SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHINKE I RAČUNARSTVA

SEMINAR

MREŽE RAČUNALA

TEORIJA GRAFOVA

MLADEN MARINOVIĆ
0036378799

Sadržaj

1. Uvod 1

2. Osnovni pojmovi u teoriji grafova 2

2.1 Primjeri 4

2.2 Zadaci 5

3. Razapinjajuća stabla 6

3.1 Primov algoritam 6

3.2 Kruskalov algoritam 9

3.3 Primjeri 11

3.4 Zadaci 13

4. Najkraće udaljenosti u grafu 15

4.1 Dijkstrin algoritam za promalaženje najkraćih puteva 15

4.2 Floydov algoritam za pronalaženje najkraćih puteva 15

4.3 Zadaci 15

5. Maksimalni protok u mreži 15

5.1 Ford-Fulkersonov algoritam 15

5.2 Zadaci 15

6. Rješenja zadataka 15

6.1 Osnovni pojmovi u teoriji grafova 15

6.2 Razapinjajuća stabla 15

6.3 Najkraće udaljenosti u grafu 15

6.4 Maksimalni protok u mreži 15

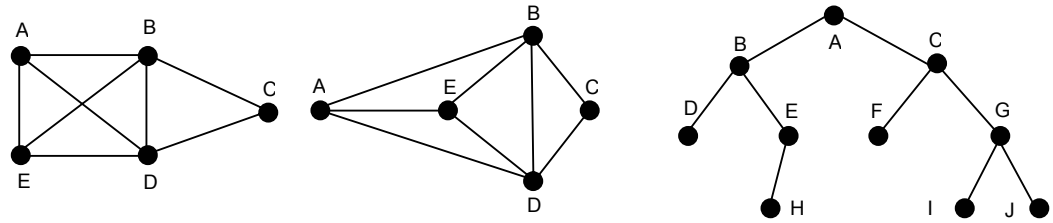
7. Literatura 15

1. Uvod

Ovaj seminar iz Mreža računala bavi se teorijom grafova. Teorija grafova jedna je od grana matematike koja nalazi veliku primjenu na području mreža računala, primjerice na područjima algoritmama usmjeravenja, traženja puteva kroz mrežu te opisivanju topologije mreže. Seminar je zamišljen kao uvod u teoriju grafova kako bi se lakše savladalo gradivo iz predmeta Mreže računala vezano uz teoriju grafova. U narednim poglavljima bit će objašnjeni osnovni pojmovi i definicije vezane uz teoriju grafova, najvažniji algoritmi koji se javljaju u primjeni, a isti će biti i detaljno objašnjeni na primjerima. Kako bi ovaj seminar poslužio za savladavanje tog važnog područja, nakon svakog poglavlja dani su i zadaci za vježbu kako bi se provjerilo gradivo iznešeno u istom poglavlju.

2. Osnovni pojmovi u teoriji grafova

Graf je matematički objekt čiju ćemo definiciju navesti malo kasnije u ovom poglavlju, a za sada dali bismo nekoliko primjera grafova kako bi se s istima lakše upoznali.



a) b) c)

Slika 1. Primjeri grafova

Kako je sa slike 1 vidljivo graf se sastoji od vrhova (na slici 1 označeni točkama A, B, C itd.) i bridova koji te vrhove povezuju. Grafovi su nam posebno zanimljivi budući da njima možemo modelirati složene probleme veoma jednostavno, npr. predstavljanje prometnica u jednoj državi, predstavljanje električnih mreža te mreža računala i sl.

Formalna definicija grafa je:

Definicija: *Jednostavan graf G sastoji se od nepraznog konačnog skupa skupa V(G) čije elemente zovemo vrhovi i konačnog skupa E(G) različitih parova elemenata V(G) koje zovemo bridovi.*

Uobičajeno je u definicijama i teoremima u teoriji grafova vrhove označavati malim slovima *u* i *v*, a bridove malim slovim *e* i *f*. Pri tome *e={u,v}* označava brid koji spaja vrhove *u* i *v*, a to možemo kraće zapisati kao *e=uv*. Često se bridu u grafu pridružuje još jedan broj koji predstavlja njegovu težinu, a taj se dodatni podatak najčešće koristi pri modeliranju problema grafom. U našim razmatranjima promatrat ćemo samo jedan graf odjednom pa nema potrebe posebno naglašavati u oznaci kojem grafu pojednini skup vrhova ili bridova pripada, zato ćemo graf označavati oznakom G(V,E). Radi jednostavnosti za oznaku vrhova na primjerima koristimo velika slova A,B,C itd. Skupovi V i E za primjer sa slike 1.a su:

V={A,B,C,D,E)
E={AB,AD,AE,BC,BD,BE,CD,DE}

Kako bi se bolje razumjeli algoritmi koje ćemo u narednim poglavljima prikazivati potrebno je definirati još nekoliko pojmova vezanih uz grafove, a to su prvenstveno susjednost vrhova, susjednost bridova, put u grafu i povezanost grafa.

Definicija: *Za vrhove u i v kažemo da su susjedni ako postoji brid e=uv u tom grafu koji ih spaja. Za bridove e i f kažemo da su susjedni ako postoji vrh u u tom grafu koji je njima zajednički.*

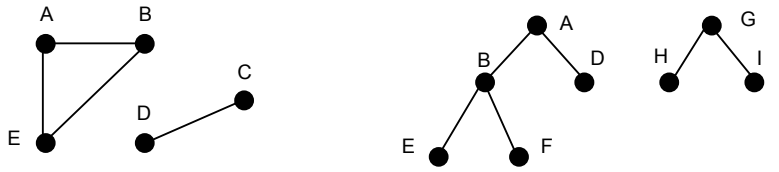
Na primjeru sa slike 1.a možemo uočiti da su vrhovi A i B susjedni, isto tako B i C su susjedni itd. Susjedni bridovi na tom istom grafu su npr. AB i AE jer im je vrh A zajednički.

Definicija: *Put u grafu G je konačan slijed bridova $v_1v_2, v_2v_3, \ldots, v_{n-1}v_n$ u kojem su svaka dva brida susjedna i svi su vrhovi različiti, osim eventualno početni i krajnji. Put možemo označavati i kao $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \ldots \rightarrow v_n$.*

Definicija: *Za graf G kažemo da je povezan onda i samo onda ako postoji put između svaka dva vrha.*

Put u primjeru sa slike 1.a je npr. A → B → C → D → E

Svi su grafovi na slici 1 povezani jer postoji put od svakog vrha do svih ostalih vrhova u tim grafovima. Primjer nepovezanog grafa dan je na slici 2.



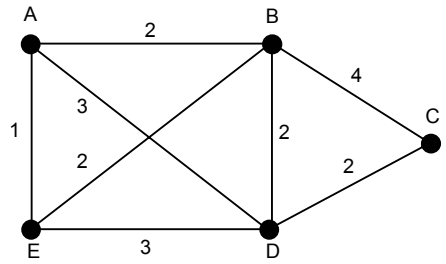
a) b)

Slika 2. Primjeri nepovezanih grafova

Kao što je sa slike 2.a vidljivo ne postoji nikakav slijed bridova koji bi činio put od vrha D do vrha A.

Najjednostavniji način prikazivanj grafa je tablicom incidencije. To je tablica u kojoj redovi i stupci predstavljaju vrhove grafa, a polje u tablici na križanju retka *i* i stupca *j* predstavlja težinu brida koji spaja vrhove *i* i *j*. Težinu brida najčešće koristimo kao mjeru udaljenosti dva vrha. Za vrhove koji nisu spojeni bridom podrazumjeva se udaljenost ∞, ali se taj zapis u tablici incidencije ostavlja praznim.

Budući da na niti jednom grafu do sada nismo označili udaljenosti između vrhova na slici 3 dan je graf sa slike 1.a na kojemu smo te udaljenosti označili kako bismo prikazali kako za taj isti graf izgleda tablica incidencije.



Slika 3. Primjer grafa s težinama pridjeljenim bridovima

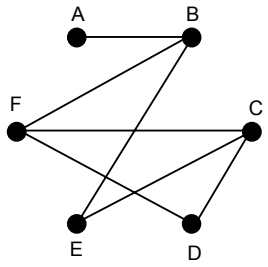
Za graf sa slike 3 pripadna tablica incidencije glasi:

	A	B	C	D	E
A	0	2		3	1
B	2	0	4	2	2
C		4	0	2	
D	3	2	2	0	3
E	1	2		3	0

Važno je uočiti da je tablica incidencije za ovakvu vrstu grafa simetrična s obzirmo na glavnu dijagonalu. Treba primjetiti da se na glavnoj dijagonali pojavljuju samo nule, a to znači da u ovakvom prikazu uzimamo da nam je udaljenost vrha od samog sebe jedanka nuli, što ne mora biti u svakom modelu kojeg grafom opisujemo.

2.1 **Primjeri**

P2.1: Za graf G zadan slikom napisati skupove V i E.



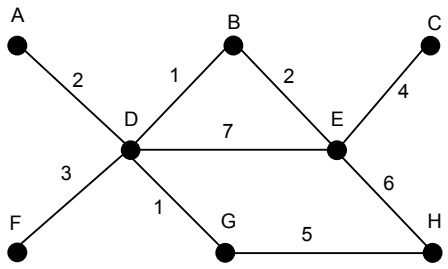
Rješenje:
V={A,B,C,D,E,F}
E={AB,BE,BF,CD,CE,CF,DF}

P2.2: Za graf iz prethodnog primjera napisati tablicu incidencije, pri tome uzeti da su težine bridova jedanke 1.

Rješenje:

	A	B	C	D	E	F
A	0	1				
B	1	0			1	1
C			0	1	1	1
D			1	0		1
E		1	1		0	
F		1	1	1		0

P2.3: Za graf zadan slikom napisati tablicu incidencije.

