53. ročník Matematické olympiády – 2003/2004

Úlohy krajského kola kategorie P

Krajské kolo 53. ročníku MO kategorie P se koná v úterý 6. 1. 2004 v dopoledních hodinách. Na řešení úloh máte 4 hodiny čistého času. V krajském kole MO-P se neřeší žádná praktická úloha, pro zajištění rovných podmínek řešitelů ve všech krajích je použití počítačů při soutěži zakázáno.

Řešení každého příkladu musí obsahovat:

- **Popis řešení,** to znamená slovní popis použitého algoritmu, argumenty zdůvodňující jeho správnost (případně důkaz správnosti algoritmu), diskusi o efektivitě vašeho řešení (časová a paměťová složitost). Slovní popis řešení musí být jasný a srozumitelný i bez nahlédnutí do samotného zápisu algoritmu (do programu).
- **Program.** V úlohách **P-II-1**, **P-II-2** a **P-II-3** je třeba uvést dostatečně podrobný zápis algoritmu, nejlépe ve tvaru zdrojového textu nejdůležitějších částí programu v jazyce Pascal nebo C. Nemusíte podrobně popisovat jednoduché operace jako vstupy, výstupy, implementaci jednoduchých matematických vztahů, vyhledávání v poli, třídění apod. V řešení úlohy **P-II-4** zapište úplný program pro dvousměrný registrový počítač.

Hodnotí se nejen správnost programu, ale také kvalita popisu řešení a efektivita zvoleného algoritmu.

Vzorová řešení úloh naleznete krátce po soutěži na Internetu na adrese *http://mo.mff.cuni.cz/*. Na stejném místě bude na konci února zveřejněn i seznam postupujících do celostátního kola. Naleznete zde také popis prostředí, v němž budete na celostátním kole řešit praktické úlohy.

P-II-1 Síť

Firma Truhlík a syn má ve městě N budov a chce všechny svoje budovy propojit počítačovou sítí. Vedení firmy rozhodlo, že pro K ($1 \leq K \leq N$) budov zakoupí vysokorychlostní připojení na Internet. Kromě toho mezi některými dvojicemi budov vybudují propojení optickým kabelem.

Dvě budovy se nacházejí v téže komponentě sítě, pokud lze mezi nimi komunikovat pomocí optických kabelů (buď mají přímé spojení, nebo jsou spojeny nepřímo přes několik jiných budov). Aby bylo možné komunikovat mezi dvěma budovami ležícími v různých komponentách sítě, musí každá z těchto komponent obsahovat aspoň jeden počítač připojený na Internet.

Soutěžní úloha: Na vstupu jsou dána čísla N a K a pro každou dvojici budov jedno kladné celé číslo – cena za vybudování optického kabelu, který by propojil tuto dvojici budov. Navrhněte efektivní algoritmus, jenž určí, kterých K budov se má připojit na Internet a které dvojice budov se mají propojit optickým kabelem tak, aby mezi každými dvěma budovami bylo možné komunikovat a přitom aby celková cena vybudovaných optických kabelů byla co nejmenší.

Příklad:

Vstup:
$N = 4, K = 2$
Ceny spojení:
(1,2): 100
(1,3): 10
(1,4): 100
(1,5): 300
(2,3): 100
(2,4): 10
(2,5): 300
(3,4): 47
(3,5): 27
(4,5): 74

Výstup:
Na Internet připojíme budovy 1 a 2,
kabelem spojíme dvojice budov $(1, 3)$, $(2, 4)$ a $(3, 5)$.
Cena kabelů bude 47.

P-II-2 AttoSoft

Již z domácího kola znáte Vaškovu programátorskou firmu AttoSoft. Vaškovi se nyní podařilo získat druhého klienta. Ten mu dal opět za úkol naprogramovat N jednoduchých programů.

