JMBAG **IME I PREZIME**

Algoritmi i strukture podataka – 3. Ispitni rok

18. rujna 2013.

*Nije dopušteno korištenje globalnih i statičkih varijabli te naredbe **goto**. Neefikasna rješenja mogu donijeti manje bodova. Nerekurzivne funkcije se ne priznaju kao rješenja u zadatcima u kojima se traži rekurzivna funkcija. Pismeni ispit donosi najviše **70** bodova, s time da je za prolazak potrebno ostvariti najmanje **35** bodova. Ovaj obrazac morate predati zajedno s cijelim Vašim uradkom.*

Zadatak 1. (11 bodova)

Jedan od načina računanja binomnih koeficijenta $\binom{n}{k}$, n≥0, k≥0, n≥k, je korištenjem rekurzivne formule $\binom{n}{k}=\binom{n-1}{k-1}+\binom{n-1}{k}$, s time da je $\binom{n}{0}=\binom{n}{n}=1$. Napišite rekurzivnu funkciju `binKoef` koja izračunava binomni koeficijent $\binom{n}{k}$, a kao argumente prima nenegativne cijele brojeve n i k.

Zadatak 2. (14 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. (13 bodova)

Zadan je niz brojeva: **23, 16, 7, 21, 2, 5, 6, 12**.

a) Ilustrirajte (nacrtajte stablo nakon svake promjene) stvaranje gomile s relacijom ***manji od*** (min heap) od zadanog polja brojeva algoritmom čija je složenost u najgorem slučaju *O(n).*

b) Za gomilu iz a) dijela zadatka prikažite postupak silaznog *heapsorta*. Prikažite svaki korak sortiranja (nacrtajte stablo i polje sortiranih elemenata nakon svake izmjene).

Zadatak 4. (15 bodova)

Tekst formatiranog dokumenta prikazan je jednostruko povezanom listom čiji su elementi strukture sljedeće građe:

```
struct s {
	char znak;
	int je_masan;
	int je_kos;
	struct s *slijed;
};
typedef struct s znak;
```

Varijabla `znak` sadrži ASCII vrijednost znaka u tekstu, varijabla `slijed` sadrži pokazivač na sljedeći znak ili NULL ako je to posljednji znak u dokumentu, varijabla `je_masan` sadrži logičku vrijednost koja označava je li znak masan (**bold**), a varijabla `je_kos` sadrži logičku vrijednost koja označava je li znak kos (*italic*).

a) Napisati funkciju prototipa

```
int brisi_ukrasene(znak **tekst, int masne, int kose);
```

koja će iz liste zadane pokazivačem na glavu (argument `tekst`) ovisno o argumentima `masne` i `kose` izbrisati masne znakove (za argument: `masne != 0`), bez obzira na to jesu li kosi, ili kose znakove (za argument: `kose != 0`), bez obzira na to jesu li masni, te vratiti broj izbrisanih elemenata.

b) *O čemu ovisi apriorna složenost te funkcije i kolika je apriorna složenost korištenog algoritma za tekst duljine n?*

Zadatak 5. (17 bodova)

Svaki zapis datoteke organizirane po načelu raspršenog adresiranja sadrži podatke o jednom proizvodu: *sifru proizvoda(int), naziv proizvoda (duljine do 50 znakova) te cijenu proizvoda(double)*.

Šifra nula (0) označava prazan zapis. Veličina bloka na disku je 4096 B. Očekuje se najviše 150000 zapisa, a kapacitet tablice je 15% veći. Prilikom upisa primjenjuje se metoda cikličkog preljeva. Ključ zapisa je *sifra*, a pretvorba ključa u adresu se obavlja već pripremljenom funkcijom

```
int adresa(int sifra);
```

Napišite funkciju koja će dinamički alocirati jednodimenzionalno polje i u njega upisati šifre i nazive proizvoda iz onih zapisa koji su završili u preljevu, a da je cijena proizvoda manja ili jednaka prosječnoj cijeni prozvoda čiji su zapisi završili u preljevu. Funkcija treba vratiti pokazivač na alocirano polje i broj elemenata. Prototip funkcije treba biti:

```
povratni_tip* fun(FILE *f, int *brZapisa);
```

Definirajte sve potrebne strukture i tipove podataka.

Rješenja:

Zad 1.

