35. ročník matematické olympiády na středních školách

Kategória P

In: František Zítek (editor); Leo Boček (editor); Karel Horák (editor); Jozef Hvorecký (editor); Branislav Rovan (editor): 35. ročník matematické olympiády na středních školách. Zpráva o třetím kole soutěže konané ve školním roce 1985/86. 27. mezinárodní matematická olympiáda. (Slovak). Praha: Státní pedagogické nakladatelství, 1988. pp. 134–180.

Kategória P

ÚLOHY I. KOLA

P - I - 1

Máme presne N lístkov papiera. Na každý lístok napíšeme z každej strany jedno prirodzené číslo od 1 po N (čísla na opačných stranách lístka môžu byť rôzne) tak, že každé číslo je napísané práve dva razy.

a) Dokážte, že pre ľubovoľný spôsob rozpísania čísel na lístky je možné poukladať lístky na stôl tak, že na ich vrchnej strane vidíme práve čísla 1 až N.

b) Nájdite algoritmus (predpis pre človeka), pomocou ktorého je možné poukladať lístky tak, aby spĺňali podmienku z bodu a). Algoritmus zapíšte v prirodzenej reči za pomoci matematických symbolov a označení tak, ako by ste ho vysvetľovali (zasvätenému) kamarátovi. Dávajte však pozor na to, aby bol presný, jednoznačný a logický. Dokážte, že algoritmus pracuje pre ľubovoľné rozpísanie čísel.

Riešenie. a) Dokážeme, že lístky možno vždy pootáčať tak, aby sme videli všetky čísla od 1 po N, matematickou indukciou vzhľadom na N.

1. Keď $N = 1$, lístok môžeme položiť ľubovoľne, t. j. úloha je vyriešená.
2. Ukážeme, že keď vieme rozložiť N lístkov s ľubovoľným rozpísaním čísel, tak vieme rozložiť aj $N + 1$ lístkov.

V jednoduchšom prípade je číslo $N + 1$ na oboch stranách jedného lístka. V tom prípade rozmiestnime najprv lístky

s číslami 1 až N a nakoniec N + 1-vý lístok položíme ľubovoľnou stranou navrch.

V opačnom prípade je N + 1 na dvoch rôznych lístkoch. Tieto dva lístky »zlepíme« číslami N + 1 k sebe. Tým vznikne sada N lístkov s číslami 1, 2, ..., N, ktorú podľa indukčného predpokladu vieme rozložiť. Potom zlepený lístok rozlepíme a na stol položíme oba lístky tak, aby ten, ktorý bol v zlepenej dvojici »dolu«, mal na hornej strane číslo N + 1 a ten, ktorý bol »hore«, mal na hornej strane to isté číslo ako vtedy, keď boli lístky zlepené. Tým sme požadovaným spôsobom rozložili všetkých N + 1 lístkov.

b) Na základe predchádzajúceho dôkazu možno navrhnúť nasledujúci algoritmus:

1. Označ všetky lístky ako nezaradené.
2. Medzi nezaradenými lístkami vyhľadaj lístky s číslom N.
3. Ak si našiel len jeden lístok (t. j. číslo je na jeho oboch stranách), tak ho polož medzi zaradené, pokračuj krokom 5.
4. Ak si našiel dva lístky, potom ich polož na seba tak, aby číslo N bolo na vnútornej strane »zlepenia«. Takto vzniknutú dvojicu lístkov považuj za jeden a polož ho medzi nezaradené.
5. Zníž N o 1 a ak N je väčšie ako 0, pokračuj krokom 2.
6. Teraz sú všetky lístky zaradené. Tie, ktoré sa skladajú z viacerých na sebe položených lístkov, rozlož na jednotlivé lístky tak, aby sa pritom žiaden z nich neprevrátil. Vtedy sa bude každé číslo nachádzať na vrchnej strane práve raz, čo sme chceli dosiahnuť. Tým teda algoritmus končí. Uvedený algoritmus pracuje presne podľa myšlienky uvedenej v predchádzajúcom dôkaze, takže ďalšie dokazovanie je zbytočné.

Def. 1: Konečnú postupnosť A(0), A(1), ..., A(N), kde N je prirodzené číslo a platí $A(i) = 0$ alebo 1 pre všetky i, nazveme »dobrou«, ak pre všetky i platí

$$A(i) = 0 \text{ alebo } A(i + 1) = 0.$$

Def. 2: Nekonečná postupnosť prirodzených čísel $F(i)$ je definovaná vzťahmi:

$$F(0) = F(1) = 1$$

$$F(i + 2) = F(i + 1) + F(i) \quad \text{pre } i \geqq 0$$

Def. 3: Ku každej konečnej postupnosti A

$$A(0), A(1), \ldots, A(N)$$

priradíme číslo H(A, N) podľa predpisu

$$H(A, N) = A(0).F(0) + A(1).F(1) + \ldots + A(N).F(N).$$

Def. 4: Dve konečné postupnosti A:

$$A(0), A(1), \ldots, A(N)$$

a B:

$$B(0), B(1), \ldots, B(M)$$

nazveme príbuznými, keď platí

$$H(A, N) = H(B, M).$$

Nájdite algoritmus, ktorý pre danú konečnú postupnosť $A(i)$, $i = 0, 1, \ldots, N$ nájde príbuznú »dobrú« postupnosť $B(j)$, $j = 0, 1, \ldots, M$. Pritom predpokladajte, že nie je možné vypočítať hodnotu $H(A, N)$, lebo by mohla byť príliš veľká.

Dokážte, že pre každú zadanú postupnosť dá algoritmus požadovaný výsledok.

Riešenie. V texte riešenia sa bude všade namiesto termínu »postupnosť núl a jedničiek $A(i)$, $i = 0, 1, \ldots, N$« používať termín »postupnosť A«. Podobne miesto »postupnosť núl a jedničiek $B(j)$, $j = 0, 1, \ldots, M$« budeme písať »postupnosť B«.

Algoritmus

1. Skopíruj postupnosť A do postupnosti B, tj. vytvor novú postupnosť B takú, že

$$A(i) = B(i) \text{ pre } i = 0, 1, \ldots, N,$$

   a polož $M = N$.
2. Ak je postupnosť B dobrá, koniec.
3. Nech k je najväčšie také číslo, že

$$B(k) = B(k - 1) = 1.$$

4. Ak $k = M$, polož $M = M + 1$ a $B(M) = 0$.
5. Polož $B(k) = B(k - 1) = 0$, polož $B(k + 1) = 1$, pokračuj krokom 2.

Dôkaz. Dôkaz rozložíme na dve časti. Najprv dokážeme, že ak výpočet algoritmu skončí, tak dá požadovaný výsledok (táto vlastnosť algoritmu sa nazýva *čiastočná správnosť*). Potom dokážeme, že algoritmus skončí pre ľubovoľnú konečnú postupnosť A (táto vlastnosť sa nazýva *konečnosť* algoritmu).

Čiastočná správnosť. Treba dokázať, že ak výpočet skončí (v kroku 2), tak postupnosť B bude »dobrá« a zároveň postupnosti A a B budú príbuzné.

a) To, že po skončení výpočtu bude postupnosť B »dobrá«, je zrejmé. Algoritmus končí v kroku 2 iba za predpokladu, že B je »dobrá« postupnosť.

b) Teraz stačí dokázať, že postupnosti A a B sú príbuzné vždy, keď vykonávame príkaz 2. Platia dve tvrdenia, z ktorých príbuznosť A a B vyplýva podľa matematickej indukcie:

1. Keď do 2 prídeme z 1, tak sú postupnosti A a B určite príbuzné, lebo sú identické.
2. Ak sú pri vykonávaní 2. kroku algoritmu postupnosti A a B príbuzné, tak budú príbuzné aj po vykonaní cyklu pozostávajúceho z krokov 3, 4, 5, 2 v uvedenom poradí. To vyplýva zo skutočnosti, že
   - príkaz 3 sa vykonáva len v tom prípade, keď postupnosť B nie je »dobrá«. Teda zaručene existuje také k, že

$$B(k) = B(k - 1) = 1,$$

   - príkaz 4 nezmení hodnotu H(B, M),
   - pred vykonaním 5 platí, že $B(k + 1) = 0$. V opačnom prípade by k nebolo najväčšie číslo, pre ktoré $B(k) = B(k - 1) = 1$.

