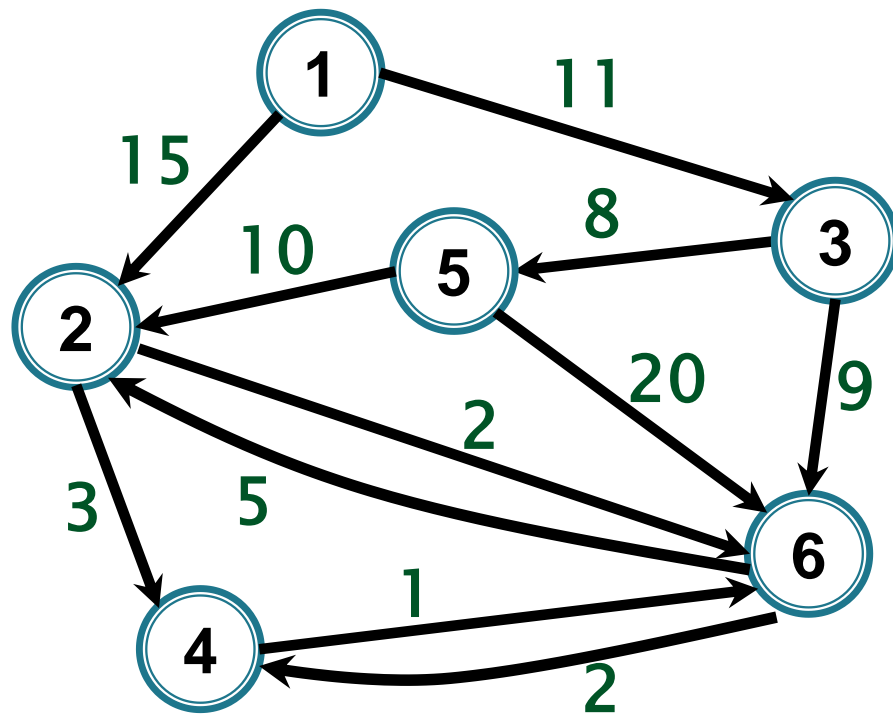


# Algoritmul lui Dijkstra

## ► Ipoteză:

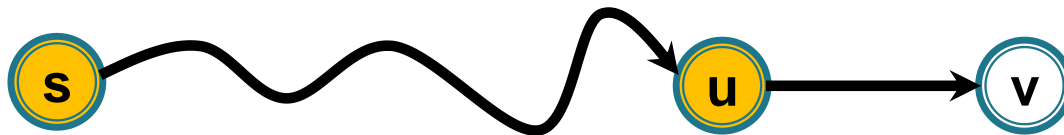
Presupunem că arcele au cost pozitiv  
(graful poate conține circuite)



# Algoritmul lui Dijkstra

Idee: La un pas este ales ca vârf curent (vizitat) 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 (vizitat) 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

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



# Algoritmul lui Dijkstra

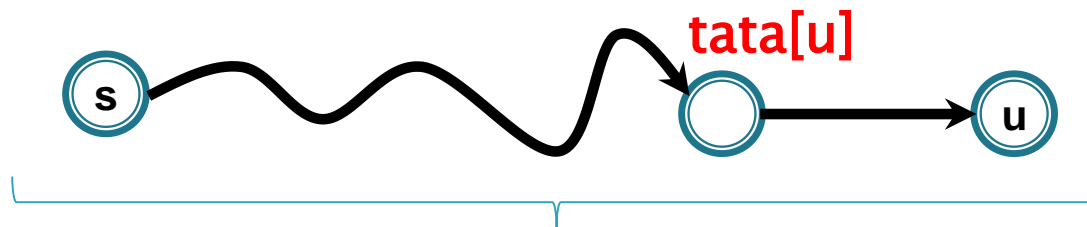
- generalizare a ideii de parcurgere BF
- dacă toate arcele au cost egal Dijkstra  $\equiv$  BF

# Pseudocod

# Algoritmul lui Dijkstra

- ▶ Reținem pentru fiecare vârf etichetele

- $d[u]$
- $tata[u]$



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



# Algoritmul lui Dijkstra

## ▶ La un pas

- este selectat un vârf  $u$  (neselectat) care “pare” cel mai apropiat de  $s \Leftrightarrow$  are eticheta  $d$  minimă

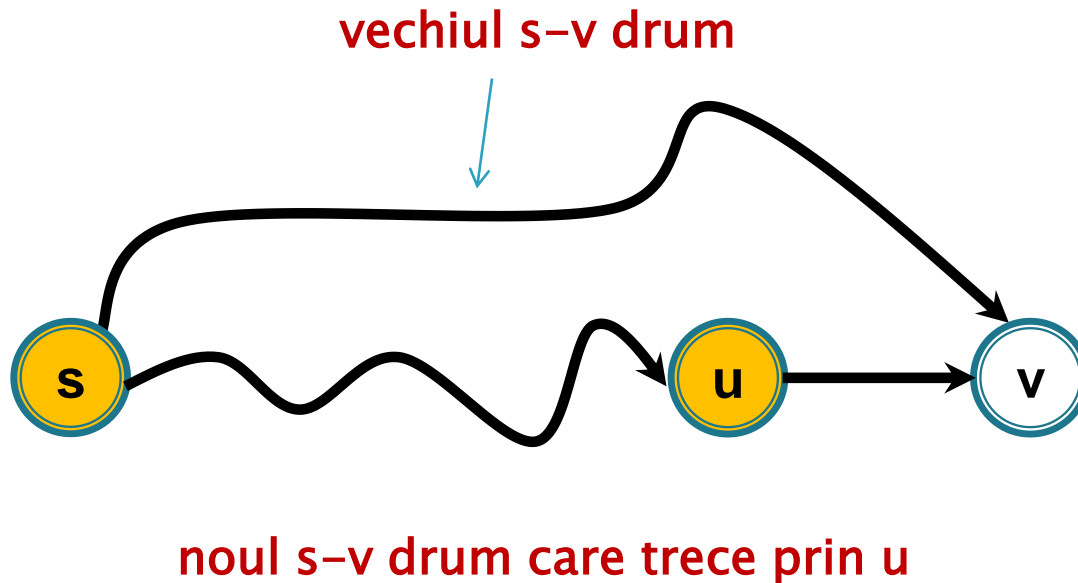
# Algoritmul lui Dijkstra

## ► 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  $u$  – extinzând drumul minim deja găsit de la  $s$  la  $u$  cu arcul  $uv$ 
  - tehnica de relaxare a arcelor care ies din  $u$

# Algoritmul lui Dijkstra

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



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

Dijkstra( $G, w, s$ )

**inițializează** mulțimea vârfurilor neselectate  $Q$  cu  $V$

Dijkstra( $G, w, s$ )

inițializează mulțimea vârfurilor neselectate  $Q$  cu  $V$

pentru fiecare  $u \in V$  executa

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

Dijkstra( $G, w, s$ )

inițializează mulțimea vârfurilor neselectate  $Q$  cu  $V$

pentru fiecare  $u \in V$  executa

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

$d[s] = 0$

Dijkstra( $G, w, s$ )

inițializează mulțimea 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 // pentru  $i=1, n$  executa

