[dokumenty-10] 40 let matematické olympiády (v Československu)

Ivan Libicher; Pavel Töpfer
Pražský korespondenční seminář z programování

In: Karel Horák (editor): [dokumenty-10] 40 let matematické olympiády (v Československu). (Czech). Praha: Jednota českých matematiků a fyziků, 1993. pp. 50–54.

Persistent URL: http://dml.cz/dmlcz/405382

Pražský korespondenční seminář z programování

Ivan Libicher, Pavel Töpfer

Mezi studenty středních škol získaly v poslední době velkou oblibu různé korespondenční semináře. Po řadě matematických seminářů se objevily semináře fyzikální a v posledních letech také programátorské. Z nich pravděpodobně nejmladší, ale rozsahem své působnosti a počtem účastníků největší je pražský korespondenční seminář z programování. Seminář vznikl na začátku školního roku 1987/88 z iniciativy studentů matematicko-fyzikální fakulty Univerzity Karlovy. Pokusný „nultý" ročník semináře proběhl nad očekávání úspěšně, a proto se organizátoři rozhodli pokračovat v pořádání semináře i v dalších letech. Dnes má pražský korespondenční seminář z programování každoročně více než 200 účastníků.

Podobně, jako je tomu u jiných korespondenčních seminářů, je i tento organizován v několika kolech. Každé kolo je tvořeno čtyřmi soutěžními úlohami různé obtížnosti, z nichž si studenti mohou vybrat k řešení podle vlastního uvážení jen ty, které zvládnou. Všechny úlohy jsou zaměřené na návrh algoritmů a tvorbu programů. Samotný zápis programu by ale jako řešení úlohy samozřejmě nestačil. Je nutné připojit také slovní popis řešení, vysvětlení způsobu práce algoritmu a zdůvodnění jeho správnosti. Hodnotí se i kvalita navrženého algoritmu, zejména jeho efektivita. Zaslaná řešení úloh opravují organizátoři semináře — studenti informatiky z různých ročníků MFF UK v Praze. Opravené úlohy zasílají zpět soutěžícím spolu s komentáři a se vzorovými řešeními úloh a s výsledkovými listinami. Na závěr každého ročníku semináře připravují pro nejlepší účastníky týdenní soustředění s bohatým programem odborným i oddechovým.

Soutěžní úlohy mají podobný charakter jako úlohy kategorie P matematické olympiády. Z minulých ročníků semináře jsme pro vás vybrali na ukázku tři úlohy. Pro nedostatek místa uvádíme jejich vzorová řešení bez výsledných programů.

Úlohy

1. Největší díra (KSP 1–4–2)

Napište program, který k zadanému celému číslu n, $n \geq 2$, a zadané posloupnosti reálných čísel délky n (která se vejde do paměti!) vytiskne dvě čísla z posloupnosti taková, že žádné číslo z posloupnosti neleží „mezi nimi" (tj. není menší než jedno a větší než druhé z nich) a že absolutní hodnota jejich rozdílu je maximální. Např. pro $n = 4$ a posloupnost 2, 5.3, 2.7, −20.1 program vytiskne dvojici čísel 2, −20.1.

2. Zplodiny (KSP 2–3–3)

Nechť $p_1, \ldots, p_n$ ($n \geqq 0$) je posloupnost celých čísel. *Zplodinovými operacemi* nazveme následující změny posloupnosti: a) vypuštění prvního členu (neprázdné posloupnosti): $p_1, \ldots, p_n \mapsto p_2, \ldots, p_n$, b) přidání libovolného čísla r na konec posloupnosti: $p_1, \ldots, p_n \mapsto p_1, \ldots, p_n, r$, c) změnu některého členu p_i, $1 \leqq i \leqq n$, na libovolné celé číslo r:

$$p_1, \ldots, p_n \mapsto p_1, \ldots, p_{i-1}, r, p_{i+1}, \ldots, p_n.$$

Navrhněte algoritmus, který pro dané posloupnosti celých čísel, $p_1, \ldots, p_n$ ($n \geqq 0$) a $q_1, \ldots, q_m$ ($m \geqq 0$) určí minimální počet zplodinových operací potřebný ke změně $p_1, \ldots, p_n$ na $q_1, \ldots, q_m$. Např. ke změně posloupnosti 5, 2, 4, 1, 12 na 4, 3, 12 jsou třeba nejméně tři zplodinové operace:

$$5, 2, 4, 1, 12 \mapsto 2, 4, 1, 12 \mapsto 4, 1, 12 \mapsto 4, 3, 12.$$

3. Ďábelské mocniny (KSP 2–2–1)

D-číslem nazvěme každé kladné celé číslo takové, že se dá vyjádřit ve tvaru $3^i 4^j 5^k$, kde $i, j, k \geqq 0$ jsou celá čísla. Navrhněte algoritmus, který pro dané kladné celé číslo n vytiskne prvních n D-čísel.

Např. pro $n = 10$ algoritmus vytiskne čísla 1, 3, 4, 5, 9, 12, 15, 16, 20, 25.

Řešení

1. Postup řešení se bude skládat z následujících pěti kroků:

1. Nalezneme minimální a maximální prvek ze zadaných n čísel a označíme je *Min* a *Max*. Platí tedy $Min \leqq a_i \leqq Max$ pro všechna i od 1 do n. To je možné provést velmi snadno jedním sekvenčním průchodem danými n čísly, tedy s lineární časovou složitostí.

2. Spočítáme hodnotu $D = (Max - Min)/(n-1)$. Rozdíl $Max - Min$ udává velikost celkového intervalu na číselné ose, který sledujeme. Údaj $n-1$ určuje, na kolik nejvýše úseků je tento interval rozdělen zadanými n čísly, tzn. kolik existuje „děr" mezi čísly. Hodnota D má proto význam dolního odhadu velikosti maximální díry. Kdyby byla všechna ostatní čísla rozložena mezi *Min* a *Max* zcela rovnoměrně, měly by všechny díry mezi nimi velikost přesně D. Při jakémkoliv jiném rozložení čísel mezi *Min* a *Max* bude některá díra menší, a proto musí být jiná větší než D.

Poznámka. Zvláštním případem je situace, kdy se všech n daných čísel sobě rovná. Potom $Max = Min$ a vyjde nám tedy $D = 0$. Největší díra mezi čísly má nulovou velikost a najdeme ji mezi libovolnou dvojicí zadaných čísel.

3. Rozdělíme nyní celý interval $\langle Min, Max \rangle$ na $n-1$ úseků velikosti D a tyto úseky očíslujeme ve vzestupném pořadí od 1 do $n-1$. Všimněte si, že pro libovolné číslo X z intervalu $\langle Min, Max \rangle$ dokážeme snadno (v konstantním čase) určit, do kterého úseku patří. Pořadové číslo příslušného úseku je dáno výrazem $\lfloor (X - Min)/D \rfloor + 1$, kde závorky $\lfloor$, $\rfloor$ označují dolní celou část z hodnoty výrazu v nich uzavřeného, tzn.

$\lfloor V \rfloor$ je rovno největšímu celému číslu, které je menší nebo rovno hodnotě výrazu V. Poznamenejme ještě, že úseky chápeme jako intervaly zdola uzavřené a shora otevřené. Číslo X ležící přesně na hranici dvou úseků bude tedy zařazeno do vyššího z nich. Pro $X = Max$ dává náš výraz pořadové číslo úseku n. Číslo $X = Max$ bude tvořit samo další, v pořadí n-tý úsek, který je degenerován do jediného bodu.

4. Vytvoříme si dvě pomocná pole velikosti n označená R a S. Tato pole zaplníme tak, aby pro každé i od 1 do n mělo R_i hodnotu minimálního a S_i hodnotu maximálního ze zadaných čísel, které náleží do i-tého úseku (viz bod 3). Pokud do některého z úseků nepadne ani jedno ze zadaných čísel, dosadíme do R_i a S_i zvláštní předem zvolenou hodnotu, která není z intervalu $\langle Min, Max \rangle$, např. $Max + 1$, $Min - 1$. Správné hodnoty polí R a S získáme snadno s lineární časovou složitostí. Pro každé z n zadaných čísel stačí určit, do kterého úseku patří (podle vzorce z bodu 3), a poté toto číslo porovnat s do té doby platnými hodnotami polí R a S odpovídajícími tomuto úseku:

pro každé $i = 1, \ldots, n$ **proveď**
    $R_i := Max + 1$;
    $S_i := Min - 1$;
pro každé $i = 1, \ldots, n$ **proveď**
    $U := \lfloor (a_i - Min)/D \rfloor + 1$; (* číslo úseku *)
    jestliže $a_i < R_U$ **potom** $R_U := a_i$;
    jestliže $a_i > S_U$ **potom** $S_U := a_i$

