

Drumuri minime în grafuri ponderate

Aplicații



- ▶ Dată o hartă rutieră, să se determine
 - un drum minim între două orașe date
 - câte un drum minim între oricare două orașe de pe hartă

Aplicații

- ▶ Determinarea de drumuri minime/distanțe – numeroase aplicații
 - probleme de rutare
 - robotică
 - procesarea imaginilor
 - strategii jocuri

Cadru

Fie:

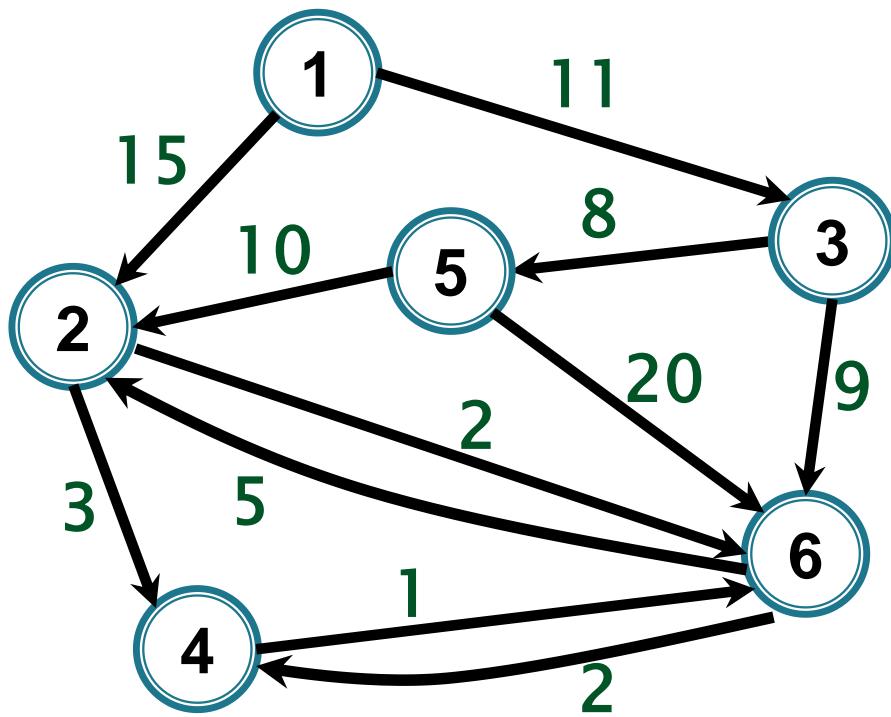
- ▶ G – graf **orientat** ponderat
- ▶ P – drum

$$w(P) = \sum_{e \in E(P)} w(e)$$

costul/ponderea/lungimea drumului P

Cadru

- ▶ Fie $s, v \in V$, $s \neq v$.
- ▶ **Distanța de la s la v**
$$\delta_G(s, v) = \min\{ w(P) \mid P \text{ este drum de la } s \text{ la } v\}$$
 - $\delta_G(s, s) = 0$
 - **Convenție.** $\min \emptyset = \infty$
- ▶ Un drum P de la s la v se numește **drum minim de la s la v** dacă $w(P) = \delta_G(s, v)$



Tipuri de probleme de drumuri minime

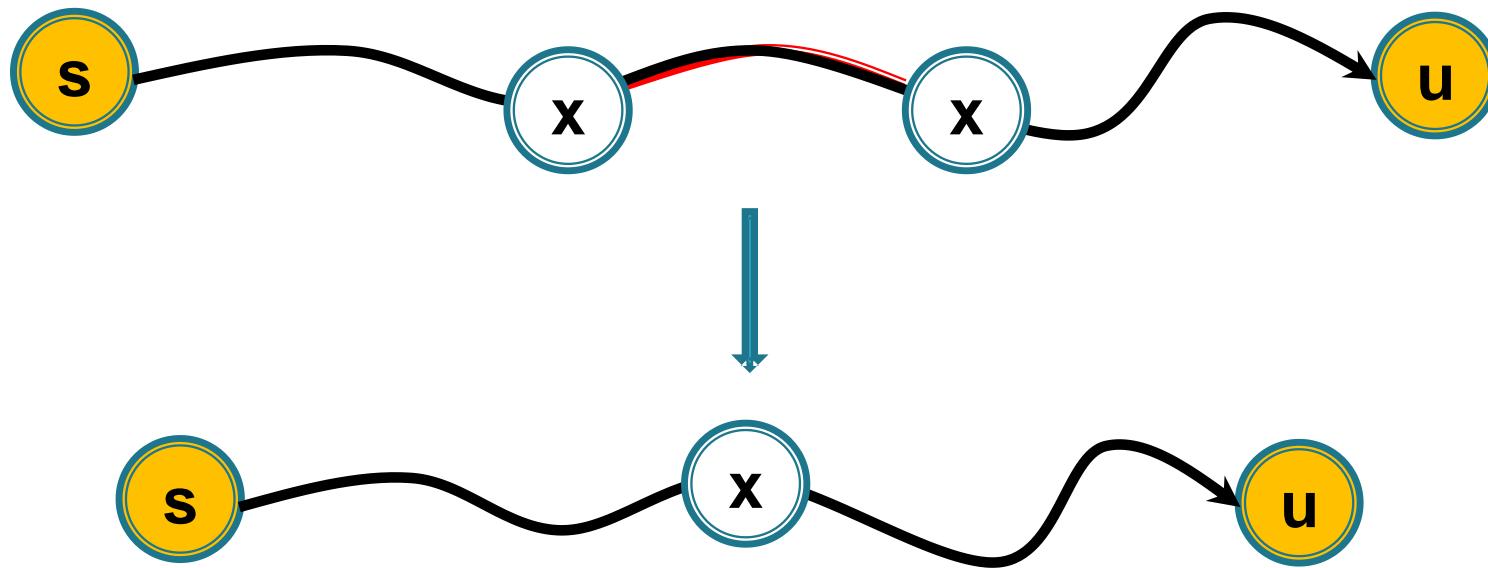
- Între două vârfuri date
- de la un vârf la toate celelalte
- Între oricare două vârfuri

Situări:

- Sunt acceptate și arce de cost negativ?
- Graful conține circuite?
- Graful conține circuite de cost negativ?

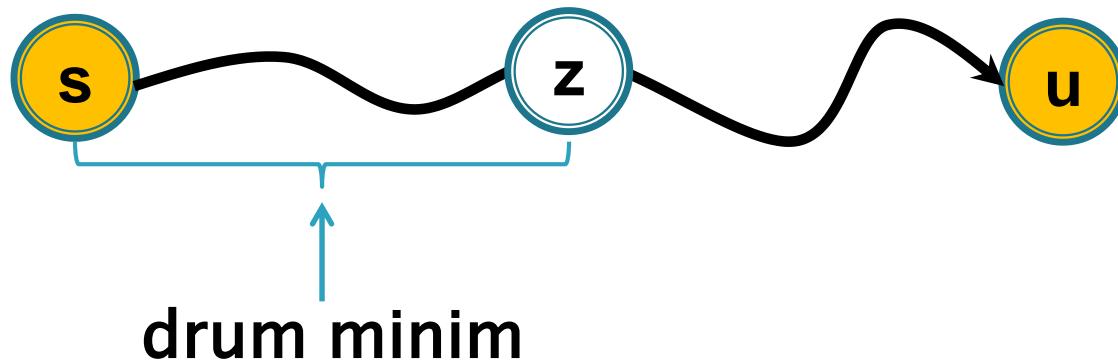
Observații

- ▶ **Observația 1.** Dacă P este un drum minim de la s la u și **nu există circuite de cost negativ**, atunci P este drum elementar.

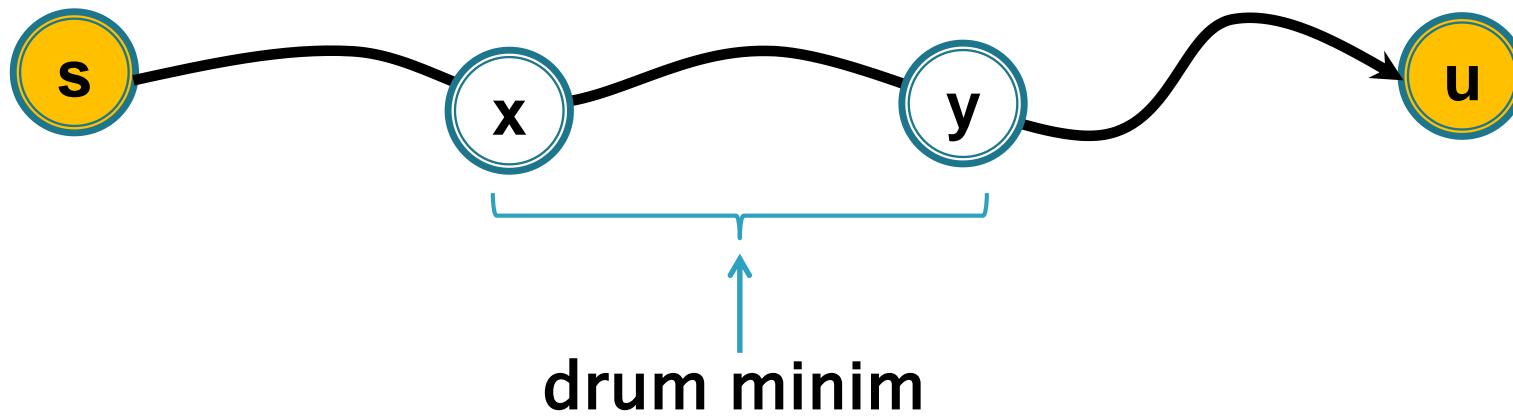


Observații

- ▶ **Observația 2.** Dacă P este un drum minim de la s la u și z este un vârf al lui P , atunci subdrumul lui P de la s la z este drum minim de la s la z .



Observații



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

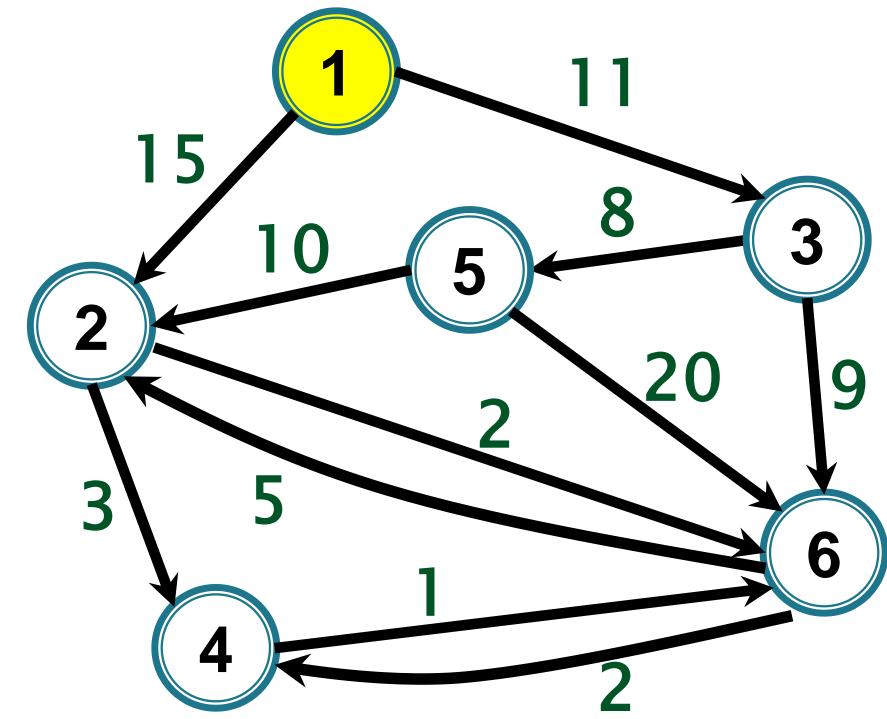
Problema drumurilor minime de sursă unică (de la s la celealte vârfuri)

Se dau:

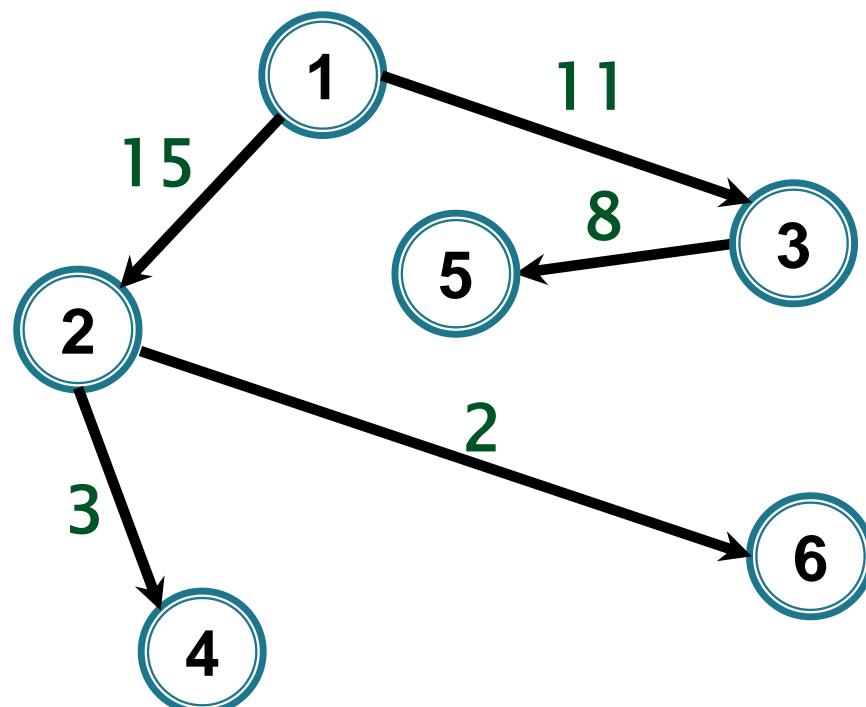
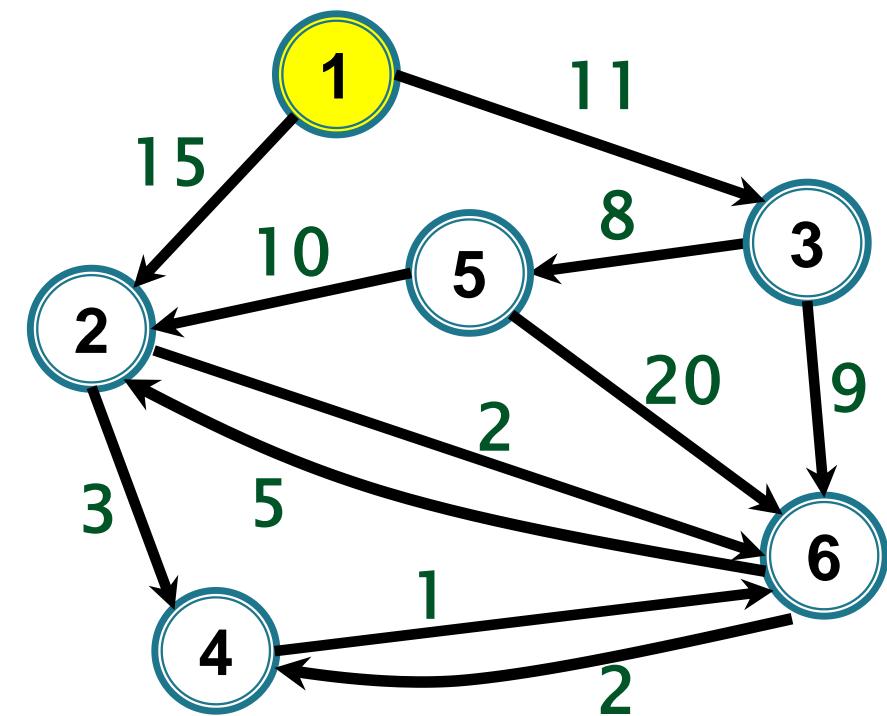
- un graf **orientat** ponderat $G = (V, E, w)$, cu
 $w : E \rightarrow \mathbb{R}$
- un vârf de start **s**