— V príkaze 5 sa hodnota H(B, M) nezmení, lebo pre členy postupnosti F platí

$$F(k + 1) = F(k) + F(k - 1).$$

Konečnosť. Činnosť algoritmu skončí, keď je zaručené, že skončí vykonávanie každého cyklu, ktorý obsahuje. Náš algoritmus obsahuje jediný cyklus 2—5.

V tomto cykle sa pri každom prechode zaručene vykoná príkaz 5. V ňom sa dve jednotky v postupnosti B nahradia jednou — teda pri každom prechode cyklom sa počet jednotiek o jednu zmenší. Toto znižovanie nemôže trvať nekonečne dlho, lebo postupnosť, ktorá obsahuje iba jednu jednotku, je zaručene dobrá. Cyklus teda skončí najneskôr po N krokoch.

Zhrnutie. Podarilo sa dokázať, že

— ak sa výpočet podľa algoritmu skončí, tak dá správny výsledok pre ľubovoľnú vstupnú postupnosť A;

— výpočet podľa algoritmu skončí pre každú postupnosť A.

Z toho vyplýva, že výpočet skončí a dá správny výsledok pre ľubovoľnú vstupnú postupnosť A.

Ešte poznamenávame, že takto formulovaný algoritmus je dostatočne presný pre »ručné« spracovanie. Pre výpočet na počítači by bolo treba presnejšie špecifikovať krok 3. Hľadanie čísla k totiž nemusí vždy začínať od konca, ale od toho miesta, kde sme naposledy našli dve jednotky vedľa seba (túto hodnotu udáva premenná i). Keď zameníme dve jednotky za jednu, stačí preveriť, či sme tým nevytvorili novú dvojicu. Dôkaz správnosti takto modifikovaného algoritmu by bol podobný, iba by obsahoval viac technických detailov.

Na záver uvádzame takto upravený algoritmus v jazyku Basic:

```
110 FOR I = 1 TO N
120 LET B(I) = A(I)
130 NEXT I
140 M = N
150 FOR I = N TO 2 STEP -1
160       LET K = I
170       IF B(K) = 0 OR B(K - 1) = 0 THEN GOTO
          240
180       IF K = M THEN LET M = M + 1
190       B(K) = 0
200       B(K - 1) = 0
210       B(K + 1) = 1
220       K = K + 2
230       GO TO 170
240 NEXT I
```

P - I - 3

Nasledujúci program v jazyku Basic hľadá v N-prvkovom poli čísel úsek, ktorého súčet je maximálny:

```
100 LET M = 0
110 FOR DLZ = 1 TO N
120       FOR ZAC = 1 TO N - DLZ + 1
130             LET S = 0
140             FOR I = ZAC TO ZAC + DLZ - 1
150                   LET S = S + A(I)
160             NEXT I
170             IF S > M THEN LET M = S
180       NEXT ZAC
190 NEXT DLZ
200 END
```

Označme S(N) počet vykonaní priraďovacieho príkazu v riadku 150. Vyjadrite S(N) ako funkciu zadaného N.

Riešenie. V ďalšom texte budeme namiesto »cyklus pre DLZ v riadkoch 110 až 190« písať »cyklus pre DLZ« (a podobne pre ZAC a I).

Program začneme analyzovať od najvnútornejšieho cyklu. Cyklus pre I sa vykoná DLZ-krát, pričom v každom priebehu cyklom sa vykoná práve jedno sčítanie. Teda v celom cykle pre I sa vykoná DLZ sčítaní. Cyklus pre ZAC sa vykonáva N − DLZ + 1 razy, teda v celom cykle pre ZAC sa vykoná

$$(N - DLZ + 1).DLZ$$

sčítaní. Hľadaný počet - funkciu S(N) zistíme, keď vyjadríme počet sčítaní v cykle pre DLZ.

Cyklus pre DLZ sa vykonáva od 1 po N. Teda počet sčítaní S(N) je vyjadrený sumou:

$$S(N) = (N - 1 + 1).1 + (N - 2 + 1).2 + \ldots +$$

$$+ (N - N + 1).N$$

Roznásobením každej zátvorky dostávame vzťah:

$$S(N) = (N + 1).1 - 1.1 + (N + 1).2 - 2.2 +$$

$$+ \ldots + (N + 1).N - N.N$$

Jednotlivé členy združíme takto:

$$S(N) = (N + 1).(1 + 2 + 3 + \ldots + N) -$$

$$- (1^2 + 2^2 + 3^2 + \ldots + N^2) \qquad (S)$$

Teraz vyjadríme zvlášť každý člen rozdielu.

Prvý člen obsahuje súčet aritmetickej postupnosti s diferenciou 1. Teda

$$(N + 1).(1 + 2 + 3 \ldots + N) = (N + 1).N.(N + 1)/2 \qquad (S1)$$

Druhý člen (súčet druhých mocnín) označíme ako D(N). Zo známej rovnosti

$$A^3 - B^3 = (A - B).(A^2 + A.B + B^2)$$

vieme, že

$$1^3 - 0^3 = 1.(1^2 + 1.0 + 0^2)$$

$$2^3 - 1^3 = 1.(2^2 + 2.1 + 1^2)$$

$$3^3 - 2^3 = 1.(3^2 + 3.2 + 2^2)$$

$$\ldots \quad \ldots \quad \ldots$$

$$N^3 - (N - 1)^3 = 1.(N^2 + N.(N - 1) + (N - 1)^2.$$

Sčítaním všetkých týchto rovností dostávame:

$$N^3 - 0^3 =$$

$$= D(N) + (1.0 + 2.1 + 3.2 + \ldots + N.(N - 1)) +$$

$$+ D(N - 1)$$

Vieme, že platí

$$N.(N - 1) = N^2 - N$$

a

$$D(N) = D(N - 1) + N^2.$$

Preto

$$N^3 = D(N) + (D(N) - (1 + 2 + 3 \ldots + N)) +$$

$$+ D(N) - N^2.$$

Už sme ukázali, že

$$1 + 2 + 3 + \ldots + N = N.(N + 1)/2,$$

takže

$$N^3 = 3.D(N) - N^2 - N.(N + 1)/2.$$

Z tohoto vzťahu určíme D(N):

$$3.D(N) = N.(N^2 + N + (N + 1)/2) =$$

$$= N.(N + 1)(2N + 1)/2$$

$$D(N) = \frac{N.(N + 1).(2N + 1)}{6} \qquad (S2)$$

Po dosadení vzťahov (S1) a (S2) do (S) dostávame

$$S(N) = N(N + 1)(N + 1)/2 - N(N+1)(2N + 1)/6,$$

a teda

$$S(N) = \frac{N^3 + 3N^2 + 2N}{6}.$$

Poznámka. Vzťah

$$1^2 + 2^2 + 3^2 + \ldots + N^2 = N.(N + 1).(2N + 1)/6$$

stačilo nájsť v literatúre a dokázať indukciou. My sme sa snažili podať jeho konštruktívny dôkaz, lebo môže poslúžiť ako návod na odvodenie podobných súčtov v budúcnosti.

P - I - 4

Modifikovaný Minského stroj (MMS) je teoretický model jednoduchého počítača. Pozostáva z konečného, dostatočne veľkého počtu pomenovaných premenných (mená sú obyčajné identifikátory) a »riadiacej jednotky«. Obsahom každej premennej je *nezáporné celé* číslo. Riadiaca jednotka dokáže manipulovať s premennými, ale vykonáva len veľmi jednoduché operácie:

– pripočítať k premennej jednotku, čo označujeme »meno premennej« +,
– odčítať od premennej jednotku, čo označujeme »meno premennej« −,
– testovať obsah premennej na rovnosť nule a podľa výsledku riadiť ďalší výpočet. Existujú dva testy:
»meno premennej« = 0,
resp.
»meno premennej« ≠ 0.

Poznamenávame, že
– rozsah čísel nie je zhora ohraničený,
– od premennej s hodnotou nula môžete odčítať jednotku; výsledkom je nula v premennej (t. j. jej hodnota sa nezmení),
– MMS nepoužíva žiaden príkaz vstupu ani výstupu. Všetky údaje potrebné k výpočtu musia byť v premenných dané vopred a výpočet skončí s výsledkami vo vopred určených premenných.

