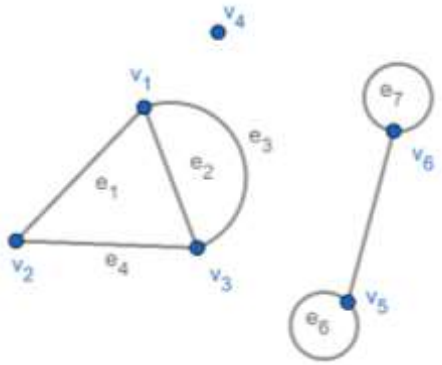
Grafovi

1. Posmatrajmo graf prikazan na slici sa desne strane.



a) Odrediti skup čvorova V i skup grana E posmatranog grafa. Za svaku granu posebno odrediti njene krajeve.

b) Odrediti sledeće skupove:
   - skup grana incidentnih sa v_1 (skup E_1),
   - skup čvorova susednih sa v_1 (skup N_{v_1}),
   - skup grana susednih sa e_1 (N_{e_1}),
   - skup svih petlji (skup L),
   - paralelnih grana (skup P),
   - skup čvorova susednih sami sebi (skup A)
   - skup izolovanih čovorova (skup I).

Rešenje:

a) $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$
$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$

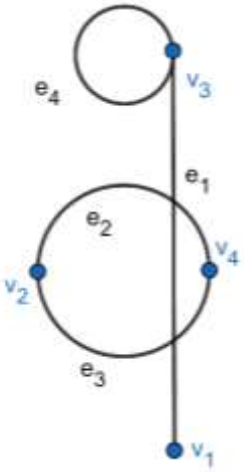
Grana	Krajevi grane
e_1	$\{v_1, v_2\}$
e_2	$\{v_1, v_3\}$
e_3	$\{v_1, v_3\}$
e_4	$\{v_2, v_3\}$
e_5	$\{v_5, v_6\}$
e_6	$\{v_5\}$
e_7	$\{v_6\}$

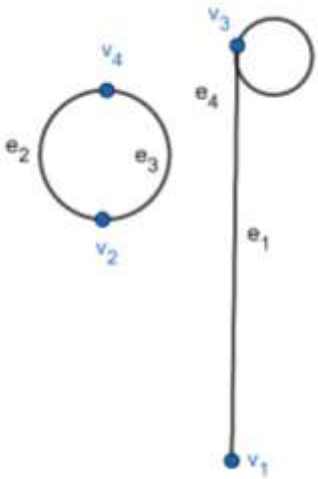
b) $E_1 = \{e_1, e_2, e_3\}$
$N_{v_1} = \{v_2, v_3\}$
$N_{e_1} = \{e_2, e_3, e_4\}$
$L = \{e_6, e_7\}$
$P = \{e_2, e_3\}$
$A = \{v_5, v_6\}$
$I = \{v_4\}$

2. Nacrtajte graf koji ima sledeću strukturu
$V = \{v_1, v_2, v_3, v_4\}$, $E = \{e_1, e_2, e_3, e_4\}$

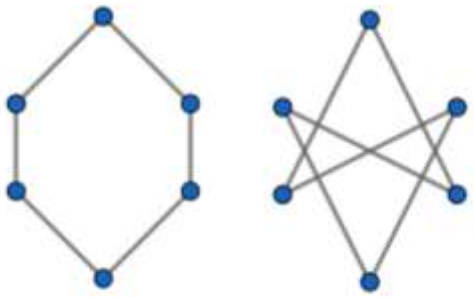
Grana	Krajevi grane
e_1	$\{v_1, v_3\}$
e_2	$\{v_2, v_4\}$
e_3	$\{v_2, v_4\}$
e_4	$\{v_3\}$

Rešenje:
Zadati graf se može prkazati na više načina, predlažemo dva

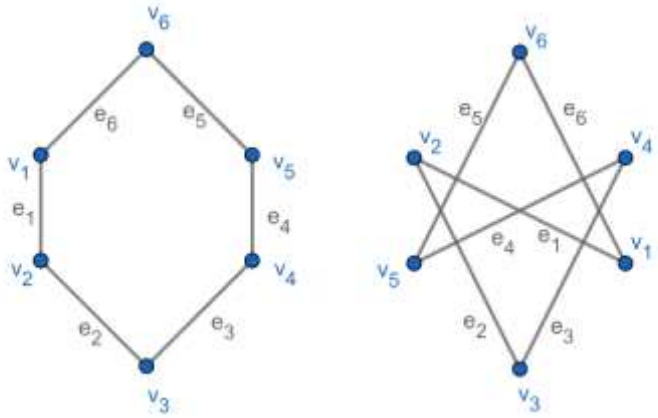




3. Obeležite grane i čvorove grafova tako da oni predstavljaju isti graf

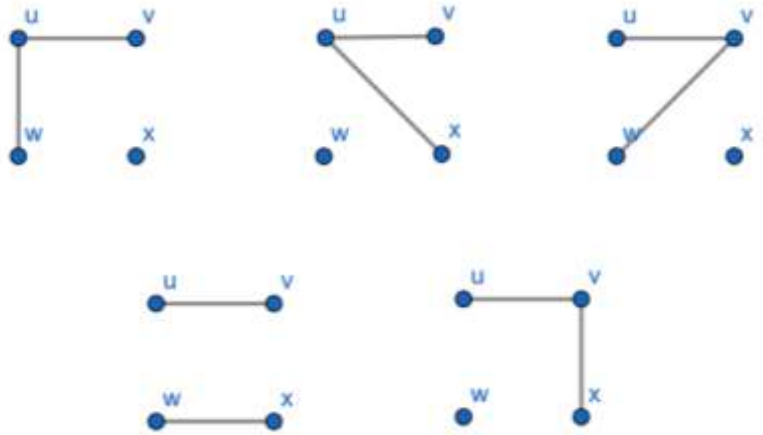


Rešenje:

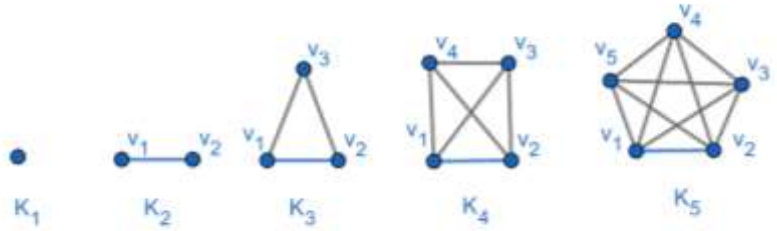


Grafovi imaju široku primenu danas. Pomoću grafova se mogu predstaviti komunikacioni sistemi, mreža puteva i slično. Grafovi se mogu klasifikovati po svojoj strukturi.

- Graf je **prost** ukoliko nema petlji niti paralelnih grana. Na primer, posmatrajmo graf sa skupom čvorova $V = \{u, v, w, x\}$ i skupom grana $E = \{e_1, e_2\}$ definisanih tako da je $e_1 = \{u, v\}$.

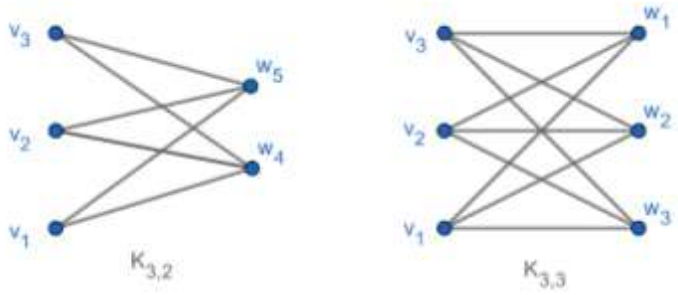


- Prost graf se naziva **kompletnim** ukoliko za svaki par čvorova postoji tačno jedna grana koja ih povezuje. Za kompletne grafove sa n čvorova koristimo oznaku K_n.

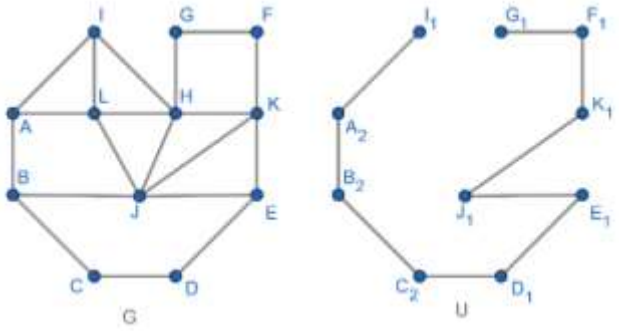


- Graf se naziva **kompletnim bipartitnim grafom** sa m i n čvorova, u oznaci $K_{m,n}$ ako se svi njegovi čvorovi mogu grupisati u dva skupa čvorova, označimo ih sa $v_1, v_2, \dots, v_m$ i $w_1, w_2, \dots, w_n$, tako da važi sledeće: za svako $i, k = 1, \dots, m$ i $j, l = 1, \dots, n$
  - postoji grana koja povezuje čvorove v_i i w_j
  - ne postoji grana između čvorova v_i i v_k
  - ne postoji grana između čvorova w_j i w_l

  Na slici niže su prikazana dva bipartitna grafa, $K_{3,2}$ i $K_{3,3}$

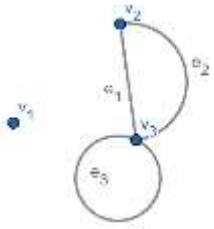


- Graf $U = (V_U, E_U)$ se naziva **podgrafom** grafa $G = (V_G, E_G)$ ukoliko je $V_U \subseteq V_G$ i ako su čvorovi V_U povezani tako da važi $E_U \subseteq E_G$.



- **Stepen čvora**, u oznaci $\deg v$, predstavlja broj grana koje iz tog čvora „izlaze“. Suma stepena svih čvorova predstavlja dvostruku vrednost broja grana tog grafa.

4. Odrediti stepen svakog čvora grafa koji je prikazan na slici



Rešenje
$\deg(v_1) = 0$ jer nije susedan ni sa jednim čvorom
$\deg(v_2) = 2$ jer su dve grane incidentne sa njim
$\deg(v_3) = 4$ jer su grane e_1, e_2 incidentne sa njim, kao i grana e_3 koja se računa dvostruko

$$\deg_G = \deg(v_1) + \deg(v_2) + \deg(v_3) = 0 + 2 + 4 = 6$$

Teorema
Stepen grafa jednak je dvostrukom broju njegovih grana.

Posledica
Stepen grafa je paran broj.

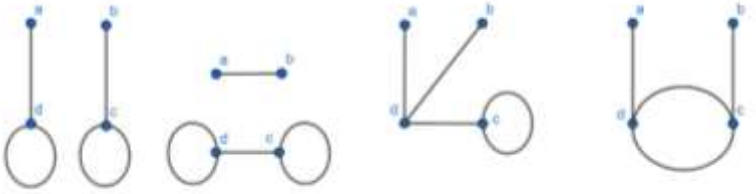
Posledica

U svakom grafu može biti samo paran broj čvorova neparnog stepena.

5. Nacrtati graf koji ima sledeću strukturu (ili pokazati da takav graf ne postoji)
   a. Graf sa 4 čvora stepena 1,1,2 i 3.
   b. Graf sa 4 čvora stepena 1, 1, 3 i 3
   c. Prost graf sa 4 čvora stepena 1,1, 3 i 3

Rešenje

a. Na osnovu posledice se može zaključiti da ovakav graf ne postoji (1+1+2+3=7, neparna vrednost)
b. Neki primeri grafa su prikazani niže



c. Ne postoji prost graf sa zadatom strukturom (na osnovu pretpostavki da je graf prost, možemo uzeti da su a i b čvorovi stepena 1, a c i d čvorovi stepena 3. Sa obzirom da je čvor prost, sledi da je nema paralelnih grana niti petlji, tako da čvor c mora biti susedan sa 3 čvora, a to su a, b i d. Takođe, čvor d mora biti susedan sa 3 čvora, a to su a, b i c, što je u kontradikciji sa pretpostvkom da su stepeni čvorova a i b jednaki 1.

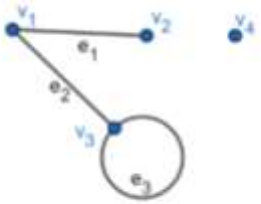
6. Da li se u grupi svako od devetoro ljudi može povezati sa tačno petoro drugih?