Să se determine distanța de la s la fiecare vârf x al lui G / la un vârf dat t (și un arbore al distanțelor față de s/ un drum minim de la s la t)

$s = 1$



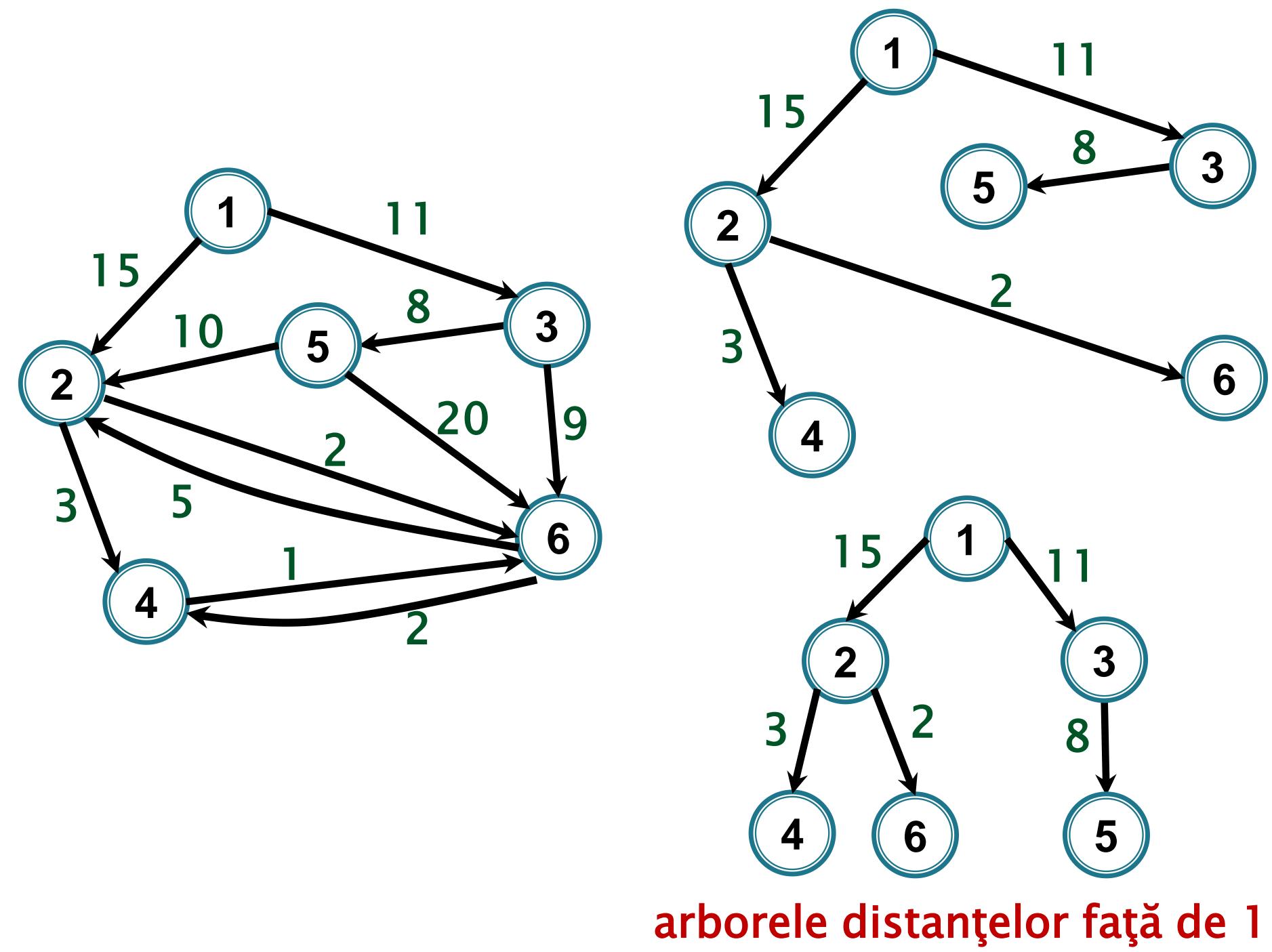
Soluție



Definiție: Pentru un vârf dat s un arbore al distanțelor față de s = un subgraf T al lui G care conservă distanțele de la s la celealte vârfuri accesibile din s

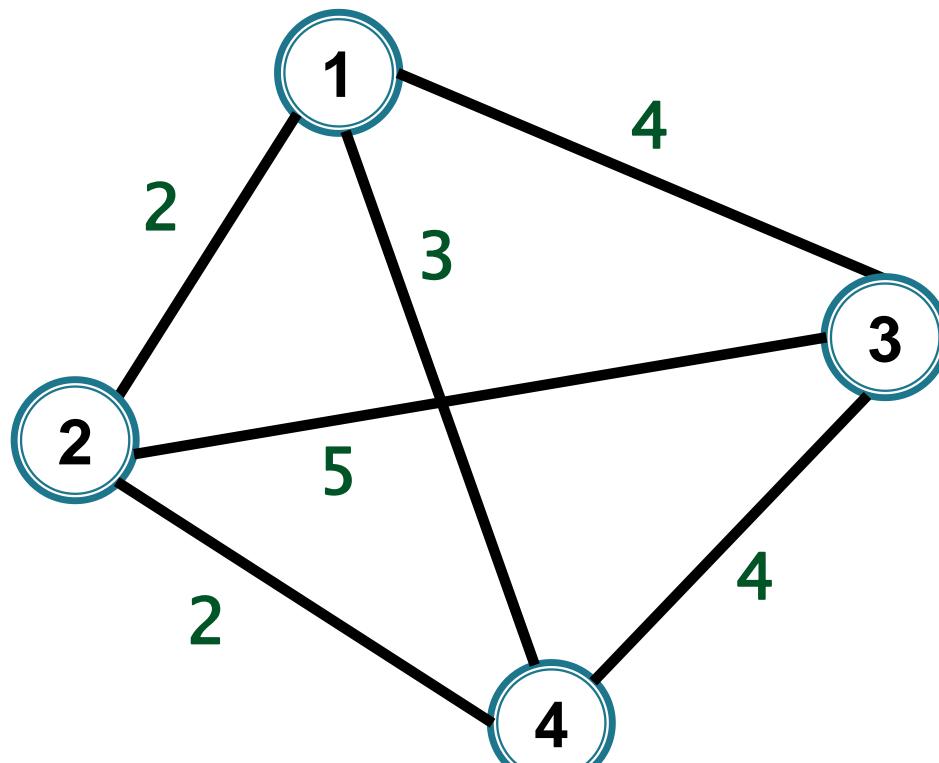
$$\delta_T(s, v) = \delta_G(s, v), \quad \forall v \in V \text{ accesibil din } s,$$

graful neorientat asociat lui T fiind arbore cu rădăcina în s
(cu arcele corespunzătoare orientate de la s la frunze)

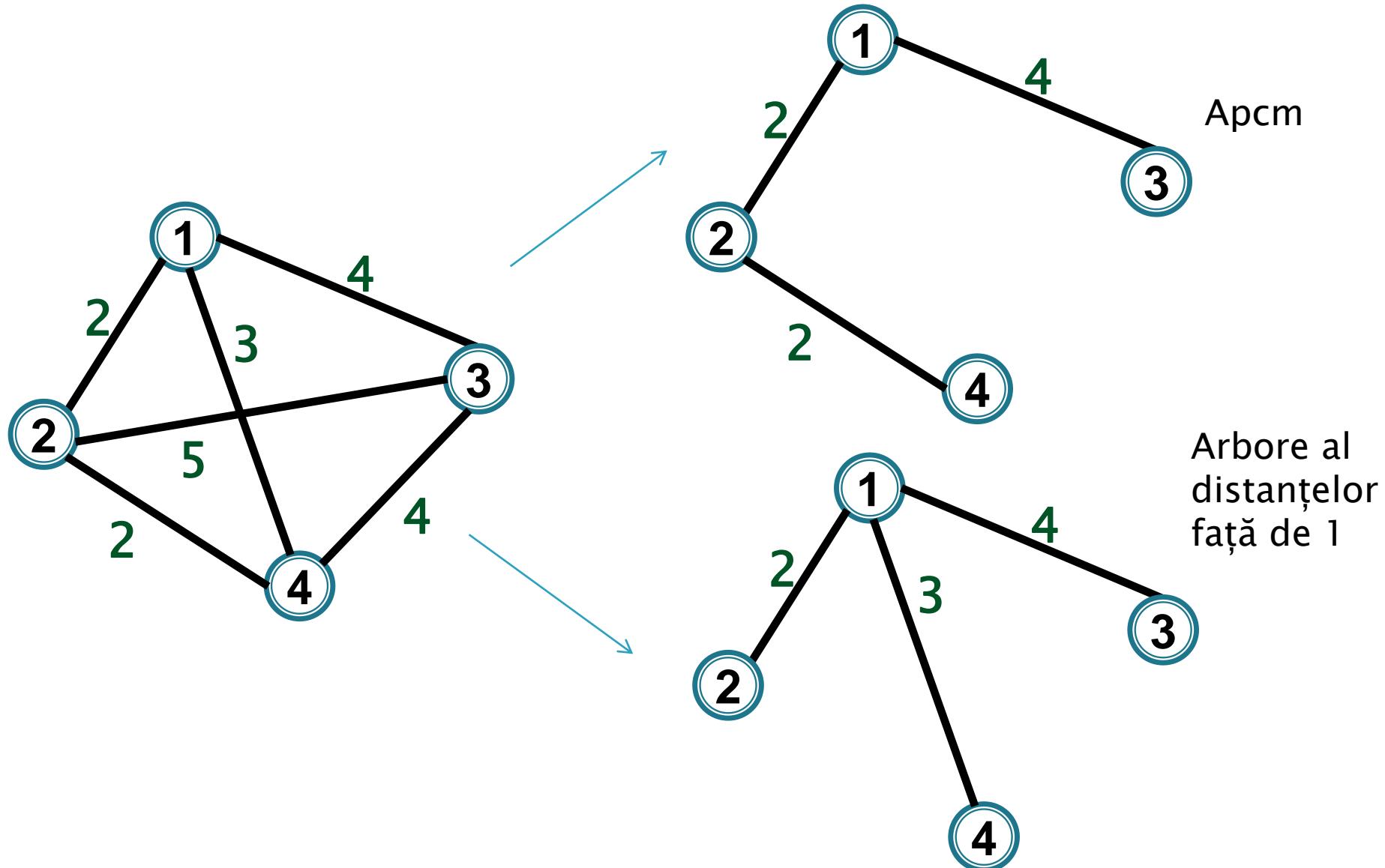


- ▶ Presupunem că **toate vârfurile sunt accesibile din s**
- ▶ Problema drumurilor minime de sursă unică este echivalentă cu determinarea unui **arbore al distanțelor față de s**

- ▶ Un arbore parțial de cost minim nu este neapărat un arbore de distanțe minime



Un arbore parțial de cost minim nu este neapărat un arbore de distanțe minime



Drumuri minime de sursă unică s



- ▶ Dacă G **nu** este ponderat, cum putem calcula distanțele față de s ?

Drumuri minime de sursă unică s

► Amintim

În cazul unui graf neponderat, problema se rezolvă folosind parcurgerea BF din s

Drumuri minime de sursă unică s



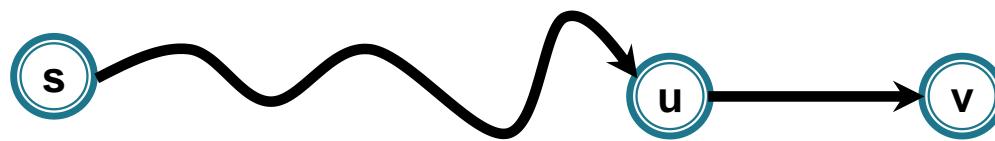
- ▶ În ce ordine considerăm vârfurile pentru a calcula distanțele față de s?

Drumuri minime de sursă unică s

- ▶ În ce ordine considerăm vârfurile pentru a calcula distanțele față de s?



