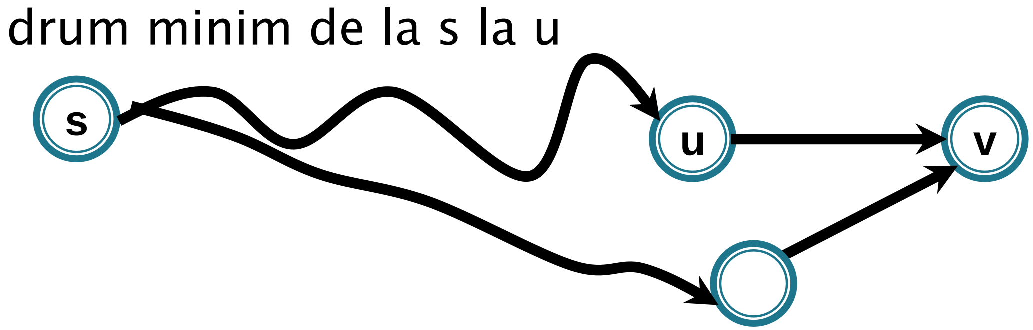
SINTEZĂ

Drumuri minime de sursă unică – algoritmi

Drumuri minime de sursă unică în grafuri aciclice

- **Relație de recurență**

$$\delta(s,v) = \min\{\delta(s,u) + w(u,v) \mid uv \in E\}$$



Deoarece:

- un drum minim de la s la predecesor u al lui v + arcul uv = un drum de la s la v (de cost $\geq \delta(s,v)$)
- Un drum minim P de la s la v = un drum minim de la s la predecesor u al lui v + arcul uv

Drumuri minime de sursă unică în grafuri aciclice

- **Relație de recurență**

$$\delta(s,v) = \min\{\delta(s,u) + w(u,v) \mid uv \in E\}$$

Există ordine de calcul în DAG = sortare topologică

Drumuri minime de sursă unică în grafuri aciclice

- Relație de recurență – pentru costuri **pozitive**

$$\delta(s,v) = \min\{\delta(s,u) + w(u,v) \mid uv \in E,\ \delta(s,u) < \delta(s,v)\ ?\ \}$$

Predecesorul u al lui v pe un s-v drum minim este mai aproape de sursa s decât v, deoarece arcele au cost pozitiv

Drumuri minime de sursă unică în grafuri aciclice

- **Relație de recurență – pentru costuri pozitive**

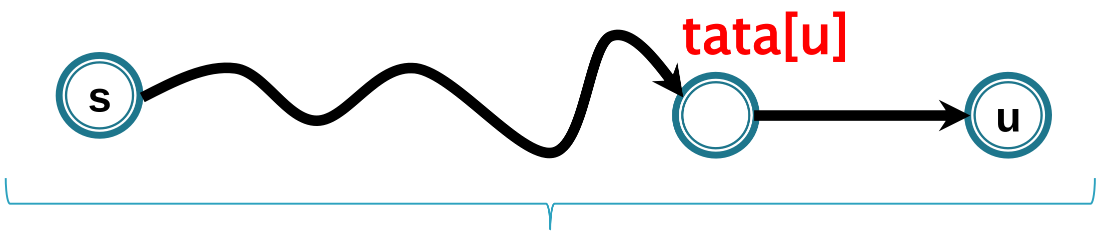
$$\delta(s,v) = \min\{\delta(s,u) + w(u,v) \mid uv \in E, \ \delta(s,u) < \delta(s,v)\ ?\ \}$$

Algoritmul lui DIJKSTRA – Vârfurile considerate în funcție de distanța estimată față de sursă, la fiecare pas este vizitat vârful cel mai apropiat de sursă și extinse drumurile din el

Caz neponderat – Vârfurile considerate în ordinea dată de BFS = ordine crescătoare în raport cu distanța față de sursă

Drumuri minime de sursă unică s

- **Idei comune:** **Pe parcursul algoritmului fiecare vârf are asociate informațiile:**
  - **d[u]** – etichetă de distanță
  - **tata[u]**



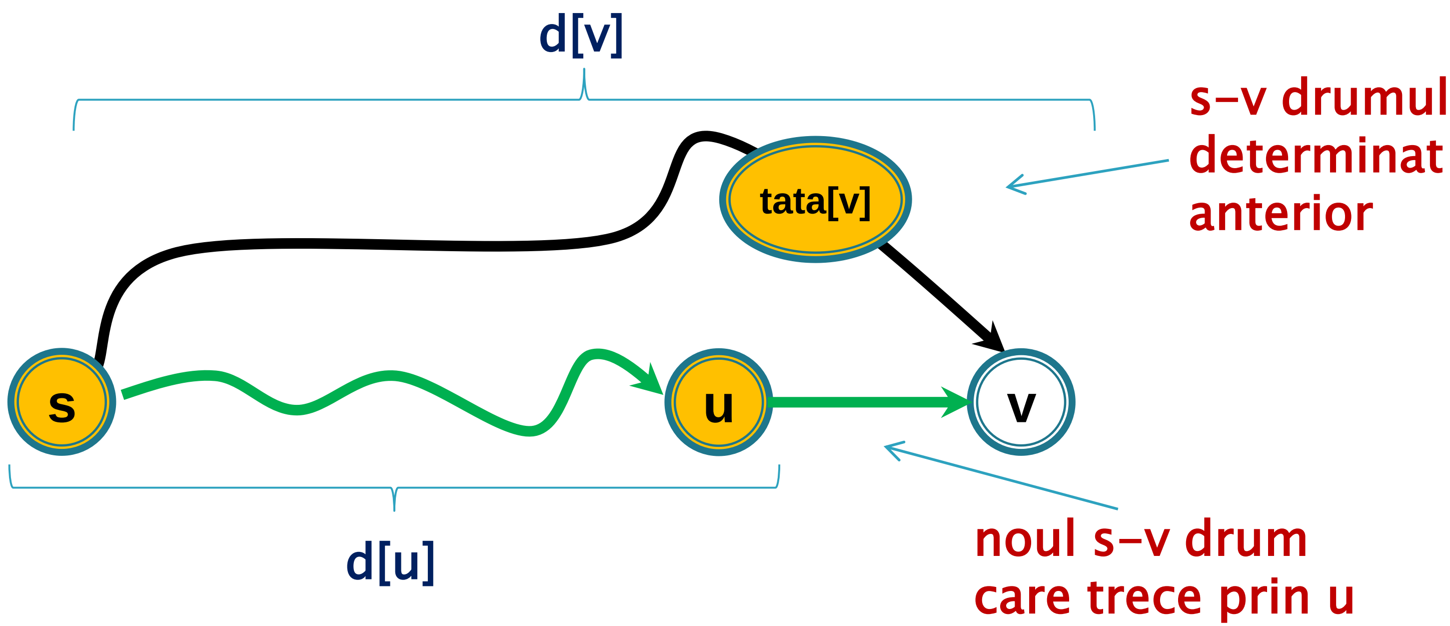
d[u] = costul minim al unui drum de la s la u descoperit până la acel moment

tata[u] = **predecesorul** lui u pe drumul de cost minim de la s la u descoperit până la acel moment

Drumuri minime de sursă unică s

- **Relaxarea unui arc (u, v)** = a verifica dacă d[v] poate fi îmbunătăţit extinzând drumul minim deja găsit de la s la u cu arcul uv

```
dacă d[u] + w(u,v) < d[v] atunci
     d[v] = d[u] + w(u,v);
     tata[v] = u
```



Algoritmi – G=(V, E) graf orientat

G – neponderat
Parcurgere lățime BF

BF(s)

```
coada C ← ∅;
adauga(s, C)
```

G – ponderat, ponderi >0
Algoritmul lui Dijkstra

Dijkstra(s)

```
Q ← V
{se putea incepe doar cu Q ← {s}
+vector viz; v∈Q ⇔ v nevizitat}
```