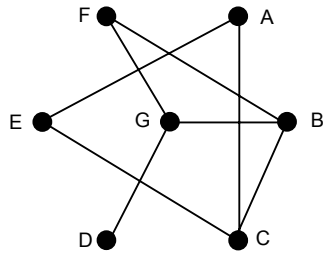


Rješenje:

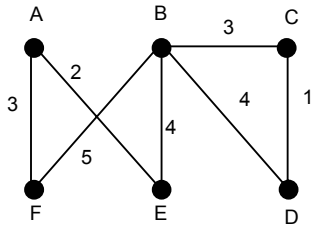
	A	B	C	D	E	F	G	H
A	0			2				
B		0		1	2			
C			0		4			
D	2	1		0	7	3	1	
E		2	4	7	0			6
F				3		0		
G				1			0	5
H					6		5	0

2.2 **Zadaci**

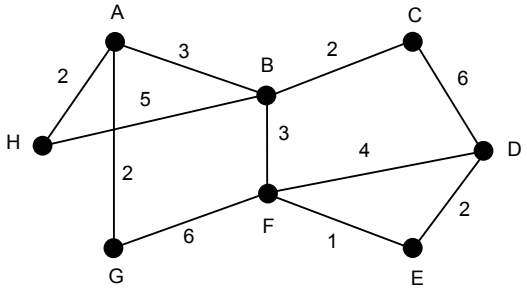
Z2.1: Za graf zadan slikom napisati skupove V i E.



Z2.2: Za graf zadan slikom napisati tablicu incidencije.

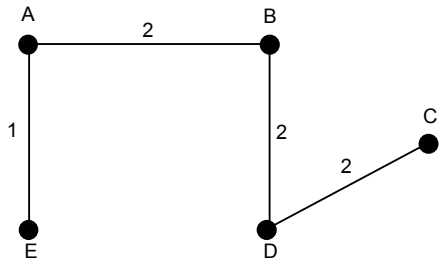


Z2.3: Za graf zadan slikom napisati tablicu incidencije.



3. Razapinjajuća stabla

Kako bi razumjeli što je razapinjajuće stablo poslužimo se primjerom sa slike 1.a. Graf na toj slici posjeduje cilkuse, tj. puteve u grafu koji započinju i završavaju istim vrhom. Ukoliko bismo uklonili dovoljan broj bridova iz grafa dobili bismo graf koji nema ciklusa te takav graf zovemo stablo. Primjer stabla možemo vidjeti na slici 1.c. Stablo koje dobivamo uklanjanjem određenog broja bridova iz grafa, a da pritim dobiveni graf ostane povezan, zovemo razapinjajuće stablo.



Slika 4. Minimalno razapinjajuće stablo

Općenito konstruiranje razapinjajućeg stabla jednostavan je postupak, ali u primjeni je poželjno da takvo razapinjajuće stablo ima i neka dodatna svojstva, kao što je minimalna/maksimalna suma težina bridova. Graf na slici 4. predstavlja minimalno razapinjajuće stablo dobiveno iz grafa na slici 3. Za dobivanje takvih razapinjajućih stabla postoje dva poznata algoritma koja će biti opisana u nastavku.

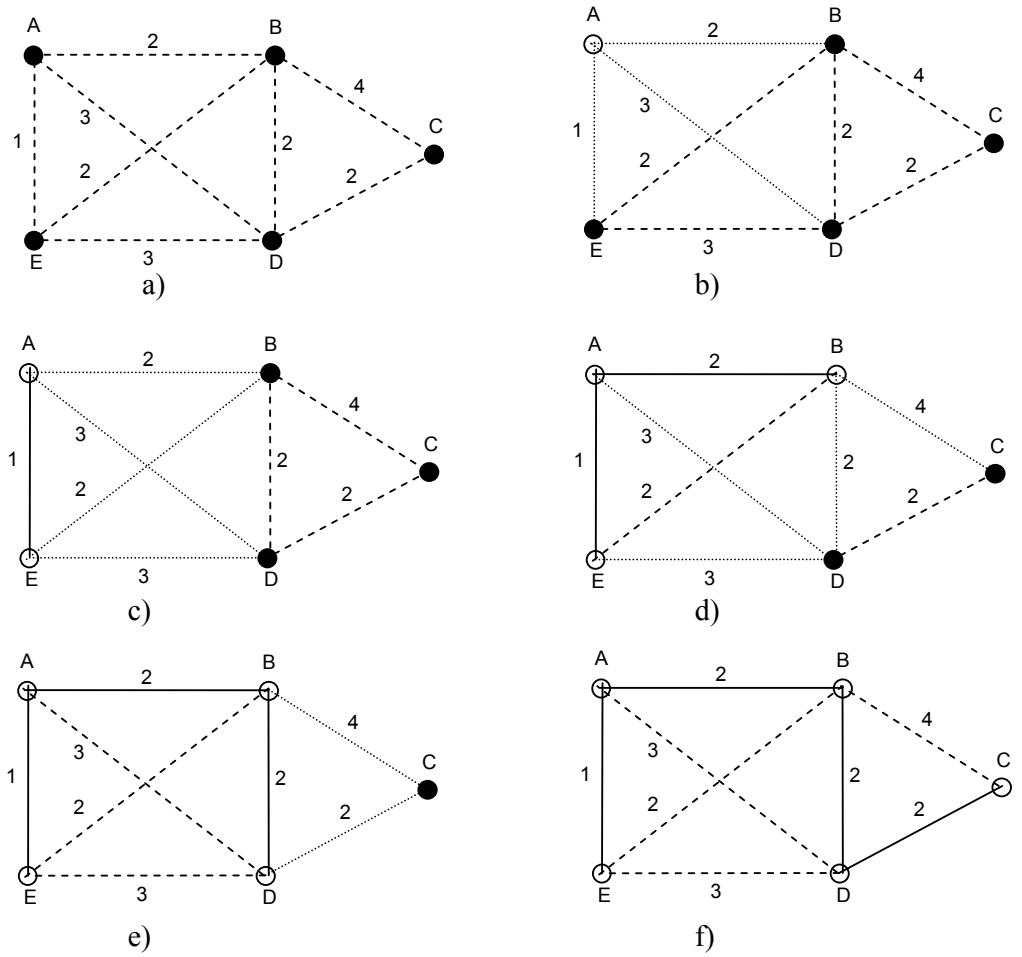
3.1 Primov algoritam

Dobivanje razapinjajućeg stabla iz zadanog grafa Primovim algoritmom možemo prikazati sljedećim pseudokodom:

```
Stablo = ∅
Izaberi proizvoljni vrh iz V(Graf) i stavi ga u V(Stablo).
Dok je Broj_Vrhova(Stablo) < Broj_Vrhova(Graf) ponavljaj
      Izaberi vrh koji nije u V(Stablo) a susjedan je nekom vrhu iz
      V(Stablo) i pri tome je težina brida koja ih spaja minimalna.
      Stavi taj vrh zajedno s njemu pripadajućim bridom u Stablo
```

Ovim algoritmom pokušavamo iz zadanog grafa, `Graf`, izgraditi razapinjajuće stablo, `Stablo`. U početku je `Stablo` prazno te ga počinjemo graditi dodavanjem proizvoljnog vrha iz skupa vrhova početnog grafa. Postupak nastavljamo dodavanjem brida u `Stablo` koji ima svojstvo da povezuje jedan vrh koji se već nalazi u `Stablo` i jedan koji se se u njemu ne nalazi, pazeći pri tome da je težina tog brida minimalna

Na kraju algoritma graf `Stablo` nam predstavlja traženo minimalno razapinjajuće stablo. Prikažimo sad izvođenje Primovog algoritma na grafu sa slike 3 u svim koracima algoritma. Bridovi početnog grafa bit će označeni crtkanom linijom, a oni dodani u razapinjajuće stablo bit će označeni punim linijom, pri tome ćemo bridove koje u danom koraku promatramo označiti točkastom linijom. Izvođenje algoritma započet ćemo od vrha A.



Slika 5. Prikaz izvođenja Primovog algoritma

Kako bismo lakše proveli taj algoritam koristimo se tabličnim zapisom. Prikažimo taj postupak na istom primjeru. Za graf prvo treba napisati tablicu incidencije te u njoj u prvom stupcu tablice zaokružimo oznaku vrha kojim započinjemo i prekrižimo sve brojeve u stupcu pridjeljenim tom vrhu, kao što je prikazano u tablici incidencije pod a). Tada zapišemo prvi korak algoritma u tablicu za traženje razapinjajučeg stabla, kao što je prikazano u tabličnom zapisu pod b). U prvom stupcu zapišemo ime vrha koji se nalazi u razapinjajućem stablu, u drugom stupcu zapišemo ime vrha koji nije u razapinjajućem stablu, a susjedan je vrhu iz prvog stupca. Treći stupac sadrži težinu brida koji ta dva vrha spaja. Kako bi se uštedjelo u pisanju za svaki vrh iz razapinjajućeg stabla u tablicu zapišemo samo najbliže susjedne vrhove koji nisu u razapinjajućem stablu, to lako vidimo pogledamo li u redak tablice incidencije pridruženom tom vrhu. Ukoliko postoji više vrhova s tim svojstvom upišemo ih sve. Nakon što smo popisali sve vrhove izaberemo vrh koji nije u stablu i čiji je pripadni brid najmanje težine te ga dodamo u stablo. Ime tog vrha zapišemo u četvrti stupac tablice, a u peti stupac zapišemo dužinu pripadnog brida te u šesti ime pripadnog brida. Nakon što smo to učinili prekrižimo sve brojeve u stupcu tablice incidencije koji pripada tom vrhu kako ih više nebismo koristili u računu i zaokružimo ime brida u njemu pripadnom retku matrice incidencije, kao što je prikazano u tablici incidencije pod c). U sljedećim tablicama dani su svi koraci Primovog algoritma za prethodni primjer.

