JMBAG **IME I PREZIME**

Algoritmi i strukture podataka – ispit

14. srpnja 2020.

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);
};
```

Nadalje pretpostavlja se da za tip T postoji funkcija `dobar`, čiji je prototip

```
bool dobar(T item);
```

Potrebno je napisati funkciju `izmijesaj` koja treba imati prototip

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

Funkcija `izmijesaj` treba preurediti ulazni stog na način da se počevši od vrha stoga uzastopno izmjenjuju elementi koji su „dobri“ (za koje funkcija `dobar` vraća `true`) i oni koji „loši“ (funkcija `dobar` vraća `false`). „Dobri“ elementi trebaju biti u istom poretku u kakvom su bili u originalnom stogu, dok elementi koji su „loši“ trebaju biti u suprotnom poretku.

Primjer: oznaka **D#** označava „dobre“ elemente, a **L#** „loše“ elemente. Ako u nekoj grupi („dobrih“, odnosno „loših“ elemenata) ostane više elemenata, oni trebaju biti smješteni na dno stoga u odgovarajućem poretku.

ulazni stog	D1,	D2,	L1	L2,	L3,	D3,	L4,	L5,	D4,	L6
	vrh stoga									
izlazni stog	D1,	L6,	D2,	L5,	D3,	L4,	D4,	L3,	L2,	L1

Zadatak 2. (15 bodova)

Neka je zadano polje **a** koje se sastoji od **n** pozitivnih cijelih brojeva sortiranih silazno. Napišite rekurzivnu funkciju **`postojiZbroj`** koja će za zadani cijeli broj **m** vratiti 1, ako je **m** moguće napisati kao zbroj elemenata polja **a**, odnosno 0 ako to nije moguće. Podrazumijeva se da se elementi polja **a** mogu upotrijebiti samo po jednom. Prototip funkcije **`postojiZbroj`** treba biti

```
int postojiZbroj(int a[], int n, int m);
```

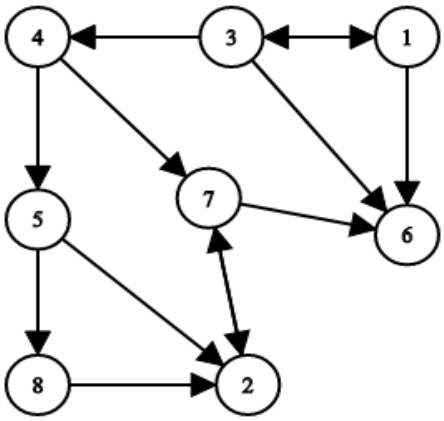
Zadatak 3. (10 bodova)

U cjelobrojnom polju pohranjen je sljedeći niz brojeva:

`10, 8, 6, 3, 5, 9, 1, 2, 4, 7`

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 bez navođenja koraka. 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 **BFS načina obilaska grafa (nerekurzivno)**, uz pretpostavku da se kreće od čvora s vrijednošću **1**. 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)

Zadan je razred `List<T>` kojim se implementira dinamička lista (lista realizirana pokazivačima).

```
template <typename T>
class List {
   ListElement<T> *head = nullptr;
};
```

```
template <typename T>
struct ListElement {
  T data;
  ListElement<T> *next;
};
```

Napisati člansku funkciju razreda `List<T>` koja modificira postojeću listu tako da između dva uzastopna elementa liste umetne veći od njih, ali samo ako su oba elementa manja od elementa `item` zadanog argumentom funkcije. U obzir uzimati samo izvorne čvorove liste.

Primjer: Neka je `item=10`. Originalna peteročlana lista (glava liste je prvi element slijeva) koja se sastoji od elemenata 2 8 12 9 7 nakon proširenja izgleda 2 **8** 8 12 9 **9** 7 (dodatni elementi su označeni masno i podcrtani).

Za usporedbu elemenata tipa T može se koristiti operatore >, <, >=, <= ili != (nije ih potrebno implementirati). Članska funkcija treba imati prototip:

```
void prosiri(T item)
```

Napomena: u izradi funkcije `prosiri` *nije dozvoljeno koristiti gotove funkcije za rad s listom (insert, delete i sl).*

Zadatak 6. (6 bodova)

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

```
if (n <= 5) {
   g(n); /* obavlja se u Θ(n) vremenu */
}
else {
   for (i = n - 1; i > 0; i--) {
     if (A[i] - A[i - 1] < 0) {
       g(i);
     }
   }
}
```

O	
Ω	
Θ	

1. zadatak (14 bodova)

```
template <class T>
void izmijesaj(Stack <T>* S) {
    T item, originalniVrh;
    Stack <T> SDobar, SLos1, SLos2;
    int brojD, brojL, i;

    brojD = 0; brojL = 0;
    // stavi svaki element na svoj stog ovisno o "dobroti"
    // prebroji koliko ima svakih ..
    while (S->pop(item)) {
        if (!dobar(item)) {
            SDobar.push(item);
            brojD++;
        }
        else {
            SLos1.push(item);
            brojL++;
        }
    }
    // "premotaj" loše jer trebaju na izlazu biti u suprotnom poretku
    while (SLos1.pop(item)) {
        SLos2.push(item);
    }
    // ako ima više dobrih, stavi prvo višak dobrih  u S
    if (brojD > brojL) {
        for (i = brojD; i > brojL; i--) {
            SDobar.pop(item);
            S->push(item);
       }
    }  // ako ima više loših, prvo stavi njihov višak u S
    else if (brojL>brojD) {
        for (i = brojL; i > brojD; i--) {
            SLos2.pop(item);
            S->push(item);
        }
    }
    // sada u oba stoga (dobar, los) ima jednak broj elemenata
    // pa ih mogu vratiti na polazni stog, prvo stavljam loš, pa dobar
    while (SLos2.pop(item)) {
        S->push(item);
        SDobar.pop(item); // ne treba provjeravati...
        S->push(item);
    }
}
```

2. zadatak (15 bodova)

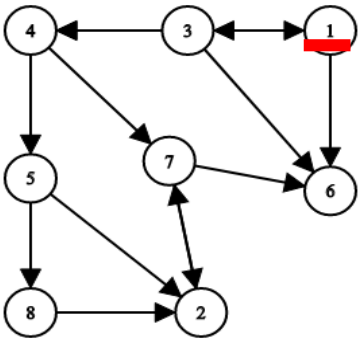
```
int postojiZbroj(int a[], int n, int m) {
    int i;
    if (n == 0) return 0; // ako u polju nema elemenata prekid
    for (i = 0; i < n; i++) {
        if (a[i] == m) return 1; // uspjeh
        // ako je element polja manji od m možeš probati
        if (a[i] < m) {
            // ako je uspjeh, prekini, ali ako nije uspjeh, treba probati s
            // drugim elementom ..
            if (postojiZbroj(&(a[i + 1]), n - i, m - a[i]) == 1) return 1;
        }
    }