Rešenje

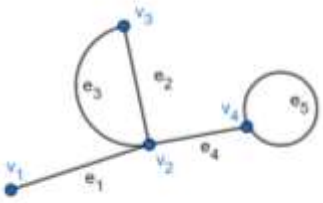
Ne. Predstavimo ljude kao čvorove grafa koji će međusobno biti povezani jedino ako su ljudi međusobno povezani. U tom slučaju se dobija da je stepen grafa 9*5=45 što ne može jer je neparan broj.

Zadaci za vežbu

1. Grafovi su prikazani na slici. Definisati skupove čvorova, grana, za svaku granu odrediti krajnje čvorove.
   a.



   b.



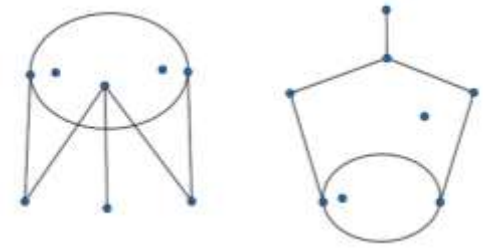
2. Nacrtati graf G definisan skupom čvorova $V = \{v_1, v_2, v_3, v_4, v_5\}$ i skupom grana $E = \{e_1, e_2, e_3, e_4\}$ ako su krajevi svake grane prikazani u tablici
   a.

Grana	Krajevi grane
e_1	$\{v_1, v_2\}$
e_2	$\{v_1, v_2\}$
e_3	$\{v_2, v_3\}$
e_4	$\{v_2\}$

   b.

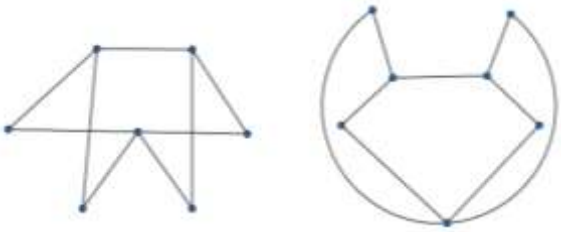
Grana	Krajevi grane
e_1	$\{v_1\}$
e_2	$\{v_2, v_3\}$
e_3	$\{v_2, v_3\}$
e_4	$\{v_1, v_5\}$

3. Da li su na slikama prikazani isti grafovi? Obrazložiti odgovor.
   a)



   b)

   c)

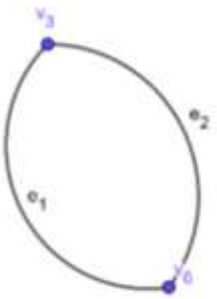


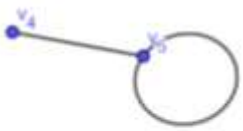
4. Za grafove sa slike odrediti sledeće skupove:
   - skup grana incidentnih sa v_1 (skup E_1),
   - skup čvorova susednih sa v_3 (skup N_{v_3}),
   - skup grana susednih sa e_1 (N_{e_1}),
   - skup svih petlji (skup L),
   - paralelnih grana (skup P),
   - odrediti stepen čvora v_3 kao i stepen grafa,
   - skup izolovanih čovorova (skup I).

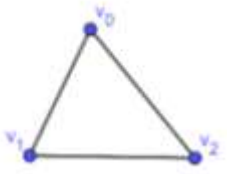
5. Da li se korišćenjem dve kofe, A i B, kapaciteta 3 i 5 litara (tim redom) može izmeriti tačno jedan litar ako su dozvoljene sledeće radnje: obe kofe se mogu napuniti do vrha, obe kofe se mogu isprazniti u celosti, dozvoljeno je prespanje tečnosti iz jedne kofe u drugu.

6. Nacrtati sledeće grafove, ukoliko takvi grafovi postoje
   - Graf sa pet čvorova stepena 1,2 ,3,3 i 5
   - Graf sa četiri čvora stepena 1,2,3 i 3
   - Graf sa četiri čvora stepena 1, 1, 1 i 4
   - Prost graf sa četiri čvora stepena 1,2,3 i 4
   - Prost graf sa pet čvorova stepena 2,3,3,3 i 5.

7. Odrediti sve neprazne podgrafe grafova sa slike



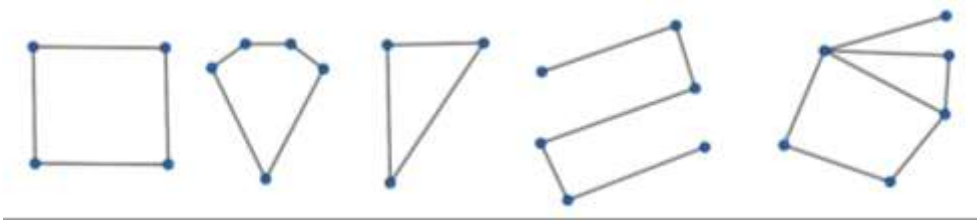




8. Da li je moguće da u grupi od 15oro ljudi svako od njih ima tačno troje prijatelja.
9. U grupi od četvoro ljudi da li je moguće da svako od njih bude prijatelj sa tačno troje iz grupe.
10. Koliko grana ima graf ako su stepeni čvorova sledeći 0,2,2,3 i 9.
11. Koliko grana ima graf ako su stepeni čvorova sledeći 1,1,4,4 i 6.
12. Nacrtati sledeće kompletne bipartitne grafove $K_{4,2}$, $K_{1,3}$ i $K_{3,4}$. Koliko čvorova imaju nacrtani grafovi. Kog stepena su posmatrani grafovi.

Bipartitni graf je prost graf čiji se skup čvorova može grupisati tako da svaki čvor iz jedne grupe može biti povezan sa čvorovima iz druge grupe ali ne sme biti povezan sa čvorovima iz svoje grupe.

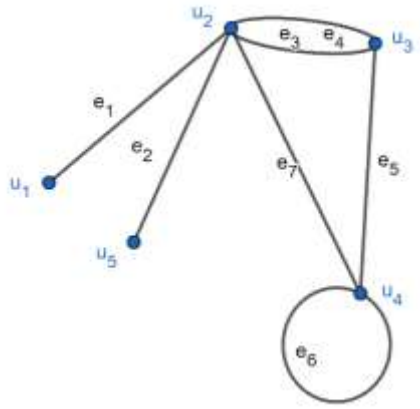
13. Među prikazanim grafovima odredite bipartitne



Graf G' se naziva **komplementom prostog grafa G** ukoliko važi sledeće: Skup čvorova grafa G' je identičan skupu čvorova grafa G, dok su dva čvora grafa G' povezana granom akko nisu povezana granom u grafu G.
Na primer, njegov komplement je nepovezan graf sa istim brojem čvorova i praznim skupom grana.

Putanje i cikli u grafovima

Posmatrajmo graf prikazan na slici



Alternirajući niz susednih čvorova i grana od čvora u_1 do čvora u_4 se naziva **šetnjom po grafu** i može se opisati na sledeći način:

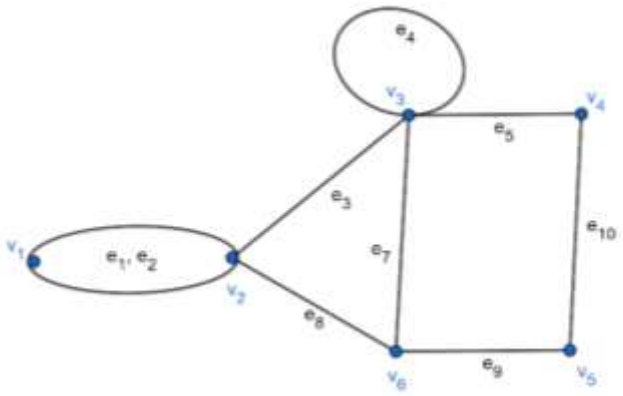
$$u_1e_1u_2e_3u_3e_4u_2e_3u_3e_5u_4e_7u_2e_3u_3e_5u_4.$$

- **Šetnja je trivijalna** ukoliko se sastoji samo iz jednog čvora.
- **Šetnja je zatvorena** ako su početni i krajnji čvor isti.
- **Put** u grafu je šetnja u kojoj se ni jedna grana ne ponavlja.
- Put je **prost** ukoliko se u njemu ni jedan čvor ne ponavlja.
- **Cikl** je zatvoren put.
- **Prost cikl** je zatvoreni prost put.
- **Trivijalni cikl** je šetnja koja se sastoji samo iz jednog čvora i ni jedne grane.

Napomena:

- Zapis $e_1e_3e_4e_2$ podrazumeva zapis $u_1e_1u_2e_3u_3e_4u_2e_2v_5$, dok se zapis e_1 može protumačiti na dva načina: $u_1e_1u_2$ i $u_2e_1u_1$.
- Zapis u_1u_2 podrazumeva jedinstven zapis $u_1e_1u_2$ dok se zapis u_2u_3 može protumačiti kao $u_2e_3u_3$ i $u_2e_4v_3$.
- Zapis $u_2u_4u_4u_3$ podrazumeva jedinstvenu putanju $u_2e_7u_4e_6u_4e_5u_3$

1. Neka je graf G prikazan na slici desno. Za svaki od ponuđenih odgovora odrediti da li on predstavlja putanju, prostu putanju, kružnicu ili prostu kružnicu.



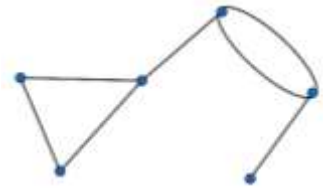
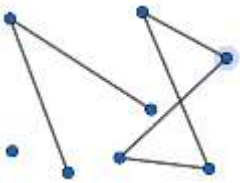
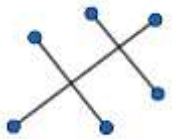
   a. $v_1\ e_1\ v_2\ e_3\ v_3\ e_4\ v_3\ e_5\ v_4$
   b. $e_1e_3e_5e_6$
   c. $v_2v_3v_4v_5v_6v_2$
   d. $v_2v_3v_4v_5v_6v_2$
   e. $v_2v_3v_4v_6v_3v_2$
   f. v_1

Rešenje

a. Opisana šetnja ima čvor koji se ponavlja ali nema granu koja se ponavlja, tako da ona predstavlja putanju od v_1 do v_4 koja nije prosta.
b. U pitanju je šetnja od v_1 do v_5. Nije u pitanju putanja je ima granu koja se ponavlja.
c. Opisana je šetnja bez ponavljajućih grana koja počinje i završava se u čvoru v_2. Dakle, u pitanju je cikl. Kako se čvor u središtu ponavlja, nije prosti cikl.
d. Opisana šetnja počinje i završava se u v_2, a da pri tom ne obilazi ni jednu granu niti čvor dva puta. Dakle, u pitanju je prost cikl.
e. Opisana šetnja predstavlja zatvorenu šetnju koja počinje i završava se u čvoru v_2. Nije u pitanju ni cikl niti prost cikl zato što se grana e_3 i čvor v_3 ponavljaju.
f. Prvi i poslednji čvor šetnje su isti. Takođe šetnja ne ponavlja ni jedan čvor niti granu. Dakle, u pitanju je prost cikl koji se još naziva **trivijalnim ciklom.**

Neka je G graf. Dva čvora *u* i *w* su **povezana** ako i samo ako postoji neka šetnja od čvora *u* do čvora *w.* **Graf je povezan** graf akko za svaki par čvorova posmatranog grafa postoji šetnja od jednog do drugog.
Na primer, graf pod a) je povezan dok grafovi pod b) i c) to nisu

a) b) c)

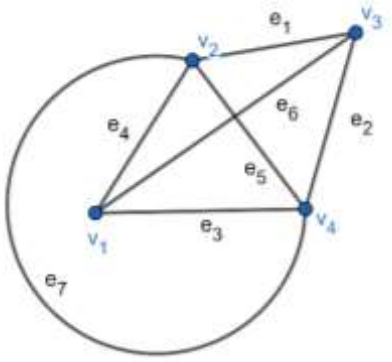




Definicija
Cikl koji sadrži sve grane grafa naziva se **Ojlerovim ciklom**. Graf se naziva Ojlerovim ako nema izolovanih čvorova i ima Ojlerov cikl.

Teorema
Ukoliko graf ima Ojlerov cikl, tada je stepen svakog čvora tog grafa paran.

Teorema
Ako je graf G povezan i ako je stepen svakog njegovog čvora paran, tada takav graf ima Ojlerov cikl.

Da li se za graf na slici može odrediti Ojlerov cikl?



Rešenje:
Stepen čvorova v_1 i v_3 je 3, što je neparna vrednost. Stoga se na osnovu date teoreme može zaključiti da za ovakav graf Ojlerov cikl ne postoji.