Tablični zapis Primovog algoritma

a)

	A	B	C	D	E
(A)	~~0~~	2		3	1
B	~~2~~	0	4	2	2
C		4	0	2	
D	~~3~~	2	2	0	3
E	~~1~~	2		3	0

b)

1	A	E	1	E	1	AE

c)

	A	B	C	D	E
(A)	~~0~~	2		3	~~1~~
B	~~2~~	0	4	2	~~2~~
C		4	0	2	
D	~~3~~	2	2	0	~~3~~
(E)	~~1~~	2		3	~~0~~

d)

1	A	E	1	E	1	AE
2	A	B	2	B	2	AB
	E	B	2			

e)

	A	B	C	D	E
(A)	~~0~~	~~2~~		3	~~1~~
(B)	~~2~~	~~0~~	4	2	~~2~~
C		~~4~~	0	2	
D	~~3~~	~~2~~	2	0	~~3~~
(E)	~~1~~	~~2~~		3	~~0~~

f)

1	A	E	1	E	1	AE
2	A	B	2	B	2	AB
	E	B	2			
3	A	D	3	D	2	B
	B	D	2			
	E	D	3			

g)

	A	B	C	D	E
(A)	~~0~~	~~2~~		~~3~~	~~1~~
(B)	~~2~~	~~0~~	4	~~2~~	~~2~~
C		~~4~~	0	~~2~~	
(D)	~~3~~	~~2~~	2	~~0~~	~~3~~
(E)	~~1~~	~~2~~		~~3~~	~~0~~

h)

1	A	E	1	E	1	AE
2	A	B	2	B	2	AB
	E	B	2			
3	A	D	3	D	2	BD
	B	D	2			
	E	D	3			
4	B	C	4	C	2	DC
	D	C	2			

Kao što je iz tablica vidljivo, dodali smo još jedan stupac na početku tablice kako bi se vidjelo koji dio tablice pripada kojem koraku algoritma.

U prvom koraku, na slici 5.b, promatramo bridove AB, AD i AE. U tabličnom zapisu zapisujemo samo najkraći brid koji povezuje vrh A s nekim drugim vrhom, tj. brid AE i to je prikazano u tabličnom zapisu pod b). Brid AE najkraći je od svih bridova zapisanih u tablicu te ga zato dodajemo u razapinjajuće stablo. U tablici incidencije zaokružimo oznaku vrha E te prekržimo sve brojeve u stupcu matrice incidencije pridružene tom vrhu, kao što je prikazano u tablici incidencije c). U drugom koraku, na slici 5.c, promatramo bridove AB, AD, BE i DE. U tabličnom zapisu međutim zapišemo samo AB i BE budući da su oni najkraći, te izaberemo brid AB kao najkraći, kao što je prikazano u tabličnom zapisu pod d). U tablici incidencije zaokružimo oznaku vrha B te prekržimo sve brojeve u stupcu matrice incidencije pridružene tom vrhu, kao što je prikazano u tablici incidencije e). U trećem koraku promatramo bridove AD, BD, BC i DE dok u tablicu zapišemo bridove AD, BD i ED. Važno je uočiti da se brid BE više ne uzima u obzir jer bi njegovo dodavanje stvorilo ciklus. Najkraći je brid BD i njega dodajemo u razapinjajuće stablo, kao što je prikazano u tabličnom zapisu pod f). U tablici incidencije zaokružimo oznaku vrha D te prekržimo sve brojeve u stupcu matrice incidencije pridružene tom vrhu, kao što je prikazano u tablici incidencije g). U posljednjem koraku biramo između bridova BC i CD te izaberemo CD kao najkraći, kao što je vidljivo iz

tabličnog zapisa pod h). Bridove AD i ED ne promatramo jer bi njihovim dodavanjem napravili ciklus.

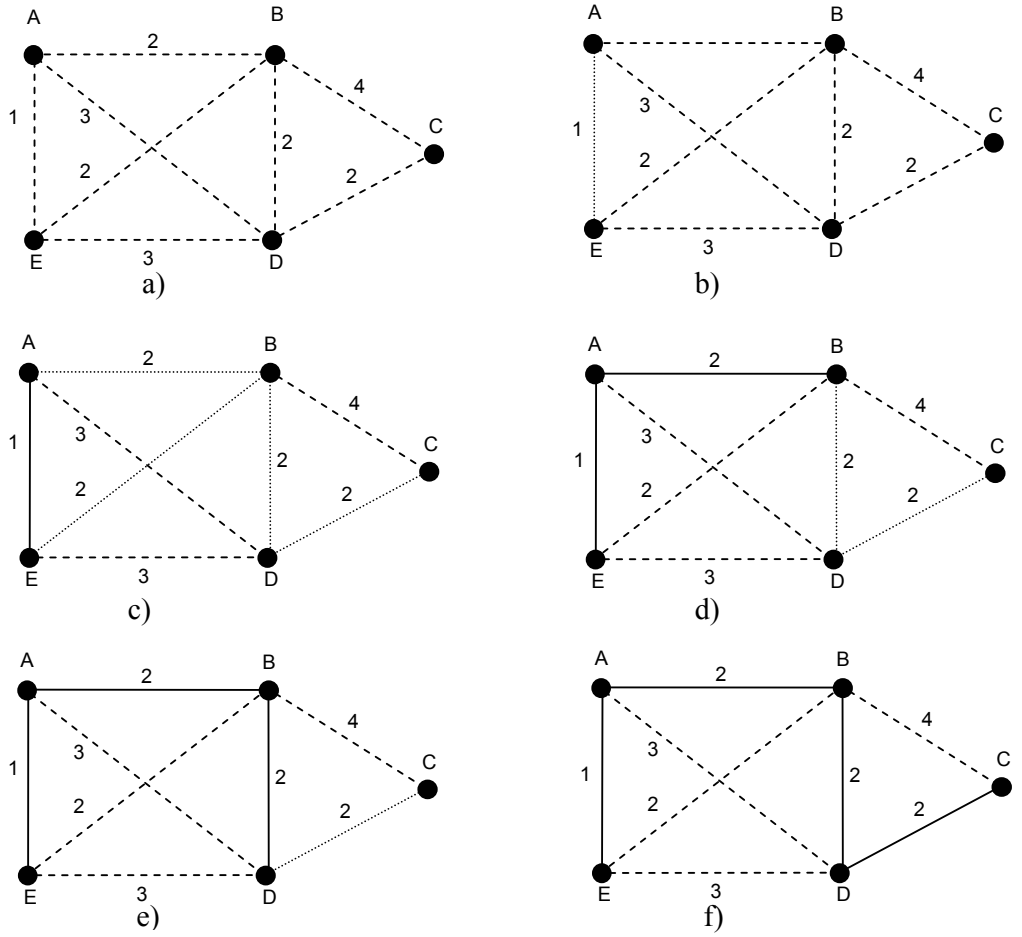
3.2 **Kruskalov algoritam**

Dobivanje razapinjajućeg stabla iz zadanog grafa Kruskalovim algoritmom možemo prikazati sljedećim pseudokodom:

```
Stablo = ∅
Bridove grafa poredaj u padajući niz.
Dok postoji brid čije dodavanje u Stablo ne tvori ciklus ponavljaj
      Uzmi najmanji brid koji ne tvori ciklus s bridovima iz Stabla i
      dodaj ga u Stablo.
```

Kao kod Primovog algoritma započinjemo sa praznim razapinjajućim stablom, `Stablo`, te dodajemo bridove iz početnog grafa, `Graf`, u razapinjajuće stablo. U svakom koraku dodajemo po jedan brid u `Stablo` i to tako da je dužina brida kojeg dodajemo najmanja i da pritom dodani brid ne tvori ciklus s ostalim bridovima u `Stablo`. Algoritam se zaustavlja kad u Stablo ne možemo dodati više brid koji ne bi tvorio ciklus. Na kraju algoritma `Stablo` nam predstavlja razapinjajuće stablo.

Prikažimo sad izvođenje Kruskalovog algoritma na prethodnom primjeru.



Slika 7. Prikaz izvođenja Kruskalovog algoritma

Za razliku od Primovog algoritma, razapinjajuće stablo nam ne mora biti povezano za vrijeme izvođenja algoritma.

U prvom koraku prikazanom na slici 7.b u obzir dolazi samo brid AE budući da je njegova težina najmanja. U drugom koraku, na slici 7.c, u obzir dolaze bridovi AB, BD, BE i CD, pa ćemo uzeti bilo koji od njih, recimo AB. U trećem koraku, na slici 7.d, preostaju nam samo bridovi BD i CD te uzmemo brid BD. Važno je uočiti da brid BE ne možemo staviti u stablo jer bismo tako napravili ciklus. U četvrtom koraku, na slici 7.e, preostaje nam samo brid CD te njegovim stavljanjem u stablo završavamo gradnju istog.

Isti postupak možemo provesti i bez crtanja grafa, jednostavnim poznavanjem samo bridova grafa i njihovih težina. Primjenimo sad taj postupak na istom zadatku. Prvo poredamo sve bridove u uzlazni niz po njihovim težinama:

AE-1, AB-2 ,BD-2, BE-2, CD-2, AD-3, DE-3, BC-4

Kako bismo vidjeli da li dodavanje nekog brida stablu čini ciklus trebamo pamtiti samo skupove vrhova koji nam predstavljaji stablo. U početku imamo pet jednočlanih skupova od kojih svaki sadrži jedan vrh. Zbog jednostavnosti zapisa takve jednočlane skupove nećemo zapisivati u postupku. U prvom koraku u graf dodajemo brid s najmanjom težinom i njega izbacimo iz prethodnog niza bridova te to zapišemo ovako:

{A,E} {AE} AB-2, BD-2 ,BE-2, CD-2, AD-3, DE-3, BC-4

U prvom stupcu imamo skupove vrhova koji nam predstavlaju stablo, u drugom stupcu nalaze se bridovi koji čine stablo, a u trečem stupcu imamo bridove koje još možemo ubaciti u stablo. U sljedećem koraku ubacujemo sljedeći brid u stablo, brid AB. Budući da se oznaka vrha B i vrha A ne nalaze u istom skupu bridova u prvom stupcu, ubacivenjem tog brida nećemo stvoriti ciklus. Zapis toga sad izgleda:

{A,B,E} {AB, AE} BD-2 ,BE-2, CD-2 ,AD-3, DE-3, BC-4

U sljedećem koraku ubacujemo brid BD jer isti ne čini ciklus s već ubačenim bridovima. Zapis toga je:

{A,B,D,E} {AB,AE,BD} BE-2,CD-2,AD-3,DE-3,BC-4

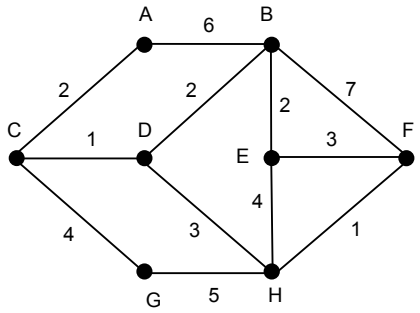
U narednom koraku ne možemo ubaciti brid BE jer se oznake vrhova B i E nalaze u istom skupu vrhova u prvom stupcu. To znači da u stablu već postoji put koji povezuje vrhove B i E te ukoliko bismo dodali još jedan brid koji ih povezuje direktno stvorili bismo ciklus. Sljedeći brid koji dolazu u obzir je CD te njegovim ubacivenjem ne stvaramo ciklus u stablu. Zapis nakon tog koraka je:

{A,B,C,D,E} {AB,AE,BD,CD} AD-3,DE-3,BC-4

Budući da smo u stablo ubacili maksimalan broj bridova algoritam završava. Važno je uočiti da smo u svakom koraku brid koji smo ubacivali birali proizvoljno. Od drugog koraka nadalje mogli smo ubaciti u stablo bilo koji brid dužine 2 i time nebismo narušili minimalnost rješenja jer se od jednog grafa mogu napraviti više različitih minimalnih razapinjajućih stabla, tj. jednostavnije rečeno, rješenje općenito nije jednistveno.

