Algoritmi i strukture podataka - međuispit

25.travnja 2012.

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

Zadatak 1. *(5 bodova)*

Zadana je tablica raspršenog adresiranja s 8472 zapisa pohranjena u datoteci *"podaci.dat"* koja sadrži zapise definirane strukturom

```
typedef struct {
      int sifra;              //šifra studenta
      char prezimeIme[70+1];  //Prezime i ime studenta
      int logickiObrisan;     //Je li zapis logički obrisan
}  zapis;
```

Zapis je prazan ako je **`sifra = 0`**. Zapis je logički obrisan ako je **`logickiObrisan = 1`**. Preljevi su realizirani ciklički, upisom u prvi sljedeći slobodni pretinac. Pretinci su usklađeni s veličinom bloka od 512 okteta. Hash-tablica predimenzionirana je za 30%. Neka je ključ šifra studenta. Transformacija ključa u adresu obavlja se zadanom funkcijom **`int adresa (int sifra)`**. Napišite pretprocesorske direktive **`#define`** kojima se određuju parametri raspršenog adresiranja.

a) Napišite funkciju za logičko brisanje zapisa. Funkcija vraća 1 ako je pronašla i obrisala zapis a 0 inače. Funkcija ima prototip:
   `int brisi(int sifra, FILE *f)`
b) Napišite funkciju za dodavanje zapisa koja će zapis upisati na mjesto prvog praznog zapisa ili logički obrisanog zapisa u pretincu. Funkcija vraća 1 ako je uspješno dodala zapis a 0 inače. Funkcija ima prototip:
   `int upisi(int sifra, zapis novi, FILE *f)`

Zadatak 2. *(5 bodova)*

a) Napišite rekurzivnu funkciju koja će tehnikom raspolavljanja izračunati i vratiti sumu članova polja.
b) Napišite glavni program u kojem ćete učitati elemente polja s tipkovnice. Broj elemenata polja nije unaprijed poznat, a učitavanje se prekida kada se učita broj 0. Za pohranu elemenata potrebno je koristiti dinamički alocirano polje, a elementi se smiju učitati samo jednom.

Napomena: Nerekurzivno rješenje neće se priznavati.

Zadatak 3. *(5 bodova)*

a) Napišite bilo kakvu funkciju vremenske složenosti $O(2^n)$. **n** neka bude neki cjelobrojni ulazni argument.
b) Kolika je vremenska složenost izračunavanja faktorijela nekog broja **n** u O notaciji?
c) Kolika je vremenska složenost pronalaska svih permutacija nekog niza u O notaciji (u ovisnosti o **n**, duljini niza)?

Zadatak 4. *(5 bodova)*

a) Napišite matematičku definiciju algoritma *quicksort* u 4 koraka (quicksort(S))
b) Napišite **rekurzivnu** funkciju prototipa:

```
int qsort(int* niz, int lijevo, int desno);
```

koja će sortirati ulazni niz quicksort metodom. Na raspolaganju sljedeće funkcije:

```
int medijan(int* niz, int lijevo, int desno);
```

Funkcija kao argumente prima niz te indekse krajnjeg lijevog i krajnjeg desnog elementa podniza unutar kojega se traži stožerni element. Funkcija vraća indeks stožernog elementa (medijana).

```
int preslozi(int* niz, int stozerInd, int lijevo, int desno);
```

Funkcija kao argumente prima niz, indeks stožera te indekse krajnjeg lijevog i krajnjeg desnog elementa podniza unutar kojega se radi preslagivanje. Funkcija vraća indeks novog stožera.

Zadatak 5. *(5 bodova)*

Zadano je polje brojeva s elementima: **9, 6, 7, 2, 1, 8, 4, 5, 0, 3**. Ilustrirajte uzlazno sortiranje zadanog niza brojeva (ispišite polje nakon svake promjene i označite sve brojeve relevantne za sljedeći korak) algoritmom **mergesort** i algoritmom **shellsort** za niz koraka {4,3,1}.

Rješenja

1. zadatak

```
#define BLOK 512L
#define N 8472
#define C ((int) (BLOK / sizeof (zapis)))
#define M ((int) (N / C *1.3))
typedef struct {
        int sifra;
        char prezimeIme[70+1];
        int logickiObrisan;
}  zapis;

int brisi(int sifra, FILE *f){
        zapis pretinac [C];
        int i = 0, poc = 0;
        int adr = adresa(sifra);
        poc = adr;
        do{
                //Procitaj pretinac odredjen sa adresom
                fseek(f, adr * BLOK, SEEK_SET);
                fread (pretinac, sizeof (pretinac), 1, f);
                //Provjeri zapise u pretincu
                for(i=0; i<C; i++){
                        if(pretinac[i].sifra == sifra){
                                pretinac[i].logickiObrisan = 1;
                                //Upis
                                fseek(f, -1 * BLOK, SEEK_CUR);
                                fwrite (pretinac, sizeof (pretinac), 1, f);
                                return 1;
                        }
                }
                //Pretinac je pun, prijeđi ciklički na sljedećega
                adr = (adr + 1) % M;
        } while(adr != poc);
        return 0;
}

int upisi(int sifra, zapis novi, FILE *f){
        zapis pretinac [C];
        int i = 0, poc = 0;
        int adr = adresa(sifra);
        poc = adr;
        do{
                //Procitaj pretinac odredjen sa adresom
                fseek(f, adr * BLOK, SEEK_SET);
                fread (pretinac, sizeof (pretinac), 1, f);
                for(i=0; i<C; i++){
                        if(pretinac[i].sifra == 0 ||
                                pretinac[i].logickiObrisan == 1){
                                //Zapis je prazan ili je logicki obrisan
                                pretinac[i] = novi;
                                //Upis
                                fseek(f, -1 * BLOK, SEEK_CUR);
                                fwrite (pretinac, sizeof (pretinac), 1, f);
                                return 1;
                        }
                }
```

```
                //Pretinac je pun, prijeđi ciklički na sljedećega
                adr = (adr + 1) % M;
        } while(adr != poc);
        return 0;
}
```