5. Vzhledem k tomu, že podle bodu 3 nemá žádný z úseků délku větší než D a přitom podle bodu 2 je hodnota D dolním odhadem velikosti maximální díry mezi zadanými čísly, má buď maximální díra velikost právě D, nebo je větší, ale pak nemůže ležet celá uvnitř jediného úseku, musí do ní padnout předěl mezi dvěma sousedními úseky. Výslednou velikost maximální díry proto stačí hledat vždy mezi minimální a maximální hodnotou čísel náležejících do sousedních úseků. Přitom je ještě třeba dávat pozor na ty úseky, které neobsahují žádné ze zadaných n čísel. Předělů mezi úseky je přibližně n, takže i tato část výpočtu má lineární časovou složitost:

$MaxDira := D - 1$; (* dolní odhad velikosti max. díry *)
$i := 1$;
dokud $i < n - 1$ **prováděj**
    (* S_i je začátek právě zkoumané díry *)
    $j := i + 1$;
    dokud $R_j = Max + 1$ **prováděj** $j := j + 1$;
        (* hledáme další neprázdný úsek *)
        (* nehrozí přetečení, neboť $R_n = S_n = Max$ *)
    (* R_j je konec právě zkoumané díry *)
    $Dira := R_j - S_i$;
    jestliže $Dira > MaxDira$ **potom**
        $MaxDira := Dira$;
        $C1 := S_i$; $C2 := R_j$;
    $i := j$
(* $MaxDira$ je velikost maximální díry, $C1$, $C2$ jsou hledaná čísla *)

Správnost algoritmu byla zdůvodněna ve výše uvedeném rozboru. Konečnost výpočtu vyplývá ze skutečnosti, že počet průchodů všemi cykly je omezen počtem zpracovávaných čísel n. Algoritmus má lineární časovou složitost, neboť každý z jeho kroků má buď lineární nebo konstantní časovou složitost a jednotlivé kroky výpočtu následují jeden po druhém.

2. Posloupnosti zplodinových operací převádějící $p_1, \ldots, p_n$ na $q_1, \ldots, q_m$ budeme stručně nazývat p-q-posloupnostmi. Pro libovolné posloupnosti $p_1, \ldots, p_n$ a $q_1, \ldots, q_m$ existuje p-q-posloupnost délky $\max(m, n)$:

1) pro $n < m$ tvořená n operacemi c) a $m - n$ operacemi b),
2) pro $n \geqq m$ tvořená $n - m$ operacemi a) a m operacemi c).

Dále je jasné, že každá p-q-posloupnost musí obsahovat nejméně $|n-m|$ operací a) a b), aby se délka změnila z n na m. Z toho plyne, že má-li být nějaká p-q-posloupnost kratší než $\max(m, n)$, musí v ní být méně než $\min(m, n)$ operací c), tj. některé prvky posloupnosti p musí přejít do q beze změny.

Ukážeme, že zplodinové operace v nejkratší p-q-posloupnosti lze přeuspořádat tak, že nejprve se provedou všechny operace a), pak c) a nakonec b): Žádná operace c) jistě nemění člen, který by pozdější operace a) vypustila (pak by totiž p-q-posloupnost nebyla nejkratší); můžeme tedy všechny operace a) přesunout před operace c). Podobně žádná operace c) jistě nemění člen dříve přidaný nějakou operací b) (tyto dvě operace by totiž bylo možné nahradit jedinou operací b)); můžeme tedy operace b) přesunout za operace c). Konečně žádná operace a) jistě nevypouští prvek přidaný dříve operací b) (obě operace by bylo možné vypustit). Hledejme tedy nejkratší p-q-posloupnost ve tvaru

$$\begin{array}{lll} p_1, \ldots, p_n & \mapsto \ldots k \text{ operací a)} \ldots & \mapsto p_{k+1}, \ldots, p_n \\ p_{k+1}, \ldots, p_n & \mapsto \ldots c(k) \text{ operací c)} \ldots & \mapsto q_1, \ldots, q_{n-k} \\ q_1, \ldots, q_{n-k} & \mapsto \ldots m - n + k \text{ operací b)} \ldots & \mapsto q_1, \ldots, q_m, \end{array}$$

kde $\max(n - m, 0) \leqq k \leqq n$ a $c(k)$ je minimální počet operací c) potřebný ke změně $p_{k+1}, \ldots, p_n$ na $q_1, \ldots, q_{n-k}$:

$$c(k) = |\{i;\ 1 \leqq i \leqq n - k, p_{k+i} \neq q_i\}|.$$

Označíme-li d délku nejkratší p-q-posloupnosti, platí

$$d = \min(k + c(k) + (m - n + k)),$$

kde k prochází hodnoty od $\max(n - m, 0)$ do n. Tento vzoreček spolu s definicí $c(k)$ dává kompletní algoritmus pro výpočet d. Jeho správnost vyplývá z odvození.