3.3 **Primjeri**

P3.1: Za graf zadan slikom pronaći minimalno razapinjajuće stablo Primovim algoritmom.



Rješenje:

Prvo napišemo tablicu incidencije a zatim počnemo od proizvoljnog vrha, u ovom slučaju od vrha C.

	A	B	C	D	E	F	G	H
A	0	6	~~2~~					
B	6	0		2	2	7		
Ⓒ	2		~~0~~	1			4	
D		2	~~1~~	0				3
E		2			0	3		4
F		7			3	0		1
G			~~4~~				0	5
H				3	4	1	5	0

1	C	D	1	D	1	CD

	A	B	C	D	E	F	G	H
A	0	6	~~2~~					
B	6	0		~~2~~	2	7		
Ⓒ	2		~~0~~	~~1~~			4	
Ⓓ		2	~~1~~	~~0~~				3
E		2			0	3		4
F		7			3	0		1
G			~~4~~				0	5
H				~~3~~	4	1	5	0

2	C	A	2	A	2	AC
	D	B	2			

	~~A~~	B	C	D	E	F	G	H
Ⓐ	~~0~~	6	~~2~~					
B	~~6~~	0		~~2~~	2	7		
Ⓒ	~~2~~		~~0~~	~~1~~			4	
Ⓓ		2	~~1~~	~~0~~				3
E		2			0	3		4
F		7			3	0		1
G			~~4~~				0	5
H				~~3~~	4	1	5	0

3	A	B	6	B	2	BD
	C	G	4			
	D	B	2			

	A	B	C	D	E	F	G	H
(A)	~~0~~	~~6~~	~~2~~					
(B)	~~6~~	~~0~~		~~2~~	2	7		
(C)	~~2~~		~~0~~	~~1~~			4	
(D)		~~2~~	~~1~~	~~0~~				3
E		~~2~~			0	3		4
F		~~7~~			3	0		1
G			~~4~~				0	5
H				~~3~~	4	1	5	0

4	B	E	2	E	2	BE
	C	G	4			
	D	H	3			

	A	B	C	D	E	F	G	H
(A)	~~0~~	~~6~~	~~2~~					
(B)	~~6~~	~~0~~		~~2~~	~~2~~	7		
(C)	~~2~~		~~0~~	~~1~~			4	
(D)		~~2~~	~~1~~	~~0~~				3
(E)		~~2~~			~~0~~	3		4
F		~~7~~			~~3~~	0		1
G			~~4~~				0	5
H				~~3~~	~~4~~	1	5	0

5	B	F	7	H	3	DH
	C	G	4			
	D	H	3			
	E	F	3			

	A	B	C	D	E	F	G	H
(A)	~~0~~	~~6~~	~~2~~					
(B)	~~6~~	~~0~~		~~2~~	~~2~~	7		
(C)	~~2~~		~~0~~	~~1~~			4	
(D)		~~2~~	~~1~~	~~0~~				~~3~~
(E)		~~2~~			~~0~~	3		~~4~~
F		~~7~~			~~3~~	0		~~1~~
G			~~4~~				0	~~5~~
(H)				~~3~~	~~4~~	1	5	~~0~~

6	B	F	7	F	1	FH
	C	G	4			
	E	F	3			
	H	F	1			

	A	B	C	D	E	F	G	H
(A)	~~0~~	~~6~~	~~2~~					
(B)	~~6~~	~~0~~		~~2~~	~~2~~	~~7~~		
(C)	~~2~~		~~0~~	~~1~~			4	
(D)		~~2~~	~~1~~	~~0~~				~~3~~
(E)		~~2~~			~~0~~	~~3~~		~~4~~
(F)		~~7~~			~~3~~	~~0~~		~~1~~
G			~~4~~				0	~~5~~
(H)				~~3~~	~~4~~	~~1~~	5	~~0~~

7	C	G	4	G	4	CG
	H	G	5			

P3.2: Pronaći minimalno razapinjajuće stablo Kruskalovim algoritam za graf iz prethodnog primjera.

Rješenje:
Prvo poredamo bridove u rastući slijed po njihovim težinama:

CD-1,FH-1,AC-2,BD-2,BE-2, DH-3,EF-3, CG-4,EH-4, GH-5,AB-6,BF-7

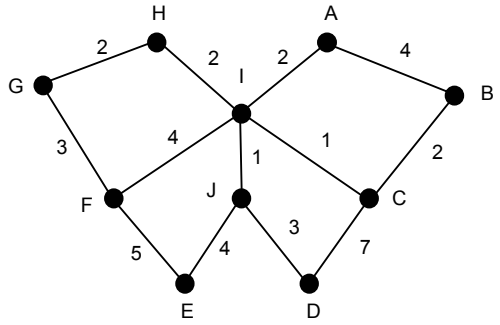
Nako što smo to učinili možemo započeti sa umetanjem bridova u stablo.

{C,D}	{CD}	FH-1, AC-2, BD-2, BE-2, DH-3, EF-3, CG-4, EH-4, GH-5, AB-6, BF-7
{C,D} {F,H}	{CD,FH}	AC-2, BD-2, BE-2, DH-3, EF-3, CG-4, EH-4, GH-5, AB-6, BF-7

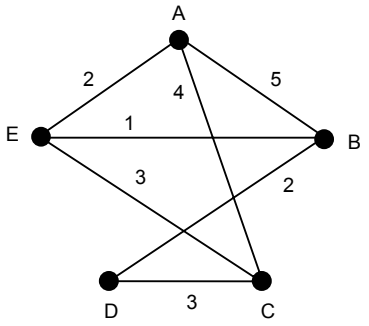
{A,C,D}	{F,H}	{AC,CD,FH}	BD-2, BE-2, DH-3, EF-3, CG-4, EH-4, GH-5, AB-6, BF-7
{A,B,C,D}	{F,H}	{AC,BD,CD,FH}	BE-2, DH-3, EF-3, CG-4, EH-4, GH-5, AB-6, BF-7
{A,B,C,D,E}	{F,H}	{AC,BD,BE,CD,FH}	DH-3, EF-3, CG-4, EH-4, GH-5, AB-6, BF-7
{A,B,C,D,E,F,H}		{AC,BD,BE,CD, DH,FH}	EF-3, CG-4, EH-4, GH-5, AB-6, BF-7
{A,B,C,D,E,F,G,H}		{AC,BD,BE,CD,CG, DH,FH}	EH-4, GH-5, AB-6, BF-7

3.4 **Zadaci**

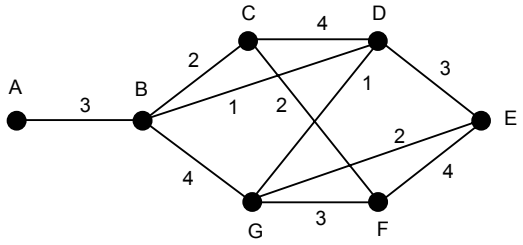
Z3.1: Za graf sa slike pronaći minimalno razapinjajuće stablo Kruskalovim algoritmom.



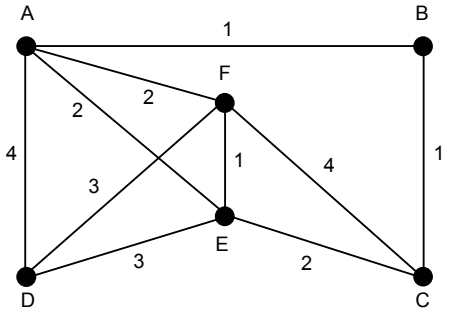
Z3.2: Za graf sa slike pronaći minimalno razapinjajuće stablo Primokvim algoritmom.



Z3.3: Za graf sa slike pronaći minimalno razapinjajuće stablo Primovim algoritmom.

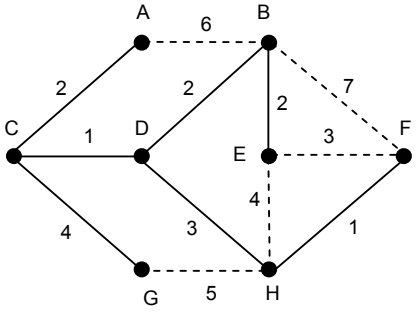


Z3.4: Za graf sa slike pronaći minimalno razapinjajuće stablo Kruskalovim algoritmom.



4. Najkraće udaljenosti u grafu

Udaljenost u grafu definiramo kao sumu težina bridova koji sačinjavaju put od počernog vrha do krajnjeg. Budući da u grafu općenito možemo imati i više različitih puteva između dva čvora njihove duljine ne moraju nužno biti iste. U ovom poglavlju proučavat ćemo puteve u grafu koji predstavljaju najmanje udaljenosti između dva vrha.