Modifikovaný Minského stroj sa programuje tak, že sa postupne vytvárajú predpisy na stále zložitejšie činnosti. Takýto postup predvedieme:

Činnosť »NULUJ X«
rob pokiaľ X ≠ 0
rob X− **koniec**
koniec

Táto činnosť pozostáva len z elementárnych akcií, známych MMS. Avšak zo skôr definovaných činností možno skladať komplikovanejšie; popíšeme činnosť, ktorá presunie obsah premennej A do premennej B a súčasne premennú A vynuluje:

činnosť »PRESUŇ A → B«
rob NULUJ B; ... B sa dosadilo za X pomocou skôr definovanej činnosti
pokiaľ A ≠ 0
rob A−; B+ **koniec**
koniec

Takto by sme mohli pokračovať a naprogramovať veľmi zložité činnosti. Jazyk na zápis týchto činností stručne definujeme:

— príkazmi sú elementárne činnosti, napr. A−, G+ atď;
— podmienky smú byť len typu »premenná = 0« alebo »premenná ≠ 0«, logické operácie AND, OR atď. nie sú k dispozícii;
— z riadiacich štruktúr sú povolené nasledujúce:
— *zložený príkaz*
rob
⟨postupnosť príkazov oddelených bodkočiarkami⟩
koniec
— *príkaz vetvenia*
ak ⟨podmienka⟩ **tak** ⟨príkaz⟩ **inak** ⟨príkaz⟩
resp.
ak ⟨podmienka⟩ **tak** ⟨príkaz⟩
— dva *príkazy cyklu* (s podmienkou na začiatku a s podmienkou na konci)
pokiaľ ⟨podmienka⟩
⟨zložený príkaz⟩
Vykonávanie tela cyklu (zloženého príkazu, obsahujúceho povinne zátvorky **rob** — **koniec**) sa opakuje tak dlho, pokiaľ podmienka cyklu platí. Teda vykonávanie skončí vtedy, keď podmienka pri testovaní je prvýkrát neprav-

divá. Druhým príkazom cyklu je

opakuj

⟨zoznam príkazov, oddelených bodkočiarkami⟩

kým ⟨podmienka⟩

ktorého vykonávanie sa skončí, keď podmienka prvýkrát platí (t. j. návrat na začiatok cyklu sa uskutoční, keď podmienka neplatí).

Ak chcete použiť cyklus FOR, musíte ho simulovať vhodne definovanou činnosťou. (Činnosť definujeme názvom a zloženým príkazom, ktorý opisuje túto činnosť a odvoláva sa iba na skôr definované činnosti.)

Zadanie úloh

1. V premennej A je číslo X. Napíšte činnosť (jednu hlavnú a niekoľko pomocných), ktorá vypočíta celú časť z druhej odmocniny čísla X a uloží ju do A.
   (Pozn.: Hodnoty ostatných premenných nás nezaujímajú.)
2. V premennej A je číslo P. Napíšte činnosť, ktorá do premennej B uloží hodnotu 1, ak je P prvočíslo, inak do B uloží nulu.

Riešenie

1. Prístupov k riešeniu môže byť veľmi mnoho. Predvedieme riešenie, založené na jednoduchom tvrdení:

$$1 + 3 + 5 + 7 \ldots + (2N - 1) = N^2,$$

ktoré pre prirodzené N možno ľahko dokázať matematickou indukciou. V tomto vzťahu N je jednak počet

sčítancov a jednak »celá časť druhej odmocniny N^2«. To využijeme v nasledujúcom algoritme:

```
činnosť »do A daj celú časť odmocniny z A«
  N : = 0;                  ...počet sčítancov
  I : = 1;                  ...sčítanec
  pokiaľ A ≠ 0
    rob
        N : = N + 1;
        odčítaj I od A;
        I : = I + 2;
  koniec;
ak výsledok_odčítania_je_záporný tak N : = N − 1;
A : = N;
koniec;
```

Algoritmus nie je zapísaný v jazyku MMS. Na prepis do jazyka MMS potrebujeme dodefinovať činnosť »odčítaj A OD B«, ktorá okrem odčítania indikuje zápornosť výsledku.

To realizujeme tak, že pomocnú premennú CHYBA nastavíme na 1, ak výsledok odčítania je záporný (a teda výsledok, ktorý získame pri odčítaní, je chybný).

```
Činnosť »ODČÍTAJ A OD B«
rob NULUJ CHYBA;
    NULUJ POM1;
    pokiaľ A ≠ 0
          rob ak B = 0 tak rob CHYBA+;
                           NULUJ A koniec
                           inak rob A−; B−;
                           POM1− koniec
          koniec;
```

```
    PRESUŇ POM1 → A
koniec;
```

Teraz už ľahko prepíšeme algoritmus do jazyka MMS:

```
Činnosť »DO A DAJ CELÚ ČASŤ ODMOCNINY
Z A«
rob NULUJ N;                    ...N : = 0
    NULUJ I ; I+ ;              ...I : = 1
    pokiaľ A ≠ 0
                rob N+ ;        ...N : = N + 1
                    ODČÍTAJ I OD A;
                                ...A : = A − I
                                   (s »chybou«)
                    I+; I+      ...I : = I + 2
                koniec;
    ak CHYBA ≠ 0 tak N− ;
    PRESUŇ N → A
koniec
```

2. Pri riešení druhého problému postupujeme podobne. Prvočíselnosť zisťujeme hľadaním deliteľa len medzi číslami menšími, nanajvýš rovnými ako celá časť druhej odmocniny A.

```
Činnosť »DO B DAJ 1, AK A JE PRVOČÍSLO, INAK
DAJ 0«
rob X := A;
    DO Y DAJ CELÚ ČASŤ      ...to umožní
    ODMOCNINY Z X;             urýchliť výpočet
    NULUJ B ; B+ ;          ...B := 1
    pokiaľ X ≠ 0
          rob DO C DAJ A MODULO X;
              X− ;          ...zmenši X a presved-
```

```
                                    či sa, či si nedelil
                                    číslom 1
            ak X ≠ 0 tak
               ak C = 0 tak ...keď nie, vyšetri
                                    deliteľnosť
                  rob B− ; NULUJ X koniec
                                ...A nie je prvočíslo
         koniec;
koniec;
```

Dodefinujeme činnosť na výpočet zvyšku po celočíselnom delení.

```
Činnosť »DO C DAJ A MODULO B«
rob POM1:= A;
    pokiaľ POM2 ≠ 0
          rob C := POM2;      ...tu je výsledok
              ODČÍTAJ B           vtedy, keď zvyšok
              OD POM2;            je nenulový
              koniec
    ak CHYBA = 0              ...tu sa priradí
    tak NULUJ C                   výsledok pri
                                  nulovom zvyšku
koniec
```

a činnosť Y := X

```
Činnosť »Y := X«
rob NULUJ Y;
    NULUJ POM3;               ...pomocné premenné
    pokiaľ X ≠ 0
          rob X− ; Y+ ; POM3+ koniec
    PRESUŇ POM3 → X
koniec
```

P - II - 1

Najmenším nasledovníkom prirodzeného čísla A nazveme najmenšie také číslo B, ktoré spĺňa nasledujúce podmienky:
— číslo B sa skladá z tých istých cifier ako A (t. j. B vznikne nejakou permutáciou cifier čísla A);
— číslo B je väčšie ako A.
Sformulujte (slovami) a dokážte algoritmus, ktorý pre dané prirodzené číslo A nájde najmenšieho nasledovníka. Predpokladajte, že číslo A je zadané v poli C »rozbité po cifrách«, teda, že

$$A = C(1).10^{N-1} + C(2).10^{N-2} + \ldots + C(N-1).10 + C(N).$$

Výsledok — číslo B stačí vytvoriť v tom istom poli.

Príklad. Pre číslo 123 542 je jeho najmenší nasledovník číslo 124 235.

