Tutorijal 2A: ALGORITMI RUTIRANJA

Cilj routing protokola je da odluče ispravnu rutu (sekvencu rutera) kroz mrežu od izvora do odredišta. Routing algoritmi opisuju mrežu preko grafova. Čvorovi u tim grafovima predstavljaju rutere, pri cemu su svi ruteri povezani preko linkova koji nose određene parametre. Vaznost proračunavanja ruta prilikom rutiranja zavisi od različitih faktora kao što su: bandwith, delay, throughput; no nas će najviše zanimati rutiranje koje je u zavisnosti od cijene (cost) linkova.

Postoji više načina dodjeljivanja cijene odeđenom linku:

- Svakom linku se dodijeli cijena vrijednosti 1 - Smanjuje se broj hopova.
- Svakom linku se dodjeli cijena koja ima vrijednost suprotnu brzini prenosa - Maksimizira se korištenje high speed linkova.
- Dodjeljivanje cijene iz političkih/poslovnih razloga

Ruta predstavlja skup povezanih rutera od izvora do destinacije i teži se da:

- Cijena rute bude jednaka sumi pojedinacnih dijelova rute
- Ili se tezi ka minimizaciji cijena rute

Ovaj tutorijal će obraditi dvije vrste algoritama: Dijkstra i Bellman-Fordov algoritam koji spadaju u grupu shortest path algoritama tj. algoritmi sa rutiranja najkraćeg puta, čiji je cilj da nađu najkraći put od izvora do odredišta.

Bellman-Ford-ov algoritam

Bellman-Ford algoritam koristi jednostavan princip da izračuna najkraći put između dva čvora. Ukoliko su dva čvor *i* i *j* direktno povezano postoji određena cijena između tih linkova d_{ij}; ukoliko nisu onda se cijena postavlja na beskonačnost. Ovaj algoritam pronalazi najkraći put tj. najmanju cijenu između dva čvora u mreži. Bellman-Ford radi na slijedecem principu Bellman-Ford jednačine. Uvedimo slijedece oznake:

D(i,j) = cijena za najbolju rutu od i do krajnjeg j

c(i,j) ili d(i,j) = cijena od rutera i do susjeda j

Ukoliko je *i=j* ili *i* i *j* nisu susjedi cijena se postavlja na beskonačnost. Na osnovu uvedenih oznaka možemo spisati jednačinu kao:

D(i,j)= min {d(i,k) + D(k,j)}, pri cemu k predstavlja susjeda.

Dabismo proračunali D(i,j), čvor i mora da poznaje d(i,k)' i D(k,j)' od svojih susjeda. Problem se javlja kod računanja D(k,j)'. Dabismo proračunali ovu cijenu potrebno je da za svaki čvor *i* prvo pronađemo najkraći put od *i* do *j* koristeći jedan link, D(i,j)[1].

Najkraća cijena puta koji je sačinjen od dva linka je označena sa D(i,j)[2] i mora da zavisi od najkraće cijene prethodnog dijela puta D(i,j)[1], pa možemo da pišemo:

D(i,j)[2]= min{d(i,j)+D(i,j)[1]}

Indukcijom, dobijamo najkrači put za h+1 hop (link) između čvora *i* i *j* kao:

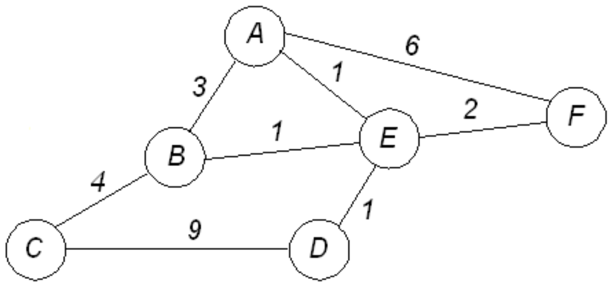
D(i,j)[h+1]=min{d(i,k)+D(k,j)[h]} , k – označava susjede, h=0,1,...

Zaključujemo da prilikom računanja cijene puta od izvorišta *i* do odredišta *j*, pri čemu taj put se sastoji od [h+1] linkova, cijena u velikoj mjeri zavsi od prethodne vrijednosti cijene linka do hopa [h].

