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

Algoritmul lui BELLMAN – FORD

- **Ipoteză:**

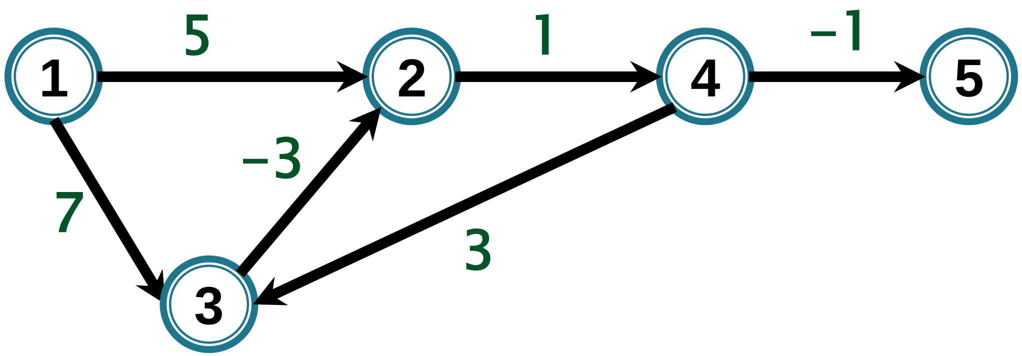
  Arcele pot avea și cost negativ

  Graful **nu** conține circuite de cost negativ

  (dacă există – algoritmul le va detecta => nu soluție)

Algoritmul Bellman Ford

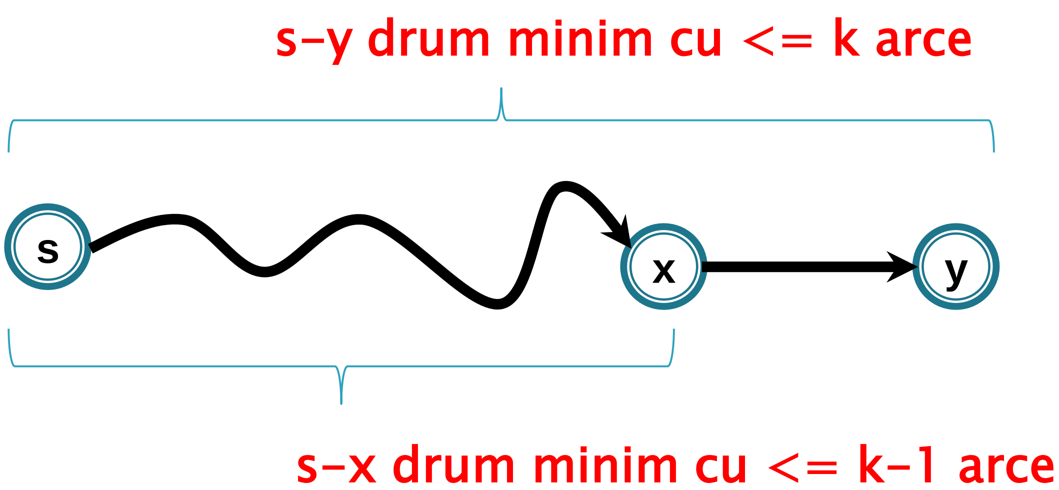
Algoritmul lui BELLMAN – FORD



Algoritmul lui Dijkstra – doar pentru ponderi nenegative

Algoritmul lui BELLMAN - FORD

- **Idee:** **La un pas relaxăm toate arcele**
- De câte ori?

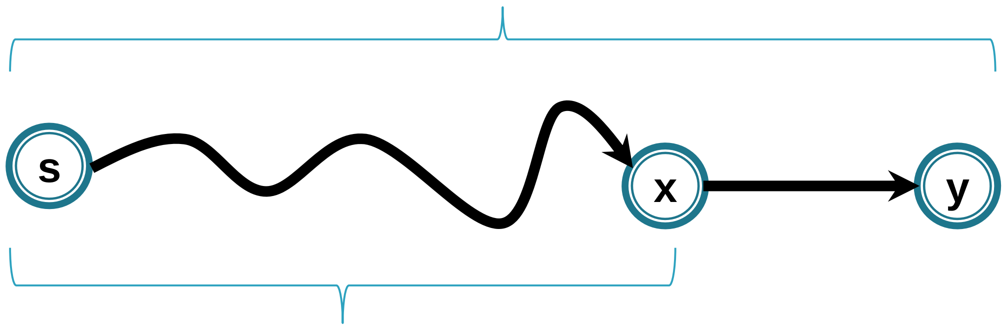


Algoritmul lui BELLMAN – FORD

- **Idee: La un pas relaxăm toate arcele**

- De câte ori?