Riešenie. Použijeme toto označenie:

C, resp. C′ budú polia, v poli C′ bude nasledovník C;

$C(i)$ označuje i-tý prvok poľa;

$C(i:j)$ označuje postupnosť prvkov poľa $C(i)$, $C(i+1)$, …, $C(j)$;

VYMEŇ (C, i, j) bude pole, ktoré vznikne z poľa C výmenou prvkov $C(i)$ a $C(j)$;

$C(i:j) \leqq C(k)$ znamená, že všetky prvky z intervalu $C(i:j)$ sú menšie, nanajvýš rovné prvku $C(k)$;

OTOČ C(i : j) je operácia, ktorá zmení poradie prvkov v intervale C(i : j) na opačné, t. j. na C(j), C($j - 1$), ..., C(i).

Algoritmus pracuje takto:

Najprv nájdeme číslo i, pre ktoré platí

$$C(i) < C(i + 1)$$

tak, aby C($i + 1$: N) bola nerastúca postupnosť (t. j. i je najväčší index, pre ktorý platí vyššie uvedená nerovnosť). Keď takéto i neexistuje, potom postupnosť C(1 : N) je nerastúca a pre ňu najmenší nasledovník neexistuje. V tomto prípade algoritmus končí.

Rozoberme preto iba prípad, že také i existuje. Aby sme našli najmenšieho nasledovníka, musí platiť, že

$$C'(1 : i - 1) = C(1 : i - 1)$$

a prvok C'(i) musí byť najmenší z prvkov v postupnosti C($i + 1$: N), ktorý je väčší ako C(i). Taký prvok určite existuje. Keď ich je viac, zoberieme ten, ktorý má najväčší index — označme ho j.

Teraz vymeníme v poli C i-tý a j-tý prvok operáciou VYMEŇ(C, i, j), a takto ich vložme do poľa C'. Teda postupnosť C'(1 : i) má väčšiu hodnotu ako C(1 : i), ale ide o »najmenšie možné zväčšenie«, aké sme mohli dosiahnuť.

Potrebujeme už iba vytvoriť zvyšok postupnosti C'($i + 1$: N). Mohlo by sa zdať, že čísla v tejto postupnosti treba usporiadať podľa veľkosti (aj to by totiž viedlo k riešeniu). Stačí však omnoho menej. Pretože i je najvyšší index,

pre ktorý platí

$$C(i) < C(i + 1),$$

znamená to, že všetky ostatné čísla sú usporiadané klesajúco. Stačí teda ich poradie obrátiť operáciou OTOČ C(i + 1 : N).

Výsledný algoritmus má teda podobu:

1. Urči i a j.
2. VYMEŇ(C, i, j).
3. OTOČ C (i + 1 : N).

V tomto prípade dostanete nasledovníka v tom istom poli.

P - II - 2

Nasledujúci program v jazyku Pascal počíta pre zadané prirodzené čísla A a B ich súčin:

```
program NASOB (input, output);
var A, B, K, SÚČIN, POČET, PRÍRASTOK: integer;
begin readln(A, B);
      K : = 0; SÚČIN := 0: POČET := 0;
      PRÍRASTOK := 0;
      while K < B do
            begin
                  PRÍRASTOK := PRÍRASTOK + A;
                  POČET := POČET + 1;
                  K := K + POČET;
                  SÚČIN := SÚČIN + PRÍRASTOK
            end;
      while K ≠ B do
            begin SÚČIN := SÚČIN – A;
```

```
            K := K - 1
        end;
    writeln (SÚČIN)
end
```

Ten istý program napíšeme aj v jazyku Basic:

```
 10 REM PROGRAM NASOBENIA
 20 INPUT A, B
 30 K = 0 : SUCIN = 0 : POCET = 0 :
    PRIRASTOK = 0
 40 IF K ≥ B THEN GOTO 100
 50    PRIRASTOK = PRIRASTOK + A
 60    POCET = POCET + 1
 70    K = K + POCET
 80    SUCIN = SUCIN + PRIRASTOK
 90 GOTO 40
100 REM KONIEC PRVÉHO CYKLU
110 IF K = B THEN GOTO 150
120    SUCIN = SUCIN - A
130    K = K - 1
140 GOTO 110
150 PRINT SUCIN
```

Program pracuje tak, že v prvom cykle (v Basicu riadky 40–90) sa vypočíta súčin K.A, kde $K \geq B$, a v druhom cykle sa robí korekcia na hodnotu B.A niekoľkonásobným odčítaním A od premennej SÚČIN.

Prirodzené číslo B_0 nazveme najhorším prípadom, ak pre zadané ľubovoľné A a $B = B_0$ platí po skončení prvého cyklu pri realizácii programu na počítači

$$K = B + \text{POČET} - 1.$$

Inak povedané, pri poslednom pripočítavaní

sme hodnotu súčinu »prestrelili« najviac, ako sme mohli. Najhoršími prípadmi sú napríklad čísla 4 a 4 657.

a) Vyjadrite explicitne (tj. nejakým vzorcom) všetky najhoršie prípady.

b) Nech B je najhorší prípad. Nazvime S(B) počet aritmetických operácií (t.j. počet skutočne vykonaných sčítaní a odčítaní), ktoré sa vykonajú pri výpočte podľa programu pre ľubovoľné A a pevné B. (Všimnite si, že počet operácií nezávisí od A.) Vyjadrite S(B) ako funkciu B.

Riešenie. Uveďme najprv niekoľko prípravných úvah:

1. Pre priebeh výpočtu sú významné zmeny hodnôt premennej K. Hodnota K však vzniká ako súčet

$$K = 1 + 2 + 3 + \ldots + n \text{ pre } n = 1, 2, \ldots$$

2. Výpočet prvého cyklu skončí vtedy, keď $K \geqq B$.
3. Najväčší počet krokov späť v druhom cykle treba vykonať pre také B_0, pre ktoré z posledného prírastku K musíme odpočítať $(n - 1)$-krát.

a) Z (1) a (3) vyplýva, že najhorší prípad nastáva pri číslach tvaru:

$$B_0 = (1 + 2 + 3 + \ldots + n) - (n - 1) \qquad (*)$$

$$B_0 = (1 + 2 + 3 \ldots + (n - 1)) + 1$$

$$B_0 = \frac{n.(n - 1)}{2} + 1 = \frac{n^2 - n + 2}{2}$$

b) Keď sa teraz vrátime ku vzťahu (*) a rozoberieme význam jeho členov, vidíme, že prvých n členov súčtu vyjadruje prírastky K v jednotlivých krokoch prvého cyklu (t. j. prvý cyklus sa opakuje presne n-krát). Posledný člen je zhodný s počtom opakovaní druhého cyklu. Je teda treba určiť hodnotu n v závislosti od B.

Z tvrdení 1 a 2 vyplýva nerovnica

$$1 + 2 + 3 + \ldots + n \geqq \mathrm{B},$$

ktorej najmenšie kladné celočíselné riešenie (ak existuje) udáva tú hodnotu n, pri ktorej prvý cyklus končí. Hodnota o jednotku menšia je potom počet opakovaní druhého cyklu pre najhorší prípad. Po úpravách dostávame

$$\frac{n.(n+1)}{2} \geqq \mathrm{B}$$

$$n^2 + n - 2\mathrm{B} \geqq 0$$

$$n = \left[\frac{-1 + \sqrt{1 + 8\mathrm{B}}}{2}\right],$$

kde $[x]$ je horná celá časť čísla x.

Pretože prvý cyklus obsahuje štyry sčítania a druhý cyklus dve odčítania, tak hľadaná funkcia S(B) má tvar

$$\mathrm{S(B)} = 4.n + 2.(n-1) = 6 \cdot \left[\frac{-1 + \sqrt{1 + 8\mathrm{B}}}{2}\right] - 2.$$

Pre dostatočne veľké B stačí uvádzať približnú hodnotu

$$S(B) \doteq 3.\sqrt{1 + 8B}.$$

P - II - 3a

Napíšte program v ľubovoľnom vyššom programovacom jazyku, ktorý prečítá postupnosť zloženú z núl a jedničiek, ktorá je ukončená číslom 2. Program vypíše »ÁNO«, ak postupnosť obsahovala párny počet núl a zároveň nepárny počet jedničiek. V opačnom prípade program vypíše »NIE«.