2. zadatak

```
int suma(int a[], int lijevo, int desno) {
        int sred = 0;
        if(lijevo > desno) {
                return 0;
        } else if(lijevo == desno){
                return a[lijevo];
        } else {
                sred = ((desno - lijevo) / 2) + lijevo;
                return a[sred] + suma(a,lijevo,sred-1) + suma(a,sred+1,desno);
        }
}
```

ili

```
int suma (int a[], int duljina){
    if (duljina == 0) {
        return 0;
    }
    if (duljina == 1) {
        return a[0];
    }
    return suma (a, duljina/2) + suma (a+duljina/2, duljina-duljina/2);
}

int main(){
        int *polje = NULL, element, cnt = 0, s = 0;

        do {
                printf("Unesite %d. element: ", cnt + 1);
                scanf("%d", &element);
                if(element != 0){
                        polje = (int*)realloc(polje, sizeof(int) * (cnt + 1));
                        polje[cnt] = element;
                        cnt++;
                }
        } while(element != 0);

        s = suma(polje, 0, cnt - 1);
        printf("Suma clanova polja je %d", s);
        return 0;
}
```

3. zadatak

a) Npr.

```
#include <math.h>
void func(int n){
        int pon=pow(2,n)
        for (i=0;i<pon;i++);
}
```

Česta valjana alternativna rješenja

```
int func(int n){
        return func(n-1)+ func(n-1);
}

int func(int n){    //fibonnaci
        return func(n-1)+ func(n-2);
}
```

b) O(n)

c) O(n!)

4. zadatak

a) Slajdovi s predavanja:
4 koraka – *quicksort (S)*

- ako je broj članova polja *S* jednak 0 ili 1, povratak u pozivni program
- odabrati bilo koji član *v* u polju *S*. To je stožer (*pivot*)
- podijeli preostale članove polja *S*, $S \setminus \{v\}$ u dva odvojena skupa:
  - $S_1 = \{ x \in S \setminus \{v\} \mid x \leq v\}$ (sve što je manje od stožera, preseli lijevo)
  - $S_2 = \{ x \in S \setminus \{v\} \mid x \geq v\}$ (sve što je veće od stožera, preseli desno)
- vrati niz sastavljen od $\{quicksort\ (S_1), v, quicksort\ (S_2)\}$

b) Rekurzivna funkcija

```
void qsort(int* niz, int lijevo, int desno){
  int stozerInd;
  if(lijevo >= desno)
    return;
  stozerInd = medijan(niz, lijevo, desno);
  stozerInd = preslozi(niz, stozerInd, lijevo, desno);
  qsort(niz, lijevo, stozerInd - 1);
  qsort(niz, stozerInd + 1, desno);
}
```

5. zadatak

1. MergeSort

   (u slučaju uvijek veće ili jednake lijeve polovice)

   9, 6, 7, 2, 1, 8, 4, 5, 0, 3
   6, 9, 7, 2, 1, 8, 4, 5, 0, 3
   6, 7, 9, 2, 1, 8, 4, 5, 0, 3
   6, 7, 9, **1, 2**, 8, 4, 5, 0, 3
   1, 2, 6, 7, 9, 8, 4, 5, 0, 3
   1, 2, 6, 7, 9, **4, 8**, 5, 0, 3
   1, 2, 6, 7, 9, **4, 5, 8**, 0, 3
   1, 2, 6, 7, 9, **0, 3, 4, 5, 8**
   0, 1, 2, 3, 4, 5, 6, 7, 8, 9

   (u slučaju uvijek veće ili jednake desne polovice)

   9, 6, 7, 2, 1, 8, 4, 5, 0, 3
   6, 9, 7, 2, 1, 8, 4, 5, 0, 3
   6, 9, 7, **1, 2**, 8, 4, 5, 0, 3
   6, 9, **1, 2, 7**, 8, 4, 5, 0, 3
   1, 2, 6, 7, 9, 8, 4, 5, 0, 3
   1, 2, 6, 7, 9, **4, 8**, 5, 0, 3
   1, 2, 6, 7, 9, 4, 8, **0, 3, 5**
   1, 2, 6, 7, 9, **0, 3, 4, 5, 8**
   0, 1, 2, 3, 4, 5, 6, 7, 8, 9

2. Shell sort {4,3,1}

   9, 6, 7, 2, 1, 8, 4, 5, 0, 3
   1, 6, 7, 2, **9**, 8, 4, 5, 0, 3 t=4
   1, 6, **4**, 2, 9, 8, **7**, 5, 0, 3
   0, 6, 4, 2, **1**, 8, 7, 5, **9**, 3
   0, **3**, 4, 2, 1, **6**, 7, 5, 9, **8**
   0, **1**, 4, 2, **3**, 6, 7, 5, 9, 8 t=3
   0, 1, **2**, **4**, 3, 6, 7, 5, 9, 8 t=1
   0, 1, 2, **3**, **4**, 6, 7, 5, 9, 8
   0, 1, 2, 3, 4, **5**, **6**, **7**, 9, 8
   0, 1, 2, 3, 4, 5, 6, 7, **8**, **9**