s–y drum minim cu <= k arce



s–x drum minim cu <= k–1 arce

Dacă P este s–y drum minim cu <= k arce =>

[s $\underline{P}$ x] este s–x drum minim cu <= **k–1 arce**

Un drum minim are cel mult n–1 arce => n–1 etape

Algoritmul lui BELLMAN - FORD

- **Idee: La un pas relaxăm toate arcele**

(nu relaxăm arcele dintr-un vârf selectat u, ci din toate vârfurile)

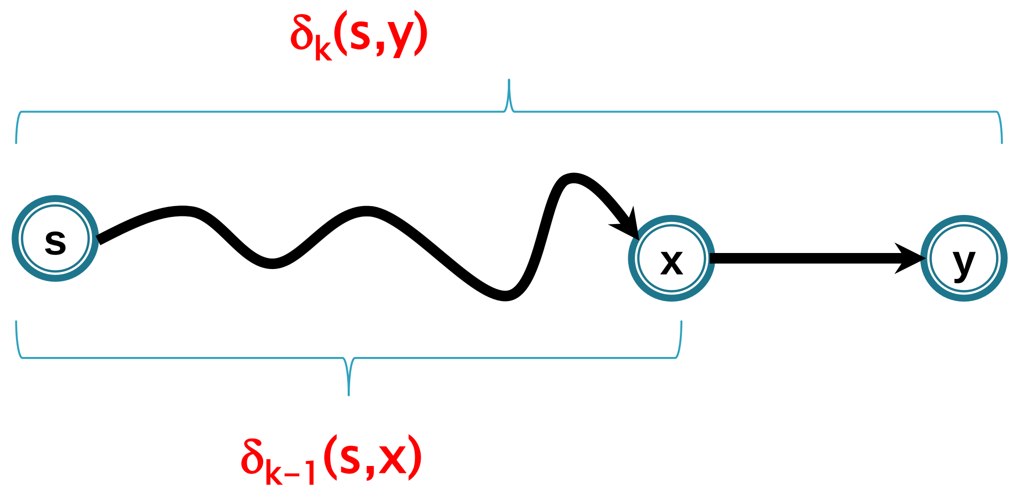
- **n-1 etape** - după k etape d[u] ≤ costul minim al unui drum de la s la u cu cel mult k arce (**programare dinamică**)

=> după k etape sunt corect calculate etichetele d[u] pentru acele vârfuri u pentru care **există un s-u drum minim cu cel mult k arce**

Algoritmul lui BELLMAN – FORD

- Notăm

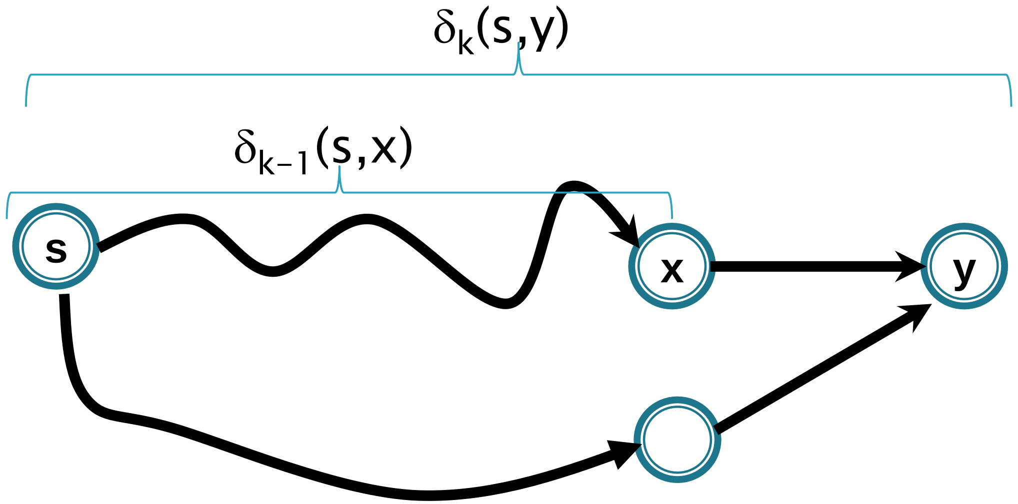
$\delta_k(s,x)$ = costul minim al unui s−x drum cu cel mult k arce



