Fakultet tehničkih nauka, Novi Sad
Animacija u inženjerstvu

20. VI 2013. godine

Diskretne i kombinatorne metode za računarsku grafiku

1. Napisati pseudo kod za funkciju koja za ulazni niz veličine n ($n \geq 2$) vraća redni broj drugog po veličini elementa.

```
1: function DRUGI PO VELICINI(A)
2:     if A[2] > A[1] then
3:         rbr1 ← 2;
4:         rbr2 ← 1;
5:     else
6:         rbr1 ← 1;
7:         rbr2 ← 2;
8:     end if
9:     for i ← 3 to length(A) do
10:        if A[i] > A[rbr1] then
11:            rbr2 ← rbr1;
12:            rbr1 ← i;
13:        else if A[i] > A[rbr2] then
14:            rbr2 ← i;
15:        end if
16:    end for
17:    return rbr2
18: end function
```

2. Za pseudo kod iz zadatka 1 izračunati broj poređenja ako je ulazni niz dužine n sortiran rastuće i ako je sortiran opadajuće.

Za **rastuće** sortiran niz broj poređenja je $1:$ u liniji broj 2 plus $n-2:$ unutar for-petlje u liniji broj 10, ukupno $1+n-2=n-1$.

Za **opadajuće** sortiran niz broj poređenja je $1:$ u liniji broj 2 plus $n-2:$ unutar for-petlje u liniji broj 10 plus $n-2:$ unutar for-petlje u liniji broj 13, ukupno $1+2(n-2)=2n-3$.

3. Šta pseudo kod iz zadatka 1 vraća kao rezultat za ulazni niz
   (a) $A=[2,2,1]$? (b) $A=[2,1,1]$?

(a) Vraća 2, redni broj druge dvojke (dvojke su i prvi i drugi po veličini).
(b) Vraća 2, redni broj prve jedinice (jedinice su i drugi i treći po veličini).

4. Odrediti red veličine broja poređenja za niz dužine n koji predstavlja worst case za pseudo kod iz zadatka 1?

Worst case broj poređenja je slučaj kada se pored poređenja u liniji 2 vrše sva poređenja u for-petlji, linije 10 i 13.

Ovaj slučaj smo imali u zadatku 2, za opadajući niz, ukupno $2n-3$.

Red veličine je $2n-3=\Theta(n)$.

5. Dati definiciju "velikog O" ponašanja.

Za niz $g(n)$

$$O(g) = \{f | (\exists c_1 > 0)(\exists n_0 \in \mathbb{N})(\forall n)(n \geq n_0) \Rightarrow (0 \leq f(n) \leq c_1 g(n))\}$$

Umesto da pišemo $f \in O(g)$, pišemo $f = O(g)$ i čitamo: funkcija f se ponaša kao $O(g)$ (kao veliko O od g).

6. Pokazati da je $2n^2 \ \ln n + n = O(n^3)$.
   Da li je $n^2 \ \ln n + n = \Theta(n^3)$?
   Da li je $n^2 \ \sqrt{n} = O(n^2)$?

Niz $\dfrac{2n^2 \ln n + n}{n^3}$ je konvergentan (konvergira ka nuli), zato je ograničen, to jest postoji c_1 takvo da počev od nekog n_0 važi $\dfrac{2n^2 \ln n + n}{n^3} \leq c_1$, što je ekvivalentno sa $2n^2 \ln n + n \leq c_1 n^3$.

$n^2 \ \ln n + n$ nije $\Theta(n^3)$.

$n^2 \ \sqrt{n}$ nije $O(n^2)$.

7. Apstraktni tip Stek je realizovan nizom:

stack.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "stack.h"
#define MAXS 1000

struct stack{
     stackdata last[MAXS];
     int length;};

void makenull(node **S){
     (*S)=(node *) malloc(sizeof(node));
     (*S)->length = -1;}

int isempty(node *S){
     return (int) (S->length==-1);}

void push(node **S, stackdata d){
     (*S)->last[(*S)->length+1] = d;
     (*S)->length = (*S)->length + 1;}

void clear(node **S){
     free(*S);}
```

Pritom je sadržaj heder fajla:

stack.h

```
typedef char stackdata;
typedef struct stack node;

void makenull(node **);
int isempty(node *);
void push(node **, stackdata);
stackdata pop(node **);
stackdata top(node *);
void clear(node **);
```

Napisati u programskom jeziku C procedure "pop" i "top".

Rešenje

```
stackdata pop(node **S){
     stackdata d = (*S)->last[(*S)->length];
     (*S)->length = (*S)->length - 1;
     return d;}

stackdata top(node *S){
     stackdata d = S->last[S->length];
     return d;}
```

8. Nacrtati potpuni bipartitni graf $K_{2,2}$ i dati njegovu reprezentaciju matricom susedstva.

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

9. U tabeli su date udaljenosti između 5 gradova.

	1	2	3	4	5
1	-	28	115	45	110
2	28	-	87	30	93
3	100	87	-	75	115
4	45	30	75	-	135
5	120	93	110	135	-

(a) Definisati problem trgovačkog putnika.

(b) Polazeći od čvora 1, metodom najbržeg suseda naći približno rešenje problema trgovačkog putnika.

(c) Za isti problem naći Mađarskom metodom angažovanje koje je rešenje relaksiranog problema trgovačkog putnika.

(d) Znajući rešenja (b) i (c), u kojim granicama se nalazi optimalno rešenje problema trgovačkog putnika?

(a) Polazeći od jednog grada obići sve ostale gradove tačno jednom, vratiti se u početni grad tako da ukupan pređeni put bude minimalan.

(b) $1 \to 2 \to 4 \to 3 \to 5 \to 1$

Dužina puta $28 + 30 + 75 + 115 + 120 = 368$.

(c) Posle redukcija matrice, pivotizacije i određivanja angažovanja, (preskačemo račun) dobijamo jedno rešenje $\begin{array}{c} 1 \to 2 \\ 2 \to 4 \\ 4 \to 1 \\ 3 \to 5 \\ 5 \to 3 \end{array}$, čija dužina puta je $28 + 30 + 115 + 45 + 110 = 328$.

(d) Pošto u relaksiranom rešenju imamo podciklove $1 \to 2 \to 4 \to 1$ i $3 \to 5 \to 3$, zaključujemo da je optimalno rešenje između rešenja pod (b) i (c): $328 \leq \zeta^* \leq 368$.

Uzgred, optimalno rešenje je $\begin{array}{c} 1 \to 2 \\ 2 \to 5 \\ 5 \to 3 \\ 3 \to 4 \\ 4 \to 1 \end{array}$, čija dužina puta je $28 + 93 + 110 + 75 + 45 = 351$.