“din aproape în aproape”



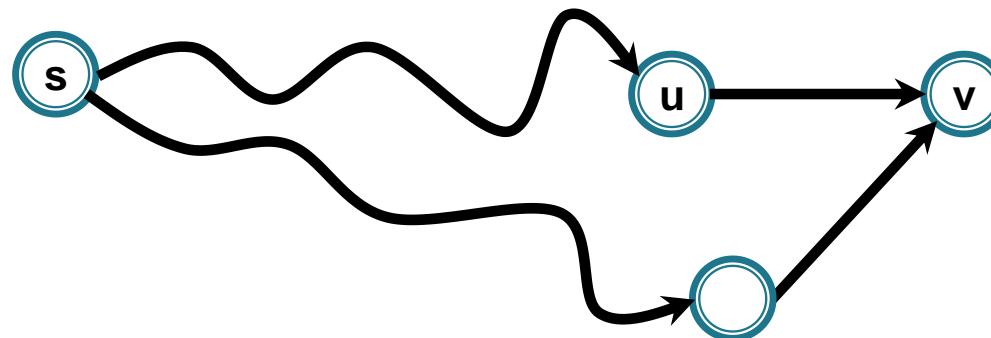
Drumuri minime de sursă unică s

- În ce ordine considerăm vârfurile pentru a calcula distanțele față de s?



“din aproape în aproape”

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



Drumuri minime de sursă unică s

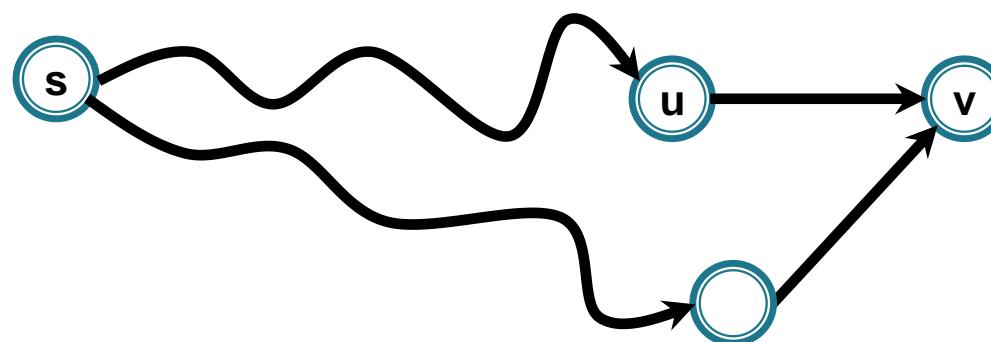
- În ce ordine considerăm vârfurile pentru a calcula distanțele față de s?



“din aproape în aproape” \Rightarrow

când considerăm un vârf v , **pentru a calcula $\delta(s,v)$** ar fi util să știm deja $\delta(s,u)$ pentru orice u cu $uv \in E$

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



Drumuri minime de sursă unică s

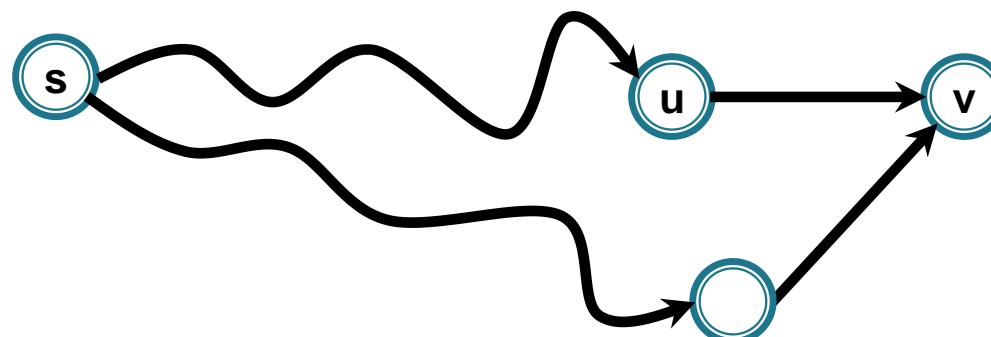
- ▶ În ce ordine considerăm vârfurile pentru a calcula distanțele față de s?



“din aproape în aproape” \Rightarrow

când considerăm un vârf v , pentru a calcula $\delta(s,v)$ ar fi util să știm deja $\delta(s,u)$ pentru orice u cu $uv \in E$

- Ar fi utilă o ordonare a vârfurilor astfel încât dacă $uv \in E$, atunci u se află înaintea lui v



Drumuri minime de sursă unică s

- ▶ În ce ordine considerăm vârfurile pentru a calcula distanțele față de s?
 - “din aproape în aproape” \Rightarrow când considerăm un vârf v, pentru a calcula $\delta(s,v)$ ar fi util să știm deja $\delta(s,u)$ pentru orice u cu $uv \in E$
 - Ar fi utilă o ordonare a vârfurilor astfel încât dacă $uv \in E$, atunci u se află înaintea lui v



O astfel de ordonare nu există dacă graful conține circuite

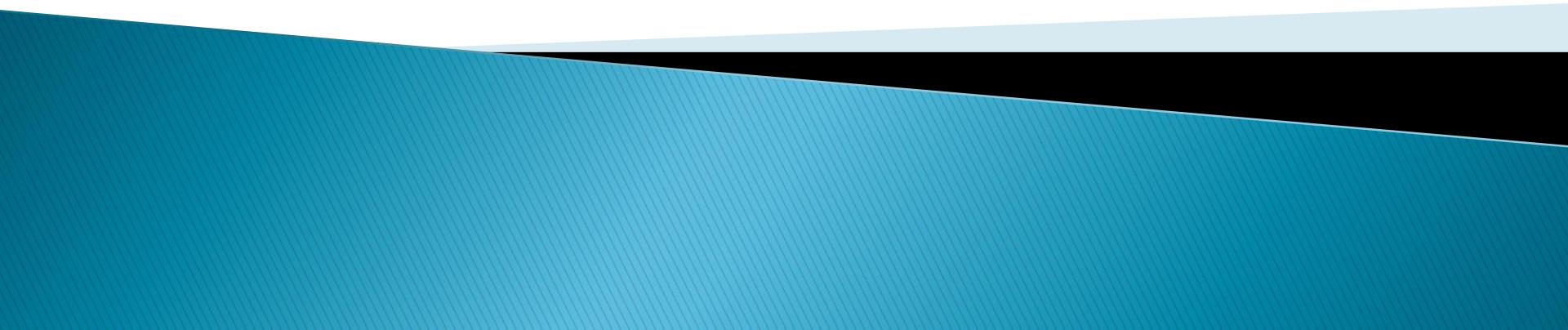
Drumuri minime de sursă unică s

- ▶ În ce ordine considerăm vârfurile pentru a calcula distanțele față de s?
“din aproape în aproape”



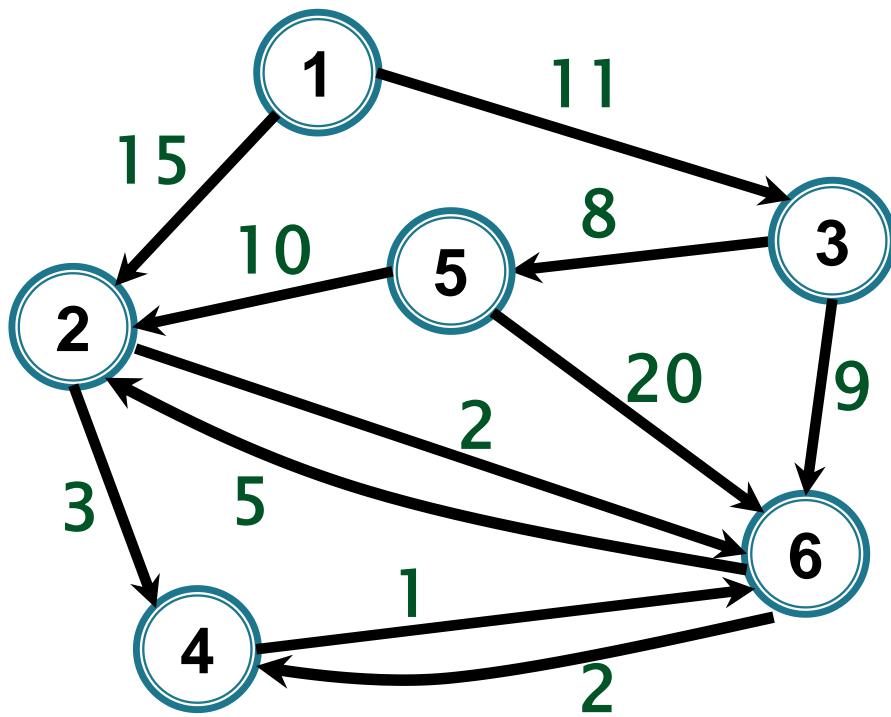
Idee: **estimăm** distanțele pe parcursul algoritmului și considerăm vârful care este estimat a fi cel mai aproape de s

Algoritmul lui Dijkstra



► Ipoteză:

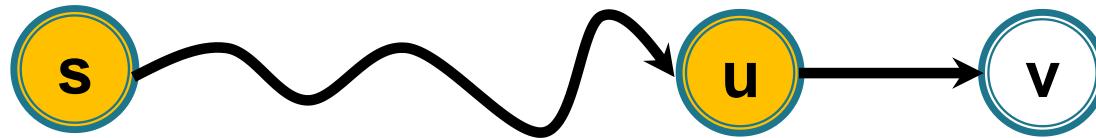
Presupunem că arcele au cost nenegativ
(graful poate conține circuite)



Algoritmul lui Dijkstra

Idee: La un pas este ales ca **vârf curent** vârful u care estimat a fi cel mai apropiat de s

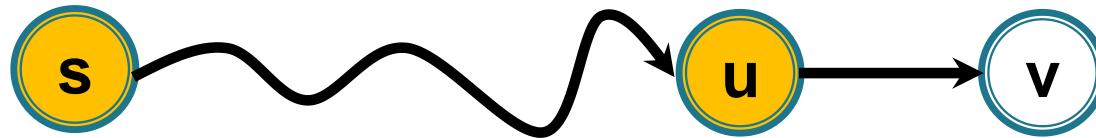
Estimarea pentru u = cel mai scurt drum de la s la u determinat până la pasul curent



Algoritmul lui Dijkstra

Idee: La un pas este ales ca vârf curent vârful u care **estimat** a fi cel mai apropiat de s

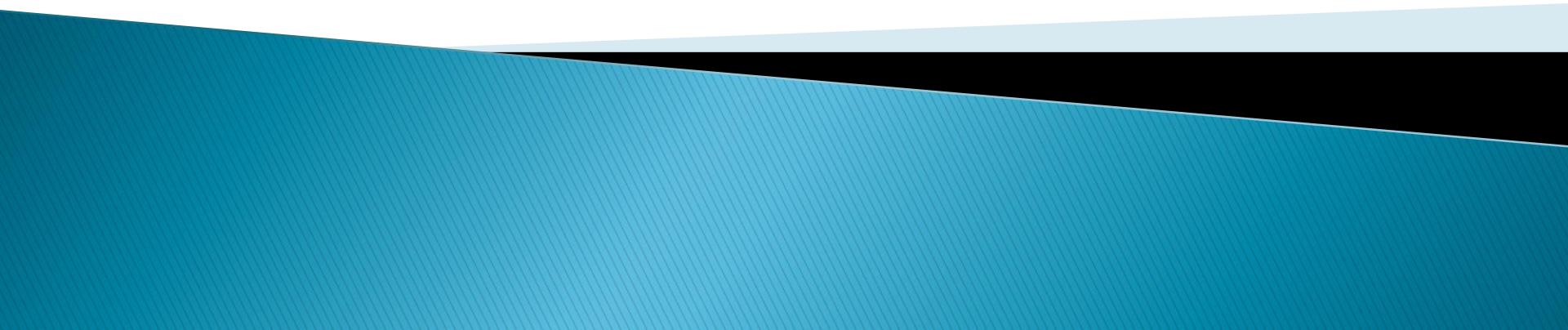
+ se descoperă noi drumuri către vecinii lui \Rightarrow
se actualizează distanțele estimate pentru vecinii



Algoritmul lui Dijkstra

- generalizare a ideii de parcurgere BF
- dacă toate arcele au cost egal Dijkstra \equiv BF

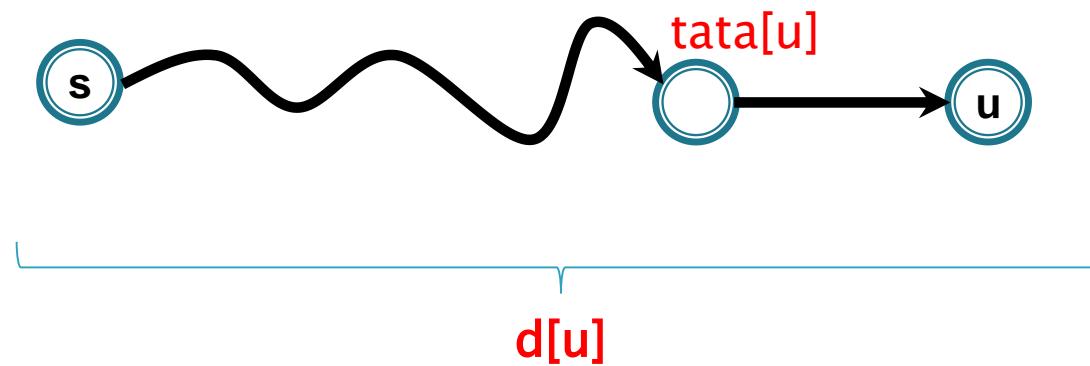
Pseudocod



Algoritmul lui Dijkstra

► Reținem pentru fiecare vârf etichetele

- $d[u]$ – etichetă de distanță
- $tata[u]$



Algoritmul lui Dijkstra

- ▶ Etichetă de distanță

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

Algoritmul lui Dijkstra

- ▶ Etichetă de distanță
 - $d[u]$ = **costul minim al unui drum de la s la u descoperit până la acel moment**
- ▶ $d[u]$ estimare superioară ($\geq d(s,u)$)

Algoritmul lui Dijkstra

- ▶ Memorare drumuri
 - reținem **predecesor** = tată în arborele distanțelor față de vârful s:
tata[u] = **predecesorul** lui u pe drumul de cost minim de la s la u descoperit până la acel moment

