Algoritmi i strukture podataka

Završni ispit

2. srpnja 2010.

Napomena za sve zadatke:

- Nije dopušteno korištenje naredbe **goto** te statičkih i globalnih varijabli.

Zadatak 1. *(7 bodova)*

U jednostruko povezanoj listi sortiranoj silazno po prosjeku spremljeni su podaci o studentima. Lista je zadana strukturama:

```
struct zapis {
  char imeprezime[80+1];
  float prosjek;
};
struct at {
  struct zapis element;
  struct at *sljed;
};
typedef struct at atom;
```

Napišite funkciju koja će stvoriti novu listu u kojoj će podaci o studentima biti sortirani uzlazno. Ulazna lista mora ostati očuvana. Prototip funkcije je:

```
atom *okreniListu(atom *glava);
```

Zadatak 2. *(7 bodova)*

Binarno stablo sadrži cjelobrojne elemente te je čvor definiran sljedećim kodom:

```
typedef struct s {
 int vrijednost;
 struct s *lijevo, *desno;
} cvor;
```

Za svaki čvor binarnog stabla definiramo balans kao vrijednost čvora umanjenu za balanse njegovog lijevog i desnog djeteta. Ako čvor nema neko dijete, računa se da je balans tog djeteta nula.

a) Napišite rekurzivnu funkciju `balans` koja računa balans zadanog čvora stabla. Prototip funkcije je:

```
int balans(cvor *cv);
```

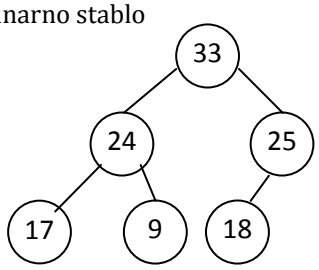
b) Koristeći funkciju `balans`, napišite rekurzivnu funkciju `najmanjiBalans` koja određuje kolika je vrijednost najmanjeg balansa čvora u stablu. Prototip funkcije je:

```
int najmanjiBalans(cvor *korijen);
```

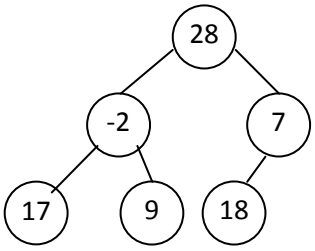
Funkciju `najmanjiBalans` možete napisati i ako ne napišete funkciju `balans` (pretpostavite da funkcija `balans` postoji i da ispravno radi). Pretpostavite da binarno stablo nije prazno. Nije dopušteno korištenje pomoćnih struktura (red, stog i slično).

Primjer: Na slici lijevo nalazi se primjer binarnog stabla, a desno se nalaze balansi čvorova stabla (npr. balans korijena je 28, a njegovog lijevog djeteta je -2).

Binarno stablo



Balansi čvorova stabla



Zadatak 3. *(6 bodova)*

Zadan je niz brojeva: 15, 8, 14, 35, 60, 6, 39, 37, 1.

a) Ilustrirajte (nacrtajte stablo nakon svake promjene) stvaranje gomile od zadanog polja brojeva algoritmom čija je složenost za najgori slučaj *O(n).*

b) Ilustrirajte (nacrtajte stablo nakon svake promjene) stvaranje gomile od zadanog polja brojeva algoritmom čija je složenost za najgori slučaj *O(nlogn).*

Zadatak 4. *(2 boda)*

Koristeći prostor nad crtama, napišite naredbe tako da funkcija iz jednostruko povezane liste izbacuje prvih N elemenata. Ako u listi ima manje od N elemenata, funkcija izbacuje sve elemente.

```
void izbaci(atom **glava, int n){
      int brojac = 0;
      atom *stari;

      _____________________________
      while ((pomocni != NULL) && (brojac < n)){

            _____________________________
            free (*glava);
            *glava = pomocni;
            brojac++;
      }
}
```

Zadatak 5. *(2 boda)*

Nacrtati binarno sortirano stablo za ulazni niz brojeva: 8, 12, 3, 5, 6, 27, 31, 4, 17, 20, 2.