Algoritam za određivanje Ojlerovog cikla:

Korak 1: Proizvoljno izabrati čvor grafa kao početni

Korak 2: Izabrati bilo koju cikl koji počinje i završava se u izabranom čvoru ne ponavljajući ni jednu granu. Označimo konstruisan cikl sa C.

Korak 3: Proverite da li C sadrži svaku granu i svaki čvor grafa G. Ako je odgovor potvrdan, C je Ojlerova putanja i algoritam je završen. Inače se primenjuju naredni koraci.

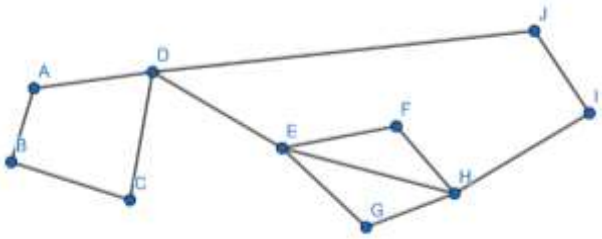
Korak 3a: Ukloniti sve grane iz grafa G koje odgovaraju ciklu C i sve čvorove koji će ostati izolovani nakon brisanja grana. Nazovimo rezultujući podgraf G1. Pri tom, može da se dogodi da graf G1 ne bude povezan i da stepen svakog njegovog čvora bude paran.

Korak 3b: Izabrati bilo koji čvor *w* koji pripada i ciklu C i grafu G1

Korak 3c: Na grafu G1 izabrati bilo koji cikl koji počinje i završava se u *w* bez ponavljanja grana. Nazovimo rezultujući cikl C1.

Korak 3d: Spojimo ciklove C i C1 u cikl C2 na sledeći način: Polazeći od *v* koristimo cikl C do *w*, a zatim duž cikla C1 idemo nazad do *w*. Dalje se od čvora *w* duž cikla C vraćamo do čvora *v*.

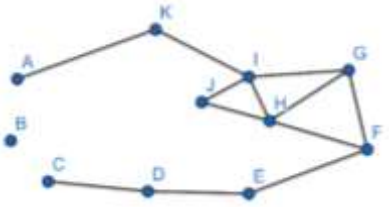
2. Za posmatrani graf odrediti Ojlerov cikl (ukoliko postoji).



Rešenje:
Primetićemo da je stepen svakog čvora paran i da je graf povezan. Stoga Ojlerov cikl postoji. Odredićemo ga tako što ćemo da pratimo korake opisane u prethodnom algoritmu.
Neka je cikl C definisan na sledeći način C: $ABCDA$.
Kako opisan cikl ne prolazi kroz sve čvorove grafa, formiraćemo pomoćni cikl $C1$: $DEGHIJD$
Spajanjem ciklova C i C1 dobija se cikl $C2$: $ABCDEGHIJDA$
Dobijeni cikl označimo kao C i proveravamo da li je on Ojlerov. Međutim, dobijen cikl nije Ojlerov jer ne prolazi kroz čvor f.
Određujemo pomoćni cikl $C1$: EFHE. Spajamo ciklove C i $C1$ u novi cikl $C2$: $ABCDEFHEGHIJDA$. Izjednačimo C2 sa C. Dobijeni cikl je Ojlerov.

Ojlerov put je put koji sadrži sve grane tog grafa.

3. Za posmatrani graf odrediti Ojlerov put od čvora A do čvora B.



Hamiltonov cikl

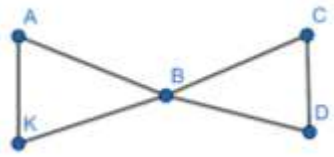
Hamiltonovim ciklom grafa G nazivamo prost cikl koji prolazi kroz svaki čvor grafa tačno jedanput (osim prvog i poslednjeg koji su isti) prolazeći kroz sve grane grafa.

Hamiltonovim putem od *u* do *v* nazivamo put koji sadrži svaki čvor grafa tačno jednom a čiji su krajevi čvorovi *u* i *v*.

Ako graf H ima netrivijalni cikl, onda graf G ima podgraf H putanju sa sledećim osobinama

1) H sadrži svaki čvor grafa G
2) H je povezan graf
3) H ima isti broj čvorova i grana
4) Svaki čvor u H ima stepen 2

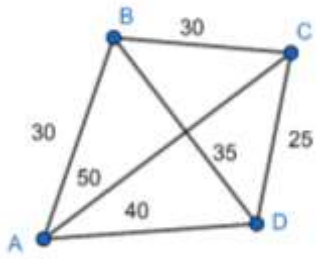
4. Da li sledeći graf ima Hamiltonov put?



Rešenje:
Ako bi posmatrani graf imao Hamiltonov put, tada bi mogao da se izdvoji podgraf H sa 5 čvorova i 5 grana takvih da svaki čvor grafa ima stepen 2. Pošto je stepen čvora b jednak 4 potrebno je da se obrišu dve grane. Ako obrišemo granu $\{a, b\}$ čvor a će imati stepen 1 a ne 2 kao što treba. Slično bi moglo da se pokaže i za ostale grane, što nas dovodi do zaključka da se za posmatrani graf ne može pronaći Hamiltonov put.

Specijalan primer Hamiltonovog puta je **problem Trgovačkog putnika** koji se može opisati na sledeći način:

Neka je na grafu niže postavljena mreža gradova i tačno rastojanje među njima kao težina grana. Ako Trgovački putnik treba da poseti svaki grad tačno jedanput i ako svoju putanju treba da započne i završi u gradu A, odrediti putanju Trgovačkog putnika tako da ukupno pređeno rastojanje bude minimalno.



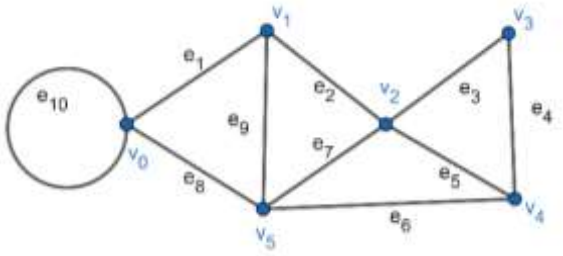
Rešenje:
Rešićemo problem Trgovačkog putnika tako što ćemo da izračunamo dužine svih putanja a zatim da izaberemo najkraću među njima:

ABCDA : 30+30+25+40=125
ABDCA : 30+35+25+50=140
ACBDA : 50+30+35+40=155
ACDBA : 140 (put ABDCA samo unazad)
ADBCA : 155 (put ACBDA samo unazad)
ADCBA : 125 (put ADCBA samo unazad)

Najkraći putevi imaju 125 kilometara: ABCDA i ADCBA

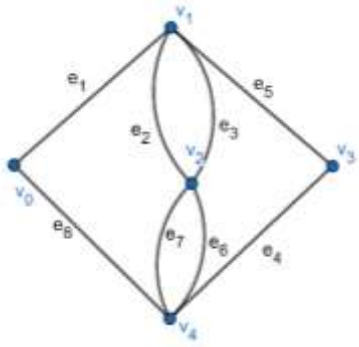
Zadaci za vežbu

1. Za svaku šetnju odrediti da li ona predstavlja put, prost put, zatvoreni put, cikl, prost cikl ili ništa od navedenog
   a. $v_1 e_2 v_2 e_3 v_3 e_4 v_4 e_f v_2 e_2 v_1 e_1 v_0$
   b. $v_2 v_3 v_4 v_5 v_2$
   c. $v_4 v_2 v_3 v_4 v_5 v_2 v_4$
   d. $v_0 v_5 v_2 v_3 v_4 v_2 v_1$
   e. $v_2 v_1 v_5 v_2 v_3 v_4 v_2$
   f. $v_5 v_4 v_2 v_1$

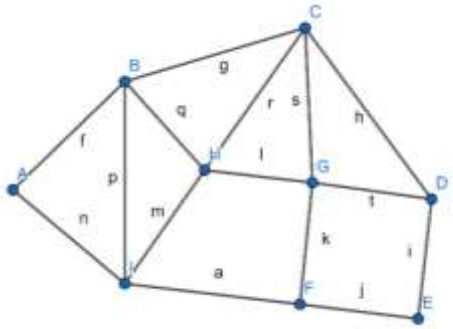


2. Da li grafovi na slici sadrze Ojlerov cikl

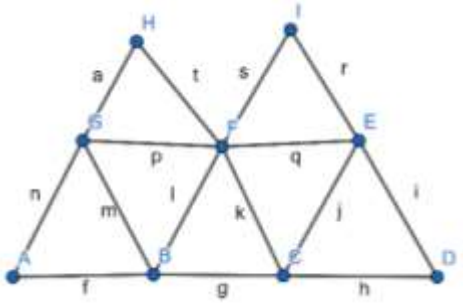
   a.



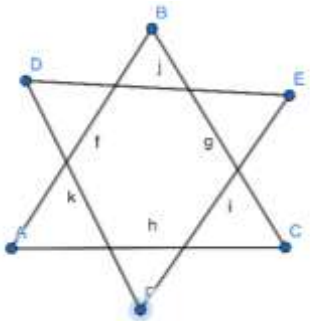
   b.



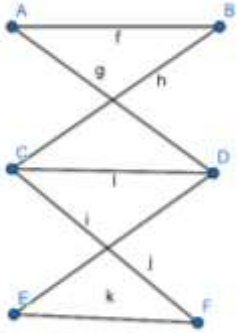
   c.



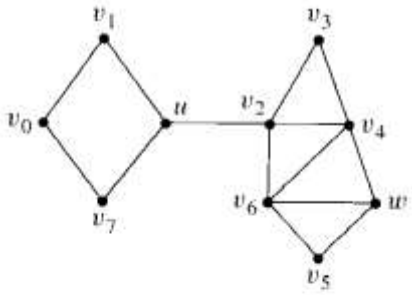
   d.

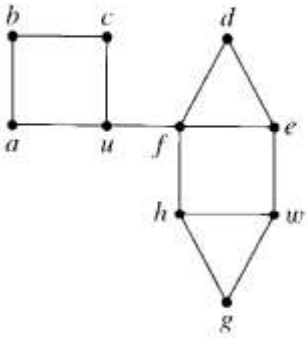


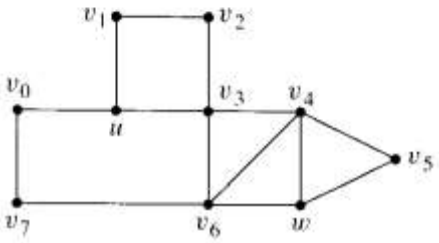
   e.



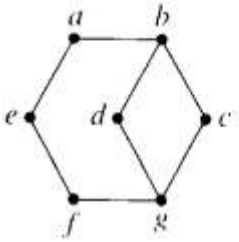
3. Da li postoji Ojlerov put od *u* do *w*



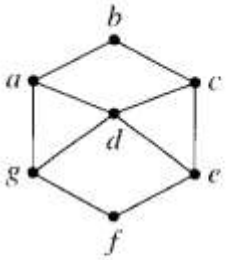


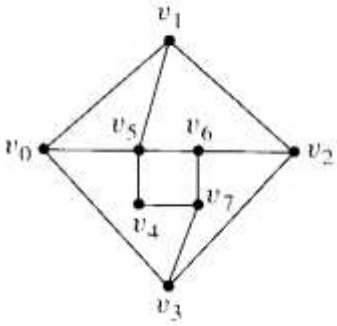


4. Naci Hamiltonove cikle

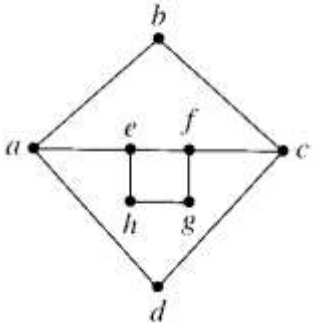


29.



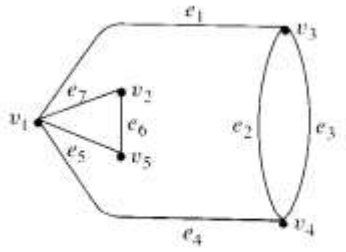


31.

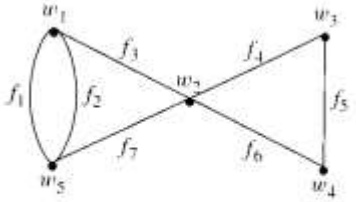


Kažemo da su dva grafa $G_1 = (V_1, E_1)$ i $G_2 = (V_2, E_2)$ **izomorfna** akko postoje bijekcije $g: V_1 \to V_2$ i $h: E_1 \to E_2$ takve da važi: ako je v kraj grane e u grafu G_1, onda je I $g(v)$ kraj grane $h(e)$ u grafu G_2.

