**DISKRETNA MATEMATIKA**

**TEŽINSKI GRAFOVI**

Definicija: TEŽINSKI GRAF je uređeni par gdje je *G = (V, E)* graf, a funkcija koju nazivamo **težinska funkcija.**

Prirodno se postavlja pitanje pronalaska minimalnog (u smislu težine) razapinjujućeg stabla.

1. **Optimalno (minimalno) razapinjuće stablo**

**Kruskalov algoritam za nalaženje optimalnog (minimalnog)razapinjućeg stabla**

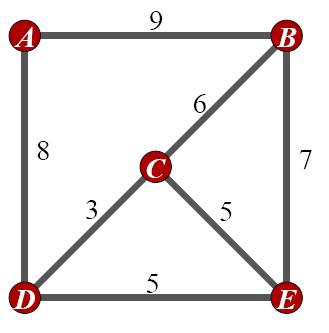
* efikasno rješava problem povezivanja npr. gradove treba povezati mrežom autocesta tako da gradnja bude čim više ekonomična,ako je poznata cijena izgradnje 1km autoceste.

Neka je *G = (V, E)*povezan graf i *ω* nenegativna težinska funkcija na *E*.

1. Stavimo
2. Odaberemo brid *e*najmanje težine i
3. Ako su odabrani bridovi odaberemo sljedeći brid takav da je:
4. acikličan
5. minimalna
6. Završimo kada (3) više ne možemo provesti.

Napomena: Kruskalov algoritam je primjer pohlepnog algoritma. Lokalno nalazi najbolje rješenje.

ZADATAK: Nađite minimalno razapinjujućestablo za težinski graf sa slike



**Primov algoritam**

- Ovim algoritmom pokušavamo iz zadanog grafa izgraditi minimalnorazapinjajuće stablo.

Neka je *G = (V, E)*povezan težinski graf, i *ω* nenegativna težinska funkcija na *E*.

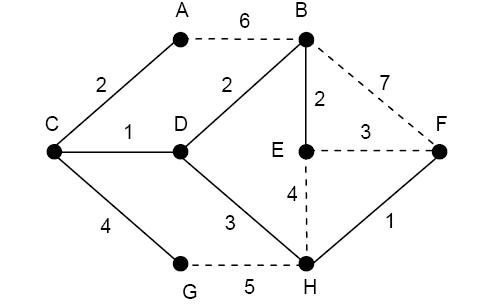
1. Postavimo (stablo) T=. Odaberimo i definiramo , , .
2. Odaberimo brid minimalne težine, takav da je, odnosno brid minimalne težine koji je incidentan s jednim vrhom iz skupa vrhova stabla T i drugim vrhom iz skupa S.
3. Definiramo: , , .
4. Ponavljamo korake (2) i (3) sve dok tj. dok je broj bridova u stablu S manji od maksimalnog mogućeg broja.

ZADATAK: Primovim algoritmom nađite minimalno razapinjujuće stablo za težinski graf iz prethodnog zadatka.

1. **Problem najkraćeg puta**

**Udaljenost dvaju vrhova d(u,v)** u težinskom grafu definiramo kao **minimalna suma težina bridova** koji čine put od početnog vrha do krajnjeg, tj. duljina najkraćeg puta.

Budući da u grafu općenito možemo imati i više različitih puteva izmeđudva čvora njihove duljine ne moraju nužno biti iste.



Na slici vidimo minimalne udaljenosti od vrha A do svih ostalihvrhova. Konstruiranje takvog minimalnog puta metodom pokušaja i pogrešaka za manje jegrafove jednostavno, ali problemi nastaju kad se broj vrhova grafa počinje povećavati na višedesetaka.

Napomena: Ako , tada identificiramo vrhove *u* i *v*. Ako , onda se težina definira:.

Ograničit ćemo se na jednostavne grafove.