Slika 9. Najkraće udaljenosti od vrha A

Na slici 9 vidimo kako se ostvaruju minimalne udaljenosti od vrha A do svih ostalih vrhova. Konstruiranje takvog minimalnog puta metodom pokušaja i pograšaka za manje je grafove jednostavno, ali problemi nastaju kad se broj vrhova grafa počinje povećavati na više desetaka. Zbog toga se koristimo Dijkstrinim algoritmom za pronalaženje najkraćeg puta od jednog vrha do svih ostalih.

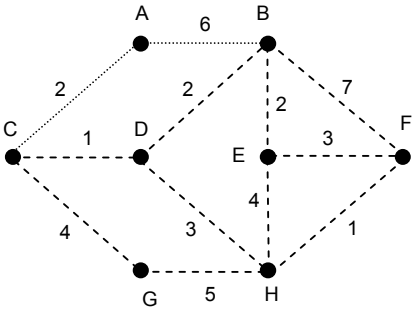
4.1 Dijkstrin algoritam za promalaženje najkraćih puteva

Dijkstrin algoritam je u biti proširenje Primovog algoritma prikazanog u poglavlju 3. Ideja je u tome da se gradi stablo koje se sastoji od bridova koji čine minimalne puteve od početnog vrha do svih ostalih vrhova. Kao u Primovom algoritmu u jednom koraku biramo brid koji spaja vrh koji nije u stablu sa stablom, ali pri tome za svaki vrh računamo udaljenost od početnog vrha. Od svih bridova koje u jednom koraku možemo izabrati uzimamo onaj za kjoji je pripadni vrh, koji nije u stablu, najmanje udaljen od početnog vrha.

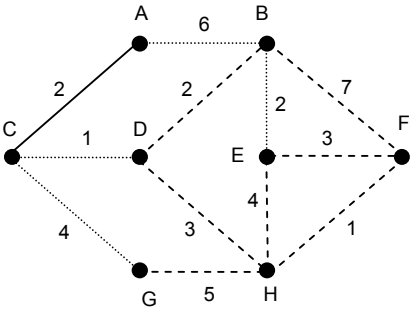
Prikažimo sad taj posupak pseudokodom:

```
Stablo = ∅
Stavi u stablo početni vrh i označi ga s udaljenosti 0.
Dok je Broj_Vrhova(Stablo) < Broj_Vrhova(Graf) ponavljaj
      Za sve vrhove koji su susjedni nekom vrhu iz stabla izračunaj
      najmanju udaljenost od početnog vrha kao udaljenost vrha u stablu
      kojem je taj vrh susjedan + težina brida koji ih spaja.
      Od svih tih vrhova izaberi onaj koji ima najmanju vrijednost
      udaljenosti i stavi ga u stablo zajedno s pripadajućim bridom.
```

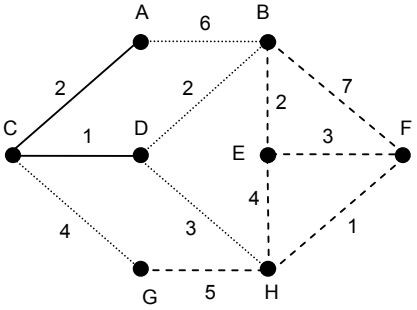
Sljedećim primjerom prikazat ćemo kako smo izgradili puteve minimalne udaljenosti s grafom prikazanom na grafu na slici 9.



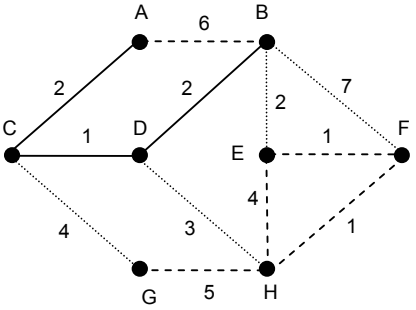
a)



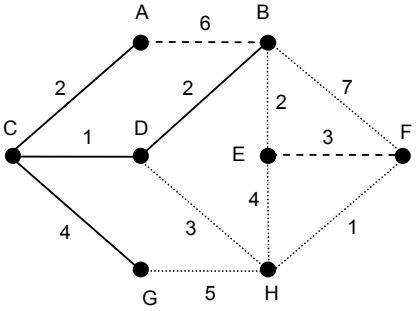
b)



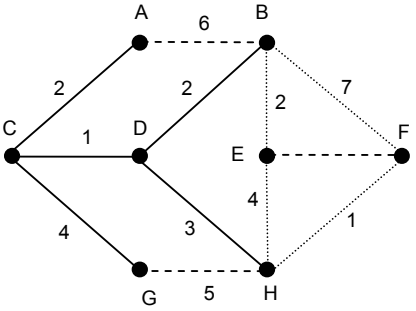
c)



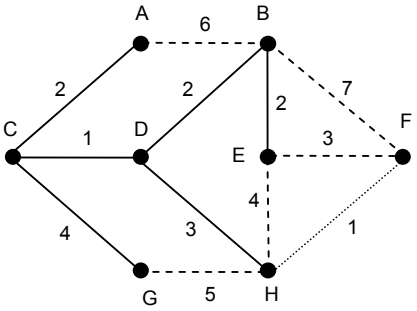
d)



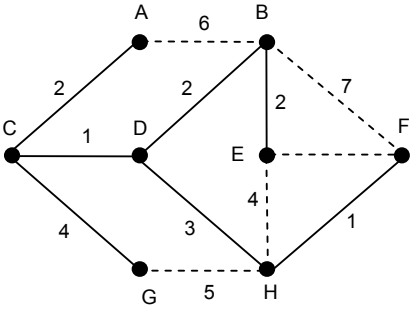
e)



f)



g)



h)

U prvom koraku, a), biramo između bridova AB i AC. Izaberemo brid AC jer je udaljenost do C (2) manja nego do B (6). U drugom koraku, b), biramo između AB, CD i CG te uzimamo CD jer je D najbliži (3). U trećem koraku izaberemo DB jer je B najmanje udaljen (5). U četvtom koraku CG jer je udaljenost do G 6, zatim DH jer je H udaljen 6. U posljednja dva koraka biramo BE jer je udaljenost do E 7 i HF jer je udaljenost do F 7.

Kako bismo algoritam lakše proveli koristimo se tabličnim zapisom kao kod Primovog algoritma. Jedina je razlika ta što u trećem stupcu zapisujemo udaljenosti od početnog vrha.

	A	B	C	D	E	F	G	H
Ⓐ	~~0~~	6	2					
B	~~6~~	0		2	2	7		
C	~~2~~		0	1			4	
D		2	1	0				3
E		2			0	3		4
F		7			3	0		1
G			4				0	5
H				3	4	1	5	0

1	A	C	2	C	2	AC

	A	B	C	D	E	F	G	H
Ⓐ	~~0~~	6	~~2~~					
B	~~6~~	0		2	2	7		
Ⓒ	~~2~~		~~0~~	1			4	
D		2	~~1~~	0				3
E		2			0	3		4
F		7			3	0		1
G			~~4~~				0	5
H				3	4	1	5	0

2	A	B	6	D	3	CD
	C	D	3			

	A	B	C	D	E	F	G	H
Ⓐ	~~0~~	6	~~2~~					
B	~~6~~	0		~~2~~	2	7		
Ⓒ	~~2~~		~~0~~	~~1~~			4	
Ⓓ		2	~~1~~	~~0~~				3
E		2			0	3		4
F		7			3	0		1
G			~~4~~				0	5
H				~~3~~	4	1	5	0

3	A	B	6	B	5	BD
	C	G	6			
	D	B	5			

	A	B	C	D	E	F	G	H
Ⓐ	~~0~~	~~6~~	~~2~~					
Ⓑ	~~6~~	~~0~~		~~2~~	2	7		
Ⓒ	~~2~~		~~0~~	~~1~~			4	
Ⓓ		~~2~~	~~1~~	~~0~~				3
E		~~2~~			0	3		4
F		~~7~~			3	0		1
G			~~4~~				0	5
H				~~3~~	4	1	5	0

4	B	E	7	G	6	CG
	C	G	6			
	D	H	6			

	A	B	C	D	E	F	G	H
Ⓐ	~~0~~	~~6~~	~~2~~					
Ⓑ	~~6~~	~~0~~		~~2~~	2	7		
Ⓒ	~~2~~		~~0~~	~~1~~			~~4~~	
Ⓓ		~~2~~	~~1~~	~~0~~				3
E		~~2~~			0	3		4
F		~~7~~			3	0		1
Ⓖ			~~4~~				~~0~~	5
H				~~3~~	4	1	~~5~~	0

5	B	E	7	H	6	DH
	D	H	6			
	G	H	11			

	A	B	C	D	E	F	G	H
A	~~0~~	~~6~~	~~2~~					
B	~~6~~	~~0~~		~~2~~	2	7		
C	~~2~~		~~0~~	~~1~~			~~4~~	
D		~~2~~	~~1~~	~~0~~				~~3~~
E		~~2~~			0	3		~~4~~
F		~~7~~			3	0		~~1~~
G			~~4~~				~~0~~	~~5~~
H				~~3~~	4	1	~~5~~	~~0~~

6	B	E	7	E	7	BE
	H	F	7			

	A	B	C	D	E	F	G	H
A	~~0~~	~~6~~	~~2~~					
B	~~6~~	~~0~~		~~2~~	~~2~~	7		
C	~~2~~		~~0~~	~~1~~			~~4~~	
D		~~2~~	~~1~~	~~0~~				~~3~~
E		~~2~~			~~0~~	3		~~4~~
F		~~7~~			~~3~~	0		~~1~~
G			~~4~~				~~0~~	~~5~~
H				~~3~~	~~4~~	1	~~5~~	~~0~~

7	B	F	12	F	7	HF
	E	F	10			
	H	F	7			

Ovim algoritmom veoma jednostavno dobivamo najmanje udaljenosti od jednog vrha do svih ostalih vrhova u grafu, međutim ako nam trebaju vrijednosti najmanjih udaljenosti između svaka dva vrha u grafu trebali bismo ovaj algoritam ponoviti onoliko puta koliko ima vrhova u grafu. Nije teško primjetiti da je to veoma zahtjevan posao i zato se koristimo jednim drugim algoritmom iz teorije grafova. Rječ je o Floydovom algoritmu s pomoću kojeg veoma jednostavno možemo pronaći vrijednosti najmanjih udaljenosti između svih parova vrhova u grafu.