Algoritmul lui BELLMAN – FORD

- Notăm

$\delta_k(s,x)$ = costul minim al unui s-x drum cu cel mult k arce



Avem

$$\delta_k(s,y) = \min\{\delta_{k-1}(s,y),\ \min\{\delta_{k-1}(s,x)+w(xy) \mid xy \in E\}\}$$

Algoritmul lui BELLMAN – FORD

```
pentru fiecare u∈V executa
    d[u] = ∞; tata[u]=0
d[s] = 0

pentru i = 1,n-1 executa
   pentru fiecare uv∈E executa
         daca d[u]+w(u,v)<d[v] atunci
                d[v] = d[u]+w(u,v)
                tata[v] = u
```

Algoritmul lui BELLMAN – FORD

```
pentru fiecare u∈V executa
     d[u] = ∞; tata[u]=0
d[s] = 0

pentru i = 1,n-1 executa
   pentru fiecare uv∈E executa
         daca d[u]+w(u,v)<d[v] atunci
               d[v] = d[u]+w(u,v)
               tata[v] = u
```

Complexitate: O(nm)

Algoritmul lui BELLMAN - FORD

- Optimizări

La un pas este suficient să relaxăm arcele din vârfuri ale căror etichetă s-a modificat anterior=>

Algoritmul lui BELLMAN - FORD

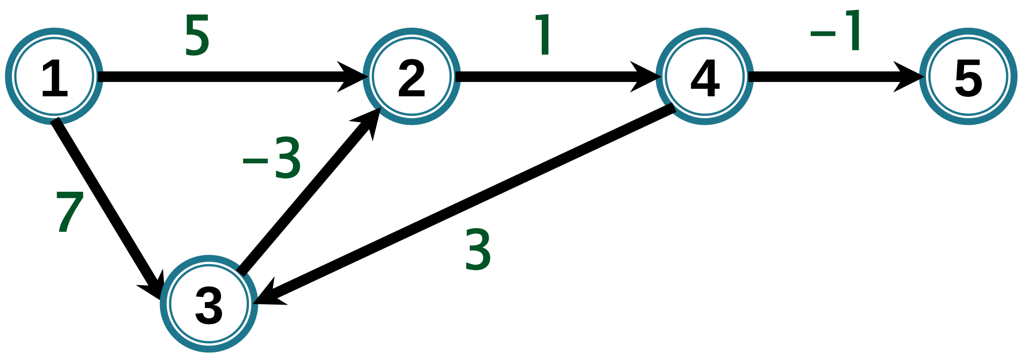
- **Optimizări**

La un pas este suficient să relaxăm arcele din vârfuri ale căror etichetă s-a modificat anterior=>