Dijkstra( $G, w, s$ )

inițializează mulțimea 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

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



# Dijkstra( $G, w, s$ )

inițializează mulțimea 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( $G, w, s$ )

inițializează mulțimea 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  ~~$v \in Q$~~  și  $d[u] + w(u, v) < d[v]$  atunci

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

$tata[v] = u$

## Dijkstra( $G, w, s$ )

inițializează mulțimea 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  $d, tata$

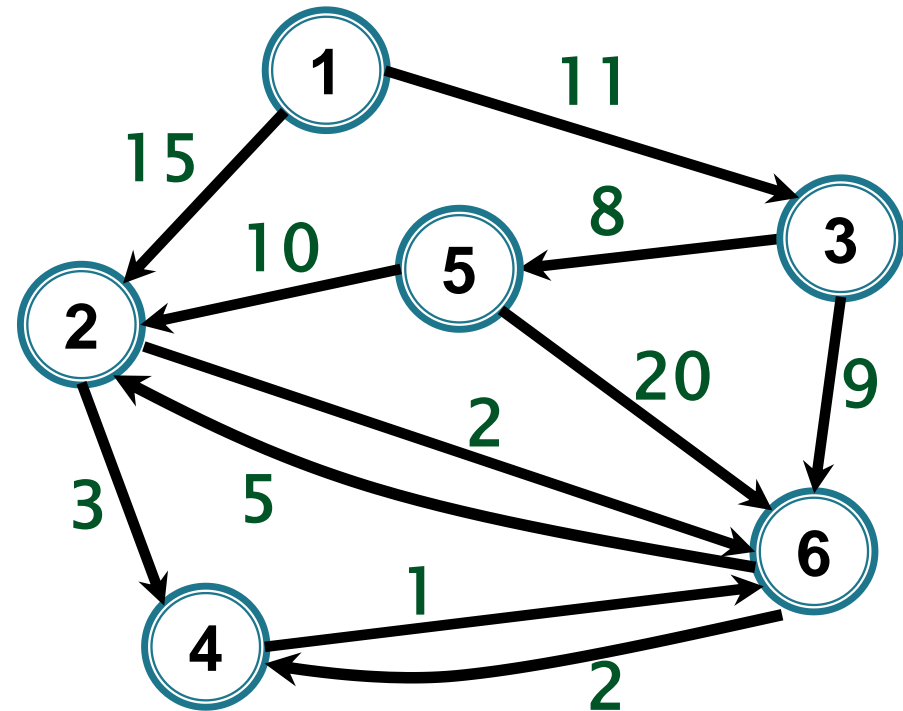
//scrie drum minim de la  $s$  la  $t$  un varf  $t$  dat folosind  $tata$

# 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**  $\Rightarrow \nexists v \in Q$

# Exemplu



d/tata

**1**  
[ 0/0,

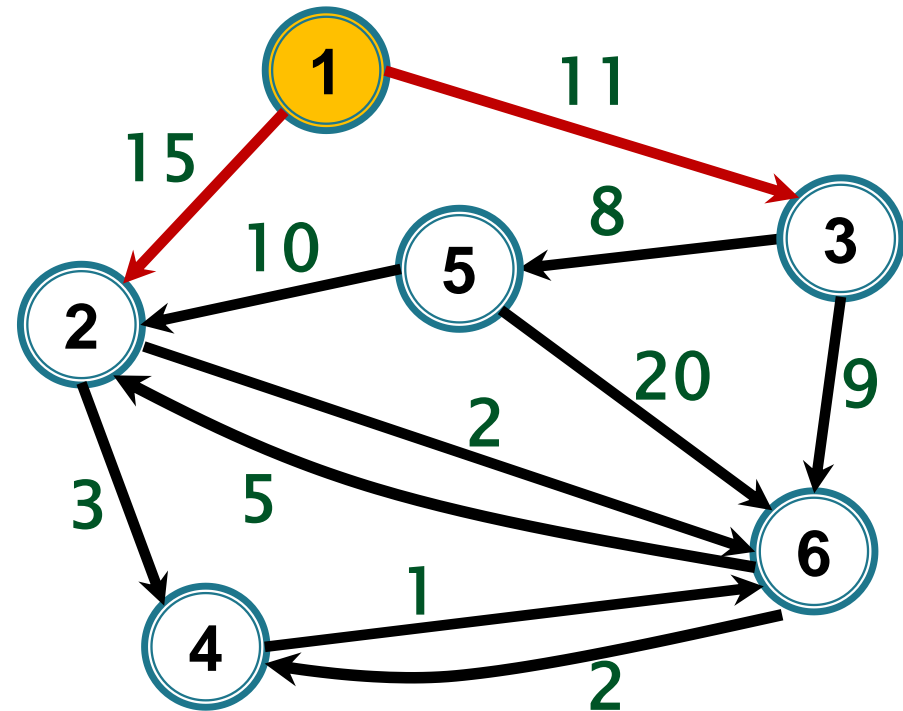
**2**  
 $\infty$ /0,

**3**  
 $\infty$ /0,

**4**  
 $\infty$ /0,

**5**  
 $\infty$ /0,

**6**  
 $\infty$ /0 ]



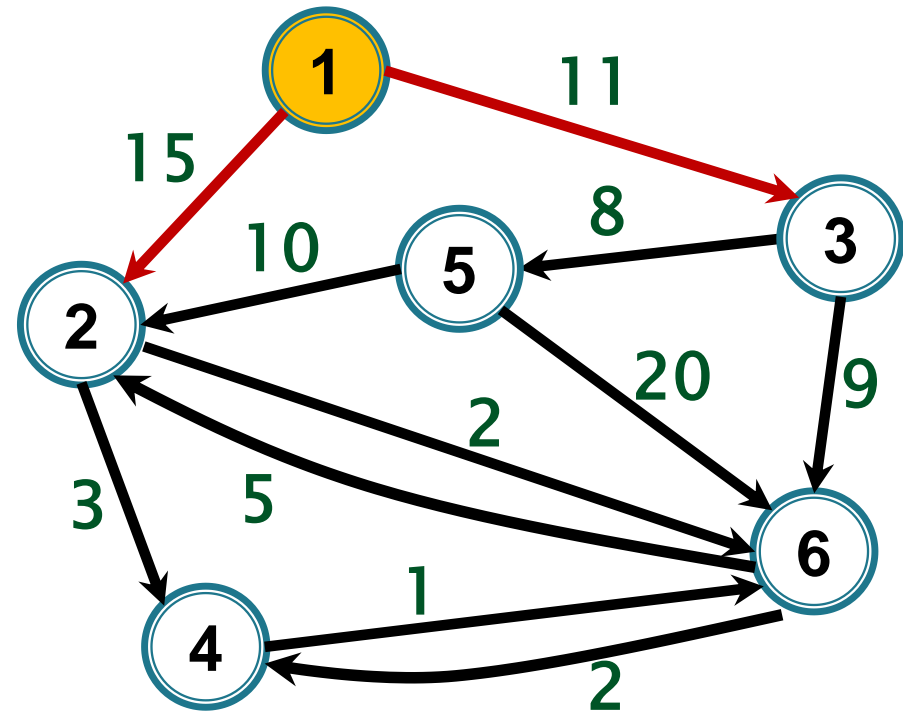
1

d/tata

	1	2	3	4	5	6
	[ 0/0,	$\infty$ /0,	$\infty$ /0,	$\infty$ /0,	$\infty$ /0,	$\infty$ /0 ]

