METODE OPTIMIZACIJE
GRAFOVI, ALGORITMI

Dr Tina Dašić
Dr Miloš Stanić

Građevinski fakultet Univerziteta u Beogradu
2014.

UVOD

Podela prema vrsti optimizacionog problema

1. Broj kriterijumskih funkcija
   - jednokriterijumska
   - višekriterijumska
2. Ograničenja
   - bez ograničenja
   - sa ograničenjima
     - linearna
     - nelinearna
3. Forma kriterijumske funkcije
   - linearna
   - nelinearna
     - kvadratna
     - ...
4. Domen promenljivih
   - diskretan
   - realan
   - mešovit

Podela prema metodama optimizacije

1. Deterministčke
   - linearno programiranje (LP)
   - kvadratno programiranje (QP)
   - kombinatorna optimizacija (grafovi)
   - nelinearno programiranje (NLP)
     - sa gradijentima
     - bez gradijenata
2. Stohastičke (heuristike)
   - Skup rešenja
   - Jedno rešenje

1. GRAFOVI - UVOD

Grafovi su veoma česta matematička apstrakcija za rešavanje niza problema koji se mogu predstaviti u formi stvarnih ili apstraktnih objekata i stvarnih ili apstraktnih veza između tih objekata.

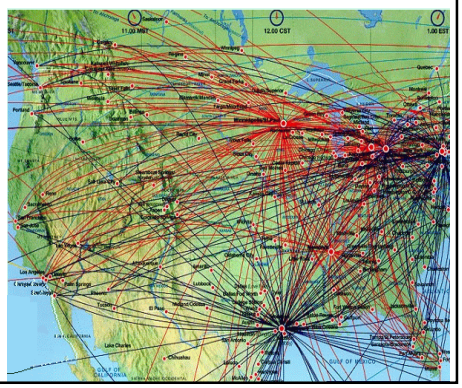
Primeri:

Saobraćajna, vodovodna, drenažna,

hidrografska mreža,

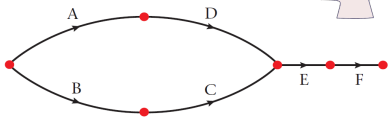
Telekomunikaciona, računarska,

elektro distributivna mreža



1. GRAFOVI - UVOD

Primeri abstraktnih mreža:

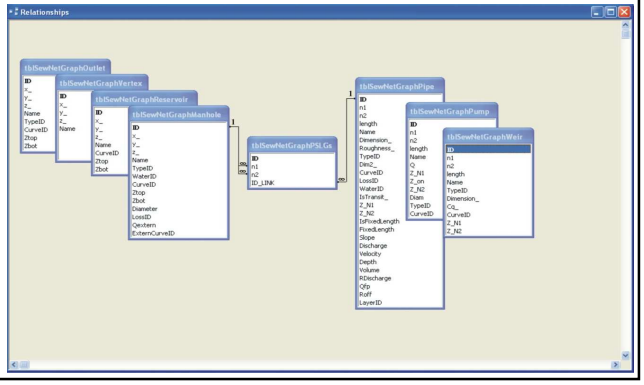


Dinamički plan

Bankarsko poslovanje (finansijsko računovodstvo)

Baze podataka (MOV)

Socijalne mreže

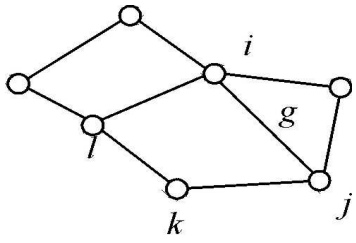


1. GRAFOVI - UVOD

Osnovni pojmovi

Definicija - Graf *G* čine dva konačna skupa: skup čvorova *N* i skup veza *C* takav da svaka veza $g \in C$ spaja dva čvora *i* i *j* iz *N* - $(i,j) \subset N$.

$$G = (N,C)$$



1. GRAFOVI - UVOD

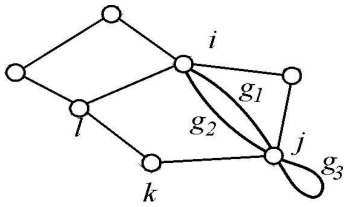
Osnovni pojmovi

Prost graf *G* je onaj u kome ne postoje:

paralelne veze – veze koje spajaju dva ista čvora
ciklične veze – veze koje spajaju čvor sa samim sobom

U prostom grafu par čvorova jednoznačno određuje vezu:

$$g = (i,j) = (j,i)$$



1. GRAFOVI - UVOD

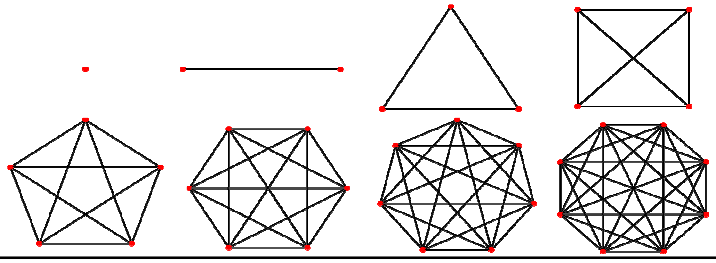
Osnovni pojmovi

Najveći broj veza u prostom grafu je jednak $NC(NC-1)/2$, gde je NC brojnost skupa čvorova N

Dokaz:

Najveći broj veza je NC^2, ali kada se izostave veze koje su ciklične i kojih ima NC i ostatak podeli sa 2 jer je u NC^2 svaka veza računata dva puta dobija se $(NC^2-NC)/2$

Kompletan graf je prost graf koji ima maksimalan broj veza.



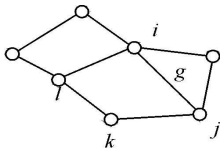
1. GRAFOVI - UVOD

Osnovni pojmovi

• Kada postoji veza koja spaja dva čvora kaže se da su ***čvorovi susedni***.

• ***Stepen čvora deg(i)*** je broj veza koji se sustiču u čvoru ***i*** .

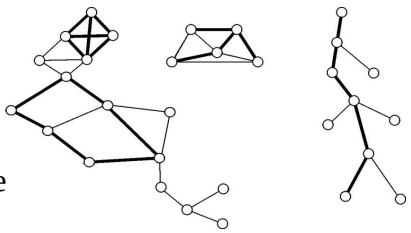
$\deg(i) = 4$



• ***Podgraf*** grafa G je podskup veza i grupe čvorova koji pripadaju tim vezama.

• Ako su čvorovi grafa konkretni objekti koji imaju definisanu poziciju u prostoru onda se radi o ***Euklidovim grafovima***.

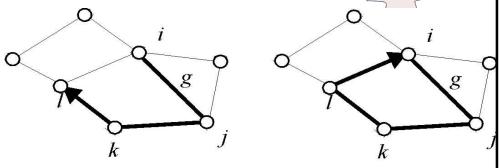
• ***Planarni graf*** je onaj Euklidov graf u kome se veze nacrtane u ravni ne seku.



1. GRAFOVI - UVOD

Osnovni pojmovi

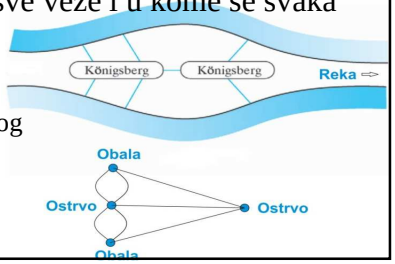
Definicija - ***Putanja*** je niz čvorova iz *N* gde svaka dva susedna čvora predstavljaju vezu iz *C*.



Ako putanja počinje i završava se u istom čvoru, onda se radi o ***prstenu*** (ciklusu ili cikličnoj putanji).

Prost put je niz u kome se čvorovi ne ponavljaju.
Euler-ov put (ciklus) je put (ciklus) koji sadrži sve veze i u kome se svaka veza pojavljuje samo jednom.

Ukoliko u povezanom grafu imamo tačno 2 čvora neparnog stepena tada graf poseduje Euler-ovu putanju. Ako nema čvorova neparnog stepena, postoji Euler-ov ciklus.



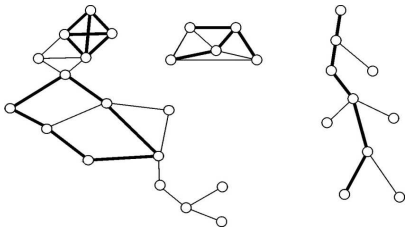
1. GRAFOVI - UVOD

Osnovni pojmovi

Definicija - ***graf je povezan*** ukoliko postoji barem jedna putanja koja spaja bilo koja dva čvora.

Graf koji nije povezan se sastoji od skupa povezanih delova, koji su ***maksimalni povezani podgrafovi***.

Pojam maksimalni povezani podgraf podrazumeva da ne postoji veza koja spaja čvor iz podgrafa sa bilo kojim čvorom izvan tog podgrafa.



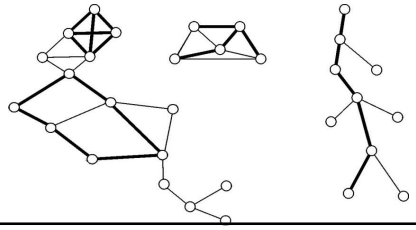
1. GRAFOVI - UVOD

Osnovni pojmovi

Definicija - ***Drvo*** - granata mreža, je povezan graf u kome nema prstenova.

