 12 a 17 pôjdu do rôznych radov. Koč 9 postavíme pred koč 12, koč 23 za koč 17, koč 16 pred koč 17 a nakoniec koč 14 pred koč 16.

vstup

```
12
1 3 2 5 4 7 6 9 8 11 10 12
```

výstup

```
6
```

V jedinom optimálnom riešení budú po zaparkovaní rady: (1 2), (3 4), (5 6), (7 8), (9 10) a (11 12).

Pomalšie riešenia

Maximálne 8 bodov: Riešte tú istú úlohu pre $N \leq 1000$.
Maximálne 6 bodov: Riešte tú istú úlohu pre $N \leq 100$.
Maximálne 4 body: Riešte tú istú úlohu pre $N \leq 10$.

A-II-2 Dlhopisy

Kleofáš nedávno zdedil po bohatej tetičke Anastázii kopu peňazí. Nevedel, čo s nimi, tak sa rozhodol investovať ich do dlhopisov.

Pre jednoduchosť budeme predpokladať nasledujúce skutočnosti:

- Každý typ dlhopisu má svoju pevnú cenu, rovnakú pri kúpe aj predaji.
- Každý typ dlhopisu má pevne daný ročný výnos.
- Každého typu dlhopisu sa dá nakúpiť ľubovoľne veľa.

Uvažujme napríklad nasledujúcu situáciu: Banka ponúka dva typy dlhopisov: Dlhopisy za 4000 korún s ročným výnosom 400 a dlhopisy za 3000 korún s ročným výnosom 250.

Ak má Kleofáš 10 000 korún, najlepšie, čo s nimi môže spraviť, je kúpiť dva dlhopisy po 3000 a jeden za 4000, čím dostane ročný výnos 900 korún.

Keď prejdu dva roky a Kleofáš dostane dvakrát výnosy, bude mať dokopy kapitál 11 800 korún. V tejto chvíli sa mu oplatí jeden dlhopis za 3000 predať a namiesto neho kúpiť dlhopis za 4000. Po treťom roku takto jeho kapitál narastie na 12 850.

Súťažná úloha

Napíšte program, ktorý načíta Kleofášov začiatočný kapitál, ceny a výnosy dlhopisov a počet rokov, a spočíta, koľko najviac peňazí vie mať Kleofáš po uplynutí daného počtu rokov.

Formát vstupu

V prvom riadku je celé číslo K ($1 \leq K \leq 1\,000\,000$) udávajúce Kleofášov štartovný kapitál.

V druhom riadku je počet typov dlhopisov D ($1 \leq D \leq 100$).

V treťom riadku je D dvojíc celých čísel c_i, v_i predstavujúcich ceny a výnosy dlhopisov ($0 < c_i \leq 10^9$, $0 \leq v_i \leq c_i/10$, c_i je násobkom $T = 1000$).

V poslednom riadku je počet rokov R ($1 \leq R \leq 40$).

Časovú zložitosť svojho algoritmu vyjadrite pomocou K, D, T a R.

Formát výstupu

Vypíšte jeden riadok a v ňom maximálnu hodnotu Kleofášovho kapitálu po R rokoch. (Môžete predpokladať, že sa táto hodnota zmestí do bežnej celočíselnej premennej.)

Príklady

vstup

```
10000
2
4000 400 3000 250
4
```

výstup

```
14050
```

Príklad zo zadania. Vo štvrtom roku Kleofáš bude vlastniť 3 dlhopisy po 4000, čím zarobí ďalších 1200.

vstup

```
100000
3
103000 9001 47000 7 83000 100
31
```

výstup

```
112001
```

Kleofáš kúpi jeden dlhopis za 83 000. *Tým za 30 rokov zarobí 3000 korún. Na posledný rok si konečne môže kúpiť prvý dlhopis za* 103 000. *V poslednom roku teda zarobí ďalších* 9001.

vstup

```
100000
3
103000 9001 47000 7 83000 100
37
```

výstup

```
166014
```

Pokračovanie z predchádzajúceho príkladu. Po roku 36 Kleofáš dokúpi dlhopis za 47 000, *čím v poslednom roku zarobí o 7 korún viac.*

Pomalšie riešenia

Maximálne 8 bodov: Riešte pôvodnú úlohu pre $D \leq 10$ a $R \leq 10$.
Maximálne 7 bodov: Riešte pôvodnú úlohu pre $D \leq 10$, $R \leq 10$, $c_i \leq 10^6$.
Maximálne 4 body: Riešte pôvodnú úlohu pre $K \leq 45\,000$ a $R = 1$.

