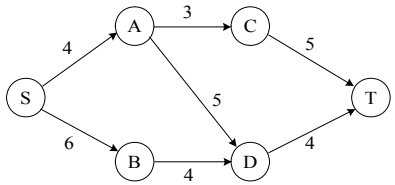
ISPIT IZ ALGORITAMA I STRUKTURA PODATAKA *6. februar 2008.*

Oni koji hoće da im se računa rezultat sa kolokvijuma rade samo ***zadatke 4, 5 i 6*** *u trajanju od* ***2 sata****. Ostali rade* ***sve zadatke*** *u trajanju od* ***4 sata.***

1. [16] Objasniti realizaciju FIFO reda za čekanje u vidu kružnog bafera predstavljenog nizom $Q[0:n-1]$. Skicirati i objasniti operacije za umetanje u red i brisanje iz njega. Koje su prednosti i mane u odnosu na realizaciju u redu $Q[1:n]$.

2. [16] Kodirati poruku FANFANLALA primenom statičkog i dinamičkog Huffman-ovog algoritma i LZW algoritma. Porediti dužine kodiranih poruka. Kod LZW algoritma pretpostaviti da je dužina kodiranog simbola saglasna veličini tabele simbola u trenutku kodiranja.

3. [18] Skicirati i objasniti *Ford-Fulkerson*-ov algoritam. Ilustrovati ga preko stanja protočnog i rezidualnog grafa na primeru grafa zadatog slikom. Šta ukoliko ima više izvora i odredišta?



4. [20] Usvojiti efikasnu implementaciju skoro kompletnog binarnog stabla, a zatim napisati na jeziku C ili C++ kompletan program za uređivanje niza celih brojeva u neopadajućem poretku metodom *heapsort*. Obavezno komentarisati program i obrazložiti izbor implementacije binarnog stabla.

5. [16] Pitanja:
   a) Kako se može optimizovati sekvencijalno pretraživanje niza ili liste, ako su poznate verovatnoće uspešnog pretraživanja p_i. Napisati izraz za prosečan broj poređenja pri uspešnom pretraživanju.
   b) Skicirati i objasniti operaciju leve rotacije u AVL stablu, pokazati korektnost i ilustrovati opštom slikom.
   d) Uporediti operacije umetanja u B i B*-stablo.

6. [14] Posmatra se prazna heš tabela sa 11 ulaza. Heširanje se obavlja primenom dvostrukog heširanja gde je sekundarna heš funkcija g(K)=2+Kmod3. U tabelu se redom umeću ključevi 13, 23, 18, 34, 35, 29, 28, 59. Odrediti prosečan broj pristupa prilikom uspešne i neuspešne pretrage. Odrediti verovatnoću popunjavanja za svaku praznu lokaciju tabele pri prvom narednom umetanju.

Rešenja:

4.

```
#ifndef HEAPSORT_H_
#define HEAPSORT_H_
class HeapSort : public SortAlgorithm{
protected:
      void makeHeap(int *array, int array_len);
      void rearrangeHeap(int *array, int array_len);
public:
      virtual void sort(int *array, int array_len);
};
#endif
```

```
#include "Heapsort.h"
void HeapSort::makeHeap(int *array, int array_len){
      for(int i = 1; i < array_len; i++) {
      int k = i;

            while( k > 0 )  {
                  if( array[k] > array[(k-1)>>1] ){
                  int t = array[k];

                        array[k] = array[(k-1)>>1];
                        array[(k-1)>>1] = t;
                        k = (k-1)>>1;
                  }
                  else  break;
            }
      }
}

void HeapSort::rearrangeHeap(int *array, int array_len) {
int i = 0;
      while( i < array_len ) {
      int left = (i<<1)+1, right = (i<<1)+2;
      int maxChildIndex = left;

            if( left >= array_len ) break;

            if( right < array_len &&
                array[left] < array[right])
                  maxChildIndex = right;

            if( array[maxChildIndex] > array[i] ) {
            int t = array[maxChildIndex];

                  array[maxChildIndex] = array[i];
                  array[i] = t;
                  i = maxChildIndex;
            }
            else
                  break;
      }
}
```

```
void HeapSort::sort(int *array, int array_len) {
      makeHeap(array, array_len);

      for(int i = array_len-1; i > 0; i--){
      int t = array[0];

            array[0] = array[i];
            array[i] = t;

            rearrangeHeap(array, i);
      }
}
```

```
#include <iostream>
using namespace std;

#include "heapsort.h"


void printArray(int *array, int array_len)
{
      for(int i = 0; i < array_len; i++)
            cout << array[i] << ' ';
}

void main()
{

int array[] = { 5, 3, 18, 22, 4, 97, 53, 23, 65, 47 };
int array_len = sizeof(array)/sizeof(int);

      printArray(array, array_len);
      cout << endl;

      Heapsort hs;
      hs.sort(array, array_len);

      printArray(array, array_len);
      cout << endl;

}
```

```
Ispis:
------
5 3 18 22 4 97 53 23 65 47

3 4 5 18 22 23 47 53 65 97
```

2. Staticki: 00 11 01 00 11 01 10 11 10 11 – 20 bita
Dinamicki: F 0A 00N 0 11 101 100L 11 001 11 (00 001 0010 0 11 101 10011 11 001 11) – 27 bita
LZW: 00 001 010 100 010 0011 0001 1001 (0 1 2 4 2 3 1 9) – 26 bita

6.

Ulaz	ključ
0	29
1	23
2	13
3	
4	34
5	
6	35
7	18
8	59
9	28
10	

K mod 3 = 0	K mod 3 = 1	K mod 3 = 2
0	0	0
2	3	4
4	6	8
6	9	1
8	1	5
10	4	9
1	7	2
3	10	6
5	2	10
7	5	3
9	8	7

Ulaz	Verovatnoća
3	1/3*(2/11+3/11+1/11) = 6/33
5	1/3*(1/11+2/11+6/11)= 9/33
10	1/3*(8/11+6/11+4/11)=18/33

Prosečan broj pristupa za uspešnu pretragu: 13/8 = 1.625

Prosečan broj pristupa za neuspešnu pretragu:

$$1+\frac{8}{11}+\frac{8\cdot 7}{11\cdot 10}+\frac{8\cdot 7\cdot 6}{11\cdot 10\cdot 9}+\frac{8\cdot 7\cdot 6\cdot 5}{11\cdot 10\cdot 9\cdot 8}+\frac{8\cdot 7\cdot 6\cdot 5\cdot 4}{11\cdot 10\cdot 9\cdot 8\cdot 7}+\frac{8\cdot 7\cdot 6\cdot 5\cdot 4\cdot 3}{11\cdot 10\cdot 9\cdot 8\cdot 7\cdot 6}+\frac{8\cdot 7\cdot 6\cdot 5\cdot 4\cdot 3\cdot 2}{11\cdot 10\cdot 9\cdot 8\cdot 7\cdot 6\cdot 5}+\frac{8\cdot 7\cdot 6\cdot 5\cdot 4\cdot 3\cdot 2\cdot 1}{11\cdot 10\cdot 9\cdot 8\cdot 7\cdot 6\cdot 5\cdot 4}=\frac{2970}{990}=3$$

Aproksimativno: $\frac{1}{1-\frac{8}{11}}=\frac{11}{3}=3.67$