Grafovi na slici su izomorfni:

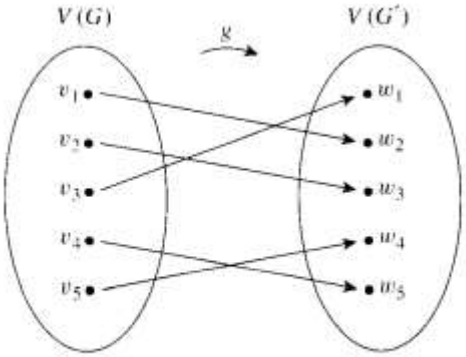


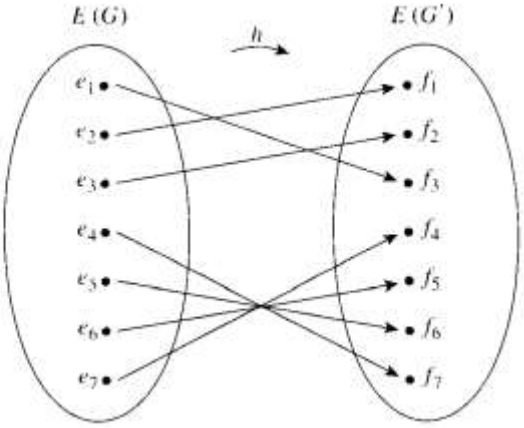
G



G'

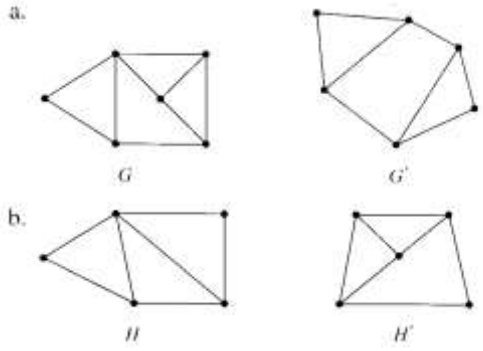
Njima odgovarajuće bijekcije su





Na slici su prikazani grafovi koji imaju dva čvora i dve grane, a koji međusobno nisu izomorfni, tj. svaki drugi graf koji ima dva čvora i dve grane je izomorfan jednom od prikazana 4 grafa

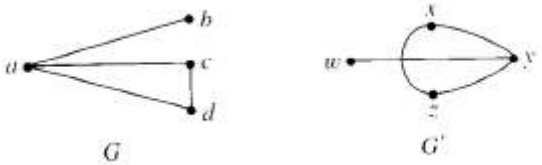
7. Da li su prikazani grafovi međusobno izomorfni



Rešenje

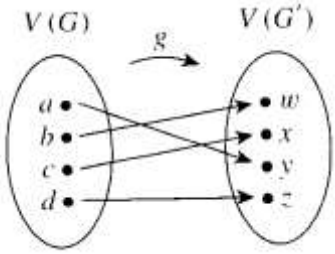
a. G ima 9 čvorova dok G’ ima samo 8. Nisu izomorfni
b. H ima čvor stepena 4, dok H’ nema takav čvor. Dakle, nisu izomorfni

8. Da li su grafovi na slici međusobno izomorfni? Definisati izomorfizam



Rešenje:

Jesu. Izomorfizam $f: V(G) \to V(G')$ je definisan na sledeći način:



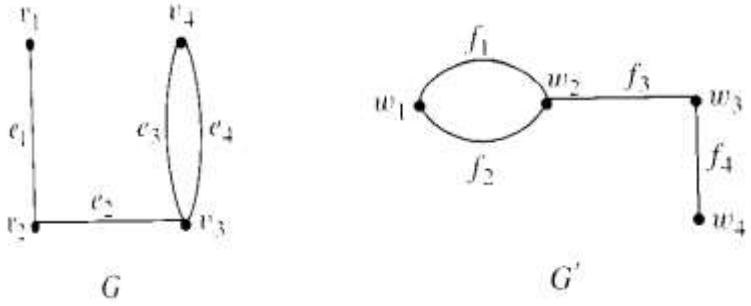
Dok je funkcija g definisana na sledeći način

Grana	Grane u G'
$\{a,b\}$	$\{y,w\} = \{g(a), g(b)\}$
$\{a,c\}$	$\{y,x\} = \{g(a), g(c)\}$
$\{a,d\}$	$\{y,z\} = \{g(a), g(d)\}$
$\{c,d\}$	$\{x,y\} = \{g(c), g(d)\}$

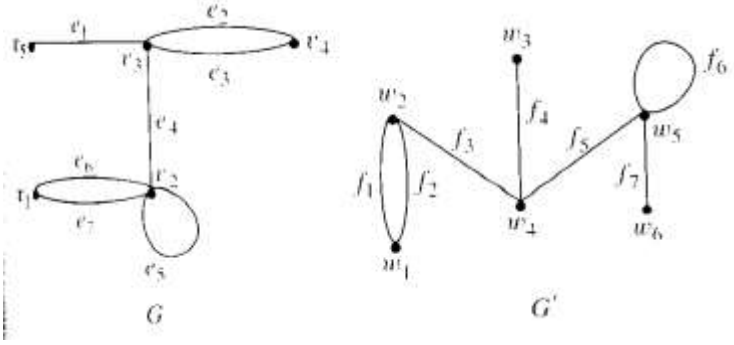
Zadaci za vežbu

1. Da li su seledeći grafovi izomorfni. Ako jesu, odrediti preslikavanja g, i h. Ako nisu, odrediti njima izomorfne varijante po kojima se razlikuju.

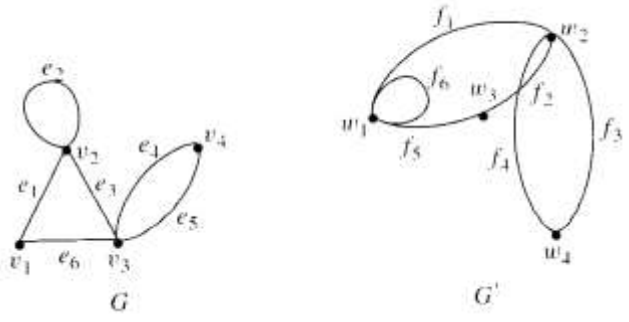
a.



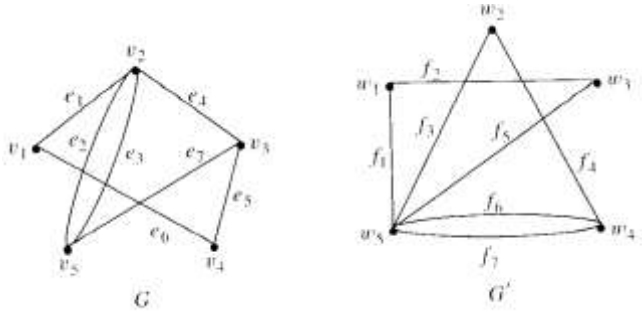
b.



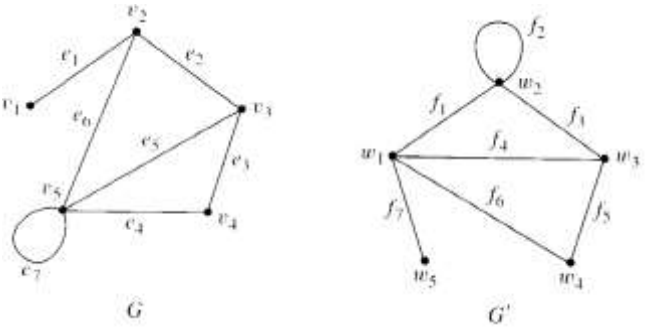
c.



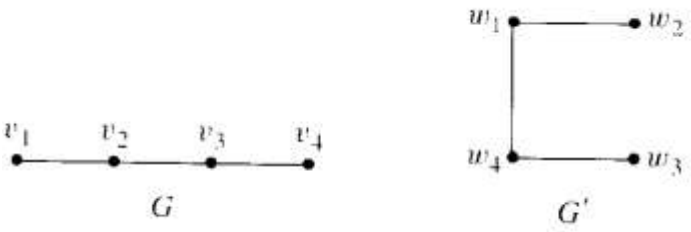
d.



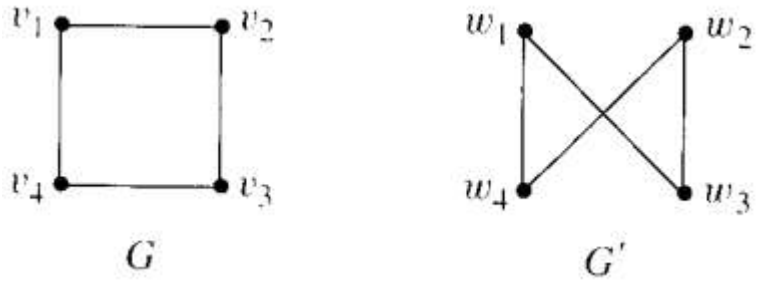
e.



f.



g.



2. Nacrtati sve neizomorfne proste grafove sa tri čvora
3. Nacrtati sve neizomorfne proste grafove sa četiri čvora
4. Nacrtati sve neizomorfne grafove sa tri čvora i sa ne više od dve grane

MATLAB FUNKCIJE VEZANE ZA GRAFOVE

U okviru ovog poglavlja predstavićemo neke od Matlab-ovih ugrađenih funkcija za rad sa grafovima. Pored funkcija za crtanje grafova, biće reči o funkcijama za dodavanje/brisanje čvorova/grana, funkcijama za određivanje najkraće putanje između dva čvora, maksimalnog protoka, kao i o funkcijama za određivanje pozicije određenih čvorova/grana.

Graf se definiše pomoću skupa čvorova i skupa grana, odnosno preko matrice susedstva. Dovoljno je da se unese matrica susedstva ili da se unesu skupovi čvorova i skupovi grana. Prilikom definisanja imena čvorova Matlab koristi promenljivu nodenames koja je tipa cell array ili vektor karaktera. Grane mogu da se predstave preko dva celobrojna vektora (s i t) koja predstavljaju skup čvorova iz kojih grana izlazi i skup čvorova u koje grane ulaze (s i t sadrže indekse tih čvorova). Grane mogu biti težinske i u tom slučaju se koristi promenljiva weights (koja može da bude skalar, vektor, matrica ili višedimenzioni niz).

Broj čvorova se pamti kao pozitivni celobrojni skalar num, dok se informacije o čvorovima čuvaju u tabeli NodeTable. Za svaki čvor definisane su funkcije poput:

- degree (određuje stepen čvora)
- neighbors (određuje susede čvoru grafa)
- nearest (traži najbližeg suseda u okviru oređenog radijusa)
- outedges (određuje grane koje izlaze iz čvora)

Informacije o granama se čuvaju u tabeli EdgeTable. Čvorovima i granama se pristupa preko sledećih pomoćnih funkcija:

- addedge
- rmedge
- addnode
- rmnode
- findedge
- findnode
- numedges
- numnodes
- edgecount
- reordernodes
- subgraph

Od funkcija koje se odnose na pretragu na grafu i proveru strukture razlikuju se sledeće funkcije

- bfsearch (pretraga u širinu)
- dfsearch (pretraga u dubinu)
- centrality (određuje bitnost čvora)
- conncomp (određuje broj povezanih kompomenti)
- biconncomp (određuje broj povezaanih bikomponenti)
- bctree (formira „blok-cut stablo“)
- maxflow (određuje maksimalni protok)
- minspantree (najmanje razapinjujuće stablo)
- isisomorphism (proverava da li su dva grafa izimorfna)
- isomorphism (računa izomorfnost između dva grafa)
- ismultigraph (da li graf ima višestruke grane)
- simplify (svodi graf sa višestrukim granama na prosti)

Najkraće rastojanje između dva čvora grafa određuje se korišćenjem funkcija

- shortestpath
- shortestpatrhtree
- distances

- Graf G=(V,E) za koji važi V={1,2,3},E={{1,2},{1,3}} definišemo na sledeći način:

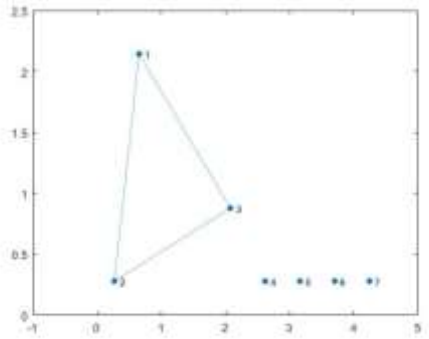
```
  >> G = graph([1 1], [2 3])
graph with properties:

  Edges: [2×1 table]
  Nodes: [3×0 table]
```

- Matrici **e** dodeljujemo grane

```
   >> e = G.Edges
   e =
       EndNodes
       ________

       1     2
       1     3
```

- Grafu G dodajemo granu između čvorova 2 i 3

```
  >> G = addedge(G,2,3);
```



- Grafu G dodajemo 4 čvora

```
  >> G = addnode(G,4);
```

- Crtamo graf

```
  >> plot(G);
```

Funkcije za rad sa grafovima

`G = graph()` - prazan neorijentisan graf koji ne sadrži čvorove niti grane.