A-II-3 O víťazovi turnaja

Samko sa rozhodol, že zorganizuje programátorský turnaj. Vyzval teda svojich priateľov, aby mu poslali programy, ktoré budú hrať piškvorky. Priatelia nelenili, programy poslali a Samko všetkých pozval na veľký turnaj.

Samko vymyslel pre svoj turnaj jednoduché pravidlá: V každom kole sa náhodne vyžrebujú dva programy, zahrajú si, a ten z nich, ktorý prehrá, z turnaja vypadne.

V noci pred turnajom Samka premohla zvedavosť a pozrel si súboje pre niektoré dvojice programov. Ráno však zistil, že sa tým pripravil o časť prekvapenia. Keďže všetky programy jeho priateľov sú deterministické (t. j. nepoužívajú pri rozhodovaní náhodu), dopadne súboj každých dvoch programov vždy rovnako. Na základe súbojov, ktoré v noci videl, teraz už Samko o niektorých programoch vedel, že turnaj vyhrať nemôžu.

Súťažná úloha

Pre dané výsledky zápasov, ktoré si Samko v noci pozrel, zistite, ktoré programy ešte môžu turnaj vyhrať.

Formát vstupu

V prvom riadku vstupu je jedno celé číslo N ($1 \leq N \leq 100\,000$) – počet programov v turnaji. Programy sú očíslované od 1 do N.

Nasleduje N riadkov, pričom i-ty z nich popisuje zápasy, ktoré vyhrá program i. Presnejšie, i-ty z týchto riadkov začína číslom d_i, ktoré hovorí, koľko zápasov program i v noci vyhral. Nasleduje d_i čísel programov, nad ktorými vyhral. Tieto čísla sú utriedené podľa veľkosti.

Označme $M = d_1 + \cdots + d_n$ počet hier, ktoré si Samko v noci pozrel. Môžete predpokladať, že $0 \leq M \leq 1\,000\,000$. Tiež môžete predpokladať, že vstup je korektný (ak nejaký program x vyhral nad y, tak y nevyhral nad x).

Formát výstupu

Vypíšte jeden riadok a v ňom zoznam čísel programov, ktoré ešte môžu vyhrať turnaj.

Príklady

vstup

```
4
2  2 3
0
1  2
1  2
```

výstup

```
1 3 4
```

Vstup obsahuje nasledujúce údaje: Program 1 určite vyhrá nad programmi 2 a 3. Program 3 určite vyhrá nad programom 2. Program 4 určite vyhrá nad programom 2.

Ako príklad ukážeme, ako môže program 3 vyhrať turnaj. Jedna možnosť je, že v prvom kole vyhrá 3 nad 2, v druhom 4 nad 1 a nakoniec v treťom kole vyhrá 3 nad 4.

vstup

```
5
2  2 3
0
1  2
1  2
0
```

výstup

```
1 2 3 4 5
```

Tentokrát môže turnaj vyhrať ľubovoľný program.
Program 2 vie vyhrať napr. takto: V prvom kole 3 porazí 4, v druhom 5 porazí 3, v treťom 5 porazí 1 a vo štvrtom 2 porazí 5.

Pomalšie riešenia

Maximálne 7 bodov: Riešte pôvodnú úlohu pre $N \leq 1\,000$.
Maximálne 5 bodov: Riešte pôvodnú úlohu pre $N \leq 100$.
Maximálne 4 body: Riešte pôvodnú úlohu pre $N \leq 20$.
Maximálne 3 body: Riešte pôvodnú úlohu pre $N \leq 10$.

A-II-4 Prekladacie stroje

Študijný text nájdete za zadaním súťažnej úlohy. Je zhodný so študijným textom z domáceho kola.

