Cursul 10/11: Grafuri ponderate

Căi cu lungime ponderată minimă. Algoritmul lui Bellman-Ford, algoritmul lui Dijkstra, și algoritmul lui Floyd-Warshall

noiembrie 2020

Un graf ponderat este un graf $G = (V, E)$ cu o funcție $w : E \to \mathbb{R}$ care atribuie o pondere $w(e)$ fiecărei muchii $e \in E$.

- Ponderile pot reprezenta distanțe dintre noduri dar și alte metrici precum timpi, costuri, penalizări, pierderi sau alte cantități care se acumulează liniar de-a lungul unui drum și pe care vrem să le minimizăm.
- Vom studia doar **grafuri ponderate simple**, adică
  - fără bucle
  - cu cel mult o muchie de la un nod la alt nod
- Vom scrie $w(x, y)$ în loc de $w(e)$ atunci când e este muchia $x{-}y$ sau arcul $x{\to}y$.
- Deasemenea, vom presupune că $w(x, x) = 0$ și $w(x, y) = +\infty$ dacă nu există muchie de la x la y.

Scriem $x \overset{\pi}{\rightsquigarrow} y$ pentru a indica faptul că π este o cale de la x la y.

Scriem $x \overset{\pi}{\rightsquigarrow} y$ pentru a indica faptul că π este o cale de la x la y.

Lungimea ponderată a unei liste de noduri $\pi = [x_1, x_2, \ldots, x_n]$ este

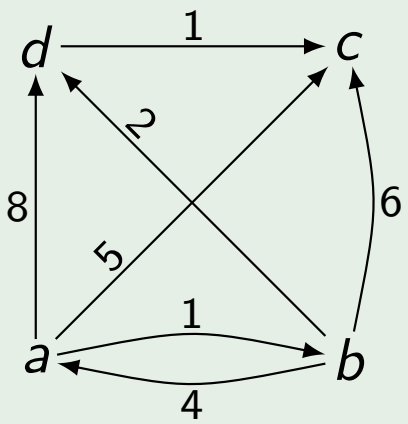
$$\text{length}_w(\pi) = \sum_{i=1}^{n-1} w(x_i, x_{i+1}).$$

Dacă $n = 1$ atunci $\pi = [x_1]$ și $\text{length}_w(\pi) = 0$.

Distanța ponderată de la x la y în G este

$$\delta_w(x, y) = \begin{cases} +\infty & \text{dacă } x \not\rightsquigarrow y, \\ \min\{\text{length}_w(\pi) \mid x \overset{\pi}{\rightsquigarrow} y\} & \text{în caz contrar.} \end{cases}$$

Exemplu



$\delta_w(x,y)$	$y=a$	$y=b$	$y=c$	$y=d$
$x=a$	0	1	4	3
$x=b$	4	0	3	2
$x=c$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
$x=d$	$+\infty$	$+\infty$	$+\infty$	$+\infty$

$\text{length}_w([a,b,c]) = 7,$
$\text{length}_w([a,d,c]) = 9,$
$\text{length}_w([a,b,d,c]) = 4.$

Vom prezenta soluții algoritmice la problemele următoare:

1. Să se determine căi cu lungime ponderată minimă de la un nod sursă s la toate nodurile la care se poate ajunge din s.
2. Să se determine căi cu lungime ponderată minimă de la x la y pentru toate perechile de noduri conectate $x \rightsquigarrow y$.

Vom prezenta soluții algoritmice la problemele următoare:

1. Să se determine căi cu lungime ponderată minimă de la un nod sursă s la toate nodurile la care se poate ajunge din s.
2. Să se determine căi cu lungime ponderată minimă de la x la y pentru toate perechile de noduri conectate $x \rightsquigarrow y$.

Remarcă

Dacă $\pi = [x_1, x_2, \ldots, x_k]$ este o cale de la x_1 la x_k cu $\text{length}_w(\pi) = \delta_w(x_1, x_k)$, atunci pentru toți $1 \leq i \leq j \leq n$:

- Dacă $\pi_{i,j} = [x_i, x_{i+1}, \ldots, x_j]$ atunci $\text{length}_w(\pi_{i,j}) = \delta(x_i, k_j)$.

Altfel spus, *toate subcăile unei căi cu lungime ponderată minimă au lungime ponderată minimă*.

Cicluri cu lungimi ponderate negative

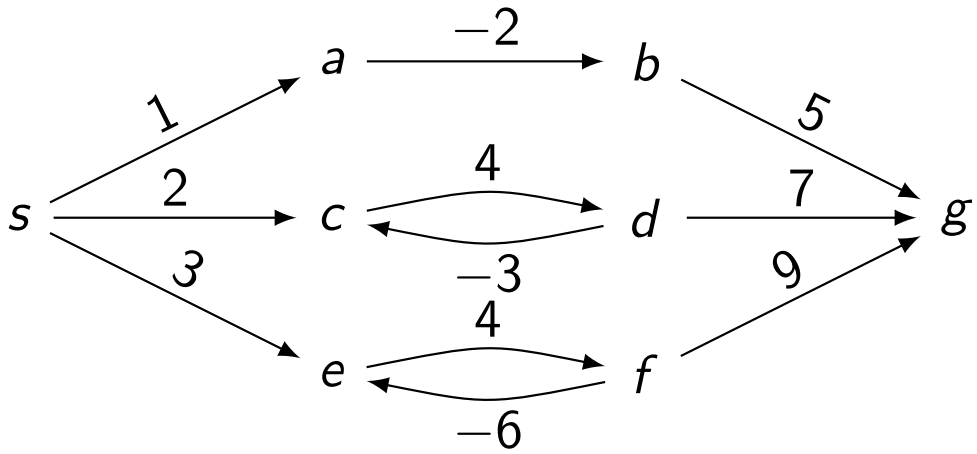
Muchiile e cu $w(e) < 0$ pot forma cicluri cu lungimi ponderate negative $\Rightarrow$ pentru orice noduri x, y:

- Dacă există un nod z într-un ciclu c cu lungime ponderată negativă și $x \rightsquigarrow z \rightsquigarrow y$ atunci nu există $x \overset{\pi}{\rightsquigarrow} y$ cu lungime ponderată minimă fiindcă traversarea repetată a lui c produce căi cu lungime ponderată ce tinde la $-\infty$. În acest caz definim $\delta_w(x, y) = -\infty$.
- În caz contrar, $\delta_w(x, y) \in \mathbb{R}$ și există $x \overset{\pi}{\rightsquigarrow} y$ cu $\text{length}_w(\pi) = \delta_w(x, y)$.

Cicluri cu lungimi ponderate negative

Exemplu

Digraful de mai jos are cicluri cu lungime ponderată negativă:



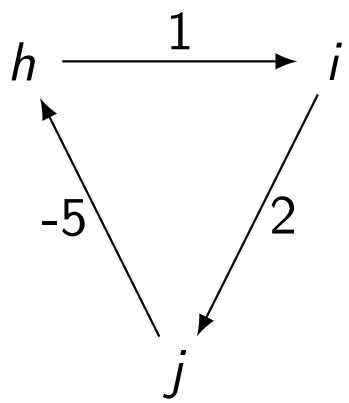
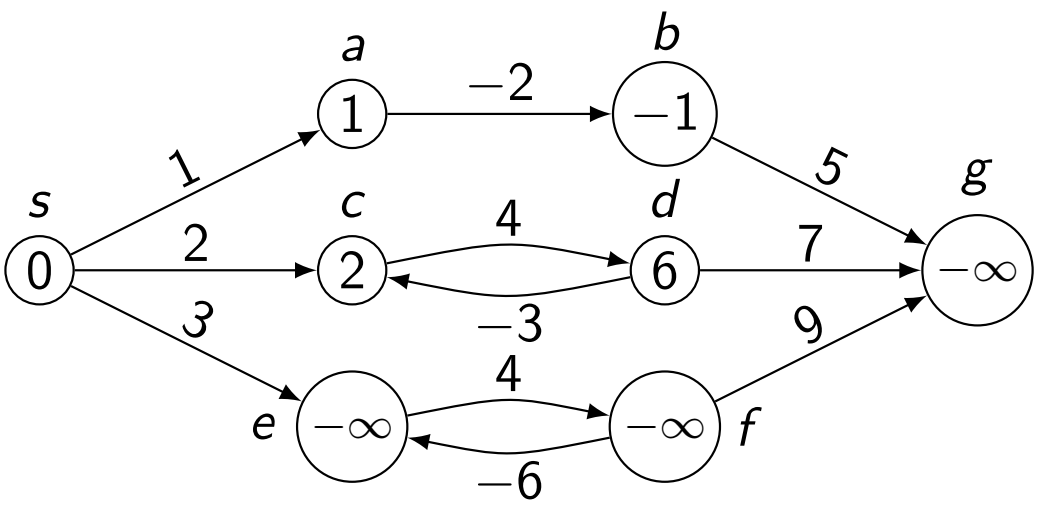
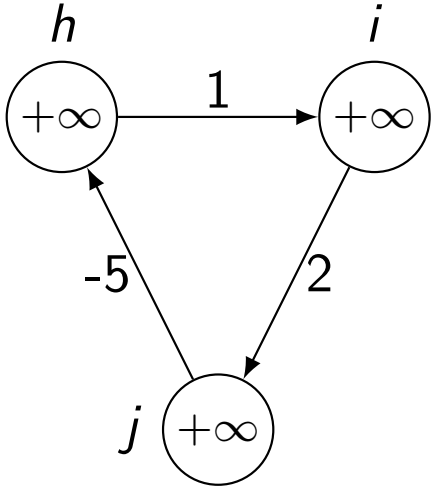


Figura următoare indică valorile lui $\delta_w(s, x)$ pentru toate nodurile x:





Căi cu lungime ponderată minimă

Observații

Fie $x \overset{\pi}{\rightsquigarrow} y$ o cale cu lungime ponderată minimă. Observăm că

Fie $x \overset{\pi}{\rightsquigarrow} y$ o cale cu lungime ponderată minimă. Observăm că

1. π nu poate conține un ciclu cu lungime ponderată strict negativă pentru că ar rezulta că $\delta_w(x, y) = -\infty$.

Fie $x \overset{\pi}{\rightsquigarrow} y$ o cale cu lungime ponderată minimă. Observăm că

1. π nu poate conține un ciclu cu lungime ponderată strict negativă pentru că ar rezulta că $\delta_w(x, y) = -\infty$.
2. π nu poate conține un ciclu cu lungime ponderată strict pozitivă pentru că dacă-l eliminăm din π obținem $x \overset{\pi'}{\rightsquigarrow} y$ cu $\text{length}_w(\pi') < \text{length}_w(\pi) = \delta_w(x, y)$, contradicție.