Sledeća svojstva grafa koji je drvo se mogu dokazati:

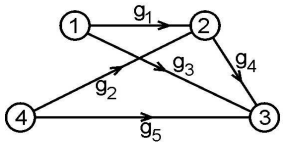
- broj veza je za jedan manji od broja čvorova
- postoji samo jedan prost put koji povezuje bilo koja dva čvora
- izostavljanjem bilo koje veze iz grafa on postaje nepovezan



1. GRAFOVI - UVOD

Osnovni pojmovi

Orijentisani (direkcioni) graf ili ***digraf***, je graf u kome se veza prikazuje kao uređeni par tačaka i (i,j) ≠ (j,i).



- U digrafu su dozvoljene paralelne veze koje imaju različitu orijentaciju.
- Prsten (ciklična putanja) u digrafu je niz veza koji se završava u početnoj tački.
- Najmanji broj tačaka koje čine prsten u digrafu je 2.
- Povezani digraf u kome nema prstenova se naziva ***Orijentisani aciklični graf*** - ***DAG*** (*Directed Acyclic Graph*).
- Kada se uz svaku vezu ili čvor pamti neki atribut koji je vezan za taj objekat, onda se radi o ***težinskom grafu***.

1. GRAFOVI - UVOD

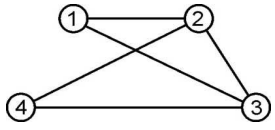
Načini za predstavljanje i čuvanje grafa

Načini za predstavljanje i čuvanje grafa:
- ***Matrica susednosti (A)***
- ***Matrica povezanosti***
- ***Lista susednih veza ili čvorova***

Matrica susednosti *A* je kvadratna NC x NC gde je NC broj čvorova

$$A = [a_{ij}]$$

$$a_{ij} = \begin{cases} 1, & (i,j) \in C \\ 0, & (i,j) \notin C \end{cases}$$



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

1. GRAFOVI - UVOD

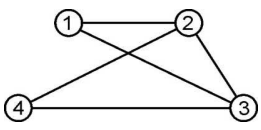
Načini za predstavljanje i čuvanje grafa

$$A = [a_{ij}]$$

$$a_{ij} = \begin{cases} 1, & (i,j) \in C \\ 0, & (i,j) \notin C \end{cases}$$

Matrica susednosti *A* je:
- Kvadratna NC x NC
- Za neorijentisane grafove matrica A je simetrična
- Za prost graf (bez cikličnih i paralelnih veza) matrica A ima nule po dijagonali
- Grafovi sa paralelnim vezama se ne mogu prikazati u okviru matrice A.
- Stepen čvora se može dobiti iz matrice *A:*

$$\deg(i) = \sum_j a_{ij}$$



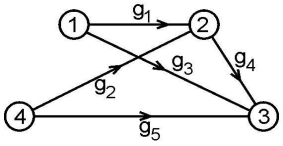
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

1. GRAFOVI - UVOD

Načini za predstavljanje i čuvanje grafa

Proizvod matrice susednosti A sa samom sobom (A^2) ukazuje na broj putanja čija je dužina jednaka 2. Pod dužinom putanje se podrazumeva broj veza sa kojima se stiže iz jedne u drugu tačku.

Ako je $a_{2,ij}$ oznaka za članove matrice A^2 onda se iz množenja matrica dobija: $a_{2,ij} = \sum_k a_{ik} a_{kj}$



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad A^2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Na isti način se može odrediti broj putanja dužine n:

$$A^n = A^{n-1} \cdot A$$

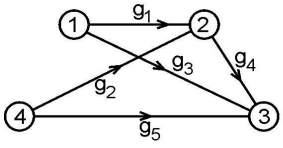
$$a_{n,ij} = \sum_k a_{n-1,ik} a_{kj}$$

1. GRAFOVI - UVOD

Načini za predstavljanje i čuvanje grafa

Ako je matrica ***B***: $B = A + A^2 + ... + A^n$

elementi matrice B (b_{ij}) predstavljaju ukupan broj putanja od čvora i do čvora j.



$$B = A + A^2 + \overset{0}{A^3} = \begin{bmatrix} 0 & 1 & 2 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

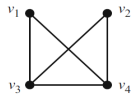
U orijentisanom grafu koji ima NC čvorova, maksimalna dužina putanje je NC-1, pa je to i maksimalna vrednost stepena n za određivanje matrice A^n.

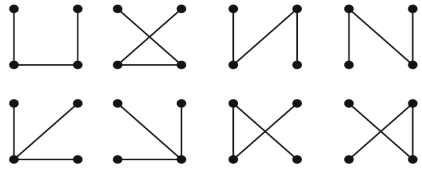
Stepenovanje matrice ima smisla dok se ne dobije nula matrica ($A^3==0$).

1. GRAFOVI - UVOD

Načini za predstavljanje i čuvanje grafa

*Definicija – **Razapinjuće stablo*** (Spanning Tree - **ST**) je podgraf povezanog grafa koji sadrži sve njegove čvorove i predstavlja drvo.





Broj granatih podgrafova (*Nst*)

$$D = [d_{ij}]$$

$$d_{ij} = \begin{cases} \deg(v_i), & i = j \\ 0, & i \neq j \end{cases}$$

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix},$$

Ako je M_{ij} je minor matrice *D-A*
Nst je kofaktor matrice *D-A*

$$N_{st} = (-1)^{i+j} M_{ij}$$

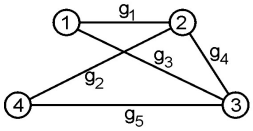
$$D - A = \begin{bmatrix} 2 & 0 & -1 & -1 \\ 0 & 2 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}.$$

1. GRAFOVI - UVOD

Načini za predstavljanje i čuvanje grafa

Predstavljanje grafova preko liste povezanosti može imati dve forme:

- Lista povezanosti (susednosti) koja se odnosi na čvorove
- Lista povezanosti koja se odnosi na veze



Čvor	Susedne veze
1	g_1 g_3
2	g_1 g_2 g_4
3	g_3 g_4 g_5
4	g_2 g_5

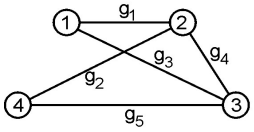
Veze	Krajnji čvorovi
g_1	1,2
g_2	4,2
g_3	1,3
g_4	2,3
g_5	4,3

Kada je u pitanju efikasnost po pitanju zauzeća memorijskog prostora prilikom čuvanja grafa, onda je sigurno prednost na strani čuvanja u formi lista (NC+NV).
Za graf čija je brojnost NC čvorova matrica A ima NC^2 elemenata.

1. GRAFOVI - UVOD

Načini za predstavljanje i čuvanje grafa

- Obično se u bazi podataka formira samo lista čvorova bez podataka o susednim vezama i lista veza sa podacima o paru čvorova koji čine tu vezu, da bi se izbegla redundantnost podataka.
- Kada je u pitanju implementacija algoritama za pretraživanje grafa, efikasnije je imati formirane kompletne liste u kojima bi se u okviru objekta „čvor“ čuvali i „pokazivači“ na liste susednih veza.



Čvor	Susedne veze
1	g_1 g_3
2	g_1 g_2 g_4
3	g_3 g_4 g_5
4	g_2 g_5

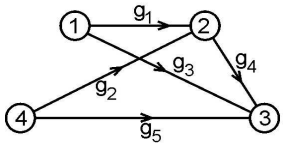
Veze	Krajnji čvorovi
g_1	1,2
g_2	4,2
g_3	1,3
g_4	2,3
g_5	4,3

1. GRAFOVI - UVOD

Načini za predstavljanje i čuvanje grafa

- Moguća je i implementacija u kojoj bi se u čvoru čuvale liste susednih čvorova, što bi praktično odgovaralo načinu na koji se čuva prazna matrica. Neodstatak ovog pristupa je u tome što bi se svaki put u listi veza morao tražiti objekat koji odgovara paru tačaka.

- Ako se radi o digrafu, onda se u čvoru odvojeno čuvaju veze koje idu iz čvora i veze koje idu u čvor.



Čvor	Veze iz čvora	Veze u čvor
1	g_1 g_3	
2	g_4	g_1 g_2
3		g_3 g_4 g_5
4	g_2 g_5	

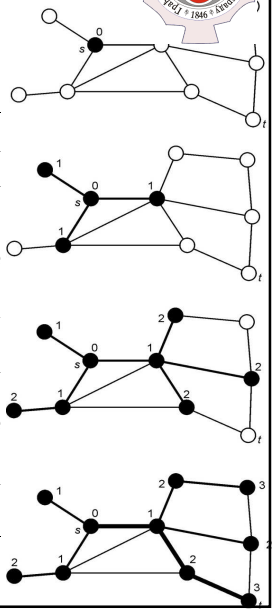
Veze	Čvor *start*	Čvor *end*
g_1	1	2
g_2	4	2
g_3	1	3
g_4	2	3
g_5	4	3

2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Pretraživanje po širini (Breadth First Search – BFS)

Moore-ov BFS algoritam za pronalaženje prostog puta

1. Čvor *s* se označi kao nulti i inicijalizuje se oznaka ranga čvora *r*=0.
2. Od skupa čvorova u grafu *N*, formiraju se dva komplementarna skupa N_1 i N_2. U prvom koraku u skupu N_1 nalazi se samo označeni čvor *s*, dok je skup N_2 sastavljen od svih preostalih čvorova u mreži.
3. Traže se grane koje povezuju čvorove ranga *r* iz N_1 sa neoznačenim čvorovima iz N_2. Identifikuju se sve veze g_k ($k=1,2,...$) koje ispunjavaju taj uslov. Ako ne postoje grane koje ispunjavaju ovaj uslov, algoritam se prekida
4. Čvorovi *j* iz N_2 koji su susedni čvorovima ranga *r* iz N_1, dobijaju vrednost ranga *r+1* i prebacuje se u N_1 čime je komplementarnost ovih skupova očuvana.
5. Proverava se da li je neki od čvorova (koji su upravo označeni) koji su najvišeg ranga (*r+1*) istovremeno i terminalni čvor *t*:
   - Ako još nije dostignut terminalni čvor, brojač ranga se povećava za 1: *r=r+1* i ide se ponovo na korak 3.
   - Ako jeste, putanja od *s* do *t* se nalazi tako što se kreće od krajnjeg čvora *t* i nalazi se njemu susedni čvor koji ima rang za jedan manji od njegovog ranga. Pretraživanje se nastavlja sve do čvora ranga 0, a to je čvor *s*.