Vašek chce tentokrát ušetřit ještě více, a proto místo programátorů zaměstnal N studentů, na každý program jednoho studenta. Firma AttoSoft vlastní stále jen jeden počítač a na něm může v každém okamžiku pracovat jen jeden student. Hlavní problém ale spočívá v tom, že studenti mohou pracovat jen ve volných chvílích mezi přednáškami.

Student i potřebuje p_i hodin času na napsání přiděleného programu, přijde do firmy v čase s_i a musí odejít nejpozději v čase t_i. Svůj program nemusí psát najednou, může občas práci přerušit a počítač uvolnit jiným studentům.

Soutěžní úloha: Napište program, jenž určí, kdy má počítač používat který student, aby všichni stihli napsat své programy za jeden den, nebo zjistí, že to není možné.

Příklad 1:

Vstup:
$N = 3$
$p_1 = 1, s_1 = 3, t_1 = 4$
$p_2 = 2, s_2 = 2, t_2 = 5$
$p_3 = 5, s_3 = 1, t_3 = 10$

Výstup:
Student 1 pracuje od 3 do 4.
Student 2 pracuje od 2 do 3 a od 4 do 5.
Student 3 pracuje od 5 do 10.

Příklad 2:

Vstup:
$N = 2$
$p_1 = 200, s_1 = 300, t_1 = 500$
$p_2 = 200, s_2 = 400, t_2 = 600$

Výstup:
Nelze.

P-II-3 Bageta

Kleofáš dostal dnes ráno hlad a rozhodl se připravit si obloženou bagetu se sýrem. Bagetu si můžeme představit jako úsečku dlouhou N cm, nebo přesněji jako uzavřený interval $\langle 0, N \rangle$. Každý kousek sýra tvoří rovněž uzavřený interval celočíselné délky. Jelikož Kleofáš je pedant, pokládá na bagetu kousky sýra tak, aby souřadnice jejich začátků i konců byla celá čísla. Kleofáš by chtěl, aby bageta byla pokryta sýrem přesně podle jeho představ. Na to ale potřebuje jednoduchý počítačový program, který by mu pomohl.

Soutěžní úloha: Na vstupu je dána velikost bagety N (N je celé číslo, $1 \leq N \leq 10^6$). Následuje P příkazů ($1 \leq P \leq 10^9$), přičemž každý z nich má jeden z následujících možných tvarů:

`PRIDEJ` a b	Kleofáš přidal kousek sýra sahající od a do b.
`KOLIK` c	Program vypíše zprávu, kolik kusů sýra leží na pozici c.

Váš program musí zpracovávat příkazy v pořadí, v jakém jsou uvedeny na vstupu. Pro každý příkaz `KOLIK` vypište jedno číslo – počet dosud položených kusů sýra, které leží nad souřadnicí c. (Jelikož kousky sýra jsou uzavřené intervaly, počítají se i ty kousky, pro něž je c souřadnice jejich začátku nebo konce.)

Příklad:

Vstup:
$N = 20$
`PRIDEJ 1 10`
`PRIDEJ 6 12`
`KOLIK 5`
`KOLIK 6`
`PRIDEJ 4 14`
`KOLIK 5`
`KOLIK 16`

Výstup:
`1`
`2`
`2`
`0`

P-II-4 Registrový počítač

V tomto kole se budeme zabývat tzv. *dvousměrnými* registrovými počítači. Od registrových počítačů z domácího kola se liší tím, že se při čtení vstupního slova dovedou vracet zpět. Jejich formální definici najdete ve studijním textu, který následuje za zadáním soutěžní úlohy. Změna oproti původní definici registrových počítačů z domácího kola je zvýrazněna.