Znači, da bismo u potpunosti izračunali krajnju najkraću putanju id izvorišta do odredišta potrebno je da ponavljamo ovaj postupak za h=0,1,2...H, gdje je H maksimalni broj cvorova u mreži.

Primjer Bellman-Ford-ovog algoritma:

Posmatrajmo slijedecu sliku:



Pretpostavimo da C čvor želi da pronađe najkraći put do svake destinacije (čvora) u mreži.

Rješenje:

Prvo se računa najkraći put sa jednim linkom za svaki čvor.

D(i,j)[1]=min{d(i,k) + D(k,j)[0]}

Pošto tražimo cijene linkova za susjedne čvorove svakog čvora k=j, a D(k,j)[0]=beskonačno. Pa relacija postaje slijedeča:

D(i,j)[1]=d(i,j)

Posmatrajući graf dobivamo slijedece razultate:

D(C,B)[1] = 4; D(C,D)[1] = 9;

D(B,A)[1] = 3; D(B,E)[1] = 1; D(B,C)[1] = 4;

$D(D,E)[1] = 1; D(D,C)[1] = 9;$

$D(A,B)[1] = 3; D(A,E)[1] = 1; D(A,F)[1] = 6;$

$D(E,A)[1] = 1; D(E,B)[1] = 1; D(E,D)[1] = 1; D(E,F)[1] = 2;$

$D(F,A)[1] = 6; D(F,E)[1] = 2;$

Potreno je napomenuti da je cijena $D(i,j)[1]=D(j,i)[1]$.

Drugi korak u rješavanju ovog problema je da se računa najkraći put za h=2 linkova.

Tražimo cijenu linka čvora C sa cvorom A:

$D(C,A)\ [2] = \min\{d(C,k) + D(k,A)\}$

$D(C,A)\ [2] = \min\{d(C,B) + D(B,A)[1], d(C,D) + D(D,A)[1]\} = \min\{4 + 3, 9 + \infty\} = 7$

Pošto čvorovi D i A nisu direktno povezani cijena je beskonačna.

Tražimo cijenu linka čvora C sa čvorom E:

$D(C,E)\ [2] = \min\{d(C,k) + D(k,E)\}$

$D(C,E)\ [2] = \min\{d(C,B) + D(B,E)[1], d(C,D) + D(D,E)[1]\} = \min\{4 + 1, 9 + 1\} = 5$

Tražimo cijenu linka čvora C sa čvorom F:

$D(C,F)\ [2] = \min\{d(C,k) + D(k,F)\}$ *za svako k su cvorovi B i D, pa imamo:*

$D(C,F)\ [2] = \min\{d(C,B) + D(B,F)[1], d(C,D) + D(D,F)[1]\} = \min\{4 + \infty, 9 + \infty\} = \infty$

D(D,F)[1] i D(B,F)[1] su cijene jednog linka (h=1). Pošto D i B nisu direktno povezani jednim linkom do čvora F ove cijene su stoga beskonačne.

Dalje tražimo cijenu linka čvora B sa čvorom F:

$D(B,F)\ [2] = \min\{d(B,k) + D(k,F)\}$

$D(B,F)\ [2] = \min\{d(B,A) + D(A,F)[1], d(B,E) + D(E,F)[1], d(B,C) + D(C,F)[1]\} = \min\{3 + 6, 1 + 2, 4 + \infty\} = 3$

Tražimo cijenu linka čvora B sa čvorom E:

$D(B,E)\ [2] = \min\{d(B,k) + D(k,E)\}$

$D(B,E)\ [2] = \min\{d(B,A) + D(A,E)[1], d(B,C) + D(C,E)[1]\} = \min\{3 + 1, 4 + \infty\} = 4$

Tražimo cijenu linka čvora B sa čvorom D:

D(B,D) [2] = min{d(B,k) + D(k,D)}

D(B,D) [2] = min{d(B,E) + D(E,D)[1], d(B,C) + D(C,D)[1]} = min{1 + 1, 4 + 9} = 2

Tražimo cijenu linka čvora D sa čvorom A:

D(D,A) [2] = min{d(D,k) + D(k,A)}

D(D,A) [2] = min{d(D,C) + D(C,A)[1], d(D,E) + D(E,A)[1]} = min{9 + ∞*, 1 + 1} = 2*

Tražimo cijenu linka čvora D sa čvorom F:

D(D,F)[2] = min{d(D,k) + D(k,F)}

D(D,F)[2] = min{d(D,C) + D(C,F)[1], d(D,E) + D(E,F)[1]} = min{9 + ∞*, 1 + 2} = 3*

Potreno je napomenuti da je cijena D(i,j)[2]=D(j,i)[2].

Treci korak u rješavanju ovog problema je da se računa najkraći put na h=3 linkova.

Računanje cijene od čvora C do čvora F:

D(C,F)[3] = min{d(C,B) + D(B,F)[2], d(C,D) + D(D,F)[2]}

D(D,F)[2] je racunato u prethodnoj iteraciji i ono je 3, a D(B,F)[2]=3 tako da slijedi:

D(C,F) = min{4 + 3, 9 + 3} = 7

Pošto je dijametar H=3, sto znači da je maksimalan broj linkova tri. Time smo pronašli najkrači put po Bellman-Fordovom algoritmu za ovu topologiju, i otkili smo vrijednosti cijena između pojedinih rutera i njihovih veza.

Dijkstra algoritam

Dijkstra algoritam vrši rutiranje tako što odabere skup susjednih čvorova da bi se odredio najkraći put od odredišta. Jedna od znacajnih osobina ovog algoritma je da računa najkraći put do svih odredišta od izvora, umjesto za određeni par izvornog i odredišnog čvora, što je veoma korisno u komunikacionim mrežama.

Obicno se koristi Dijkstra algoitam koji koristi slijedece notacije:

- c(i,j) ili d(i,j) – cijena linka od cvora i ka j
- D(v) – trenutna vrijednost cijene puta od izvora do destinacije v
- p(v) – rethodni cvor u putu od izvora do odredišta
- N – set cvorova za koje je poznata cijena putanje

Princip rada Dijkstra agloritma se temelji na sljedecem algoritmu:

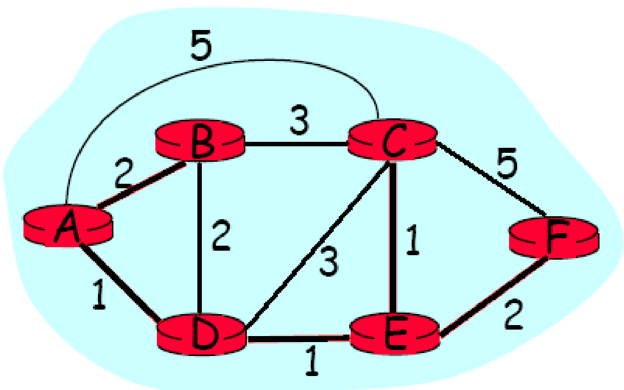
```
Initialization:
  N = {A}
  for all nodes v
    if v adjacent to A
      then D(v) = c(A,v)
      else D(v) = infty

Loop
  find w not in N such that D(w) is a minimum
  add w to N
  update D(v) for all v adjacent to w and not in N:
     D(v) = min( D(v), D(w) + c(w,v) )
  /* new cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
until all nodes in N
```

Ovu petlju ćemo detaljnije razmotriti u narednom primjeru.

Primjer Dijkstra algoritma:

Posmatrajmo slijedeću sliku:



Princip rada Dijkstra-ovog algoritma je dat u sljedecoj tabeli:

Step	start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					

Dabismo rijesili ovaj zadatak ključni dio je da se upoznamo sa principom rada Dijakstra algoritma. Kljucni dio u odredivanju putanje je dio kod od 8 do 15 linije koda.