Na program sa kladie jedno podstatné obmedzenie: Smie sa použiť len jediná premenná Z, v ktorej je vždy uložená hodnota jediného načítaného čísla (0, 1 alebo 2). Obsah premennej Z sa môže meniť len tým spôsobom, že sa do nej zo vstupu prečíta ďalšie číslo. Čísla sa môžu čítať len v tom poradí, v akom sú zapísané v postupnosti, t. j. nie je dovolené vracať sa k číslam už raz prečítaným, ani ich preskakovať.

Pre jednoduchosť predpokladajte, že čísla vstupujú zapísané vždy jedno v jednom riadku. Pritom vstupná postupnosť neobsahuje iné čísla než 0, 1 a 2. Program preto môže skončiť svoju činnosť, keď prečíta znak rôzny od nuly a jednotky.

Príklady. Pre postupnosť 0011011012 (ktorú sme pre úsporu miesta zapísali v jednom riadku) program vypíše »ÁNO«. Pre postupnosť 0011002 program vypíše »NIE«, pre postupnosť 1112 program vypíše »ÁNO«, lebo aj nula je párne číslo.

Riešenie. Algoritmus, ktorý spracováva vstupnú postupnosť, sa nachádza vždy v jednom zo štyroch možných stavov:

1. Doposiaľ načítal párny počet núl a párny počet jednotiek.
2. Doposiaľ načítal párny počet núl a nepárny počet jednotiek.
3. Doposiaľ načítal nepárny počet núl a nepárny počet jednotiek.
4. Doposiaľ načítal nepárny počet núl a párny počet jednotiek.

Vzájomné prechody medzi týmito štyrmi stavmi sa uskutočňujú v závislosti na tom, či sa načíta nula alebo jednotka. Ak však načítame číslo 2, tak »ÁNO« vytlačíme iba v prípade, keď sa práve nachádzame v stave č. 2. V ostatných prípadoch vypíšeme »NIE«. Z toho vyplýva i činnosť nasledujúcich programov.

Program v Basicu:

```
 10 INPUT Z : REM STAV 1
 11 ON Z + 1 GOTO 40, 20, 200
 20 INPUT Z : REM STAV 2
 21 ON Z + 1 GOTO 30, 10, 100
 30 INPUT Z : REM STAV 3
 31 ON Z + 1 GOTO 20, 40, 200
 40 INPUT Z : REM STAV 4
 41 ON Z + 1 GOTO 10, 30, 200
100 PRINT »ÁNO«
101 STOP
200 PRINT »NIE«
201 STOP
```

Program v Pascale:

```
program POSTUPNOST(input, output);
var Z: integer;
procedure S2; forward;
procedure S3; forward;
procedure S4; forward;
procedure S1;
```

```
begin readln(Z);
      case Z of 0: S4; 1: S2; 2: writeln(»NIE«) end
end;
procedure S2;
begin readln(Z);
      case Z of 0: S3; 1: S1; 2: writeln(»ÁNO«) end
end;
procedure S3;
begin readln(Z);
      case Z of 0: S2; 1: S4; 2: writeln(»NIE«) end
end;
procedure S4;
begin readln(Z);
      case Z of 0: S1; 1: S3; 2: writeln(»NIE«) end
end;
begin
      S1
end.
```

P - II - 3b

Naprogramujte pre Modifikovaný Minského stroj (viď P - I - 4) nasledujúcu úlohu:

V premenných A, B, C sú prirodzené čísla. Napíšte činnosť »triedenie«, ktorá do premenných A, B, C uloží takú permutáciu ich pôvodných hodnôt, že bude platiť

$$A \leqq B \leqq C,$$

teda, že na konci výpočtu budú ich hodnoty usporiadané vzostupne.

Riešenie. Popíšeme jedno z možných riešení. Najprv však definujeme pomocnú činnosť ZORAĎ na správne zoradenie dvoch premenných.
(Využijeme aj činnosti, definované pri riešení úlohy P - I - 4.)

```
Činnosť »ZORAĎ X Y«;
rob POMX := X;                    ...uložíme pôvodné
                                     hodnoty X a Y
    POMY := Y;
    ODČÍTAJ X OD Y;
    ak CHYBA = 0                  ...t. j. ak X ≦ Y
       tak rob  X := POMX;        ...ponechaj pôvodné
                Y := POMY            hodnoty
            koniec
       inak rob Y := POMX;        ...vymeň obsahy
                X := POMY            premenných
            koniec
koniec
```

Pomocou tejto činnosti ľahko definujeme činnosť UTRIEĎ, ktorá usporiada obsahy všetkých troch premenných:

```
Činnosť »UTRIEĎ A B C«
rob ZORAĎ A B;                    ...A ≦ B
    ZORAĎ A C;                    ...A ≦ C
    ZORAĎ B C;                    ...B ≦ C
koniec
```

Ľahko sa možno presvedčiť, že teraz platí

$$A \leqq B \leqq C.$$

P - III - 1

Dané sú dve prázdné nádoby. Prvá má objem M litrov, druhá N litrov, kde M a N sú prirodzené čísla. Máme k dispozícii neobmedzený zdroj vody.
Povolené sú nasledujúce operácie:

NAPLŇ I,

kde I = 1 alebo 2, znamená naplniť vodou zo zdroja I-tú nádobu až po okraj.

PRELEJ z I do J,

kde I $\neq$ J, pričom I, J = 1 alebo 2, znamená preliať obsah I-tej nádoby do J-tej tak, že buď J-tú nádobu naplníme až po okraj (ak je v I-tej dostatok vody), alebo I-tú nádobu úplne vyprázdnime (ak je v nej menej).

VYLEJ I,

kde I = 1 alebo 2, znamená vyliať obsah I-tej nádoby do kanála.

Postupom prelievania nazveme ľubovoľnú konečnú postupnosť zloženú z vyššie uvedených operácií. Určite:

a) Pre aké prirodzené čísla M, N a K existuje taký postup prelievania, že po jeho vykonaní zostane v niektorej z nádob presne K litrov vody?

b) Zostavte algoritmus, ktorý pre zadané prirodzené čísla M, N a K navrhne taký postup prelievania, že po jeho vykonaní ostane v niektorej nádobe práve K litrov vody. Ak taký postup existuje, tak ho algoritmus vypíše a na jeho konci uvedie

VÝSLEDOK JE V NÁDOBE I.

Ak taký postup neexistuje, algoritmus vydá správu
POSTUP PRELIEVANIA NEEXISTUJE!

Príklad. Pre M = 9, N = 5 a K = 2 je postup nasledujúci:

Príkaz	Obsah nádob	
	1.	2.
NAPLŇ 1	9	0
PRELEJ 1 2	4	5
VYLEJ 2	4	0
PRELEJ 1 2	0	4
NAPLŇ 1	9	4
PRELEJ 1 2	8	5
VYLEJ 2	8	0
PRELEJ 1 2	3	5
VYLEJ 2	3	0
PRELEJ 1 2	0	3
NAPLŇ 1	9	3
PRELEJ 1 2	7	5
VYLEJ 2	7	0
PRELEJ 1 2	2	5
VÝSLEDOK JE V NÁDOBE 1		

Riešenie. Najprv urobme niektoré jednoduché úvahy, ktoré sú na prvý pohľad zrejmé a umožňujú skrátiť dĺžku algoritmu:

1. Dve operácie — NAPLŇ a VYLEJ — zanechajú v nádobách maximálny, resp. nulový obsah. Z toho vyplýva, že keď chceme v niektorej nádobe dosiahnuť iný (netriviálny) počet litrov vody, nesmieme do tejto nádoby nalievať zo zdroja, resp. vylievať jej (netriviálny) obsah do kanála.
2. Zbytočné je prelievať vodu z jednej nádoby do druhej a zase naspäť.
3. Nemôže sa stať, že by obidve nádoby obsahovali netriviálny počet litrov vody. (Po VYLEJ a NAPLŇ je to zrejmé, operáciou PRELEJ buď jednu nádobu naplníme až po okraj, alebo druhú celkom vyprázdnime.)
4. Výsledné množstvo K litrov vody máme získať v jednej z nádob. Nutnou podmienkou existencie riešenia preto bude platnosť vzťahu »K je menšie, nanajvýš rovné väčšiemu z čísel M a N«.

Z toho vyplýva, že účelné využitie nádob je také, že jedna (a to napríklad väčšia) bude slúžiť iba na naberanie vody zo zdroja a na prelievanie do druhej a druhá iba na vylievanie do kanála a na prijímanie vody z prvej.