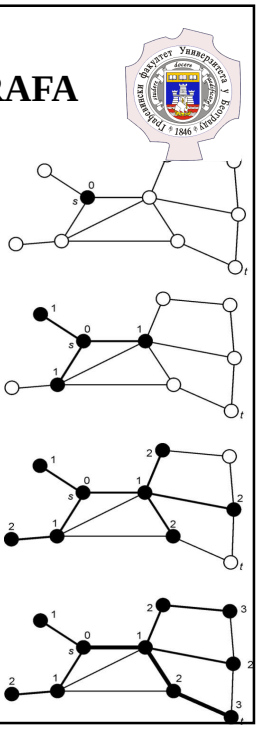
2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Pretraživanje po širini (**B**readth **F**irst **S**earch – **BFS)**

Pretraživanje po širini podrazumeva da se za svaki novi čvor, do koga se dosegne u pretraživanju, pronadju svi njemu susedni čvorovi koji nisu već bili posećeni. Pretraga se potom nastavlja korišćenjem novo označenih čvorova.

Oznaka *r* (*rang*) je rastojanje (*dužina puta merena brojem veza*) od početnog čvora (*s*)

BFS je jedan od najčešće korišćenih algoritama za pronalaženje prostog puta izmedju dva čvora: *s* i *t*.



2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Pretraživanje po širini (Breadth First Search – BFS)

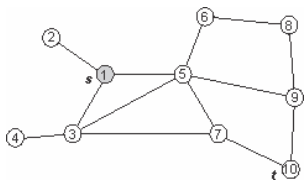
- Prost put od čvora *s* do čvora *t* nije jednoznačan
- Postoje 3 puta koji su ranga 3 sa kojima je moguće povezati čvorove *s* i *t*.
- Ovaj podatak se može dobiti iz matrice A^3

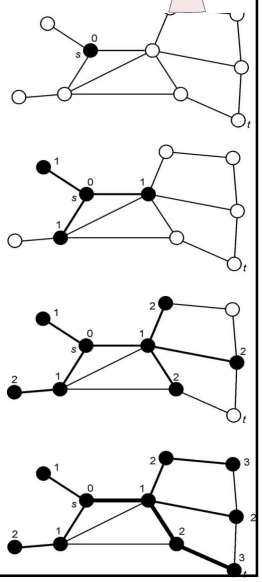
$A =$

1	2	3	4	5	6	7	8	9	10	
0	1	1	0	1	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	2
1	0	0	1	1	0	1	0	0	0	3
0	0	1	0	0	0	0	0	0	0	4
1	0	1	0	0	1	1	0	1	0	5
0	0	0	0	1	0	0	1	0	0	6
0	0	1	0	1	0	0	0	0	1	7
0	0	0	0	0	1	0	0	1	0	8
0	0	0	0	1	0	0	1	0	1	9
0	0	0	0	0	0	1	0	1	0	10

$A^3 =$

1	2	3	4	5	6	7	8	9	10	
2	3	7	1	8	1	2	2	1	3	1
3	0	1	1	1	1	2	0	1	0	2
7	1	4	4	8	2	7	2	3	2	3
1	1	4	0	2	1	1	0	1	1	4
8	1	8	2	4	7	9	0	9	1	5
1	1	2	1	7	0	1	4	0	3	6
2	2	7	1	9	1	2	3	1	5	7
2	0	2	0	0	4	3	0	5	0	8
1	1	3	1	9	0	1	5	0	5	9
3	0	2	1	1	3	5	0	5	0	10



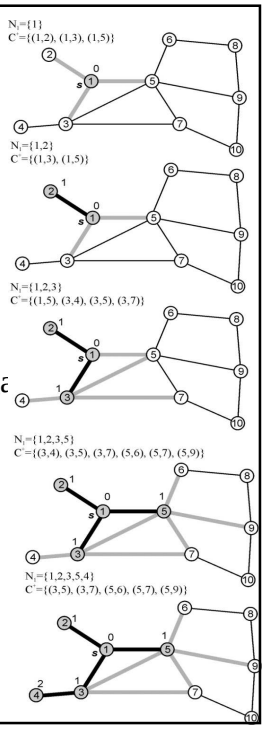


2. ALGORITMI ZA PRETRAŽIVANJE

Pretraživanje po širini (Breadth First Search –BFS)

Bolja implementacija BFS algoritma (omogućava generalizaciju algoritama za pretraživanje grafa):

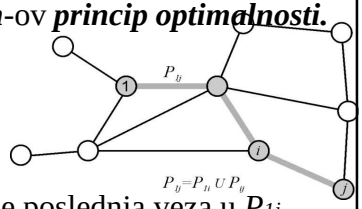
- Uvodi se skup (lista) veza C^+ koje su kandidati za propagaciju
- Inicijalno skup C^+ je prazan i u prvom prolazu se u njega prebacuju veze koje spajaju čvor *s* čiji je rang *r*=0 i neoznačene čvorove iz N_2.
- U sledećem koraku se bira prva veza iz liste C^+ - (**F**irst **I**n **F**irst **O**ut) za koju se utvrdi da spaja označen i neoznačen čvor u mreži
- Uzimanjem uvek prvih veza iz skupa C^+ - (FIFO), obezebeđuje se da propagacija uvek bude od čvorova najmanjeg ranga.



2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Pronalaženje najkraćeg puta (Dijkstra algoritma)

- Problem vezan za ***težinske grafove*** je pronalaženje najkraćeg puta između dva čvora: *s* i *t*.
- Prethodno opisan Moore-ov algoritam bi odgovarao slučaju kada su težine svih veza (*i*,*j*) jednake 1: w_{ij}=1.
- Kada veze imaju svoje težine, problem se može rešiti algoritmom koji je formulisao ***Dijkstra*** i koji se oslanja na ***Bellman***-ov ***princip optimalnosti.***



Bellman-ov princip optimalnosti

Ako je P_{1j}: 1-*j* najkraći put od 1 do *j* u grafu, a (*i*,*j*) je poslednja veza u P_{1j}, onda je putanja do čvora *i*, P_i:1-*i* (koja se dobija izostavljenjem veze (*i*,*j*)) najkraći put od od 1 do *i*.

2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

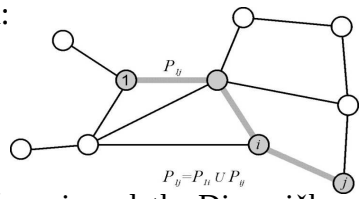
Pronalaženje najkraćeg puta (Dijkstra algoritma)

Bellman-ov princip optimalnosti

Dokaz: ako postoji put P_{1i}* od 1 do *i* koji je kraći od P_{1i} onda bi dodavanje težine veze *(i,j)*: w_{ij} tom putu dovelo do toga da je taj drugi put kraći od P_{1j}, što je suprotno pretpostavci.

Matematička formulacija ovog principa je sledeća:

$$W_1 = 0$$
$$W_j = \min_i (W_i + w_{ij})$$



Bellman-ov princip optimalnosti se koristi i za rešavanje zadatka Dinamičkog programiranja.

2. ALGORITMI ZA PRETRAŽ.

Pronalaženje najkraćeg puta (Dijkstra alg.)

1 Čvor 1 se označi kao nulti i inicijalizuje se težina čvora W_1=0. Svi ostali čvorovi iz grafa dobijaju težinski faktor W_k=∞, k=2, ... ,NC

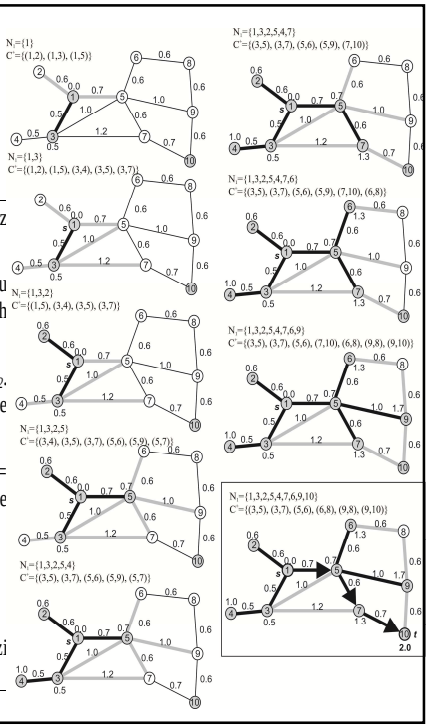
2 Od skupa čvorova, formiraju se dva komplementarna skupa N_1 i N_2. U prvom koraku u skupu N_1 nalazi se samo označeni čvor 1, dok je skup N_2 sastavljen od svih preostalih čvorova u mreži.