**ALGORITAM OPTIMIZACIJE - Algoritam za pronalaženje najkraćeg puta (PROCES RASTA STABLA)**

**Dijkstrin algoritam**

- algoritmom pronalazimo najkraći *(u,v)-*put ali i najkraće puteve od *u* do svih ostalih vrhova.

Neka je takav da je , te označimo sa njegov skupovni komplement u skupu svih vrhova.

Neka je najkraći put od *u* do skupa (pri čemu udaljenost ili najkraćim putom od vrha *u* do skupa vrhova prirodno podrazumijevamo najkraću od svih udaljenosti ), onda je vrh sasvim sigurno iz skupa S i dio puta P od vrha *u* do vrha je najkraći put između ta dva vrha. Zato vrijedi:

, za neki ,

pa je udaljenost od vrha *u* do skupa vrhova dana formulom:

(1)

**Algoritam se temelji na ovoj formuli.**

Počinjemo sa skupom *S0*={*u0*}, te konstruiramo rastući (u smislu inkluzije) niz S0,S1,S2,...,S|V(G)|-1

podskupova od V(G) tako da na kraju i-tog koraka znamo najkraće puteve od zadanog početnog vrha *u0*do svih vrhova skupa S*i*.

PRVI KORAK:

Tražimo vrh koji je najbliži početnom vrhu u0. Taj vrh ćemo lako odrediti izračunamo li , pa onda izaberemo taj vrh u1є za koji vrijedi . Takvih vrhova može biti i više,a mi izaberemo jedan od njih. Da bismo izračunali , primijenimo formulu (1) prema kojoj slijedi da je

što je zapravo najmanja težina brida incidentnog s početnim vrhom *u0*, pa sejednostavno izračuna tražena udaljenost .

Stavimo da je S1={u0,u1}.

P1=u0,u1.

k-ti KORAK:

U ovaj korak ulazimo nakon što smo odredili skup vrhova Sk={u0,u1,...,uk} i pripadne najkraće puteve P1, P2,...,Pk. Pomoću formule (1) računamo , te izaberemo vrh uk+1 є kao onaj vrh za koji se postiže jednakost =. S obzirom da je

= za neki j ≤ k, put Pk+1 konstruiramo tako da putu Pjdodamo brid {uj,uk+1}.

Stavimo da je Sk+1={u0,u1,...,uk+1}.

U svakom koraku najkraći putevi zajedno čine stablo.

**Profinjenje opisanog algoritma**.

Razmotrimo li glavnu formulu (1) vidimo da je potrebno dosta uspoređivanja dok se ne ustanovi traženi minimum, te će se mnoga uspoređivanja vršiti višekratno. Da bi se to izbjeglo, uvest ćemo dodatni podatak koji ćemo pamtiti i prenositi iz koraka u korak algoritma.

Svakom vrhu zadanog težinskog grafa pridružujemo realni broj koji označava gornju među za traženu udaljenost (ta gornja među u izvjesnom trenutku postat će traženi minimum).

PRVI KORAK

*l(u0)=0*

*l(v)=,* za *v ≠u0*

S0={ *u0*}, te *i=0*.

DRUGI KORAK

zamijeni *l(v)* s min{ *l(v), l(ui)*+*(ui,v)*}, za neki *ui* є Si. Izračunaj min{ *l(v)* | *v* є }, te odredi ui+1 kao onaj vrh za koji se postiže taj minimum. Stavi Si+1=Si.

TREĆI KORAK

Ako je *i* = |V(G)|-1, algoritam je gotov.

Ako je *i*< |V(G)|-1, zamijeni *i* s *i+1* i vrati se na drugi korak.

Nakon izvršenja algoritma, svim vrhovima zadanog težinskog grafa pridijeljena je vrijednost *l(v)* koja predstavlja duljinu najkraćeg puta od danog početnog vrha *u0* do njih samih.

ZADATAK:

Dijkstrinovim algoritmom odredite nakraće puteve od *u0* do svih ostalih vrhova grafa:

