

Algoritmi i strukture podataka – ispit

14. veljače 2020.

Nije dopušteno korištenje globalnih i statičkih varijabli te naredbe goto. Ispit donosi maksimalno 70 bodova. Ovaj primjerak ispita trebate predati s upisanim imenom i prezimenom te JMBAG-om.

Zadatak 1. (14 bodova)

Zadan je razred Stack<T> kojim se implementira stog:

```
template <class T> class Stack{
public:
    Stack();
    bool push(T item);
    bool pop(T &item);
};
```

Potrebno je napisati funkciju uSredinu koja treba imati prototip

```
template <class T>
void uSredinu(Stack <T>* S);
```

Funkcija uSredinu treba preurediti ulazni stog na sljedeći način: neka je s X označen element na vrhu ulaznog stoga. Na izlazu, X treba biti na „pravom“ mjestu u smislu da se svi elementi stoga manji od X nalaze na stogu „ispod“ X, a svi elementi stoga veći od X se nalaze „iznad“ X. Ako se X pojavljuje više puta u originalnom stogu, u preuređenom stogu se treba pojaviti samo jednom. Redoslijed elemenata manjih od X treba ostati isti kao i originalnom stogu, dok redoslijed elemenata većih od X treba biti obrnut.

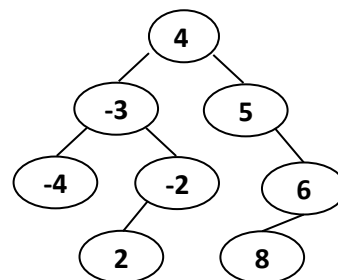
Primjer: oznaka **V#** označuje elemente veće od X, a **M#** elemente manje od X.

ulazni stog	X,	V1,	M1	M2,	M3,	V2,	M4,	X,	V3,	M5
	vrh stoga									
izlazni stog	V3,	V2,	V1,	X,	M1,	M2,	M3,	M4,	M5	

Zadatak 2. (15 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;
    ...
};
```



Primjer: Putevi u stablu od korijena do listova su
 4-> -3 -> -4 (zbroj -3)
 4-> -3 -> -2 -> 2 (zbroj 1)
 4-> 5 -> 6 -> 8 (zbroj 23)
 Funkcija brojPuteva bi trebala vratiti 1.

Potrebno je napisati javni funkcijski član brojPuteva razreda BStablo, koji će vratiti broj puteva od korijena do nekog lista stabla u kojima je zbroj vrijednosti vrhova manji od nule, a zadan je prototipom:

```
int brojPuteva();
```

Dozvoljeno je koristiti pomoćni funkcijski član i pomoćne funkcije. Napišite i odsječak glavnog programa u kojemu se poziva funkcijski član brojPuteva.

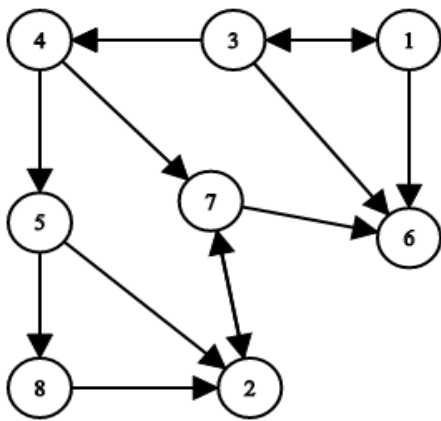
Zadatak 3. (10 bodova)

U cjelobrojnopolju pohranjen je sljedeći niz brojeva:

15, 2, 12, 8, 13, 4, 5, 9, 16, 9, 3

Ilustrirajte uzlazno sortiranje algoritmom Shellsort s koracima {4, 2, 1}. Potrebno je prikazati sadržaj polja nakon svake promjene.

Zadatak 4. (10 bodova)



Na slici lijevo prikazan je zadani **usmjereni graf**.

Prikažite kako izgleda ispis vrhova grafa korištenjem **DFS načina obilaska grafa (nerekurzivno)**, uz pretpostavku da se kreće od čvora s vrijednošću **3**. Prikažite korake u ispisu vrijednosti vrhova grafa (izlaz), redoslijed posjećivanja vrhova te sadržaj prateće pomoćne strukture u svakom koraku.

Napomena: prilikom odabira neobiđenih susjednih vrhova grafa, vrhove odabirati prema **rastućoj** vrijednosti vrha.

Zadatak 5. (15 bodova)

Zadana su sučelja koja služe implementaciji tablice raspršenog adresiranja.

```
#define c1 0.5
#define c2 0.5
...
template<typenameT,typenameK>class IHashableValue {
public:
    virtualKGetKey()const=0;
};
template<typenameT,typenameK>class IHash {
protected:
    size_tsize;
    IHashableValue<T,K>***hash;
public:
    virtual void Add(IHashableValue<T,K>*element) const=0;
    virtual IHashableValue<T,K>*Get(Kkey)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. Napišite javni funkcijski član Efikasnost sučelja IHash koji vraća udio elemenata upisanih u hash tablicu bez kolizije. Ako ne postoje elementi upisani u hash tablicu, funkcija treba vratiti -1. Prototip funkcije je

double Efikasnost();

Napomena: za funkciju određivanja adrese pretinca koristiti: int adr(K key);

Zadatak 6. (6 bodova)

Odredite vrijeme izvođenja u O , Ω i, ako je moguće, Θ notaciji za funkciju **f1**. Ako se vrijeme izvođenja u Θ notaciji ne može odrediti, navedite tako u rješenju. Rješenja upišite u tablicu s desne strane zadatka.

```
// pretpostavka:
//  $O(f(x,y)) = \Omega(f(x,y)) = \Theta(f(x,y)) = x$ 

int f1(int n){
    s=0;
    for(i=1; i<=n; i*=2)
        for(j=1; j<=n; j++)
            s+= f(n,j)+ f(n,i);
    return s;
}
```

O	
Ω	
Θ	

Rješenja:

1. zadatak (14 bodova)

```
template <class T>
void uSredinu(Stack <T>* S) {
    T item, originalniVrh;
    Stack <T> veci, manji;

    if (!S->pop(originalniVrh)) return;
    while (S->pop(item)) {
        if (item > originalniVrh) {
            veci.push(item);
        }
        else if (item < originalniVrh) {
            manji.push(item);
        }
    }
    // vrati manje na stog
    while (manji.pop(item)) S->push(item);
    // pa originalni vrh
    S->push(originalniVrh);
    // preokreni vece ...
    while (veci.pop(item)) manji.push(item);
    // i vrati na originalni
    while (manji.pop(item)) S->push(item);
}
```

2. zadatak (15 bodova)

```
int brojPuteva() { return brojPuteva(root, 0); }

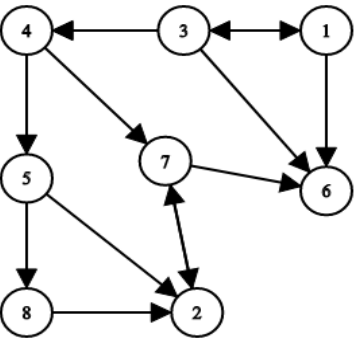
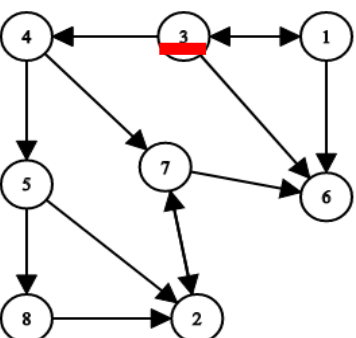
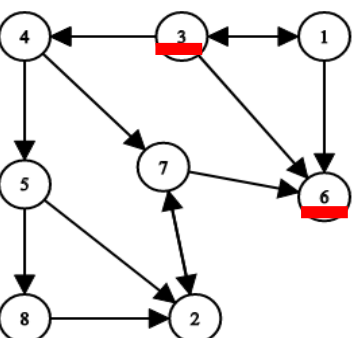
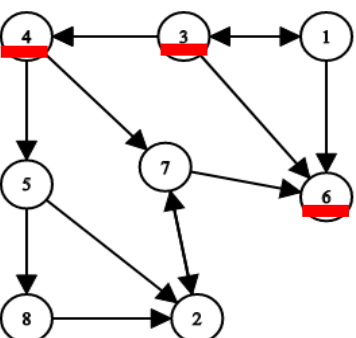
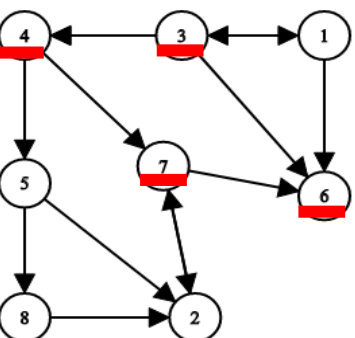
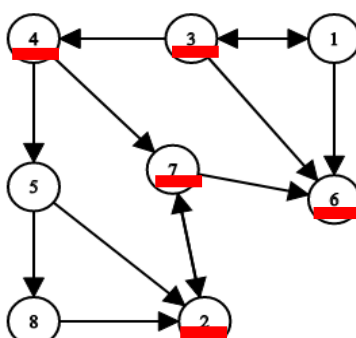
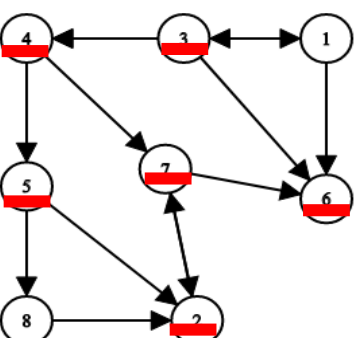
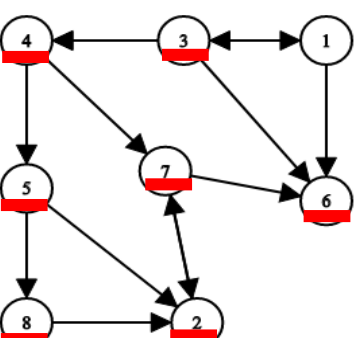
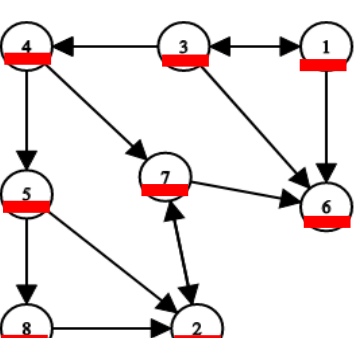
template <typename T>
int BinaryTree<T>::brojPuteva(shared_ptr<Node>& node, int prijenos) {
    // node je trenutni korijen podstabla, a u prijenos se nalazi zbroj
    // vrijednosti čvorova stabla od korijena do trenutnog čvora
    // ako smo u listu ...
    if (!node->left && !node->right) {
        // ako je zbroj vrhova manji od 0 vrati 1, 0 inače
        if (prijenos + node->item < 0) {
            return 1;
        }
        else return 0;
    }
    int no = 0;
    // ako postoji odgovarajuće dijete, kreni na tu stranu. Pazi da ažuriraš
    // prijenos (zbroj vrijednosti čvorova koji su prije na putu)
    if (node->left) no = brojPuteva(node->left, prijenos + node->item);
    if (node->right) no += brojPuteva(node->right, prijenos + node->item);
    return no;
}
```

3. zadatak (10 bodova)

Uokvireni su oni elementi koji se zamijenjuju. Siva boja pozadine označava elemente koji se dodatno zamijenjuju u nekom prolazu da bi se očuvala k-sortiranost niza.

15	2	12	8	13	4	5	9	16	9	3	
13	2	12	8	15	4	5	9	16	9	3	
13	2	5	8	15	4	12	9	16	9	3	
13	2	3	8	15	4	5	9	16	9	12	4-sortirano
13	2	3	8	15	4	5	9	16	9	12	
3	2	13	8	15	4	5	9	16	9	12	
3	2	13	4	15	8	5	9	16	9	12	
3	2	5	4	13	8	15	9	16	9	12	
3	2	5	4	12	8	13	9	15	9	16	2-sortirano
3	2	5	4	12	8	13	9	15	9	16	
3	2	5	4	12	8	13	9	15	9	16	
2	3	5	4	12	8	13	9	15	9	16	
2	3	4	5	12	8	13	9	15	9	16	
2	3	4	5	8	12	13	9	15	9	16	
2	3	4	5	8	9	12	13	15	9	16	
2	3	4	5	8	9	9	12	13	15	16	sortirano

4. zadatak (10 bodova)

	<p>Korak:1 Stog: 3 Ispis:</p>		<p>Korak:2 Stog: 1, 4, 6 Ispis: 3</p>
	<p>Korak: 3 Stog: 1, 4 Ispis: 3, 6</p>		<p>Korak: 4 Stog: 1, 5, 7 Ispis: 3, 6, 4</p>
	<p>Korak: 5 Stog: 1, 5, 2 Ispis: 3, 6, 4, 7</p>		<p>Korak: 6 Stog: 1, 5 Ispis: 3, 6, 4, 7, 2</p>
	<p>Korak: 7 Stog: 1, 8 Ispis: 3, 6, 4, 7, 2, 5</p>		<p>Korak: 8 Stog: 1 Ispis: 3, 6, 4, 7, 2, 5, 8</p>
	<p>Korak: 9 Stog: Ispis: 3, 6, 4, 7, 2, 5, 8, 1</p>	<p>Ako je sve točno 10 bodova, -2 boda po pogrešci. Procijeniti razumije li ispitanik algoritam. Rečeno je kako nije potrebno crtati graf za svaki korak.</p>	

5. zadatak

```
double efikasnost() {
    size_t i;
    size_t noItems, noSuccess;
    noItems = 0; noSuccess = 0;
    for (i = 0; i < this->size; i++) {
        if (hash[i] != nullptr){
            noItems++;
            // provjeri bi li hash[i] doista završio u pretincu i, pa ako je to istina
            // uvećaj brojač uspjeha za 1
            if (adr(hash[i]->GetKey()) == i)
                noSuccess++;
        }
    }
    if (noItems>0) return noSuccess / (1.0*noItems);
    return -1.0;
}
```

6. zadatak

sve složenosti su $\log_2 n$ n^2