Súťažná úloha

a) *(6 bodov)* Na začiatku máme množinu M_1, ktorá obsahuje práve všetky reťazce z písmen a, b, ktoré obsahujú rovnako veľa písmen a ako b. Teda napr. $abbbaa \in M_1$, ale $aabab \notin M_1$.

Nové množiny môžeme zostrojovať nasledovnými operáciami:
– prekladom nejakej už zostrojenej množiny nejakým strojom (môžeme použiť zakaždým iný stroj)
– zjednotením dvoch už zostrojených množín
– prienikom dvoch už zostrojených množín

Na čo najmenej operácií zostrojte množinu G, ktorá obsahuje práve všetky reťazce z písmen a, b, c, ktoré obsahujú rovnako veľa písmen každého druhu. Teda napr. $aabbcc \in G$, $bac \in G$, ale $abcc \notin G$.

b) *(4 body)* Množina X obsahuje desiatkové zápisy tých kladných celých čísel, ktoré obsahujú rovnako veľa cifier 1 a 2. Teda napr. $1122 \in X$, $21231 \in X$, $47 \in X$, ale $112 \notin X$ a $031221 \notin X$ (desiatkový zápis nemôže obsahovať nulu na začiatku).

Množina Y obsahuje desiatkové zápisy kladných celých čísel deliteľných 7. Teda napr. $140 \in Y$, $7707 \in Y$, ale $47 \notin Y$ a $07 \notin Y$.

Zostrojte prekladací stroj, ktorý preloží množinu X na Y, alebo dokážte, že takýto prekladací stroj neexistuje.

Študijný text

Prekladací stroj je stroj, ktorý dostáva ako vstup reťazec znakov. Tento reťazec postupne číta a podľa vopred zvolenej sady pravidiel (teda svojho programu) občas nejaké znaky zapíše na výstup. Po tom, ako stroj spracuje celý vstupný reťazec a úspešne skončí výpočet, zoberieme reťazec znakov zapísaný na výstup a nazveme ho **prekladom** vstupného reťazca.

Výpočet stroja nemusí byť jednoznačne určený. Inými slovami, sada pravidiel môže niekedy stroju umožniť rozhodnúť sa. V takomto prípade sa môže stať, že niektorý reťazec bude mať viac rôznych prekladov.

Naopak, môže sa stať, že v nejakej situácií sa podľa danej sady pravidiel nebude dať v preklade pokračovať vôbec. V takomto prípade sa môže stať, že niektorý reťazec nebude mať vôbec žiadny preklad.

Formálnejšia definícia prekladacieho stroja

Každý prekladací stroj pracuje nad nejakou vopred zvolenou konečnou množinou znakov. Túto budeme volať **abeceda** a značiť Σ. V súťažných úlohách bude vždy Σ známa zo zadania úlohy. Abeceda nikdy nebude obsahovať znak \$, ten budeme používať na označenie konca vstupného reťazca.

Stroj si môže počas prekladu reťazca pamätať konečne veľkú informáciu. Formálne toto definujeme tak, že stroj sa v každom okamihu prekladu nachádza v jednom z *konečne veľa* stavov. Nutnou súčasťou programu prekladacieho stroja je teda nejaká konečná **množina stavov**, v ktorých sa môže nachádzať. Túto množinu označíme K. Okrem samotnej množiny stavov je taktiež potrebné uviesť, v ktorom stave sa stroj nachádza na začiatku každého prekladu. Tento stav budeme volať **začiatočný stav**.

Program stroja sa bude skladať z konečného počtu prekladových pravidiel. Každé pravidlo je štvorica (p, u, v, q), kde $p, q \in K$ sú nejaké dva stavy a u, v sú nejaké dva reťazce znakov z abecedy Σ.

Stavy p a q môžu byť aj rovnaké. Reťazec u môže ako svoj jediný znak obsahovať znak \$. Reťazce u a v môžu byť aj rovnaké a môžu byť aj prázdne. Aby sa program ľahšie čítal, budeme namiesto prázdneho reťazca písať ε.

Význam takéhoto pravidla je nasledujúci: „Ak je stroj práve v stave p a neprečítaná časť vstupu začína reťazcom u, môže tento reťazec zo vstupu prečítať, na výstup zapísať reťazec v a zmeniť stav na q.“ Všimnite si, že pravidlo tvaru (p, ε, v, q) môžeme použiť vždy, keď sa stroj nachádza v stave p, bez ohľadu na to, aké znaky ešte zostávajú na vstupe.