Sel. 1:

$$d[v] = \min\{d[v], d[u] + w(u, v)\}$$



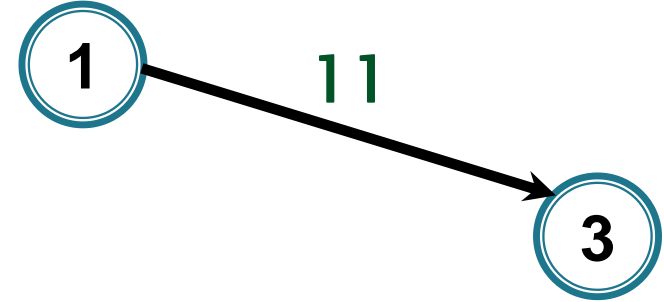
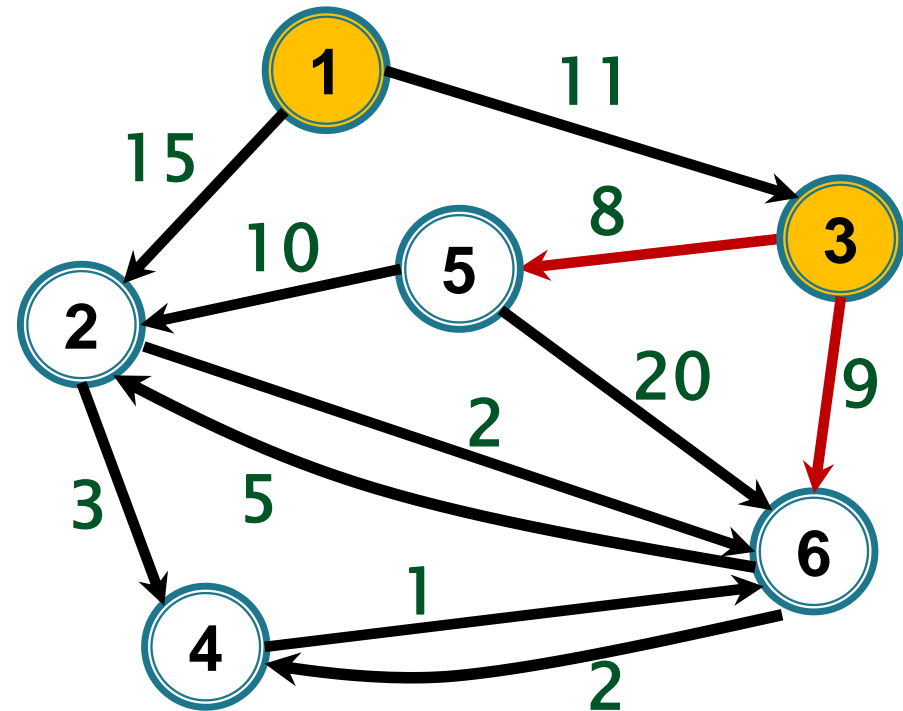
1

d/tata

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

$$d[v] = \min\{d[v], d[u] + w(u, v)\}$$





d/tata

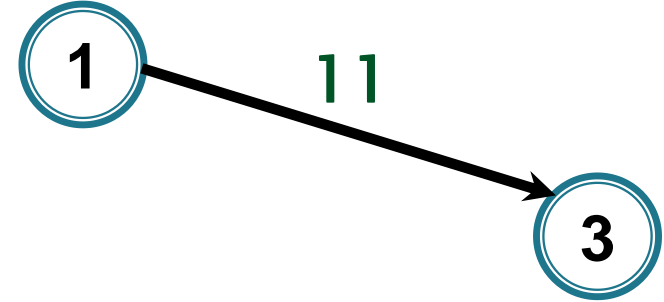
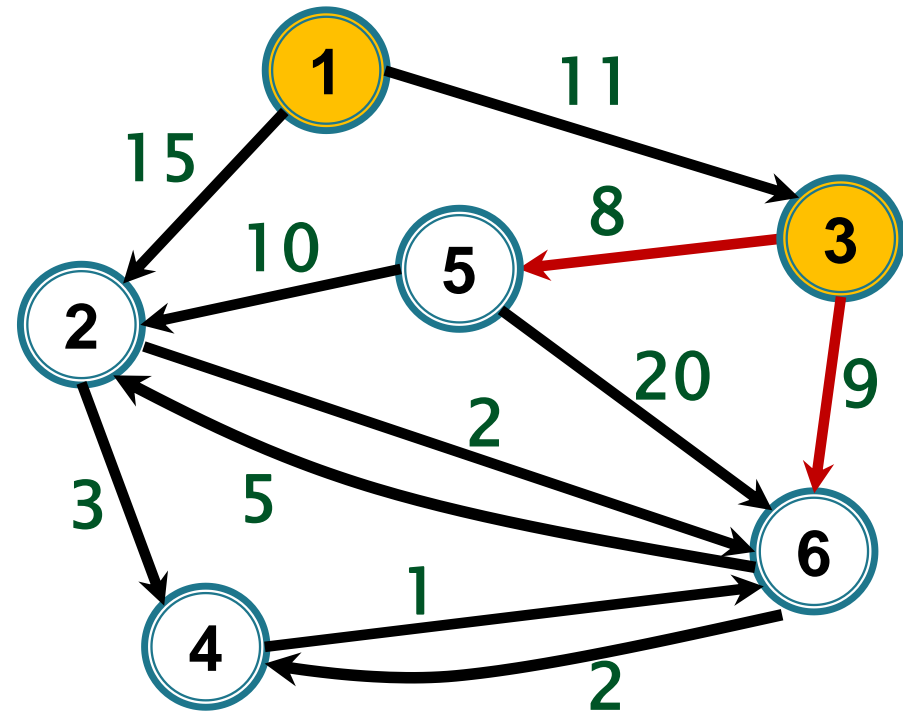
	1	2	3	4	5	6
d/tata	[ 0/0,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ]