coadă cu vârfurile ale căror etichetă s-a actualizat

Detalii - laborator

Etapa 1



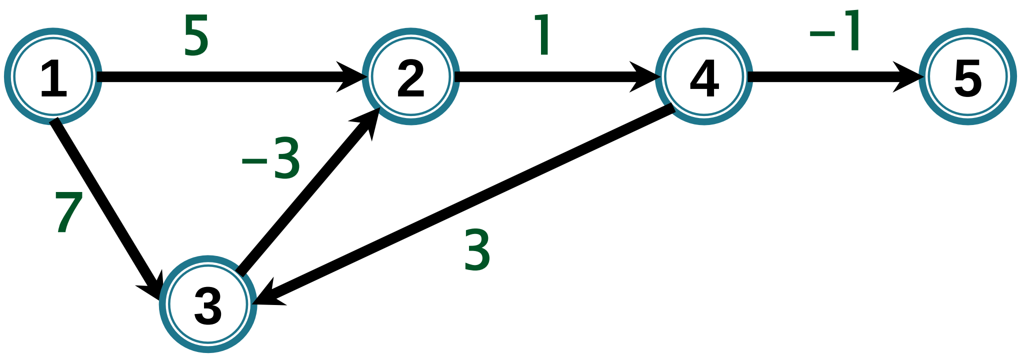
Relaxăm

1 2

1 3

	1	2	3	4	5
d	0	5	7	∞	∞

Etapa 1



Relaxăm

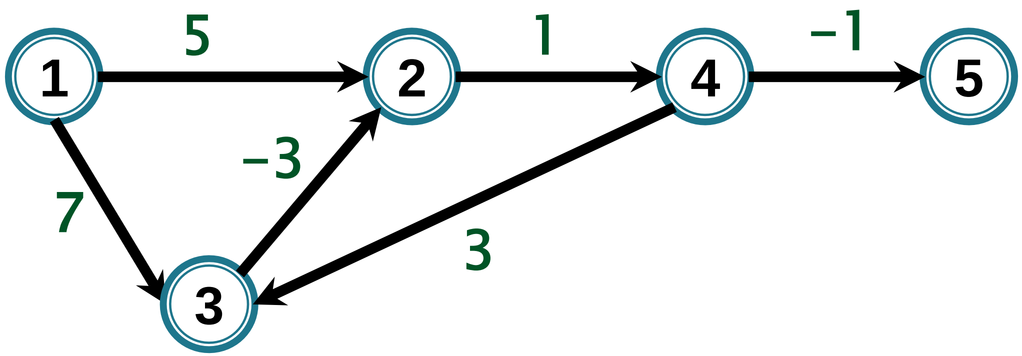
1 2

1 3

2 4

	1	2	3	4	5
d	0	5	7	6	∞

Etapa 1



Relaxăm

1 2

1 3

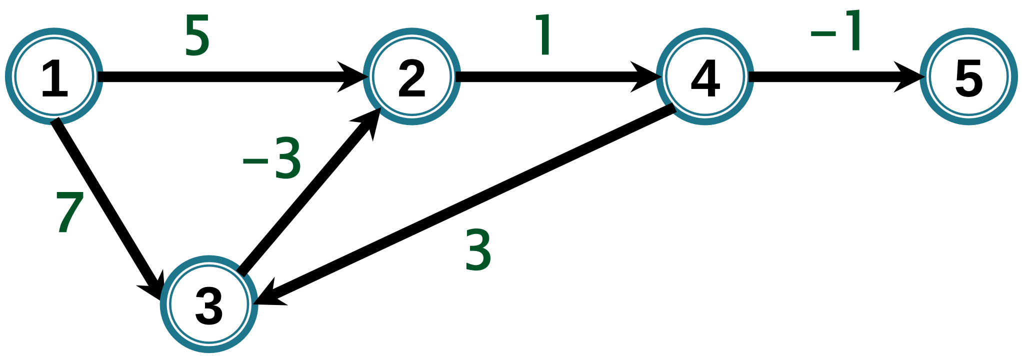
2 4

4 3

4 5

	1	2	3	4	5
d	0	5	7	6	5

Etapa 1



Relaxăm

1 2

1 3

2 4

4 3

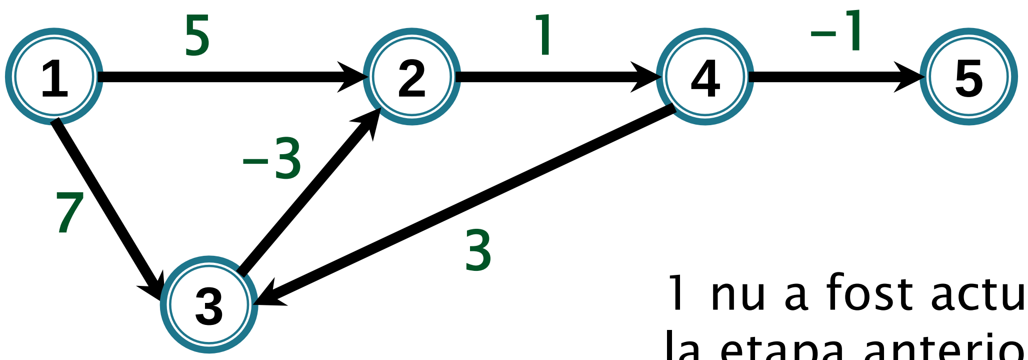
4 5

3 2

	1	2	3	4	5
d	0	4	7	6	5

Etapa 2

Relaxăm



1 2

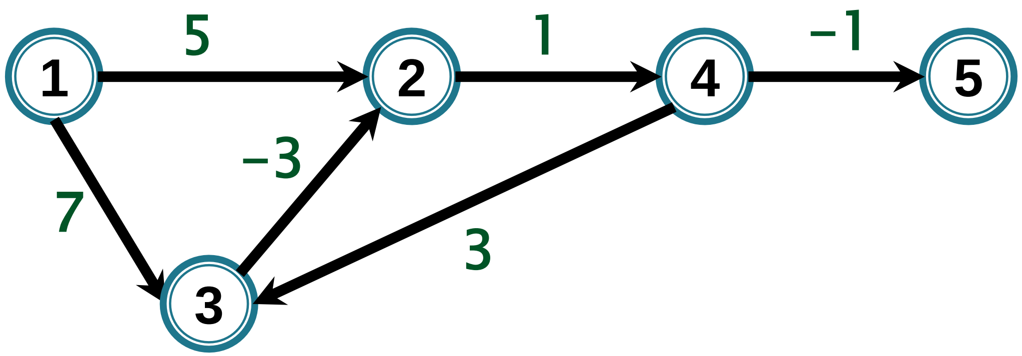
1 3

1 nu a fost actualizat
la etapa anterioara,
le putem ignora

	1	2	3	4	5
d	0	4	7	6	5

Etapa 2

Relaxăm



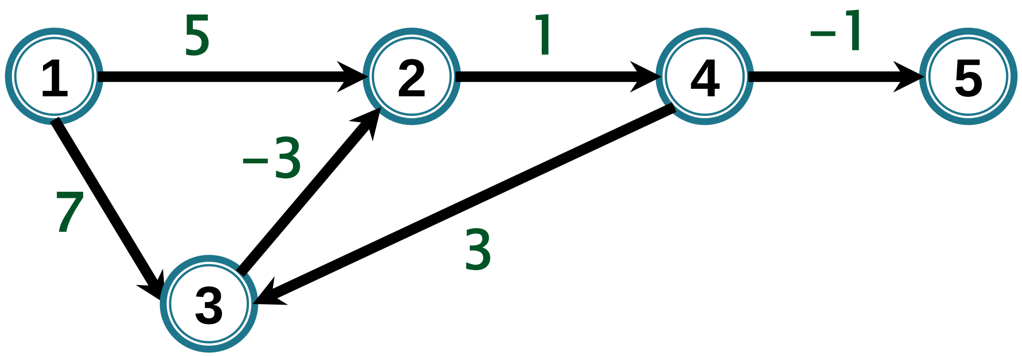
1 2

1 3

2 4

	1	2	3	4	5
d	0	4	7	5	5

Etapa 2



Relaxăm

1 2