Ešte potrebujeme povedať, kedy preklad úspešne skončil. V prvom rade budeme vyžadovať, aby prekladací stroj dočítal celý vstupný reťazec. Okrem toho umožníme stroju „povedať“, či sa mu preklad podaril alebo nie. Toto spravíme tak, že niektoré stavy stroja nazveme **ukončovacie stavy**. Množinu ukončovacích stavov budeme značiť F.

Úplne formálna definícia prekladacieho stroja

Prekladací stroj je usporiadaná pätica (K, Σ, P, q_0, F), kde Σ a K sú *konečné* množiny, $q_0 \in K$, $F \subseteq K$ a P je *konečná* množina prekladových pravidiel popísaných vyššie. Presnejšie, nech Σ^* je množina všetkých reťazcov tvorených znakmi zo Σ, potom P je *konečná* podmnožina množiny $K \times (\Sigma^* \cup \{\$\}) \times \Sigma^* \times K$.

(Pre každé $q \in K$ budeme množinu pravidiel, ktorých prvá zložka je q, volať „prekladové pravidlá zo stavu q“.)

Ak teda chceme definovať konkrétny prekladací stroj, musíme uviesť všetkých päť vyššie uvedených objektov.

Keď už máme definovaný konkrétny stroj $A = (K, \Sigma, P, q_0, F)$, môžeme definovať, ako tento stroj prekladá konkrétny reťazec. Povieme najskôr formálnu definíciu, potom ju slovne vysvetlíme.

Množina **platných prekladov** reťazca u prekladacím strojom A je:

$$A(u) = \Big\{ \quad v \quad \Big| \quad \begin{array}{l} \exists n \geq 0; \quad \exists (p_1, u_1, v_1, r_1), \ldots, (p_n, u_n, v_n, r_n) \in P; \\ (\forall i \in \{1, \ldots, n-1\};\ r_i = p_{i+1}) \ \wedge \ p_1 = q_0 \ \wedge \ r_n \in F \ \wedge \\ \wedge \ \exists k \geq 0;\ u_1 u_2 \ldots u_n = u \underbrace{\$ \ldots \$}_{k} \ \wedge \ v_1 v_2 \ldots v_n = v \end{array} \Big\}$$

Definícia hovorí, kedy je reťazec v prekladom reťazca u. Vysvetlíme slovne význam jednotlivých riadkov definície:

- Prvý riadok hovorí, že aby sa dalo u preložiť na v, musí existovať nejaká postupnosť prekladových pravidiel, ktorú pri tomto preklade použijeme. Zvyšné dva riadky popisujú, ako táto postupnosť musí vyzerať.

- Druhý riadok zabezpečuje, aby stavy v použitých pravidlách boli správne: Prvé pravidlo musí byť pravidlom zo začiatočného stavu, každé ďalšie pravidlo musí byť pravidlom z toho stavu, do ktorého sa stroj dostal použitím predchádzajúceho pravidla.

  Navyše stav, v ktorom bude stroj na konci výpočtu, musí byť ukončovací.

- Posledný riadok hovorí o reťazcoch, ktoré stroj pri použití dotyčných prekladových pravidiel prečíta a zapíše.

  Reťazec, ktorý pri použití týchto pravidiel stroj prečíta zo vstupu, musí byť naozaj zadaný reťazec u, prípadne sprava doplnený vhodným počtom znakov \$.

  Reťazec, ktorý stroj zapíše na výstup, musí byť presne reťazec v.

Na čo budeme používať prekladacie stroje?

Nám budú prekladacie stroje slúžiť na preklad jednej množiny reťazcov na inú množinu reťazcov. Ak A je prekladací stroj a $M \subseteq \Sigma^*$ nejaká množina reťazcov, tak preklad množiny M strojom A je množina $A(M) = \bigcup_{u \in M} A(u)$.

Inými slovami, výslednú množinu $A(M)$ zostrojíme tak, že zoberieme všetky reťazce z M a pre každé z nich dáme do $A(M)$ všetky jeho platné preklady.