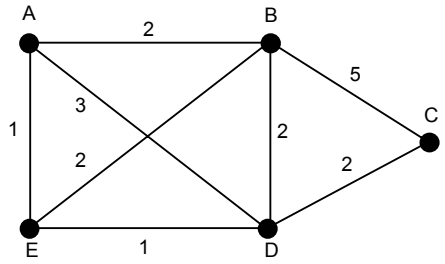
4.2 Floydov algoritam za pronalaženje najkraćih puteva

Floydovim algoritmom pronalazimo najmanje udaljenosti između svih parova vrhova u grafu. Ideja algoritma je ispitivanje svih mogućih puteva u grafu, ali se pri takvom ispitivanju koristi činjenica da ovakav problem ima optimalnu substrukturu te da se do ukupnog minimuma može doći spajanjem minimuma problema manjeg reda. Ovaj je algoritma u biti tipičan primjer dinamičkog programiranja, a dokaz da je rezultat izvođenja algoritma ispravan je veoma složen. Prikažimo sad algoritam pseudokodom, pri tome je matrica incidencije označena s `G[i,j]`.

```
Za vrh_kroz = 1 do broj_vrhova
    Za vrh_od = 1 do broj_vrhova
        Ako G[vrh_od,vrh_kroz]!=∞
            Za vrh_do = 1 do broj_vrhova
                Ako G[vrh_kroz,vrh_do]!=∞
                    Ako G[vrh_od,vrh_do]==∞ ili
                    G[vrh_od,vrh_do]>G[vrh_od,vrh_kroz]+G[vrh_kroz,vrh_do]
                        G[vrh_od,vrh_do]=G[vrh_od,vrh_kroz]+G[vrh_kroz,vrh_do]
```

Ovako napisano algoritam izgleda veoma složeno no u biti je veoma jednostavan. Za svaki vrh, `vrh_kroz`, iz grafa pokušavamo poboljšati put između neka druga dva vrha, `vrh_od` i `vrh_do` tako da provjerimo da li se udaljenost smanjuje ako put između ta dva vrha prođe kroz `vrh_kroz`. Algoritam tokom izvođenja mijenja tablicu incidencije te na kraju izvođenja u tablici na mjestu *i,j* imamo najmanju udaljenost između vrhova *i* i *j*.

Algoritam ćemo prikazati na jednom manjem primjeru budući da broj koraka raste s brojem vrhova grafa. Graf na kojem ćemo prikazati izvođenje algoritma prikazan je na slici 10.



Slika 10. Graf na za prikaz Floydovog algoritma.

Tablica incidencije zadanog grafa je:

	A	B	C	D	E
A	0	2		3	1
B	2	0	5	2	2
C		5	0	2	
D	3	2	2	0	1
E	1	2		1	0

U prvom koraku uzimamo vrh A za vrh kroz koji ćemo pokušati poboljšati put između ostalih vrhova te redom zapisujemo puteve koje želimo poboljšati.

B→C – Ne možemo poboljšati jer trenutačno nemamo put od A do C
B→D – Ne poboljšavamo jer je BA + AD jednak 5 a tu već imamo 2
B→E – Ne možemo poboljšati već postojeći put
C→D – Ne možemo poboljšati jer trenutačno nemamo put od C do A
C→E – Ne možemo poboljšati jer trenutačno nemamo put od C do A
D→E – Ne možemo pobolkjšati već postojeći put

Ovdje vidimo da prvi korak uopće nije promijenio matricu incidencije te da s vrhom A ne možemo poboljšati već postojeće puteve. Prelazimo na drugi korak s vrhom B.

A→C – Budući da put ne postoji mi ga stvaramo i označimo ga dužinom 7
A→D – Ne možemo poboljšati već postojeći put
A→E – Ne možemo poboljšati već postojeći put
C→D – Ne možemo poboljšati već postojeći put
C→E – Budući da put ne postoji mi ga stvaramo i označimo ga dužinom 7
D→E – Ne možemo poboljšati već postojeći put

Matrica incidencije nam je sad:

	A	B	C	D	E
A	0	2	7	3	1
B	2	0	5	2	2
C	7	5	0	2	
D	3	2	2	0	1
E	1	2		1	0

U trećem koraku koristimo vrh C.

A→B – Ne možemo poboljšati već postojeći put
A→D – Ne možemo poboljšati već postojeći put
A→E – Ne možemo poboljšati već postojeći put
B→D – Ne možemo poboljšati već postojeći put
B→E – Ne možemo poboljšati već postojeći put
D→E – Ne možemo poboljšati već postojeći put

Treći korak također nije ništa promijenio, a algoritam se nastavlja s vrhom D

A→B – Ne možemo poboljšati već postojeći put
A→C – Udaljenost AD+DC je 5 što je manje od 7 te mjenjamo tablicu incidencije
A→E – Ne možemo poboljšati već postojeći put
B→C – Udaljenost BD+DC je 4 što je manje od 5 te mjenjamo tablicu incidencije
B→E – Ne možemo poboljšati već postojeći put
C→E – Budući da put ne postoji mi ga stvaramo i označimo ga dužinom 3

Matrica incidencije tad je:

	A	B	C	D	E
A	0	2	5	3	1
B	2	0	4	2	2
C	5	4	0	2	3
D	3	2	2	0	1
E	1	2	3	1	0

U posljedenjem koraku koristimo vrh E.

A→B – Ne možemo poboljšati već postojeći put
A→C – Udaljenost AE+EC je 4 što je manje od 5 te mjenjamo tablicu incidencije
A→D – Udaljenost AE+ED je 2 što je manje od 3 te mjenjamo tablicu incidencije
B→C – Ne možemo poboljšati već postojeći put
B→D – Ne možemo poboljšati već postojeći put
C→D – Ne možemo poboljšati već postojeći put

Matrica incidencije tad je:

	A	B	C	D	E
A	0	2	4	2	1
B	2	0	4	2	2
C	4	4	0	2	3
D	2	2	2	0	1
E	1	2	3	1	0

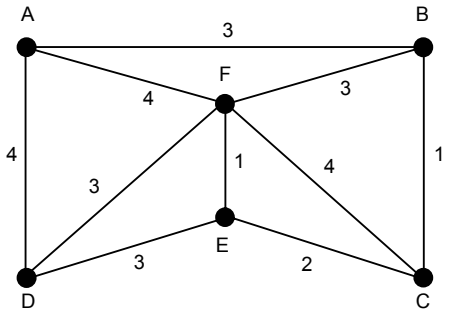
Kao što je iz matrice vidljivo sad imamo vrijednosti minimalnih udaljenosti za sve parove vrhova u grafu, međutim ne znamo kako se te minimalne udaljenosti ostvaruju. Pratimo li izvođenje algoritma možemo rekonstruirati te puteve tako da pratimo kako smo promjenili težine u matrici incidencije u svakom koraku.

2: A→C ≡ A→B→C
   C→E ≡ C→B→E
4: A→C ≡ A→D→C
   B→C ≡ B→D→C
   C→E ≡ C→D→E
5: A→C ≡ A→E→C ≡ A→E→D→C
   A→D ≡ A→E→D

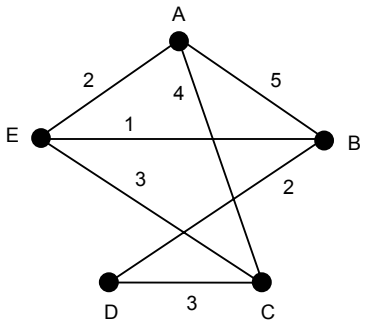
Ostali putevi koji nisu navedeni ostvareni su direktnom vezom, tj. samo jednim bridom.

4.3 **Zadaci**

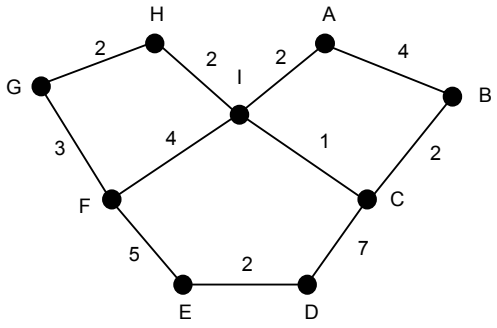
Z4.1: Za graf sa slike pronaći najkraće putove od vrha A do svih ostaloh vrhova u grafu.



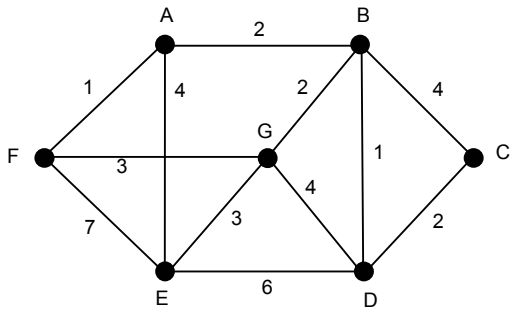
Z4.2: Za graf sa slike pronaći najkraće udaljenosti između svih vrhova u grafu.



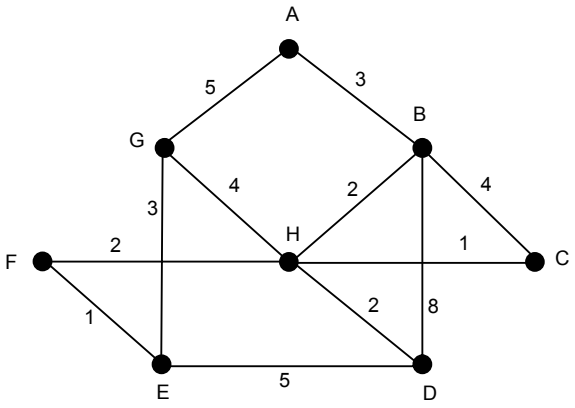
Z4.3: Za graf sa slike pronaći najkraće puteve od vrha D do svih ostalih vrhova.



Z4.4: Za graf sa slike pronaći najkraće puteve od vrha C do svih ostalih vrhova.

