Algoritmi i strukture podataka
vežbe 5

Mirko Stojadinović

6. novembar 2015

[1]

1 Grafovi

1.1 Osnovni pojmovi

Graf G = (V, E) se sastoji od skupa čvorova V i skupa grana E, pri čemu grane predstavljaju relacije između čvorova. Grane usmerenog grafa su uređeni parovi čvorova i redosled dva čvora koje povezuje grana je bitan. Grane neusmerenog grafa su neuređeni parovi čvorova.

Neusmereni graf G je stablo ako je ispunjen bilo koji od ekvivalentnih uslova:

1. G je povezan aciklični graf
2. G nema ciklusa i ciklus se formira dodavanjem bilo koje grane u G
3. bilo koja dva čvora mogu se povezati jedinstvenim putem
4. Graf G je povezan i ima n čvorova i $n-1$ granu
5. Graf G nema ciklusa i ima n čvorova i $n-1$ granu

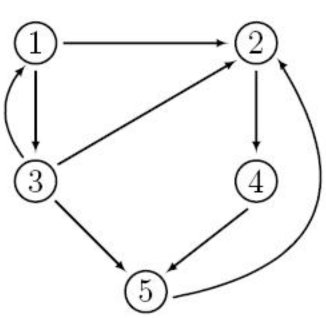
Ako je na stablu potrebno definisati hijerarhiju, onda se sve grane mogu usmeriti od "korena". Takva stabla se nazivaju korenska stabla. Koren je poseban izdvojen čvor stabla, a listovi stabla su čvorovi koji nemaju naslednike.

1.2 Predstavljanje grafova u računaru

Uobičajena su dva načina predstavljanja grafa:

1. matricom povezanosti (susedstva)
2. listom povezanosti (susedstva) - preko niza i jednostruko povezanih listi

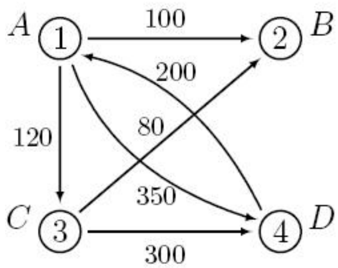
Matrica povezanosti grafa G je kvadratna matrica A reda n (|V| $=n$), gde A[i,j]=1 ako postoji grana od čvora v_i do čvora v_j. Ostali elementu su nule. Ako je graf G neusmeren, matrica je simetrična. Ako je broj grana u G mali, onda će većina elemenata matrice biti nule, ali će ona i dalje zauzimati prostor veličine n^2, što je nedostatak ovog tipa reprezentacije ako se koristi za retke grafove.



	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	1	0
3	1	1	0	0	1
4	0	0	0	0	1
5	0	1	0	0	0

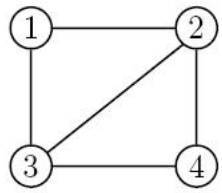
Matrica povezanosti težinskog grafa može se formirati na sledeći način:

[1]Materijali velikim delom preuzeti od Jelene Hadži-Purić: `www.math.rs/~jelenagr`



	1	2	3	4
1	∞	100	120	350
2	∞	∞	∞	∞
3	∞	80	∞	300
4	200	∞	∞	∞

Matrica povezanosti neusmerenog grafa (neusmerena grana (a,b) se predstavlja kao dve usmerene grane (a, b) i (b, a)):



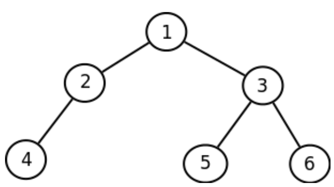
	1	2	3	4
1	0	1	1	0
2	1	0	1	1
3	1	1	0	1
4	0	1	1	0

Lista povezanosti pak omogućuje da se ne vrši eksplicitno predstavljanje nepostojećih grana. Postoje dve implementacije: preko niza (statički) i preko jednostruko povezanih listi (dinamički).

