KSI 2012
Úloha 2-5: Abstraktní kód

Jan Horáček
Gymnázium, Brno, Vídeňská 47; jan.horacek@seznam.cz

18. listopadu 2012

1 Úvod

Tak přesně takovéto úlohy podkopávají snahu učitelů naučit své studenty psát pěkný a přehledný zdrojový kód. Doufám, že takovouto šílenost zažívám poprvé a naposled.

2 Popis funkce funkce ProvedNeco

Po poněkud zdlouhavém dešifrování jsem se dostal asi k následujícímu:

Funkce *ProvedNeco* provede sjednocení polí X a Y do pole Z, které je následně seřazeno vzestupně za použití bubble sortu, přičemž N je délka pole X a M je délka pole Y

Někdo by mohl namítnout, že můj popis není správný. A měl by pravdu. Jedinou chybou v tomto vysvětlení je fráze "...následně seřazeno..."- tedy předpoklad, že jednotlivé fáze probíhají sekvenčně, tedy nejdřív sjednocení a pak seřazení. Naneštěstí se orgové rozhodli, že tyto 2 kroky neuvěřitelně obskurním způsobem vecpou do sebe. Vše ve jménu zvýšení časové složitosti...

3 Proč je daný způsob řešení neefektivní?

Bubble sort, to je jediné slovo, které vystihuje neefektivnost řešení. Možná ještě $O(n^2)$.

Bubble sort je pěkný algoritmus na vysvětlování proncipů třídění na středních školách, ale do praxe (a hlavně na velké množiny) není moc použitelný. A to zejména díky jeho časové náročnosti $O(n^2)$.

Když pominu bubble sort, tak se v programu nachází nesmyslné části kódu, které jen zvyšují čas, po který program běží. Tím je například otáčení celé vstupní množiny hned na začátku. Anebo repeat-until na začátku funkce *ProvedNeco*, který stejně vždy proběhne jen 1x. Zbytečné instrukce navíc zvyšující nepřehlednost, časovou a prostorovou náročnost.

Na konec pak zmiňme bubble sort, který proběhne částečně ve funkci *Oblacno*, která mimochodem vpodstatě jen přidává prvek do množiny, a pak podruhé ve funkci *ProvedNeco*.

Tolik asi k neefektivnnosti.

4 Jak by to šlo udělat lépe?

4.1 Smazat funkci *Zatazeno*

Nevidím důvod, proč by se měla vstupní množina otáčet. Tedy, kromě zvýšení nepřehlednosti...

4.2 Modifikovat funkci *Oblacno*

A to takovým způsobem, že tato funkce provede sjednocení daných vstupních polí do prvního pole *pole*1.

A pak přejmenování této funkce na nějaký název vyjadřující podstatu funkce.

Tato funkce by tak mohla vypadat třeba takto:

```
procedure Sjednot(var pole1:TIntArray;pole2:TIntArray;
      pole1_c:Integer;pole2_c:Integer);
var i:Integer;
begin
 for i := 1 to pole2_c do
   pole1[pole1_c+i] := pole2[i];
end;//procedure
```

Listing 1: sjednocení polí

4.3 Napsat efektivně řazení

A to nejlépe quicksortem, který je detailně popsaný na wikipedii.

Funkce pro quicksort by pak mohla vypadat třeba takto:

```
procedure QuickSort(var ar:TIntArray;l, r: integer);
var
  i, j, pivot, pom: integer;
begin
  i := l;
  j := r;
  pivot := ar[(l + r) div 2];
  repeat
    while ((i < r) and (ar[i] < pivot)) do i := i + 1;
    while ((j > l) and (pivot < ar[j])) do j := j - 1;
    if (i <= j) then
     begin
      if (i < j) then
       begin
        pom := ar[i];
        ar[i] := ar[j];
        ar[j] := pom;
       end;
      if (i < r) then i := i + 1;
      if (j > l) then j := j - 1
     end;
  until (i > j);
  if (j > l) then quicksort(ar, l, j);
  if (i < r) then quicksort(ar, i, r)
end;
```

Listing 2: řazení

Zde je volen pivot velmi jednoduchou metodou prostředního čísla. Pokud bychom stáli o ještě lepší průměrnou časovou náročnost, doporučoval bych volit pivota jako medián 3 hodnot, například:

```
//vraci primo pivot
function FindPivot(ar:TIntArray;l,r:Integer):Integer;
var pivots:T0IntArray;
    i:Integer;
begin
 for i := 0 to 2 do pivots[i] := ar[Random(r)+l];
 BubbleSort(pivots,3);
 Result := pivots[1];
end;//function
```

Listing 3: volba pivotu

5 Implemetace vlastního řešení

K tomuto popisu je přiloženo finální řešení v souboru *AbstrKod.dpr*. Tento soubor je validním zdrojovým kódem pro jazyk Object Pascal v prostředí Delphi 2009.

Tento zdrojový kód obsahuje hledání pivotu přes mediána tří.

6 Závěr

Při průměrných vstupních hodnotách lze říci, že časová náročnost quicksortu je $O(NlogN)$, ba i lepší, protože jsme pivota zvolili relativně sofistikovaně a přesně. Časová náročnost slučování polí je $O(n_{pole2})$.

Na závěr bych chtěl poznamenat, že číslování od jedničky je strašná věc!

Reference

[1] Wikipedia *Quick sort*
http://cs.wikipedia.org/wiki/Quicksort