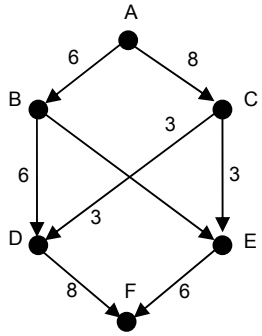


Z4.5 Za graf sa slike pronaći najkraće puteve od vrha E do svih ostalih vrhova.



5. Maksimalni protok u mreži

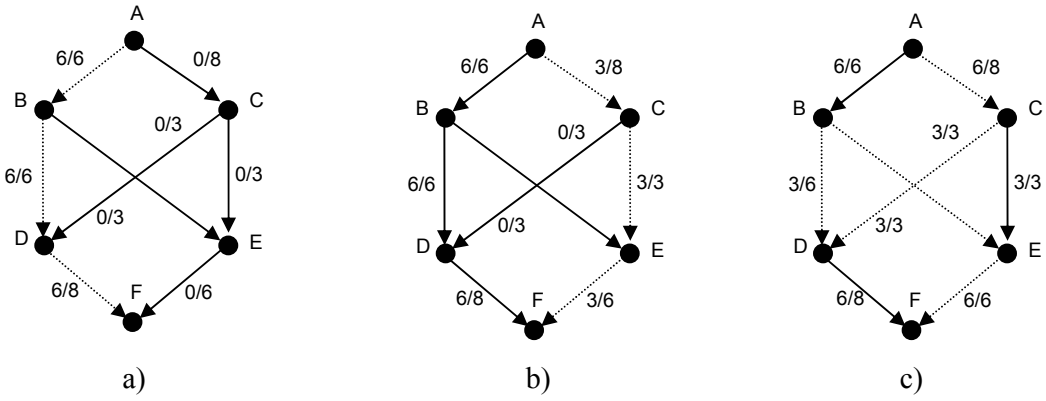
Kao što smo u uvodu naglasili, grafovima možemo veoma jednostvno modelirati probleme iz stvarnog života. Jedna od uobičajenih primjena grafova je opisivanje mreže naftovoda, prometnica i sličnih protočnih struktura. Promotrimo primjer naftovoda kojeg opisjemo sljedećim grafom.



Slika 11. Model naftovoda prikazan grafom

Kao što je sa slike 11 vidljivo bridovi koji prikazuju cjevovode označeni su strelicama kako bi se naglasilo da nafta kroz taj brid može teći u samo jednom smjeru. Takav graf tada nazivamo usmjerenim težinskim grafom jer su mu bridovi usmjereni i svaki od njih ima pridjeljenu težinu, odnosno u ovom ćemo modelu to nazivati kapacitetom. Vrh iz koga sve strelice izlaze nazivat ćemo izvorom, u primjeru to je vrh A, a onaj u koji su usmjerene sve strelice nazvat ćemo ponorom, u primjeru vrh F. Problem koji se prirodno nameće je koliki kapacitet svakog brida moramo koristiti da bi ukupni protok od izvora do ponora bio maksimalan. Rješenje takvog problema za veće mreže nije uopće trivijalno te se zato koriste razni algoritmi kako bi se takav problem efikasno riješio.

Za prikladno opisivanje problema proširit ćemo oznaku težine brida na dva broja od kojih će nam drugi govoriti koliki je kapacitet brida, tj. koliki je maksimalni protok kroz taj brid, a prvi će nam označavati koliki dio kapaciteta brida koristimo te ćemo to nazivati protokom kroz taj brid. Na sljedećoj je slici prikazano kako u nekoliko koraka maksimiziramo protok kroz zadanu mrežu.



Slika 12. Određivanje maksimalnog protoka kroz mrežu

Kao što je vididljivo sa slike 12, povećavanje protoka provodimo postupno sve dok se protok više ne možemo povećati. U prvom koraku, slika 12.a, povećavamo protok kroz bridove AB, BD i DF sa 0 na 6. U drugom koraku, slika 12.b, povećavamo protok kroz bridove AC, CE i EF sa 0 na 3. U trećem koraku, slika 12.c, povećanje protoka je malo kompliciranije i sastoji se u tome da se protok kroz brid BD smanji za 3 te taj dio protoka preusmjerimo bridovim BE i EF. Budući da sad u vrhu D imamo manjak, taj manjak nadoknađujemo dodatnim protokom kroz AC i CD. Kao što je iz ovog kratkog primjera vidljivo rješenje ovakvog problema nije trivijalno. Algoritam koji efikasno rješava ovaj problem je Ford-Fulkersonov algoritam.

5.1 Ford-Fulkersonov algoritam

Kao što je već rečeno Ford-Fulkersonovim algoritmom pronalazimo maksimalni protok kroz zadanu mrežu. Algoritam provodimo u nizu koraka te u svakom koraku povećavamo ukupni protok u mreži sve dok ne dostignemo maksimum.

Ideja algoritma je pronaći u grafu put koji povećava trenutni protok i tada promjeniti protok po bridovima tog puta te tako povećati ukupni protok. Kad više nije moguće pronaći put koji bi poboljšao protok algoritam završava. Prikažimo to sad pseudokodom:

```
Protok kroz sve bridove staviti na 0.
Dok možeš pronaći put koji poboljšava protok
    Promjeni protok kroz bridove pronađenog puta.
```

U samom algoritmu nije specificirano kako pronaći put koji poboljšava protok a to znači da bilo koji postupak pronalaženja tog puta daje ispravno rješenje. Međutim, broj koraka u kojem ćemo postići rješenje uvelike ovisi o načinu pronalaženje takvog puta. Idealno bi bilo da u svakom koraku pronađemo put koji maksimalno povećava protok i to možemo učiniti algoritmom koji je veoma sličan Dijkstrinom algoritmu promatranom u prethodnom poglavlju. No prije nego objasnimo kako provodimo taj algoritam trebalo bi detaljnije objasniti kako ćemo označavati protok kroz bridove. Kao što je iz samog problema vidljivo koristili smo se usmjerenim bridovima kako bismo opisali da nafta može teći samo u jednom smjeru. Zbog potreba algoritma uvest ćemo i bridove u suprotnom smjeru koje će imati iste vrijednosti protoka i kapaciteta kao brid u pravom smjeru, ali će zato predznaci biti suprotni.

Mogući iznos za koji možemo povećati protok kroz neki brid u pravom smjeru je jednak razlici kapaciteta i trenutnog protoka, tj. možemo kroz taj brid progurati još nafte dok ne postignemo iznos kapaciteta. Ono što je možda zbunjujuće ovdje je činjenica da protok kroz brid u soprotnom smjeru možemo povećati za apsolutni iznos njegova protoka, u biti time smanjujemo protok nafte u pravom smjeru. Zašto bi to trebalo povećati ukupni protok bit će u potpunosti jasno nakon što prikažemo kako smo ostvarili maksimalni protok na mreži sa slike 11.

Ideja algoritma kojeg ćemo primjeniti je pronaći vrijednosti maksimalnih protoka od izvora do svakog drugog vrha u grafu. Razliku u algoritmu u odnosu na Dijkstrin algoritam očitujemo u trećem stupcu tablice koju gradimo. U taj stupac upisujemo manji od brojeva koji predstavlja protok do vrha koji se nalazi u stablu i mogućeg protoka kroz brid koji vodi od vrha u stablu do vrha koji nije u stablu. Prikažimo sad izvođenje Ford-Fulkersonovog algoritma na već zadanoj mreži.

Kao prvo napišemo tablicu incidencije koja nam predstavlja protok kroz mrežu. Svaka će nam oznaka predstavljati protok/kapacitet brida.

	A	B	C	D	E	F
A		0/6	0/8			
B	0/-6			0/6	0/3	
C	0/-8			0/3	0/3	
D		0/-6	0/-3			0/8
E		0/-3	0/-3			0/6
F				0/-8	0/-6	

Iz te tablice konstruiramo tablicu incidencije s kojom ćemo pronaći put koji povećava protok. Vrijednosti u tablici će nam predstavljati koliko možemo povećati protok kroz pripadni brid. Način na koji se računaju te vrijednosti opisan je u tekstu, a tu treba još samo reći da za bridove kod kojih je iznos za koji možemo povećati protok nula ne uzimamo u obzir u algoritmu. Treba još samo reći da ćemo u svakom koraku umjesto najmanje vrijednosti kao u Dijkstrinom algoritmu u retku promatranog vrha uzeti najveću vrijednost jer tražimo maksimalno povećavanje protoka. U susjednoj tablici prikazani su koraci algoritma.

	A	B	C	D	E	F
A		6	8			
B				6	3	
C				3	3	
D						8
E						6
F						

1	A	C	8	C	8	AC
2	A	B	6	B	8	AB
	C	D	3			
	C	E	3			
3	B	D	6	D	6	BD
	C	D	3			
4	B	E	3	F	6	DF
	C	E	3			
	D	F	6			

Postupak smo zaustavili kad smo dostigli ponor, tj. vrh F. Iz tablice je sad vidljivo da je maksimalno povećanje protoka iznosa 6 i to kroz put A→B→D→F. Sad shodno tome promjenimo protoke kroz te bridove i to unesemo u našu početnu tablicu incidencije za protoke.

	A	B	C	D	E	F
A		6/6	0/8			
B	-6/-6			6/6	0/3	
C	0/-8			0/3	0/3	
D		-6/-6	0/-3			6/8
E		0/-3	0/-3			0/6
F				-6/-8	0/-6	

Protok koji je ovom tablicom definiran prikazan je na slici 12.a. Iz ove tablice konstruirmo kao prije tablicu incidencije uz pomoć koje tražimo daljnja povećanja protoka. Ta nam je tablica tad:

	A	B	C	D	E	F
A			8			
B	6				3	
C				3	3	
D		6				2
E						6
F				6		

1	A	C	8	C	8	AC
2	C	D	3	D	3	CD
	C	E	3			
3	C	E	3	E	3	CE
	D	B	3			
4	D	B	3	B	3	DB
	E	F	3			
5	E	F	3	F	3	EF

U ovom koraku protok možemo povećati za 3 i to kroz put A→C→E→F. Promjenjena tablica incidencija za protoke je:

	A	B	C	D	E	F
A		6/6	3/8			
B	-6/-6			6/6	0/3	
C	-3/-8			0/3	3/3	
D		-6/-6	0/-3			6/8
E		0/-3	-3/-3			3/6
F				-6/-8	-3/-6	

Tablica prikazuje stanje na slici 12.b.