Sel. 1: [ - , 15/1, 11/1,

Sel. 3

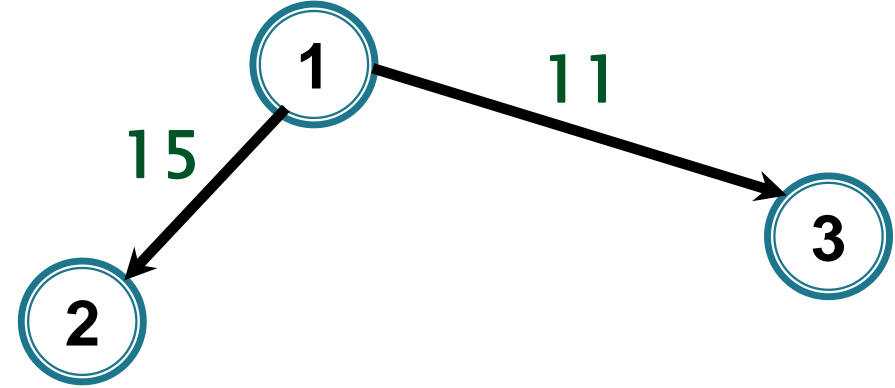
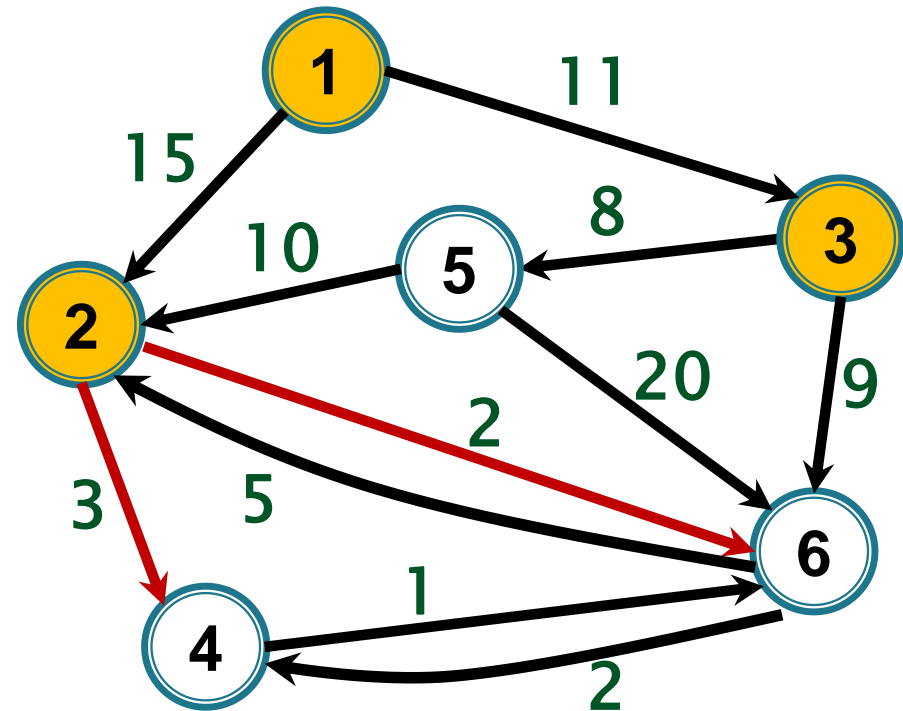
$$d[5] = \min\{d[5], d[3] + w(3, 5)\}$$

$$d[v] = \min\{d[v], d[u] + w(u, v)\}$$



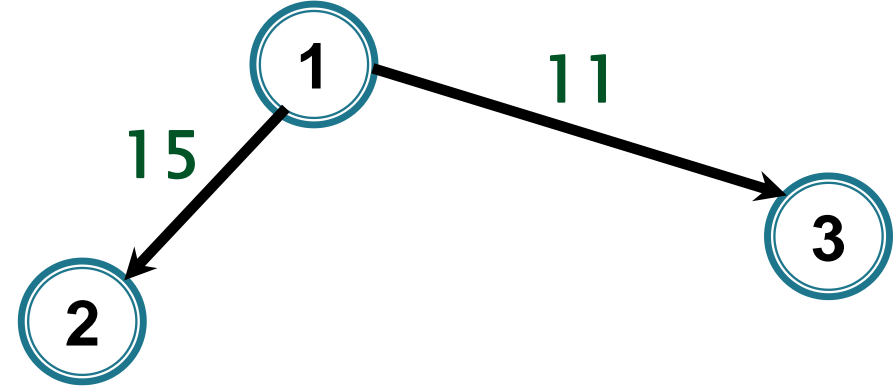
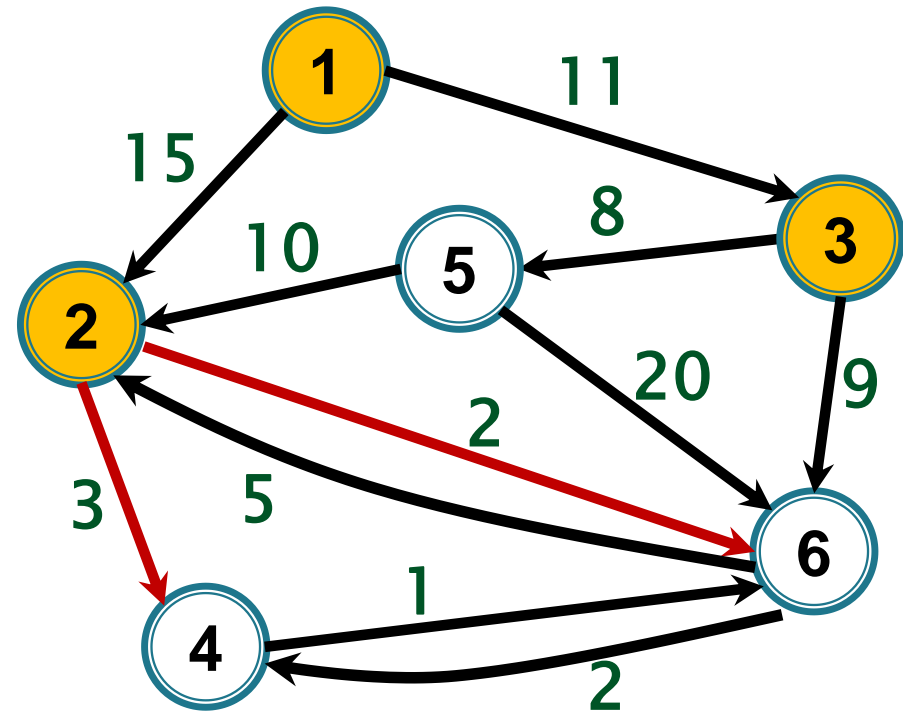
d/tata	1	2	3	4	5	6
	[ 0/0, ]	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0,$	$\infty/0$ ]
Sel. 1:	[ - ,	15/1,	11/1,	$\infty/0,$	$\infty/0,$	$\infty/0$ ]
Sel. 3:	[ - ,	15/1,	- ,	$\infty/0,$	19/3,	20/3 ]

$$d[v] = \min\{d[v], d[u] + w(u, v)\}$$



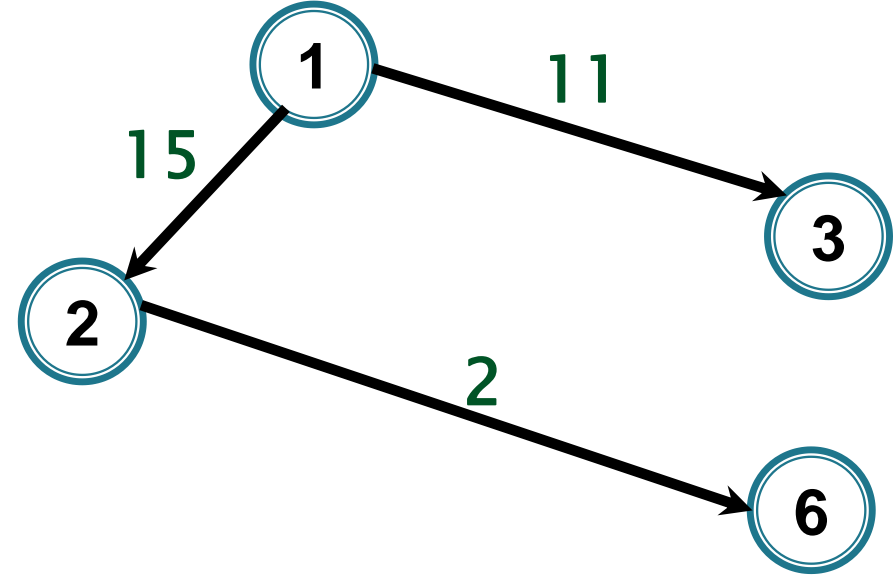
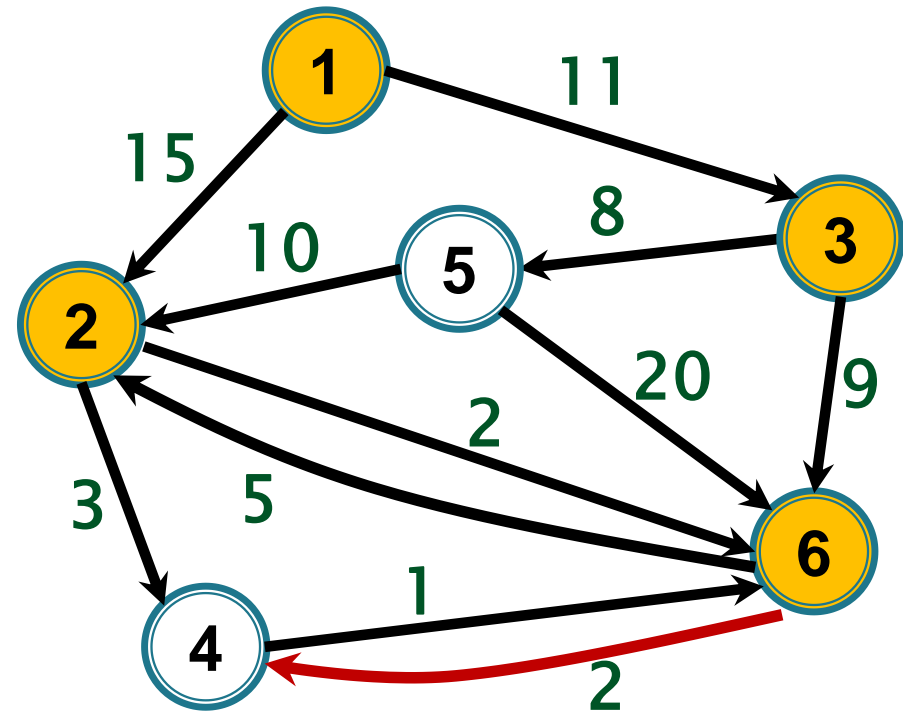
d/tata	1	2	3	4	5	6
	[ 0/0, ]	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ]
Sel. 1:	[ - , ]	15/1,	11/1,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ]
Sel. 3:	[ - , ]	15/1,	- ,	$\infty/0$ ,	19/3,	20/3 ]
Sel. 2:						

$$d[v] = \min\{d[v], d[u] + w(u, v)\}$$



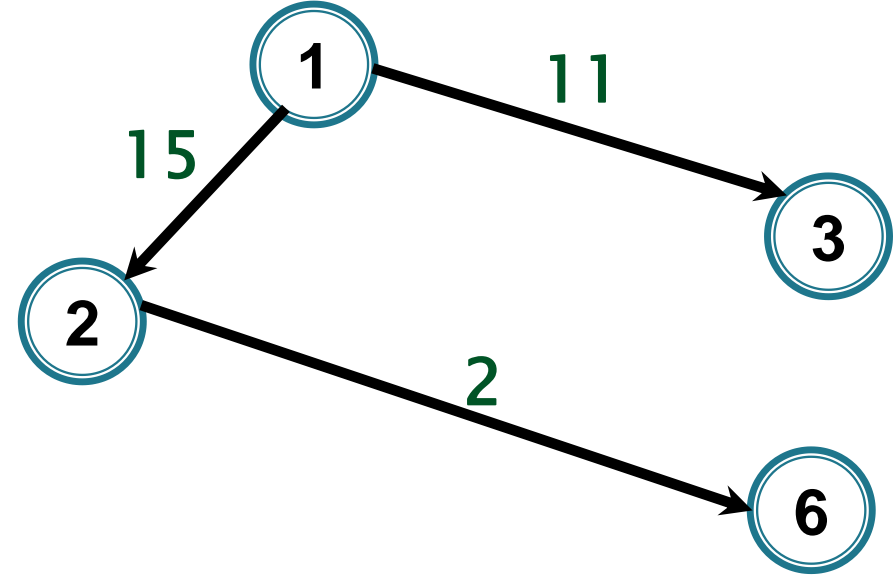
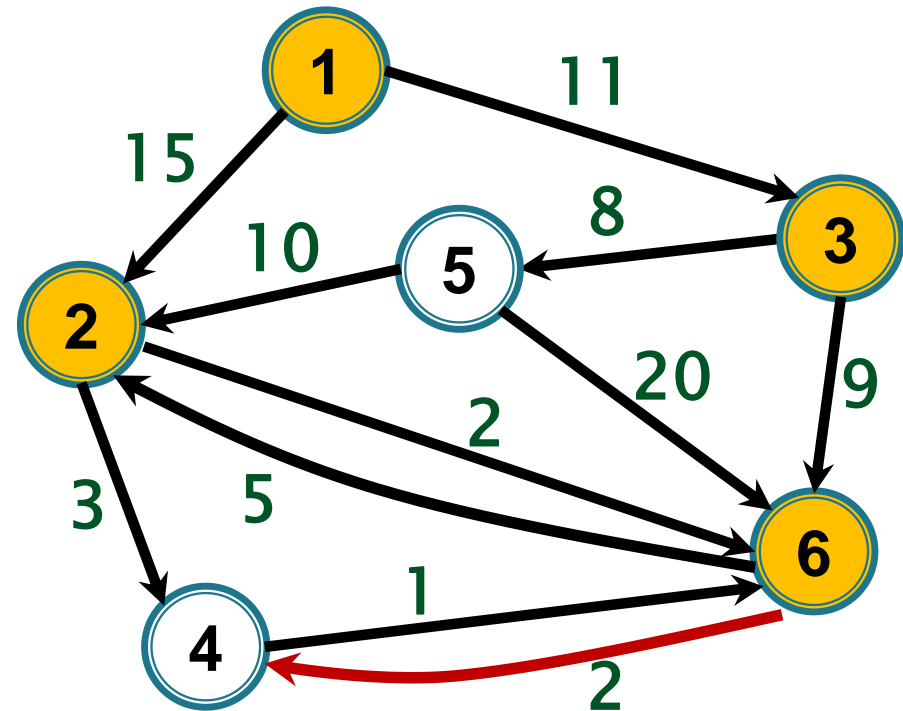
d/tata	1	2	3	4	5	6
	[ 0/0,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ]
Sel. 1:	[ - ,	15/1,	11/1,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ]
Sel. 3:	[ - ,	15/1,	- ,	$\infty/0$ ,	19/3,	20/3 ]
Sel. 2:	[ - ,	- ,	- ,	18/2,	19/3,	17/2 ]

$$d[v] = \min\{d[v], d[u] + w(u, v)\}$$



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:						