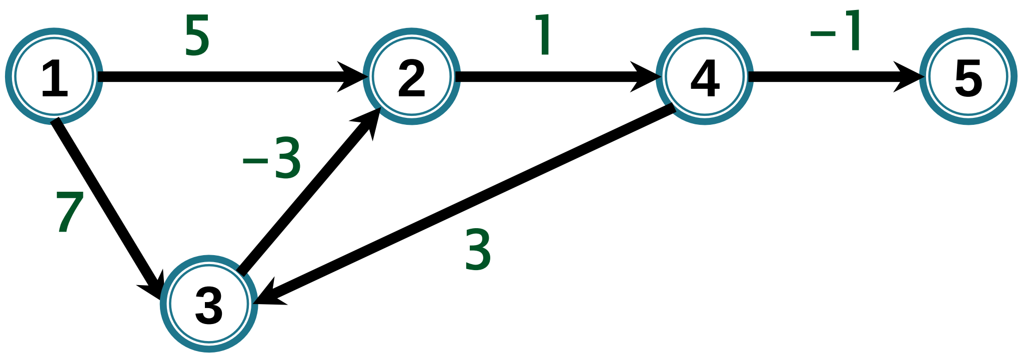
1 3

2 4

4 3

	1	2	3	4	5
d	0	4	7	5	5

Etapa 2



Relaxăm

1 2

1 3

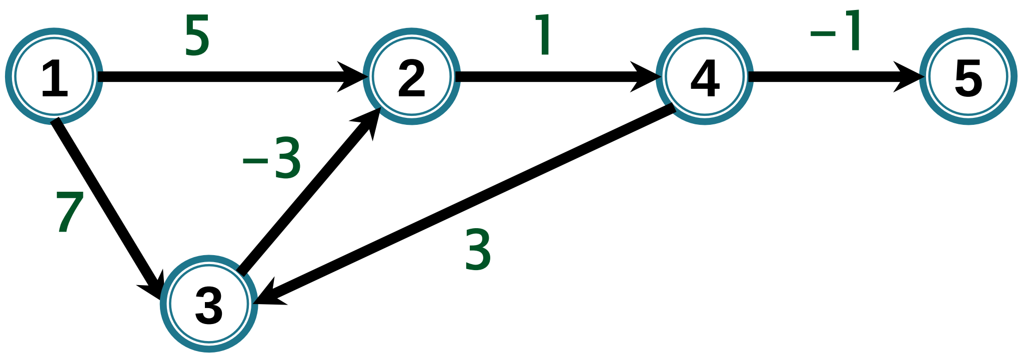
2 4

4 3

4 5

	1	2	3	4	5
d	0	4	7	5	4

Etapa 2



Relaxăm

1 2

1 3

2 4

4 3

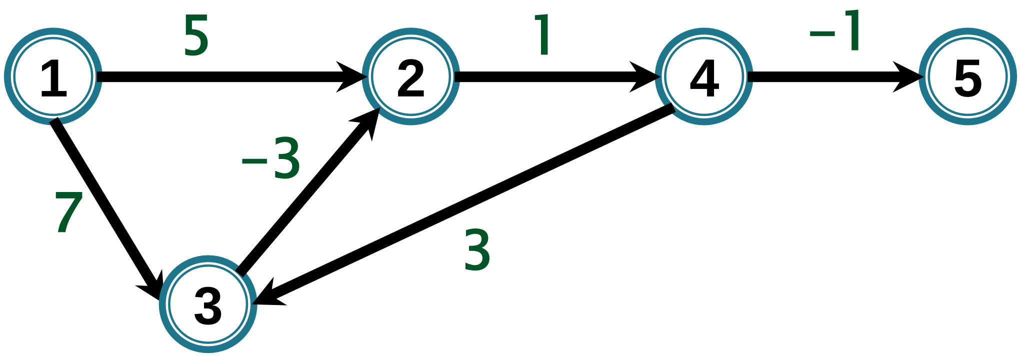
4 5

3 2

	1	2	3	4	5
d	0	4	7	5	4

Etapa 3

Relaxăm



1 2

1 3

2 4

4 3

4 5

3 2

Nu se mai actualizează nimic - stop

	1	2	3	4	5
d	0	4	7	5	4

Corectitudinea Algoritmului lui Bellman-Ford

Corectitudine

- **Lema 1(comună).** Pentru orice $u \in V$, la orice pas al algoritmului avem:

  a) dacă $d[u] < \infty$, există un drum de la s la u în G de cost d[u] și acesta se poate determina din vectorul tata:

  tata[u]= predecesorul lui u pe un drum de la s la u de cost d[u]

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

Corectitudine

- **Demonstrație:** Inducție după numărul de etape (o etapa = relaxarea tuturor muchiilor):

După k iterații

$d[x] \leq \delta_k(s,x)$

$=$ costul minim al unui s-x drum cu cel mult k muchii

Corectitudine

- $d[x] \le \delta_k(s,x)$

  $= $ costul minim al unui s−x drum cu cel mult k muchii

- k = 0: d[s] = 0 = w([s])

- k−1 => k. Presupunem că înainte de iterația k

  $d[x] \le \delta_{k-1}(s,x)$ pentru orice x

Eticheta unui vârf y la iterația k se actualizează astfel:

Corectitudine

ipoteza de inducție

$$d[y] \le \min\{d[y], \min\{d[x]+w(x,y) \mid xy \in E\} \le$$

Corectitudine

ipoteza de inducție

$$d[y] \le \min\{d[y], \min\{d[x]+w(x,y) \mid xy \in E\} \le$$

$$\le \min\{\delta_{k-1}(s,y)\ , \min\{\delta_{k-1}(s,x) + w(x,y) \mid xy \in E\} =$$