Paměťová složitost (určená délkou posloupností p a q) je lineární, časová složitost je $O(n^2)$, neboť výpočet $c(k)$ (jehož složitost je $O(n - k)$) se provede nejvýše $(n + 1)$-krát.

3. Všechna D-čísla vzestupně očíslujeme takto: $1 = d_1$, $3 = d_2$, $4 = d_3$, $5 = d_4$, $9 = d_5$, $12 = d_6$, $15 = d_7$, atd.

TVRZENÍ. Nechť $k > 1$, $1 \leqq p, q, r < k$ jsou celá čísla s těmito vlastnostmi:

$$d_p = \min\{d_i;\ 1 \leqq i < k \text{ a } 3d_i > d_{k-1}\},$$
$$d_q = \min\{d_i;\ 1 \leqq i < k \text{ a } 4d_i > d_{k-1}\},$$
$$d_r = \min\{d_i;\ 1 \leqq i < k \text{ a } 5d_i > d_{k-1}\}.$$

Potom platí $d_k = \min\{3d_p, 4d_q, 5d_r\}$.

DŮKAZ. Využijeme toho, že D-čísla větší než jedna jsou právě čísla tvaru $3D$, $4D$ nebo $5D$, kde D je nějaké menší D-číslo (plyne to přímo z definice D-čísla). Číslo $d = \min\{3d_p, 4d_q, 5d_r\}$ je tedy D-číslo. Navíc d je nejmenší z D-čísel větších než d_{k-1}. Kdyby totiž mezi d_{k-1} a d ležela nějaká D-čísla, pak nejmenší z nich by bylo tvaru $3d_i$, $4d_i$ nebo $5d_i$, $1 \leqq i < k$, takže d by nebylo minimální. Tím je důkaz tvrzení proveden.

Známe-li tedy D-čísla $d_1, \ldots, d_{k-1}$ $(k > 1)$ a čísla p, q, r z uvedeného tvrzení, potom $d_k = \min\{3d_p, 4d_q, 5d_r\}$. „Nové" hodnoty čísel p, q, r splňující

$$d_p = \min\{d_i;\ 1 \leqq i \leqq k \text{ a } 3d_i > d_k\},$$
$$d_q = \min\{d_i;\ 1 \leqq i \leqq k \text{ a } 4d_i > d_k\},$$
$$d_r = \min\{d_i;\ 1 \leqq i \leqq k \text{ a } 5d_i > d_k\}$$

získáme úpravou „starých" hodnot takto: pokud $3d_p = d_k$, pak číslo p zvětšíme o jednu, dále pokud $4d_q = d_k$, pak číslo q zvětšíme o jednu, dále pokud $5d_r = d_k$, pak číslo r zvětšíme o jednu.

Tím máme jednak indukční krok pro generování dalšího D-čísla, jednak invariant, který nám zaručí správnost následujícího algoritmu:

```
(* n je počet hledaných D-čísel *)
(* (D_1, ..., D_n) je pole celých čísel *)
D_1 := 1; p := 1; q := 1; r := 1;
pro k = 2, ..., n proveď
    (* D_1, ..., D_{k-1} je prvních k - 1 D-čísel, *)
    (* p, q, r mají vlastnost z výše uvedené úvahy *)
    d_k := min{3d_p, 4d_q, 5d_r};
    jestliže 3d_p = d_k potom p := p + 1;
    jestliže 4d_q = d_k potom q := q + 1;
    jestliže 5d_r = d_k potom r := r + 1;
pro k = 1, ..., n proveď tiskni(d_k)
```

Konečnost algoritmu je zřejmá. Invariant uvedený v komentáři v těle prvního cyklu zaručuje správnost algoritmu.

Časová i paměťová složitost tohoto algoritmu je $O(n)$.