	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. Riječ je o Floydovom algoritmu s pomoću kojeg veoma jednostavno možemo pronaći vrijednosti najmanjih udaljenosti između svih parova vrhova u grafu.

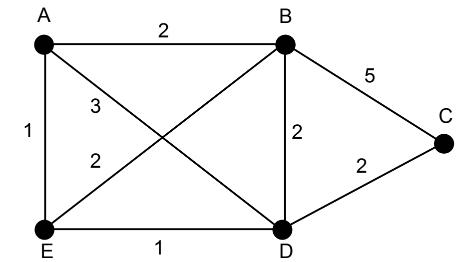
4.2 Floydov algoritam za pronalaženje najkraćih puteva

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.