3 Traže se grane koje povezuju čvorove iz N_1 sa neoznačenim čvorovima iz N_2. Identifikuju se sve veze g_k (k=1,2,...) koje ispunjavaju taj uslov. Ako ne postoje grane koje ispunjavaju ovaj uslov, algoritam se prekida

4 Bira se veza g_k=(i,j) (i iz N_1 i j iz N_2) za koju je ispunjen uslov minimalnosti: W_j = min_k (W_i+w_k) gde je w_k težina veze g_k. Izabrani čvor j se briše iz skupa N_2 i prebacuje u N_1. Za čvor j se pamti i podatak da je prethodni čvor čvor i.

5 Proverava se da li je čvor j istovremeno i terminalni čvor t.

- Ako još nije dostignut terminalni čvor i ide se ponovo na korak 2.
- Ako jeste, putanja od *1* do t se nalazi tako što se kreće od krajnjeg čvora t i nalazi se prethodni čvor. Pretraživanje se nastavlja sve do čvora 1.



2. ALGORITMI ZA PRETRAŽIVANJE GRAF

Pretraživanje po dubini (**D**epth **F**irst **S**earch – **DFS)**

Rekurzivno pretraživanje po dubini **DFS** zajedno sa **BFS** je jedan od najčešće korišćenih algoritama.

DFS algoritam

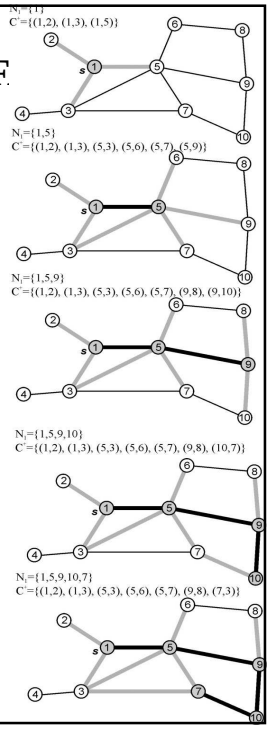
1 Čvor s je početni ili se iz skupa čvorova N proizvoljno izabere čvor sa kojim se kreće u pretraživanje.

2 Od skupa čvorova u grafu N, formiraju se dva komplementarna skupa N_1 i N_2. U prvom koraku u skupu N_1 nalazi se samo označeni čvor s, dok je skup N_2 sastavljen od svih preostalih čvorova u mreži. Formira se skup (lista) veza C^+ u koji se prebacuju sve veze koje imaju čvor s.

3 Iz skupa C^+ bira se poslednja veza (LIFO) g=(i,j), koja zadovoljava uslov da povezuje čvor iz N_1 (obeleženi – posećeni) i čvor iz N_2. Ako ne postoji veza koja ispunjava ovaj uslov, algoritam se prekida.

4 Izabrana veza g, ima jedan čvor u N_1 (npr. čvor i) i drugi čvor u N_2 (npr. čvor j). Čvor iz N_2 se obeležava i prebacuju u N_1 čime je komplementarnost ovih skupova očuvana. Iz C^+, koji predstavlja skup veza koje su kandidati za dalju pretragu, briše se g=(i,j).

5 Pronalaze se sve veze koje spajaju prethodno obeleženi čvor j i neobeležene čvorove iz N_2. Veze se dodaju na kraj skupa C^+. Povratak na korak 3.



2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

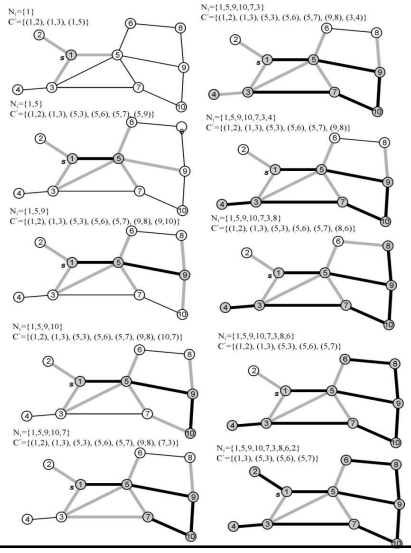
Pretraživanje po dubini (Depth First Search – DFS)

Primer:

10 čvorova

14 veza

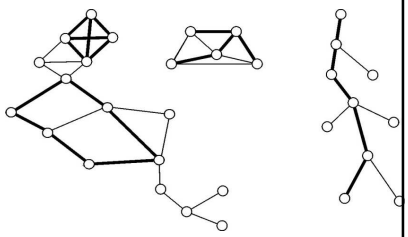
Početni čvor za propagaciju: **1**



2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Primena algoritama za pretraživanje

Primene algoritama za pretraživanje su brojne i veoma često deo drugih algoritama sa kojima se obavljaju specifični zadaci nad grafom:



–***Test povezanosti*** grafa

–Pronalaženje ***podgrafova***

–Pronalaženje ***proste putanje***

–Identifikacija ***prstenova*** (cikličnih putanja)

–Pronalaženje ***mostova*** u grafu (most je veza čijim se izostavljanjem iz povezanog grafa isti deli na dva povezana podgrafa)

– Pronalaženje ***razdelnih čvorova***

–***Topološko sortiranje*** grafa

2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Primena algoritama za pretraživanje

Elementarna analiza rezultata algoritama za pretragu grafa

Povezanost grafa - ako je nakon pretrage brojnost skupa N_1 jednaka NC, to znači da su u svi čvorovi obeleženi – posećeni i da je graf povezan.

Ispitivanje da li je graf drvo (granata mreža) - ako je graf povezan i brojnost skupa čvorova N_1 je za jedan manja od brojnosti skupa veza NV, onda je graf drvo: NV=NC-1.

Razapinjuće stablo (Spanning Tree – ST) - rezultat DFS (kao i BFS) algoritma kod povezanih grafova je maksimalni podgraf koji je razapinjuće stablo. To znači da je broj označenih veza u ST za jedan manji od broja čvorova.

2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Primena algoritama za pretraživanje

Elementarna analiza rezultata algoritama za pretragu grafa

Broj prstenova u mreži – ako je nakon okončanja DFS (ili BFS) algoritma, mreža povezana, onda sve veze koje nisu iskorišćene za propagaciju i nisu deo podgrafa ST čine prstenove.

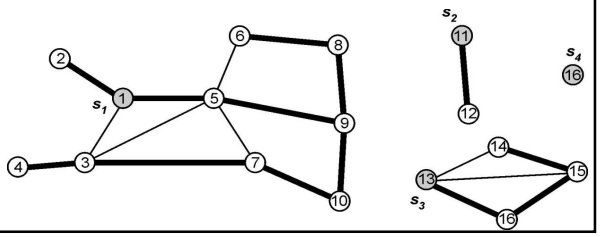
Kod povezanog grafa broj prstenova NP se može sračunati i kao broj veza umanjen za minimalni broj veza sa kojima je graf povezan: NP=NV-(NC-1).

2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Primena algoritama za pretraživanje

Elementarna analiza rezultata algoritama za pretragu grafa

Broj povezanih podgrafova u nepovezanom grafu - ako se kao rezultat algoritma dobije da graf nije povezan, maksimalni podgrafovi se dobijaju jednostavnim ponovnim pozivanjem DFS algoritma, pri čemu se kao čvor od koga se nastavlja pretraga bira čvor koji nije iz N_1. Znači pretraga se nastavlja sa nekim od prethodno nepovezanih čvorova iz N (u DFS algoritmu ovaj skup je označen sa N_2). Postupak se nastavlja sve dok skup preostalih – nepovezanih čvorova ne bude prazan.



2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Primena algoritama za pretraživanje

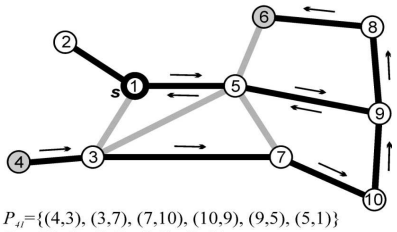
Pronalaženje prostog puta u povezanom grafu

Najjednostavniji način za pronalaženje prostog puta između bilo koja dva čvora (*i* i *j*) u povezanom grafu je da se, prilikom primene algoritma za pretragu, za svaki čvor u uređenom skupu N_1, pamti i čvor iz koga se došlo prilikom propagacije.

Brojnost tog skupa N_p će biti za jedan manja od skupa N_1. jer čvor *s*, od koga se startovalo nema prethodni čvor.

N_1={1,5,9,10,7,3,4,8,6,2}

N_p={1,5,9,10,7,3,9,8,1}



P_{41}={(4,3), (3,7), (7,10), (10,9), (9,5), (5,1)}
P_{16}={(1,5), (5,9), (9,8), (8,6)}

P_{46} = {(4,3), (3,7), (7,10), (10,9), (9,8), (8,6)}

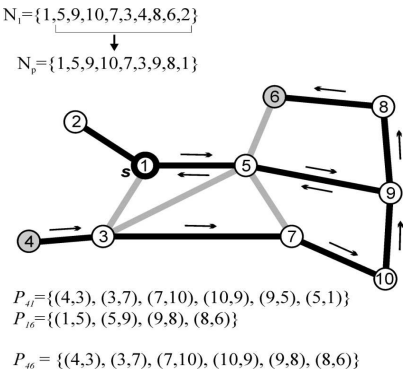
2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Primena algoritama za pretraživanje

Pronalaženje prostog puta u povezanom grafu

Pronalaženje putanje od nekog čvora *j* do *s: p*ronalazi se čvor *j* u N_1 i rekurzivno se ide u njemu prethodni čvor sve dok se ne stigne do s.

