4 Drumuri minime în grafuri orientate ponderate

4.1 Drumuri minime de sursă unică (de la un vârf dat la celelalte)

4.1.1 Algoritmul lui Dijkstra

Intrare: $G = (V, E, w)$ - graf orientat ponderat cu ponderi **pozitive** ($w : E \longrightarrow \mathbb{R}_+^*$), s - vârf de start
Ieşire:

- vector d de distanţe, cu semnificaţia $d[x] =$ distanţa de la s la vârful x

- vectorul $tata$, reprezentând un arbore al distanţelor faţă de s (din care se poate determina câte un drum minim de la s la fiecare vârf x)

Idee: Fiecare vârf u are asociată o etichetă de distanţă $d[u]$ = costul minim al unui drum de la s la u descoperit până la acel pas = estimare superioară pentru distanţa de la s la u.

La un pas este ales ca vârf curent vârful u care estimat a fi cel mai apropiat de s şi se descoperă noi drumuri către vecinii lui u, actualizându-se etichetele de distanţă ale acestora.

Pseudocod:

```
iniţializează mulţimea vârfurilor nevizitate Q cu V
pentru fiecare u ∈ V executa
    d[u] = ∞; tata[u] = 0
d[s] = 0
cat timp Q ≠ ∅ executa
    u = extrage un vârf cu eticheta d minimă din Q
    pentru fiecare uv ∈ E executa //relaxarea arcului uv
        daca d[u] + w(u, v) < d[v] atunci
            d[v] = d[u] + w(u, v)
            tata[v] = u
scrie d, tata
//Variantă: scrie drum minim de la s la t un vârf t dat folosind tata
```

Exemplu. Complexitate. Detalii implementare: v slide-uri+laborator.

Corectitudinea algoritmului lui Dijkstra

Următoarele observaţii generale privind un drum minim între două vârfuri sunt evidente, dar importante în demonstrarea corectitudinii algoritmilor de drumuri minime.

Observaţie 4.1.
1. *Dacă P este un drum minim de la s la u, atunci P este drum elementar.*
2. *Dacă P este un drum minim de la s la u şi z este un vârf al lui P, atunci subdrumul lui P de la s la z este drum minim de la s la z.*
   Mai general, dacă x şi y sunt două vârfuri din P, atunci subdrumul lui P de la x la y este drum minim de la x la y.

Lema 4.2. *Pentru orice $u \in V$, la orice pas al algoritmului lui Dijkstra avem:*

(a) dacă $d[u] < \infty$, atunci există un drum de la s la u în G de cost $d[u]$ şi acesta se poate determina din vectorul tata, mai exact $tata[u]$= predecesorul lui u pe un drum de la s la u de cost $d[u]$.

(b) $d[u] \geq \delta(s, u)$

Demonstraţie. a) Demonstrăm afirmaţia prin inducţie după numărul de iteraţii (execuţii ale ciclului "cat timp").

Iniţial singura etichetă finită este $d[s] = 0$, corespunzătoare drumului $[s]$. La prima iteraţie este extras din Q vârful s şi pentru el afirmaţia se verifică.

Presupunem afirmaţia adevărată până la un pas.

Fie $u \in Q$ vârful extras la iteraţia curentă. Fie v cu $uv \in E$ un vecin al lui u căruia i se modifică eticheta la acest pas prin relaxarea arcului uv. Avem:

- după relaxarea arcului uv: $d[v] = d[u] + w(uv)$; $tata[v] = u$
- din ipoteza de inducţie există P un drum de la s la u de cost $d[u]$.

Atunci drumul $R = [s \overset{P}{-} u, v]$ este un drum de la s la v de cost

$$w(R) = w(P) + w(uv) = d[u] + w(uv) = d[v]$$

şi $u = tata[v]$ este predecesorul lui v pe acest drum.

