JMBAG **IME I PREZIME**

Algoritmi i strukture podataka – završni ispit

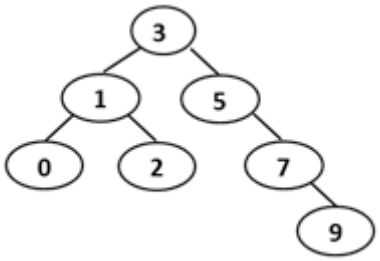
29. siječnja 2021.

Ispit donosi maksimalno 45 bodova. Ovaj primjerak ispita trebate predati s upisanim imenom i prezimenom te JMBAG-om.

Zadatak 1. (9 bodova)

Zadan je razred `BStablo` kojim se implementira binarno stablo:

```
template <typename T> class BStablo {
public:
   BStablo() : korijen(nullptr) {}
   …
protected:
   struct Cvor {
      T elem;
      shared_ptr<Cvor> lijevo, desno;
      Cvor(const T &novi) : …
   }
   shared_ptr<Cvor> korijen;
   …
};
```



Broj čvorova koji imaju samo jedno dijete: 2
To su čvorovi 5 i 7
Funkcija `brJednoDijete` za ovo stablo treba vratiti 2

a) Potrebno je napisati javni funkcijski član `brJednoDijete` razreda `BStablo`, koji će izbrojati koliko u stablu ima čvorova koji imaju samo jedno dijete, a koji je zadan prototipom:

```
int brJednoDijete();
```

   Dozvoljeno je koristiti pomoćni funkcijski član i pomoćne funkcije.

b) Napišite odsječak glavnog programa u kojemu se poziva funkcijski član `brJednoDijete`.

Zadatak 2. (9 bodova)

Potrebno je napisati konstruktor zadane klase `Graph` koji će inicijalizirati objekt, te rekurzivnu DFS metodu za obilazak i ispis grafa u dubinu počevši od zadanog vrha. Susjedi svakoga vrha su spremljeni u polje listi susjeda `*listeSusjeda`. Polju se pristupa pomoću indeksa vrha. Informacija je li pojedini vrh obiđen sprema se u polje `obidjen`. Napomena: za kretanje po listi stvoriti iterator.

```
#include<iostream>
#include <list>
using namespace std;
class Graph {
       int brojVrhova;
       list<int> *listeSusjeda; // polje listi susjeda
       bool *obidjen; // polje u kome je sadržana informacija je li pojedini vrh već obidjen
  public:
       Graph(int vrhovi);
       void dodajBrid(int izvor, int odrediste); // dodavanje bridova
       void postavi(); // postavljanje svih članova polja obidjen u false
       void DFS(int vrh);
};
// Inicijaliziraj graf
Graph::Graph(int vrhovi) …
// DFS algorithm
void Graph::DFS(int vrh) … // vrh od kojeg se kreće u grafu
```

Zadatak 3. (9 bodova)

Izračunati LPS polje koje se koristi u **Knuth-Morris-Pratt** algoritmu za niz `AACAAACAAAACA.`

A	A	C	A	A	A	C	A	A	A	A	C	A

Zadatak 4. (9 bodova)

Zadana su sučelja koja služe implementaciji tablice raspršenog adresiranja.

```
…
template <typename T, typename K> class IHashableValue {
 public:
   virtual K GetKey() const = 0;
};
template <typename T, typename K> class IHash {
 protected:
   size_t size;
   IHashableValue<T, K> **hash;

 public:
   virtual void Add(IHashableValue<T, K> *element) const = 0;
   virtual IHashableValue<T, K> *Get(K key) const = 0;
};
```

Svaki objekt ili zapis koji se upisuje u tablicu raspršenog adresiranja implementira sučelje `IHashableValue`. Varijabla `IHashableValue<T, K> **hash` služi pohrani tablice raspršenog adresiranja, a varijabla `size` određuje njenu veličinu.

Potrebno je napisati klasu `HashDoubleHashing` koja implementira sučelje `IHash` tako da koristi tehniku otvorenog adresiranja s dvostrukim raspršenim adresiranjem. Klasa `HashDoubleHashing` treba imati implementirane metode `Add i Get`.

Napomena: Konstruktor i destruktor nije potrebno implementirati.
Kao funkcije raspršenja potrebno je koristiti unaprijed definirane funkcije `int HashMultiplicationMethod1(int key) i int HashMultiplicationMethod2(int key).`

Zadatak 5. (9 bodova)

Zadan je niz brojeva:

`8, 7, 2, 1, 3, 4, 6, 5`

Ilustrirajte uzlazno sortiranje algoritmom **Quicksort**. Stožer za Quicksort bira se metodom aproksimacije medijana temeljem prvog, srednjeg i zadnjeg člana, pri čemu vrijedi da je cutoff = 3 nakon čega se sortira algoritmom Insertion sort.
Potrebno je prikazati sadržaj polja nakon svake promjene.

Rješenja:

1. zadatak (9 bodova)

a)

```
template <typename T> int BStablo<T>:: brJednoDijete() {
     if (korijen != nullptr) brJednoDijete (korijen);
};

template <typename T> int BStablo<T>:: brJednoDijete(shared_ptr<Cvor> &cvor) {
   int l, d;
   if (cvor != nullptr) {
      l = brJednoDijete(cvor->lijevo);
      d = brJednoDijete(cvor->desno);
      if ((cvor->lijevo == nullptr && cvor->desno!= nullptr) ||
          (cvor->lijevo!= nullptr && cvor->desno == nullptr)) {
         return l + d + 1;
      }
      return l + d;
   }
   return 0;
}
```

b) Odsječak glavnog progama, npr.:

```
BStablo<int> bStablo = BStablo<int>();
int rezultat = bStablo. brJednoDijete();
```

2. zadatak (9 bodova)

```
// Inicijaliziraj graf
Graph::Graph(int vrhovi) {
  brojVrhova = vrhovi;
  listeSusjeda = new list<int>[vrhovi];
  obidjen = new bool[vrhovi];
}

// DFS algorithmto
void Graph::DFS(int vrh) {
  obidjen[vertex] = true;
  list<int> listaSusjeda = listeSusjeda[vrh];
  cout << vertex << " ";
  list<int>::iterator i;
  for (i = listaSusjeda.begin(); i != listaSusjeda.end(); ++i)
    if (!obidjen[*i])
      DFS(*i);
}
```

3. zadatak (9 bodova)

A	A	C	A	A	A	C	A	A	A	A	C	A
0	1	0	1	2	2	3	4	5	6	2	3	4

4. zadatak (9 bodova)

```
template <typename T, typename K> class HashDoubleHashing : public IHash<T, K>
{
 private:
   IHashableValue<T, K> **hash;
   int HashMultiplicationMethod1(int key) const { return key % M; }
   int HashMultiplicationMethod2(int key) const { return 1 + key % (M - 1); }

 public:
   virtual void Add(IHashableValue<T, K> *element) const {
      int h1 = HashMultiplicationMethod1(element->GetKey());
      int h2 = HashMultiplicationMethod2(element->GetKey());
      int index;
      for (int i = 0; i < this->size; i++) {
         index = (h1 + i * h2) % this->size;
         if (hash[index] == nullptr) {
            hash[index] = element;
            break;
         }
      }
   }
   virtual IHashableValue<T, K> *Get(K key) const {
      int h1 = HashMultiplicationMethod1(key);
      int h2 = HashMultiplicationMethod2(key);
      int index;
      for (int i = 0; i < this->size; i++) {
         index = (h1 + i * h2) % this->size;
         if (hash[index] != nullptr && (*hash[index]).GetKey() == key) {
            return hash[index];
         }
      }
      return nullptr;
   }
};
```

5. zadatak (9 bodova)

8, 7, 2, 1, 3, 4, 6, 5 medijan

1, 7, 2, **5**, 3, 4, 6, 8 skrivanje st.

1, 7, 2, 6, 3, 4, **5**, 8 stožer je 5, zamjena 7 i 4

1, 4, 2, 6, 3, 7, **5**, 8 zamjena 6 i 3

1, 4, 2, 3, 6, 7, **5**, 8 vraćanje stožera na mjesto el. 6

1, 4, 2, 3, **5**, 7, 6, 8 L: medijan

1, **3**, 2, 4, **5**, 7, 6, 8 L: skrivanje stožera

1, 2, **3**, 4, **5**, 7, 6, 8 L: Stožer se vraća na samoga sebe (3 na 3)

1, 2, **3**, 4, **5**, 7, 6, 8 L: lijevi dio od 5 je sortiran (nema više posla)

1, 2, **3**, 4, **5**, 7, 6, 8 D: desno dio se sortira Insertion sortom

1, 2, **3**, 4, **5**, 7, 7, 8

1, 2, **3**, 4, **5**, 6, 7, 8