U slučaju jednostruko povezanih listi, svakom čvoru pridružuje se povezana lista koja sadrži sve susedne čvorove.

U slučaju niza, ako je G statički graf (tj. ne vrše se umetanja, brisanja), onda se za realizaciju liste povezanosti koristi niz dužine |V| +|E|, gde su prvih |V| elemenata pridruženi čvorovima grafa, a vrednost na poziciji j pridružena čvoru v_j sadrži indeks početka spiska čvorova susednih čvoru v_j.

Primer: slika grafa i njegova predstava preko liste povezanosti (niz):



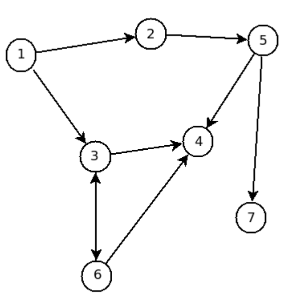
7	9	11	14	15	16	2	3	1	4	1	5	6	2	3	3
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Zadatak: Predstaviti prethodni graf preko jednostruko povezane liste.

Zadatak: Nacrtati graf koji je predstavljen listom povezanosti (preko niza):

8	10	11	-	13	15	-	2	3	5	4	6	4	7	3	4
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Rešenje:



Podrazumevano je da je graf usmeren (ukoliko se ne naglasi drukčije). Kod grafova imamo dva parametra koji određuju veličinu ulaza, ne jedan. To su broj čvorova ($n =$|V|) i broj grana ($m =$|E|).

1. Za graf koji ima n čvorova i m grana odrediti potreban prostor za predstavljanje grafa matricom povezanosti ili listom povezanosti.

Rešenje:

	matrica	lista
usmereni	n^2	$n+m$
neusmereni	$\frac{n\cdot(n+1)}{2}$	$n+2m$

Zašto je bitno da se složenost kod grafova procenjuje u odnosu na dva parametra, i n i m? Na početku kursa je bio zadatak da se pokaže da je $max(f,g) = \theta(f+g)$, ovde f i g mogu da se zamene sa n i m. Razmotriti ovo na primeru vrlo retkih i gustih grafova kada je potrebno odrediti broj grana koje vode do nekog čvora (npr. čvora 1) a koriste se liste povezanosti.

2. Za graf koji ima n čvorova i m grana odrediti da li je predstavljanje grafa pogodnije matricom povezanosti ili listom povezanosti u sledećim situacijama:

1. provera da li grana (x, y) pripada grafu
2. redak graf (m<< n^2)
3. dodavanje ili brisanje grana iz grafa
4. obilazak grafa

Rešenje:

1. matrica povezanosti (brže, zbog direktnog pristupa očitavanjem, tj. O(1) nasuprot O(duž.liste))
2. lista povezanosti (manje memorije tj. (m+n) nasuprot n^2)
3. matrica povezanosti (vreme O(1) nasuprot O(duž.liste))
4. lista povezanosti (brže, tj. O(m+n) nasuprot O(n^2))

Retki grafovi se danas češće javljaju nego gusti grafovi pa je to razlog što se liste povezanosti češće koriste. Ako se Web posmatra kao jedan veliki graf onda je on redak, jer od svake stranice postoji mali broj grana do drugih stranica (prosečno desetak). Za algoritme koji se odnose na obilaske mnogo bolja reprezentacija je lista povezanosti (to je velika većina algoritama na ovom kursu). Zato ćemo pri proceni složenosti pretpostavljati da je graf zadat listom povezanosti, ukoliko drukčije nije naglašeno. **Ako se u zadatku traži da algoritam bude složenosti $O(|\mathbf{V+E}|)$ onda je automatski određeno da je graf predstavljen listom povezanosti.**

Kod kojih operacija se razlikuje efikasnost implementacija preko niza i jednostruko povezanih listi?