Căi cu lungime ponderată minimă

Observații

Fie $x \overset{\pi}{\rightsquigarrow} y$ o cale cu lungime ponderată minimă. Observăm că

1. π nu poate conține un ciclu cu lungime ponderată strict negativă pentru că ar rezulta că $\delta_w(x, y) = -\infty$.
2. π nu poate conține un ciclu cu lungime ponderată strict pozitivă pentru că dacă-l eliminăm din π obținem $x \overset{\pi'}{\rightsquigarrow} y$ cu $\text{length}_w(\pi') < \text{length}_w(\pi) = \delta_w(x, y)$, contradicție.
3. Putem presupune că π nu conține cicluri cu lungime ponderată 0 fiindcă ele se pot elimina din π fără să-i afecteze lungimea ponderată.

Fie $x \overset{\pi}{\rightsquigarrow} y$ o cale cu lungime ponderată minimă. Observăm că

1. π nu poate conține un ciclu cu lungime ponderată strict negativă pentru că ar rezulta că $\delta_w(x, y) = -\infty$.
2. π nu poate conține un ciclu cu lungime ponderată strict pozitivă pentru că dacă-l eliminăm din π obținem $x \overset{\pi'}{\rightsquigarrow} y$ cu $\text{length}_w(\pi') < \text{length}_w(\pi) = \delta_w(x, y)$, contradicție.
3. Putem presupune că π nu conține cicluri cu lungime ponderată 0 fiindcă ele se pot elimina din π fără să-i afecteze lungimea ponderată.

Deci ne putem limita să căutăm doar căi aciclice $i \overset{\pi}{\rightsquigarrow} j$ cu lungime ponderată minimă. Căile aciclice conțin cel mult $|V| = n$ noduri distincte, deci cel mult $n - 1$ muchii.

Ambii algoritmi calculează reprezentarea cu predecesori a unui arbore T_s cu rădăcina s astfel încât

1. mulțimea de noduri a lui T_s este $S_s = \{x \in V \mid s \rightsquigarrow x\}$
2. pentru fiecare $s \in S_s$, lista de noduri pe ramura de la s la x în T_s este o cale cu lungime ponderată minimă de la s la x în G.

Un astfel de arbore se numește arbore de căi cu lungimi ponderate minime de la s în G.

Ambii algoritmi calculează reprezentarea cu predecesori a unui arbore T_s cu rădăcina s astfel încât

1. mulțimea de noduri a lui T_s este $S_s = \{x \in V \mid s \rightsquigarrow x\}$
2. pentru fiecare $s \in S_s$, lista de noduri pe ramura de la s la x în T_s este o cale cu lungime ponderată minimă de la s la x în G.

Un astfel de arbore se numește arbore de căi cu lungimi ponderate minime de la s în G.

- **Algoritmul lui Dijkstra** este definit pentru grafuri ponderate cu $w(e) > 0$ pentru toate muchiile e.

Ambii algoritmi calculează reprezentarea cu predecesori a unui arbore T_s cu rădăcina s astfel încât

1. mulțimea de noduri a lui T_s este $S_s = \{x \in V \mid s \rightsquigarrow x\}$
2. pentru fiecare $s \in S_s$, lista de noduri pe ramura de la s la x în T_s este o cale cu lungime ponderată minimă de la s la x în G.

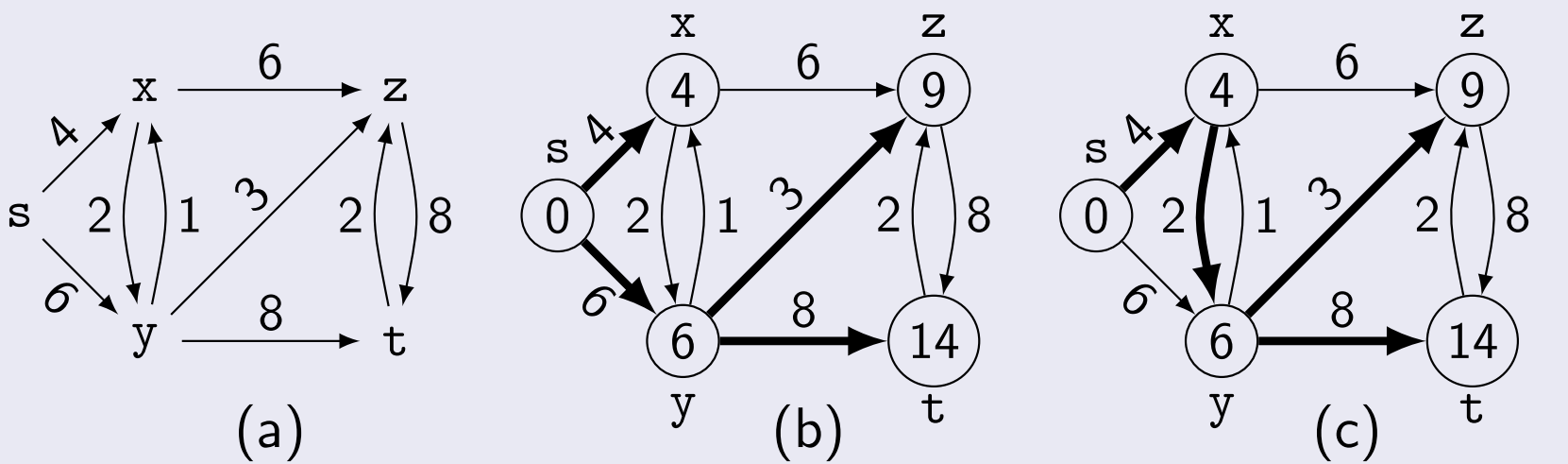
Un astfel de arbore se numește arbore de căi cu lungimi ponderate minime de la s în G.

- **Algoritmul lui Dijkstra** este definit pentru grafuri ponderate cu $w(e) > 0$ pentru toate muchiile e.
- **Algoritmul lui Bellman-Ford** este definit pentru cazul general, când putem avea muchii e cu $w(e) < 0$.
  - Detectează eventualele cicluri cu lungime ponderată negativă la care se poate ajunge din s. În acest caz, returnează `false` pentru a semnala existența unui astfel de ciclu și abandonează calculul arborelui T_s.

Căi cu lungime ponderată minimă de la un nod sursă s

Exemplu ilustrat

Digraful ponderat din figura (a) are 2 arbori de căi cu lungimi ponderate minime cu rădăcina s. Figurile (b) și (c) ilustrează muchiile celor doi arbori cu linii îngroșate, iar valoarea lui $\delta_w(s, x)$ este indicată în interiorul fiecărui nod x.



Algoritmii operează cu:

1. reprezentarea cu predecesori a unui arbore A_s cu rădăcina s și mulțimea de noduri V. Vom presupune că, pentru fiecare $x \in V$, π_x este lista de noduri pe ramura de la s la x în A_s.
2. $\mathtt{d}[x]$: o margine superioară a lui $\text{length}_w(\pi_x)$:

$$\forall x \in V . \delta_w(s, x) \leq \text{length}_w(s, x) \leq \mathtt{d}[x].$$

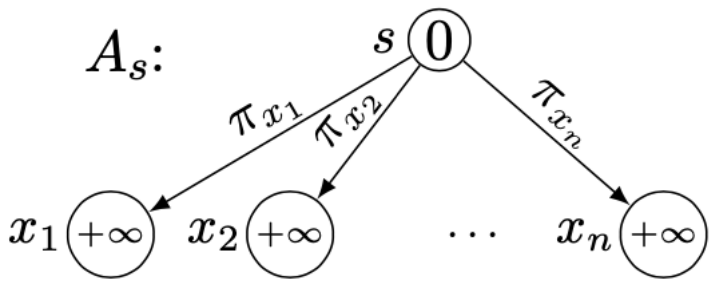
Algoritmii operează cu:

1. reprezentarea cu predecesori a unui arbore A_s cu rădăcina s și mulțimea de noduri V. Vom presupune că, pentru fiecare $x \in V$, π_x este lista de noduri pe ramura de la s la x în A_s.
2. $\mathtt{d}[x]$: o margine superioară a lui $\text{length}_w(\pi_x)$:

$$\forall x \in V . \delta_w(s, x) \leq \text{length}_w(s, x) \leq \mathtt{d}[x].$$

Valorile inițiale sunt

- $\mathtt{p}[s] = \mathtt{null}$ și $\mathtt{p}[x] = s$ pentru toți $x \in V - \{s\}$, și
- $\mathtt{d}[s] = 0$ și $\mathtt{d}[x] = +\infty$ pentru toți $x \in V - \{s\}$.



unde $V = \{s, x_1, x_2, \ldots, x_n\}$.
Valorile lui $\mathtt{d}[x]$ sunt indicate în interiorul nodurilor respective.

Valorile lui `d[]` și `p[]` se modifică efectuând un număr finit de relaxări de muchii și garantează că, atunci când se termină:

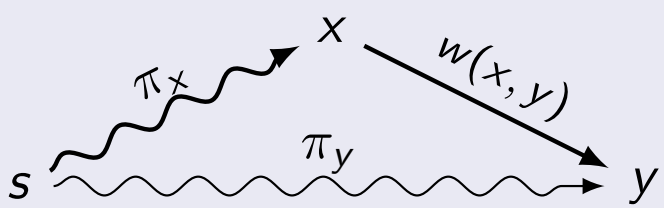
- A_s este arbore de căi cu lungimi ponderate minime de la s în G.
- $\mathrm{d}[x] = \delta_w(s, x)$ pentru toți $x \in V$.