8 Petlja

9 nađi w koje nije u N tako da je D(w) minimalno

10 dodaj w u N

11 ažuriraj D(v) za sve v obližne sa w koji nisu u N:

12 D(v)=min(D(v), D(w) + c(w,v))

13 I 14 /* nova cijena v je ili stara vrijednost do v ili proračunata vrijednost cijene najkraćeg puta do w + cijena od w do v */

15 dok svi čvorovi u N

Dabismo riješili zadatak pratimo petlju:

<u>U koraku 0</u>, provjerava se povezanost čvora A sa drugim čvorovima u mreži; sto vidimo iz 3, 4, 5, 6 i 7 kolone iz prethodne tabele. Cijena linka sa cvorom B je D(B)=2, a prethodni čvor je p(B)=A. Povezanost čvora A sa čvorom C je preko linka sa cijenom D(C)=5 p(C)=A. Povezanost A sa D je preko linka sa cijenom D(D)=1, a posto nema direktne povezanosti sa cvorovima E i F cijena se postavlja na beskonacnost.

<u>Korak 1</u>

U prethodnom koraku smo vidjeli sa kojim čvorovima je povezan pocetni čvor i kolike su im cijene. U ovom koraku odabiremo najmanju cijenu; to je cijena prema čvoru D i ponovo vršimo provjeru prema Dijakstra algoritmu.

Povezanost čvora D sa čvorom B je sa cijenom D(B)=min (D(B), D(D)+c(D,B)) iz cega zakljucujemo da je D(B)=min(2,3)=2, a prethodni čvor u putanji je čvor p(B)=A.

Povezanost čvora D sa čvorom C je:

D(C)=min(D(C), D(D)+c(D,C))=min(5,1+3)=4, a prethodni čvor je p(C)=D.

Povezanost čvora D sa čvorom E je:

D(E)=min(D(E), D(D)+c(D,E))=min(∞ ,1+1)=2, a prethodni čvor je p(C)=D.

Zaključujemo da je naredni link sa najmanjom cijenom link od D ka E čvoru.

<u>Korak 2</u>

Povezanost čvora E sa čvorom B je:

D(B)=min(D(B), D(E)+c(E,B))=min(2, ∞)=2, a prethodni čvor je p(B)=A.

Povezanost čvora E sa čvorom C je:

D(C)=min(D(C), D(E)+c(E,C))=min(5,2+1)=3, a prethodni čvor je p(C)=E.

Povezanost čvora E sa čvorom F je:

D(F)=min(D(F), D(E)+c(E,F))=min(∞ ,2+2)=4, a prethodni čvor je p(F)=E.

Najmanja dobivena vrijednost je prema čvoru D(B)=2.

Korak 3

Povezanost cvora B sa cvorom C je:

D(C)=min(D(C), D(B)+c(B,C))=min(5,2+1)=3, a prethodni čvor je p(F)=E.

Povezanost čvora B sa čvorom F je:

D(F)=min(D(F), D(B)+c(B,F))=min(∞ ,2+2)=4, a prethodni čvor je p(F)=E.

Najmanja dobivena vrijednost je prema čvoru D(C)=3.

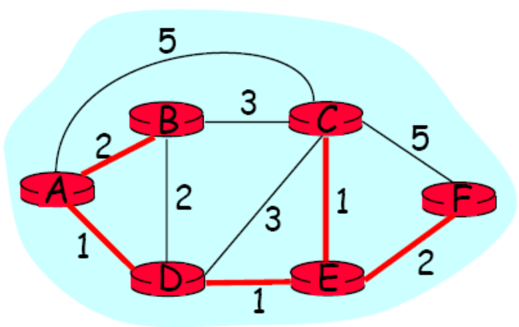
Korak 4

Povezanost čvora C sa čvorom F je:

D(F)=min(D(F), D(C)+c(C,F))=min(5,2+2)=4, a prethodni čvor je p(F)=E.

Najmanja dobivena vrijednost je prema čvoru D(F)=4.

Ovom procedurom smo dosli do zaključka koje su najkraće rute. Te rute su obilježene crvenom bojom na slici.



Problem koji se javlja kod dijakstra algoritma lezi u njegovoj kompleksnosti prilikom racunanja algoritma:

- svaka interacija mora do provjeri sve čvorove
- n*(n+1)/2 poređenja