Sisteme de comutație și rutare

Virgil Dobrotă

Universitatea Tehnică din Cluj-Napoca, România

Virgil.Dobrota@com.utcluj.ro

Curs 11 - Cuprins

30. Algoritmul lui Dijkstra
31. Algoritmul Floyd-Warshall
32. Comparatie intre algoritmii de rutare Bellman-Ford, Dijkstra, Floyd-Warshall

UC Labs
Unified Communications Laboratories

30. Algoritmul lui Dijkstra

Algoritmul lui Dijkstra

1. Fie *d* nodul destinație. Se notează cu *P* mulțimea nodurilor *i* pentru care D_i este deja distanța cea mai scurtă iar Q este mulțimea nodurilor pentru care nu s-a determinat distanța cea mai scurtă.

   Trecerea unui nod din *Q* în *P* se numește relaxare *(Edge Relaxation)* pentru că distanța inițial supra-estimată se aduce la valoarea minimă.

2. $P = \{d\}, Q = \{1,2,..j\}, D_d = 0, D_j = d_{jd}, \forall j \neq d$

   Valoarea supra-estimată a distanței de la nodul *j* la nodul destinație *d* se va numi etichetă *(label)* când devine definitivă.

Algoritmul lui Dijkstra

3. **Primul pas: determinarea următorului nod cel mai apropiat** *(Find the Next Closest Node)*

   Determinarea nodului $i \notin P$ astfel încât $D_i = \min_{j \notin P} D_j$

   Se pune $P = P \cup \{i\}$.

   Dacă P conține toate nodurile, algoritmul se oprește.

   Dacă P nu conține toate nodurile, algoritmul continuă.

4. **Al doilea pas: actualizarea etichetelor** *(Updating of Labels)*

   Pentru toate nodurile

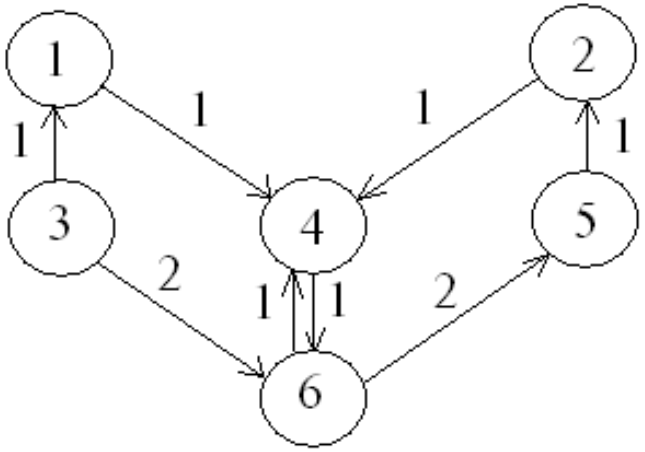
   $$j \in Q, D_j = \min_{i \in P}[D_j, d_{ji} + D_i]$$

   Se trece la primul pas.

Algoritmul lui Dijkstra (III)

PROBLEMA 2:

a) Aplicați pas cu pas algoritmul lui Dijkstra pentru graful direcționat din figura de mai jos, considerând ca 6 este nodul destinație și 1,2,3,4,5 sunt nodurile sursă.
b) Determinați spanning-tree pentru calea cea mai scurtă.



Algoritmul lui Dijkstra (IV)

REZOLVARE:

31. Algoritmul Floyd-Warshall

Algoritmul Floyd-Warshall

NOTAȚIE:

D_{ij}^{n} = distanța (lungimea cea mai scurtă) între nodul i și nodul j, folosind numai nodurile *1,2, ..., n* ca noduri intermediare pe calea cea mai scurtă.

N = numărul de noduri

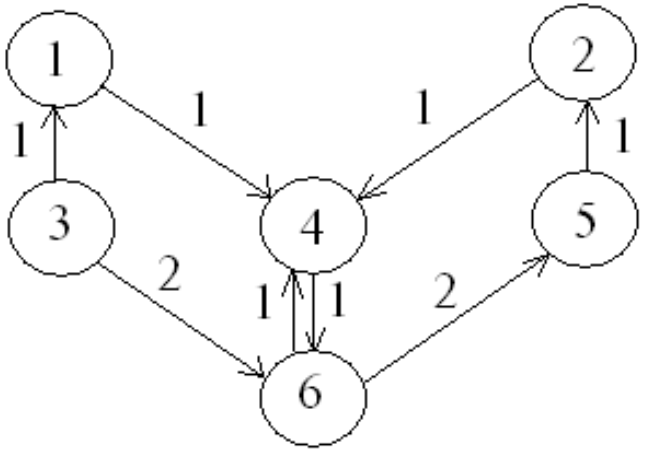
ALGORITMUL:

1. $D_{ij}^{0} = d_{ij}, \forall i \neq j$
2. $D_{ij}^{n+1} = \min[D_{ij}^{n}, D_{i(n+1)}^{n} + D_{(n+1)j}^{n}], \forall i \neq j, n = 0,1,..,N-1$
3. Algoritmul se oprește după maxim N pași.