Relaxarea unei muchii de la x la y

Dacă $\mathrm{d}[x] + w(x, y) < \mathrm{d}[y]$ și considerăm calea $\pi'_y = s \overset{\pi_x}{\rightsquigarrow} x \to y$ atunci

$$\delta_w(x, y) \leq \mathrm{length}_w(\pi'_y) = \mathrm{length}_w(\pi_x) + w(x, y) \leq \mathrm{d}[x] + w(x, y) < \mathrm{d}[y]$$

$\Rightarrow$ putem înlocui $\mathrm{p}[y]$ cu $\mathrm{p}[x]$ și $\mathrm{d}[y]$ cu $\mathrm{d}[x] + w(x, y)$.




```
relax(x,y) {
  if (d[x] + w(x,y) < d[y]) {
      p[y] = x; d[y] = d[x] + w(x,y);
  }
}
```

Algoritmul lui Bellman-Ford

Pseudocod

```
boolean BellmanFord(G,s) {
  initializeaza(G,s);
  for i=1 to G.V()-1
     foreach x ∈V(G)
       for (y:adj[x])
           relax(x,y);
  foreach x ∈V(G)
     for (y:adj[x])
         if (d[x] > d[y]+w(x,y))
             return false;
  return true;
}
```

Algoritmul lui Bellman-Ford

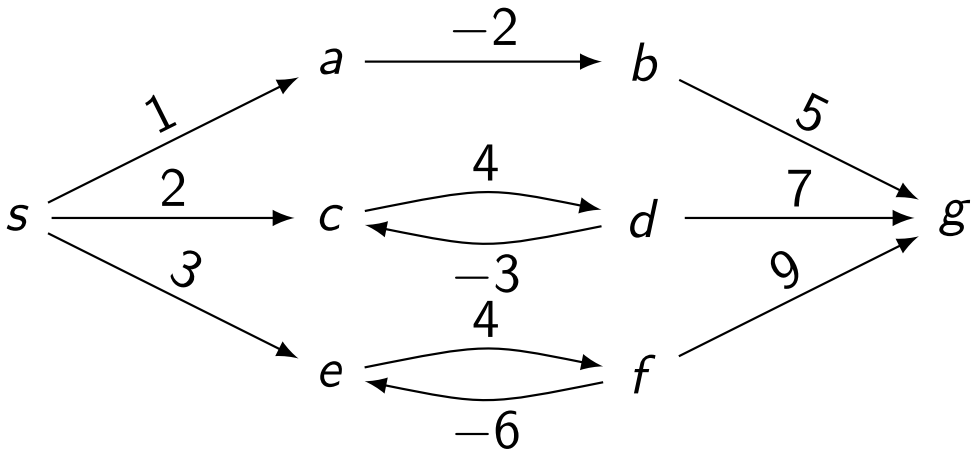
Pseudocod

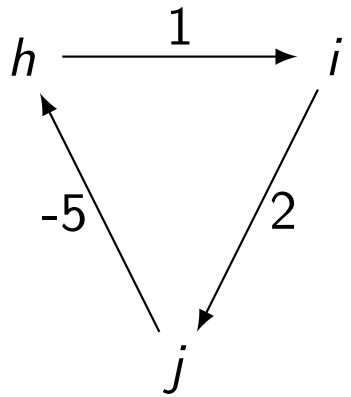
```
boolean BellmanFord(G,s) {
  initializeaza(G,s);
  for i=1 to G.V()-1
     foreach x ∈V(G)
       for (y:adj[x])
           relax(x,y);
  foreach x ∈V(G)
     for (y:adj[x])
         if (d[x] > d[y]+w(x,y))
             return false;
  return true;
}
```

Complexitate (timp de execuție): $O(n \cdot m^2)$

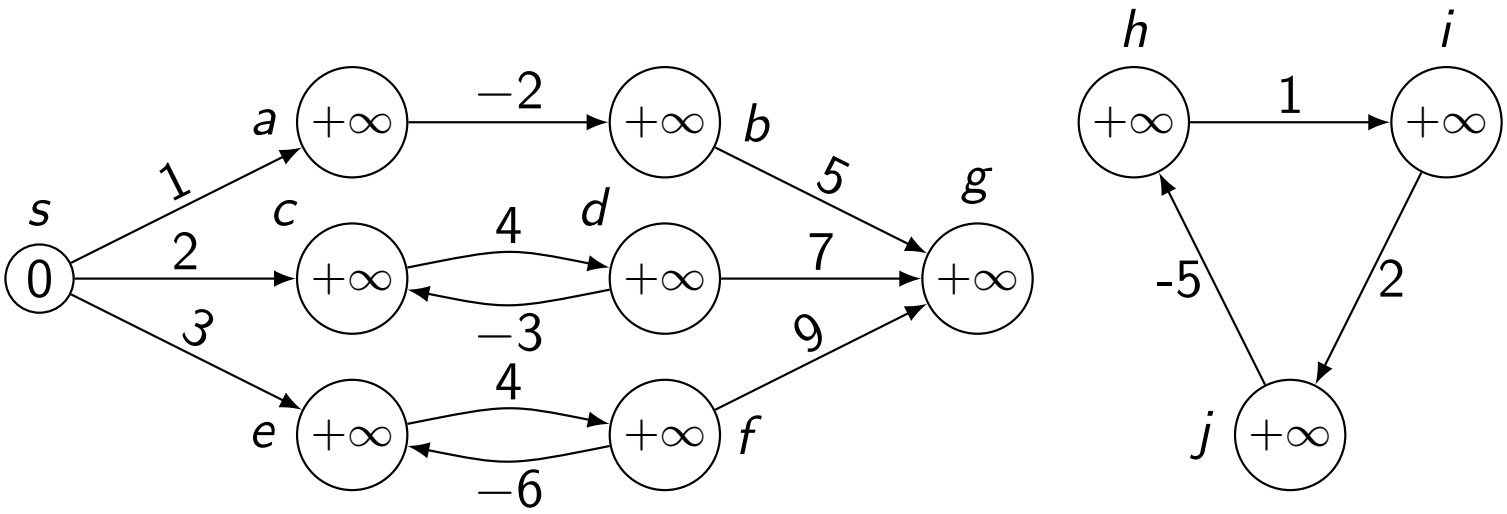
Algoritmul lui Bellman-Ford

Exemplu ilustrat

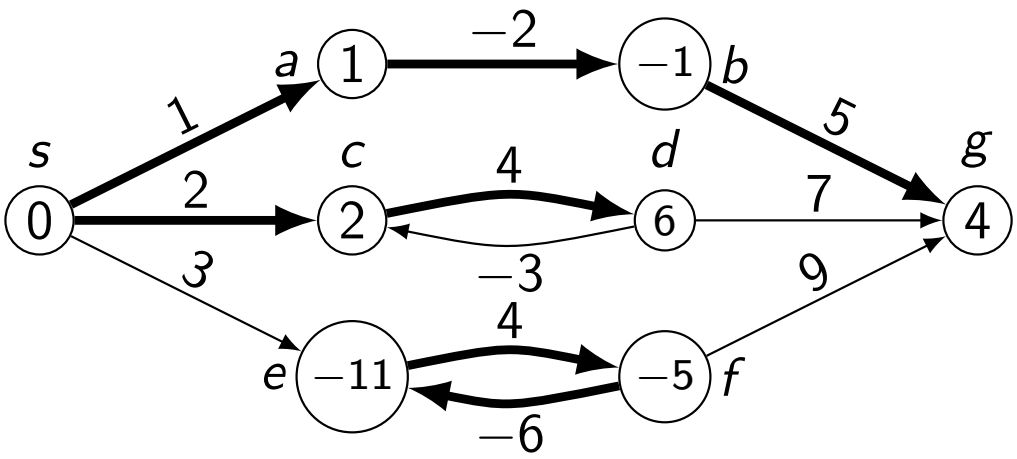


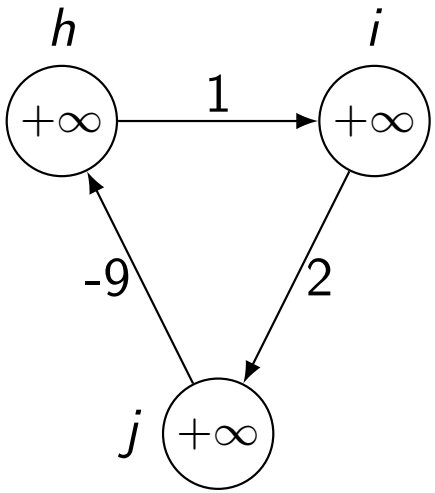


După pasul de inițializare:

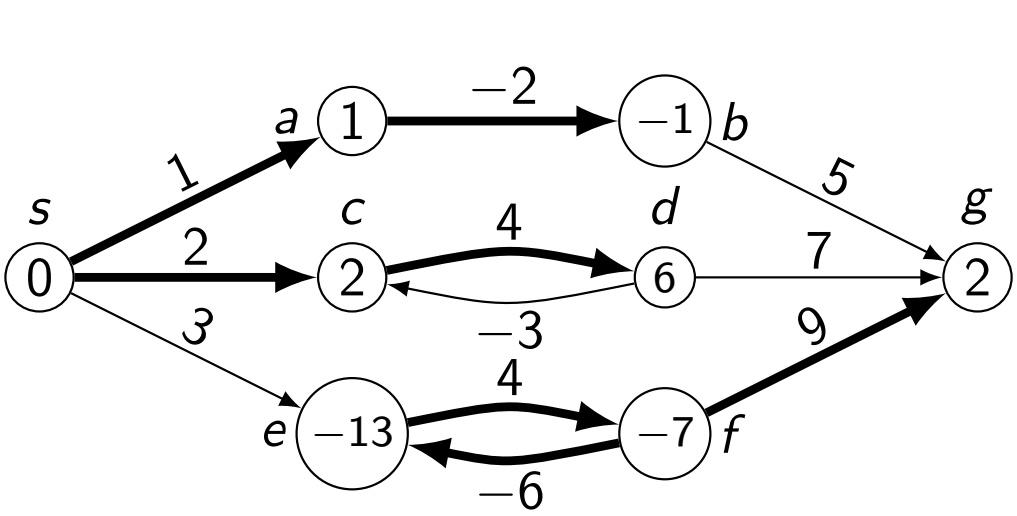


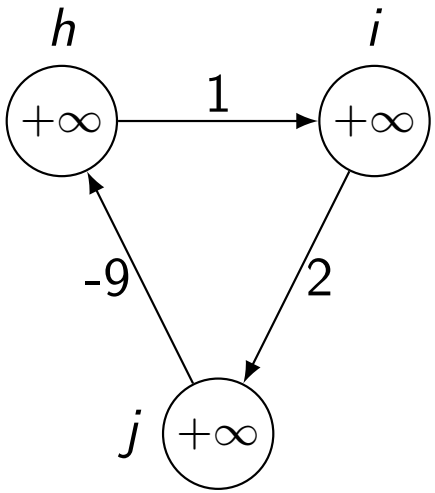
După a 6-a buclă `for`:



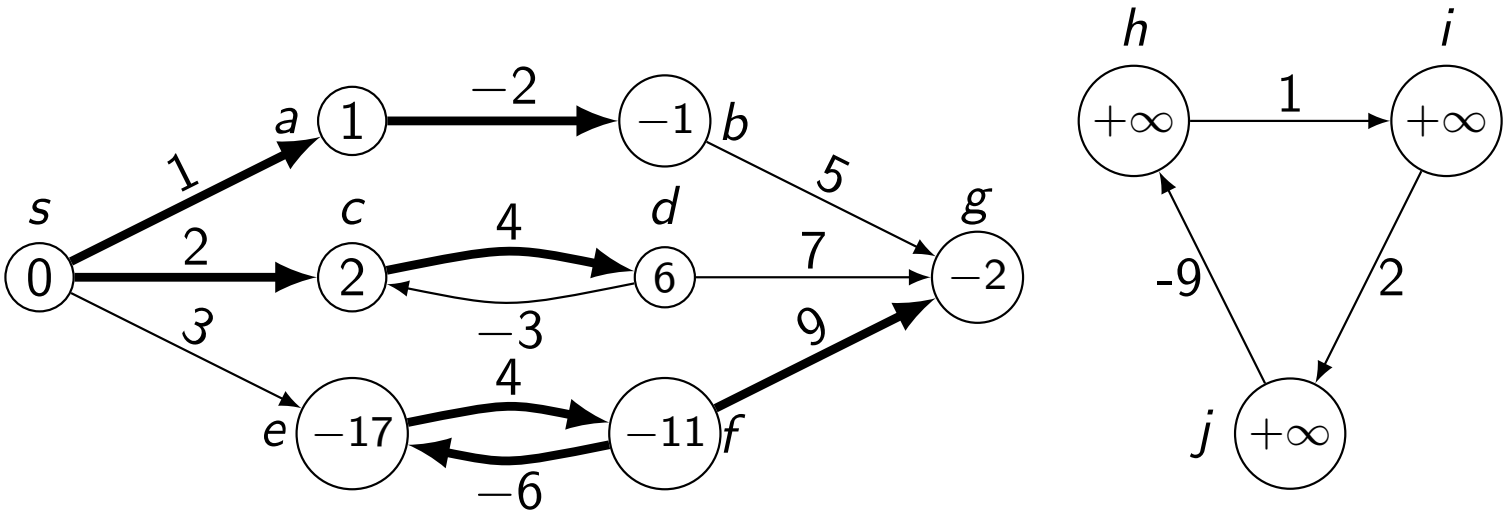


După a 8-a buclă `for`:





După a 10-a buclă `for`:



Algoritmul returnează `false` pentru că detectează

$$\mathrm{d}[f] = -11 > \mathrm{d}[e] + w(e, f).$$

Algoritmul lui Dijkstra

Pseudocod

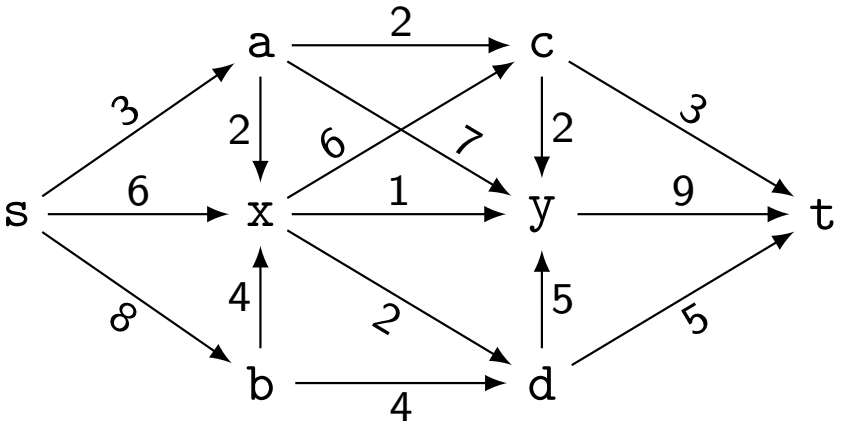
```
void Dijkstra(G,s) {
  initializeaza(G,s);
  Q=mulţimea de noduri a lui G;
  while (!Q.isEmpty()) {
     extrage u cu d[u] = min{d[x] | x ∈ Q} din Q;
     for (v:G.adj(u))
        if (Q.contains(v))
           relax(u,v);
  }
}
```

Algoritmul lui Dijkstra

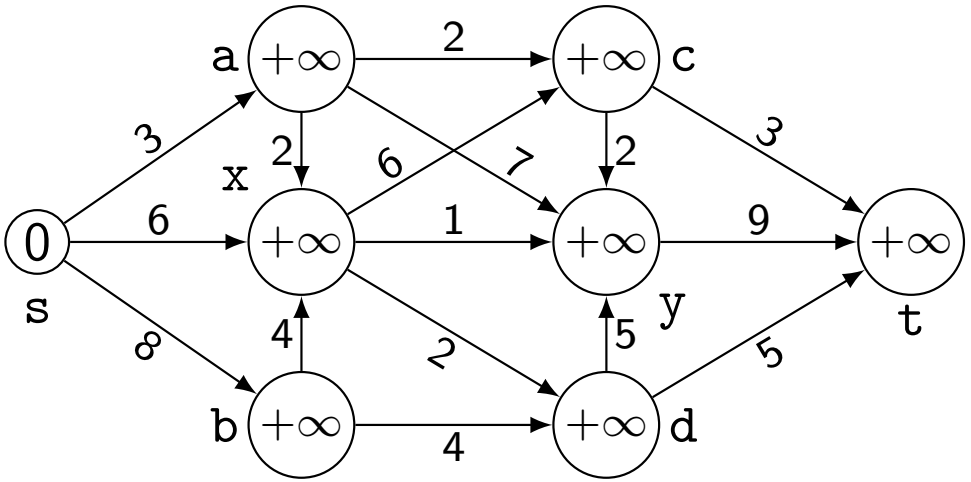
Pseudocod

```
void Dijkstra(G,s) {
  initializeaza(G,s);
  Q=mulţimea de noduri a lui G;
  while (!Q.isEmpty()) {
     extrage u cu d[u] = min{d[x] | x ∈ Q} din Q;
     for (v:G.adj(u))
        if (Q.contains(v))
           relax(u,v);
  }
}
```