    // ako se došlo do ovdje, nije bilo uspjeha ...
    return 0;
}
```

3. zadatak (10 bodova)

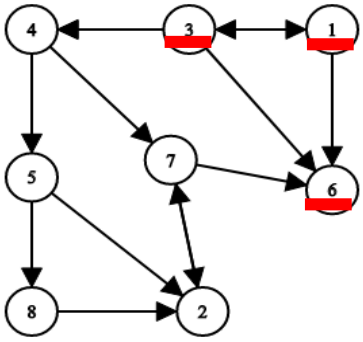
Boldani i podcrtani su elementi koji sudjeluju u operaciji

10	8	6	3	**5**	9	1	2	4	**7**	odabir stožera
5	8	6	3	**7**	9	1	2	**4**	10	sklanjanje stožera
5	**8**	6	3	4	9	1	**2**	7	10	i->2; 8<-j; zamjena
5	2	6	3	4	**9**	**1**	8	7	10	i->6; 7<-j; zamjena
5	2	6	3	4	1	**9**	8	**7**	10	povratak stožera
5	2	**6**	3	4	**1**	7	8	9	10	odabir stožera
1	2	**5**	3	**4**	6	7	8	9	10	sklanjanje stožera
1	2	4	3	**5**	6	7	8	9	10	stožer na pravom mjestu
1	**2**	4	**3**	5	6	7	8	9	10	odabir stožera
1	**2**	**4**	3	5	6	7	8	9	10	sklanjanje stožera
1	**4**	**2**	3	5	6	7	8	9	10	povratak stožera
1	2	**4**	**3**	5	6	7	8	9	10	sortiraj napamet
1	2	3	4	5	6	7	8	9	10	gotovo

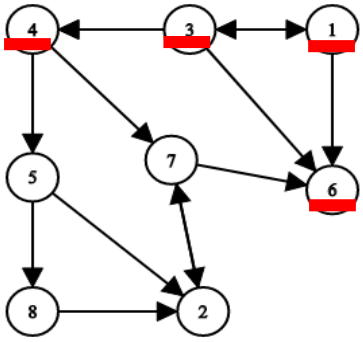
4. zadatak (10 bodova)



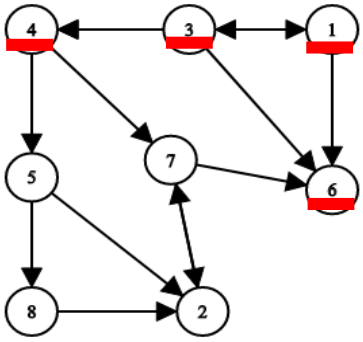
Korak:1
Red: 1
Ispis: 1



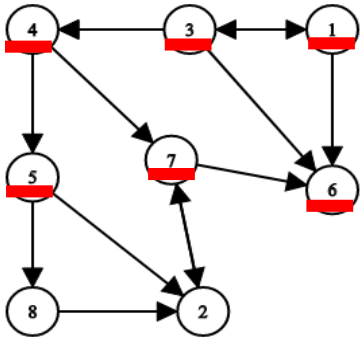
Korak:2
Red: 3, 6
Ispis: 1, 3, 6



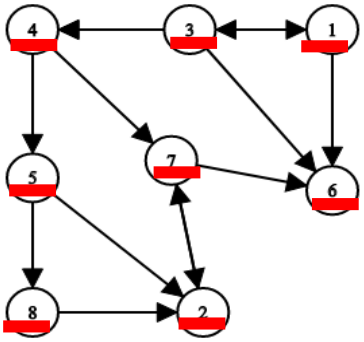
Korak: 3
Red: 6, 4
Ispis: 1, 3, 6, 4



Korak: 4
Red: 4
Ispis: 1, 3, 6, 4



Korak: 5
Red: 5, 7
Ispis: 1, 3, 6, 4, 5, 7



Korak: 6
Red: 7, 2, 8
Ispis: 1, 3, 6, 4, 5, 7, 2, 8

Još treba napraviti tri koraka u kojima se samo prazni red, ali nema nikakvih pomaka u ispisu.

5. zadatak (15 bodova)

```
void prosiri(T item) {
    ListElement<T>* p, *pom;
    ListElement<T>* novi;
    T veci;
    p = head;
    while (p) {
        // sljedeći element je pom
        pom = p->next;
        if (pom) { // ako postoji sljedeći element ...
            veci = pom->data; // koji je veći od p->data i pom->data
            if (p->data > pom->data) veci = p->data;
            if (veci < item) { // ako je item veći od veci, onda je veći od oba
                novi = new ListElement<T>; // stvori novi element
                novi->data = veci;
                novi->next = p->next; // postavi ispravno pokazivače
                p->next = novi;
            }
        }
        p = pom; // makni se dalje po originalnoj listi  (p=p->next je greška!)
    }
}
```

6. zadatak (6 bodova)

$O(n^2)$, $\Omega(n)$ (najbolji slučaj: silazno sortirano polje; najgori slučaj: uzlazno sortirano polje) ne postoji Θ