Programiranje 2

Beleške sa vežbi

Školska 2007/2008 godina

Matematički fakultet, Beograd

Jelena Tomašević

June 11, 2008

Sadržaj

1

Programski jezik C

[1]

1.1 Grafovi

Graf `G=(V,E)` sastoji se od skupa čvorova `V` i skupa grana `E`. Grane predstavljaju relacije između čvorova i odgovaraju paru čvorova. Graf može biti usmeren (orijentisan), ako su mu grane uređeni parovi i neusmeren (neorjentisan) ako su mu grane neuređeni parovi.

Uobičajena su dva načina predstavljanja grafova. To su *matrica povezanosti* grafa i *lista povezanosti*.

Matrica povezanosti je kvadratna matrica dimenzije `n`, pri čemu je `n` broj čvorova u grafu, takva da je element na preseku `i`-te vrste i `j`-te kolone jednak jedinici ukoliko postoji grana u grafu od `i`-tog do `j`-tog čvora, inače je nula.

Umesto da se i sve nepostojeće grane eksplicitno predstavljaju u matrici povezanosti, mogu se formirati povezane liste od jedinica iz `i`-te vrste za `i=1,2,...,n`. To je lista povezanosti. Svakom čvoru se pridružuje povezana lista, koja sadrži sve grane susedne tom čvoru. Graf je predstavljen vektorom lista. Svaki elemenat vektora sadrži ime (indeks) čvora i pokazivač na njegovu listu susednih čvorova.

Prvi problem na koji se nailazi pri konstrukciji bilo kog algoritma za obradu grafa je kako izvršiti pregled grafa. Postoje dva osnovna algoritma za obilazak grafa: pretraga u dubinu (`DFS`, skraćenica od depth-first-search) i pretraga u širinu (`BFS`, skraćenica od breadth-first-search).

Kod `DFS` algoritma, obilazak započinje iz proizvoljno zadatog čvora `r` koji se naziva *koren pretrage u dubinu*. Koren se označava kao posećen. Zatim se bira proizvoljan neoznačen čvor `r1`, sussedan sa `r`, pa se iz čvora `r1` rekurzivno startuje pretraga u dubinu. Iz nekog nivoa rekurzije izlazi se kada se naiđe na čvor v kome su svi susedi već označeni.

Primer 1 *Primer predstavljanja grafa preko matrice povezanosti. U program se unosi neorijentisani graf i DFS algoritmom se utvrdjuju čvrovi koji su dostižni iz cvora sa rednim brojem 0.*

```
#include <stdlib.h>
#include <stdio.h>

int** alociraj_matricu(int n)
{   int **matrica;
    int i;
    matrica=malloc(n*sizeof(int*));
```

[1] Zasnovano na materijalu `Algoritmi, Miodrag Živković` i `http://www.matf.bg.ac.yu/~filip`

```
    for (i=0; i<n; i++)
       matrica[i]=calloc(n,sizeof(int));
    return matrica;
}

void oslobodi_matricu(int** matrica, int n)
{   int i;
    for (i=0; i<n; i++)
        free(matrica[i]);
    free(matrica);
}

int* alociraj_niz(int n)
{   int* niz;
    niz=calloc(n,sizeof(int));
    return niz;
}

void oslobodi_niz(int* niz)
{   free(niz);
}

void unesi_graf(int** graf, int n)
{     int i,j;
      for (i=0; i<n; i++)
         for (j=i; j<n; j++)
         {   printf("Da li su element %d i %d povezani : ",i,j);
             do
             {  scanf("%d",&graf[i][j]);
                graf[j][i]=graf[i][j];
             } while (graf[i][j]!=0 && graf[i][j]!=1);
         }
}

void ispisi_graf(int** graf, int n)
{     int i,j;
      for (i=0; i<n; i++)
      {  for (j=0; j<n; j++)
             printf("%d",graf[i][j]);
         printf("\n");
      }
}



/* Broj cvorova grafa (dimenzija matrice) */
int n;
/* Matrica povezanosti */
int **graf;


/* Pomocni vektor koji govori o tome koji su cvorovi posecivani
```

```
   tokom DFS obilaska */
int *posecen;

/* Rekurzivna implementacija DFS algoritma */
void poseti(int i)
{    int j;
     posecen[i]=1;
     printf("Posecujem cvor %d\n",i);
     for (j=0; j<n; j++)
        if (graf[i][j] && !posecen[j])
               poseti(j);
}


main()
{    int i, j;
     printf("Unesi broj cvorova : ");
     scanf("%d",&n);

     graf=alociraj_matricu(n);
     unesi_graf(graf,n);
     ispisi_graf(graf,n);

     posecen=alociraj_niz(n);
     poseti(0);

     oslobodi_niz(posecen);
     oslobodi_matricu(graf,n);

}
```

