JMBAG **IME I PREZIME**

Algoritmi i strukture podataka – međuispit

25. studenoga 2019.

Ispit donosi maksimalno 30 bodova. Ovaj primjerak ispita trebate predati s upisanim imenom i prezimenom te JMBAG-om.

Zadatak 1. (6 bodova)

a) Napišite rekurzivnu funkciju koja provjerava nalazi li se niz `podniz` unutar niza `niz` počevši s pozicijom `pocetak`. Ako je to istina, funkcija treba vratiti logičku istinu, a inače logičku laž. Prototip funkcije je:

```
bool podnizUNizu(string niz, string podniz, int pocetak);
```

Primjeri: Ako je `niz` "Dobar dan!", `podniz` "dan" i `pocetak` 0, funkcija treba vratiti logičku laž. Ako je `niz` "Dobar dan!", `podniz` "dan" i `pocetak` 6, funkcija treba vratiti logičku istinu.

Napomene: Prazan podniz uvijek postoji unutar zadanog niza `niz`. Ako je niz `niz` prazan, funkcija treba vratiti logičku laž. Nerekurzivno rješenje se neće priznati.

b) Napišite odsječak glavnog programa u kojemu se poziva funkcija iz a) dijela zadatka.

c) Odredite složenost vaše implementacije funkcije iz a) dijela zadatka u O, Ω i, ako je moguće, Θ notaciji.

Zadatak 2. (6 bodova)

Odredite vrijeme izvođenja u O, Ω i, gdje je moguće, Θ notaciji za funkcije **f1** i **f2**. Ako se vrijeme izvođenja u Θ notaciji ne može odrediti, navedite tako u rješenju. Rješenja upišite u tablice pored zadataka.

a)

```
/* A je polje n cijelih brojeva.
 * Funkcija za sortiranje sortira niz uzlazno.
 * Funkcije implementiraju algoritme navedene u
 * imenima funkcija.
*/
void f1 (int A[], int n) {
   insertionSort(A, n);
   for (int i = n - 1; i >= 0; i--)
      binarySearch(A, n, A[i]);
}
```

O	
Ω	
Θ	

b)

```
int g(int i, int n) {
   int zbroj = 0;
   if (n <= 2)
      return 1;

   for (int k = i; k < n; k++) {
      zbroj += k;
      cout << zbroj << " " << endl;
   }
   return zbroj;
}

void f2(int n) { // n >= 0
   for (int i = 1; i <= n; i *= 2)
      cout << g(i, n) << endl;
}
```

O	
Ω	
Θ	

Okrenite stranicu

Zadatak 3. (6 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 iz liste izbacuje sve elemente veće od elementa `item` zadanog argumentom funkcije. Za usporedbu elemenata tipa T može se koristiti operatore >, <, >=, <= ili != (nije ih potrebno implementirati). Članska funkcija treba imati prototip:

```
void removeGreaterThan(T item);
```

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

Zadatak 4. (6 bodova)

Zadan je razred `Queue<T>` kojim se implementira red.

```
template <class T> class Queue {
public:
      Queue();
      bool enqueue(T data);
      bool dequeue(T &data);
};
```

Potrebno je napisati funkciju `split` koja treba imati prototip:

```
template <class T>
Queue<T> *split(Queue<T>* q);
```

Funkcija `split` red razdvaja na sljedeći način:

a) Svaki član na neparnoj poziciji gledano s izlaza iz reda (1. član, 3. član, 5. član itd.) stavlja u novi red (uz zadržani poredak članova ulaznog reda q) i tako formirani novi red vraća u pozivajuću funkciju.

b) Svaki član na parnoj poziciji gledano s izlaza iz reda (2. član, 4. član, 6. član itd.) ostavlja u ulaznom redu (uz zadržani poredak članova ulaznog reda q).

Napomena: U funkciji `split` nije dozvoljeno koristiti i/ili mijenjati internu implementaciju reda (polja ili pokazivače iz razreda `Queue<T>`), već je potrebno koristiti članske funkcije `enqueue` i `dequeue`.

Zadatak 5. (6 bodova)

Zadano je polje cijelih brojeva s elementima: **3, 9, 1, 6, 8, 5, 3, 7, 2.** Ilustrirajte uzlazno sortiranje zadanog niza brojeva (ispišite polje nakon svake promjene i podcrtajte sve brojeve relevantne za sljedeći korak) algoritmom **shellsort** uz korake K = {4, 3, 1}.

Zaokružite točan odgovor: općenito shellsort **je stabilan** / **nije stabilan** algoritam za sortiranje.

Rješenja:

1. zadatak

```
bool podnizUNizu(string niz, string podniz, int pocetak) {
      if (podniz[0] == '\0') return true;
      else if (niz[pocetak] == podniz[0]) {
            return podnizUNizu(niz, podniz.substr(1), pocetak + 1);
      }
      else return false;
}
```

b) Odsječak glavnog programa:

```
// #include <string>
string niz = "Dobar dan!", podniz = "dan";
cout << podnizUNizu(niz, podniz, 0) << endl;
cout << podnizUNizu(niz, podniz, 6) << endl;
```

c) Složenost funkcije iz a) dijela zadatka:

O(|podniz|), gdje je |podniz| duljina niza podniz; Ω(1); Θ se ne može odrediti

2. zadatak

a) $O(n^2)$, $\Omega(n \cdot \log n)$, Θ ne postoji

Za vrijeme izvođenja funkcije `insertionSort` vrijedi $\Omega(n)$ i $O(n^2)$.
Najbolji slučaj: `f1` prima uzlazno sortiran niz, pa je u tom slučaju vrijeme izvođenja $\Theta(n + n \cdot \log n) = \Theta(n \cdot \log n)$.
Najlošiji i prosječan slučaj: $\Theta(n^2 + n \cdot \log n) = \Theta(n^2)$. Za `f1` promatrano kroz sve moguće slučajeve vrijedi $\Omega(n \cdot \log n)$ i $O(n^2)$.

b) $O(n \cdot \log n)$, $\Omega(n \cdot \log n)$, $\Theta(n \cdot \log n)$
Funkcija g se poziva $\log_2 n$ puta, a za $n > 2$ petlja u funkciji g se obavlja $n - i$ puta.

3. zadatak

```
void removeGreaterThan(T item) {
   ListElement<T> **p;
   p = &head;
   while (*p) {
      for (; *p && item >= (*p)->data; p = &((*p)->next))
         ;
      if (*p) {
         ListElement<T> *tmp;
         tmp = *p;
         *p = (*p)->next;
         delete tmp;
      }
   }
}
```

4. zadatak

```
template<class T> Queue<T>* split(Queue<T>* q){
   Queue<T>* qnew = new Queue<T>();
   Queue<T> temp;
   T item;
   int sequence = 1;
   while(q->dequeue(item)){
      if(sequence % 2 == 1)
         qnew->enqueue(item);
      else{
         temp.enqueue(item);
      }
      sequence++;
   }
   while(temp.dequeue(item)){
      q->enqueue(item);
   }
   return qnew;
}
```

5. zadatak

3, 9, 1, 6, 8, 5, 3, 7, 2

	0	1	2	3	4	5	6	7	8
hk = 4 pom = 5	3	**9**	1	6	8	**5**	3	7	2
pom = 2	3	5	1	6	**8**	9	3	7	**2**
pom = 2	3	5	1	6	**8**	9	3	7	**8**
pom = 2	**3**	5	1	6	**3**	9	3	7	8
pom = 2	**2**	5	1	6	3	9	3	7	8
hk = 3 pom = 3	2	**5**	1	6	**3**	9	3	7	8
pom = 3	2	3	1	**6**	5	9	**3**	7	8
pom = 8	2	3	1	3	5	**9**	6	7	**8**
pom = 8	2	3	1	3	5	**8**	6	7	**9**
hk = 1 pom = 1	2	**3**	**1**	3	5	8	6	7	9
	2	**3**	**3**	3	5	8	6	7	9
	2	**2**	3	3	5	8	6	7	9
pom = 6	**1**	2	3	3	5	**8**	**6**	7	9
	1	2	3	3	5	**8**	**8**	7	9
	1	2	3	3	5	**6**	**8**	7	9
pom = 7	1	2	3	3	5	6	**8**	**7**	9
	1	2	3	3	5	6	**8**	**8**	9
Konačno rješenje	1	2	3	3	5	6	7	8	9