$\square$

Corolar 4.3. *Dacă la un pas al algoritmului pentru un vârf u avem relaţia $d[u] = \delta(s, u)$, atunci $d[u]$ nu se mai modifică până la final.*

Demonstraţie. Afirmaţia rezultă din Lema anterioară (nu există drum de la s la u cu cost mai mic decât cel minim, adică decât $\delta(s, u)$). $\square$

Teorema 4.4. *Fie $G = (V, E, w)$ un graf orientat ponderat cu $w : E \longrightarrow \mathbb{R}_+^*$ şi $s \in V$ fixat.*

La finalul algoritmul lui Dijkstra avem:

$$d[u] = \delta(s, u) \text{ pentru orice } u \in V$$

şi $tata$ memorează un arbore al distanţelor faţă de s.

Demonstraţie. Vom demonstra prin inducţie după numărul de iteraţii (execuţii ale ciclului "cat timp") că după fiecare iteraţie avem $d[x] = \delta(s, x)$ pentru orice $x \in V - Q$ (eticheta vârfurilor deja selectate este corect calculaltă).

Iniţial avem $d[s] = \delta(s, s)$ şi primul vârf extras din Q este s, deci afirmaţia se verifică după prima iteraţie.

Presupunem că afirmaţia este adevărată până la iteraţia curentă: $d[x] = \delta(s, x)$ pentru orice $x \in V - Q$.

Fie $u \in Q$ vârful extras la această iteraţie. Demonstrăm că $d[u] = \delta(s, u)$ (adică eticheta lui u este corect calculată, fiind egală cu distanţa de la s la u).

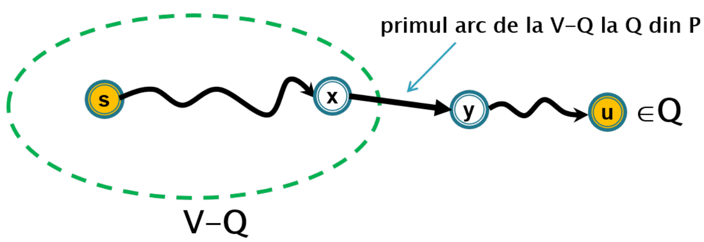

Figura 6

Dacă u nu este accesibil din s atunci, din Lema 4.2, $d[u] = \infty = \delta(s, u)$.

Altfel, fie P un drum minim de la s la u (avem $w(P) = \delta(s, u)$). Deoarece P are o extremitate $s \in V - Q$ şi cealaltă extremitate $u \in Q$, rezultă că P conţine un arc cu o extremitate în $V - Q$ şi cealaltă în Q. Fie xy primul astfel de arc din P (cu $x \in V - Q$ şi $y \in Q$) - v. fig. 6.

Din ipoteza de inducţie şi Observaţia 4.1 avem

$$d[x] = \delta(s, x) = w([s \overset{P}{-} x]).$$

La iteraţia la care a fost extras din Q vârful x, după relaxarea arcului xy avem:

$$d[y] \leq d[x] + w(xy) = w([s \overset{P}{-} x]) + w(xy) = w([s \overset{P}{-} y]).$$

Deoarece arcele au capacităţi pozitive, avem $w([s \overset{P}{-} y]) \leq w(P)$ şi deci

$$d[y] \leq w(P) = \delta(s, u) \leq d[u].$$

Dar, deoarece la pasul curent şi u şi y sunt în Q, din modul în care algoritmul alege vârful u avem

$$d[u] \leq d[y].$$

Rezultă că $d[u] = d[y] = w(P) = \delta(s, u)$ (şi P are vârfuri interne doar din mulţimea vârfurilor deja selectate).

Din Lema 4.2, pentru fiecare u, $d[u]$ este ponderea drumului de la s la u memorat în vectorul $tata$, deci acesta corespunde unui arbore al distanţelor faţă de s.

□

4.1.2 Drumuri minime în grafuri fără circuite (DAGs= Directed Acyclic Graphs)

Intrare: $G = (V, E, w)$ - graf orientat ponderat (ponderi reale) **fără circuite**, s - vârf de start
Ieşire:

- vector d de distanţe, cu semnificaţia $d[x]$ = distanţa de la s la vârful x