G – ponderat fără circuite

DAGS(s)

```
SortTop ← sortare_topologica(G)
```

Algoritmi – G=(V, E) graf orientat

G – neponderat

Parcurgere lățime BF

```
BF(s)

coada C ← ∅;
adauga(s, C)
```

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

viz[s]← 1; d[s] ← 0
```

G – ponderat, ponderi >0

Algoritmul lui Dijkstra

```
Dijkstra(s)

Q ← V
{se putea incepe doar cu Q ← {s}
+vector viz; v∈Q ⇔ v nevizitat}
```

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

d[s] = 0
```

G – ponderat fără circuite

```
DAGS(s)

SortTop ← sortare_topologica(G)
```

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

d[s] = 0
```

Algoritmi – G=(V, E) graf orientat

G – neponderat
Parcurgere lățime BF

```
BF(s)

coada C ← ∅;
adauga(s, C)

pentru fiecare u∈V
  d[u]=∞;tata[u]=viz[u]=0

viz[s]← 1; d[s] ← 0

cat timp C ≠ ∅
   u ←  extrage(C);

   pentru fiecare uv∈E
```

G – ponderat, ponderi >0
Algoritmul lui Dijkstra

```
Dijkstra(s)

Q ← V
{se putea incepe doar cu Q ← {s}
+vector viz; v∈Q ⇔ v nevizitat}

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

d[s] = 0

cat timp Q ≠ ∅
   u = extrage(Q) vârf cu eticheta
        d minimă

   pentru fiecare uv∈E
```

G – ponderat fără circuite

```
DAGS(s)

SortTop ← sortare_topologica(G)

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

d[s] = 0

pentru fiecare u ∈ SortTop

   pentru fiecare uv∈E
```

Algoritmi – G=(V, E) graf orientat

G – neponderat
Parcurgere lățime BF

```
BF(s)

coada C ← ∅;
adauga(s, C)
- - - - - - - - - - - - - - - - - -
pentru fiecare u∈V
  d[u]=∞;tata[u]=viz[u]=0

viz[s]← 1; d[s] ← 0
- - - - - - - - - - - - - - - - - -
cat timp C ≠ ∅
   u ←  extrage(C);

   pentru fiecare uv∈E
       daca viz[v]=0
          d[v] ← d[u]+1
          tata[v] ← u
          adauga(v, C)
          viz[v] ← 1
- - - - - - - - - - - - - - - - - -
scrie d, tata
```

G – ponderat, ponderi >0
Algoritmul lui Dijkstra

```
Dijkstra(s)

Q ← V
{se putea incepe doar cu Q ← {s}
+vector viz; v∈Q ⇔ v nevizitat}
- - - - - - - - - - - - - - - - - -
pentru fiecare u∈V
    d[u] = ∞; tata[u]=0

d[s] = 0
- - - - - - - - - - - - - - - - - -
cat timp Q ≠ ∅
   u = extrage(Q) vârf cu eticheta
        d minimă

   pentru fiecare uv∈E
      daca v∈Q si d[u]+w(u,v)<d[v]
             d[v] = d[u]+w(u,v)
             tata[v] = u
             repara(v,Q)
- - - - - - - - - - - - - - - - - -
scrie d, tata
```

G – ponderat fără circuite

```
DAGS(s)

SortTop ← sortare_topologica(G)
- - - - - - - - - - - - - - - - - -
pentru fiecare u∈V
    d[u] = ∞; tata[u]=0

d[s] = 0
- - - - - - - - - - - - - - - - - -
pentru fiecare u ∈ SortTop

   pentru fiecare uv∈E
        daca d[u]+w(u,v)<d[v]
             d[v] = d[u]+w(u,v)
             tata[v] = u
- - - - - - - - - - - - - - - - - -
scrie d, tata
```

Algoritmi – G=(V, E) graf orientat

G – neponderat
Parcurgere lățime BF

```
BF(s)

coada C ← ∅;
adauga(s, C)

pentru fiecare u∈V
  d[u]=∞;tata[u]=viz[u]=0

viz[s]← 1; d[s] ← 0

cat timp C ≠ ∅
   u ←  extrage(C);

   pentru fiecare uv∈E
       daca viz[v]=0
          d[v] ← d[u]+1
          tata[v] ← u
          adauga(v, C)
          viz[v] ← 1

scrie d, tata
```

O(n+m)

G – ponderat, ponderi >0
Algoritmul lui Dijkstra

```
Dijkstra(s)

Q ← V
{se putea incepe doar cu Q ← {s}
+vector viz; v∈Q ⇔ v nevizitat}

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

d[s] = 0

cat timp Q ≠ ∅
   u = extrage(Q) vârf cu eticheta
        d minimă

   pentru fiecare uv∈E
      daca v∈Q si d[u]+w(u,v)<d[v]
             d[v] = d[u]+w(u,v)
             tata[v] = u
             repara(v,Q)

scrie d, tata
```

O(m log(n))/ O(n²)
– ca Prim

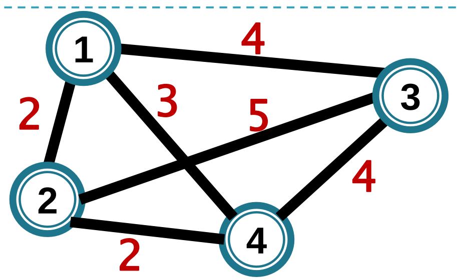
G – ponderat fără circuite

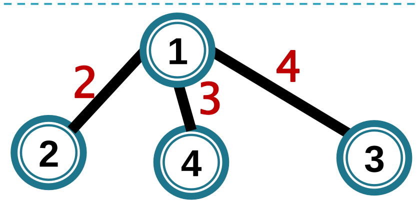
```
DAGS(s)

SortTop ← sortare_topologica(G)

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

d[s] = 0

pentru fiecare u ∈ SortTop

   pentru fiecare uv∈E
      daca d[u]+w(u,v)<d[v]
          d[v] = d[u]+w(u,v)
          tata[v] = u