Bez újmy na všeobecnosti predpokladajme, že $M \geqq N$. Pochopiteľne, nádoba s obsahom M litrov bude tá, do ktorej budeme nalievať, nádoba s obsahom N litrov tá, z ktorej budeme vylievať vodu do kanála. Povedané inak, hľadáme čísla x a y také, že po x naliatiach vody do nádoby M a po y vyliatiach z nádoby N budeme mať presne K litrov vody. To vyjadruje napríklad vzťah

$$M.x - N.y = K$$

z riešenia Marcela Polakoviča, G A. Markuša, Bratislava.

Teda

$$M.x = N.y + K$$
$$M.x = K \pmod{N}$$
$$x = K.M^{-1} \pmod{N}$$

kde M^{-1} je inverzný prvok k M v grupe zvyškových tried modulo N. Taký prvok existuje pre všetky K iba vtedy, keď M a N sú nesúdeliteľné, inak existuje len pre tie K, ktoré sú deliteľné NSD(M, N). To je vlastne odpoveď na prvú otázku.

Naznačili sme už i optimálny algoritmus. Nalievať treba vždy do väčšej nádoby. Z nej potom prelievať vodu tak dlho do menšej, kým sa väčšia nevyprázdni (menšiu, keď sa zaplní, pochopiteľne treba vyliať), alebo kým vo väčšej nebude presne K litrov vody. Postup opakujeme a pritom sústavne sledujeme počet litrov vody vo väčšej nádobe — v nej bude totiž na konci výsledok.

Existuje ešte jeden možný, veľmi podobný postup, ktorý spočíva v prelievaní vody opačným smerom — z menšej nádoby do väčšej. Samozrejme, menšia nádoba sa pri úplnom vyprázdnení napĺňa zo zdroja, väčšiu po úplnom naplnení vylievame. Pre každú trojicu K, M, N, pre ktorú riešenie existuje, je možné použiť obidva postupy. Pre niektoré trojice vedie rýchlejšie k výsledku prvý postup, pre niektoré druhý.

Prakticky všetky správne riešenia odhalili súvislosť medzi postupom prelievania a deliteľnosťou a viac-menej sledovali niektorý z uvedených algoritmov.

Daný je algoritmus, ktorého vstupom sú prirodzené čísla A a B a výstupom ich súčin. Najprv ho uvedieme v Pascale:

```
program CELOŠTÁTNE KOLO (input, output);
var A, B, K, SÚČIN, POČET, PRÍRASTOK: integer;
begin readln(A, B);
      K := 0; SÚČIN := 0;
      POČET := 1; PRÍRASTOK := A;
      while K ≠ B do
            if K + POČET ≦ B
              then begin K := K + POČET;
                         POČET := POČET + POČET;
                         SÚČIN := SÚČIN +
                         PRÍRASTOK;
                         PRÍRASTOK :=
                         PRÍRASTOK + PRÍRASTOK
                   end
              else begin POČET := 1;
                         PRÍRASTOK := A
                   end;
      writeln(SÚČIN)
end
```

Verzia v Basicu:

```
 10 INPUT A, B
100 K = 0 : SÚČIN = 0 : POČET = 1 :
    PRÍRASTOK = A
110 IF K = B THEN GOTO 180
120    IF K + POČET > B THEN GOTO 160
```

```
130       K = K + POČET : POČET = POČET +
          POČET
140       SÚČIN = SÚČIN + PRIRASTOK :
          PRÍRASTOK = PRÍRASTOK + PRÍRASTOK
150       GOTO 170
160       POČET = 1 : PRÍRASTOK = A
170 GOTO 110 : REM NÁVRAT NA ZAČIATOK
    CYKLU
180 PRINT SÚČIN
```

Najhorším prípadom nazveme výpočet pre takú hodnotu B_0, pri ktorej sa vetva **else** podmieneného príkazu v cykle (v Basicu riadok 160) vykoná viackrát než pre ktorékoľvek $B < B_0$.

a) Vyjadrite explicitne (t. j. nejakým vzorcom) všetky tie hodnoty B_0, pre ktoré bude výpočet algoritmu najhorším prípadom.

b) Vyjadrite explicitne (t. j. ako funkciu B) skutočný počet prechodov
— vetvou **then** (riadky 130 a 140)
— vetvou **else** (riadok 160)
podmieneného príkazu v najhoršom prípade.

Poznamenávame, že počet prechodov cyklom nezávisí od hodnoty A.

Riešenie. Najprv ukážeme riešenie časti a). Z neho dobre vidieť aj postup, ktorý umožní vyriešiť časť b).

a) Pre priebeh výpočtu sú charakteristické zmeny premennej POČET. Obsahom sú to vždy mocniny dvojky, a to v takomto poradí:

$$1, 2, 4, 8, \ldots, 2^i, 1, 2, 4, 8, \ldots, 2^j, 1, 2, \ldots$$

Pritom musí vždy platiť $i \geqq j$. Pritom všade, okrem posledných dvoch členov, platí ostrá nerovnosť $i > j$. Keby totiž $j = i + 1$, tak by člen 2^{i+1} musel nasledovať hneď po prvom člene 2^i. Z toho súčasne vyplýva, že súčet zvyšných členov postupnosti po 2^i musí byť menší ako 2^{i+1} (pretože inak by sme opäť mohli 2^{i+1} pripočítať po 2^i). A pretože

$$1 + 2 + 4 + 8 + \ldots + 2^i = 2^{i+1} - 1,$$

rovnosť

$$i = j$$

môže nastať len pri posledných dvoch i a j, lebo vtedy (a len vtedy) nenasleduje po 2^j ďalšia jednotka.

V dôsledku toho budú najhoršími prípadmi tie hodnoty B_0, pri ktorých K vzniká ako súčet

$$\begin{aligned} B_0 = {} & (1 + 2 + 4 + \ldots + 2^i) + \\ & + (1 + 2 + 4 + \ldots + 2^{i-1}) + \\ & + (1 + 2 + 4 + \ldots + 2^{i-2}) + \\ & + \ldots + (1 + 2) + 1 + 1. \end{aligned} \qquad (*)$$

Teda

$$\begin{aligned} B_0 = {} & (2^{i+1} - 1) + (2^i - 1) + (2^{i-1} - 1) + \ldots + \\ & + (2^2 - 1) + (2^1 - 1) + 2^0. \end{aligned}$$

Tento výraz obsahuje $i + 1$ zátvoriek. Sčítajme zvlášť ich prvé členy (a pridajme k nim 2^0) a zvlášť druhé členy:

$$B_0 = (2^{i+1} + 2^i + 2^{i-1} + \ldots + 2^2 + 2^1 + 2^0) - (i + 1) =$$

$$= (2^{i+2} - 1) - (i + 1) = 2^{i+2} - (i + 2),$$

kde $i = 0$. Po vhodnej substitúcii môžme vzťah pre všetky najhoršie prípady prepísať do tvaru

$$B_0 = 2^n - n, \text{ kde } n \geqq 2.$$

Zo vzorca (*) vidieť veľmi pekne i priebeh výpočtu, takže sa oň budeme opierať v riešení časti b) tejto úlohy.

b) Krátka analýza programu ukáže, že
— počet prechodov vetvou **else** je rovný počtu jednotiek vo vzorci (t. j. $i + 2$),
— počet prechodov vetvou **then** je rovný počtu sčítancov vo vzorci, t. j.

$$\left(\sum_{i=1}^{i+1} k\right) + 1 = \frac{(i + 1)(i + 2)}{2} + 1 = \frac{i^2 + 3i + 4}{2}.$$

Veľa účastníkov zbytočne stratilo body na tom, že oba vzorce ponechali v takomto tvare. Text úlohy bol totiž formulovaný tak, že výsledok mal vyjadrovať počet prechodov týmito vetvami v závislosti od (teda ako funkciu) premennej B. A pretože sme vyššie ukázali, že

$$B = 2^{i+2} - (i + 2)$$

pre najhoršie prípady, tak môžeme (približne) tvrdiť, že pre dostatočne veľké i platí

$$B \doteq 2^{i+2}.$$

Takže počet vykonaní vetvy **else** je zhruba

$$i \doteq \log_2 B$$

a počet vykonaní vetvy **then**

$$\frac{1}{2} \log_2^2 B.$$

(V druhom prípade sme zanedbali lineárny a absolútny člen.) Oba vzorce je možné určiť úplne presne. Pri analýze algoritmu sa však obyčajne uspokojujeme s dostatočne presným priblížením.