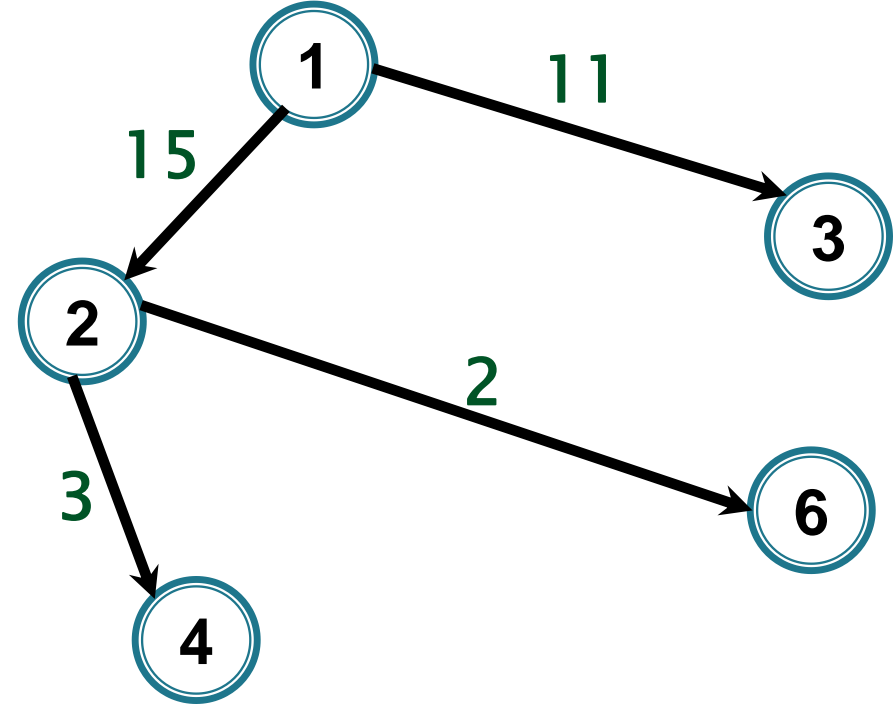
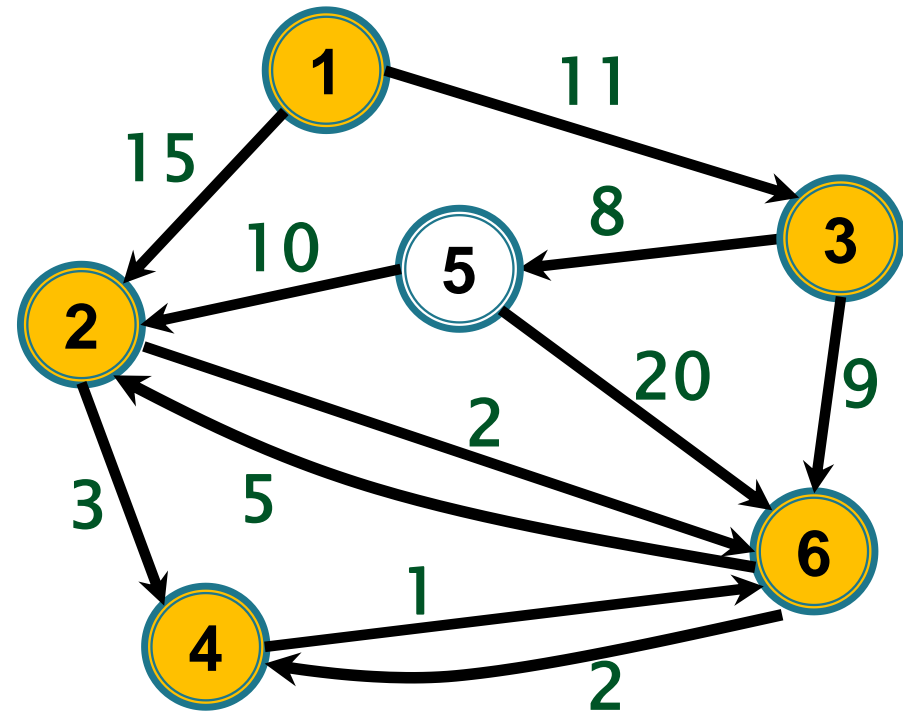
$$d[v] = \min\{d[v], d[u] + w(u, v)\}$$



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,	- ]