3. Neka je G=(V,E) stablo sa n čvorova. Cilj je formirati simetričnu kvadratnu matricu A reda n, čiji elemenat (i,j) je jednak rastojanju između čvorova v_i i v_j. Konstruisati algoritam složenosti $O(n^2)$ koji rešava ovaj problem ako je stablo zadato listom povezanosti.

Rešenje:

Koristi se indukcija:

1. Rešenje je jednostavno za stablo sa jednim ili dva čvora.
2. Pretpostavka je da znamo da rešimo problem za $n-1$ čvor. Neka je S stablo sa n čvorova i neka je v list tog stabla a w otac čvora v (ako stablo ima bar 2 čvora jasne su egzistencija i jedinstvenost ovih čvorova). Uklanjanjem lista v dobija se stablo S' sa $n-1$ čvorova za koje za vreme $O((n-1)^2)$ znamo da odredimo rastojanje između svih čvorova. U konstrukciji matrice dimenzije n polazeći od stabla dimenzije $n-1$ i dodavanjem čvora v u stablo, važe sledeća pravila:
   (a) Vrednost polja koja određuju rastojanje dva čvora od kojih su oba različita od v ostaje ista kao i u matrici dimenzije $n-1$.

(b) Udaljenost čvora v od sebe samog je 0

(c) Udaljenost čvora v od oca je 1

(d) Udaljenost čvora v do nekog od ostalih čvorova je udaljenost oca od tog čvora uvećana za 1.

Vreme izvršavanja opisanog algoritma za problem dimenzije n je T(n), tj. T(n)=T(n-1)+(2n-1)*c1 , c1 je konstanta, jer iz matrice stabla sa n-1 čvorom se za dodatnih (2n-1)*c1 koraka dolazi do matrice stabla sa n čvorova, gde vrednosti matrice za k-tu vrstu i k-tu kolonu se računaju za const*(2n-1) koraka (matrica A je dimenzije n^2). Dakle,

$T(n) =$
$= T(n-1) + (2n-1) \cdot c1 =$
$= T(n-2) + (2n-3) \cdot c1 + (2n-1) \cdot c1 =$
$= ... =$
$= T(1) + (2n-1+2n-3+...+5+3) \cdot c1 =$
$= T(1) + \sum_{k=1}^{n-1}(2n-2k+1) \cdot c1 =$
$= T(1) + 2 \cdot c1 \cdot n(n-1) - 2 \cdot c1 \cdot \frac{n \cdot (n-1)}{2} + (n-1) \cdot c1$

Dobija se da je složenost algoritma $O(n^2)$.

1.3 Obilazak grafa u dubinu (DFS - Depth-First Search)

Motivacija. Najjednostavnija primena je provera da li su svi čvorovi u grafu dostupni iz nekog čvora. Ova provera se koristi npr. da se vidi da li su telefonske ili telekomunikacije veze dobro urađene, tj. da se proveri da neki čvor nije nedostupan.

Postoje dva obilaska grafova koja se najčešće koriste. DFS se koristi za agresivan pristup gde se ide što je moguće dalje u dubinu i radi se bektreking samo kad mora (primer je obilazak lavirinta). BFS se koristi kada je potrebno čvorove obići u nivoima ili se može koristiti za određivanje minimalnog rastojanja između dva čvora u grafu. Struktura podataka koja "odgovara" DFS algoritmu je stek, a struktura podataka koja "odgovara" BFS algoritmu je red.

DFS obilazak započinje iz proizvoljnog zadatog cvora r, korena pretrage u dubinu. Koren se označava kao posećen. Zatim se bira proizvoljni neoznačeni čvor r1, susedan sa r, pa se iz čvora r1 rekurzivno startuje pretraga u dubinu.

Iz nekog nivoa rekurzije izlazi se kad se naiđe na cvor v kome su svi susedi (ako ih ima) već označeni. Ako su u trenutku završetka pretrage iz r1, svi susedi čvora r označeni, onda se pretraga za čvor r završava. U protivnom, bira se sledeći proizvoljni neoznačeni sused r2 čvora r, izvršava se pretraga polazeći od r2, itd.