P - III - 3

a) Napíšte program v ľubovoľnom vyššom programovacom jazyku, ktorý prečíta ľubovoľnú postupnosť pozostávajúcu z čísel 1, 2 a 3, ukončenú číslom 0. Program vypíše »ÁNO«, keď postupnosť obsahovala rovnaký počet čísel 1, 2 aj 3. V opačnom prípade program vypíše »NIE«.

Na program sa kladie jedno podstatné obmedzenie. Smiete

použiť len dve premenné Z a P. V premennej Z je vždy uložená hodnota jediného načítaného čísla (0, 1, 2 alebo 3). Obsah Z sa môže meniť len tým spôsobom, že sa do nej zo vstupu prečíta ďalšie číslo. Čísla sa môžu čítať len v tom poradí, v akom sú zapísané v postupnosti. Teda nie je dovolené čísla preskakovať ani vracať sa k prv prečítaným. Druhá celočíselná premenná P obsahuje ľubovoľne veľké celé číslo. Povolené sú všetky aritmetické operácie s premennými a konštantami.

Predpokladajte, že čísla vstupujú napísané vždy jedno v riadku. Pritom vstupná postupnosť neobsahuje iné čísla než 0, 1, 2 a 3, teda program musí ukončiť činnosť, keď prečíta číslo 0. Ďalej predpokladajte, že na vstupe môže byť vopred neohraničený počet čísel.

Príklady. Pre postupnosť 1232231310 (pre úsporu miesta sme čísla zapísali do riadku) program vypíše »ÁNO«. Pre postupnosť 23320 program vypíše »NIE«. Pre prázdnu postupnosť 0 program vypíše »ÁNO«.

b) Zovšeobecnite riešenie predchádzajúcej úlohy takto:

Program prečíta ľubovoľnú postupnosť zloženú z čísel 1, 2, 3, ..., N, ktorá je ukončená číslom 0. Počet vstupujúcich čísel N je vopred známa konštanta (známa aj programátorovi). Program zistí, či postupnosť obsahuje rovnaký počet každého z čísel 1 až N. Obmedzenie na 2 premenné Z a P zostáva presne to isté.

Príklady. Pre postupnosť čísel 1 až 5: 44351215320 program, zostavený pre N = 5, vypíše »ÁNO«, program, zostavený pre N = 6, vypíše »NIE« (podobne programy pre N = 7, 8, ...).

Riešenie. Najčastejším riešením tejto úlohy bolo riešenie založené na myšlienke prvočíselného rozkladu prirodzených čísel.

Ako je známe, každé prirodzené číslo sa dá jediným spôsobom rozložiť na súčin prvočísel. Pretože máme iba jednu celočíselnú premennú, bude v P uložené číslo, ktoré vytvoríme z uvažovanej postupnosti súčinom prvočísel. V časti a) stačia prvé tri, v časti b) potrebujeme prvých N:

$$q_1 = 2,\ q_2 = 3,\ q_3 = 5,\ \ldots,\ q_N = \text{N-té prvočíslo}$$

Hodnotou premennej P bude číslo, ktoré vznikne tak, že po načítaní čísla i do premennej Z vynásobíme obsah P i-tým prvočíslom. Teda prvá časť programu bude mať v Pascale tento tvar:

```
begin readln(Z); P := 1;
      while Z ≠ 0 do
            begin if Z = 1 then P := 2*P;
                  if Z = 2 then P := 3*P;
                  if Z = 3 then P := 5*P;
                  ... ... ...
                  if Z = N then P := qN*P;
                  readln(Z)
            end
```

V pôvodnej postupnosti sa nachádza rovnaký počet čísel 1, 2, 3, ..., N iba v tom prípade, keď výsledné číslo P je deliteľné číslom

$$2 * 3 * 5 * \ldots * q_N.$$

Teda na konci programu (po výstupe z cyklu) stačí skontrolovať deliteľnosť v príkaze

if P MOD (2 * 3 * 5 * ... * q_N) = 0
  then writeln ('Postupnosť mala požadovaný tvar.')
  else writeln ('Postupnosť nemala požadovaný tvar.')
end

V skutočnosti však riešenie nevyžaduje, aby čísla v ňom vystupujúce boli prvé prvočísla, dokonca ani to, aby to boli prvočísla. Stačí, aby boli relatívnymi prvočíslami (teda, aby ich najväčší spoločný deliteľ bol pre všetky dvojice rovný 1). Toto využil napr. Petr Veselý z gymnázia v Jihlave. Vo svojom programe použil prvočísla 3, 5 a 7 pri riešení časti a). Vďaka tomu namiesto zložitého rozhodovania mohol použiť jediný priraďovací príkaz

$$P := P * (2 * Z + 1).$$

Zaujímavú (i keď nekorektnú) obmenu vyššie uvedeného riešenia použil Jan Hřebíček z gymnázia vo Valašskom Meziříčí. Ignoroval požiadavku celočíselnosti P a pripustil sčítanie nekonečne dlhých reálnych čísel (s absolútnou presnosťou). Takto mohol vytvárať hodnotu P sčítaním (!) vhodných iracionálnych čísel. V prvej časti úlohy použil napr. 1, π, e. V tomto prípade je v postupnosti rovnaký počet čísel vtedy, keď vytvorené číslo P je celočíselným násobkom súčtu iracionálnych čísel (t. j. $1 + \pi + e$ v jednoduchšom prípade).

Ďalšie správne riešenie využíva hĺbku rekurzie ako pomocnú premennú popri premennej P. Je však vhodné iba pre postupnosti z troch čísel. Takto riešili úlohu dvaja súťažiaci. Jedným z nich bol Róbert Germič z gymnázia

v Žiline. Uvádzame ukážku jeho riešenia. Riešenie využíva fakt, že hoci v Basicu rekurzia nie je oficiálne povolená, existencia systémového zásobníka ju umožňuje. Tu je program:

```
 99 LET P = 0
100 INPUT Z
110 IF Z = 1 THEN GOSUB 200 : GOTO 100
120 IF Z = 2 THEN GOSUB 300 : GOTO 100
130 IF Z = 3 THEN P = P + 1 : GOTO 100
140 IF P = 0 THEN PRINT »ANO« : STOP
150 PRINT »NIE« : STOP
200 INPUT Z
210 IF Z = 1 THEN GOSUB 200 : GOTO 200
220 IF Z = 2 THEN P = P — 1 : RETURN
230 IF Z = 3 THEN P = P +1 : GOTO 200
240 PRINT »NIE« : STOP
300 INPUT Z
310 IF Z = 1 THEN P = P — 1 : RETURN
320 IF Z = 2 THEN GOSUB 300 : GOTO 300
330 IF Z = 3 THEN P = P + 1 : GOTO 300
340 PRINT »NIE«
```

Ak by takýto program nebol písaný s komentárom, bolo by asi vylúčené zistiť, čo vlastne autor zamýšľal. Zvlášť záhadné by boli dvojice príkazov

GOSUB i : GOTO i

v riadkoch 210 a 320. Autor však podal jednoduché vysvetlenie:

V premennej P si budeme pamätať rozdiel medzi počtom trojok a dvojíc (1, 2). Ak je v niektorom okamihu

počet dvojek väčší ako počet jednotiek, program pokračuje od príkazu 300 (bude hľadať k prevyšujúcej jednotke dvojku). V prípade prevyšujúcej dvojky bude k nej hľadať jednotku (teda bude v časti za príkazom 200). Z týchto častí sa vráti iba vtedy, keď sa počty jednotiek a dvojok vyrovnajú do časti za príkazom 100. Ak program skončí v časti 100 a P = 0, tak je počet dvojíc (1, 2) rovnaký ako počet trojok. V každom inom prípade sa po prečítaní nuly vytlačí »NIE«.