`G = graph(A)` - težinski graf definisan pomoću simetrične kvadratne matrice susedstva, matrice A. U matrici $A = [a_{ij}]_{nxm}$ vrednosti $a_{ij} = 0$ označavaju da između čvorova i i j ne postoji grana, dok $a_{ij} \neq 0$ predstavlja težinu grane između čvorova i i j:
A(2,4) = 15 označava da je grana između čvorova 2 i 4 težine 15.

`G = graph(A,nodenames)` - dodeljuje ime čvorovima. Vektor **nodenames** je dužine size(A,1).

`G = graph(A,NodeTable)` - definiše imena čvorova (i još neke osobine čvorova) korišćenjem tabele **NodeTable**. **NodeTable** ima isti broj vrsta kao i matrica A. Imena čvorova se nalaze u okviru promenljive **Name** tabele.

`G = graph(A,___,type)` - definiše trougaonu matricu na osnovu matrice susedstva A koju koristi pri definisanju grafa.

`G = graph(A,___,'omitselfloops')` - ignoriše dijag.elem. matrice A vraća graf bez petlji.

`G = graph(s,t)` - definiše grane grafa za svaki par čvorova (s,t).
s i **t** predstavljaju indeke čvorova ili imena čvorova.

`G = graph(s,t,weights)` - definiše težinske grane na osnovu vektora weights.

`G = graph(s,t,weights,nodenames)` - defineše imena čvorova korišćenjem **cell array** nizove string nizova **nodenames**.
s i t ne smeju da sadrže imena koja nisu u nodenames.

`G = graph(s,t,weights,NodeTable)` - definiše imena čvorova (i još neke njihove osobine) korišćenjem tabele **NodeTable**. Definisanje imena

korišćenjem promenljive **Name**. **s** i **t** ne mogu da sadrže ime koje se ne nalazi u tabeli NodeTable.

`G = graph(s,t,weights,num)` - definiše broj čvorova grafa koji sadrže numeričku promenljivu **num**.

`G = graph(s,t,___,'omitselfloops')` –ne dozvoljava petlje u grafu, tj. ignoriše grane za koje važi **`s(k) == t(k).`**

`G = graph(s,t,EdgeTable,___)` - koristi tabelu kako bi definisao neke osobine grana umesto definisanja **weights**. Tabela **EdgeTable** mora da sadrži vrstu za svaki par elemenata s i t. Težine čvorova se dodeljuju korišćenjem promenljive Weight.

`G = graph(EdgeTable)` - koristi tabelu EdgeTable kako bi definisao graf. Prva promenljive tabele **EdgeTable** je **EndNodes**, čije su prve dve kolone nizovi čvorova koji definišu grane.

`G = graph(EdgeTable,NodeTable)` - dodatno definiše imena (i još neke osobine) grafa korišćenjem NodeTable.

`G = graph(EdgeTable,___,'omitselfloops')` - ne dozvoljava petlje, tj. ako za neko k važi **`EdgeTable.EndNodes(k,1) == EdgeTable.EndNodes(k,2)`**, ona to ignoriše. Mora da se definiše EdgeTable i opciono NodeTable.

Primer1

Formirati graf sa 3 čvora (A, B i C) i dve grane tako da je jedna grana između čvorova 1 i 2 a druga između čvorova 1 i 3.

Rešenje:

```
>> G = graph([1 1],[2 3])
G =
  graph with properties:
    Edges: [2×1 table]
    Nodes: [3×0 table]

>> G.Edges
ans =
    EndNodes
    ________

    1    2
    1    3

>> G.Nodes
ans =
   empty 3-by-0 table
```

```
>> G.Nodes.Name = {'A' 'B' 'C'}';
>> G.Nodes
ans = 3×1 table
    Name
    _____

    {'A'}
    {'B'}
    {'C'}

G.Edges
ans=2×1 table
       EndNodes
    _______________

    {'A'}    {'B'}
    {'A'}    {'C'}
```

Dodavanje čvorova vrši se na sledeći način:

```
G = addedge(G,2,3)

G =
  graph with properties:

    Edges: [3x1 table]
    Nodes: [3x1 table]
```

```
G.Edges

ans=3×1 table
       EndNodes
    _______________

    {'A'}    {'B'}
    {'A'}    {'C'}
    {'B'}    {'C'}
```

Brisanje čvorova i grana vrši se korišćenjem sledećih funkcija:

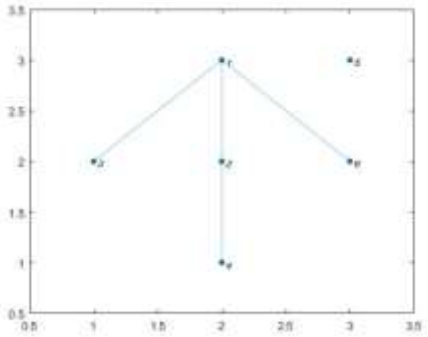
- addedge,
- rmedge,
- addnode,
- rmnode

Primer2

Grafu iz primera1 dodati dve grane, jednu između čvorova 2 i 4 a drugu između čvorova 4 i 6. Nacrtati formirani graf.

Rešenje:

```
>> G = addedge(G,[2 1], [4 6])

G =

  graph with properties:

    Edges: [4×1 table]
    Nodes: [6×0 table]

>> plot(G)
```



Primer3

Formirati matricu susedstva A za kompletan besteżinski graf reda 4.

Rešenje:

```
>> A = ones(4) - diag([1 1 1 1])
>> A = 4×4

     0     1     1     1
     1     0     1     1
     1     1     0     1
     1     1     1     0

>> G = graph(A~=0)
G =
  graph with properties:

    Edges: [6x1 table]
    Nodes: [4x0 table]

G.Edges
ans=6×1 table
    EndNodes
    ________

     1    2
     1    3
     1    4
     2    3
     2    4
     3    4
```

Primer4

Formirati matricu susedstva A za proizvoljan kompletan težinski graf reda 4

Rešenje:

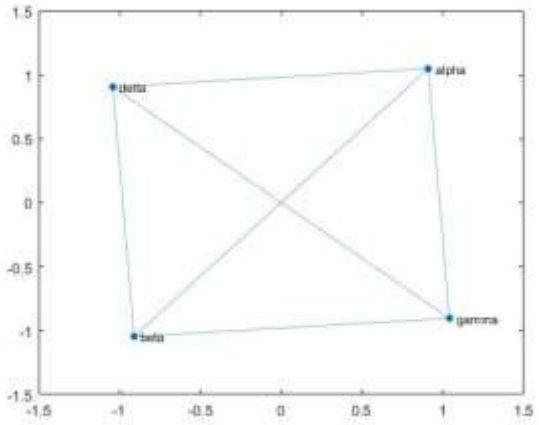
Formiraćemo proizvoljnu (magic funkcija) gornje-trougaonu matricu (triu)

```
>> A = triu(magic(4))
>> A =

    16     2     3    13
     0    11    10     8
     0     0     6    12
     0     0     0     1
```

Čvorovima dodeljujemo imena.

Koristićemo opciju `'omitselfloops'` kako bismo definisali graf bez petlji i opciju `'upper'` kako bismo označili da je u pitanju gornje trougaona matrica.

```
names = {'alpha' 'beta' 'gamma'
'delta'};
G =
graph(A,names,'upper','omitselfloops')
G =
  graph with properties:

    Edges: [6x2 table]
    Nodes: [4x1 table]
```




```
>> G.Edges
ans =
          EndNodes          Weight
    ____________________    ______

    'alpha'    'beta'        2
    'alpha'    'gamma'       3
    'alpha'    'delta'      13
    'beta'     'gamma'      10
    'beta'     'delta'       8
    'gamma'    'delta'      12
```

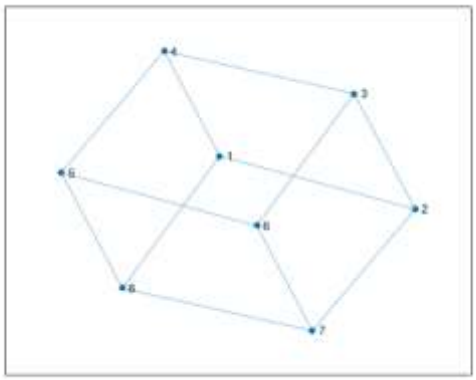
```
>> G.Nodes
ans =
     Name
    ________

    'alpha'
    'beta'
    'gamma'
    'delta'
```

Primer5

Neka su za svaku granu dati ulazni i izlazni čvor kao preko vektora **s** i **t**. Na primer, ako graf ima grane 1-3, 1-4, 2-3, vektori su sledećeg oblika: s=[1,1,2], t =[3,4,3]. Formirati graf.

Rešenje:

```
>> s = [1 1 1 2 2 3 3 4 5 5 6 7];
>> t = [2 4 8 3 7 4 6 5 6 8 7 8];
G = graph(s,t)

G =

  graph with properties:

    Edges: [12×1 table]
    Nodes: [8×0 table]
```

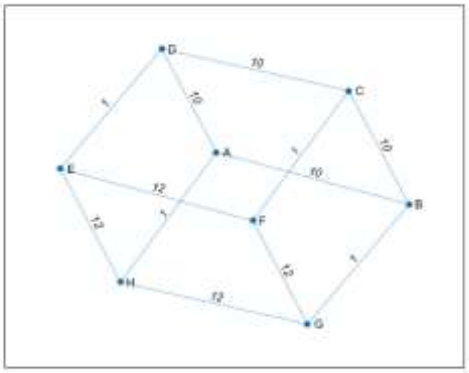


Primer6

Neka su za svaku granu dati ulazni i izlazni čvor vektorima s i t. Formirati težinski graf a čvorovima grafa dodeliti imena ‘A’ – ‘H’.

Rešenje:

```
>> s = [1 1 1 2 2 3 3 4 5 5 6 7];
>> t = [2 4 8 3 7 4 6 5 6 8 7 8];
>> weights = [10 10 1 10 1 10 1 1 12 12
12 12];
>> names = {'A' 'B' 'C' 'D' 'E' 'F' 'G'
'H'};
>> G = graph(s,t,weights,names)
G =
  graph with properties:

    Edges: [12x2 table]
    Nodes: [8x1 table]

plot(G,'EdgeLabel',G.Edges.Weight)
```

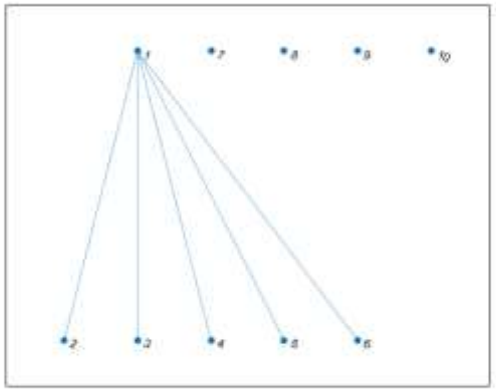


Primer7

Formirati graf korišćenjem niza čvorova. Zadati graf mora sadržati tačno 10 čvorova.

Rešenje:

```
>> s = [1 1 1 1 1];
>> t = [2 3 4 5 6];
>> weights = [5 5 5 6 9];
>> G = graph(s,t,weights,10)

G =
  graph with properties:

    Edges: [5x2 table]
    Nodes: [10x0 table]
>> plot(G)
```



Primer8

Formirati prazan graf. Formiranom grafu dodeliti tri čvora i tri grane. Grane definisati vektorima s i t korišćenjem funkcije addedge.