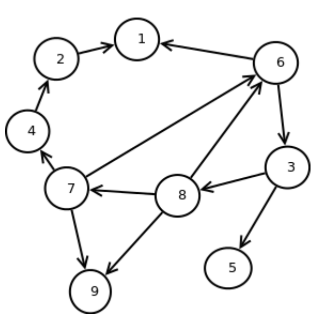
Pretraga grafa uvek se vrši sa nekim ciljem. Da bi se različite aplikacije uklopile u pretragu u dubinu, posećivanju čvora ili grane pridružuju se dve vrste obrade: ulazna obrada i izlazna obrada.

Ulazna obrada vrši se u trenutku označavanja čvora.

Izlazna obrada vrši se posle povratka nekom granom, ili kad se otkrije da neka grana vodi već označenom čvoru.

Ulazna i izlazna obrada zavise od konkretne primene DFS.

Primer DFS obilaska (iz čvora 8):



Redosled obilaska je: 8, 6, 1, 3, 5, 7, 4, 2, 9.

Redosled obilaska čvorova u opštem slučaju nije jedinstven. Mi uvodimo dodatno pravilo da se među susedima bira čvor sa najmanjom vrednošću pa obilazak u našem slučaju jeste jedinstven.

```
Algoritam DFS(G, v);
Ulaz: G = (V,E) (usmereni povezani graf) i v (cvor grafa G).
Izlaz: zavisi od primene.
{
  oznaci v;
  izvrsi ulaznu obradu na v; //ulazna obrada zavisi od primene DFS

  for sve grane (v,w) do
    if w je neoznacen then
      DFS(G,w);   //Novi rekurzivni poziv

  izvrsi izlaznu obradu na v;
}
```

4. Implementirati DFS algoritam za obilazak grafa G=(V,E) koji je zadat matricom povezanosti (npr. a[i][j]=1, ako postoji grana (i,j), inače a[i][j]=0) tako da ispiše čvorove u redosledu obilaska.
Rešenje:Naredni kod u .c fajlu

```
/* DFS implementacija u programskom jeziku C za graf dat matricom povezanosti */
/* Napomena: program ce obici sve cvorove i u slucaju NEPOVEZANOG grafa. */

#include <stdio.h>
#include <stdlib.h>
#define MAX 10 /*maksimalan broj cvorova */

void DFS(int i, int graf[][MAX], int n, int m[]);
void poseti_sve(int graf[][MAX], int n);
void ucitaj_matricu(int graf[][MAX], int n);


int main()
{
    int graf[MAX][MAX];
    int n; //broj cvorova grafa

    printf ("\nUnesite broj cvorova grafa: ");
    scanf ("%d", &n);
    if (n > MAX){
      fprintf (stderr, "n veci od alocirane dimenzije matrice.\n");
      exit (EXIT_FAILURE);
    }
    ucitaj_matricu (graf, n);
    poseti_sve (graf, n);
    return 0;
}

/* DFS obilazak grafa sa n cvorova zadatog matricom graf pocev od
   neposecenog cvora v; pri obilasku u nizu markiranih cvorova posecen,
   vrednost clana posecen[v] ce postati 1, kad se poseti cvor v. */
void DFS(int v, int graf[][MAX], int n, int posecen[])
{
    int w; //cvor koji je u grafu potencijalni sused cvora i

    //stampa se neposeceni cvor od kog krece nova poseta DFSom
    printf(" %d ", v);
    posecen[v] = 1; //markira se cvor i kao posecen

    /*ako postoji susedni cvor w koji nije markiran(posecen[w]=0),
      rekurzivno se poziva DFS za w. */
    for (w=0; w<n; w++)
       if (posecen[w] == 0 && graf[v][w] == 1)
```

```
            DFS(w, graf, n, posecen);
}

/* obilazak grafa G sa n cvorova zadatog matricom povezanosti graf */
void poseti_sve(int graf[][MAX], int n)
{
    int v, posecen[MAX]; // cvor i grafa G, niz markiranih (posecenih) cvorova posecen
    for (v=0; v<n; v++)
        posecen[v] = 0; // na pocetku su svi cvorovi neposeceni

    // ako v nije posecen, pokrenuti posetu iz v (osigurava da se svi
    // cvorovi obilaze i ako graf nije povezan)
    for (v=0; v<n; v++)
        if (posecen[v] == 0)
            DFS(v, graf, n, posecen);
    printf ("\n");
}

// Ucitavanje grana grafa sa stdin
void ucitaj_matricu(int graf[][MAX], int n)
{
    int i, j;
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            graf[i][j] = 0;

    printf("\nUnesite grane grafa u obliku i,j. Zavrsite sa EOF\n");
    while (scanf("%d%d", &i, &j) != EOF){
        graf[i][j] = 1;
    }
}
```