Primer 2 *Primer predstavljanja grafa preko niza listi suseda svakog od čvorova grafa. U program se unosi graf i DFS algoritmom se utvrdjuje koji su čvorovi dostižni iz čvora sa rednim brojem 0.*

```
#include <stdlib.h>
#include <stdio.h>

/* Cvor liste suseda */
typedef struct _cvor_liste
{  int broj; /* Indeks suseda */
   struct _cvor_liste* sledeci;
} cvor_liste;

/* Ubacivanje na pocetak liste */
cvor_liste* ubaci_u_listu(cvor_liste* lista, int broj)
{   cvor_liste* novi=malloc(sizeof(cvor_liste));
    novi->broj=broj;
    novi->sledeci=lista;
    return novi;
}


/* Brisanje liste */
```

```
void obrisi_listu(cvor_liste* lista)
{   if (lista)
    {  obrisi_listu(lista->sledeci);
       free(lista);
    }
}

/* Ispis liste */
void ispisi_listu(cvor_liste* lista)
{   if (lista)
    {  printf("%d ",lista->broj);
       ispisi_listu(lista->sledeci);
    }
}

/* Graf predstavlja niz pokazivaca na pocetke listi suseda */
#define MAX_BROJ_CVOROVA 100
cvor_liste* graf[MAX_BROJ_CVOROVA];
int broj_cvorova;


/* Rekurzivna implementacija DFS algoritma */
int posecen[MAX_BROJ_CVOROVA];
void poseti(int i)
{   cvor_liste* sused;
    printf("Posecujem cvor %d\n",i);
    posecen[i]=1;
    for( sused=graf[i]; sused!=NULL; sused=sused->sledeci)
        if (!posecen[sused->broj])
              poseti(sused->broj);
}


main()
{   int i;
    printf("Unesi broj cvorova grafa : ");
    scanf("%d",&broj_cvorova);
    for (i=0; i<broj_cvorova; i++)
    {    int br_suseda,j;

         graf[i]=NULL;

         printf("Koliko cvor %d ima suseda : ",i);
         scanf("%d",&br_suseda);
         for (j=0; j<br_suseda; j++)
         {    int sused;
              do
              {
                printf("Unesi broj %d.-tog suseda cvora %d : ",j,i);
                scanf("%d",&sused);
              } while (sused<1 && sused>broj_cvorova);
              graf[i]=ubaci_u_listu(graf[i],sused-1);
```

```
        }
    }

    for (i=0; i<broj_cvorova; i++)
    {   printf("%d - ",i);
        ispisi_listu(graf[i]);
        printf("\n");
    }

    poseti(0);
}
```

1.2 Zadaci za vežbu

Zadatak 1 *(**drugi kolokvijum 2006.**)Napisati program za:*

1. *formiranje orijentisanog grafa od n čvorova;*
2. *za par datih čvorova (čvorovi su zadati rednim brojevima) ispitati da li je jedan čvor dostupan iz drugog.*