Complexitate (timp de execuție): $O(n^2)$

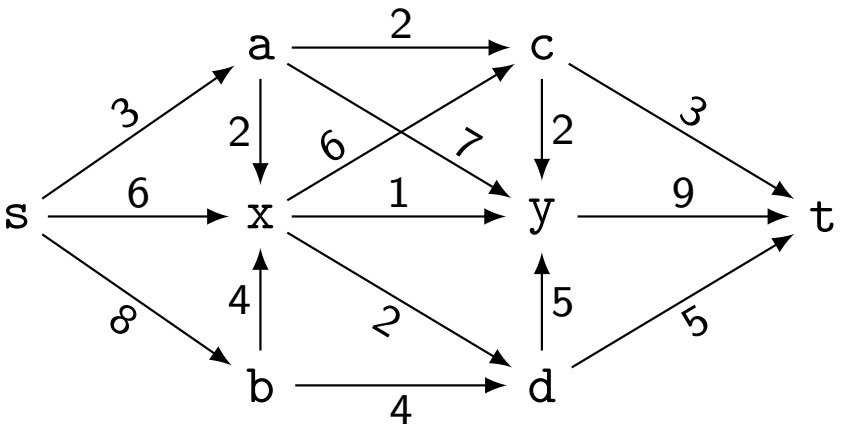


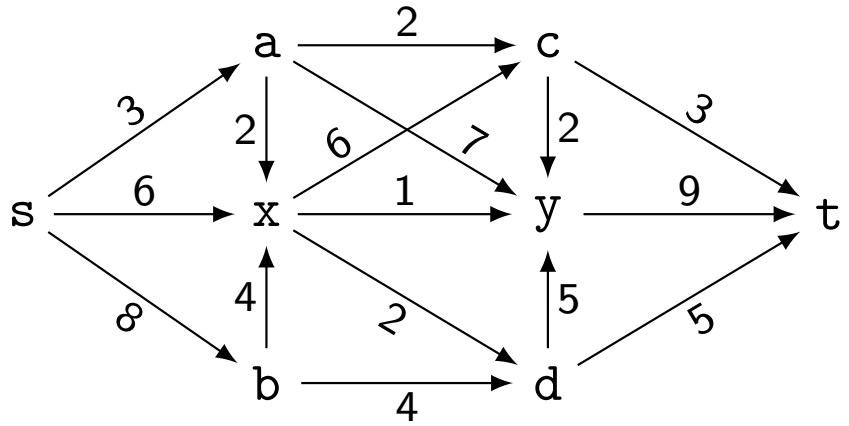
După inițializare avem



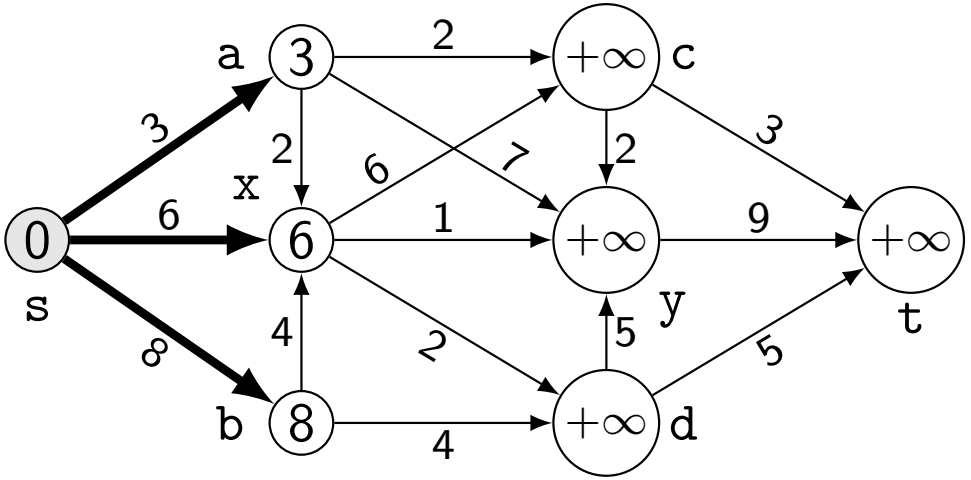
Algoritmul lui Dijkstra

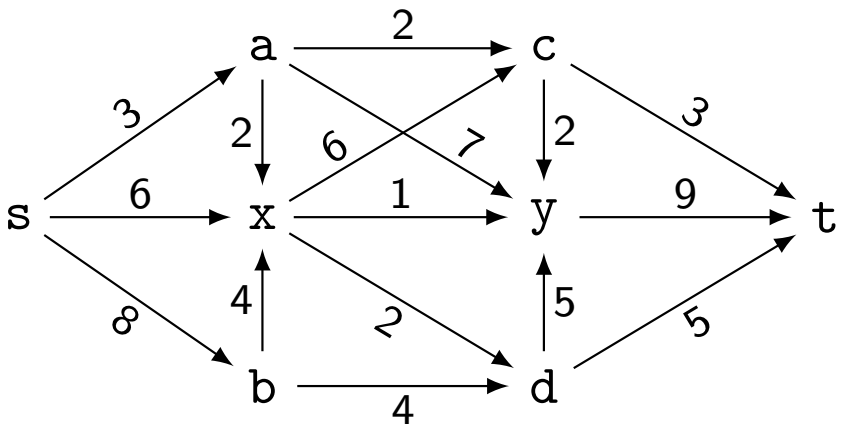
Exemplu ilustrat



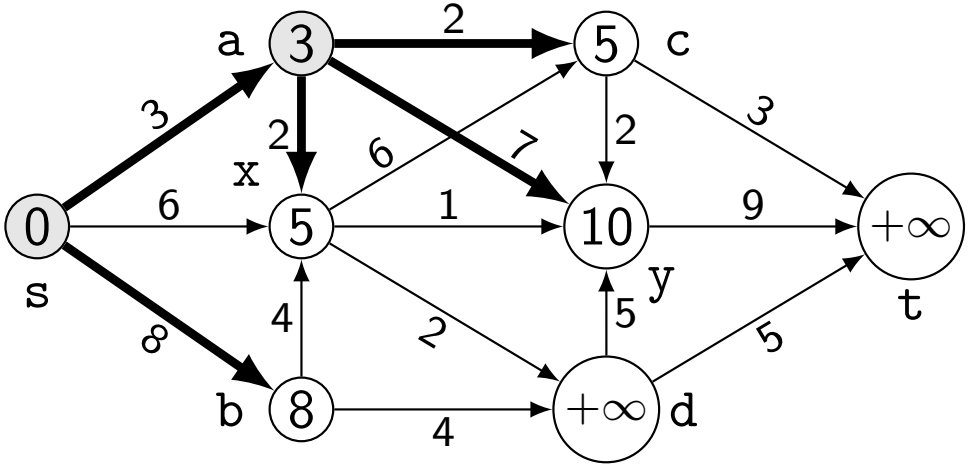


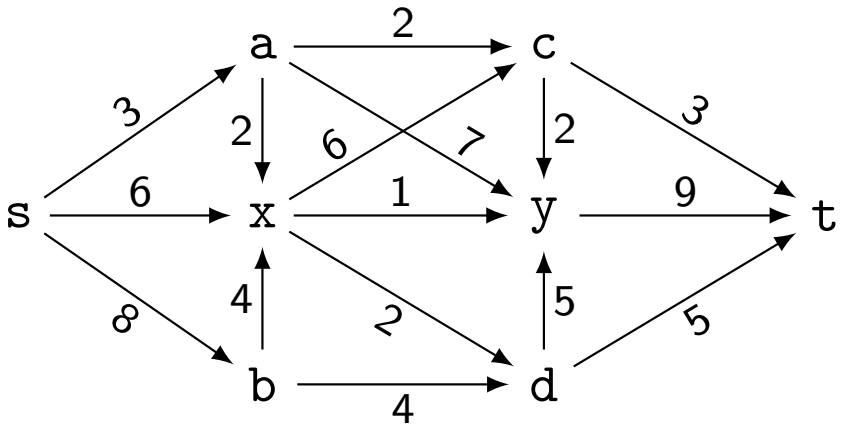
După relaxarea muchiilor din s avem



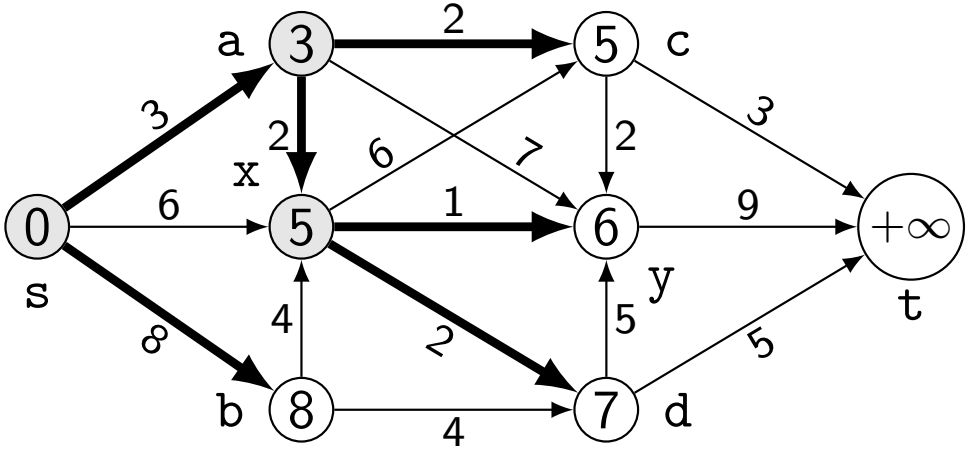


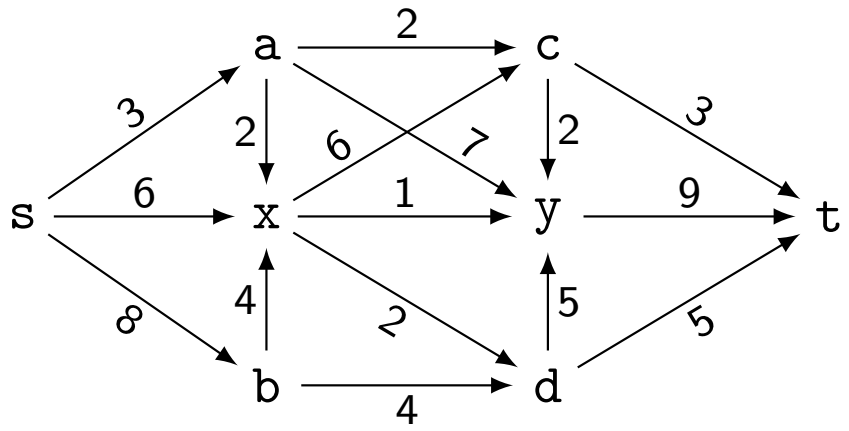
După relaxarea muchiilor din a avem



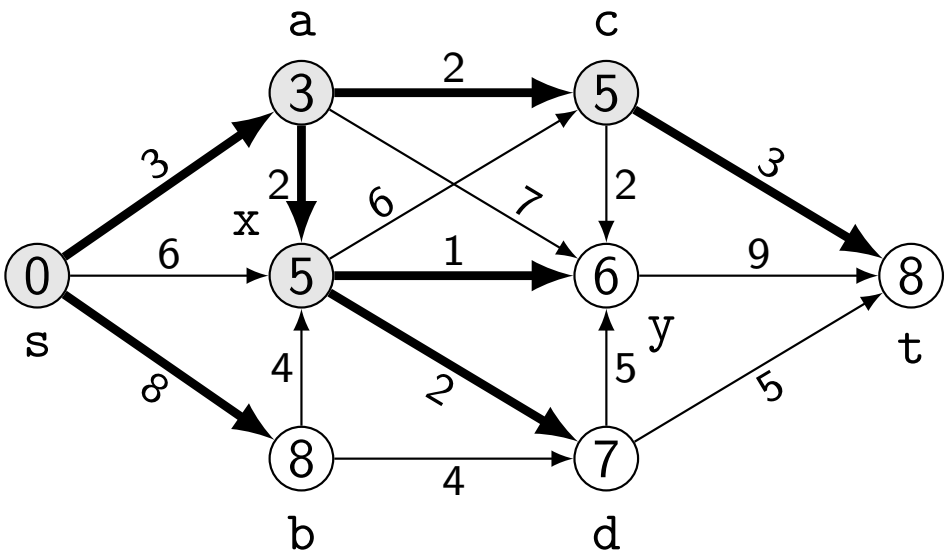


După relaxarea muchiilor din x avem



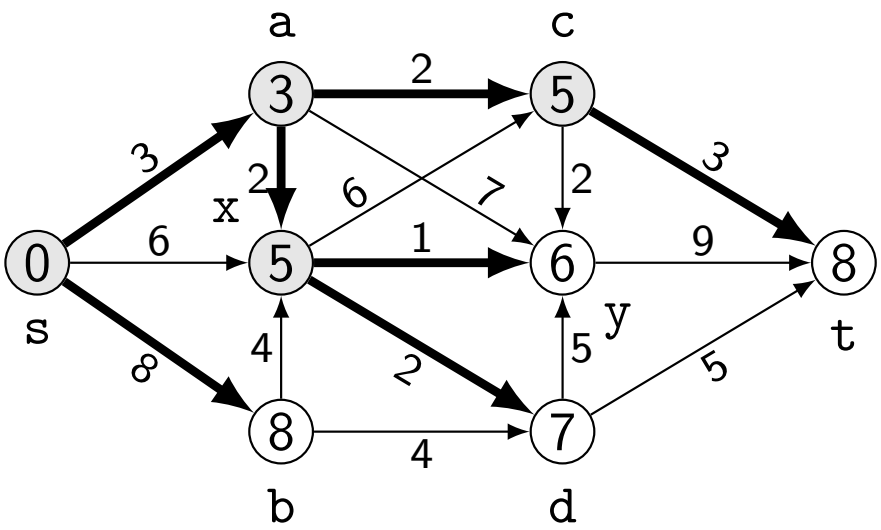


După relaxarea muchiilor din c avem



Algoritmul lui Dijkstra

Exemplu ilustrat

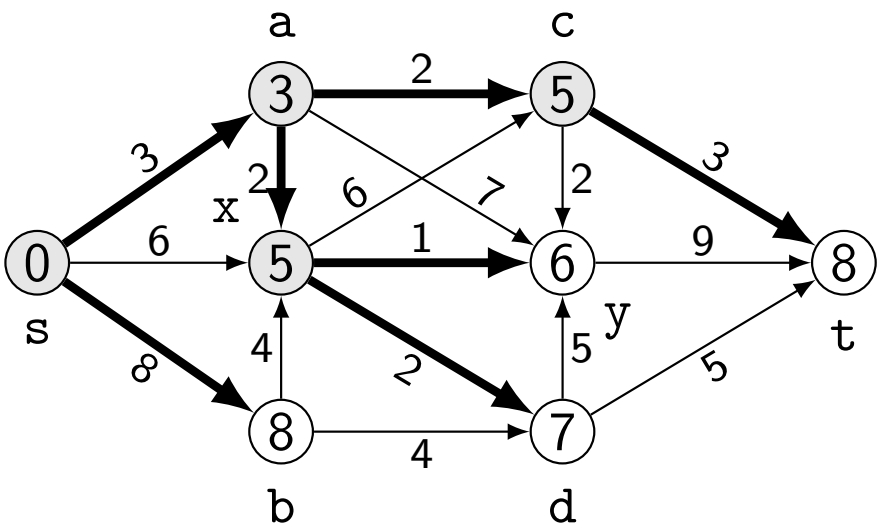


Relaxările ulterioare nu mai modifică valorile lui p[] și d[]:

x	s	a	x	b	c	y	d	t
p[x]	null	s	a	s	a	x	x	c
d[x]	0	3	5	8	5	6	7	8

Algoritmul lui Dijkstra

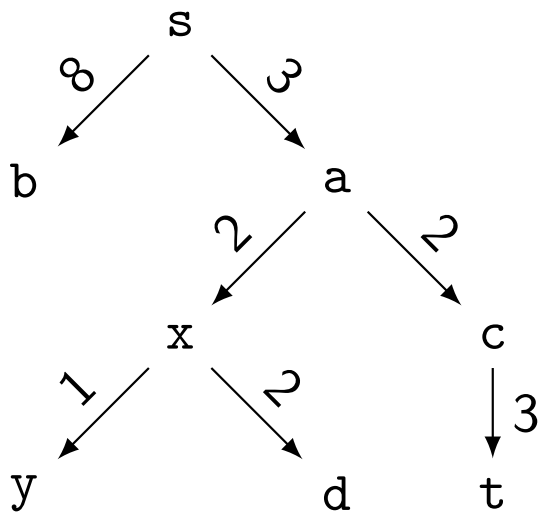
Exemplu ilustrat



Relaxările ulterioare nu mai modifică valorile lui p[] și d[]:

x	s	a	x	b	c	y	d	t
p[x]	null	s	a	s	a	x	x	c
d[x]	0	3	5	8	5	6	7	8

$\Rightarrow$ arborele de căi cu lungimi ponderate minime calculat de algoritm este



Căi cu lungime ponderată minimă între toate nodurile

Se dă un graf ponderat G cu n noduri și m muchii

Se cere: pentru toți $x, y \in V$ să se găsească $x \overset{\pi_{x,y}}{\rightsquigarrow} y$ cu $\text{length}_w(\pi_{x,y}) = \delta_w(x, y)$.