$$d[v] = \min\{d[v], d[u] + w(u, v)\}$$

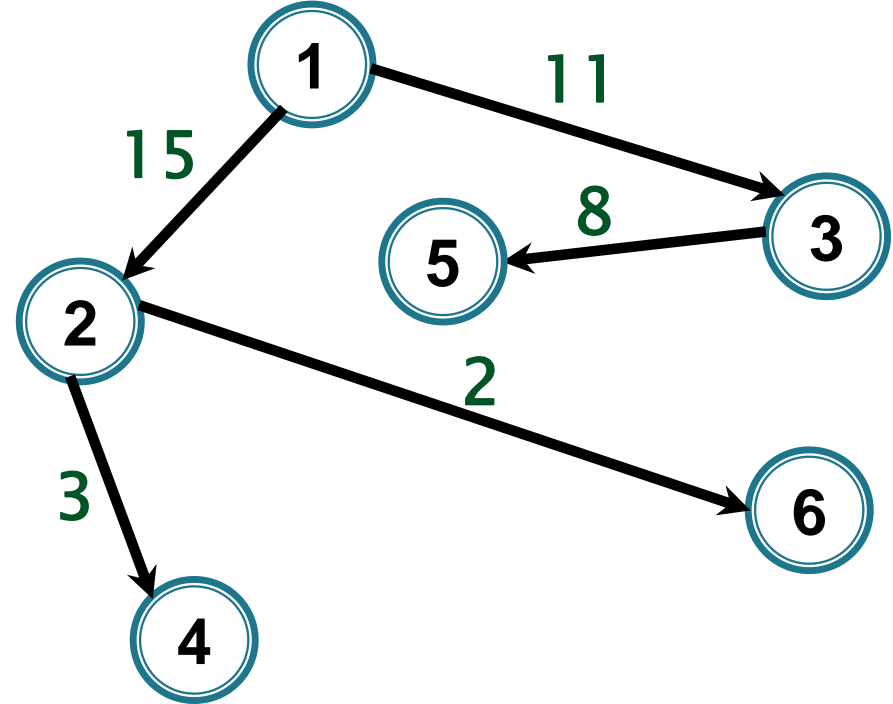


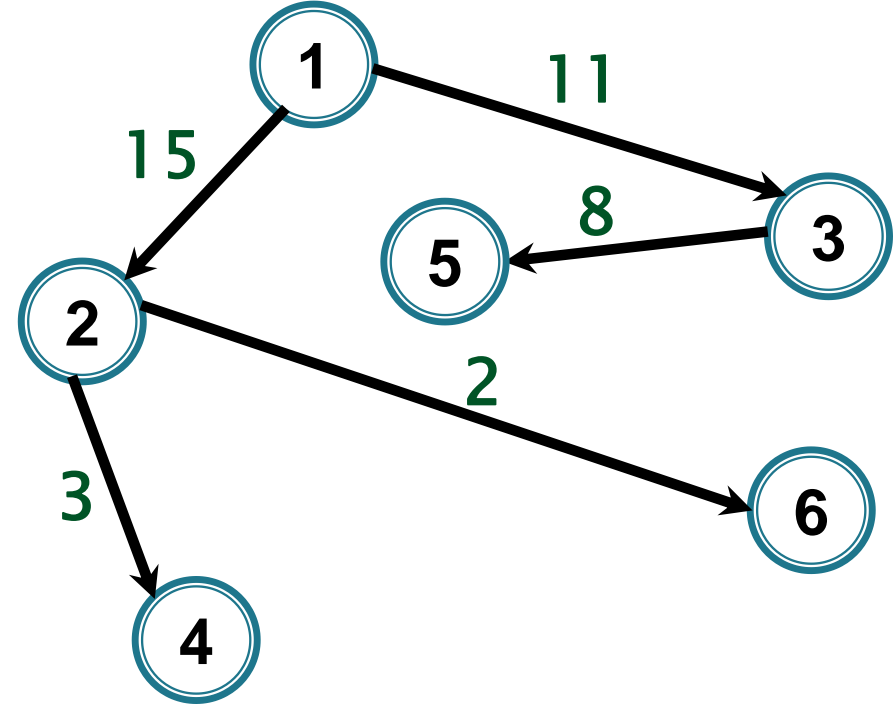
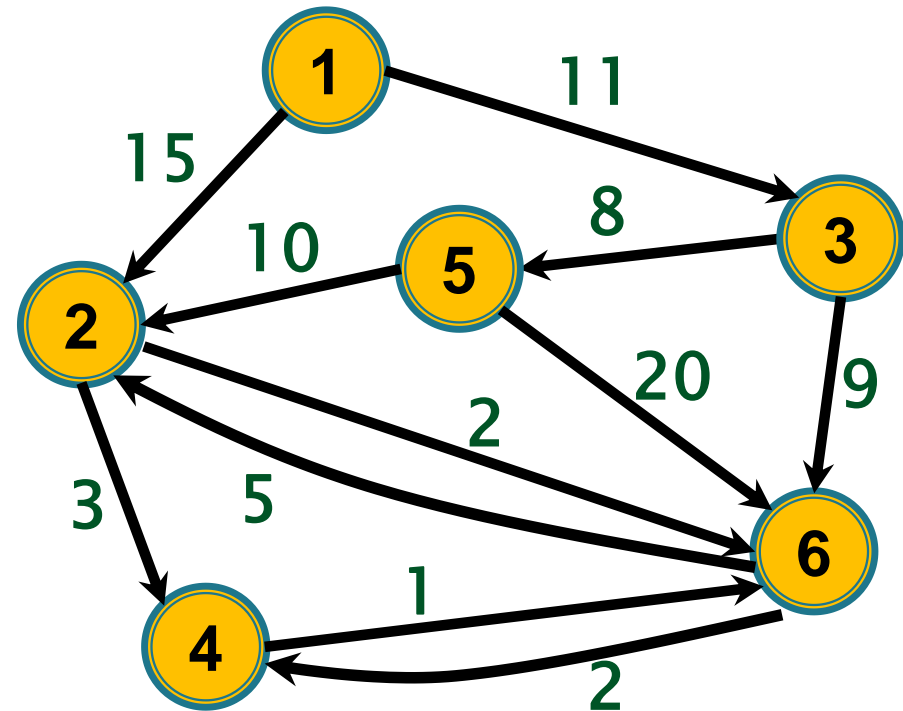


d/tata	1	2	3	4	5	6
	[ <b>0/0</b> ,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ]
Sel. 1:	[ - ,	15/1,	<b>11/1</b> ,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ]
Sel. 3:	[ - ,	<b>15/1</b> ,	- ,	$\infty/0$ ,	19/3,	20/3 ]
Sel. 2:	[ - ,	- ,	- ,	18/2,	19/3,	<b>17/2</b> ]
Sel. 6:	[ - ,	- ,	- ,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