Rešenje:

```
>> G = graph;

>> s = [1 2 1];
>> t = [2 3 3];
>> G = addedge(G,s,t)

G =
  graph with properties:

    Edges: [3x1 table]
    Nodes: [3x0 table]
```

```
>> G.Edges

ans=3×1 table
    EndNodes
    ________

     1    2
     1    3
     2    3
```

- Najbolje je da se graf formira korišćenjem funkcije graph. Dodavanje čvorova i grana kod velikih grafova se sporije izvršava.

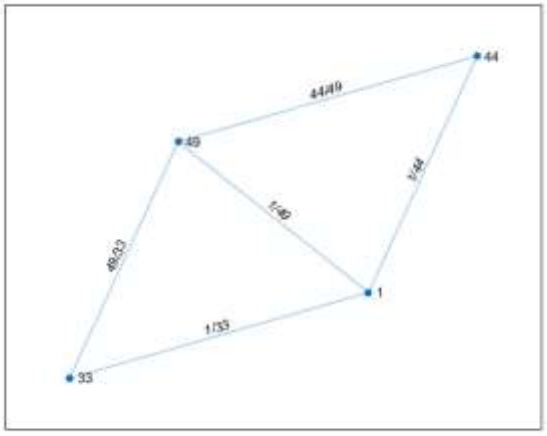
Primer9

Konstruisati tabelu grana koja sadrži promenljive EndNodes, Weights i Code. Kreirati zatim tabelu sa čvorovima koja sadrži imena čvorova Name i promenljivu County. Osobine grafova definisati u svakoj tabeli.

Rešenje:

```
>> s = [1 1 1 2 3];
>> t = [2 3 4 3 4];
>> weights = [6 6.5 7 11.5 17]';
>> code = {'1/44' '1/49' '1/33' '44/49' '49/33'}';
>> EdgeTable = table([s' t'],weights,code, ...
    'VariableNames',{'EndNodes' 'Weight' 'Code'})

>> EdgeTable=5×3 table
    EndNodes    Weight      Code
    ________    ______    _________

     1    2         6     {'1/44' }
     1    3       6.5     {'1/49' }
     1    4         7     {'1/33' }
     2    3      11.5     {'44/49'}
     3    4        17     {'49/33'}


>> names = {'USA' 'GBR' 'DEU' 'FRA'}';
>> country_code = {'1' '44' '49' '33'}';
>> NodeTable = table(names,country_code,'VariableNames',{'Name' 'Country'})

NodeTable=4×2 table
     Name      Country
    _______    _______

    {'USA'}    {'1' }
    {'GBR'}    {'44'}
    {'DEU'}    {'49'}
    {'FRA'}    {'33'}
```

```
>> G = graph(EdgeTable,NodeTable);
>> plot(G,'NodeLabel',G.Nodes.Country, 'EdgeLabel',G.Edges.Code)
```



Primer10

Konstruisati i nacrtati graf a zatim vratiti stepene svih čvorova i konkretno zadatog čvora odvojeno.

Rešenje:

```
>> s = [1 1 1 4 4 6 6 6];
>> t = [2 3 4 5 6 7 8 9];
>> G = graph(s,t);
>> plot(G)
>> deg = degree(G)
deg = 9×1

     3
     1
     1
     3
     1
     4
     1
     1
     1
```

Stepen čvora j je deg(j)

```
>> s = {'a' 'a' 'a' 'd' 'd' 'f' 'f' 'f'};
>> t = {'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i'};
>> G = graph(s,t);
>> plot(G)

>> nodeIDs = {'a' 'c' 'e'}';
>> deg = degree(G,nodeIDs)

deg = 3×1

     3
     1
     1
```

Stepen čvora nodeIDs(j) je deg(j)

U zavisnosti od načina indeksiranja čvorova, stepen određujemo na sledeći način

	Jedan čvor	Više čvorova
Index čvora	Skalar (Na primer: 1)	Vektor (Na primer: [1,2,3])
Ime čvora	Vektor karaktera (Na primer: 'A') String (Na primer: "A")	Cell array vektora karaktera (Na primer: {'A', 'B', 'C'} Niz stringova (["A" "B" "C"])

Primer11

```
D = degree(G,[3 4])
D = degree(G,{'LAX','ALB'})
```

- D je numerički niz (kolona).

Kod čvorova koji sadrže petlju stepen se uvećava za 2 za svaku petlju.

`>> eid = inedges(G, nodeID)` vraća indekse svih grana koji ulaze u čvor nodeID.
`>> [eid, nid] = inedges(G,nodeID)` dodatno vraća i indekse svih čvorova koji ulaze u njega.

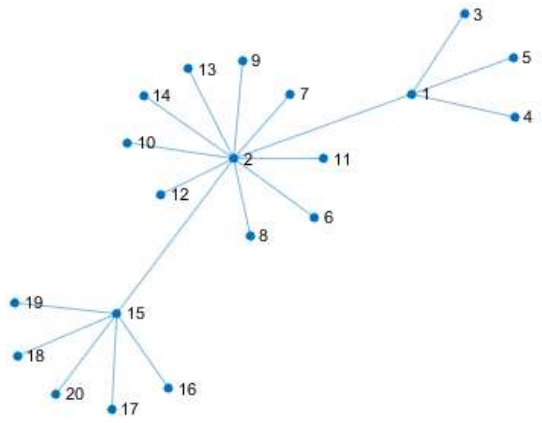
`>> eid = outedges(G, nodeID)` vraća indekse svih grana koji izlaze iz čvora nodeID.
`>> [eid, nid] = ioutedges(G,nodeID)` dodatno vraća i indekse svih čvorova koji izlaze iz njega.

```
>> highlight(p,1,'NodeColor','r','MarkerSize',10) obeležava čvor čiji je indeks 1 crvenom bojom
i večičinom 10.
>> highlight(p,nid,'NodeColor','g','MarkerSize',9)
>> highlight(p,'Edges',eid,'EdgeColor','g')
```

`>> sucIDs = successors(G,nodeID)` vraća čvorove koji izalze iz čvora čiji je indeks nodeID.

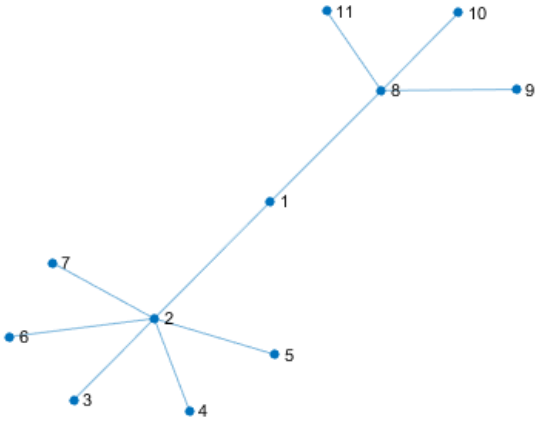
Izdvajanje podgrafa

```
>> s = [1 1 1 1 2 2 2 2 2 2 2 2 2 15 15 15 15 15];
>> t = [3 5 4 2 14 6 11 12 13 10 7 9 8 15 16 17 19 18 20];
>> G = graph(s,t);
>> plot(G,'Layout','force')
```



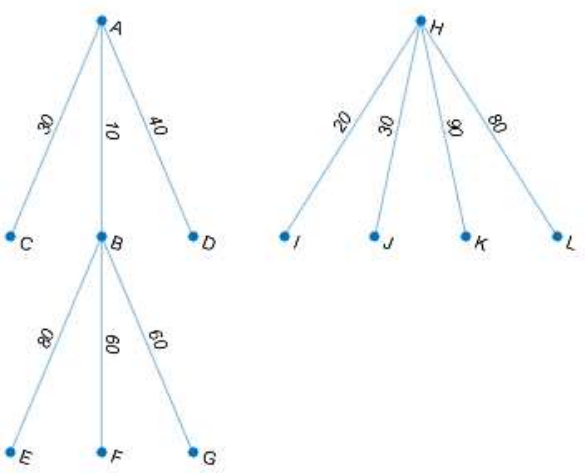
Ako želimo da izvojimo podgraf grafa G sa unapred definisanim skupom čvorova, potrebno je da popišemo indekse tih čvorova, na primer:

```
>> idx = [2 15 16 17 18 19 20 1 3 4 5];
>> H = subgraph(G,idx);
>> plot(H,'Layout','force')
```



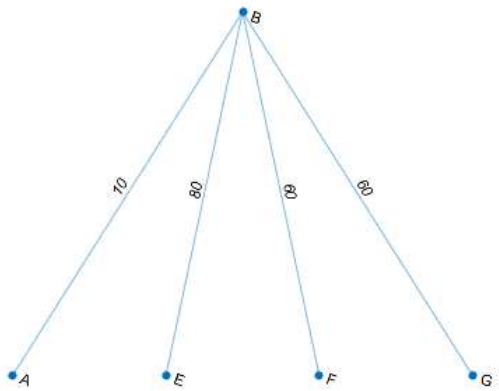
Posebno, možemo da izdvojimo čvor i njegove susede:

```
>> s = [1 1 1 2 2 2 8 8 8 8];
>> t = [2 3 4 5 6 7 9 10 11 12];
>> weights = [10 30 40 80 60 60 20 30 90 80];
>> names = {'A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K' 'L'};
>> G = graph(s,t,weights,names);
>> plot(G,'EdgeLabel',G.Edges.Weight)
```



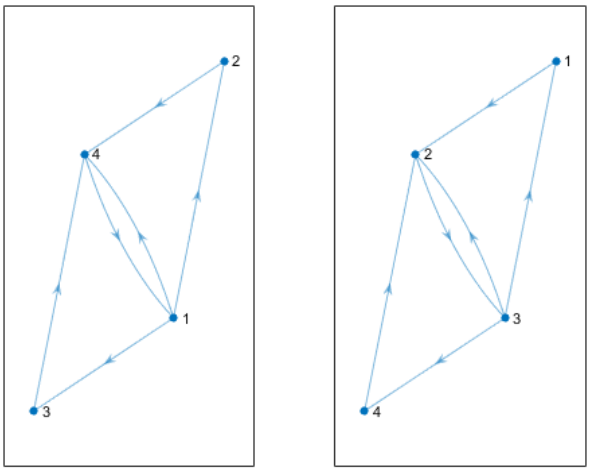
Ako želimo da izdvojimo čvor ‘B’ i sve njegove susede koristimo sledeće naredbe:

```
>> N = neighbors(G,'B');
>> H = subgraph(G, ['B'; N]);
>> plot(H,'EdgeLabel',H.Edges.Weight)
```



Proverićemo da li su dva grafa izomorfna na sledeći način:

```
>> G1 = digraph([1 1 1 2 3 4],[2 3 4 4 4 1]);   % definišemo dva grafa
>> G2 = digraph([3 3 3 2 1 4],[1 4 2 3 2 2]);
>> subplot(1,2,1)                               %nacrtamo ih jedan pored drugog
>> plot(G1)
>> subplot(1,2,2)
>> plot(G2)
```



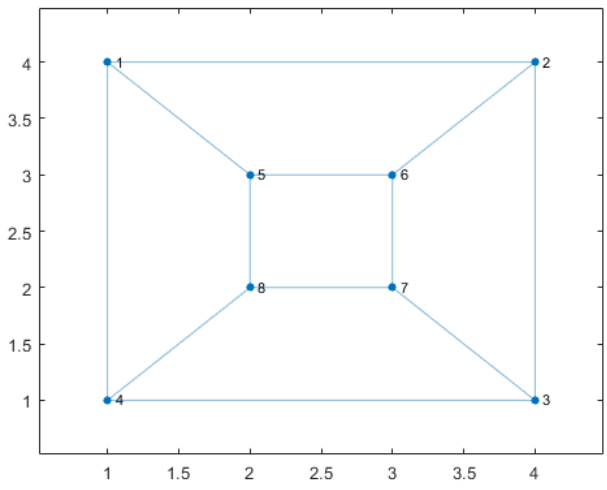

```
>> isisomorphic(G1,G2)
>> ans = logical
   1
```

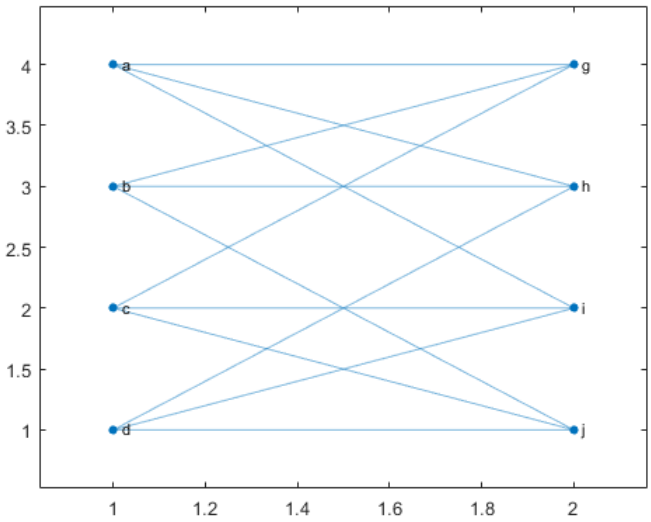
Dakle, funkcija **isisomorphic**(G1,G2) kao rezultat vraća 1 ukoliko postoji izomorfizam između dva navedena grafa, odnosno nulu u suprotnom.