Căi cu lungime ponderată minimă între toate nodurile

Se dă un graf ponderat G cu n noduri și m muchii

Se cere: pentru toți $x, y \in V$ să se găsească $x \overset{\pi_{x,y}}{\rightsquigarrow} y$ cu $\text{length}_w(\pi_{x,y}) = \delta_w(x, y)$.

OBSERVAŢII:

1. Această problemă se poate rezolva rulând de n ori unul din algoritmii deja prezentați, câte o dată pentru fiecare nod $x \in V(G)$ ca sursă.
2. Complexitate temporală:
   - $O(n^4)$ dacă se folosește alg. lui Bellman-Ford pentru cazul general, când putem avea muchii cu ponderi negative.
   - $O(n^3)$ dacă se folosește alg. lui Dijkstra pentru cazul general particular, când $w(e) > 0$ pentru toate muchiile $e \in E$.
3. Vom prezenta o metodă nouă: Algoritmul lui Floyd-Warshall:
   - Complexitate temporală: $O(n^3)$ pentru cazul când putem avea muchii cu ponderi negative, dar fără cicluri cu lungime ponderată negativă.

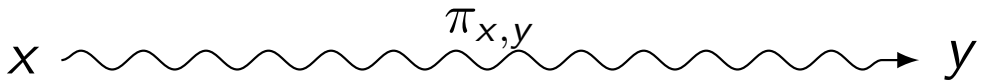
Două tablouri $n \times n$, astfel încât, pentru orice $x, y \in V$:

1. $\mathtt{d}[x][y]$: o margine superioară a lui $\delta_w(x, y)$.
2. $\mathtt{P}[x][y] \in \{\mathtt{null}\} \cup V$.

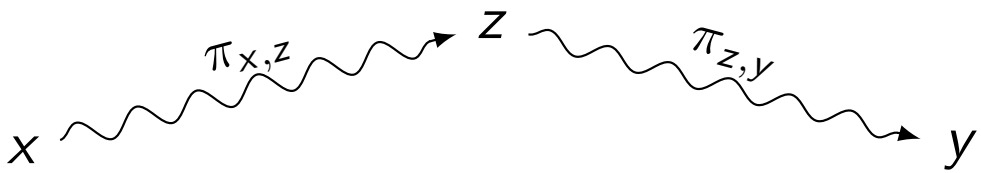
La terminarea algoritmului, valorile lui `P[][]` și `d[][]` au proprietățile următoare:

- $\mathtt{d}[x][y] = \delta_w(x, y)$.
- Dacă $x \neq y$ și există un drum cu lungime ponderată minimă de la x la y atunci $\mathtt{P}[x][y]$ este predecesorul nodului x pe o cale $x \overset{\pi}{\rightsquigarrow} y$ cu lungime ponderată minimă.

Dacă $x, y, z \in V$ atunci orice drum $\pi_{x,y}$ cu lungime ponderată minimă de la x la y are una din următoarele 2 forme:

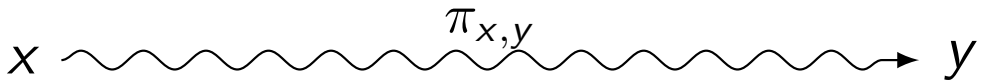
1. 

   unde z nu este nod intermediar al drumului $\pi_{x,y}$, sau

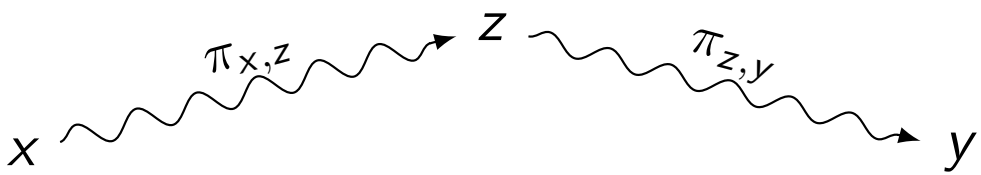
2. 

   unde z nu este nod intermediar al drumurilor $\pi_{x,z}$ și $\pi_{z,y}$.

Dacă $x, y, z \in V$ atunci orice drum $\pi_{x,y}$ cu lungime ponderată minimă de la x la y are una din următoarele 2 forme:

1. 

   unde z nu este nod intermediar al drumului $\pi_{x,y}$, sau

2. 

   unde z nu este nod intermediar al drumurilor $\pi_{x,z}$ și $\pi_{z,y}$.

$\Rightarrow$ putem defini o metodă recursivă de calcul al elementelor tablourilor `P[][]` și `d[][]`.

Fie $[x_1, x_2, \ldots, x_n]$ o enumerare fixată a nodurilor din $V(G)$.
Definim pentru $0 \leq k \leq n$ definim

- $\mathtt{d}[k][i][j]$ este cea mai mică lungime ponderată a unei căi de la x_i la x_j care trece doar prin noduri intermediare din mulțimea $\{x_1, \ldots, x_k\}$. Dacă o astfel de cale nu există, atunci $\mathtt{d}[k][i][j] = +\infty$.
- $\mathtt{P}[k][i][j]$ este `null` dacă $i = j$ sau $\mathtt{d}[k][i][j] = +\infty$. În caz contrar, $\mathtt{P}[k][i][j]$ este predecesorul nodului x_j pe un drum cu lungime ponderată minimă de la x_i la x_j care trece doar prin noduri intermediare din mulțimea $\{x_1, \ldots, x_k\}$.

Rezultă că, pentru toți $i, j \in \{1, 2, \ldots, n\}$ avem

$$\begin{aligned} \texttt{d}[0][i][j] &= w(x_i, x_j), \\ \texttt{P}[0][i][j] &= \begin{cases} \texttt{null} & \text{dacă } i = j \text{ sau } w(x_i, x_j) = +\infty, \\ x_i & \text{în caz contrar} \end{cases} \end{aligned}$$

iar dacă $1 \leq k \leq n$ atunci

$$\begin{aligned} \texttt{d}[0][i][j] &= \min(\texttt{d}[k-1][i][j], \texttt{d}[k-1][i][k] + \texttt{d}[k-1][k][j]), \\ \texttt{P}[k][i][j] &= \begin{cases} \texttt{P}[k-1][i][j] & \text{dacă } \texttt{d}[k-1][i][j] = \texttt{d}[k][i][j], \\ \texttt{P}[k-1][k][j] & \text{în caz contrar.} \end{cases} \end{aligned}$$

Rezultă că, pentru toți $i, j \in \{1, 2, \ldots, n\}$ avem

$$\begin{aligned} \texttt{d}[0][i][j] &= w(x_i, x_j), \\ \texttt{P}[0][i][j] &= \begin{cases} \texttt{null} & \text{dacă } i = j \text{ sau } w(x_i, x_j) = +\infty, \\ x_i & \text{în caz contrar} \end{cases} \end{aligned}$$

iar dacă $1 \leq k \leq n$ atunci

$$\begin{aligned} \texttt{d}[0][i][j] &= \min(\texttt{d}[k-1][i][j], \texttt{d}[k-1][i][k] + \texttt{d}[k-1][k][j]), \\ \texttt{P}[k][i][j] &= \begin{cases} \texttt{P}[k-1][i][j] & \text{dacă } \texttt{d}[k-1][i][j] = \texttt{d}[k][i][j], \\ \texttt{P}[k-1][k][j] & \text{în caz contrar.} \end{cases} \end{aligned}$$

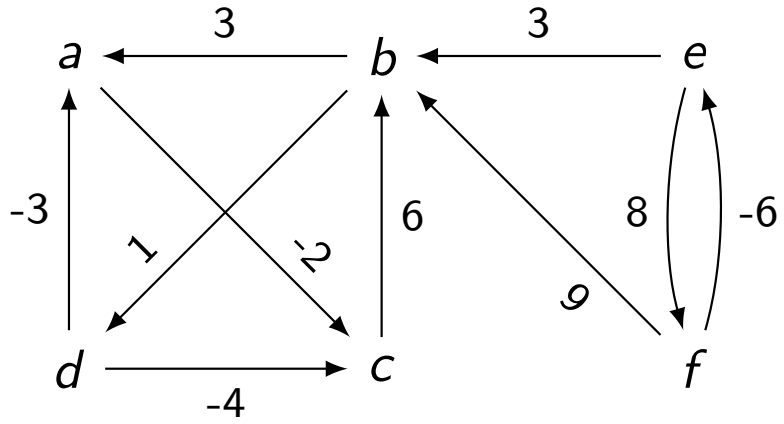
REMARCĂ FINALĂ: Deoarece nodurile intermediare ale oricărei căi sunt în mulțimea $\{x_1, x_2, \ldots, x_n\}$, putem defini

$$\texttt{d}[x_i][x_j] = \texttt{d}[n][i][j] \text{ și } \texttt{P}[x_i][x_j] = \texttt{P}[n][i][j].$$

1. Inițializarea valorilor lui $\mathtt{d}[0][i][j]$ și $\mathtt{P}[0][i][j]$ pentru $1 \leq i, j \leq n$ durează $O(n^2)$.
2. Calculul valorilor lui $\mathtt{d}[k][i][j]$ și $\mathtt{P}[k][i][j]$ din valorile pentru $k-1$ durează $O(n^2)$.
3. Acest calcul trebuie repetat pentru k de la 1 la $n \Rightarrow$ complexitatea temporală $n \cdot O(n^2) = O(n^3)$.