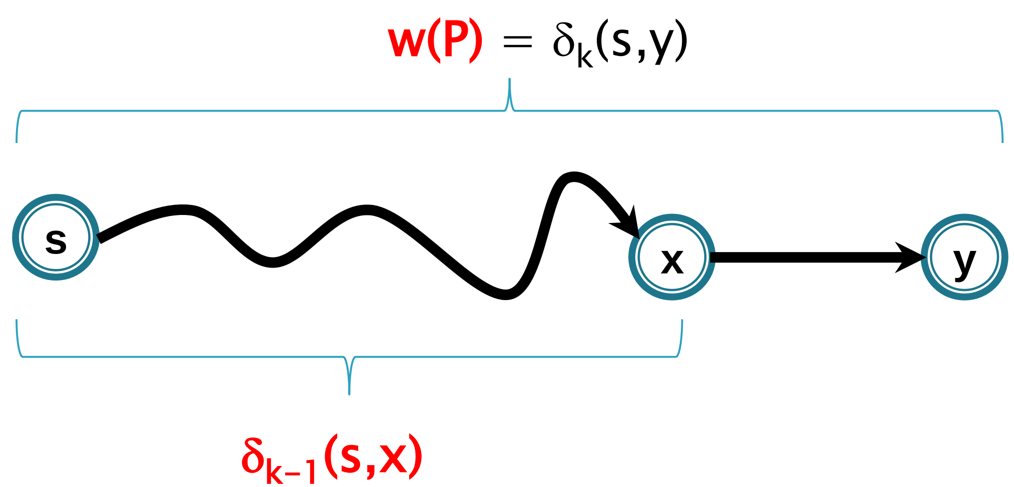
$$= \delta_k(s,y)$$

Corectitudine

- k-1 => k. Presupunem că înainte de iterația k

$$d[x] \leq \delta_{k-1}(s,x) \text{ pentru orice } x$$

Varianta 2. Fie P un s-y drum cu cost minim printre cele cu cel mult k arce ($w(P) = \delta_k(s,y)$)



Corectitudine

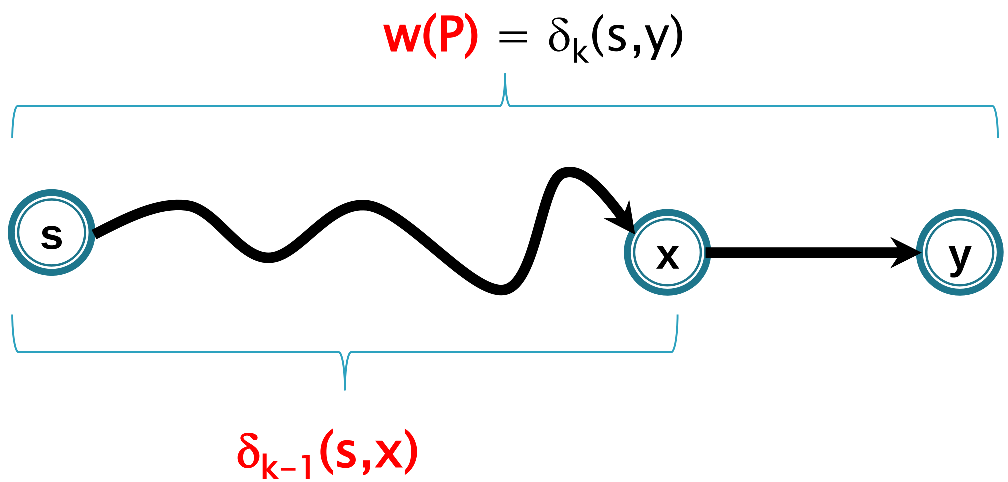
- k-1 => k. Presupunem că înainte de iterația k

$$d[x] \le \delta_{k-1}(s,x) \text{ pentru orice } x$$

Varianta 2. Fie P un s-y drum cu cost minim printre cele cu cel mult k arce ($w(P) = \delta_k(s,y)$)

=> [s $\underline{P}$ x] este s-x drum cu cost minim printre cele cu cel mult k-1 arce, deci are cost $\delta_{k-1}(s,x)$