Putanja od *j* do *s* se može pamtiti kao uređeni skup čvorova ili kao uređeni skup veza $P_{js}=\{(j,k),...,(l,s)\}$.
Putanja od od *s* do *j* u prostom grafu je isti taj skup samo je redosled veza obrnut $P_{sj}=\{(s,l),...,(k,j)\}$.
Putanja između dva proizvoljna čvora *(i,j)*, dobija se: $P_{ij} = (P_{is} \cup P_{sj}) / (P_{is} \cap P_{sj})$



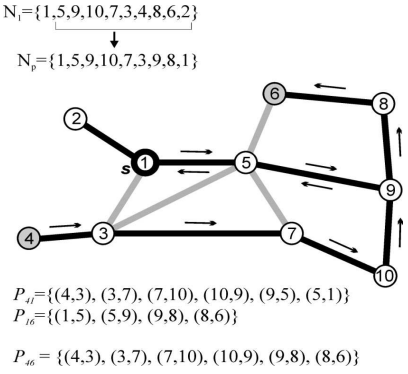
2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Primena algoritama za pretraživanje

Pronalaženje prostog puta u povezanom grafu

Sa prethodnog načina predstavljanja puta, kao uređenog skupa veza, jednostavno je preći na prikaz puta kao uređenog skupa čvorova.

Potrebno je uzeti početni čvor *i*, i po redu dodati svaki drugi čvor veze iz uređenog skupa veza P_{ij}.



2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

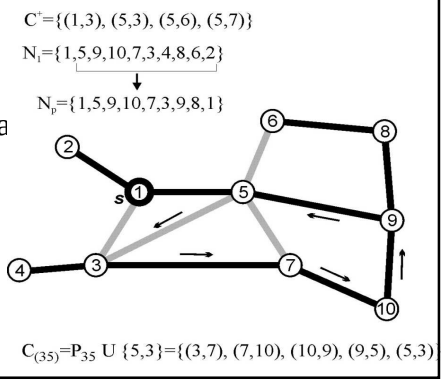
Primena algoritama za pretraživanje

Pronalaženje prstenova u grafu

Broj prstenova u povezanoj mreži odgovara brojnosti skupa C^+ nakon primene DFS ili BFS algoritma.

Za identifikaciju prstena (ciklične putanje) koji čine veza (*i*,*j*) koje nisu iskorišćene za propagaciju, sa vezama iz razapinjućeg stabla naći prost put od čvora *j* do čvora *i* i pridružiti mu vezu (*i*,*j*)

$$C_{ij} = P_{ji} \cup \{(i, j)\}$$

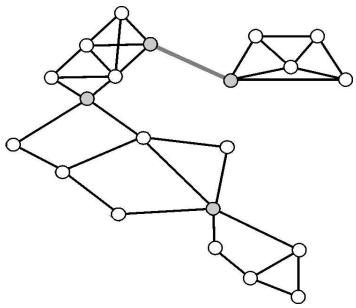


2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Primena algoritama za pretraživanje

Pronalaženje mostova u grafu

Prema definiciji, most u povezanom grafu je veza čijim se izostavljanjem isti deli na dva povezana podgrafa.
Naivna implementacija algoritma za ispitivanje, da li je veza (*i*,*j*) most, je da se iz grafa izbriše ova veza, i da se onda primeni DFS ili BFS algoritam sa čvorom *i* (ili *j*) kao startnim. Ako se propagacijom stigne do čvora *j* (ili *i*) znači da veza (*i*,*j*) nije most.

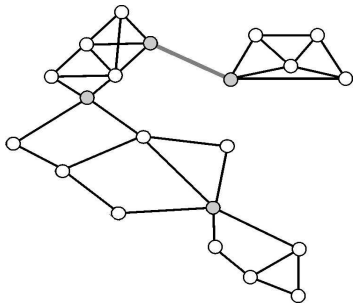


2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Primena algoritama za pretraživanje

Pronalaženje mostova u grafu

Bolja implementacija bi podrazumevala korišćenje rezultata DFS ili BFS algoritma.
Svojstvo grafa koji je drvo je da je svaka veza most.
Ako je u nekom grafu veza most, onda ta veza mora biti deo razapinjućeg stabla.
Veza koja je most ne može biti deo prstena.



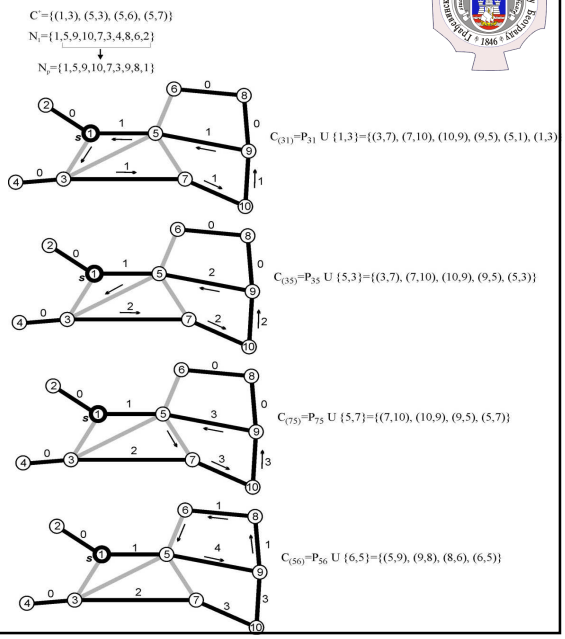
Najjednostavniji način da se identifikuju sve veze koje predstavljaju most, je da se definišu i prebroje prstenovi za svaku vezu.

2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Primena algoritama za pretraživanje

Pronalaženje mostova u grafu

Postupak: nakon primene DFS ili BFS, algoritma, potrebno za sve koje nisu iz ST, naći prstenove. Kada se označe veze iz ST koje pripadaju prstenovima, preostale veze iz ST koje ne pripadaju prstenovima su mostovi.

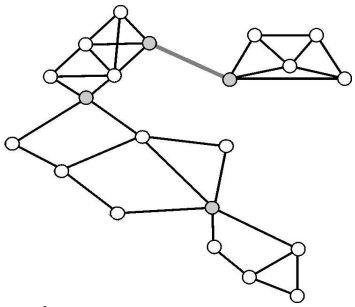


2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Primena algoritama za pretraživanje

Pronalaženje razdelnih čvorova u grafu

• Čvor u povezanom grafu G je razdelni ako se uklanjanjem tog čvora i susednih veza, graf deli na najmanje dva maksimalna podgrafa
• Ako je čvor razdelni, onda svaki prost put koji spaja čvorove iz tih podgrafova mora prolaziti kroz taj čvor.



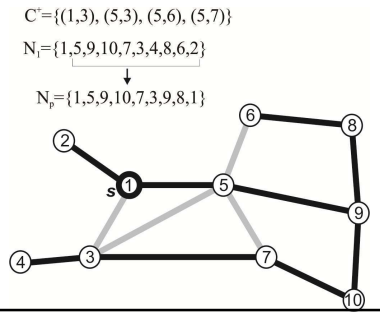
Naivna implementacija algoritma za proveru da li je neki čvor u povezanom grafu razdelni:
• ukloni se taj čvor i njemu pripadajuće veze
• primeni neki od algoritama za pretraživanje
• ako je graf i nakon uklanjanja ovog čvora znači da taj čvor nije razdelni.
Bolja implenetaciju podrazumeva primenu DFS algoritma.

2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Primena algoritama za pretraživanje

Pronalaženje razdelnih čvorova u grafu

Provera da li je neki čvor razdelni može se obaviti jednostavnom interpretacijom rezultata **DFS** algoritma za pretraživanje.
Da bi se dokazalo da čvor *v* nije razdelni, potrebno je samo pokazati da postoji veza iz C^+ koja formira prsten i spaja čvorove koji se u uređenom skupu čvorova N_1 nalaze ispred i iza čvora *v* (npr. čvor 5 nije razdelni jer postoji veza (1,3) u C^+ koja spaja čvorove 1 i 3 koji su u skupu N_1 ispred i iza čvora 5).
Kada ne bi postojala veza (1,3), očigledno je da bi čvor 5 bio razdelni. Čvorovi 1 i 3 su razdelni.
Opisani postupak se jedino ne može primeniti na čvor od koga se počelo sa propagacijom razapinjućeg stabla (čvor 1).
Za proveru da li je "izvorni" čvor 1 razdelni, potrebno je proveriti da li se u skupu *Np* čvor 1, osim na početku skupa, pojavljuje i kasnije.

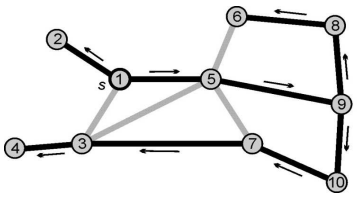


2. ALGORITMI ZA PRETRAŽIVANJE GRAFA

Primena algoritama za pretraživanje

Topološko sortiranje grafa

Rezultat primene algoritma za pretraživanje grafa (BFS ili DFS) je razapinjuće stablo ST.
Ako veze u ST orijentišemo u pravcu propagacije, onda se podgraf ST može predstaviti kao aciklični digraf (Directed Acyclic Graph – DAG).
Topološko sortiranje veza iz ST-a ima brojne praktične primene**.**



C^+={(1,3), (5,3), (5,6), (5,7)}

N_l={1,5,9,10,7,3,4,8,6,2}

N_p={1,5,9,10,7,3,9,8,1}

ST={(1,5), (5,9), (9,10), (10,7), (7,3), (3,4), (9,8), (8,6), (1,2)}

3. MINIMALNO RAZAPINJUĆE STABLO

Nad povezanim prostim grafom G, može se formirati jedno ili više razapinjućih stabala (ST).