Algoritmul lui Floyd-Warshall

Exemplu ilustrat

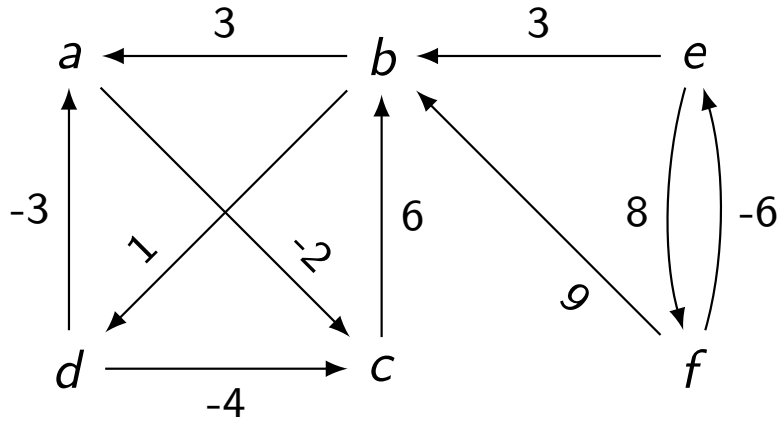


Nodurile sunt enumerate în ordinea $[a, b, c, d, e, f]$

k	$\mathrm{d}[k]$	$\mathrm{P}[k]$
0	$\begin{pmatrix} 0 & +\infty & -2 & +\infty & +\infty & +\infty \\ 3 & 0 & +\infty & 1 & +\infty & +\infty \\ +\infty & 6 & 0 & +\infty & +\infty & +\infty \\ -3 & +\infty & -4 & 0 & +\infty & +\infty \\ +\infty & 3 & +\infty & +\infty & 0 & 8 \\ +\infty & 9 & +\infty & +\infty & -6 & 0 \end{pmatrix}$	$\begin{pmatrix} \bullet & \bullet & a & \bullet & \bullet & \bullet \\ b & \bullet & \bullet & b & \bullet & \bullet \\ \bullet & c & \bullet & \bullet & \bullet & \bullet \\ d & \bullet & d & \bullet & \bullet & \bullet \\ \bullet & e & \bullet & \bullet & \bullet & e \\ \bullet & f & \bullet & \bullet & f & \bullet \end{pmatrix}$

Algoritmul lui Floyd-Warshall

Exemplu ilustrat

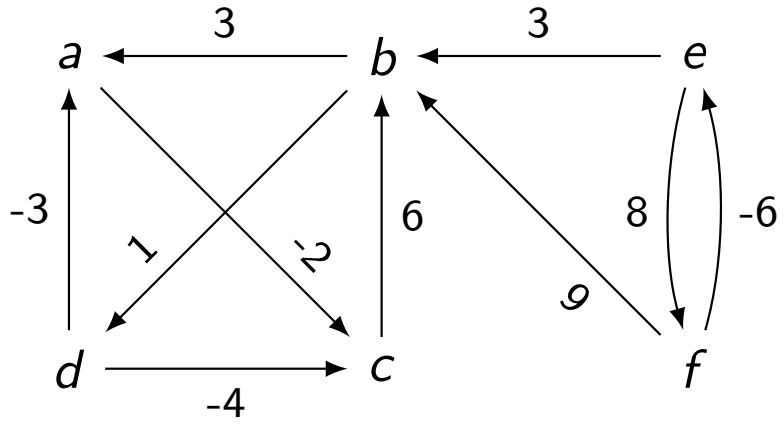


Nodurile sunt enumerate în ordinea $[a, b, c, d, e, f]$

k	$\mathrm{d}[k]$	$\mathrm{P}[k]$
1	$\begin{pmatrix} 0 & +\infty & -2 & +\infty & +\infty & +\infty \\ 3 & 0 & 1 & 1 & +\infty & +\infty \\ +\infty & 6 & 0 & +\infty & +\infty & +\infty \\ -3 & +\infty & -5 & 0 & +\infty & +\infty \\ +\infty & 3 & +\infty & +\infty & 0 & 8 \\ +\infty & 9 & +\infty & +\infty & -6 & 0 \end{pmatrix}$	$\begin{pmatrix} \bullet & \bullet & a & \bullet & \bullet & \bullet \\ b & \bullet & a & b & \bullet & \bullet \\ \bullet & c & \bullet & \bullet & \bullet & \bullet \\ d & \bullet & a & \bullet & \bullet & \bullet \\ \bullet & e & \bullet & \bullet & \bullet & e \\ \bullet & f & \bullet & \bullet & f & \bullet \end{pmatrix}$

Algoritmul lui Floyd-Warshall

Exemplu ilustrat

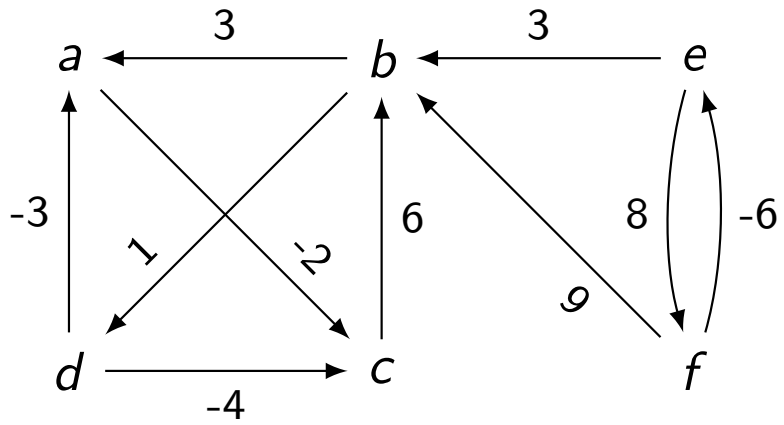


Nodurile sunt enumerate în ordinea $[a, b, c, d, e, f]$

k	d$[k]$	P$[k]$
2	$\begin{pmatrix} 0 & +\infty & -2 & +\infty & +\infty & +\infty \\ 3 & 0 & 1 & 1 & +\infty & +\infty \\ 9 & 6 & 0 & 7 & +\infty & +\infty \\ -3 & +\infty & -5 & 0 & +\infty & +\infty \\ 6 & 3 & 4 & 4 & 0 & 8 \\ 12 & 9 & 10 & 10 & -6 & 0 \end{pmatrix}$	$\begin{pmatrix} \bullet & \bullet & a & \bullet & \bullet & \bullet \\ b & \bullet & a & b & \bullet & \bullet \\ b & c & \bullet & b & \bullet & \bullet \\ d & \bullet & a & \bullet & \bullet & \bullet \\ b & e & a & b & \bullet & e \\ b & f & a & b & f & \bullet \end{pmatrix}$

Algoritmul lui Floyd-Warshall

Exemplu ilustrat

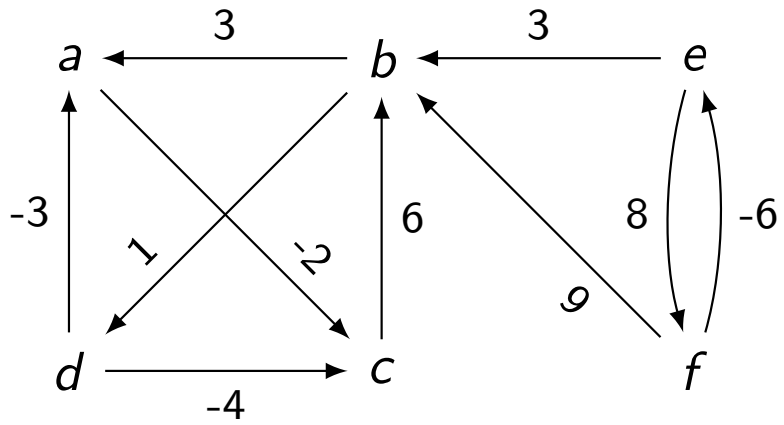


Nodurile sunt enumerate în ordinea $[a, b, c, d, e, f]$

k	$\mathrm{d}[k]$	$\mathrm{P}[k]$
3	$\begin{pmatrix} 0 & 4 & -2 & 5 & +\infty & +\infty \\ 3 & 0 & 1 & 1 & +\infty & +\infty \\ 9 & 6 & 0 & 7 & +\infty & +\infty \\ -3 & 1 & -5 & 0 & +\infty & +\infty \\ 6 & 3 & 4 & 4 & 0 & 8 \\ 12 & 9 & 10 & 10 & -6 & 0 \end{pmatrix}$	$\begin{pmatrix} \bullet & c & a & b & \bullet & \bullet \\ b & \bullet & a & b & \bullet & \bullet \\ b & c & \bullet & b & \bullet & \bullet \\ d & c & a & \bullet & \bullet & \bullet \\ b & e & a & b & \bullet & e \\ b & f & a & b & f & \bullet \end{pmatrix}$

Algoritmul lui Floyd-Warshall

Exemplu ilustrat

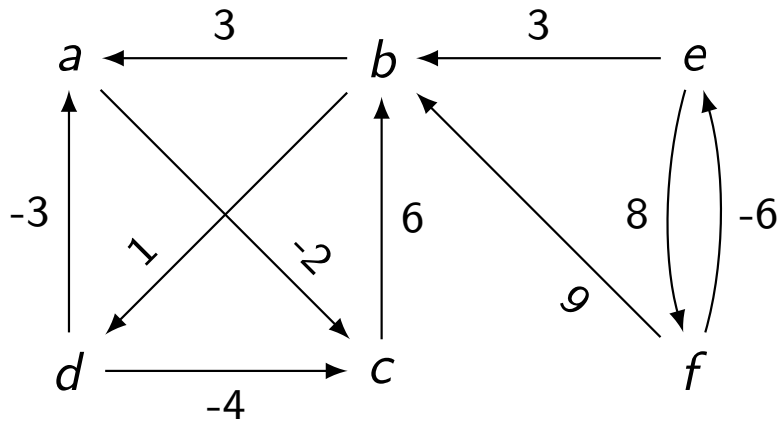


Nodurile sunt enumerate în ordinea $[a, b, c, d, e, f]$

k	$\mathrm{d}[k]$	$\mathrm{P}[k]$
4	$\begin{pmatrix} 0 & 4 & -2 & 5 & +\infty & +\infty \\ -2 & 0 & -4 & 1 & +\infty & +\infty \\ 4 & 6 & 0 & 7 & +\infty & +\infty \\ -3 & 1 & -5 & 0 & +\infty & +\infty \\ 1 & 3 & -1 & 4 & 0 & 8 \\ 7 & 9 & 5 & 10 & -6 & 0 \end{pmatrix}$	$\begin{pmatrix} \bullet & c & a & b & \bullet & \bullet \\ d & \bullet & a & b & \bullet & \bullet \\ d & c & \bullet & b & \bullet & \bullet \\ d & c & a & \bullet & \bullet & \bullet \\ d & e & a & b & \bullet & e \\ d & f & a & b & f & \bullet \end{pmatrix}$