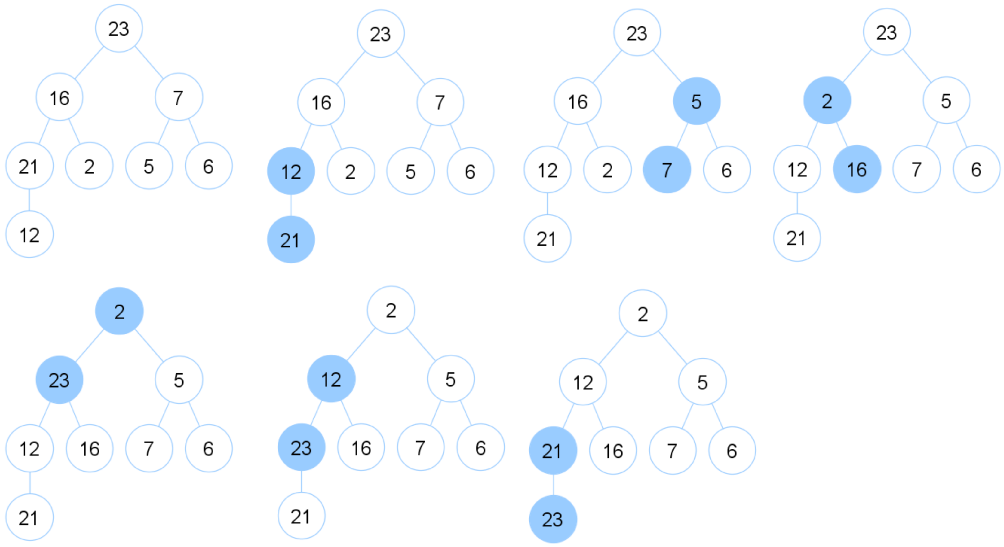
```
int binKoef(int n, int k)
{
	if (k==0) return 1;
	if (k==n) return 1;
	return binKoef(n-1,k-1)+binKoef(n-1,k);
}
```

Zad 2.

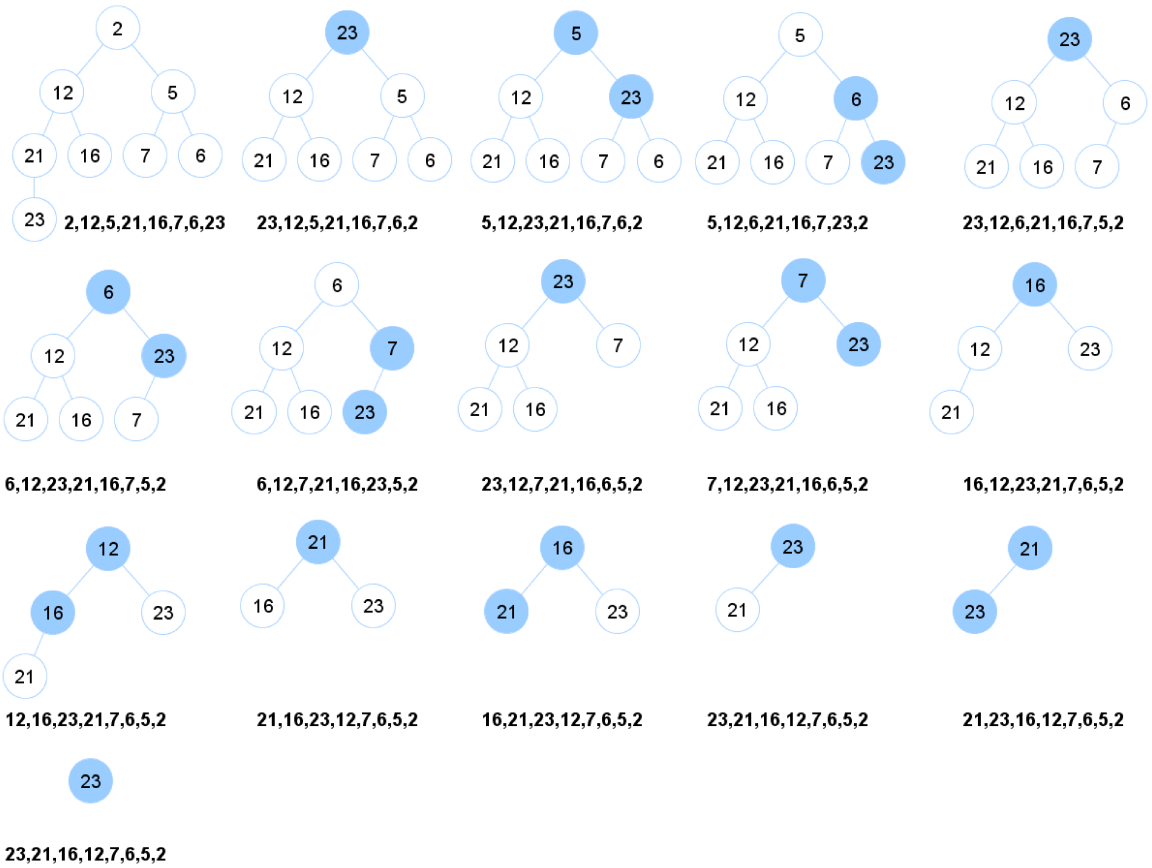
```
int postojiZbroj(int a[],int n, int m){
	int i, noviM;

	if(m<0) return 0;
	if (m==0) return 1;

	for(i=n-1; i>=0; i--){
		noviM = m - a[i];
		if (postojiZbroj(a,i,noviM)) return 1;
	}
	return 0;
}
```

Zad 3.

a)


23
16
7
21
2
5
6
12
23
16
7
12
2
5
6
21
23
16
5
12
2
7
6
21
23
2
5
12
16
7
6
21
2
23
5
12
16
7
6
21
2
12
5
23
16
7
6
21
2
12
5
21
16
7
6
23

b)


2
12
5
21
16
7
6
23
2,12,5,21,16,7,6,23
23
12
5
21
16
7
6
23,12,5,21,16,7,6,2
5
12
23
21
16
7
6
5,12,23,21,16,7,6,2
5
12
6
21
16
7
23
5,12,6,21,16,7,23,2
23
12
6
21
16
7
23,12,6,21,16,7,5,2
6
12
23
21
16
7
6,12,23,21,16,7,5,2
6
12
7
21
16
23
6,12,7,21,16,23,5,2
23
12
7
21
16
23,12,7,21,16,6,5,2
7
12
23
21
16
7,12,23,21,16,6,5,2
16
12
23
21
16,12,23,21,7,6,5,2
12
16
23
21
12,16,23,21,7,6,5,2
21
16
23
21,16,23,12,7,6,5,2
16
21
23
16,21,23,12,7,6,5,2
23
21
23,21,16,12,7,6,5,2
21
23
21,23,16,12,7,6,5,2
23
23,21,16,12,7,6,5,2

Zad4.

```
int brisi_ukrasene( znak **tekst, char masne, char kose ) {
        znak *pom = NULL;
        int pobrisani = 0;
        while( *tekst ) {
                if( ( ( *tekst ) -> je_masan && masne || ( *tekst ) -> je_kos && kose )  ) {
                        pom = *tekst;
                        *tekst = ( *tekst ) -> slijed;
                        free( pom );
                        pobrisani ++;
                }
                else {
                        tekst = & ( ( *tekst ) -> slijed );
                }
        }
        return pobrisani;
}
```

Zad5.

```
#define N 150000
#define BLOK 4096
#define C BLOK/sizeof (zapis)
#define M (int)(N*1.15/C)

typedef struct {
  int sifra;
  char naziv[50+1];
  double cijena;
} zapis;

typedef struct {
  int sifra;
  char naziv[50+1];
} povratni_tip;

int* fun(FILE *f, int *brZapisa){
  zapis pretinac[C];
  int i, j, predvidjeni_pretinac;
  int br_zapisa = 0;
  double suma = 0;
  double prosjek = 0;
  int *polje = NULL;

  //Brojanje zapisa
  for (i = 0; i < M; i++) {
    fseek (f, i*BLOK, SEEK_SET);
    fread (pretinac, sizeof (pretinac), 1, f);
    for (j = 0; j < C; j++) {
      if (pretinac[j].sifra != 0) { //preskačemo prazne zapise
        predvidjeni_pretinac = adresa(pretinac[j].sifra);
        if (predvidjeni_pretinac != i) {
          br_zapisa++;
          suma += pretinac[j].cijena;
      }
      else
        break; //nakon prvog praznog u pretincu, svi su ostali prazni
    }
  }

   //Racunanje prosjeka
   prosjek = (double) suma / br_zapisa;

   //Brojimo samo zapise koji su u preljevu i kojima je cijena <= prosjek
   br_zapisa = 0;

   //Trazenje elemenata i punjenje polja
  for (i = 0; i < M; i++) {
    fseek (f, i*BLOK, SEEK_SET);
    fread (pretinac, sizeof (pretinac), 1, f);
    for (j = 0; j < C; j++) {
      if (pretinac[j].sifra != 0) { //preskačemo prazne zapise
        predvidjeni_pretinac = adresa(pretinac[j].sifra);
        if (predvidjeni_pretinac != i && pretinac[j].cijena <= prosjek) {
          polje = (povratni_tip*) realloc(polje, (++br_zapisa) * sizeof(povratni_tip));
          polje[br_zapisa-1] .sifra= pretinac[j].sifra;
          strcpy(polje[br_zapisa-1] .naziv,  pretinac[j].naziv);
        }
      }
      else
        break; //nakon prvog praznog u pretincu, svi su ostali prazni
    }
```

```
 }

  *brZapisa = br_zapisa;

   return polje;
}
```