Broj mogućih razapinjućih stabala je *Nst*, i u prethodnom delu je objašnjeno kako se taj broj računa.

Kod težinskog grafa, čest je zadatak da se među svim razapinjućim stablima *STi*, nađe ono čiji je ukupni zbir težina veza najmanji.

$$W_{ST_i} = \sum_{(i,j)\in ST_i} w_{ij}$$

Stablo koje zadovoljava taj uslov naziva se minimalno razapinjuće stablo (Minimum Spanning Tree – MST).

$$W_{MST} = \min_i W_{ST_i}$$

3. MINIMALNO RAZAPINJUĆE STABLO

Kruskal-ov algoritam

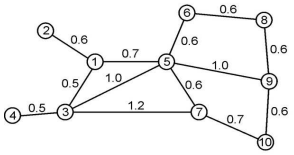
1. Prvi korak u primeni ovog algoritma je sortiranje veza prema težini.
2. Iz skupa sortiranih veza bira se ona koja ima najmanju težinu i dodaje se u skup koji će činiti MST. Ne dodaju se samo one veze koje čine prsten sa već izabranim vezama.

Najjednostavniji način za sprovođenje koraka 2 je da se inicijalno svi čvorovi označe rangom 0 ($r_i=0$). Sa dodavanjem nove veze (i,j) proverevaju se sledeće varijante:

- čvorovi su neoznačeni – oba čvora se označavaju rangom koji je za jedan veći od prethodno najvećeg, jer se dodavanje nove veze praktično formira novi podgraf.
- čvorovi su različitog ranga ($ri \neq rj$) – dodaje se veza (i,j) i ako je na primer čvor j većeg ranga, onda se čvor j i svi čvorovi tog ranga ozačavaju sa ri, jer postaju deo istog podgrafa.
- čvorovi su istog ranga – veza (i,j) se ne dodaje jer bi se time formirao prsten.

Kada je broj dodatih veza za jedan manji od broja čvorova (NC-1), prekida se algoritam.

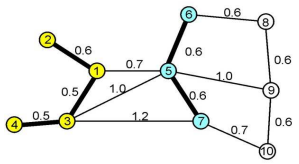
3. MINIMALNO RAZAPINJUĆE STABLO



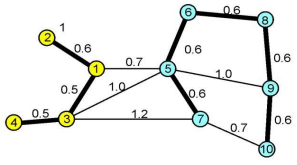
C^*	w_{ij}
(1,3)	0.5
(3,4)	0.5
(1,2)	0.6
(5,6)	0.6
(5,7)	0.6
(6,8)	0.6
(8,9)	0.6
(9,10)	0.6
(7,10)	0.7
(1,5)	0.7
(3,5)	1.0
(5,9)	1.0
(3,7)	1.2

Kruskal-ov algoritam

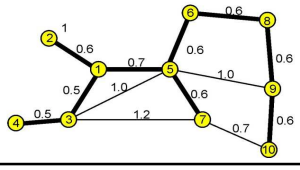
Svojstvo opisanog algoritma je da se tek na kraju dobija podgraf koji je minimalno razapinjuće stablo.



MST	1	2	3	4	5
(1,3)					
	(3,4)				
		(1,2)			
			(5,6)		
				(5,7)	



MST	6	7	8
(6,8)			
	(8,9)		
		(9,10)	



MST	9
(1,5)	

3. MINIMALNO RAZAPINJUĆE STABLO

Prim-ov algoritam

Prim-ov algoritam se jednostavno može opisati kroz već prikazanu generalizaciju algoritama za pretraživanje grafa:

Generalizacija BFS i DFS algoritama se zasniva na formiranju skupa veza C^+, u koji se u svakom koraku algoritma dodaju nove veze koje povezuju označene čvorove iz skupa N_1, i čvorove koji još nisu označeni.

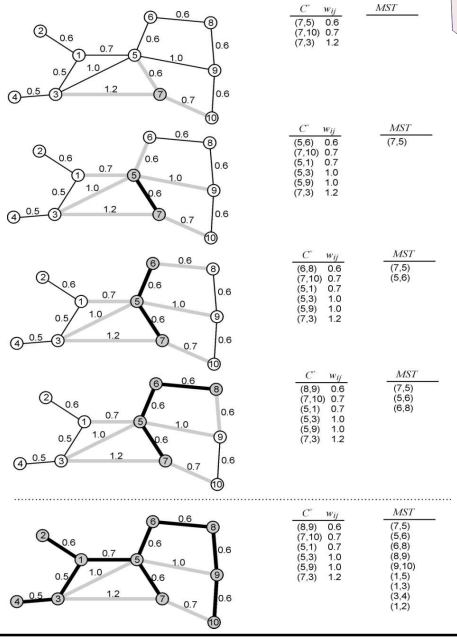
Kod DFS algoritma pretraživanje grafa, odnosno propagacija razapinjućeg stabla (ST), se nastavlja u pravcu one veze iz skupa C^+ koja je poslednja dodata (LIFO), dok je kod BFS algoritma nastavak pretraživanja u pravcu prve veze iz C^+ (FIFO).

Ako bi se veze u skupu C^+ čuvale sortirane po težini, i u svakom koraku algoritma nastavljala propagacija biranjem veze sa najmanjom težinom, onda se kao rezultat primene ovog algoritma dobija minimalno razapinjuće stablo (MST). Opisani algoritam se naziva Prim-ov. U opštem slučaju se ovaj postupak propagacije koji je vezan za neki kriterijum (u ovom slučaju kriterijum minimalnosti) naziva PFS algoritam (**P**riority **F**irst **S**earch).

3. MINIMALNO RAZAPINJUĆE STABLO

Prim-ov algoritam

Rezultat svakog koraka Prim-ovog algoritma je podgraf koji ima strukturu drveta.

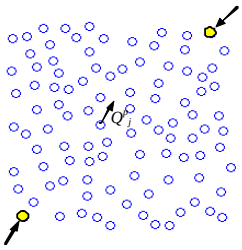


3. MINIMALNO RAZAPINJUĆE STABLO

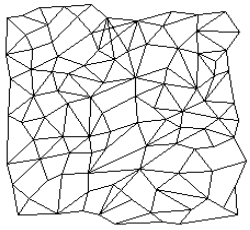
Primer:

Projektovanje nove distributivne mreže pod pritiskom.

Faza 1. Identifikacija izvornih i potroačkih čvorova u mreži



Faza 2. Povezivanje čvorova svim mogućim alternativnim trasama i kreiranje osnovnog grafa



- izvorni čvor

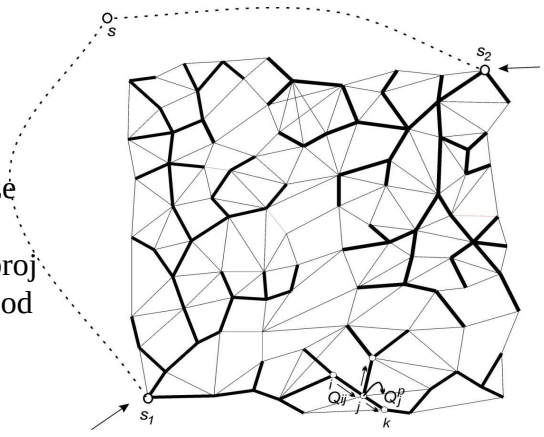
- čvor sa definisanom potrošnjom Q^p_j

3. MINIMALNO RAZAPINJUĆE STABLO

Primer:

Faza 3: Primena *Prim*-ovog algoritma za pronalaženje MST-a.

- Uvodi se fiktivni čvor s sa fiktivnim vezama dužine 0.
- Rezultat algoritma (PFS) je podgraf MST sa topološki uređenim skupom veza (veze su poredjane od uzvodnih prema nizvodnim). Ako je broj veza u MST za jedan manji od broja čvorova – mreža je povezana.