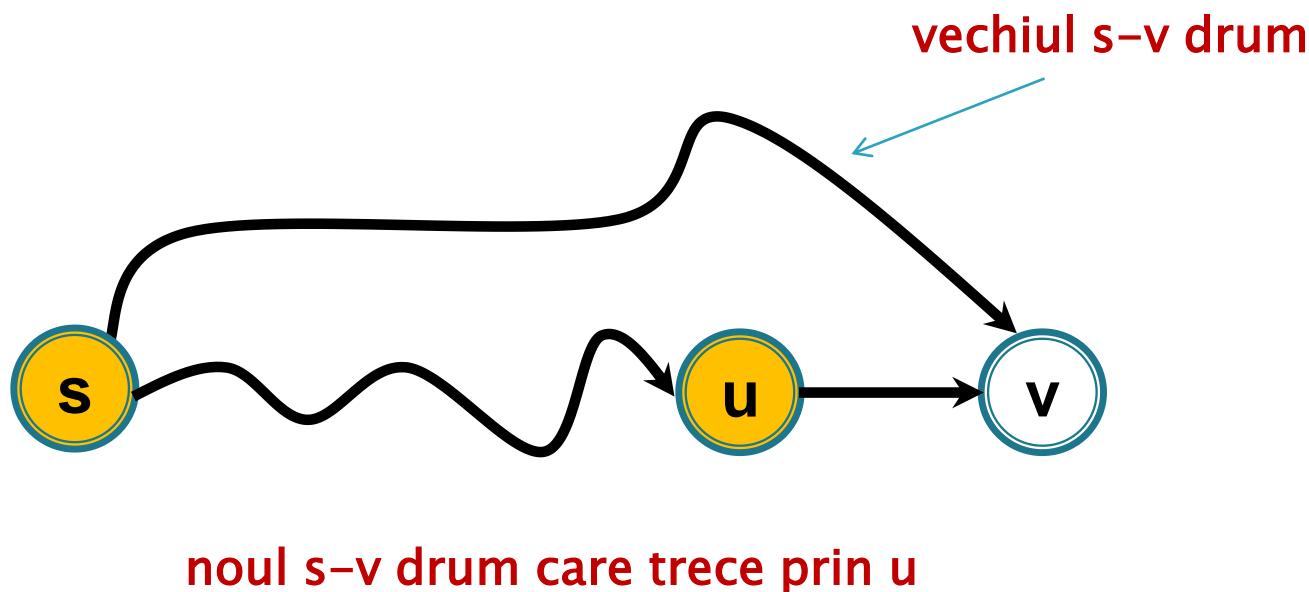
Algoritmul lui Dijkstra

► Astfel, la un pas

- este selectat un vârf u (neselectat) care “pare” cel mai apropiat de $s \Leftrightarrow$ are eticheta d minimă
- Se actualizează etichetele $d[v]$ ale vecinilor lui u – considerând drumuri care trec prin u
 - tehnică de relaxare a arcelor care ies din u

Algoritmul lui Dijkstra

- ▶ Relaxarea unui arc $(u, v) =$ a verifica dacă $d[v]$ poate fi îmbunătățit trecând prin vârful u



Algoritmul lui Dijkstra

- ▶ Relaxarea unui arc (u, v) :

dacă $d[u] + w(u, v) < d[v]$ atunci

$d[v] = d[u] + w(u, v)$;

tata[v] = u

- ▶ $\infty + X = \infty$

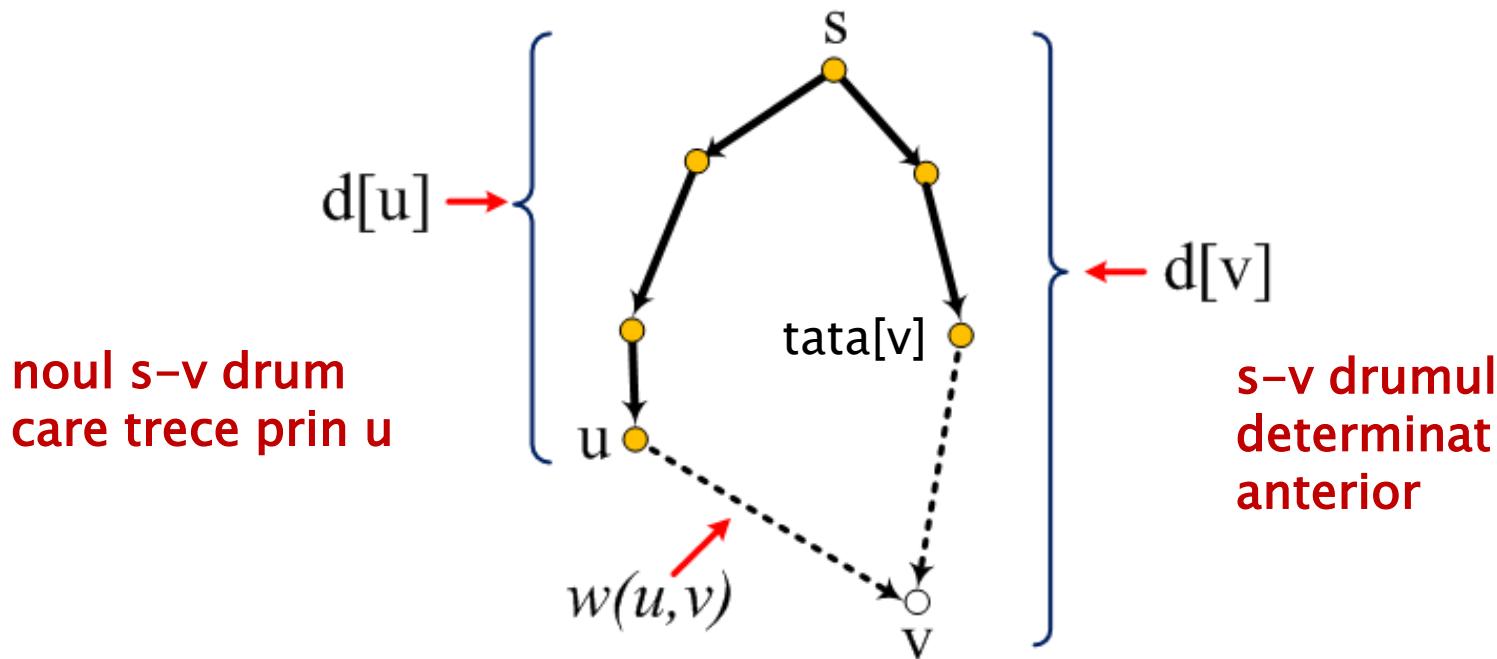
Algoritmul lui Dijkstra

- Relaxarea unui arc (u, v) :

dacă $d[u] + w(u, v) < d[v]$ atunci

$d[v] = d[u] + w(u, v)$;

$tata[v] = u$



Dijkstra(G, w, s)

initializează multimea vârfurilor neselectate Q cu V
pentru fiecare $u \in V$ executa

$d[u] = \infty$; $tata[u] = 0$

$d[s] = 0$

cat timp $Q \neq \emptyset$ executa \Leftrightarrow pentru $i=1, n$ executa

Dijkstra(G, w, s)

initializează multimea vârfurilor neselectate Q cu V
pentru fiecare $u \in V$ executa

$d[u] = \infty$; $tata[u] = 0$

$d[s] = 0$

cat timp $Q \neq \emptyset$ executa

$u = \text{extrage vârf cu eticheta } d \text{ minimă din } Q$

pentru fiecare $uv \in E$ executa //relaxare uv

Dijkstra(G, w, s)

initializează multimea vârfurilor neselectate Q cu V
pentru fiecare $u \in V$ executa

$d[u] = \infty$; $tata[u] = 0$

$d[s] = 0$

căt timp $Q \neq \emptyset$ executa

$u = \text{extrage vârf cu eticheta } d \text{ minimă din } Q$

pentru fiecare $uv \in E$ executa

daca ~~$v \in Q$~~ si $d[u] + w(u, v) < d[v]$ atunci

$d[v] = d[u] + w(u, v)$

$tata[v] = u$

Dijkstra(G, w, s)

initializează multimea vârfurilor neselectate Q cu V
pentru fiecare $u \in V$ executa

$d[u] = \infty$; tata[u]=0

$d[s] = 0$

cat timp $Q \neq \emptyset$ executa

$u = \text{extrage vârf cu eticheta } d \text{ minimă din } Q$

pentru fiecare $uv \in E$ executa

daca $d[u] + w(u, v) < d[v]$ atunci

$d[v] = d[u] + w(u, v)$

tata[v] = u

scrie d, tata

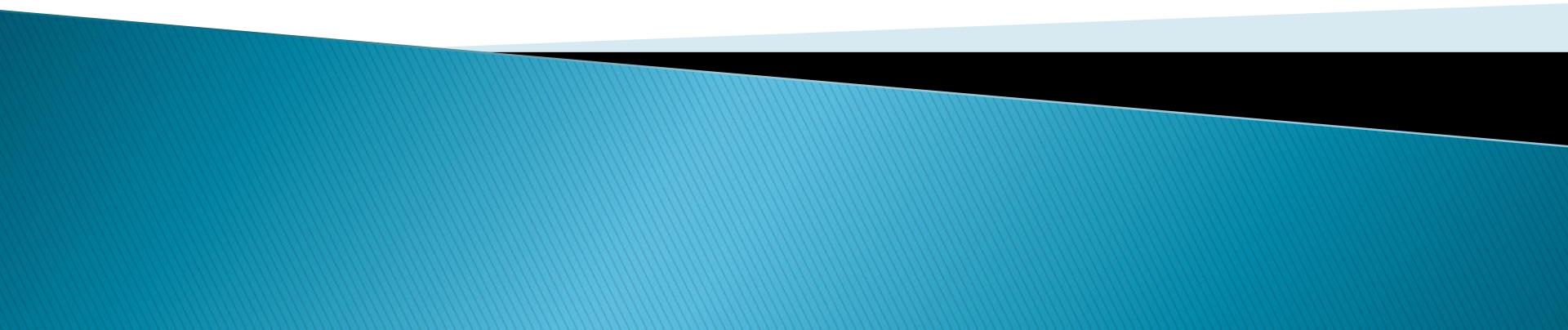
//scrie drum minim de la s la t un varf t dat folosind tata

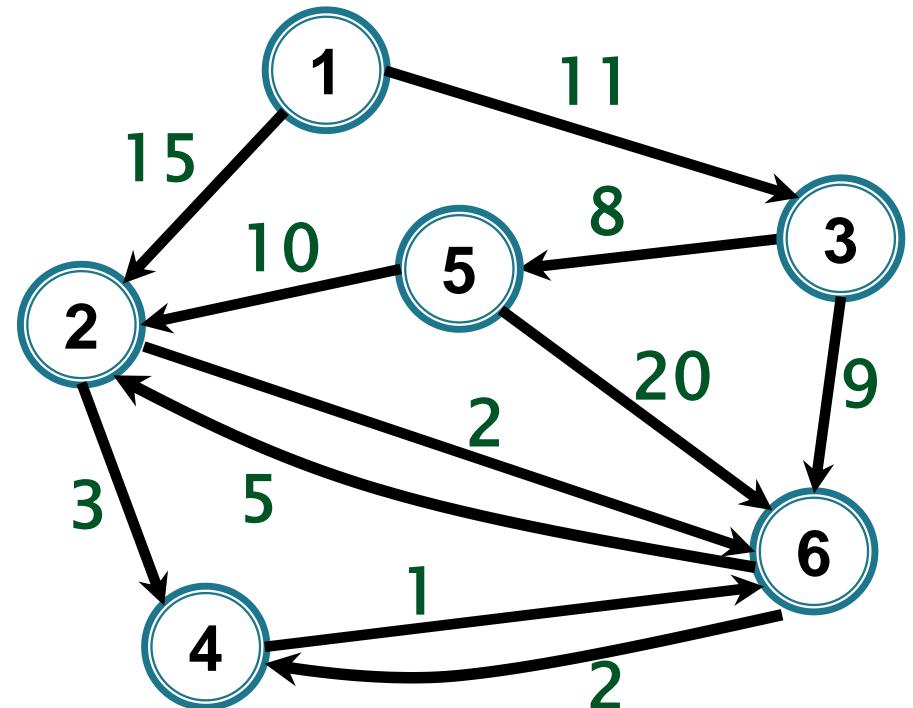
Algoritmul lui Dijkstra

▶ Observație

Vom demonstra că atunci când u este extras din Q eticheta lui $d[u]$ este chiar cu $\delta(s,u)$ (este corectă) și nu se va mai actualiza $\Rightarrow v \in Q$

Exemplu





d/tata

1
[0/0,

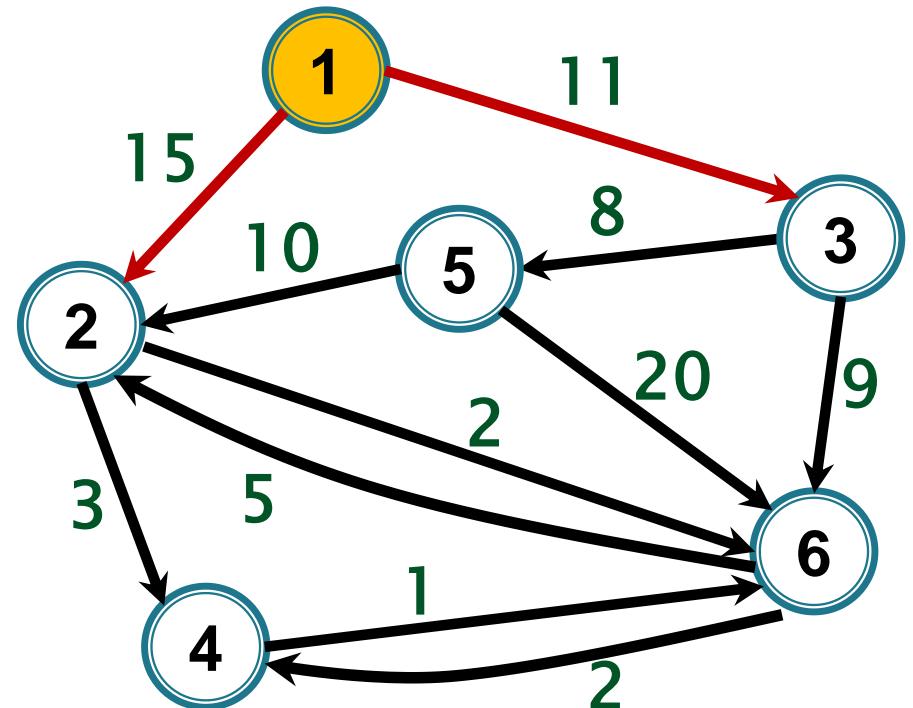
2
 $\infty/0,$

3
 $\infty/0,$

4
 $\infty/0,$

5
 $\infty/0,$

6
 $\infty/0]$



d/tata

[1
0/0,

2
 $\infty/0$,

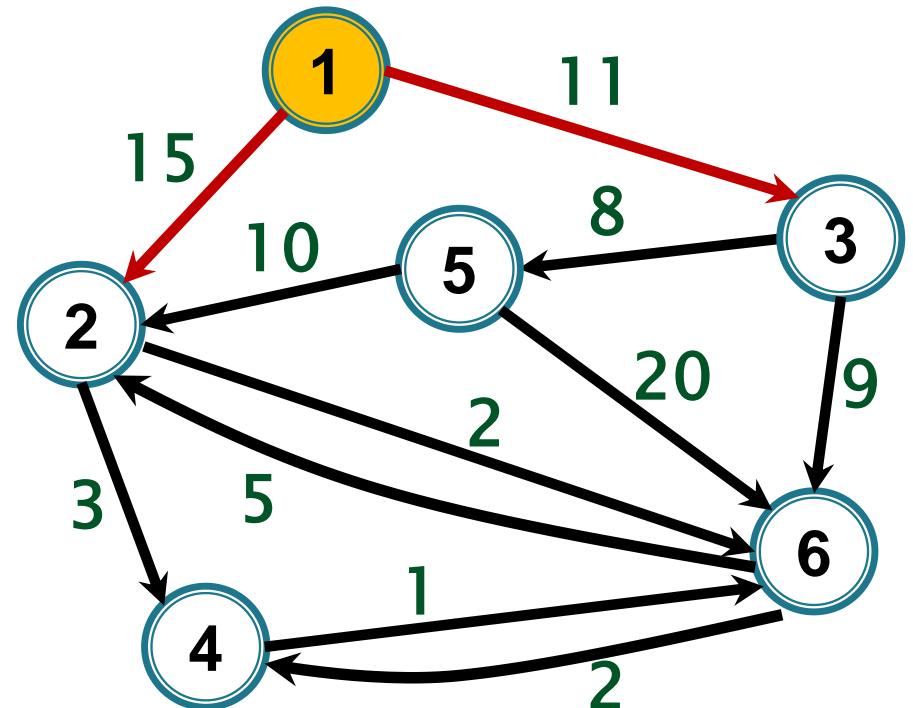
3
 $\infty/0$,

4
 $\infty/0$,

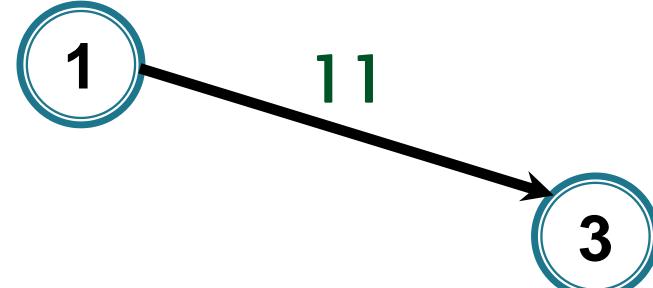
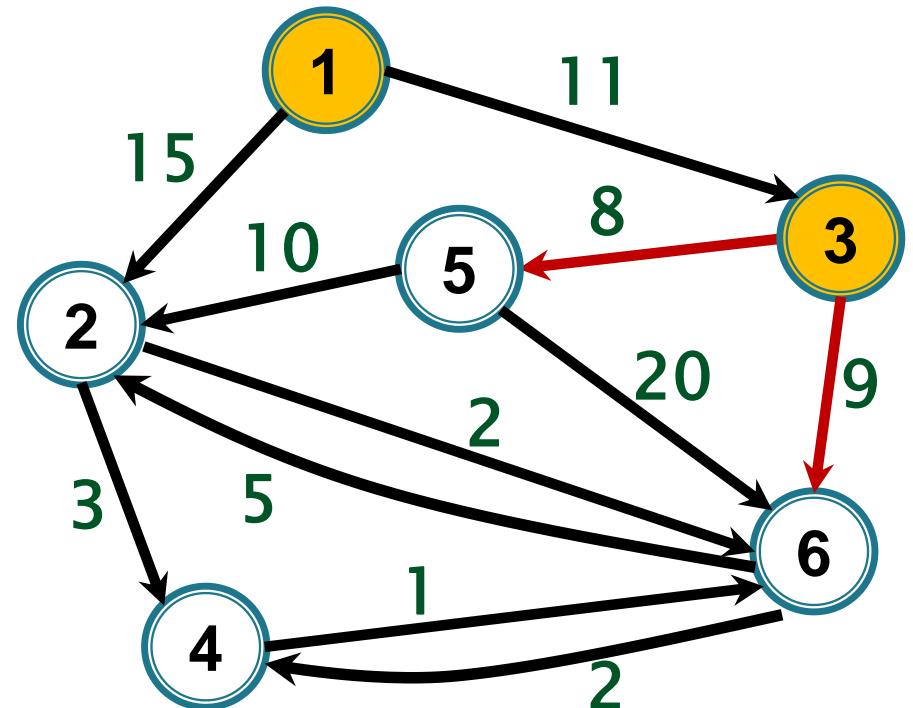
5
 $\infty/0$,

6
 $\infty/0$]

Sel. 1:



d/tata	1 [0/0 ,	2 ∞/0 ,	3 ∞/0 ,	4 ∞/0 ,	5 ∞/0 ,	6 ∞/0]
Sel. 1:	[- ,	15/1 ,	11/1 ,			



d/tata

[$0/0$, $1/1$]

$\infty/0$, $2/1$

$\infty/0$, $3/1$

$\infty/0$, $4/1$

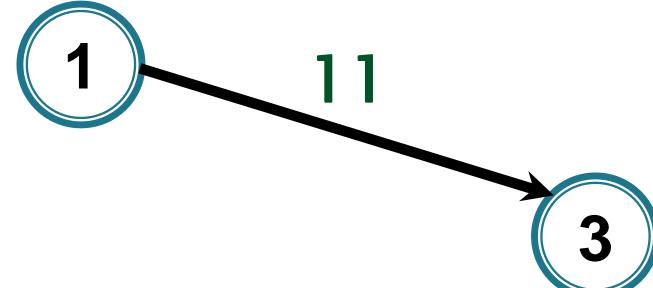
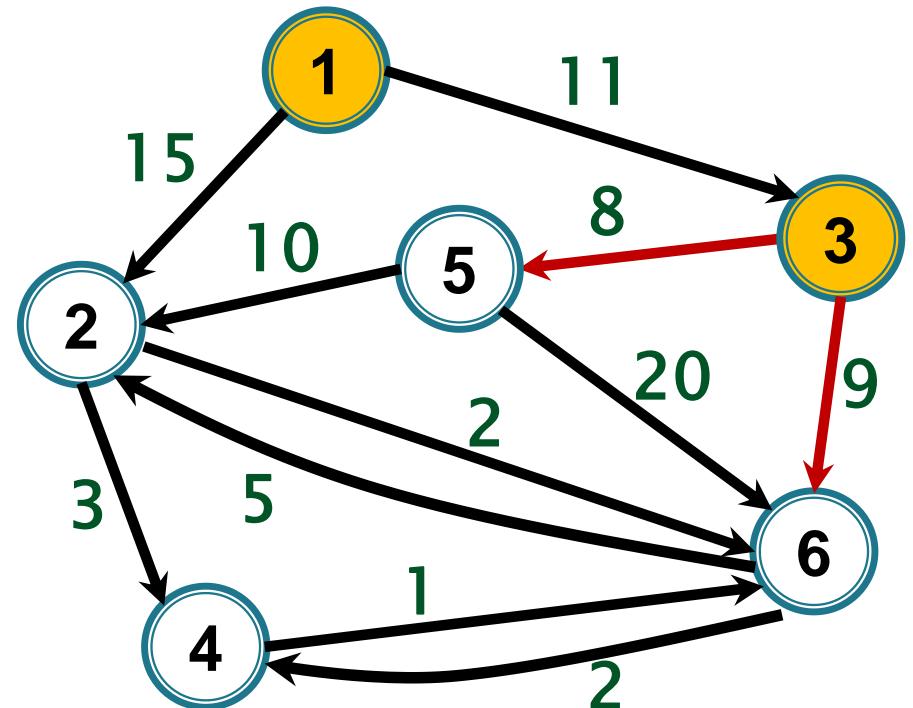
$\infty/0$, $5/1$

$\infty/0$, $6/1$

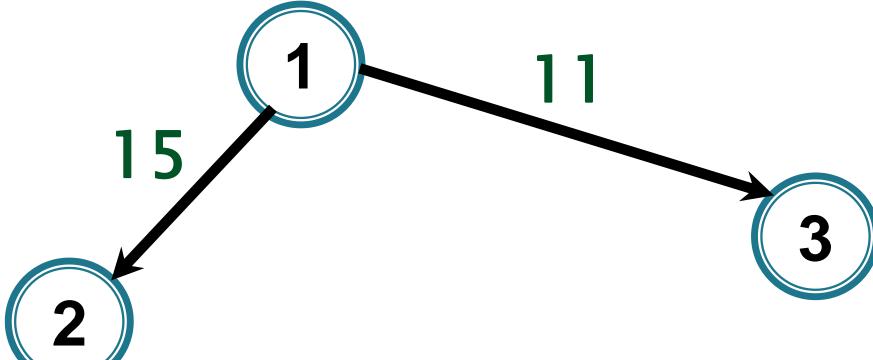
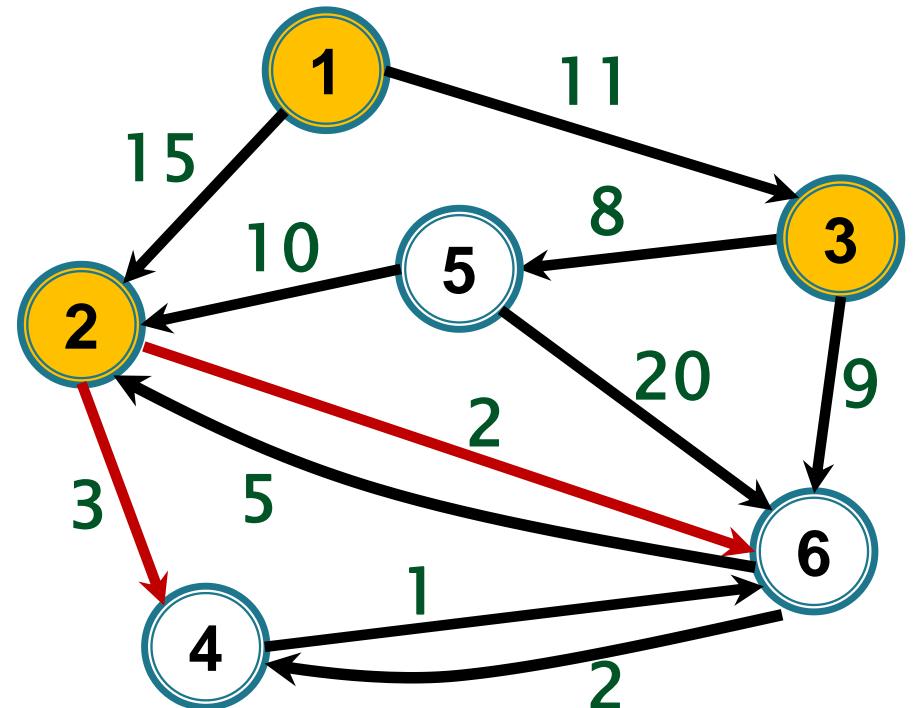
Sel. 1:

[$-$, $15/1$, $11/1$]

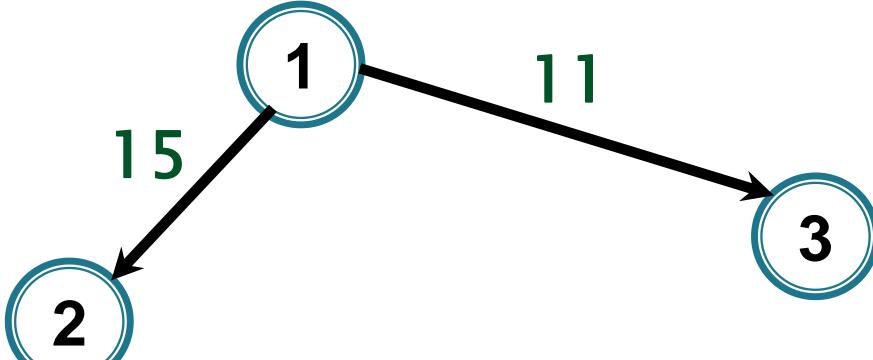
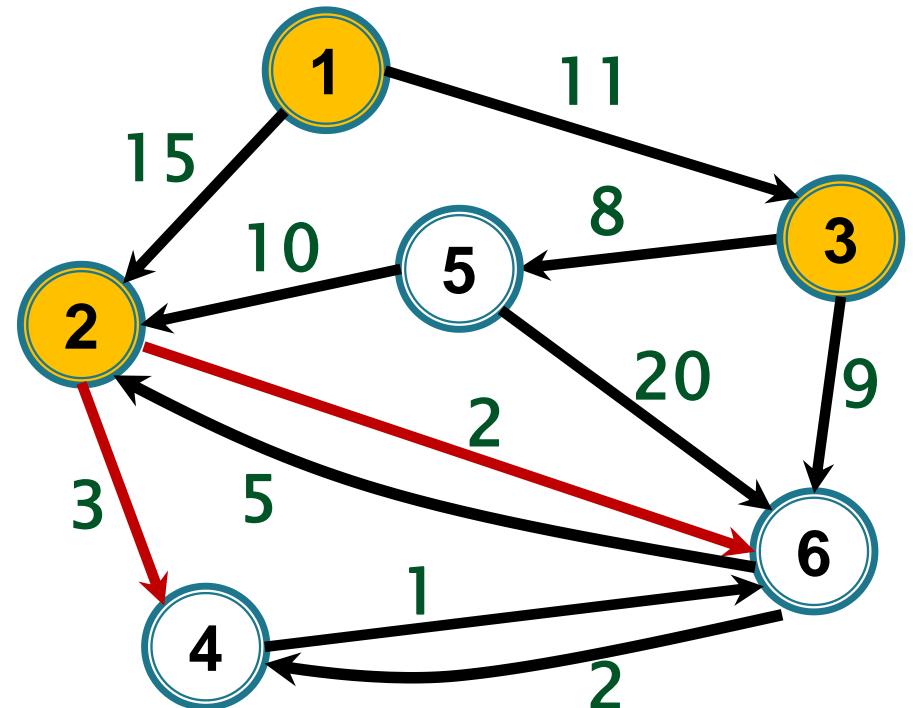
Sel. 3



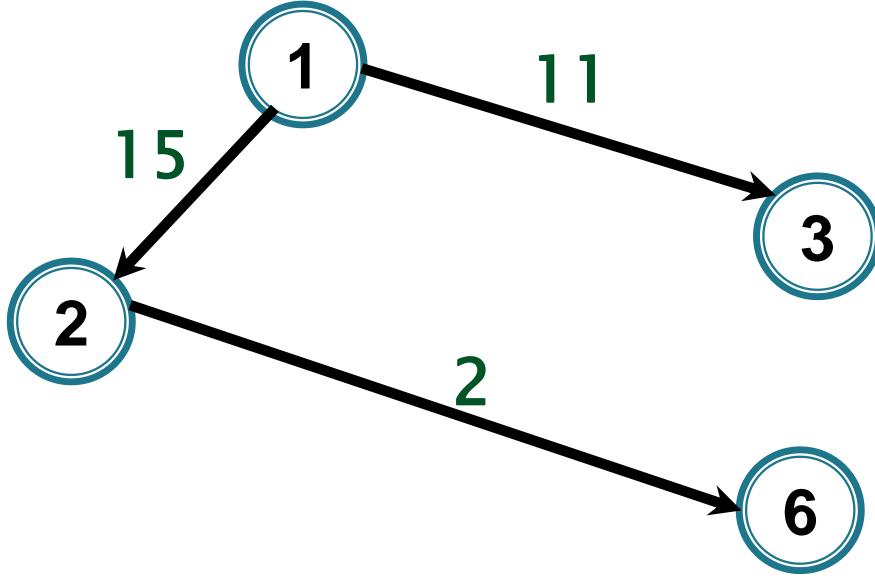
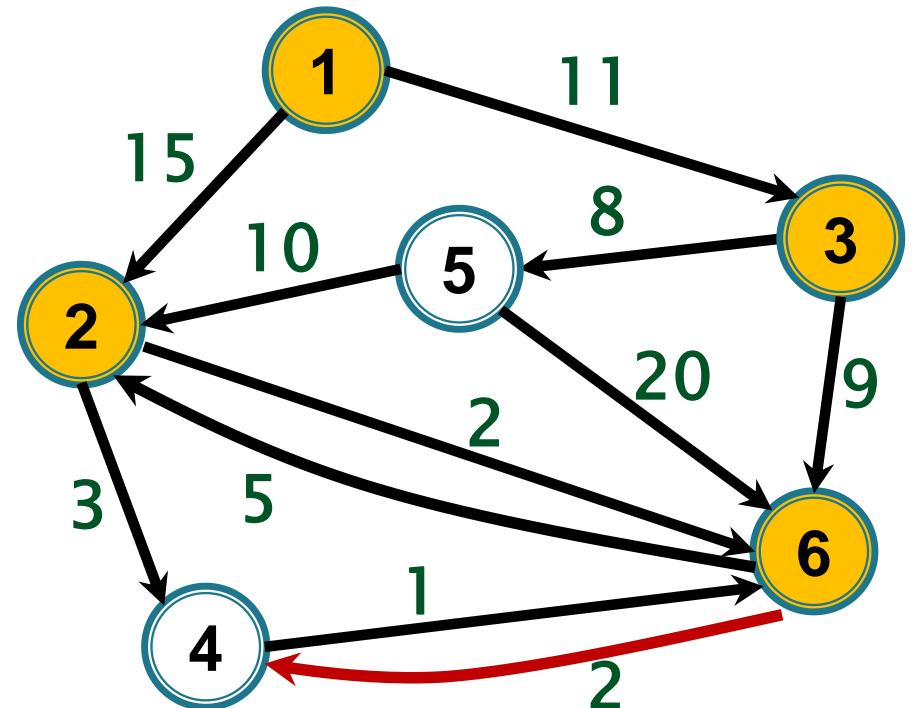
d/tata	1	2	3	4	5	6
	[0/0,	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 1:	[- ,	15/1,	11/1,	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 3:	[- ,	15/1,	- ,	$\infty/0,$	19/3,	20/3]



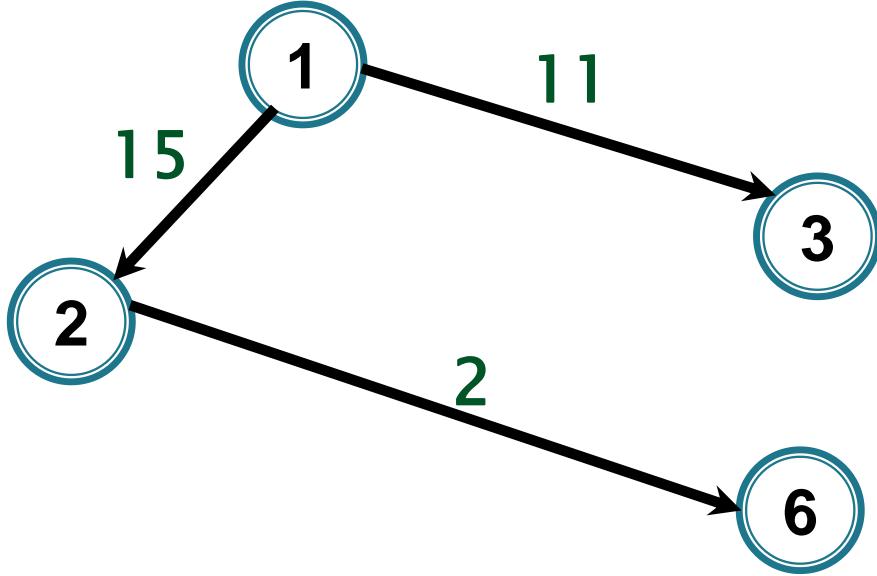
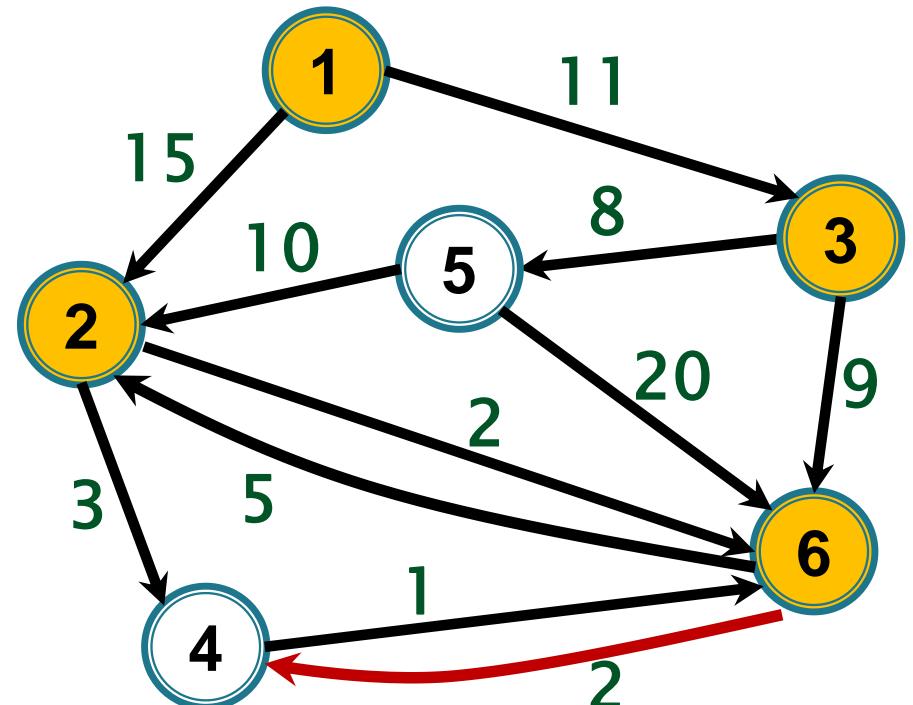
d/tata	1	2	3	4	5	6
	[0/0,	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 1:	[- ,	15/1,	11/1,	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 3:	[- ,	15/1,	- ,	$\infty/0,$	19/3,	20/3]
Sel. 2:						



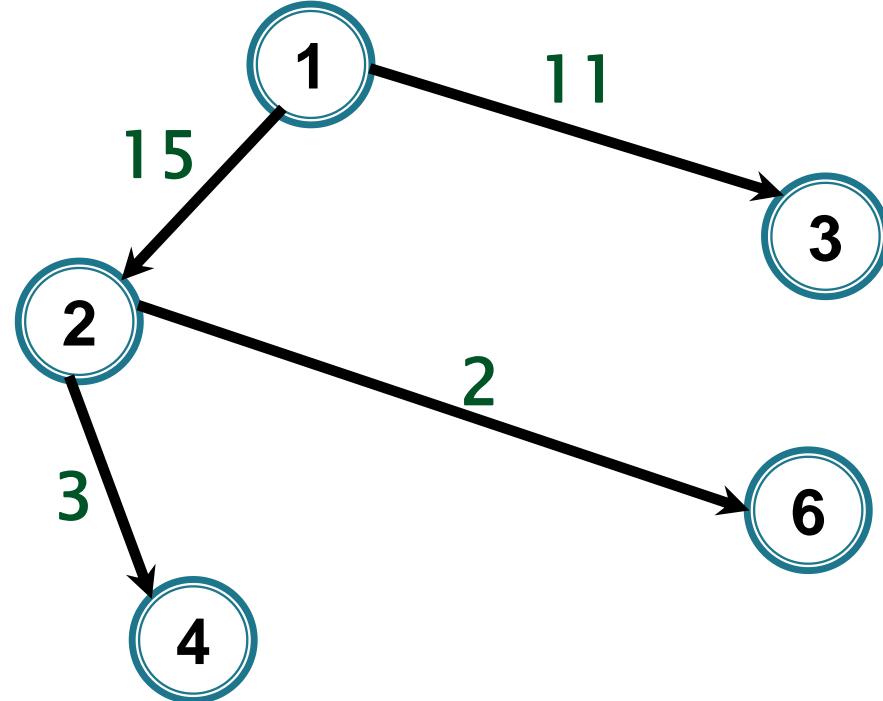
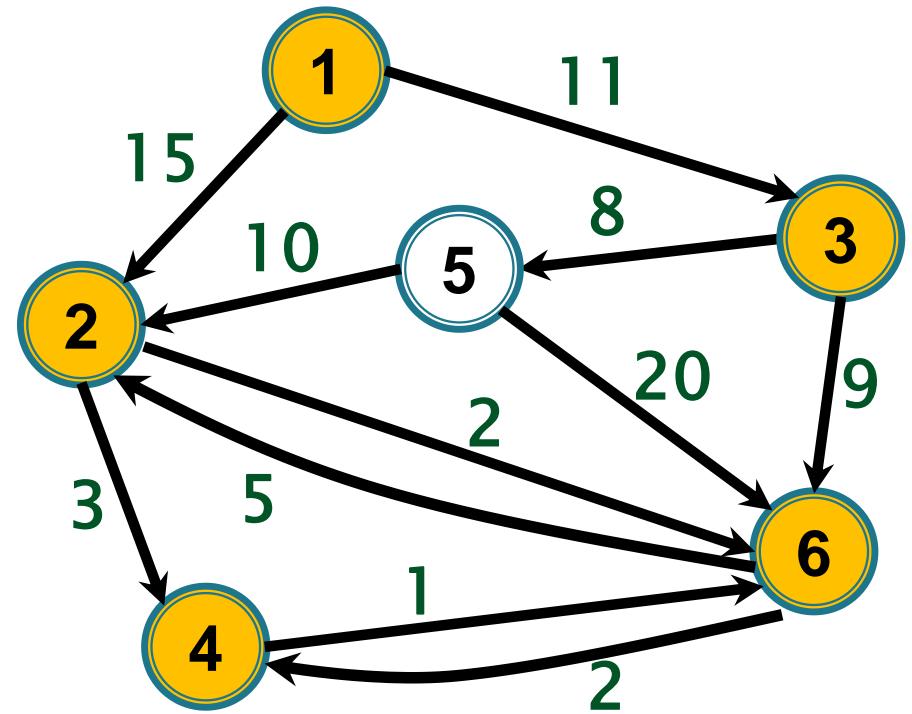
d/tata	1	2	3	4	5	6
	[0/0,	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 1:	[- ,	15/1,	11/1,	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 3:	[- ,	15/1,	- ,	$\infty/0,$	19/3,	20/3]
Sel. 2:	[- ,	- ,	- ,	18/2,	19/3,	17/2]



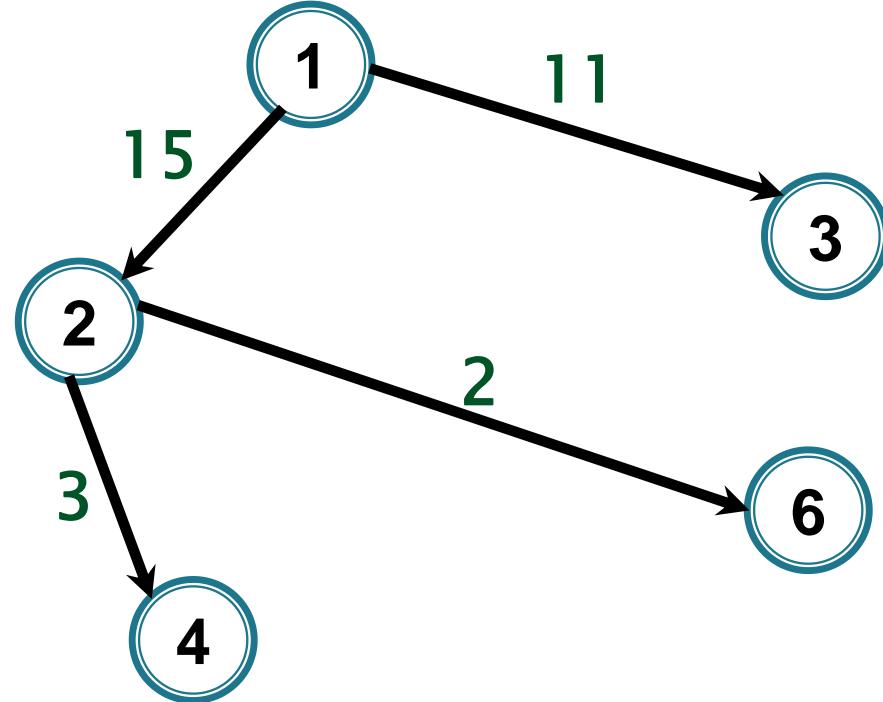
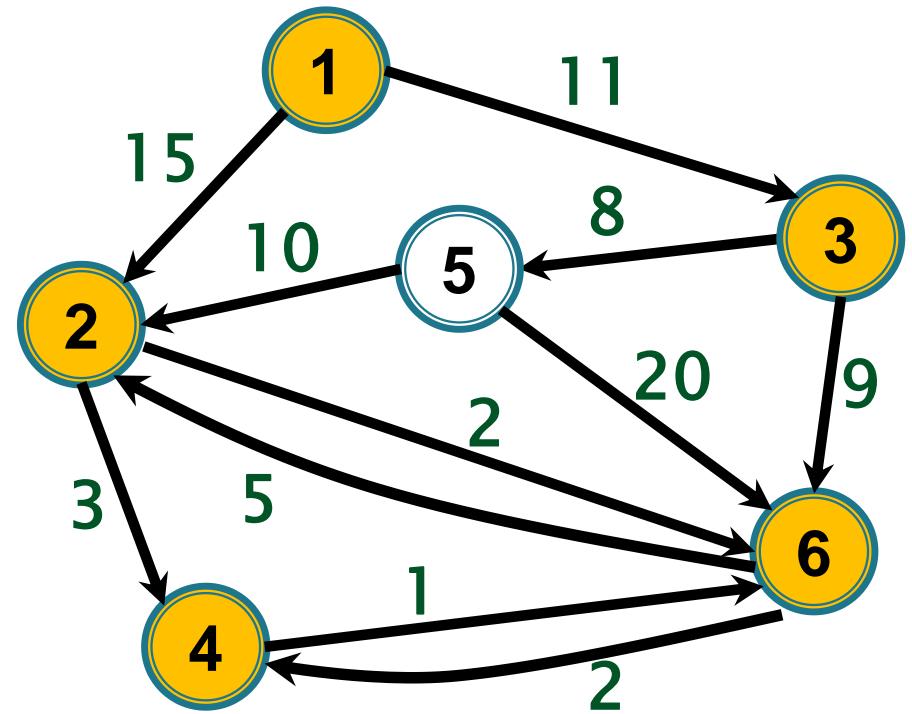
d/tata	1	2	3	4	5	6
	[0/0,	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 1:	[- ,	15/1,	11/1,	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 3:	[- ,	15/1,	- ,	$\infty/0,$	19/3,	20/3]
Sel. 2:	[- ,	- ,	- ,	18/2,	19/3,	17/2]
Sel. 6:						



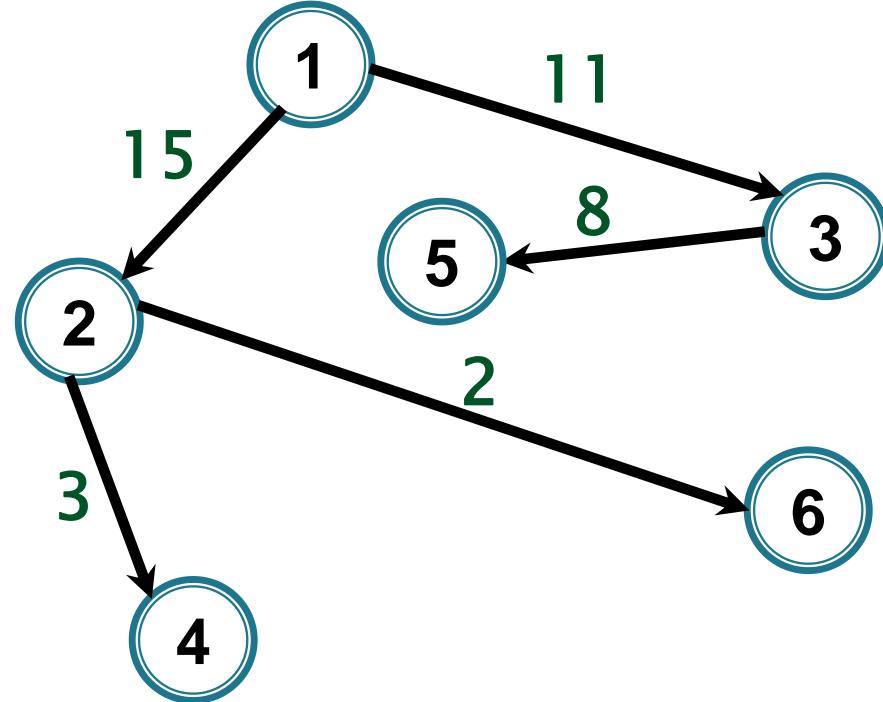
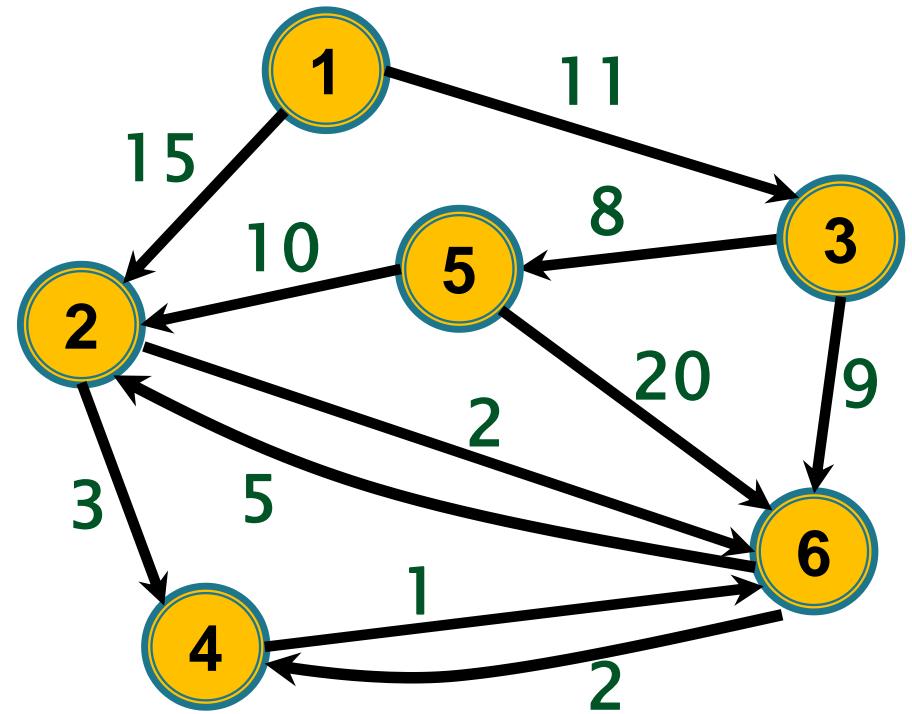
d/tata	1	2	3	4	5	6
	[0/0,	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 1:	[- ,	15/1,	11/1,	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 3:	[- ,	15/1,	- ,	$\infty/0,$	19/3,	20/3]
Sel. 2:	[- ,	- ,	- ,	18/2,	19/3,	17/2]
Sel. 6:	[- ,	- ,	- ,	18/2,	19/3,	-]



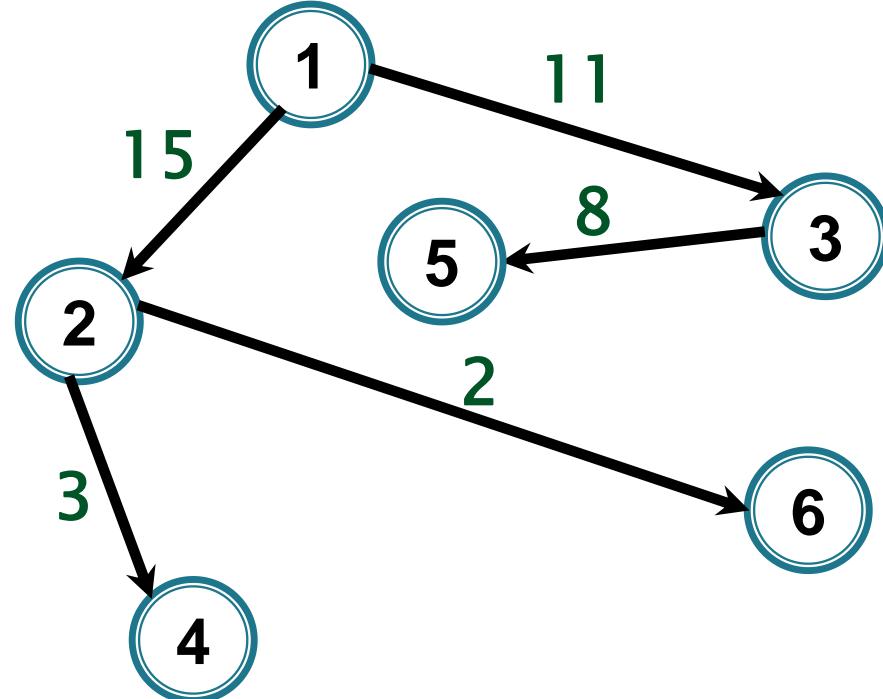
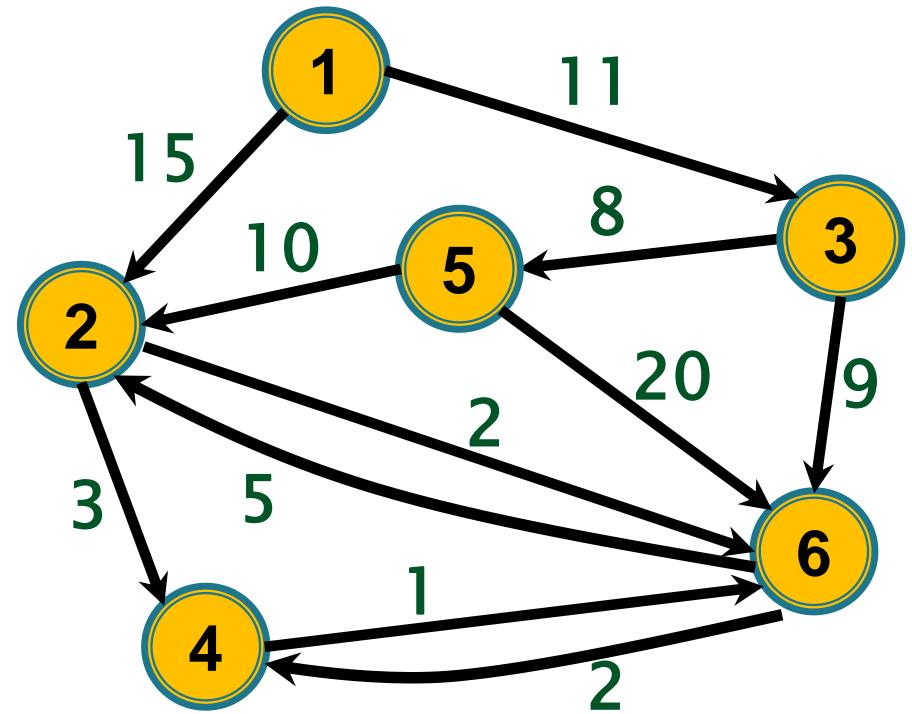
d/tata	1	2	3	4	5	6
	[0/0,	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 1:	[- ,	15/1,	11/1,	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 3:	[- ,	15/1,	- ,	$\infty/0,$	19/3,	20/3]
Sel. 2:	[- ,	- ,	- ,	18/2,	19/3,	17/2]
Sel. 6:	[- ,	- ,	- ,	18/2,	19/3,	-]
Sel. 4:						



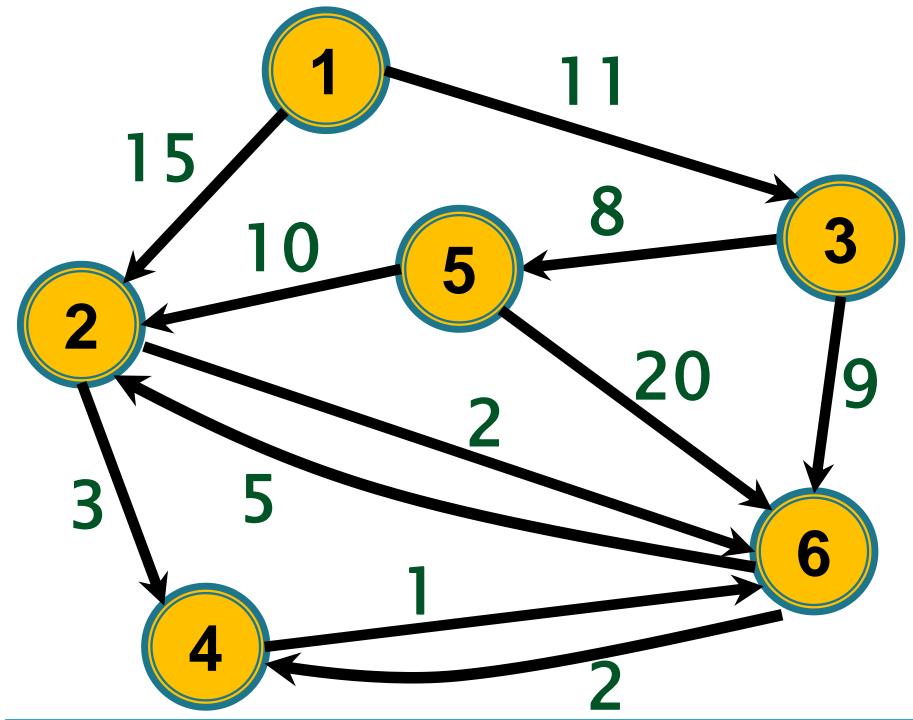
d/tata	1	2	3	4	5	6
	[0/0,	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 1:	[- ,	$15/1,$	$11/1,$	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 3:	[- ,	$15/1,$	- ,	$\infty/0,$	$19/3,$	$20/3]$
Sel. 2:	[- ,	- ,	- ,	$18/2,$	$19/3,$	$17/2]$
Sel. 6:	[- ,	- ,	- ,	$18/2,$	$19/3,$	-]
Sel. 4:	[- ,	- ,	- ,	- ,	$19/3,$	-]



d/tata	1	2	3	4	5	6
	[0/0,	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 1:	[- ,	$15/1,$	$11/1,$	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 3:	[- ,	$15/1,$	- ,	$\infty/0,$	$19/3,$	$20/3]$
Sel. 2:	[- ,	- ,	- ,	$18/2,$	$19/3,$	$17/2]$
Sel. 6:	[- ,	- ,	- ,	$18/2,$	$19/3,$	-]
Sel. 4:	[- ,	- ,	- ,	- ,	$19/3,$	-]
Sel. 5:						



d/tata	1	2	3	4	5	6
	[0/0,	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 1:	[- ,	15/1,	11/1,	$\infty/0,$	$\infty/0,$	$\infty/0]$
Sel. 3:	[- ,	15/1,	- ,	$\infty/0,$	19/3,	20/3]
Sel. 2:	[- ,	- ,	- ,	18/2,	19/3,	17/2]
Sel. 6:	[- ,	- ,	- ,	18/2,	19/3,	-]
Sel. 4:	[- ,	- ,	- ,	- ,	19/3,	-]
Sel. 5:	[- ,	- ,	- ,	- ,	- ,	-]

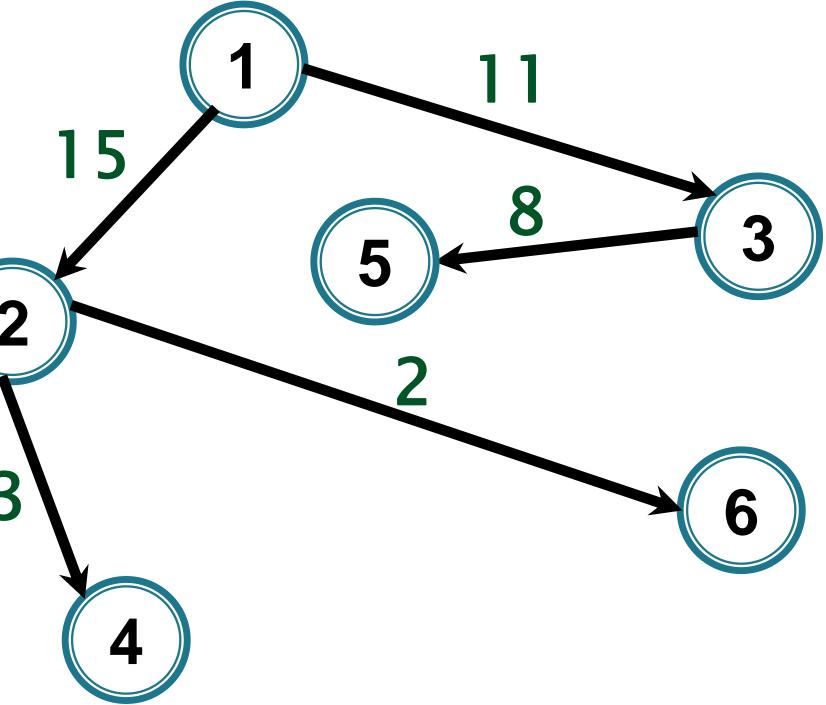


d/tata

1

2

3



4

5

6

Soluție: [0/0, 15/1, 11/1, 18/2, 19/3, 17/2]

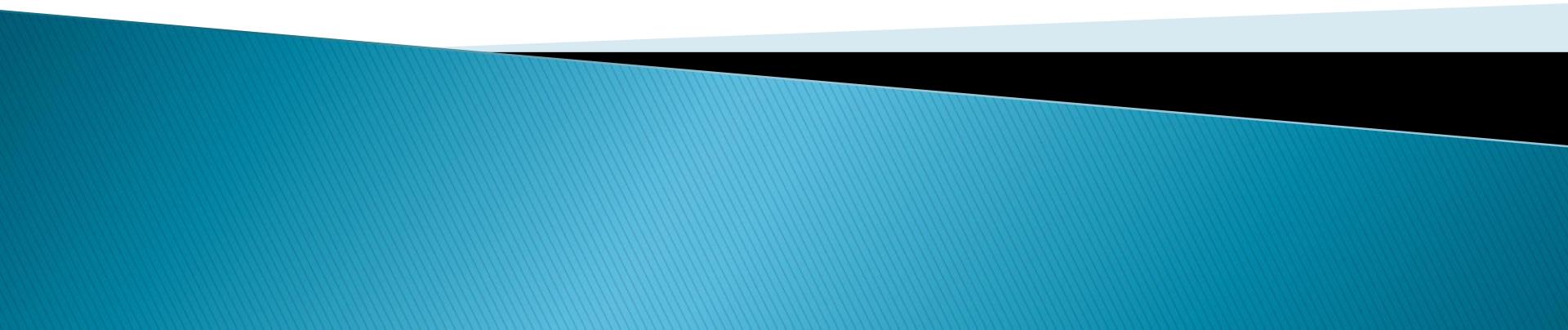
Un drum minim de la 1 la 6?

Algoritmul lui Dijkstra

▶ Observații.

1. Dacă vârful u curent are eticheta $d[u] = \infty$, algoritmul se poate opri
2. Vectorul $tata$ memorează arborele distanțelor față de s (vârfurile neaccesibile din s rămân cu tata 0)

Complexitate



Dijkstra(G, w, s)

initializează multimea vârfurilor neselectate Q cu V
pentru fiecare $u \in V$ executa

$d[u] = \infty$; tata[u]=0

$d[s] = 0$

cat timp $Q \neq \emptyset$ executa

$u = \text{extrage vârf cu eticheta } d \text{ minimă din } Q$

pentru fiecare $uv \in E$ executa

daca $d[u] + w(u, v) < d[v]$ atunci

$d[v] = d[u] + w(u, v)$

tata[v] = u

scrie d, tata

//scrie drum minim de la s la t un varf t dat folosind tata

Algoritmul lui Dijkstra



Cum memorăm $Q =$ vârfurile încă neselectate?

Algoritmul lui Dijkstra

Q poate fi (ca și în cazul algoritmului lui Prim)

▶ **vector:**

$Q[u] = 1$, dacă u este selectat ($u \notin Q$)

0, altfel ($u \in Q$)

▶ **min-ansamblu (heap)**

Algoritmul lui Dijkstra

Complexitate - reprezentarea lui Q ca vector

$$Q[u] = \begin{cases} 1, & \text{dacă } u \text{ este selectat în } V(T) \\ 0, & \text{altfel (} u \in Q \text{)} \end{cases}$$

- ▶ Inițializare Q -> O(n)
 - ▶ $n * \text{extragere vârf minim}$ -> O(n^2)
 - ▶ actualizare etichete vecini -> O(m)
-
- $O(n^2)$

Algoritmul lui Dijkstra

Complexitate – Q min-heap

- ▶ Inițializare Q →
 - ▶ $n * \text{extragere vârf minim}$ →
 - ▶ actualizare etichete vecini →
-

Dijkstra(G, w, s) - Q min-heap in raport cu d

pentru fiecare $u \in V$ executa

$d[u] = \infty$; $tata[u] = 0$

$d[s] = 0$

$Q = V$ //creare heap cu cheile din d

cat timp $Q \neq \emptyset$ executa

$u = \text{extrage_min}(Q)$

pentru fiecare $uv \in E$ executa

daca $d[u] + w(u, v) < d[v]$ atunci

$d[v] = d[u] + w(u, v)$

repara(Q, v)

$tata[v] = u$

scrie d, tata

//scrie drum minim de la s la t un varf t dat folosind tata

Algoritmul lui Dijkstra

Complexitate – Q min-heap

- ▶ Inițializare Q → $O(n)$
 - ▶ $n * \text{extragere vârf minim}$ → $O(n \log n)$
 - ▶ actualizare etichete vecini → $O(m \log n)$
-
- !!+ actualizare Q**

$O(m \log n)$

Algoritmul lui Dijkstra

- ▶ **Observație.** Pentru a determina drumul minim între două vârfuri s și t **date** putem folosi algoritmul lui Dijkstra cu următoarea modificare:
 - dacă vârful u ales este chiar t, **algoritmul se oprește**;
 - drumul de la s la t se afișează folosind vectorul tata (vezi BF)

Algoritmul lui Dijkstra

- ▶ Dijkstra \approx Prim (versiunea $O(n^2)$ / $O(m \log n)$)

Algoritmul lui Dijkstra



Algoritmul funcționează și pentru grafuri neorientate?

Algoritmul lui Dijkstra

- ▶ De ce nu funcționează corect algoritmul dacă avem arce cu cost negativ + exemplu?
- ▶ Cum putem rezolva problema dacă avem și arce de cost negativ?
 - Putem aduna o constantă la costul fiecărui arc astfel încât toate arcele să aibă cost pozitiv. Drumul minim între 2 vârfuri rămâne la fel? – **NU**



Algoritmul lui Dijkstra

- ▶ Cum putem rezolva problema dacă avem și arce de cost negativ?



Algoritmul BELLMAN – FORD (suplimentar)

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

pentru $i = 1, n-1$ execută

pentru fiecare $uv \in E$ execută

daca $d[u] + w(u, v) < d[v]$ atunci

$d[v] = d[u] + w(u, v)$

$tata[v] = u$

Corectitudinea Algoritmului lui Dijkstra

Corectitudine

- ▶ **Lema 1.** Pentru orice $u \in V$, la orice pas al algoritmului lui Dijkstra 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:
 $tata[u] = \text{predecesorul lui } u \text{ pe un drum de la } s \text{ la } u \text{ de cost } d[u]$
 - b) $d[u] \geq \delta(s, u)$
- ▶ **Consecință.** Dacă la un pas al algoritmului avem pentru un vîrf u relația $d[u] = \delta(s, u)$, atunci $d[u]$ nu se mai modifică până la final.

Corectitudine

▶ Propoziție.

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

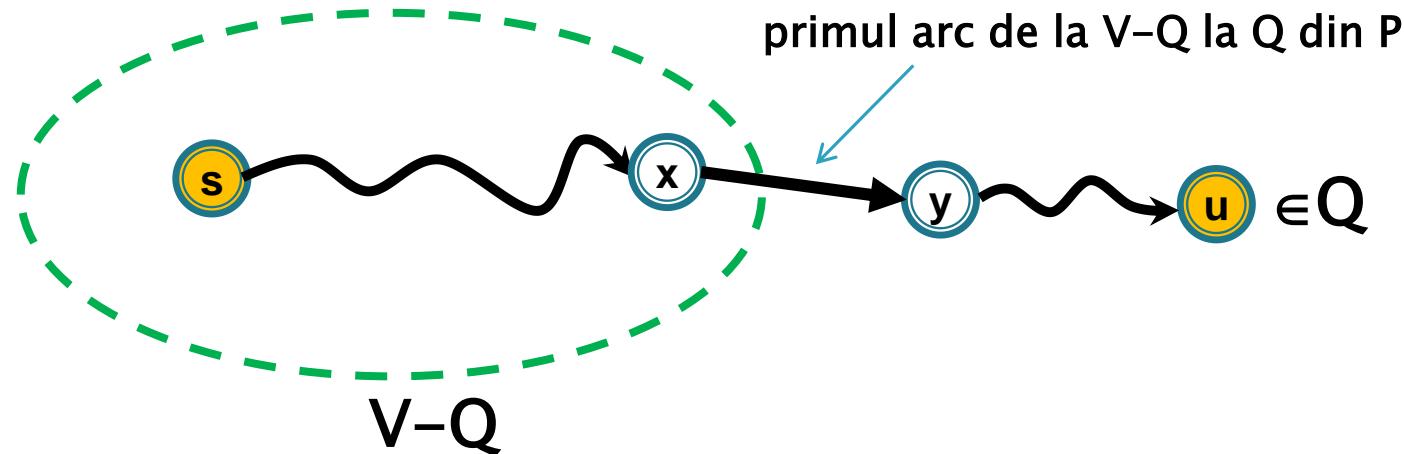
$w : E \rightarrow \mathbb{R}_+^*$, și $s \in V$ fixat.

La finalul algoritmului lui Dijkstra avem:

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

Corectitudine

- ▶ Demonstrație (idee). Inducție: $d[x] = \delta(s, x) \quad \forall x \notin Q$ (=deja selectat)
- ▶ Când un vârf u este selectat: fie P un $s-u$ drum minim



din modul în care este ales u

$$d[u] \leq d[y] \leq d[x] + w(x, y) = \delta(s, x) + w(x, y) = w(s - y) = \delta(s, y)$$

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

dupa relaxarea lui xy (mai mult, are loc chiar egalitate: $d[y] = \delta(s, x) + w(x, y) = w(s - y) = \delta(s, y)$)

$$w(s - y) = \delta(s, y)$$

ipoteza de inducție pentru x

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