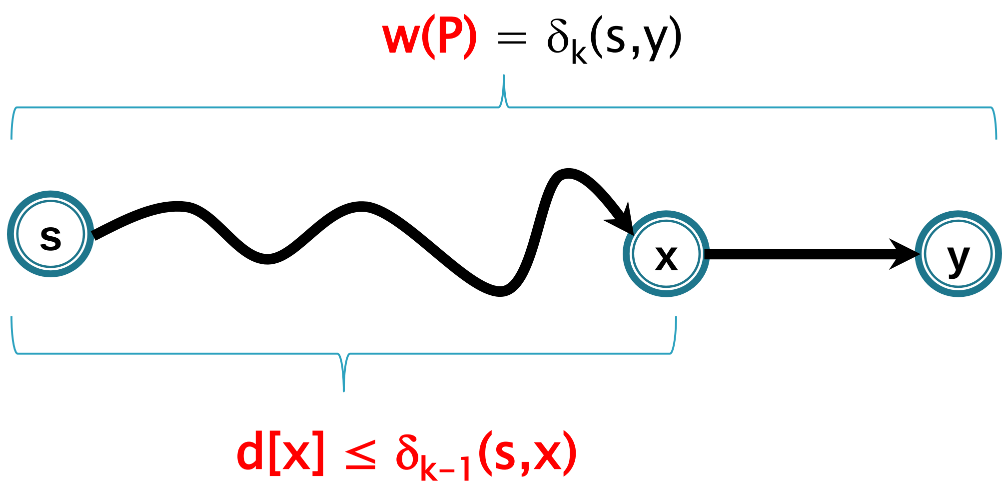
=> $d[x] \le \delta_{k-1}(s,x)$



Corectitudine

După relaxarea arcului xy:

$$
\begin{aligned}
d[y] &\le d[x] + w(xy) \le \\
&\le \delta_{k-1}(s,x) + w(xy) = \\
&= w([s\,\underline{P}\,x]) + w(xy) = w(P) = \delta_k(s,y)
\end{aligned}
$$



Detectarea de circuite negative

Detectarea de circuite negative

- Există un circuit negativ în G $\Leftrightarrow$ dacă algoritmul ar mai face o iterație s-ar mai actualiza etichete de distanță

$\Leftrightarrow$ După n-1 iterații există un arc uv cu

$$d[v] > d[u] + w(uv)$$

Detectarea de circuite negative

Demonstrație: Arătăm că

nu există cicluri negative $\Leftrightarrow$

nu se mai fac actualizări la pasul n

Detectarea de circuite negative

Demonstrație: Arătăm că

nu există cicluri negative $\Leftrightarrow$

nu se mai fac actualizări la pasul n

- Dacă nu există cicluri negative => nu se mai fac actualizări la pasul n (din corectitudine)

Detectarea de circuite negative

Demonstrație: Arătăm că

nu există cicluri negative $\Leftrightarrow$

nu se mai fac actualizări la pasul n

- Dacă nu există cicluri negative => nu se mai fac actualizări la pasul n (din corectitudine)
- Dacă nu se mai fac actualizări la pasul n, pentru orice ciclu $C=[v_0,\dots,v_p, v_0]$ => $d[v_i] <= d[v_i]+w(v_i v_{i+1})$

Însumând pe ciclu:

Detectarea de circuite negative

Demonstrație: Arătăm că

nu există cicluri negative $\Leftrightarrow$

nu se mai fac actualizări la pasul n

- Dacă nu există cicluri negative => nu se mai fac actualizări la pasul n (din corectitudine)
- Dacă nu se mai fac actualizări la pasul n, pentru orice ciclu $C=[v_0,\ldots,v_p, v_0]$ => $d[v_i]<=d[v_i]+w(v_iv_{i+1})$

Însumând pe ciclu:

$d[v_o]+\ldots+d[v_p] \leq d[v_o]+\ldots+d[v_p] +w(v_ov_1)+\ldots+w(v_pv_o)$

$\Rightarrow\ 0 \leq w(v_ov_1)+\ldots+w(v_pv_o)= w(C)$

Detectarea de circuite negative

Afișarea ciclului negative detectat – folosind tata:

Detectarea de circuite negative

Afișarea ciclului negative detectat – folosind tata:

- Fie v un vârf al cărei etichetă s-a actualizat la pasul k

Detectarea de circuite negative

Afișarea ciclului negative detectat – folosind tata:

- Fie v un vârf al cărei etichetă s-a actualizat la pasul k
- Facem n pași înapoi din v folosind vectorul tata (către s) ; fie x vârful în care am ajuns
- Afișăm ciclul care conține pe x folosind tata (din x până ajungem iar în x)