Soutěžní úloha: Napište program pro dvousměrný registrový počítač, který bude řešit následující úlohu: Vstupem programu bude řetězec písmen `a`, `b`, `c`, `d`. Počítač ho označí jako správný právě tehdy, jestliže je to palindrom, tzn. je stejný při čtení zepředu i zezadu. Formálně řečeno: slovo $a_1 a_2 a_3 \ldots a_{n-1} a_n$ je palindrom, jestliže $a_1 = a_n$, $a_2 = a_{n-1}$, $\ldots$, $a_{\lfloor n/2 \rfloor} = a_{\lceil n/2 \rceil}$. Tedy například vstupy `bacab` a `dd` jsou palindromy, zatímco vstupy `baacdbb` a `bacabdccc` nejsou.

Základním kritériem hodnocení navrženého programu bude počet registrů, které váš program používá. Pokuste se napsat program, kterému jich stačí co nejméně. Druhým kritériem pak bude doba výpočtu programu.

Studijní text:

Registr je něco podobného jako proměnná. V registru může být uloženo *libovolně velké* nezáporné celé číslo. Na rozdíl od proměnných, které mezi sebou můžeme sčítat, odčítat a násobit, s registrem lze provádět jen tři jednoduché operace: zvětšit jeho obsah o 1, zmenšit jeho obsah o 1 (pokud se pokusíme zmenšit obsah registru obsahujícího hodnotu 0, zůstane v něm 0) a otestovat registr, zda je v něm 0. Na začátku výpočtu jsou ve všech registrech nuly.

Registrový počítač může používat neomezený počet registrů označených R_0, R_1, R_2, atd. Vedle registrů má k dispozici ještě *konečně velkou* pomocnou paměť.

Program pro registrový počítač budeme zapisovat v jazyce velmi podobném programovacímu jazyku Pascal. Programovací jazyk registrového počítače bude oproti Pascalu rozšířen například o příkazy pro práci s registry, naopak některé příkazy z Pascalu v něm budou zakázány.

Registrový počítač bude řešit úlohy následujícího typu: počítač dostane na vstupu zadáno slovo (řetězec písmen) a po nějakém čase odpoví, zda je toto slovo *správné* nebo *špatné*. Aby nám mohl odpovědět, zavedeme do programovacího jazyka speciální příkazy *Accept* a *Reject*. Jakmile se během výpočtu vykoná příkaz *Accept*, vstupní slovo je správné a výpočet končí.

Jestliže se provede příkaz *Reject*, slovo je špatné a výpočet končí. Pokud se výpočet zacyklí nebo pokud skončí, aniž by se provedl příkaz *Accept* nebo *Reject*, zadané vstupní slovo je rovněž špatné.

Příkaz „přičti 1 k obsahu registru R“ budeme značit *Inc(R)*, „odečti 1 od obsahu registru R“ budeme značit *Dec(R)*. Výraz *Zero(R)* je pravdivý, jestliže je v registru R nula, v opačném případě je nepravdivý.

V každém programu můžeme použít jen konečně mnoho registrů. Kromě nich můžeme použít už jen konstantní počet pomocných proměnných typu *byte** (nemůžeme tedy používat pole!) a dále můžeme zvláštním způsobem využívat jednu speciální proměnnou *vstup* typu *char*.

Proměnná *vstup* vždy obsahuje hodnotu aktuálního písmena ze vstupu. Na začátku výpočtu je aktuálním písmenem první písmeno zadané na vstupu. Polohu aktuálního písmena můžeme v programu změnit provedením příkazů *Left* (aktuálním se stane předcházející písmeno) a *Right* (aktuálním se stane následující písmeno). Pokud by se po provedení jednoho z těchto příkazů mělo aktuální písmeno nacházet mimo zadané vstupní slovo, proměnná *vstup* bude obsahovat speciální znak `$`. (Můžeme si představovat, že před i za vstupním slovem je zapsáno dostatečně mnoho znaků `$`.)