Složenost DFS algoritma koji koristi matrice je $O(n^2)$.

Zadatak. Napisati C kod DFS algoritma kao i pomoćnih algoritama koji koriste predstavljanje grafa preko liste povezanosti pomoću:

- niza
- jednostruko povezane liste

5. Početno vreme predstavlja vreme prvog nailaska na čvor u DFS algoritmu. Završno vreme se odnosi na vreme kada su obrađeni svi susedi čvora. Napisati pseudokodove za izračunavanje dolazne i odlazne numeracije.

Rešenje:

```
- G - graf
- dolazna[1...n] - niz za dolaznu numeraciju cvorova
- odlazna[1...n] - niz za odlaznu numeraciju cvorova
- posecen - niz koji oznacava koji su cvorovi poseceni (na pocetku su sve nule)
- inicijalizacija promenljivih pre poziva: *p1=0 i *p2=0

DFS_Preorder(G, u, posecen, dolazna, *p1)
  posecen[u] = 1;
  *p1 = *p1 + 1;
  dolazna[u] = *p1;
  za svaki v susedan za u do
    if posecen[v] = 0 then
        DFS_Preorder(G, v, posecen, dolazna, *p1);
  return;
```

```
DFS_Postorder(G, u, posecen, odlazna, *p2)
  posecen[u] = 1;
  za svaki v susedan za u do
    if posecen[v] = 0 then
        DFS_Postorder(G, v, posecen, odlazna, *p2);
  *p2 = *p2 + 1;
  odlazna[u] = *p2;
  return;
```

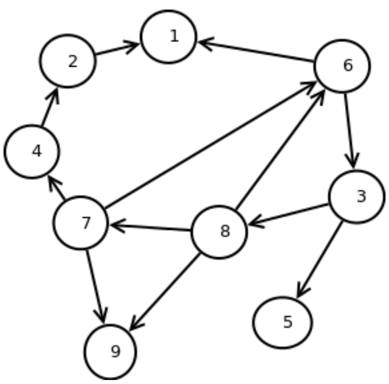
Prethodna dva algoritma mogu se spojiti u jedan tako da se vrši i ulazna i izlazna numeracija. Primetiti da dolazna numeracija odgovara KLD a odlazna LDK obilasku DFS stabla.