Iz te učinimo sljedeću tablicu incidencije za povećavanje protoka:

	A	B	C	D	E	F
A			5			
B	6				3	
C	3			3		
D		6				2
E			3			3
F				6	3	

1	A	C	5	C	5	AC
2	C	D	3	D	3	CD
3	D	B	3	B	3	DB
4	B	E	3	E	3	BE
	D	F	2			
5	D	F	2	F	3	EF
	E	F	3			

U ovom korako možemo povećati protok za 3 i to A→C→B→E→F. Kako je vidljivo sa slike 12.c prolazimo bridnom BD u suprotnom smjeru što znači da time smanjujemo protok kroz taj brid. Ono što nije odmah očito je da kako takvim smanjivanjem postižemo povećanje ukupnog protoka, međutim, promotrimo li sliku 12.c vidimo da taj dio protoka kroz BD nadoknađujemo onim iz CD, a dio koji više ne prolazi kroz BD sad prolazi bridom BE te tako povećava ukupni tok.

Tablica protoka nam sad izgleda:

	A	B	C	D	E	F
A		6/6	6/8			
B	-6/-6			3/6	3/3	
C	-6/-8			3/3	3/3	
D		-3/-6	-3/-3			6/8
E		-3/-3	-3/-3			6/6
F				-6/-8	-6/-6	

Tablica prikazuje stanje na slici 12.c.

Pokušajmo sada još povećati protok:

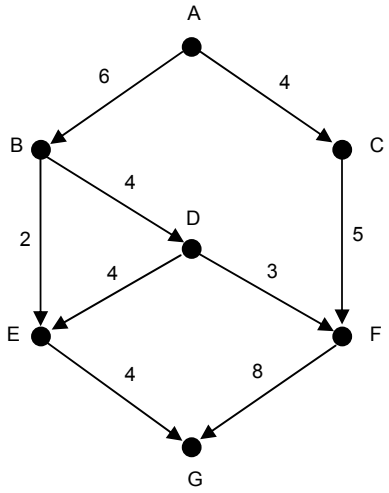
	A	B	C	D	E	F
A			2			
B	6			3		
C	6					
D		3	3			2
E		3	3			
F				6	6	

1	A	C	2	C	2	AC

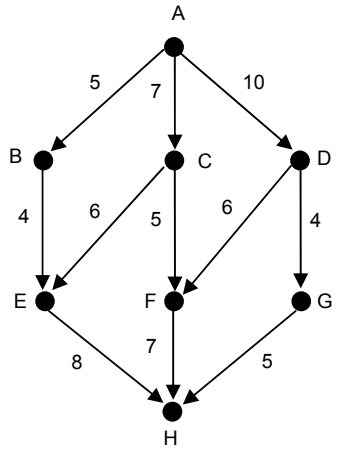
Kao što je iz postupka vidljivo, ne možemo više pronaći put kojim bismo povećali protok kroz mrežu te algoritam tu završava.

5.2 **Zadaci**

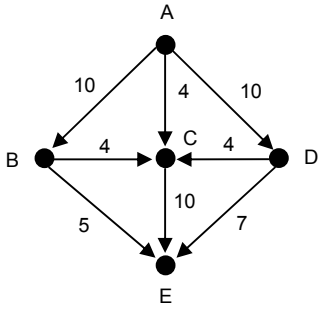
Z5.1: Za graf sa slike pronaći maksimalni protok od vrha A do vrha G.



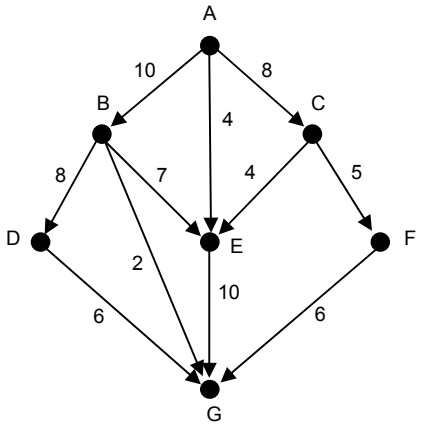
Z5.2: Za graf sa slike pronaći maksimalni protok od vrha A do H.



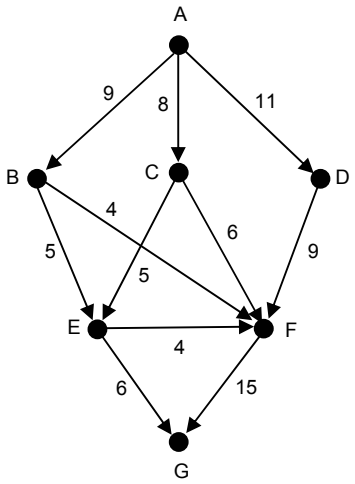
Z5.3: Za graf sa slike pronaći maksimalni protok od vrha A do vrha E.



Z5.4: Za graf sa slike pronaći maksimalni protok od vrha A do vrha G.



Z5.5: Za graf sa slike pronaći maksimalni protok od vrha A do vrha G.



6. Rješenja zadataka

6.1 Osnovni pojmovi u teoriji grafova

Z2.1:

G = {A,B,C,D,E,F,G}
E = {AC,AE,BC,BF,BG,CE,DG,FG}

Z2.2:

	A	B	C	D	E	F
A	0				2	
B		0	3	4	4	5
C		3	0	1		
D		4	1	0	7	3
E	2	4		7	0	
F		5		3		0

Z2.3:

	A	B	C	D	E	F	G	H
A	0	3						2
B	3	0	2			3		5
C		2	0	6				
D			6	0	2	4		
E				2	0	1		
F		3		4	1	0	6	
G						6	0	
H	2	5						0

6.2 **Razapinjajuća stabla**

Z3.1: Stablo čine bridovi:
{AI, BC, CI, DJ, EJ, FG, GH, HI, IJ}

Z3.2: Stablo čine bridovi:
{AE, BD, EB, EC}

Z3.3: Stablo čine bridovi:
{AB, BC, BD, CF, DG, GE}

Z3.4: Stablo čine bridovi:
{AB, AF, BC, DE, EF}

6.3 **Najkraće udaljenosti u grafu**

Z4.1: Puteve čine bridovi:
{AB, AD, AF, BC, FE}
Udaljenosti su:
A→B = 4
A→C = 4
A→D = 4
A→E = 5
A→F = 4

Z4.2: Tablica udaljenosti je:

	A	B	C	D	E
A	0	3	4	5	2
B	3	0	4	2	1
C	4	4	0	3	3
D	5	2	3	0	3
E	2	1	3	3	0

Z4.3: Puteve čine bridovi:
{AI, BC, CD, CI, DE, EF, FG, HI}
Udaljenosti su:
D→A = 10
D→B = 9
D→C = 7
D→E = 2
D→F = 7
D→G = 10
D→H = 10
D→I = 8

Z4.4: Puteve čine bridovi:
{AB, AF, BD, BG, CD, EG}
Udaljenosti su:
C→A = 5
C→B = 3
C→D = 2
C→E = 8
C→F = 6
C→G = 5

Z4.5: Puteve čine bridovi:
{AB, BH, CH, DE, EF, EG, FH}
Udaljenosti su:
E→A = 8
E→B = 5
E→C = 4
E→D = 5
E→F = 1
E→G = 3
E→H = 3

6.4 **Maksimalni protok u mreži**

Z5.1: Maksimalni protok je 10
Protok kroz brid AB je 6/6.
Protok kroz brid AC je 4/4.
Protok kroz brid BD je 4/4.
Protok kroz brid BE je 2/2.
Protok kroz brid CF je 4/5.
Protok kroz brid DE je 1/4.
Protok kroz brid DF je 3/3.
Protok kroz brid EG je 3/4.
Protok kroz brid FG je 7/8.

Z5.2: Maksimalni protok je 19
Protok kroz brid AB je 4/5.
Protok kroz brid AC je 7/7.
Protok kroz brid AD je 8/10.
Protok kroz brid BE je 4/4.
Protok kroz brid CE je 4/6.
Protok kroz brid CF je 3/5.
Protok kroz brid DF je 4/6.
Protok kroz brid DG je 4/4.
Protok kroz brid EH je 8/8.
Protok kroz brid FH je 7/7.
Protok kroz brid GH je 4/5.

Z5.3 Maksimalni protok je 22
Protok kroz brid AB je 9/10.
Protok kroz brid AC je 4/4.
Protok kroz brid AD je 9/10.
Protok kroz brid BC je 4/4.
Protok kroz brid BE je 5/5.
Protok kroz brid DC je 2/4.
Protok kroz brid CE je 10/10.
Protok kroz brid DE je 7/7.

Z5.4 Maksimalni protok je 22
Protok kroz brid AB je 10/10.
Protok kroz brid AC je 8/8.
Protok kroz brid AE je 4/4.
Protok kroz brid BD je 5/8.
Protok kroz brid BE je 3/7.
Protok kroz brid BG je 2/2.
Protok kroz brid CE je 3/4.
Protok kroz brid CF je 5/5.
Protok kroz brid DG je 5/6.
Protok kroz brid EG je 10/10.
Protok kroz brid FG je 5/6.

Z5.5 Maksimalni protok je 21

Protok kroz brid AB je 8/9.
Protok kroz brid AC je 8/8.
Protok kroz brid AD je 5/11.
Protok kroz brid BE je 4/5.
Protok kroz brid BF je 4/4.
Protok kroz brid CE je 2/5.
Protok kroz brid CF je 6/6.
Protok kroz brid DF je 5/9.
Protok kroz brid EG je 6/6.
Protok kroz brid FG je 15/15.

7. Literatura

[1] Bollobás, B., Modern Graph Theory, 1998, Springer-Verlag New York Inc.

[2] Cormen, T.H., Leisterson, C.E., Rivest, R.L., Introduction to Algorithms, 1999, MIT Press

[3] Sedgewick, R., Algorithms in C, 1996, Addison-Wesley Publising Company Inc.

[4] Wilson, R.J., Introduction to Graph Theory,1999, Longman Group Ltd