Jelikož registrový počítač má kromě registrů jen konečně mnoho paměti, nemůže si dovolit používat rekurzi (neměl by si kde pamatovat návratové adresy). My pro jistotu úplně zakážeme definovat a používat v programu procedury a funkce. Zakázáno je i volání všech standardních procedur a funkcí jazyka Pascal. V aritmetických výrazech lze používat pouze proměnné (tedy ne registry!), celočíselné konstanty, celočíselné operátory `+`, `-`, `*`, `div`, `mod` a závorky. V podmínkách se mohou používat výrazy *Zero*(R_i), běžné relační operátory (`<`, `<=`, ...), logické spojky a závorky.

Z klíčových slov jazyka Pascal jsou tedy v programovacím jazyku registrového počítače povolena pouze následující: `var`, `begin`, `end`, `if`, `then`, `else`, `case`, `of`, `while`, `do`, `repeat`, `until`, `for`, `to`, `downto`, `div`, `mod`, `and`, `or`, `not`, `xor`.

Příklad 1:

Napište program pro dvousměrný registrový počítač, který bude řešit následující úlohu: Na vstupu bude zadán řetězec písmen `a`, `b`, `c`. Nechť α označuje počet písmen `a` ve vstupním řetězci, β nechť je počet `b` a γ počet `c`. Počítač má vstupní řetězec označit za správný právě tehdy, když $\alpha = \beta = \gamma$.

Řešení: Projdeme vstupní slovo zleva doprava a v registrech R_1 a R_2 si přitom spočítáme hodnoty α a β. Když vstupní slovo dočteme, porovnáme obsahy obou registrů. Vrátíme se na začátek, při druhém průchodu vstupním slovem spočítáme v R_1 a R_2 hodnoty β a γ a opět je porovnáme. Rozmyslete si, že by stačilo použít jediný registr, v němž bychom měli hodnotu $|\alpha - \beta|$, resp. $|\beta - \gamma|$.

```
var vstup: char;
begin
   while (vstup<>'$') do begin
      if (vstup='a') then Inc(R1);
      if (vstup='b') then Inc(R2);
      Right;
   end;
   while not Zero(R1) do begin
      Dec(R1); if Zero(R2) then Reject else Dec(R2);
   end;
   if not Zero(R2) then Reject;
   { bylo stejně 'a' a 'b', vrátíme se na začátek }
   Left; while (vstup<>'$') do Left;
   Right;

   while (vstup<>'$') do begin
      if (vstup='b') then Inc(R1);
      if (vstup='c') then Inc(R2);
      Right;
   end;
   while not Zero(R1) do begin
      Dec(R1); if Zero(R2) then Reject else Dec(R2);
   end;
   if Zero(R2) then Accept;
end.
```

Příklad 2:

Napište program pro dvousměrný registrový počítač, který bude řešit následující úlohu: Na vstupu bude zadán řetězec písmen a. Počítač ho označí za správný právě tehdy, když je jeho délka mocninou tří.

Řešení: Řešení bude vypadat stejně jako na jednosměrném registrovém počítači. Projdeme vstupním slovem zleva doprava, přičemž si do registru R_1 uložíme délku tohoto slova. Potřebujeme zjistit, zda je to mocnina tří. Uloženou hodnotu proto budeme dělit třemi, dokud to půjde. Jestliže nakonec dostaneme podíl 0 a zbytek 1, původní číslo bylo mocninou tří, jinak nebylo.

* taková proměnná obsahuje jedno celé číslo z rozmezí od 0 do 255

```
var vstup: char;
    zbytek: byte;
begin
   while (vstup<>'$') do begin Inc(R1); Right; end;
   if Zero(R1) then Reject;
   while true do begin
      zbytek:=0;
      while not Zero(R1) do begin
         Dec(R1);
         zbytek:=(zbytek+1) mod 3;
         if (zbytek=0) then Inc(R2);
      end;
      if (Zero(R2) and (zbytek=1)) then Accept;
      if (zbytek<>0) then Reject;
      while not Zero(R2) do begin Dec(R2); Inc(R1); end;
   end;
end.
```