Algoritmul Floyd-Warshall (II)

PROBLEMA 1:

Aplicați pas cu pas algoritmul Floyd-Warshall pentru graful direcționat din figura de mai jos. Determinați calea cea mai scurtă între toate perechile de noduri.



Algoritmul Floyd-Warshall (III)

REZOLVARE:

Datele inițiale sunt sintetizate în *matricea distanțelor (costurilor)*

	Dest. 1	Dest. 2	Dest. 3	Dest. 4	Dest. 5	Dest. 6
Sursa 1	0	∞	∞	1	∞	∞
Sursa 2	∞	0	∞	1	∞	∞
Sursa 3	1	∞	0	∞	∞	∞
Sursa 4	∞	∞	∞	0	∞	1
Sursa 5	∞	1	∞	∞	0	∞
Sursa 6	∞	∞	∞	1	2	0

Dacă graful nu era direcționat, matricea ar fi fost simetrică față de diagonala principală.

Calculul pas cu pas este dificil de realizat manual, astfel că soluția problemei se va da la laborator.

32. Comparație între algoritmii de rutare Bellman-Ford, Dijkstra, Floyd-Warshall

Comparație între algoritmii de rutare (I)

a) După criteriul de determinare a căii celei mai scurte

Bellman-Ford (BF): -> minimizarea *numărului de arce* dintr-o cale (dacă arcele au aceeași distanță) sau minimizarea *distanței (costului)* căii (dacă arcele nu au aceeași distanță)

Dijkstra (D): -> minimizarea *distanței (costului)* căii.

Floyd-Warshall (FW): -> în funcție de *nodurile intermediare.*

b) După complexitatea de calcul în timp *(Computational Complexity)*

Pentru a descrie comportarea funcțiilor D_i^{h+1} (BF), D_i (D) și D_{ij}^{n+1} (FW) atunci când argumentul funcțiilor tinde spre o anumită valoare sau spre infinit se folosește **notația O mare** *(Big O notation).*

Ideea: *simplificarea funcțiilor cu scopul de a observa rata lor de creștere. Astfel pot să existe două funcții diferite dar care, având aceeași rată de creștere să se reprezinte cu aceeași notație O mare.*

Comparație între algoritmii de rutare (II)

Exemplu:

$f(x) = 3x^2 - 2x + 1 \Rightarrow f(x) \in O(x^2)$, întrucât se neglijează *-2x+1* și factorul 3 de la x^2 pentru $x \to \infty$

$g(x) = 5x^2 + 3 \Rightarrow g(x) \in O(x^2), x \to \infty$

Bellman-Ford (BF): numărul maxim de iterații $h \leq N$

fiecare iterație -> pentru *N-1* noduri sursă

pentru fiecare nod -> *N-1* alternative

BF are complexitatea de calcul în timp $O(N^3)$

Exemplu: Pentru graful studiat la curs *N=6* noduri, numărul de iterații a fost $h = 4 < 6$ și complexitatea maximă ar fi fost $6^3 = 216$ unități de timp.

Comparație între algoritmii de rutare (III)

Dijkstra (D): numărul maxim de iterații *N-1*

fiecare iterație -> număr de operații ~ *N*

D are complexitatea de calcul în timp $O(N^2)$

Exemplu: Pentru graful studiat la curs *N=6* noduri, numărul de iterații a fost $3 < 5$ și complexitatea maximă ar fi fost $6^2 = 36$ unități de timp.

Floyd-Warshall (FW): numărul maxim de iterații *N*

fiecare iterație -> pentru *N* noduri

pentru fiecare nod -> *N-1* noduri intermediare

FW are complexitatea de calcul în timp $O(N^3)$

Exemplu: Nu se pot face comparații exacte întrucât iterațiile de la laboratorul 12 s-au aplicat pe un graf nedirecționat. Oricum complexitatea maximă ar fi fost $6^3 = 216$ unități de timp.

Comparație între algoritmii de rutare (IV)

c) După protocoalele de rutare care implementează algoritmul

Bellman-Ford (BF):

-> aplicat de **RIP** *(Routing Information Protocol)*: foarte popular, ușor de implementat în toate sistemele de operare, dar converge lent și este predispus la bucle de rutare

-> aplicat de **WRP** *(Wireless Routing Protocol):* pentru rutarea unicast pro-activă din rețelele ad-hoc mobile (MANETs).

Dijkstra (D):

-> aplicat de **OSPF** *(Open Shortest Path First)*: mai complex, presupune CPU și memorie suficiente, converge rapid și nu există bucle de rutare

Floyd-Warshall (FW):

-> aplicat de **PEFT** *(Penalizing Exponential Flow SpliTting):* similar cu OSPF, dar penalizează în mod exponențial căile mai lungi.

Concluzii

Concluzii

- **Algoritmul lui Dijkstra** este cel mai rapid și nu există bucle de rutare, fiind recomandat de câte ori este posibil în rețele medii și mari.
- **Algoritmul Bellman-Ford** poate fi totuși folosit în rețele mai mici sau în rețele ad-hoc mobile, cu măsuri de evitare a buclelor de rutare *(se vor studia în anul IV)*
- Este posibil ca uneori **algoritmul Floyd-Warshall** să fie utilizat în locul algoritmului lui Dijkstra.

UC Labs
Unified Communications Laboratories