Zadatak 6. *(2 boda)*

Koristeći prostor nad crtama, napišite naredbe tako da funkcija ubacuje element u gomilu, pri čemu je složenost takvog algoritma za najgori slučaj $O(n\log_2 n)$.

```
     void ubaci (tip A[], int j) {
       int i, k, novi;
       k = j;
       i = j/2;
       novi = A[j];

       while (_____________________________) {
         A[k] = A[i];
         k = i;
         i /= 2;
       }

       _____________________________;
     }
1.
```

Rješenje:

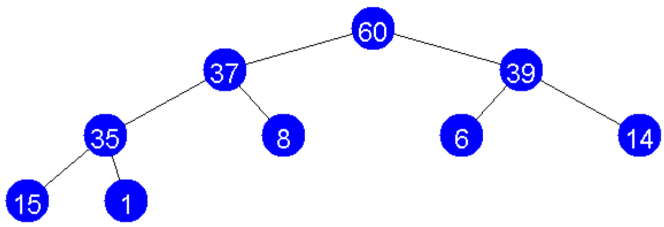
```
atom *okreniListu(atom *glava)
{
	atom *novi, *pom;
	pom =  NULL;
	while(glava != NULL)
	{
		novi = (atom *) malloc(sizeof(atom));
		novi->element.prosjek = glava->element.prosjek;
		strcpy(novi->element.imeprezime,glava->element.imeprezime);

// umjesto zadnja dva reda može i
// novi->element = glava->element;

		novi->sljed = pom;
		pom = novi;
		glava=glava->sljed;
	}
	return pom;
}
```

2.

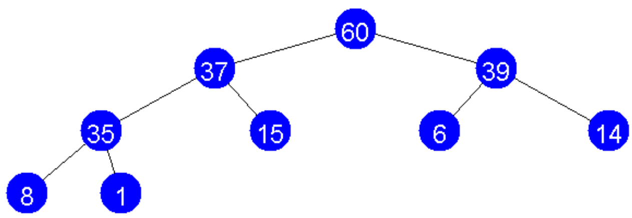
```
int balans(cvor *cv){
	int balans_lijevo, balans_desno;

	if (cv==NULL) return 0;
	balans_lijevo=balans(cv->lijevo);
	balans_desno=balans(cv->desno);
	return cv->vrijednost-(balans_lijevo+balans_desno);
}

int najmanjiBalans(cvor *korijen)
{
	// ideja: za zadani čvor korijen naći najmanji balans u podstablu
	int min, pmin;

	 // proglasim balans korijena za najmanji balans
	 min=balans(korijen);
	 // ako je korijen list, njegov balans je sigurno najmanji
	 if (korijen->lijevo==NULL && korijen->desno==NULL) return min;
	 // ako postoji lijevo podstablo, nađi najmanji balans u njemu i
	 // usporedi ga s balansom korijena te odredi novi najmanji balans
	 if (korijen->lijevo!=NULL){
		pmin=najmanjiBalans(korijen->lijevo);
		if (pmin<min) min=pmin;
	 }
	 // ako postoji desno podstablo, nađi najmanji balans u njemu i
	// usporedi ga s trenutnim najmanjim.
	 if (korijen->desno!=NULL){
		pmin=najmanjiBalans(korijen->desno);
		if (pmin<min) min=pmin;
	 }
	 return min;
}
```

3. U zadacima je trebalo ispisati korake, ovdje objavljujemo samo konačni izgled gomile.

a) napravi se potpuno stablo i radi se gomila. Rješenje:



b) dodaje se element po element i nakon svakog dodavanja podesi se gomila.



4.

```
void izbaci(atom **glava, int n){

      int brojac = 0;

       atom *pomocni = *glava;

   while ((pomocni != NULL) && (brojac < n)){

              pomocni = (*glava)->sljed;
              free (*glava);
              *glava = pomocni;
              brojac++;
       }

}
```

5.

```
                 8
            3             12
        2       5             27
             4     6      17       31
                             20
```

6.

```
void ubaci (tip A[], int j) {
  int i, k, novi;
  k = j;
  i = j/2;
  novi = A[j];
  while ( (i > 0) && (A[i] < novi ) {
    A[k] = A[i];
    k = i;
    i /= 2;
  }
A[k] = novi;

}
```