Možemo da proverimo da li su dva grafa izomorfna i u slučaju da su drugačije označeni:

```
>> G1 = graph([1 1 1 2 2 3 3 4 5 5 7 7],[2 4 5 3 6 4 7 8 6 8 6 8]);
>> plot(G1,'XData',[1 4 4 1 2 3 3 2],'YData',[4 4 1 1 3 3 2 2])
```

```
>> G2 = graph({'a' 'a' 'a' 'b' 'b' 'b' 'c' 'c' 'c' 'd' 'd' 'd'}, ...
    {'g' 'h' 'i' 'g' 'h' 'j' 'g' 'i' 'j' 'h' 'i' 'j'});
>> plot(G2,'XData',[1 2 2 2 1 2 1 1],'YData',[4 4 3 2 3 1 2 1])
```





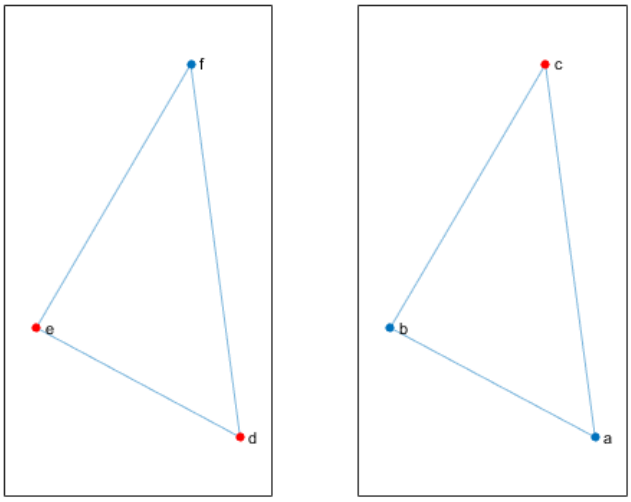

```
>> tf = isisomorphic(G1,G2)
>> tf = logical
   1
```

Odnosno, možemo da fiksiramo određene čvorove i proverimo izomorfizam u odnosu na njih:

```
>> G1 = graph({'d' 'e' 'f'},{'e' 'f' 'd'});
>> G1.Nodes.Color = {'red' 'red' 'blue'};    % dodelili smo im osobinu "boja"

>> G2 = graph({'a' 'b' 'c'},{'b' 'c' 'a'});
>> G2.Nodes.Color = {'blue' 'blue' 'red'};
```

Nacrtaćemo grafove a čvorove koji su označeni kao crveni obojiti crvenom bojom:

```
>> subplot(1,2,1)
>> p1 = plot(G1);
>> highlight(p1,{'d' 'e'},'NodeColor','r')

>> subplot(1,2,2)
>> p2 = plot(G2);
>> highlight(p2,'c','NodeColor','r')
```



Proverićemo da li su grafovi izomorfni

```
>> tf = isisomorphic(G1,G2)
>> tf = logical
   1
```

Odnosno da li su izomorfni u odnosu na osobinu "boja".

```
>> tf = isisomorphic(G1,G2,'NodeVariables','Color')
>> tf = logical
   0
```

Najmanje razapinjujuće stablo

Najmanje razapinjujuće stablo za graf G se dobija pozivanjem funkcije minspantree(G).

- `T = minspantree(G)`	- vraća najmanje razapinjujuće stablo za graf G.
- `T = minspantree(G,Name,Value)`	- formira najmanje razapinjujuće stablo Korišćenjem dodatnih uslova definisanim relacijom ime-osobina. Na primer, `minnspantree(G,'Method','sparse')` koristi Kruskalov algoritam za nalaženje minimalnog razapinjujućeg stable.
- `[T,pred] = minspantree(___)`	- pored minimalnog razapainjujućeg stable vraća i vektor **pred** koji sadrži informacije o tome koji čvor prethodi kom čvoru.

pred(I) predstavlja indeks čvora koji prethodi čvoru I.

Parametar ‘Type’ razlikuje sledeće slučajeve

Opcija	Opis
'tree'	Vraća se samo jedno stablo. Stablo sadrži koren.
'forest'	Vraća se šuma razapinjujućih stabala, tj. vraća se najmanje razapinjujuće stablo za svaku povezanu komponentu posebno.

Po konvenciji **pred**(rootNode)=0. Ukoliko je parameter Type postavljen na “tree”, tada je pred(I) = NaN za sve čvorove I koji nisu u istoj komponenti u kojoj je i njihov koren.

pred omogućava orijentisanost stablu s obzirom da su sve grane orijentisne suprotno od korena.

Po pravilu, koren je čvor koji je označen brojem 1.

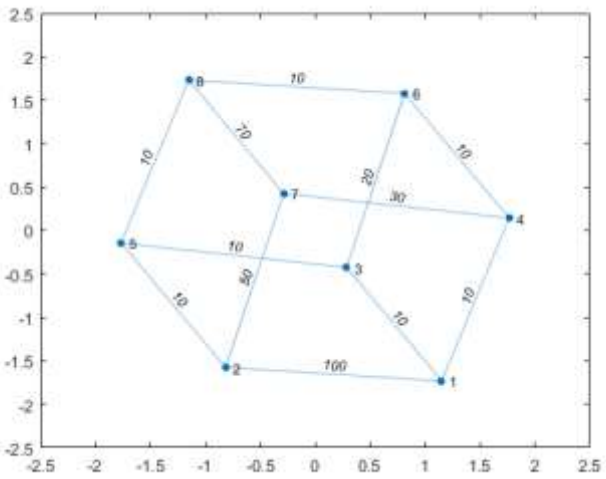
Opcija	Opis
'dense' (default)	Primov algoritam. Ovaj algoritam počevši od korena stabla dodaje grane grafa stablu tako da ne napravi cikl. Koren je prvi čvor.
'sparse'	Kruskalov algorithm. Ovaj algoritam sortira sve čvorove grafa po težini (u rastućem redosledu) a zatim dodaje granu po granu tako da ne napravi cikl. Koren se koristi samo prilikom formiranja vektora pred.

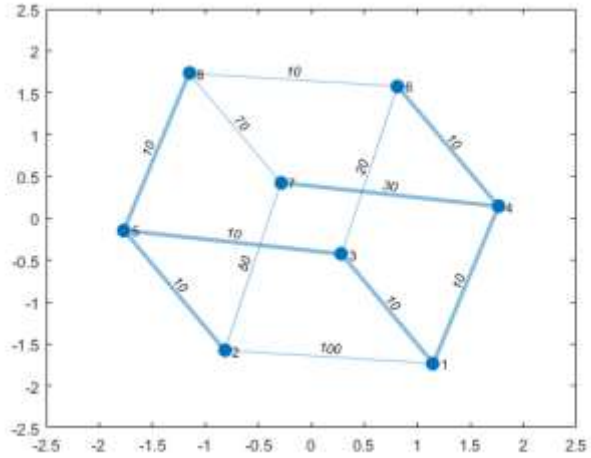
Primer1

Formirati graf sa težinskim granama, nacrtati graf i nacrtati njegovo najmanje razapinjujuće stablo.

Rešenje:

```
>> s = [1 1 1 2 5 3 6 4 7 8 8 8];
>> t = [2 3 4 5 3 6 4 7 2 6 7 5];
>> weights = [100 10 10 10 10 20 10 30 50
10 70 10];
>> G = graph(s,t,weights);
>> p = plot(G,'EdgeLabel',G.Edges.Weight);


>>[T,pred] = minspantree(G);
>> highlight(p,T)
```



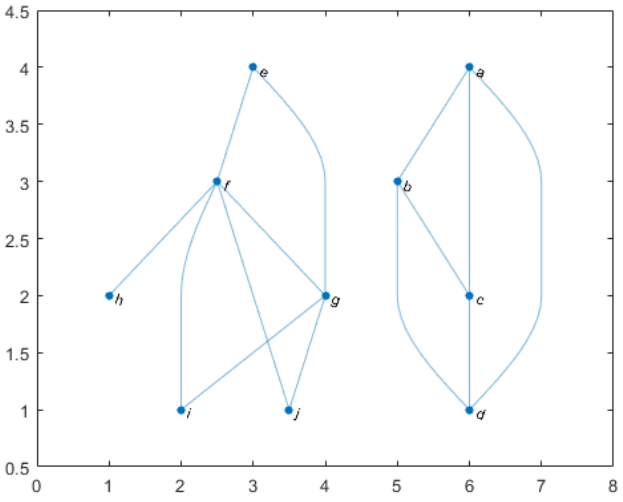


Primer2

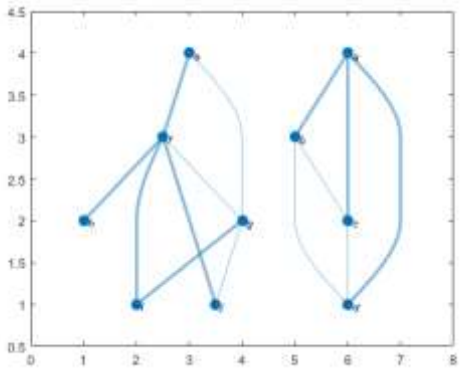
Formirati i nacrtati graf. Odrediti najmanju razapinjujuću šumu za graf počevši od čvora i. Podebljanim linijama označiti dobijeno stablo.

Rešenje:

```
>> s = {'a' 'a' 'a' 'b' 'b' 'c' 'e' 'e' 'f' 'f' 'f' 'f' 'g' 'g'};
>> t = {'b' 'c' 'd' 'c' 'd' 'd' 'f' 'g' 'g' 'h' 'i' 'j' 'i' 'j'};
>> G = graph(s,t);
>> p =
plot(G,'Layout','layered');
```

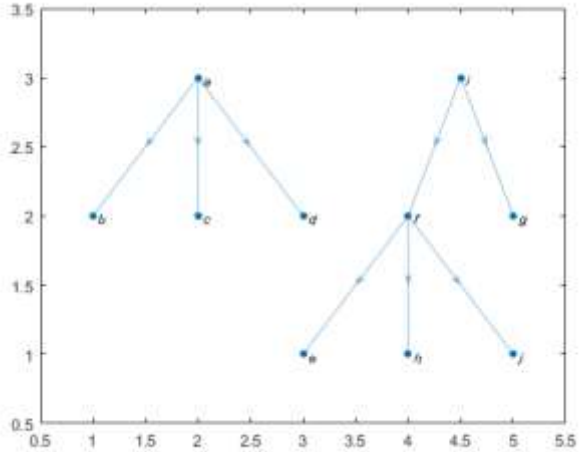



```
>> [T,pred] = minspantree(G,'Type','forest','Root',findnode(G,'i'));
>> highlight(p,T)
```



Korišćenjem vektora `pred` formirati orijentisano razapinjujuće stablo. Sve grane su usmerene u smeru od korena do lista.

```
>> rootedTree = digraph(pred(pred~=0),find(pred~=0),[],G.Nodes.Name);
>> plot(rootedTree)
```



Najkraće rastojanje između čvorova

path = shortestpath(G,s,t) - određuje najkraće rastojanje između čvorova s i t. Ako je graf definisan kao težinski, tj. G.Edges sadrži vrednosti za Weight, tada se te težine koriste kao rastojanja između čvorova. Inače se smatra da je rastojanje između čvorova dužine 1.

path = shortestpath(G,s,t,'Method',algorithm)
- definiše kojim se algoritmom određuje najkraće rastojanje. Na primer, ako je G težinski graf, funkcija

shortestpath(G,s,t,'Method','unweighted')

ignoriše težine i smatra da su rastojanja između čvorova jednaka 1.