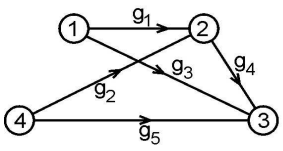
Faza 4: Proračun protoka i dimenzionisanje mreže

4. ORIJENTISANI GRAF - DIGRAF

Algoritmi za pretraživanje digrafa

- Digraf se predstavlja u formi matrica ili lista povezanosti
- Svaki čvor ima stepen ulaza i izlaza koji su broj veza koje ulaze u čvor i broj veza koje izlaze iz čvora
- Čvor čiji je ulazni stepen 0 je izvorni a čvor čiji je izlazni stepen 0 je ponorni

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$



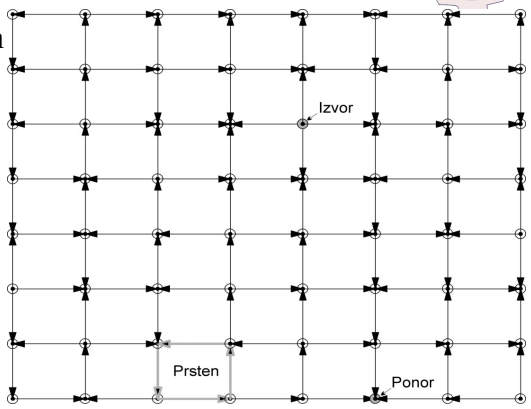
Čvor	Veze iz čvora	Veze u čvor
1	g_1 g_3	
2	g_4	g_1 g_2
3		g_3 g_4 g_5
4	g_2 g_5	

Veze	Čvor *start*	Čvor *end*
g_1	1	2
g_2	4	2
g_3	1	3
g_4	2	3
g_5	4	3

4. ORIJENTISANI GRAF - DIGRAF

Algoritmi za pretraživanje digrafa

- Digraf je kompleksnija struktura od neorijentisanog grafa
- Pitanje povezanosti digrafa nije jasno definisano
- Uvodi se pojam ***dostupnosti*** čvora *t* iz čvora *s.* Čvor *t* je dostupan iz čvora *s* ako postoji orijentisana putanja od *s* do *t.*

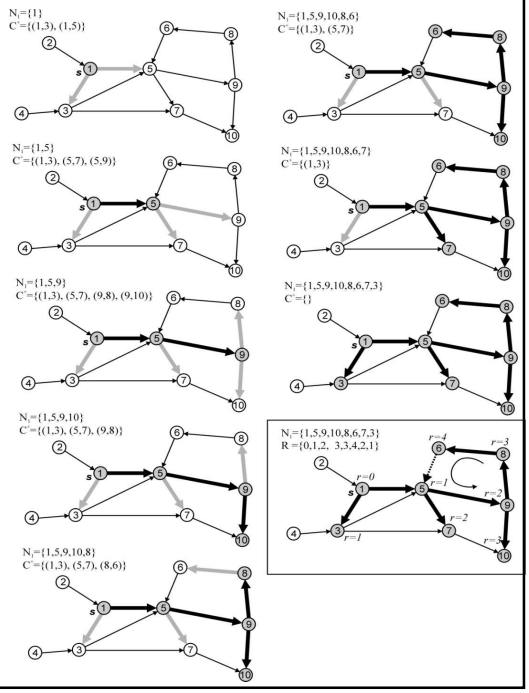


- Digraf je ***strogo povezan*** ako je svaki čvor digrafa dostupan iz svakog čvora
- Dva čvora su strogo povezana ako su deo prstena

4. ORIJENTISANI GRAF

Algoritmi za pretraživanje - DFS

- Generalizacija DFS algoritma za neorijentisane grafove
- Rezultat je orijentisani podgraf koji je stablo. Rezultat zavisi od izbora startnog čvora *s*.
- Označeni čvorovi su dostupni iz čvora *s*.
- Identifikuju se povratne veza koje čine orijentisane prstenove, kao i poprečne (redundatne) veze.
- Digraf koji nema povratne veze je acikličan digraf (DAG).



4. ORIJENTISANI GRAF - DIGRAF

Algoritam za topološko sortiranje DAG-a

Topološko sortiranje DAG-a

1 Pronalaze se svi izvorni čvorovi iz N. Svi izvorni čvorovi se dodaju u skup označenih čvorova N_1. Skup N_2 je njemu komplementaran skup neoznačenih čvorova u mreži.

2 Formira se skup (lista) veza C^+ u koji se prebacuju sve veze koje imaju početni čvor u N_1.

3 Iz skupa C^+ bira se poslednja veza (LIFO) $g=(i,j)$, koja zadovoljava uslov da povezuje čvor iz N_1 (obeleženi – posećeni) i čvor j iz N_2, uz uslov da su i svi ostali čvorovi koji ulaze u j, takođe iz N_1 (prethodno označeni). Ako ne postoji veza koja ispunjava ovaj uslov, algoritam se prekida.

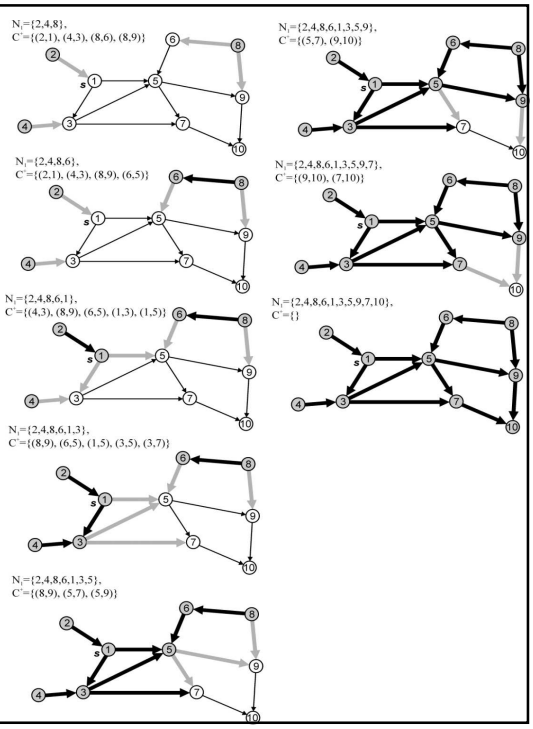
4 Izabrana veza g, ima jedan čvor u N_1 (npr. čvor i) i drugi čvor u N_2 (npr. čvor j). Čvor iz N_2 se obeležava i prebacuju u N_1 čime je komplementarnost ovih skupova očuvana. Iz C^+, koji predstavlja skup veza koje su kandidati za dalju pretragu, briše se $g=(i,j)$ i sve ostale veze koje završavaju u čvoru j.

5 Pronalaze se sve veze koje spajaju prethodno obeleženi čvor j i neobeležene čvorove iz N_2. Veze se dodaju na kraj skupa C^+. Povratak na korak 3.

4. ORIJENTISANI GRAF - DIGRAF

Topološko sortiranje DAG-a:

- Ako je nakon primene algoritma, brojnost skupa N_1, jednaka brojnosti skupa N, procesirani graf je aciklični digraf - **DAG**



4. ORIJENTISANI GRAF - DIGRAF

Topološko sortiranje DAG-a – primer primene

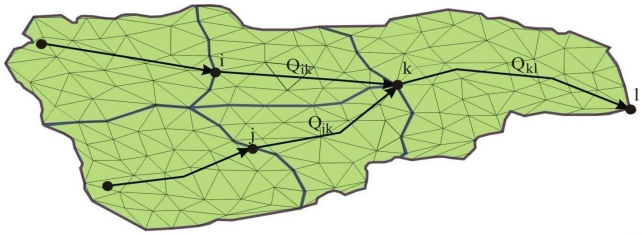
Algoritam za automatsko “dimenzionisanje” kolektora za atmosfersku i upotrebljenu vodu:

1. Postavljanje mreže u 2D
2. Odredjivanje merodavnih protoka: Delineacija + Pojednostavljen hidraulicki proračun (model kinematskog talasa)
   1. Pad kolektora jednak padu terena
   2. Dimenzije kolektora proizvoljne
3. Strategija minimalna dubina ukopavanja
   a) Proračun počinje od najuzvodnijih čvorova (šahtova) – formira se uredjeni skup kolektora (od uzvodnih prema nizvodnim)
   b) Definišu se hidraulička i geometrijska ograničenja: Maksimum uzvodnih precnika cevi je minimalan nizvodni prečnik, iz uslova maksimalne i minimalne brzine i maksimalne i minimalne dubine ukopavanja određuje se opseg raspoloživih prečnika i padova
   c) Definiše se skup mogućih rešenja u formi parova podataka: prečnik kolektora i pad kolektora

4. ORIJENTISANI GRAF - DIGRAF

Topološko sortiranje DAG-a – primer primene

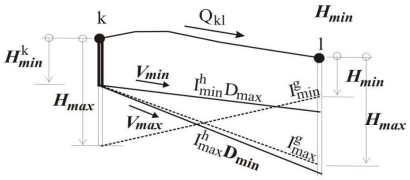
Algoritam za automatsko “dimenzionisanje” kolektora za atmosfersku i upotrebljenu vodu



$$D_{min} = \mathbf{Max}(D_{jk}, D_{ik})$$
$$H^{k}_{min} = \mathbf{Max}(H_{jk}, H_{ik})$$

$$I_{min} = \mathrm{Max}(I^{h}_{min}, I^{g}_{min})$$
$$I_{max} = \mathrm{Min}(I^{h}_{max}, I^{g}_{max})$$

$$\underset{\{I\}}{min}\ \{(D_{kl}, I_{kl})\}$$



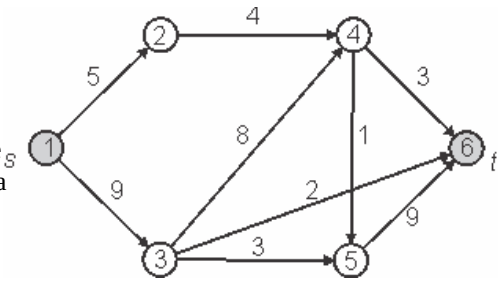
5. MAKSIMALAN PROTOK KROZ MREŽU

Mreža i protok kroz mrežu

Pod mrežom ćemo smatrati orijentisani graf (digraf) sa vezama koje imaju definisan kapacitet $Q=\{q_{ij}\}$ i jedan izvorni (s) i jedan ponorni čvor (t): $G=\{N,C,Q\}$.

Usvaja se konvencija da je čvor 1 izvorni (s) a NC ponorni (t).