scrie d, tata
```

O(n+m)

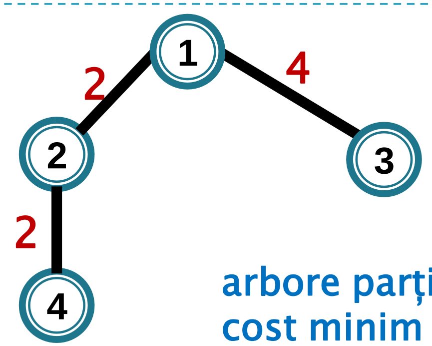
Drumuri minime din s ⇒ arbore de drumuri minime (distanțe) din s

≠ arbore parțial de cost minim – minimizează costul total





arbore al drumurilor minime față de 1



arbore parțial de cost minim

Drumuri minime din s ⇒ arbore de drumuri minime (distanțe) din s

≠ arbore parțial de cost minim – minimizează costul total

G-(ne)orientat ponderat,
ponderi >0

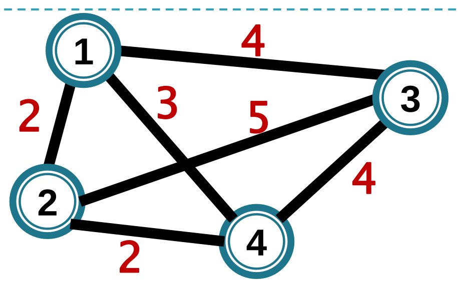
Drumuri minime din s

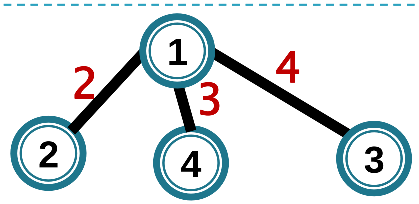
Algoritmul lui Dijkstra

G- neorientat ponderat
ponderi reale

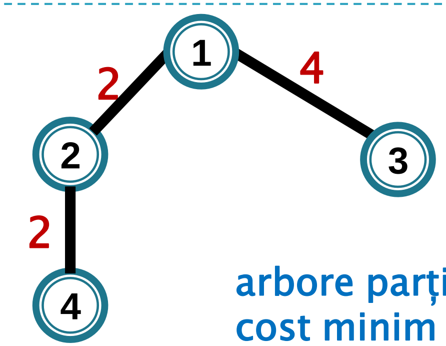
Arbore parțial de cost minim

Algoritmul lui Prim





arbore al drumurilor minime față de 1



arbore parțial de cost minim

Drumuri minime din s ⇒ arbore de drumuri minime (distanțe) din s

≠ **arbore parțial de cost minim** – minimizează costul total

G-(ne)orientat ponderat, ponderi >0

Drumuri minime din s

Algoritmul lui Dijkstra

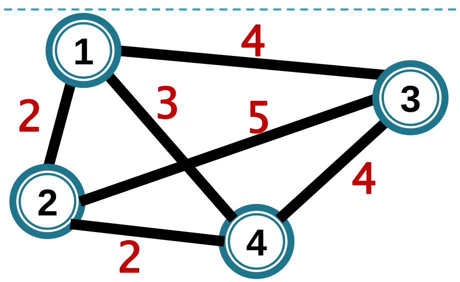
```
Dijkstra(s)
(min-heap) Q ← V
pentru fiecare u∈V
    d[u] = ∞; tata[u]=0
d[s] = 0
cat timp Q ≠ ∅
   u = extrage(Q) vârf cu eticheta
       d minimă
   pentru fiecare uv∈E
      daca ~~v∈Q~~ si d[u]+w(u,v)<d[v]
             d[v] = d[u]+w(u,v)
             tata[v] = u
             repara(v,Q)
scrie d, tata
```

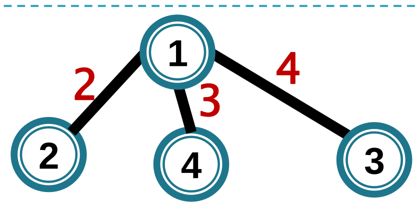
G- neorientat ponderat ponderi reale

Arbore parțial de cost minim

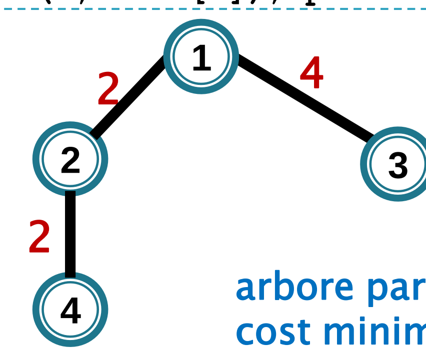
Algoritmul lui Prim

```
Prim(s)
(min-heap) Q ← V
pentru fiecare u∈V
    d[u] = ∞; tata[u]=0
d[s] = 0
cat timp Q ≠ ∅
   u = extrage(Q) vârf cu
       eticheta d minimă
   pentru fiecare uv∈E
      daca v∈Q si w(u,v)<d[v]
             d[v] = w(u,v)
             tata[v] = u
             repara(v,Q)
scrie (u, tata[u]), pentru u≠s
```





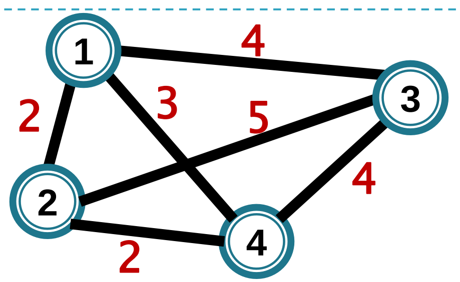
arbore al drumurilor minime față de 1



arbore parțial de cost minim

Drumuri minime din s ⇒ arbore de drumuri minime (distanțe) din s

≠ arbore parțial de cost minim – minimizează costul total



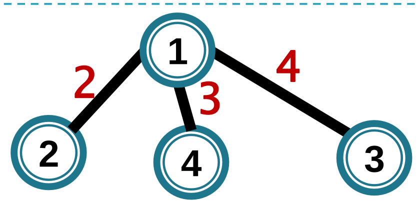
G

G-(ne)orientat ponderat, ponderi >0

Drumuri minime din s

Algoritmul lui Dijkstra

```
Dijkstra(s)
(min-heap) Q ← V
pentru fiecare u∈V
    d[u] = ∞; tata[u]=0
d[s] = 0
cat timp Q ≠ ∅
   u = extrage(Q) vârf cu eticheta
      d minimă
   pentru fiecare uv∈E
      daca v∈Q si d[u]+w(u,v)<d[v]
             d[v] = d[u]+w(u,v)
             tata[v] = u
             repara(v,Q)
scrie d, tata
```



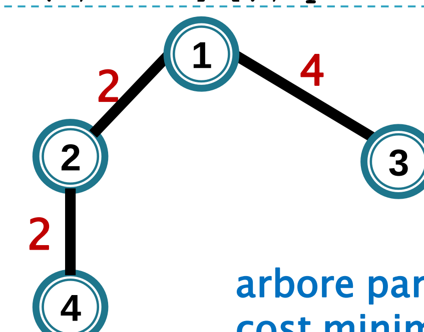
arbore al drumurilor minime față de 1

G- neorientat ponderat ponderi reale

Arbore parțial de cost minim

Algoritmul lui Prim

```
Prim(s)
(min-heap) Q ← V
pentru fiecare u∈V
    d[u] = ∞; tata[u]=0
d[s] = 0
cat timp Q ≠ ∅
   u = extrage(Q) vârf cu
      eticheta d minimă
   pentru fiecare uv∈E
      daca v∈Q si w(u,v)<d[v]
             d[v] = w(u,v)
             tata[v] = u
             repara(v,Q)
scrie (u, tata[u]), pentru u≠s
```



arbore parțial de cost minim

Drumuri minime de sursă unică

- **Relație de recurență – pentru costuri oarecare + circuite (nenegative) ???**

↓

Alt tip de subproblemă – Algoritmul BELLMAN – FORD:

La pasul k: costul minim al unui drum de la s la u cu cel mult k arce

Drumuri minime de sursă unică

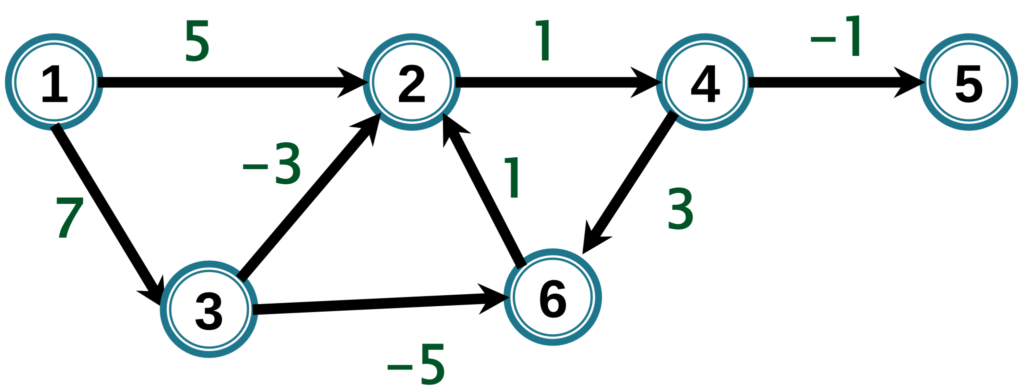
- **Relație de recurență – pentru costuri oarecare + circuite (nenegative) ???**

↓

Alt tip de subproblemă – Algoritmul BELLMAN – FORD:

La pasul k: costul minim al unui drum de la s la u cu cel mult k arce

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



$\delta_1(1, 2) = ?$

$\delta_2(1, 2) = ?$

$\delta_3(1, 2) = ?$

Drumuri minime de sursă unică

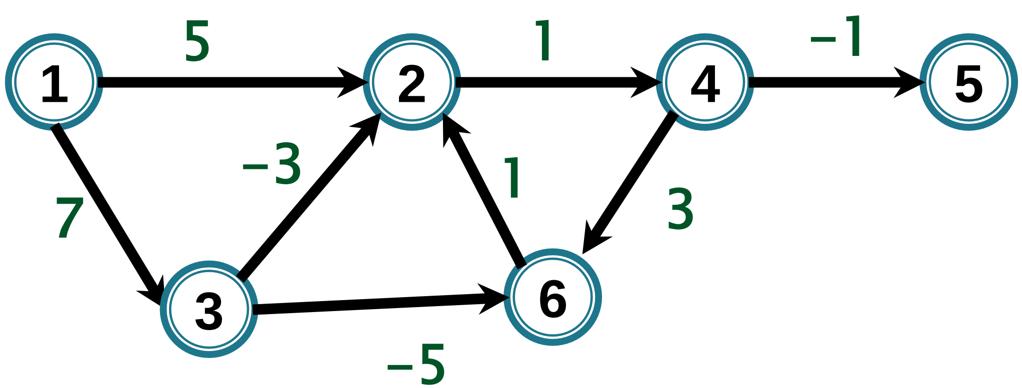
- **Relație de recurență – pentru costuri oarecare + circuite (nenegative) ???**

↓

Alt tip de subproblemă – Algoritmul BELLMAN – FORD:

La pasul k: costul minim al unui drum de la s la u cu cel mult k arce

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



$\delta_1(1, 2) = 5$ – drum [1,2]

$\delta_2(1, 2) = 4$ – drum [1,3,2]

$\delta_3(1, 2) = 3$ – drum [1,3,6,2]

Drumuri minime de sursă unică

- **Relație de recurență – pentru costuri oarecare + circuite (nenegative) ???**

$\downarrow$

Alt tip de subproblemă – Algoritmul BELLMAN – FORD:

La pasul k: costul minim al unui drum de la s la u cu cel mult k arce

$d[u] \leq \delta_k(s,u)$ = costul minim al unui s-x drum **cu cel mult k arce**

$$\delta_k(s,v) = \min\{\delta_{k-1}(s,v) , \min\{\delta_{k-1}(s,v) + w(u,v) \mid uv \in E\}$$

$\downarrow$

La pasul k vom relaxa toate arcele din graf pentru a extinde drumurile (cu cel mult k-1 arce) de la pasul anterior

Algoritmi – G=(V, E) graf orientat

G – ponderat, ponderi reale
(dar fără circuite negative)

Algoritmul lui Bellman Ford

```
Bellman_Ford(s)

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

d[s] = 0

pentru k = 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

scrie d, tata
```

O(nm)

CORECTITUDINI

Corectitudine

- **Lema.** Pentru orice $u \in V$, la orice pas al algoritmului lui Dijkstra/DAG 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) $d[u] \geq \delta(s,u)$

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

Drumuri minime de sursă unică în grafuri aciclice DAG (fără circuite)

Drumuri minime de sursă unică în grafuri aciclice

- **Algoritmul funcționează corect și dacă există arce cu cost negative, calculând etichetele după recurența**

Inductiv, vom detalia

$$d[u] = \min\{ d[x] + w(x,u) \mid xu \in E \} =$$
$$= \min\{ \delta(s,x) + w(x,u) \mid xu \in E \} = \delta(s,u)$$

Deoarece (amintim):

- un drum minim de la s la predecesor x al lui u + arcul xu= un drum de la s la u (de cost $\geq \delta(s,u)$)
- Un drum minim P de la s la u = un drum minim de la s la predecesor x al lui u + arcul xu

Drumuri minime de sursă unică în grafuri aciclice

- **Algoritmul funcționează corect și dacă există arce cu cost negativ – Inducție după numărul de iterații: după fiecare iterație etichetele vârfurilor deja parcurse sunt corecte: $d[u] = \delta(s, u)$**

Drumuri minime de sursă unică în grafuri aciclice

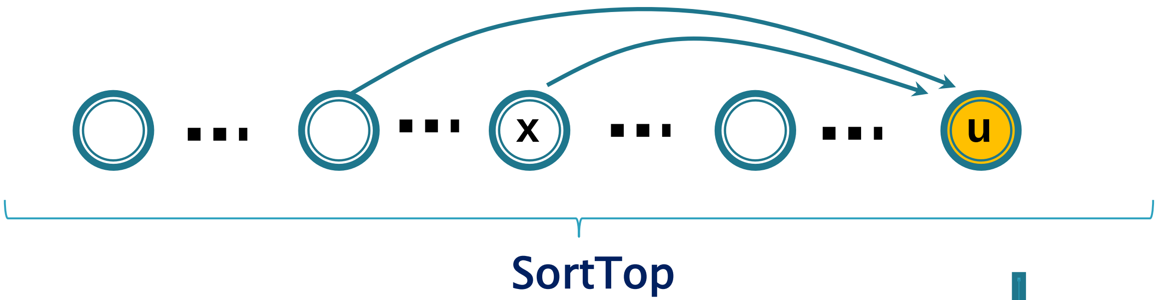
- **Algoritmul funcționează corect și dacă există arce cu cost negativ – Inducție după numărul de iterații:**

$$d[s] = 0 = \delta(s, s)$$

Drumuri minime de sursă unică în grafuri aciclice

- **Algoritmul funcționează corect și dacă există arce cu cost negativ – Inducție după numărul de iterații**

Când algoritmul ajunge la vârful u avem



relaxare arce xu

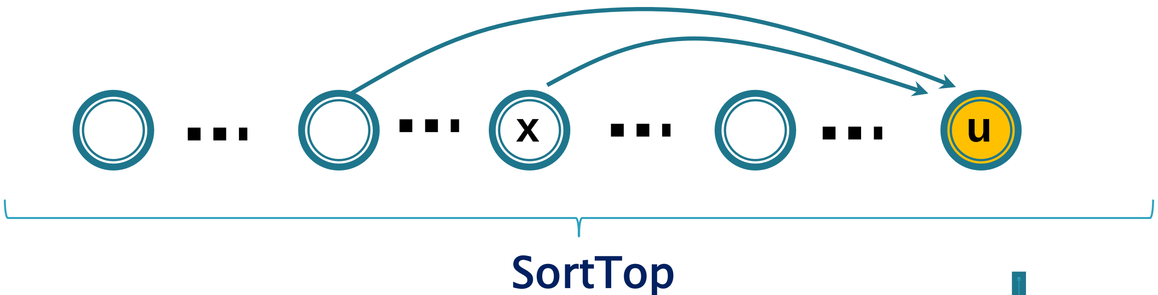
$$d[u] = \min\{\, d[x] + w(x,u) \mid xu \in E,\ \textcolor{red}{x \text{ înaintea lui } u \text{ în SortTop}} \,\}$$
$$= \min\{\, d[x] + w(x,u) \mid xu \in E \,\}$$

deja corect calculate (ipoteza inducție)

Drumuri minime de sursă unică în grafuri aciclice

- **Algoritmul funcționează corect și dacă există arce cu cost negativ – Inducție după numărul de iterații**

Când algoritmul ajunge la vârful u avem



relaxare arce xu

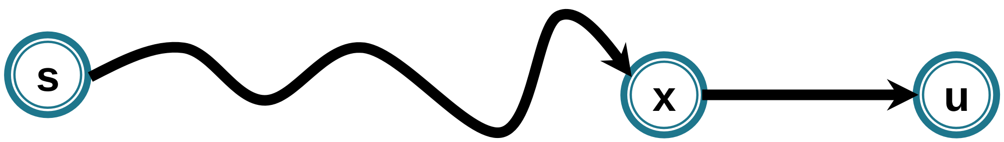
$$d[u] = \min\{\, d[x] + w(x,u) \mid xu \in E,\ \color{red}{x \text{ înaintea lui } u \text{ în SortTop}} \,\}$$
$$= \min\{\, d[x] + w(x,u) \mid xu \in E \,\} = \min\{\delta(s,x) + w(x,u) \mid xu \in E\,\}$$
$$= \delta(s,u)$$

deja corect calculate (ipoteza inducție)

Drumuri minime de sursă unică în grafuri aciclice

Varianta 2 de demonstrație – similar Dijkstra

Fie P s-u drum minim și x predecesorul lui u pe acest drum.



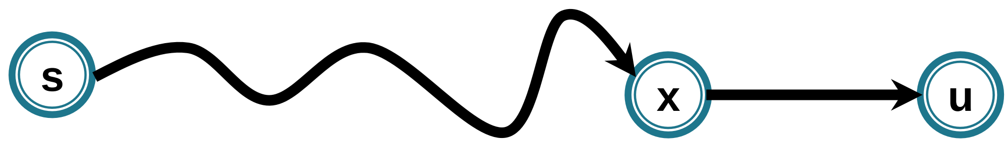
- x este înaintea lui u în SortTop => (ip. inducție)

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

Drumuri minime de sursă unică în grafuri aciclice

Varianta 2 de demonstrație – similar Dijkstra

Fie P s-u drum minim și x predecesorul lui u pe acest drum.



- x este înaintea lui u în SortTop => (ip. inducție)

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

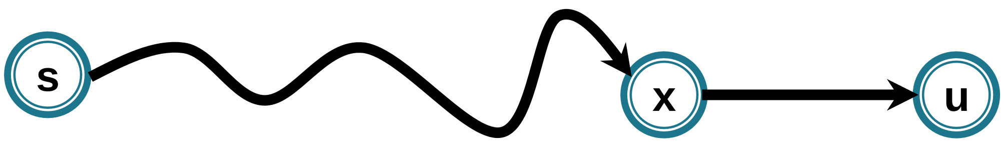
- după relaxarea arcului xu avem:

$$d[u] \leq d[x] + w(xu) = w([s \underline{P} x]) + w(xu) =$$
$$= w([s \underline{P} u]) = w(P) = \delta(s, u)$$

Drumuri minime de sursă unică în grafuri aciclice

Varianta 2 de demonstrație – similar Dijkstra

Fie P s-u drum minim și x predecesorul lui u pe acest drum.



- x este înaintea lui u în SortTop => (ip. inducție)

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

- după relaxarea arcului xu avem:

$$d[u] \le d[x] + w(xu) = w([s \underline{P} x]) + w(xu) =$$
$$= w([s \underline{P} u]) = \delta(s, u)$$

Dar $\delta(s, u) \le d[u]$ (estimare superioară) => $\delta(s, u) = d[u]$

Corectitudinea Algoritmului lui Dijkstra

Corectitudine

- **Teoremă**

  Fie $G=(V, E, w)$ un graf orientat ponderat cu

  $w : E \to \mathbb{R}_+$ și $s \in V$ fixat.

  La finalul algoritmul lui Dijkstra avem:

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

  și tata memorează un arbore al distanțelor față de s.

Corectitudine

- **Demonstraţie** Inducţie: $d[x] = \delta(s, x)\ \ \forall\ x \notin Q$ (=deja selectat)

Corectitudine

- **Demonstraţie** Inducţie: $d[x] = \delta(s, x)$ $\forall$ $x \notin Q$ (=deja selectat)

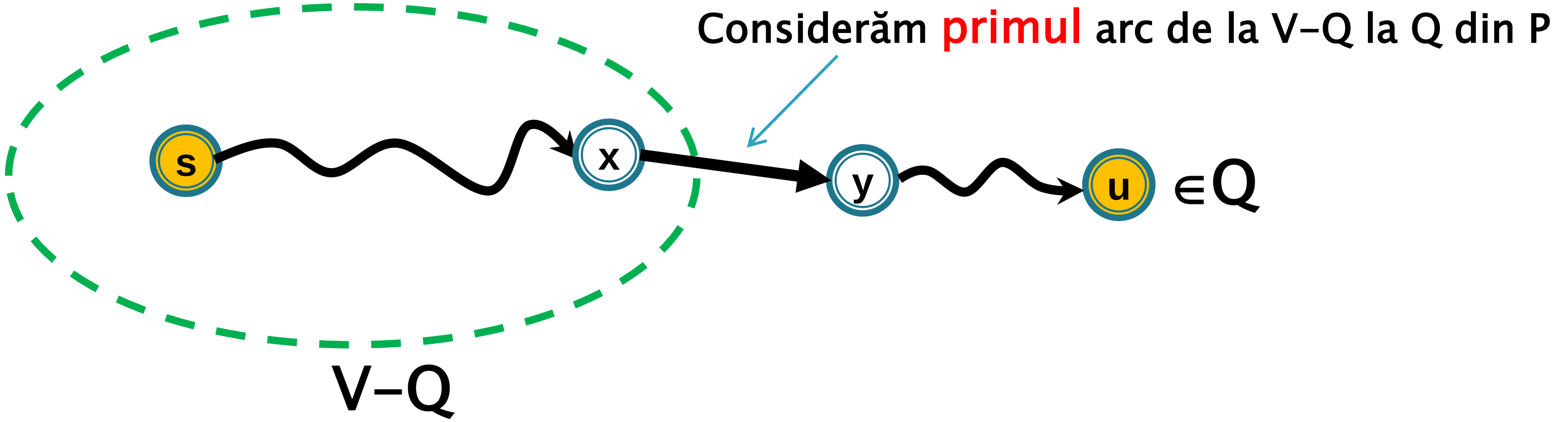
  - Primul vârf selectat este s și $d[s] = 0 = \delta(s, x)$

Corectitudine

- Presupunem la pasul curent $d[x] = \delta(s, x)$ $\forall\ x \notin Q$ (=deja selectat)
- Fie u vârful curent selectat

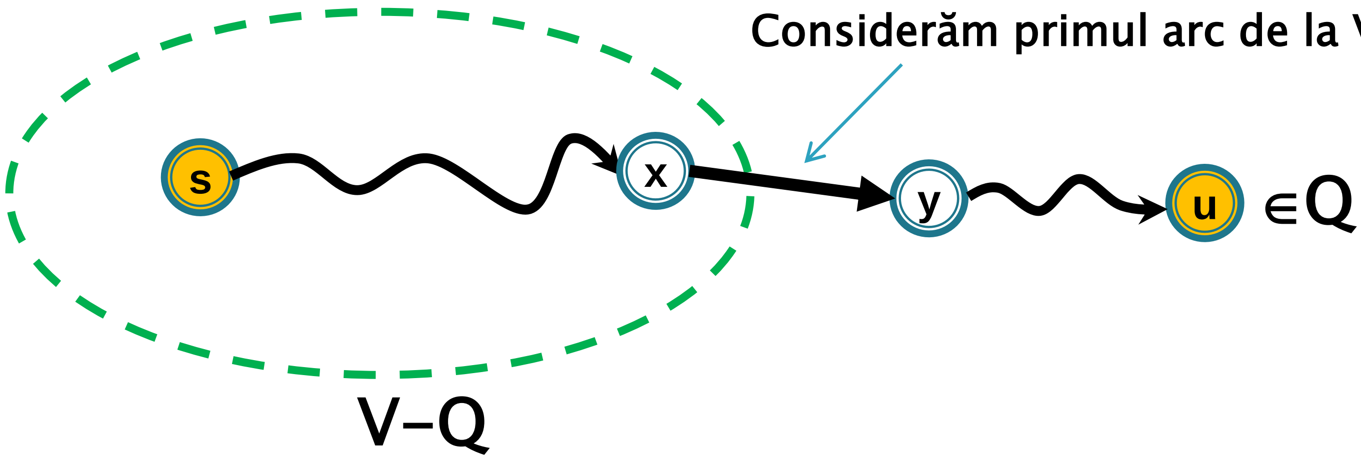
Corectitudine

- Presupunem la pasul curent $d[x] = \delta(s, x)\ \ \forall\ x \notin Q$ (=deja selectat)
- **Fie u vârful curent selectat. Fie P un s-u drum minim**



Corectitudine

- Presupunem la pasul curent d[x] = δ(s, x) ∀ x∉ Q (=deja selectat)
- Fie u vârful curent selectat. Fie P un s-u drum minim



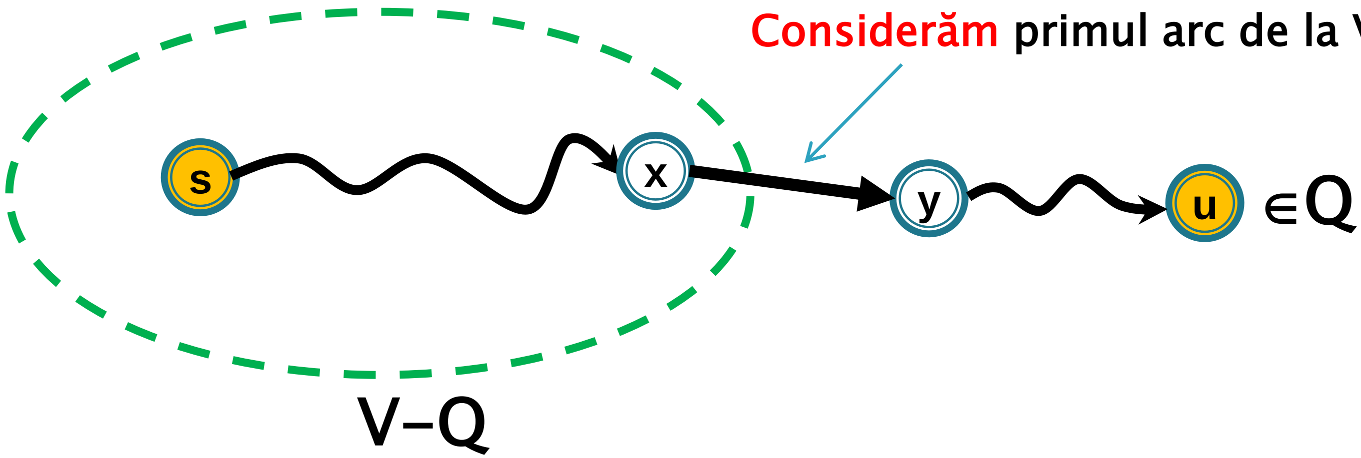
din modul în care este ales u (cu d minim)

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

Corectitudine

- Presupunem la pasul curent d[x] = δ(s, x) ∀ x∉ Q (=deja selectat)
- Fie u vârful curent selectat. Fie P un s-u drum minim

Considerăm primul arc de la V-Q la Q din P



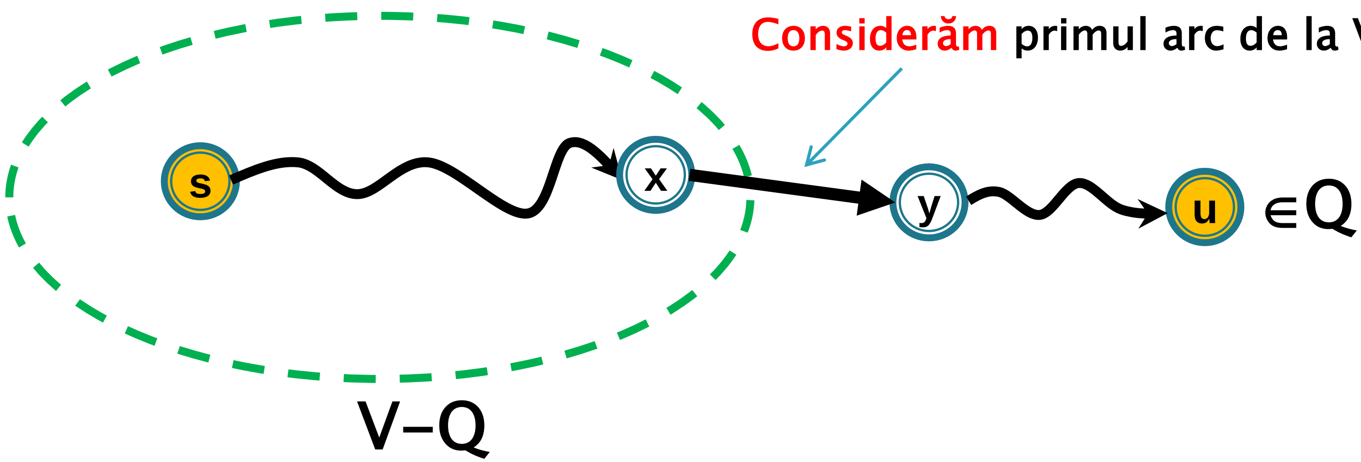
dupa relaxarea lui xy (!!x a dost deja selectat)

din modul în care este ales u

$$d[u] \le d[y] \le$$

Corectitudine

- Presupunem la pasul curent $d[x] = \delta(s, x)$ $\forall x \notin Q$ (=deja selectat)
- Fie u vârful curent selectat. Fie P un s-u drum minim



din modul în care este ales u

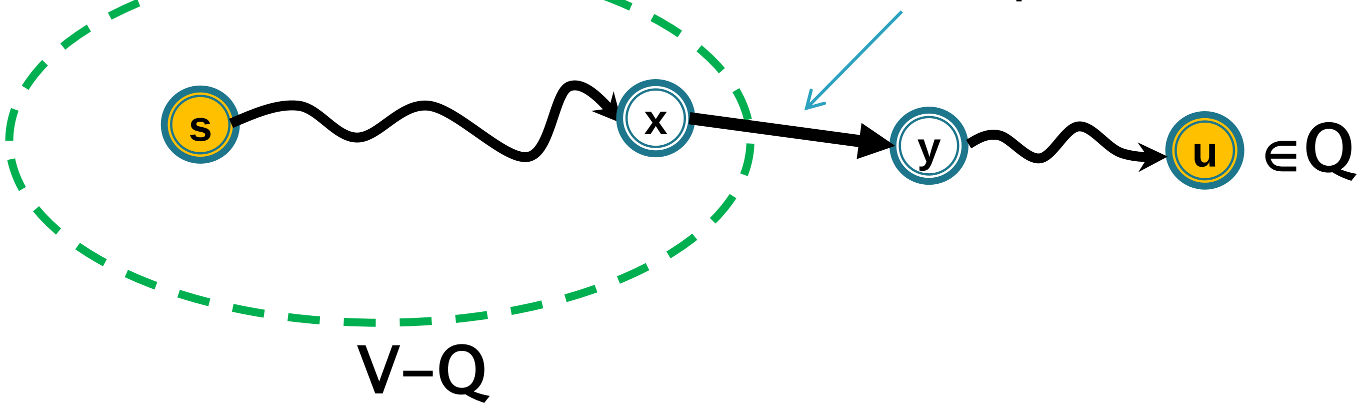
dupa relaxarea lui xy (!!x a dost deja selectat)

$$d[u] \le d[y] \le d[x] + w(x, y)$$

Corectitudine

- Presupunem la pasul curent d[x] = δ(s, x) ∀ x∉ Q (=deja selectat)
- Fie u vârful curent selectat. Fie P un s-u drum minim

Considerăm primul arc de la V-Q la Q din P



din modul în care este ales u

dupa relaxarea lui xy (!!x a dost deja selectat)

$$d[u] \le d[y] \le d[x] + w(x, y) = \delta(s, x) + w(x, y) = w(s \overset{P}{\text{—}} y)$$

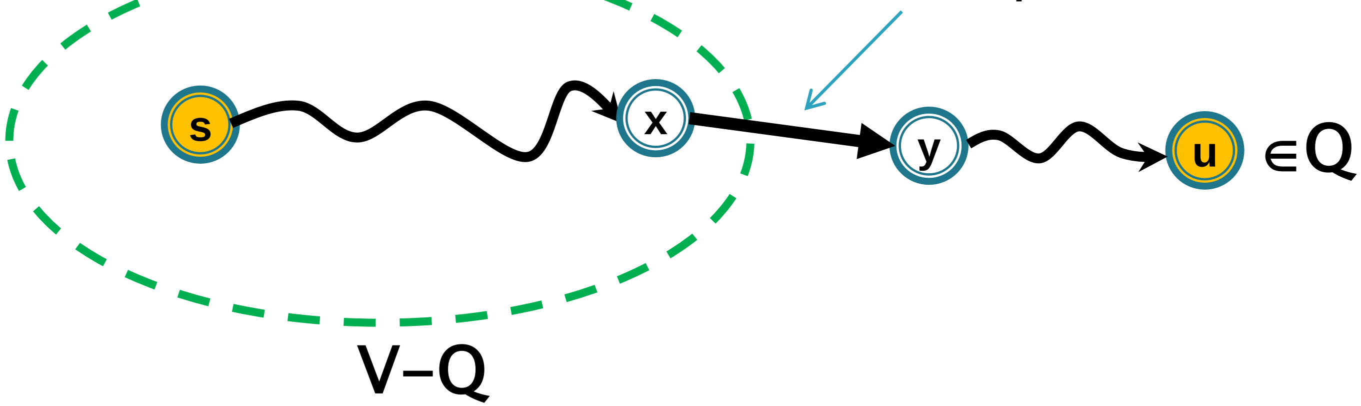
$$\le$$

ipoteza de inductie pentru x

Corectitudine

- Presupunem la pasul curent **d[x] = δ(s, x) ∀ x∉ Q (=deja selectat)**
- **Fie u vârful curent selectat. Fie P un s-u drum minim**

Considerăm **primul arc de la V-Q la Q din P**



din modul în care este ales u

dupa relaxarea lui xy (!!x a dost deja selectat)

$$d[u] \le d[y] \le d[x] + w(x,y) = \delta(s,x) + w(x,y) = w(s \overset{P}{—} y)$$

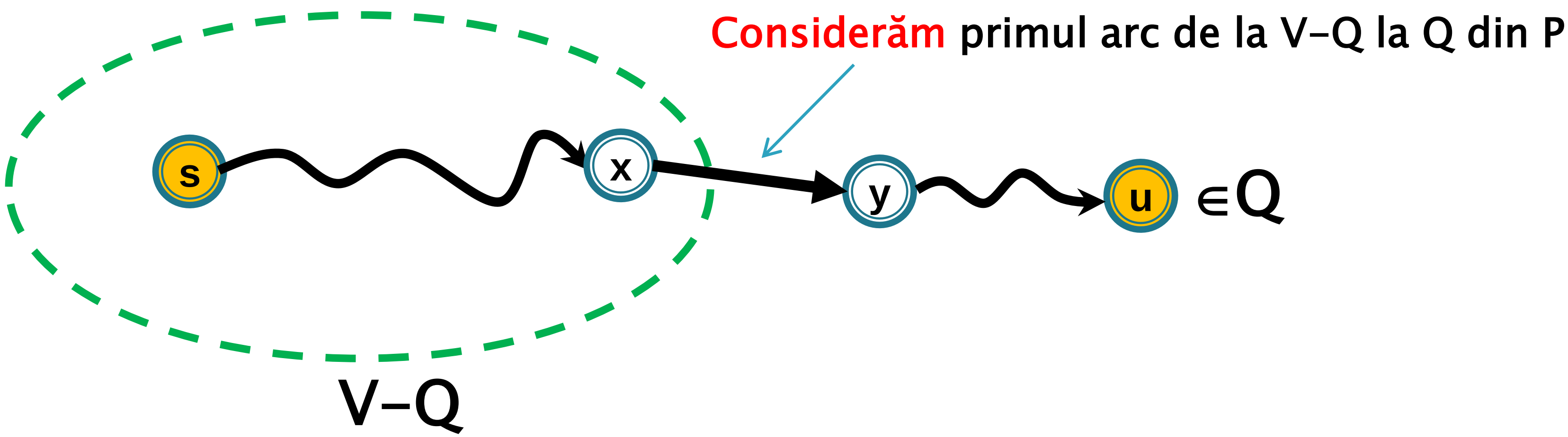
$$\le$$

ipoteza de inductie pentru x

costurile arcelor sunt nenegative

Corectitudine

- Presupunem la pasul curent **d[x] = δ(s, x) ∀ x∉ Q (=deja selectat)**
- **Fie u vârful curent selectat. Fie P un s-u drum minim**



din modul în care este ales u

dupa relaxarea lui xy (!!x a dost deja selectat)

$$d[u] \le d[y] \le d[x] + w(x,y) = \delta(s,x) + w(x,y) = w(s \overset{P}{-} y)$$

$$\le w(P)$$

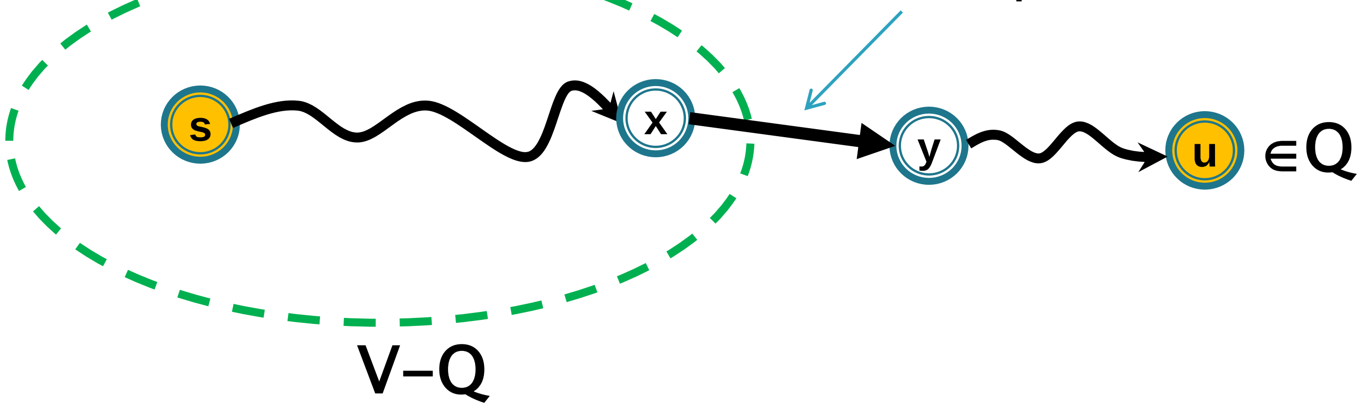
ipoteza de inductie pentru x

costurile arcelor sunt nenegative

Corectitudine

- Presupunem la pasul curent d[x] = δ(s, x) ∀ x∉ Q (=deja selectat)
- Fie u vârful curent selectat. Fie P un s-u drum minim

Considerăm primul arc de la V-Q la Q din P



din modul în care este ales u

dupa relaxarea lui xy (!!x a dost deja selectat)

$$d[u] \le d[y] \le d[x] + w(x,y) = \delta(s,x) + w(x,y) = w(s \overset{P}{—} y)$$

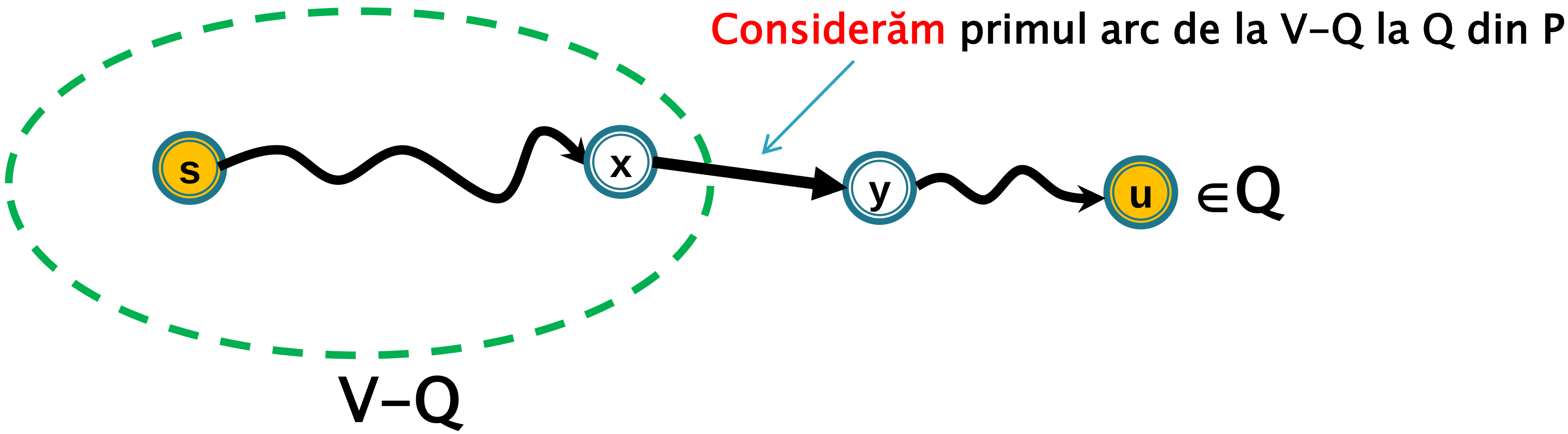
$$\le w(P) \le d[u]$$

ipoteza de inductie pentru x

Din Lema, d[u] = costul unui drum de la s la u,
deci este >= costul drumului minim de la s la u

Corectitudine

- Presupunem la pasul curent d[x] = δ(s, x) $\forall$ x$\notin$ Q (=deja selectat)
- Fie u vârful curent selectat. Fie P un s-u drum minim



$$d[u] \le d[y] \le d[x] + w(x,y) = \delta(s,x) + w(x,y) = w(s \overset{P}{\text{—}} y)$$

$$\le w(P) \le d[u]$$

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

Corectitudinea Algoritmului lui Bellman-Ford

Corectitudine

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

După k iterații

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

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

La final vom avea $\delta(s,x) \leq d[x] \leq \delta_{n-1}(s,x) = \delta(s,x)$,

deci $d[x] = \delta(s,x)$

Corectitudine

- Demonstram inductiv: $d[x] \leq \delta_k(s,x)$

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

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

Corectitudine

- Demonstram inductiv: $d[x] \leq \delta_k(s,x)$

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

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

- $k-1 \Rightarrow k$. Presupunem că înainte de iterația k

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

Eticheta unui vârf y la iterația k se actualizează astfel:

Corectitudine

se relaxează toate arcele

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

Corectitudine

ipoteza de inducție

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

Corectitudine

ipoteza de inducție

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

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

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

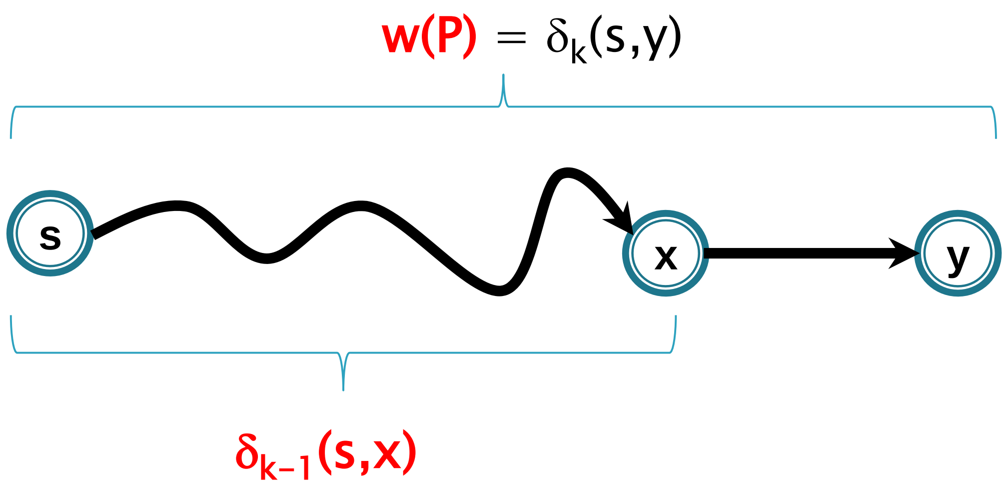
Corectitudine

Varianta 2.

- $k-1 \Rightarrow k$. Presupunem că înainte de iterația k

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

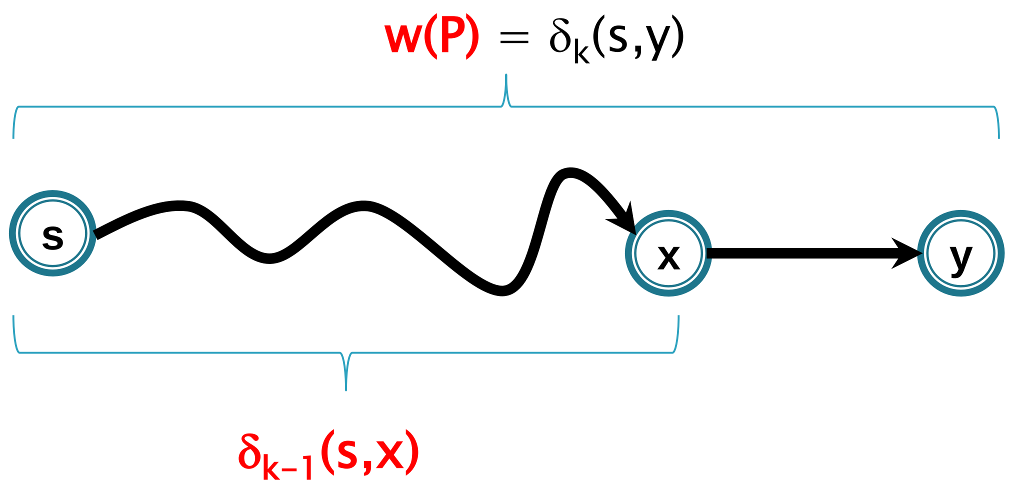
Fie P un s-y drum cu cost minim printre cele cu cel mult k arce
($w(P) = \delta_k(s,y)$)



Corectitudine

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

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



Corectitudine

După relaxarea arcului xy:

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