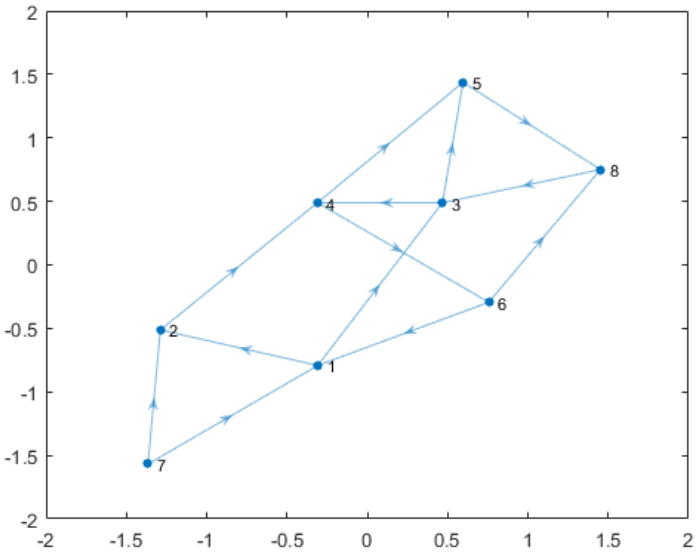
[path,d] = shortestpath(___)
- dodatno vraća dužinu najkraćeg rastojanja d

Primer1

Kreirati graf na osnovu vektora s i t, nacrtati ga a zatim odrediti najkraće rastojanje između određenih čvorova.

Rešenje:

```
s = [1 1 2 3 3 4 4 6 6 7 8 7 5];
t = [2 3 4 4 5 5 6 1 8 1 3 2 8];
G = digraph(s,t);
plot(G)
```



Najkraće rastojanje između 7 i 8:

```
path = shortestpath(G,7,8)

path =

     7     1     3     5     8
```

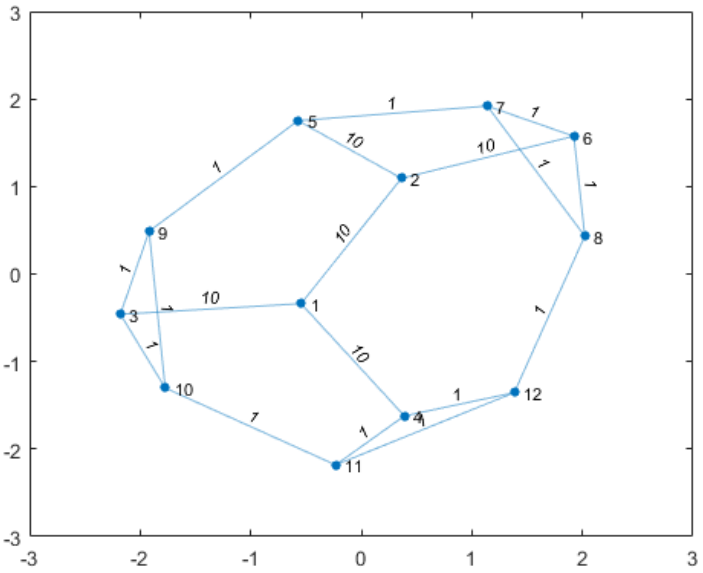
Primer2

Konstruisati i nacrtati težinski graf a zatim odrediti najkraće rastojanje između čvorova 3 i 8 (dužinu i putanju).

Rešenje:

```
s = [1 1 1 2 2 6 6 7 7 3 3 9 9 4 4 11 11 8];
t = [2 3 4 5 6 7 8 5 8 9 10 5 10 11 12 10 12 12];
weights = [10 10 10 10 10 1 1 1 1 1 1 1 1 1 1 1 1 1];
G = graph(s,t,weights);
plot(G,'EdgeLabel',G.Edges.Weight);
[path,d] = shortestpath(G,3,8)
```

```
path =

     3     9     5     7     8


d =

     4
```

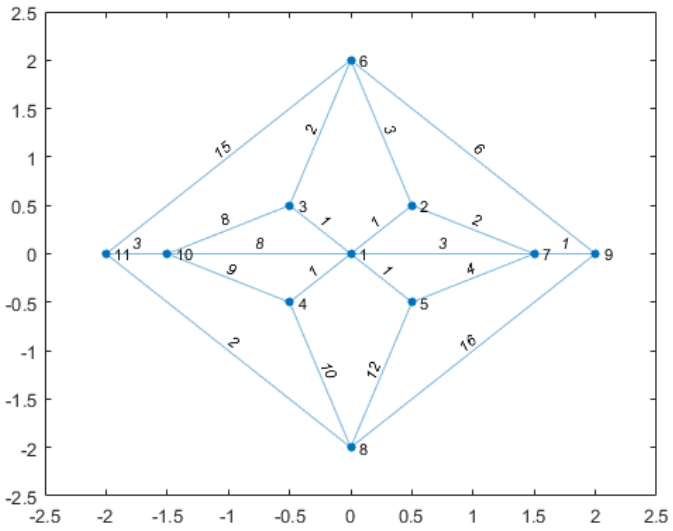


Primer3

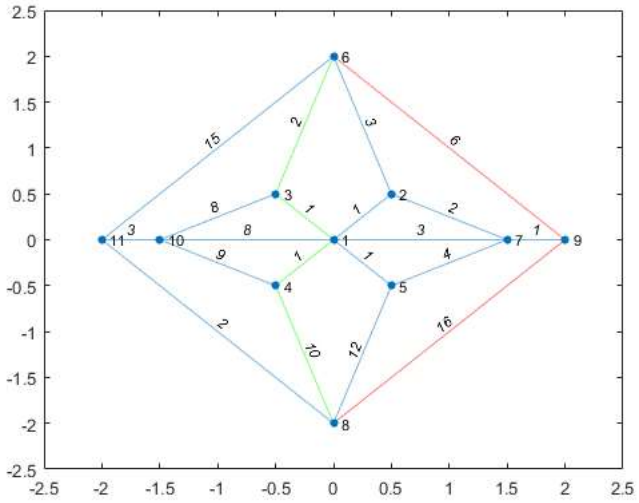
Konstruisati i nacrtati težinski graf a zatim odrediti najkraće rastojanje između čvorova 6 i 8 (dužinu i putanju) ignorišući težine. Najkraća rastojanja podebljai zelenom bojom.

Rešenje:

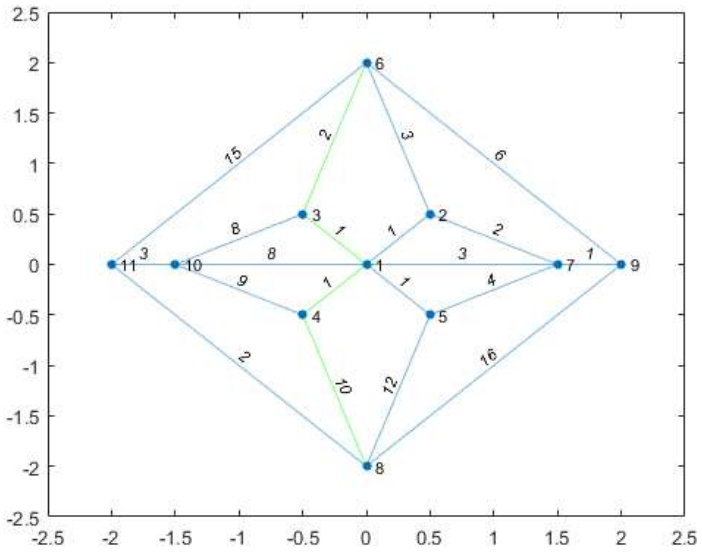
```
s = [1 1 1 1 1 2 2 7 7 9 3 3 1 4 10 8 4 5 6 8];
t = [2 3 4 5 7 6 7 5 9 6 6 10 10 10 11 11 8 8 11 9];
weights = [1 1 1 1 3 3 2 4 1 6 2 8 8 9 3 2 10 12 15 16];
G = graph(s,t,weights);

x = [0 0.5 -0.5 -0.5 0.5 0 1.5 0 2 -1.5 -2];
y = [0 0.5 0.5 -0.5 -0.5 2 0 -2 0 0 0];
p = plot(G,'XData',x,'YData',y,'EdgeLabel',G.Edges.Weight);
```



```
[path1,d] = shortestpath(G,6,8)

path1 =
     6     3     1     4     8
d =
    14
>> highlight(p,path1,'EdgeColor','g')
```




```
>> [path2,d] = shortestpath(G,6,8,'Method','unweighted')

path2 =
   6   9   8
d =
   2
>> highlight(p,path2,'EdgeColor','r')
```



Orijentisan graf se može konstruisati korišćenjem funkcije digraph.

```
path = shortestpath(G,s,t,'Method','acyclic')
```

- Funkcije `shortestpath`, `shortestpathtree` i `distances` ne podržavaju neorijentisane grafove sa negativnim težinama, odnosno ni jedan graf koji ima negativni cikl iz sledećih razloga:

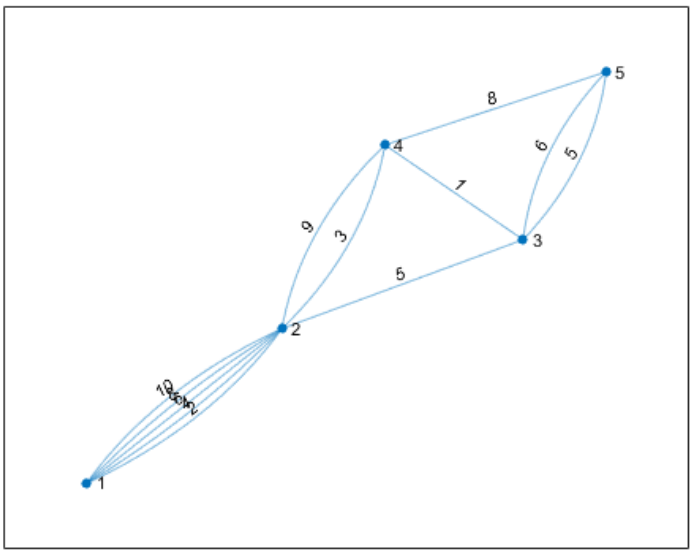
Option	Description
`'auto'` (default)	Opccija `'auto'` automatski bira algoritam za određivanje najkraćeg rastojanja. `'unweighted'` se koristi za bestežinske grafove i digrafove `'positive'` se koristi kod grafova koji imaju nenegativne težine. `'mixed'` se koristi kod digrafova kod kojih težina može imati negativne vrednosti ali grafovi ne smeju imati negativne cikle.
`'unweighted'`	Pretraga u dubinu koja smatra da su sve grane težine 1..
`'positive'`	Algoritam Dijkstre kod kojih sve težine moraju biti nenegativne
`'mixed'` (samo kod `digraph-a`)	Bellman-Ford-ov algoritam za orijentisane grafove koji zahtevaju da grafovi nemaju cikle negativnih dužine. `'mixed'` je sporiji od `'positive'` za isti problem zato što `'mixed'` dozvoljava grane negativne težine.
`'acyclic'` (samo kod `digraph-a`)	Koristi se zaorijentisane aciklične težinske grafove. `isdag` proverava da li je orijentisan graf acikličan..
Napomena: Za veći broj grafova, `'unweighted'` je najbrži, prate ga `'acyclic'`, `'positive' i 'mixed'`.	

Primer4

Kreirati težinski graf sa 5 čvorova tako da neki čvorovi budu povezani preko više od jedne grane. a zatim odrediti i nacrtati najkraće rastojanje između dva čvora u multigrafu. Označiti te grane intenzivnijom bojom.

Rešenje:

```
G = graph([1 1 1 1 1 2 2 3 3 3 4 4],[2 2 2 2 2 3 4 4 5 5 5 2],[2 4 6 8 10 5 3 1 5 6 8 9]);
p = plot(G,'EdgeLabel',G.Edges.Weight);
```



Odrediti najkraće rastojanje između čvorova 1 i 5. Budući da postoji više grana između ovih čvorova, tražiti od funkcije da vrati tri parametra: P, d i edgepath:

```
>> [P,d,edgepath] = shortestpath(G,1,5)
```

```
P = 1×5

     1     2     4     3     5

d = 11

edgepath = 1×4

     1     7     9    10
```

Dakle, najkraći put je dužine 11 i void preko sledećih čvorova:

```
>> G.Edges(edgepath,:)

ans=4×2 table
    EndNodes    Weight
    ________    ______

     1    2        2
     2    4        3
     3    4        1
     3    5        5
```

Najkraći put prikazujemo podebljanim linijama:.

```
>> highlight(p,'Edges',edgepath)
```