Zadatak je pronaći maksimalan protok kroz mrežu, takav da protok kroz pojedinačne veze (x_{ij}) bude manji od kapaciteta veze (q_{ij}) ali i da materijalni bilans bude zadovoljen u svakom čvoru, što znači da je dotok u čvor jednak isticanju iz čvora. Ovo važi za svaki čvor, osim za izvorni i ponorni.



5. MAKSIMALAN PROTOK KROZ MREŽU

Mreža i protok kroz mrežu

Opisana ograničenja se mogu formulisati na sledeći način:

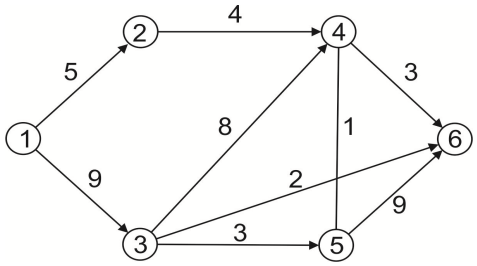
$$x_{ij} \le q_{ij}$$
$$x_{ij} \ge 0$$
$$\sum_k x_{ki} - \sum_k x_{ik} = 0, \quad i = 2,..., NC-1$$

Kako je u mreži zadovoljen materijalni bilans, protok kroz mrežu f će odgovarati sumi protoka koji izlazi iz izvornog čvora, ili sumi protoka koji dotiče u ponorni čvor:

$$f = \sum_k x_{1k} = \sum_k x_{k\,NC}$$

Kako su i kriterijumska i funkcije ograničenja linearne, očigledno je da se ovaj problem može formulisati kao problem LP-a:

$$\max_x f = \sum_k x_{1k}$$



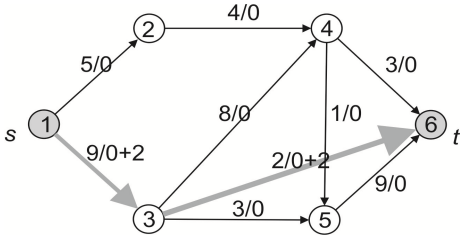
5. MAKSIMALAN PROTOK KROZ MREŽU

Ford – Fulkerson-ov algoritam

Bolji način za rešavanje problema maksimizacije protoka kroz mrežu je primena Ford-Fulkerson-ov algoritma.

Inicijalno se protoci kroz sve veze izjednačavaju sa nulom (x_{ij}=0).

Primenom BFS algoritma nalazi se put kojim su izvorni i ponorni čvor mogu spojiti sa najmanjim brojem veza.



Na izabranom prostom putu P: 1,3,6, nalaze se veze čiji su kapaciteti 9 i 2.
Kapacitet izabranog puta odgovara kapacitetu najslabije veze: q_{36}=2, pa se protok duž ovog puta povećava sa nule na 2.
Nakon ove promene protoka, veza 3 do 6 nema više kapaciteta da propusti dodatan protok, a veza od 1 do 3 ima neiskorišćen kapacitet 9-2=7.

Napomena: materijalni bilans u čvorovima na ovaj način ostaje zadovoljen.

5. MAKSIMALAN PROTOK KROZ MREŽU

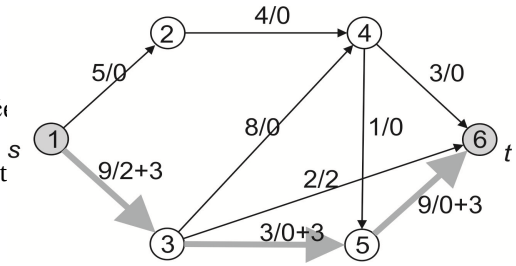
Ford – Fulkerson-ov algoritam

Uvodi se pojam rezidualnog kapaciteta r_{ij} koji se računa kao razlika maksimalnog kapaciteta i protoka: $r_{ij} = q_{ij} - x_{ij}$

Kandidati za propagaciju BFS algoritma su sve veze koje imaju rezidualni kapacitet veći od 0.

Ponovnom propagacijom BFS algoritma, izabran je prost put P: 1,3,5,6.

Minimalan rezidualni kapacitet veza duž izabranog puta je vrednost za koju je moguć uvećati protok.
Veza 3,5 ima minimalan rezidulani kapacitet r_{35}=3-0.



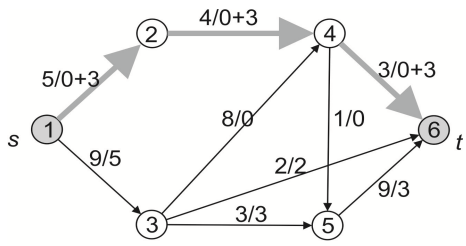
5. MAKSIMALAN PROTOK KROZ MREŽU

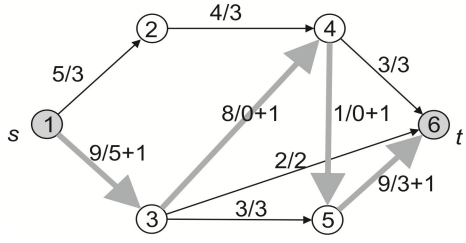
Ford – Fulkerson-ov algoritam

Naredni korci su prikazani na slici.

Ovim jednostvanim primerom nije obuhvaćen slučaj u kome je, u procesu pronalaženja maksimalnog protoka kroz mrežu, potrebno ostaviti mogućnost preraspodele protoka - smanjenja protoka u nekoj vezi zarad povećanja ukupnog protoka.

Maksimalno smanjenje protoka u nekoj vezi je jednako trenutnom protoku, jer protok ne može biti manji od 0.





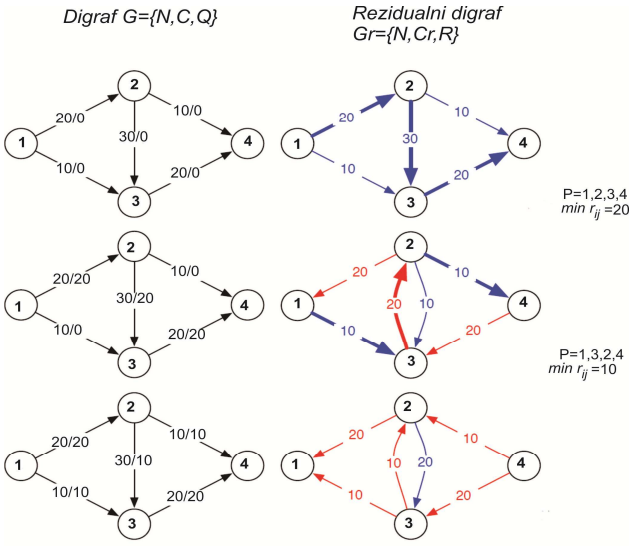
5. MAKSIMALAN PROTOK KROZ MREŽU

Ford – Fulkerson-ov algoritam

Da bi se omogućilo smanjenje protoka kroz neku vezu, za svaku vezu *ij* koja ima protok veći od nule ($x_{ij}>0$), uvodi suprotno orijentisana fiktivna veza čiji je rezidualni protok jednak protoku: $r_{ji}=x_{ij}$.

Na taj način se formira rezidualni graf, koji sadrži samo veze za koje je ispunjen uslov $r_{ij}>0$.

Nad rezidualnim grafom se u iteracijama pronalazi put od *s* do *t* algoritmom za pretraživanja grafa (BFS) i koriguje protok za minimalnu vrednost reziduala.



5. MAKSIMALAN PROTOK KROZ MREŽU

Ford – Fulkerson-ov algoritam

Ford-Fulkerson-ov algoritam za pronalaženje maksimalnog protoka

1. Definiše se digraf G={N,C,Q}, skupom čvorova *N*, čija je brojnost NC, veza *C* i kapaciteta veza $Q=\{q_{ij}\}$. Usvaja se konvencija da je 1 izvorni (*s*), a NC ponorni čvor (*t*).
2. Inicijalizuju se protoci u mreži $x^0_{ij}=0$.
3. Protok u mreži se izjednačava sa početnim $x_{ij}= x^0_{ij}$ i računaju se reziduli za povećanje ili smanjenje protoka kroz vezu *ij*: $r_{ij}=q_{ij}-x_{ij}$, $r_{ji}=x_{ij}$
4. Formira se fiktivni rezidualni graf G_r u kome su zastupljene stvarne i fiktivne veze koje ispunjavaju uslov $r_{ij}>0$.
5. Primenom BFS algoritma nad rezidualnim grafom G_r pronalazi se put od izvornog do ponornog čvora P_{st}, koji se može predstaviti preko niza veza *ij*. Ako ovakav put ne postoji, sračunat je maksimalan protok i algoritam se prekida.
6. Nalazi se minimalni rezidulani kapacitet veza koje čine put P_{st} i ova vrednost predstavlja maksimalnu korekciju protoka duž tog puta: $\Delta=\min\{r_{ij}\}$ za *ij* iz P_{st}.
7. Računa se korigovani protok duž puta P_{st}: $x_{ij}=x^0_{ij}+\Delta$ za *ij* iz P_{st}.
8. Povratak na korak 3

5. MAKSIMALAN PROTOK KROZ MREŽU

Ford – Fulkerson-ov algoritam

Generalizacija algoritma za slučaj više izvornih i više ponornih čvorova

Uvodi se jedan fiktivni izvorni čvor kojim se povezuju svi izvorni čvorovi fiktivnim vezama velikog kapaciteta.

Isto se uradi i za ponorne čvorove.

Fiktivnim vezama “beskonačnog” kapaciteta se garantuje da iste neće biti ograničenje da se ostvari maksimalan protok.