Svakom DFS obilasku odgovara jedinstveno DFS stablo. Postoje 4 vrste grana u DFS stablu:

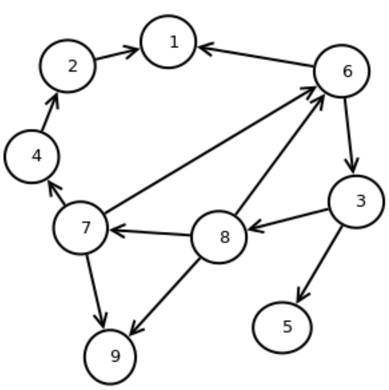
1. grana stabla (povezuje oca sa sinom)
2. povratna grana (povezuje potomka sa pretkom)
3. direktna grana (povezuje pretka sa potomkom)
4. poprečna grana povezuje čvorove koji nisu srodnici u stablu i prema dokazanoj lemi sa predavanja, ona moraju biti usmerene **zdesna ulevo**, te otud i naziv grana ulevo. Za DFS Stablo neusmerenog grafa, pak važi: grane povezanog neusmerenog grafa ne mogu biti poprečne grane za DFS stablo, tj. ne mogu povezivati čvorove na različitim putevima od korena. Kod neusmerenog grafa povratne i direktne grane se poklapaju jer nisu usmerene.

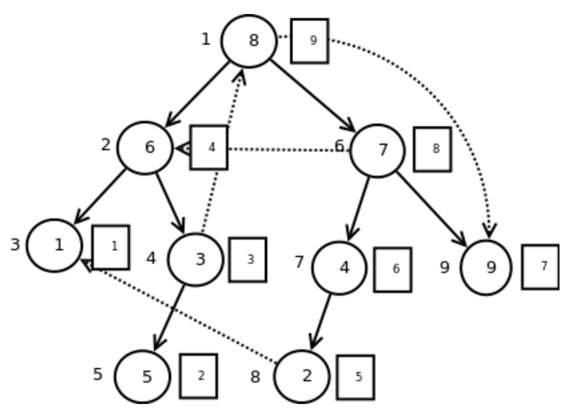
Može se pokazati da je usmereni graf ackiličan ako i samo ako DFS stablo nema povratnih grana.

6. Dat je usmeren graf na slici. Primeniti DFS obilazak, skicirati DFS stablo i za svaki čvor iz V prikazati početno vreme/završno vreme. Klasifikovati grane stabla. Napomena: grane grafa nisu uređene pa obilazak nije jedinstven.



Rešenje (početno vreme zapisano levo a završno desno od čvora):





Klasifikacija grana:

1. grane stabla: 86, 61, 63, 35, ...
2. direktne grane: 89.

3. povratne grane: 38.

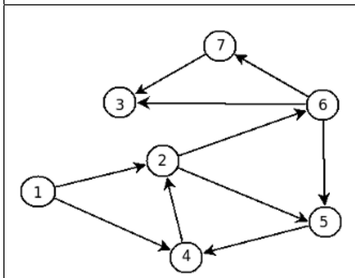
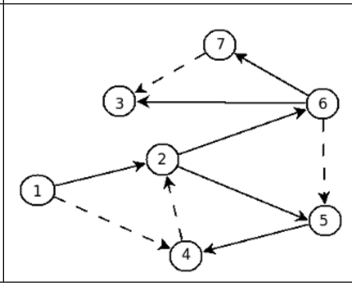
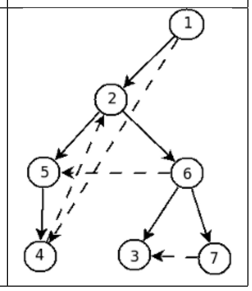
4. poprečne grane: 76, 21.

7. (april 2009) Usmeren graf G=(V,E) ima skup čvorova V=1, 2, 3, 4, 5, 6, 7 i zadat je listom povezanosti:

8	10	-	12	13	14	17	2	4	5	6	2	4	3	5	7	3
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Za početni poziv DFS(1), klasifikujte grane grafa u odnosu na DFS stablo. Pretpostavlja se da su grane (v,w) koje izlaze iz čvora v uređene numerički prema čvorovima w (u izboru suseda koji će biti posećen biramo onog sa najmanjim brojem).

Rešenje:

Graf:	Pomoćna slika:	DFS stablo
		

Grane stabla: 12, 25, 26, 54, 63, 67
Direktne: 14
Poprečne: 73, 65
Povratne: 42