Príklad 1

Majme abecedu $\Sigma = \{0, \ldots, 9\}$. Nech M je množina všetkých tých reťazcov, ktoré predstavujú zápisy kladných celých čísel v desiatkovej sústave. Zostrojíme prekladací stroj A, pre ktorý bude platiť, že prekladom tejto množiny M bude množina zápisov všetkých kladných celých čísel, ktoré sú deliteľné tromi.

Riešenie

Najjednoduchšie bude jednoducho vybrať z M tie čísla, ktoré sú deliteľné tromi. Náš prekladací stroj bude kopírovať cifry zo vstupu na výstup, pričom si bude v stave pamätať, aký zvyšok po delení tromi dáva doteraz prečítané (a zapísané) číslo. Ak je po dočítaní vstupu v stave zodpovedajúcom zvyšku 0, prejde do ukončovacieho stavu.

Formálne A bude pätica $(K, \Sigma, P, 0, F)$, kde $K = \{0, 1, 2, end\}$, $F = \{end\}$ a prekladové pravidlá vyzerajú nasledovne:

$$P = \left\{(x, y, y, z) \;\middle|\; x \in \{0, 1, 2\} \wedge y \in \Sigma \wedge z = (10x{+}y) \bmod 3\right\} \cup \left\{(0, \$, \varepsilon, end)\right\}$$

Príklad 2

Majme abecedu $\Sigma = \{a, e, i, \bullet, \blacksquare\}$. Zostrojíme prekladací stroj B, pre ktorý bude platiť, že prekladom ľubovoľnej množiny M, ktorá obsahuje len reťazce z písmen a, e a i, bude množina tých istých reťazcov v morzeovke (bez oddeľovačov medzi znakmi). Zápisy našich písmen v morzeovke sú nasledujúce: a je $\bullet\blacksquare$, e je $\bullet$ a i je $\bullet\bullet$.

Napríklad množinu $M = \{ae, eea, ia\}$ by náš stroj mal preložiť na množinu $\{\bullet\blacksquare\bullet, \bullet\bullet\bullet\blacksquare\}$. (Všimnite si, že reťazce eea a ia majú v morzeovke bez oddeľovačov rovnaký zápis.)

Riešenie

Prekladací stroj B bude jednoducho čítať vstupný reťazec po znakoch a zakaždým zapíše na výstup kód prečítaného znaku.

Formálne B bude pätica $(K, \Sigma, P, \heartsuit, F)$, kde $K = \{\heartsuit\}$, $F = \{\heartsuit\}$ a prekladové pravidlá vyzerajú nasledovne:

$$P = \{(\heartsuit, a, \bullet\blacksquare, \heartsuit),\ (\heartsuit, e, \bullet, \heartsuit),\ (\heartsuit, i, \bullet\bullet, \heartsuit)\}$$

Všimnite si, že nepotrebujeme explicitne kontrolovať, či sme na konci vstupu. Totiž počas celého prekladu sme v ukončovacom stave, a teda akonáhle prečítame posledný znak zo vstupu, je zapísaný preklad platný.

Príklad 3

Majme abecedu $\Sigma = \{a, e, i, \bullet, \blacksquare\}$. Zostrojíme prekladací stroj C, pre ktorý bude platiť, že prekladom ľubovoľnej množiny M, ktorá obsahuje len reťazce zo znakov $\bullet$ a $\blacksquare$, bude množina **všetkých** reťazcov z písmen a, e a i, ktoré keď zapíšeme v morzeovke (bez oddeľovačov medzi znakmi), dostaneme reťazec z M. Napr. množinu $M = \{\bullet\blacksquare\bullet, \bullet\bullet\bullet\blacksquare\}$ by náš stroj mal preložiť na $\{ae, eea, ia\}$.

Riešenie

Nášmu prekladaciemu stroju dáme možnosť sa v každom okamihu prekladu rozhodnúť, že ide čítať kód nejakého písmena a zapísať na výstup toto písmeno. Potom každej možnosti, ako rozdeliť vstupný reťazec na kódy písmen, bude zodpovedať jeden platný preklad.