Algoritmul lui Floyd-Warshall

Exemplu ilustrat

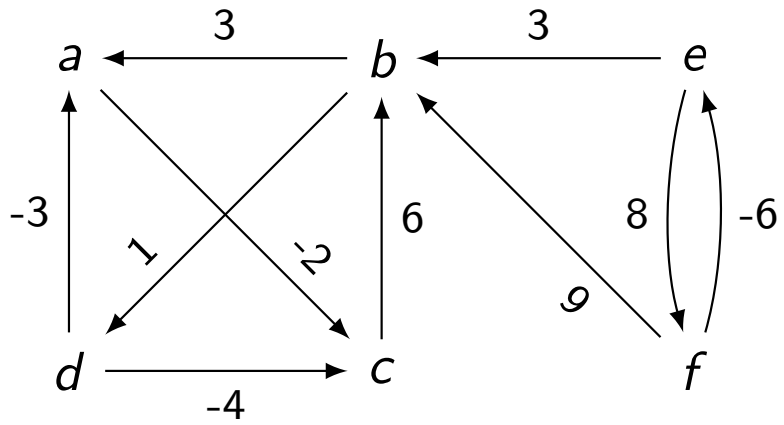


Nodurile sunt enumerate în ordinea $[a, b, c, d, e, f]$

k	$\mathrm{d}[k]$	$\mathrm{P}[k]$
5	$\begin{pmatrix} 0 & 4 & -2 & 5 & +\infty & +\infty \\ -2 & 0 & -4 & 1 & +\infty & +\infty \\ 4 & 6 & 0 & 7 & +\infty & +\infty \\ -3 & 1 & -5 & 0 & +\infty & +\infty \\ 1 & 3 & -1 & 4 & 0 & 8 \\ -5 & -3 & -7 & -2 & -6 & 0 \end{pmatrix}$	$\begin{pmatrix} \bullet & c & a & b & \bullet & \bullet \\ d & \bullet & a & b & \bullet & \bullet \\ d & c & \bullet & b & \bullet & \bullet \\ d & c & a & \bullet & \bullet & \bullet \\ d & e & a & b & \bullet & e \\ d & e & a & b & f & \bullet \end{pmatrix}$

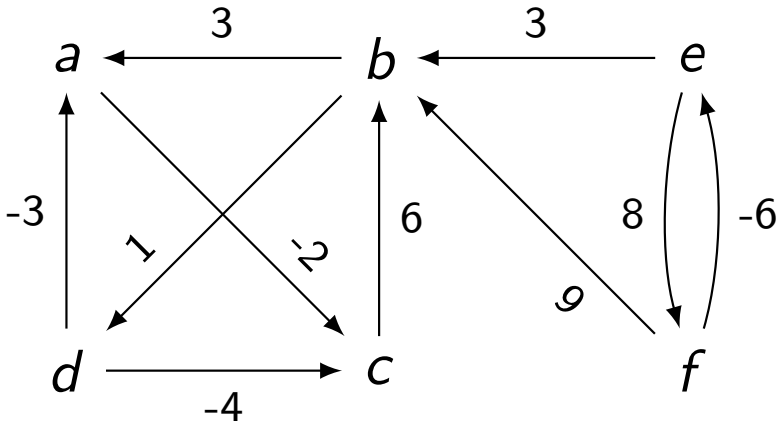
Algoritmul lui Floyd-Warshall

Exemplu ilustrat



Nodurile sunt enumerate în ordinea $[a, b, c, d, e, f]$

k	$\mathrm{d}[k]$	$\mathrm{P}[k]$
6	$\begin{pmatrix} 0 & 4 & -2 & 5 & +\infty & +\infty \\ -2 & 0 & -4 & 1 & +\infty & +\infty \\ 4 & 6 & 0 & 7 & +\infty & +\infty \\ -3 & 1 & -5 & 0 & +\infty & +\infty \\ 1 & 3 & -1 & 4 & 0 & 8 \\ -5 & -3 & -7 & -2 & -6 & 0 \end{pmatrix}$	$\begin{pmatrix} \bullet & c & a & b & \bullet & \bullet \\ d & \bullet & a & b & \bullet & \bullet \\ d & c & \bullet & b & \bullet & \bullet \\ d & c & a & \bullet & \bullet & \bullet \\ d & e & a & b & \bullet & e \\ d & e & a & b & f & \bullet \end{pmatrix}$



Nodurile sunt enumerate în ordinea $[a, b, c, d, e, f]$

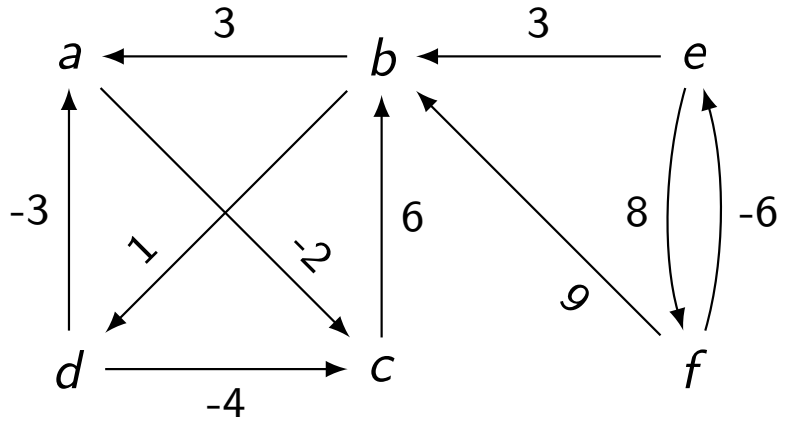
În final, elementele tablourilor `P[][]` și `d[][]` sunt

	a	b	c	d	e	f
a	•	c	a	b	•	•
b	d	•	a	b	•	•
c	d	c	•	b	•	•
d	d	c	a	•	•	•
e	d	e	a	b	•	e
f	d	e	a	b	f	•

	a	b	c	d	e	f
a	0	4	-2	5	$+\infty$	$+\infty$
b	-2	0	-4	1	$+\infty$	$+\infty$
c	4	6	0	7	$+\infty$	$+\infty$
d	-3	1	-5	0	$+\infty$	$+\infty$
e	1	3	-1	4	0	8
f	-5	-3	-7	-2	-6	0

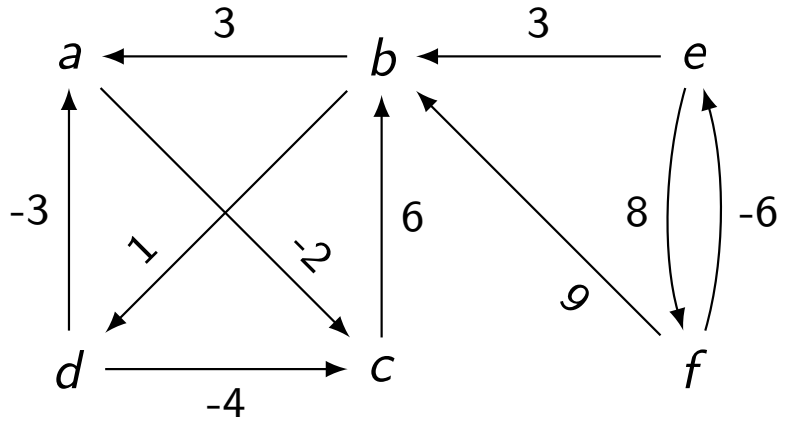
- Tabloul P se numește matrice predecesor.
- Pentru orice nod $x \in G$ putem defini arborele T_x cu rădăcina x și
  - mulțimea de noduri $\{x\} \cup \{y \in V \mid \texttt{P}[x][y] \neq \texttt{null}\}$
  - mulțimea de muchii $\{\texttt{P}[x][y] \rightarrow y \mid y \in V - \{x\}\}$.
- T_x este un arbore de căi cu lungimi ponderate minime de la x în G, și poate fi extras din linia corespunzătoare nodului x în tabloul P[][].

Să se determine un arbore de căi cu lungimi ponderate minime de la f în



$$P[n] = \begin{pmatrix} \bullet & c & a & b & \bullet & \bullet \\ d & \bullet & a & b & \bullet & \bullet \\ d & c & \bullet & b & \bullet & \bullet \\ d & c & a & \bullet & \bullet & \bullet \\ d & e & a & b & \bullet & e \\ d & e & a & b & f & \bullet \end{pmatrix}$$

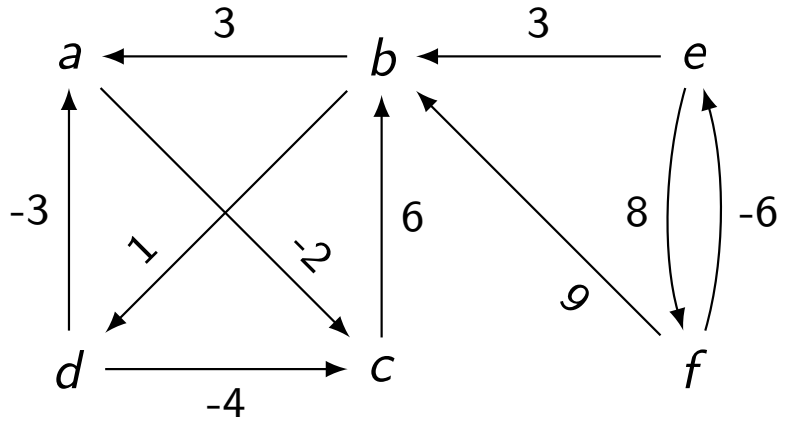
Să se determine un arbore de căi cu lungimi ponderate minime de la f în



$$P[n] = \begin{pmatrix} \bullet & c & a & b & \bullet & \bullet \\ d & \bullet & a & b & \bullet & \bullet \\ d & c & \bullet & b & \bullet & \bullet \\ d & c & a & \bullet & \bullet & \bullet \\ d & e & a & b & \bullet & e \\ d & e & a & b & f & \bullet \end{pmatrix}$$

f este al 6-lea element în enumerarea de noduri $[a, b, c, d, e, f]$, deci T_f se extrage din linia 6 a matricii P[6]:

Să se determine un arbore de căi cu lungimi ponderate minime de la f în



$$P[n] = \begin{pmatrix} \bullet & c & a & b & \bullet & \bullet \\ d & \bullet & a & b & \bullet & \bullet \\ d & c & \bullet & b & \bullet & \bullet \\ d & c & a & \bullet & \bullet & \bullet \\ d & e & a & b & \bullet & e \\ d & e & a & b & f & \bullet \end{pmatrix}$$

f este al 6-lea element în enumerarea de noduri $[a, b, c, d, e, f]$, deci T_f se extrage din linia 6 a matricii P[6]:

$$f \xrightarrow{-6} e \xrightarrow{3} b \xrightarrow{1} d \xrightarrow{-3} a \xrightarrow{-2} c$$