P - III - 4

Naprogramujte pre Modifikovaný Minského stroj (definícia viď príklad P - I - 4) nasledujúcu úlohu:

V premenných A a K sú prirodzené čísla. Napíšte činnosť »CIFRA K A«, ktorá do premennej D uloží K-tú cifru čísla A. Počet K počítame od začiatku čísla A.

Príklad. Činnosť

»CIFRA 2 5728«

vloží do premennej D hodnotu 7.

Riešenie. Táto úloha dala opravovateľom celoštátneho kola najviac práce. Máloktorý účastník si dal totiž dosť námahy s komentármi k programom, väčšina z nich sa sústredila iba na príkazovú časť. Ako sme už ukázali pri riešení minulej úlohy, takto vytvorené texty možu byť priam »nedešifrovateľné« bez podrobnejšieho popisu. Pochopiteľne, nezrozumiteľnosť textu sa prejaví stratou bodov pri hodnotení riešenia.

Výber Modifikovaného Minského stroja ako úlohy, ktorá prechádzala systematicky celým ročníkom, však nebol ná-

hodný. Primitívne prostriedky na zmenu hodnôt premenných, jediná možnosť testovať ich (na nulu) totiž sledovali dôležitý cieľ – analyzovať schopnosť súťažiacich vybudovať si systém podprogramov, zaručujúcich úplné vyriešenie úlohy. Pritom iba málo súťažiacich si uvedomilo, že systém podprogramov sa nedá účelne budovať od najjednoduchších (na MMS realizovateľných) »programíkov«, že totiž cieľavedomý postup je obyčajne presne opačný. Uvedomili si to však tí najlepší. Napríklad Alexander Szabari z gymnázia v Košiciach začal svoje riešenie slovami:

»Keď máme vziať *K*-tú cifru od konca, vtedy je program veľmi jednoduchý. Preto náš program bude pozostávať z dvoch väčších častí:

1. výmena cifier čísla *K* (napr. z čísla 5138 urobíme 8315),
2. zistenie *K*-tej cifry od konca.

Obe časti spolu umožňujú nájsť *K*-tú cifru od začiatku.«

Je teda jasné, že hľadaný program môže mať tvar

```
ČINNOSŤ »CIFRA K A«
rob PREVRÁŤ A;
    URČI K A
koniec
```

Skutočnosť, že ani jedna z vnútorných činností nie je zatiaľ definovaná, síce môže vadiť MMS pri výpočte, nie však človeku pri rozmýšľaní!

V takto navrhnutom programe je viac-menej jedno, ktorú jeho časť budeme upresňovať skôr – sú totiž nezávislé. Začneme hociktorou a ďalej ju rozkladáme, až dovtedy, kým všetky činnosti vyjadríme príkazmi MMS. Pritom sa môže stať, že podprogramy vytvorené v jednej časti sa dajú priamo alebo po miernej úprave použiť i v inej časti. Aby sme ukázali, že

na poradí rozvíjania činností naozaj nezáleží, začneme podprogramom pre URČI K A, uvedeným ako druhý.

Poznamenávame, že kým spôsob zápisu programu bol presne definovaný, spôsob jeho komentovania bol ponechaný na ľubovôli autora. V zásade možno použiť tieto spôsoby:

a) najprv uviesť program a vzápätí komentár,
b) najprv si (slovne) určiť cieľ a potom ho realizovať programom,
c) paralelne sprevádzať program komentármi,
d) napísať program v známejšom jazyku a potom ho »preložiť«.

Postupne uvedieme príklady na všetky spôsoby.

Program URČI najprv zapíšeme v tvare

```
ČINNOSŤ »URČI K A«
rob K−;
    pokiaľ K ≠ 0
            rob DEĽ A 10;
                K−
            koniec;
    MOD A 10;
    PRESUŇ A D
koniec
```

a vzápätí ho popíšeme komentárom:

»Každé číslo má aspoň jednu cifru. Preto sme najprv zmenšili K o 1, aby výsledok testu vychádzal správne. Potom (pokiaľ hľadaná cifra nie je poslednou v čísle A) delíme A celočíselne desiatimi, čím vždy odtrhneme momentálne poslednú cifru. Nakoniec vypočítame túto poslednú cifru ako A(mod 10) a výsledok uložíme do premennej D.«

Našou ďalšou úlohou bude teda vytvoriť činnosti DEĽ a MOD. Začneme podprogramom pre DEĽ, ktorý uvedieme s komentárom na začiatku, vytyčujúcim cieľ programu:

Deliť na MMS môžeme iba pomocou odčítania. Teda delenie nahradíme úlohou »zistiť, koľkokrát možno 10 odčítať od čísla A«. Podiel budeme zhromažďovať v premennej Q (jej hodnotu zväčšíme o 1 po zmenšení A o 10). Pri hraničných hodnotách by však mohlo dôjsť k chybe (treba, aby vyšlo 10/10 = 1), takže na začiatku zväčšíme A o 1 a na konci Q o 1 zmenšíme. Teda píšeme

```
ČINNOSŤ »DEĽ A 10«
rob NULUJ Q;
    A +;
    opakuj A−; A−; A−; A−; A−; A−; A−; A−;
                                        A−; A−;
           Q+;
    kým A = 0;
    Q−;
    PRESUŇ Q → A
koniec
```

Z tohoto programu veľmi jednoduchou modifikáciou získame program na výpočet zvyšku po delení. Tento program uvedieme s komentárom, paralelným s programom.

```
ČINNOSŤ »MOD A 10«
rob A+;      ... Jedno pričítanie kvôli hraničnému prípadu
    opakuj PRESUŇ A → C;          priradenie C := A
           A−; A−; A−; A−; A−; A−; A−; A−;
                                        A−; A−;
```

... Od A odčítame 10, v C je teraz A + 10. Posledná cifra A sa odčítaním desiatich nezmení.

kým A = 0;

C−; ... V A je nula, v C bolo číslo od 1 do 10. Teraz sme vyrovnali počiatočné zväčšenie A o 1.

PRESUŇ C → A ... V A bude výsledok.

koniec

Tým sme dokončili rozklad činnosti URČI K A. Podobne budeme postupovať i pri definícii činnosti PREVRÁŤ A. V tomto prípade úlohu vyriešime najprv v známom programovacom jazyku (v Pascale) a potom ho preložíme do jazyka MMS a zohľadníme pritom jeho špecifiká.

Zmeniť poradie cifier v čísle na opačné môže tento pascalovský program:

```
begin CHVOST := A MOD 10;      ... posledná cifra
                                   čísla A bude prvou
                                   cifrou OPAK-u
      OPAK := CHVOST;
      A := A DIV 10;
                                   ... v A sú všetky cifry
                                   okrem poslednej
      while A > 0 do
            begin CHVOST := A MOD 10;
                  ... posledná cifra A
                  OPAK := (OPAK * 10) + CHVOST;
                  ... prilep ju na koniec čísla OPAK
                  A := A DIV 10
```

```
                    ... znovu odsekni poslednú cifru
            end
end
```

Pri preklade nesmieme zabudnúť najmä na to, že MMS »ničí« obsahy premenných. Teda operácie MOD a DEĽ (ako sme ich definovali vyššie) nezachovajú obsah premennej A. Preto vytvoríme program KOPÍRUJ A B, ktorý skopíruje A do B a pritom obsah A zachová (použitím tretej premennej). Takže tú istú myšlienku vyjadruje podprogram v jazyku MMS:

```
ČINNOSŤ »OBRÁŤ A«
rob  KOPÍRUJ A B;          ... skopíruje obsah A do B
                               a zachová hodnotu A
     MOD B 10;             ... v B bude posledná cifra A
     PRESUŇ B OPAK;        ... realizuje
                               OPAK := A MOD 10
     DEĽ A 10;             ... známy podprogram
     pokiaľ A ≠ 0
          rob  KOPÍRUJ A B;
               MOD B 10;
               PRILEP B OPAK;
                         ... realizuje OPAK :=
                             := (OPAK * 10) + B
               DEĽ A 10
          koniec;

PRESUŇ OPAK A              ... dá obrátenú hodnotu do A
koniec
```

Ostali už len dve nedefinované činnosti: KOPÍRUJ a PRILEP. Na základe toho, čo sme povedali už skôr, však nemôže byť problémom ich doplniť. (Naviac KOPÍRUJ A B je zhodné s príkazom B := A v úlohe P-I-4.)