Formálne C bude pätica $(K, \Sigma, P, \diamondsuit, F)$, kde $K = \{\diamondsuit\}$, $F = \{\diamondsuit\}$ a prekladové pravidlá $P = \{(\diamondsuit, \bullet\blacksquare, a, \diamondsuit),\ (\diamondsuit, \bullet, e, \diamondsuit),\ (\diamondsuit, \bullet\bullet, i, \diamondsuit)\}$.

Ako príklad si ukážeme, ako mohol prebiehať preklad reťazcov z vyššie uvedenej množiny M. Sú tieto tri možnosti:
$(\diamondsuit, \bullet\blacksquare, a, \diamondsuit), (\diamondsuit, \bullet, e, \diamondsuit)$
$(\diamondsuit, \bullet\bullet, i, \diamondsuit), (\diamondsuit, \bullet\blacksquare, a, \diamondsuit)$
$(\diamondsuit, \bullet, e, \diamondsuit), (\diamondsuit, \bullet, e, \diamondsuit), (\diamondsuit, \bullet\blacksquare, a, \diamondsuit)$
Ak by sme skúsili na ľubovoľnom vstupe z M používať prekladové pravidlá v inom poradí – napr. na vstupe $\bullet\bullet\bullet\blacksquare$ použiť trikrát pravidlo $(\diamondsuit, \bullet, e, \diamondsuit)$ – nepodarí sa nám dočítať vstupný reťazec do konca.

Príklad 4

Majme abecedu $\Sigma = \{a, b, c\}$. Nech množina X obsahuje práve všetky tie reťazce, v ktorých je rovnako a a b. Teda napr. $abbccac \in X$, ale $cbaa \notin X$.

Nech množina Y obsahuje práve všetky tie reťazce, ktoré neobsahujú žiadne a, neobsahujú podreťazec bc, a písmen c je dvakrát toľko ako písmen b. Teda napr. $ccccbb \in Y$, ale $cccbcb \notin Y$ a $acacba \notin Y$.

Zostrojíme prekladací stroj D, ktorý preloží X na Y.

Riešenie

Budeme prekladať len niektoré vhodné reťazce z množiny X, presnejšie to budú tie, v ktorých nie sú žiadne c a všetky a idú pred všetkými b. Takýto reťazec preložíme tak, že najskôr každé a prepíšeme na cc, a potom skopírujeme na výstup všetky b. Teda napr. prekladom slova $aabb$ bude slovo $ccccbb$.

Formálne D bude pätica $(K, \Sigma, P, \textbf{čítaj-a}, F)$, kde $K = \{\textbf{čítaj-a}, \textbf{čítaj-b}\}$, $F = \{\textbf{čítaj-b}\}$ a prekladové pravidlá vyzerajú nasledovne:
$P = \{(\textbf{čítaj-a}, a, cc, \textbf{čítaj-a}),\ (\textbf{čítaj-a}, b, b, \textbf{čítaj-b}),\ (\textbf{čítaj-b}, b, b, \textbf{čítaj-b})\}$

Prečo tento prekladací stroj funguje? Ak vstupný reťazec obsahuje nejaké písmeno c, pri jeho prekladaní sa pri prvom výskyte c náš stroj zasekne. Preto takéto reťazce nemajú žiaden platný preklad. Podobne nemajú platný preklad reťazce, kde nie je dodržané poradie písmen a a b. Totiž akonáhle náš stroj prečíta prvé b, prejde do stavu **čítaj-b**, a ak sa teraz ešte na vstupe vyskytne a, stroj sa zasekne.

Platné preklady teda existujú naozaj len pre slová vyššie popísaného tvaru. Je zjavné, že z každého z nich vyrobíme nejaký reťazec z Y, preto $D(X) \subseteq Y$. A naopak, ak si povieme konkrétne slovo z Y, ľahko nájdeme slovo z X, ktoré sa naň preloží, preto $Y \subseteq D(X)$, a teda $Y = D(X)$.

(táto strana je úmyselne prázdna)

SLOVENSKÁ KOMISIA OLYMPIÁDY V INFORMATIKE

DVADSIATY TRETÍ ROČNÍK OLYMPIÁDY V INFORMATIKE

Zodpovedný redaktor: Michal Forišek
Sadzba programom LaTeX