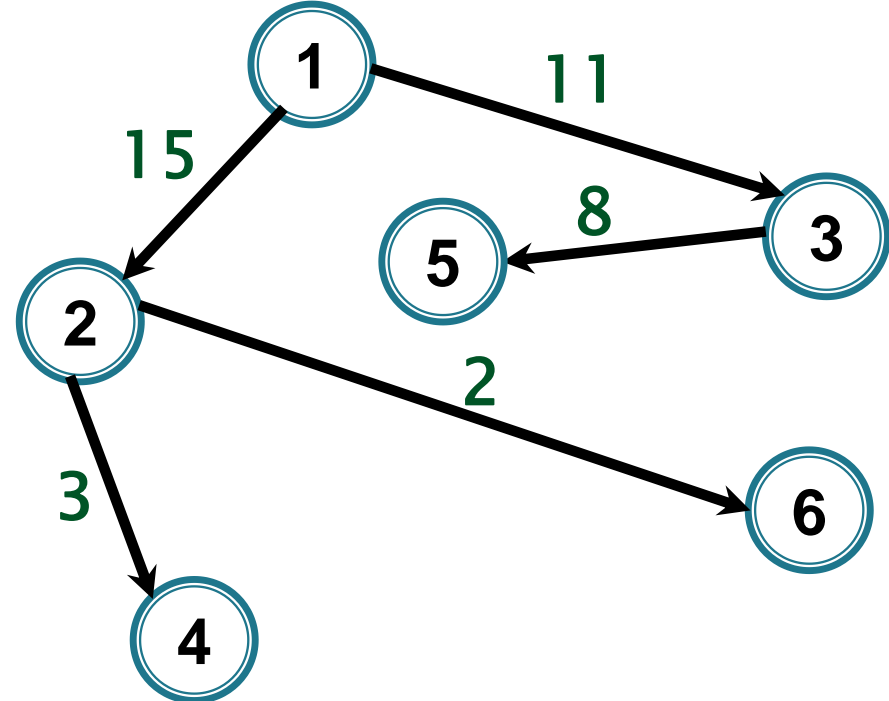
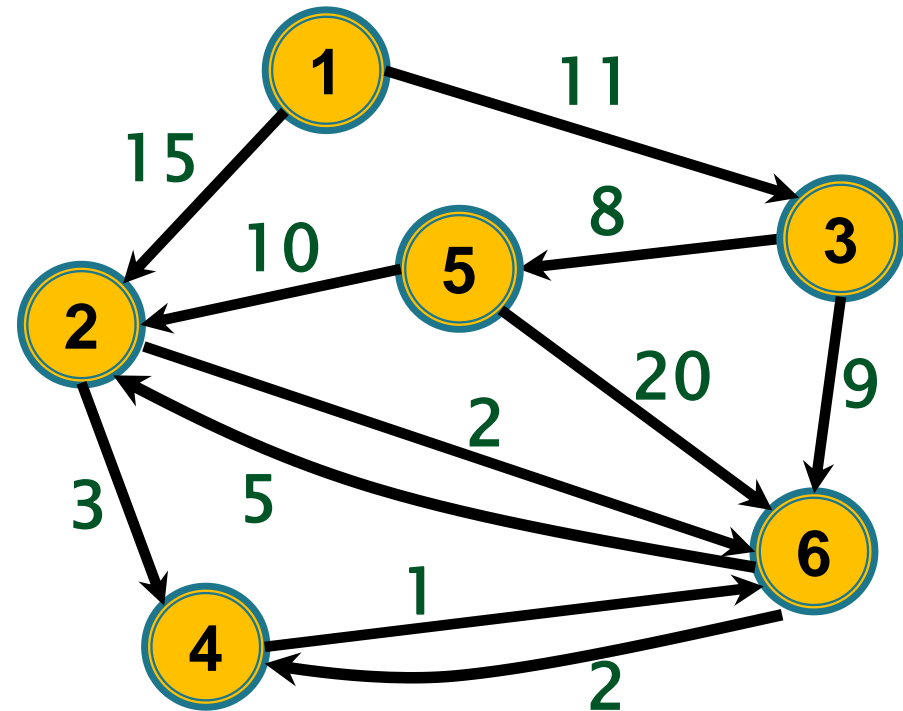
$$d[v] = \min\{d[v], d[u] + w(u, v)\}$$




$$d[v] = \min\{d[v], d[u] + w(u, v)\}$$



d/tata	1	2	3	4	5	6
	[ <b>0/0</b> ,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ]
Sel. 1:	[ - ,	15/1,	<b>11/1</b> ,	$\infty/0$ ,	$\infty/0$ ,	$\infty/0$ ]
Sel. 3:	[ - ,	<b>15/1</b> ,	- ,	$\infty/0$ ,	19/3,	20/3 ]
Sel. 2:	[ - ,	- ,	- ,	18/2,	19/3,	<b>17/2</b> ]
Sel. 6:	[ - ,	- ,	- ,	<b>18/2</b> ,	19/3,	- ]
Sel. 4:	[ - ,	- ,	- ,	- ,	<b>19/3</b> ,	- ]
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( $G, w, s$ )

```
inițializează mulțimea 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
        dacă  $d[u] + w(u, v) < d[v]$  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

$Q[u] = 1$ , dacă  $u$  este selectat în  $V(T)$   
 $0$ , altfel ( $u \in Q$ )

- ▶ Inițializare  $Q$   $\rightarrow$
  - ▶  $n$  \* extragere vârf minim  $\rightarrow$
  - ▶ actualizare etichete vecini  $\rightarrow$
-

# Algoritmul lui Dijkstra

**Complexitate** – reprezentarea lui  $Q$  ca vector

$Q[u] = 1$ , dacă  $u$  este selectat în  $V(T)$

$0$ , altfel ( $u \in Q$ )

- ▶ Inițializare  $Q$   $\rightarrow O(n)$
  - ▶  $n$  \* extragere vârf minim  $\rightarrow$
  - ▶ actualizare etichete vecini  $\rightarrow$
-

# Algoritmul lui Dijkstra

**Complexitate** – reprezentarea lui  $Q$  ca vector

$Q[u] = 1$ , dacă  $u$  este selectat în  $V(T)$   
 $0$ , altfel ( $u \in Q$ )

- ▶ Inițializare  $Q$   $\rightarrow O(n)$
- ▶  $n$  \* extragere vârf minim  $\rightarrow O(n^2)$
- ▶ actualizare etichete vecini  $\rightarrow$  \_\_\_\_\_

# Algoritmul lui Dijkstra

**Complexitate** – reprezentarea lui  $Q$  ca vector

$Q[u] = 1$ , dacă  $u$  este selectat în  $V(T)$

$0$ , altfel ( $u \in Q$ )

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

# Algoritmul lui Dijkstra

**Complexitate** – reprezentarea lui  $Q$  ca vector

$Q[u] = 1$ , dacă  $u$  este selectat în  $V(T)$

$0$ , altfel ( $u \in Q$ )

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

# Algoritmul lui Dijkstra

**Complexitate** – Q min-heap

- ▶ Inițializare Q →
  - ▶  $n$  \* 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  $\rightarrow O(n)$
  - ▶  $n$  \* extragere vârf minim  $\rightarrow O(n \log n)$
  - ▶ actualizare etichete vecini  $\rightarrow$
-



# Algoritmul lui Dijkstra

**Complexitate** – Q min-heap

- ▶ Inițializare Q  $\rightarrow O(n)$
- ▶  $n$  \* extragere vârf minim  $\rightarrow O(n \log n)$
- ▶ actualizare etichete vecini  $\rightarrow O(m \log n)$
- !!+ 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?

# Algoritmul lui Dijkstra



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

# Algoritmul lui Dijkstra

- ▶ 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?**

# Algoritmul lui Dijkstra

- ▶ 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

# Corectitudinea Algoritmului lui Dijkstra



# Corectitudine

► **Lema.** 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  $tata$ :

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

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

# Corectitudine

- ▶ **Demonstrație:** Rezultă din faptul că actualizarea unei etichete  $d[v]$  corespunde extinderii unui drum deja determinat de la  $s$  la  $u$  prin adăugarea arcului  $uv$ .
- ▶ Detaliat – Inducție

# Corectitudine

- ▶ **Lema.** 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  $tata$ :  
 $tata[u] =$  predecesorul lui  $u$  pe un drum de la  $s$  la  $u$  de cost  $d[u]$
  - b)  $d[u] \geq \delta(s, u)$
- ▶ **Consecință.** 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.

# Corectitudine

## ► Teoremă

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:

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

și tata memorează un arbore al distanțelor față de  $s$ .

# Corectitudine

- ▶ **Demonstrație** Inducție:  $d[x] = \delta(s, x) \quad \forall x \notin Q$  (=deja selectat)

# Corectitudine

- ▶ **Demonstrație** Inducție:  $d[x] = \delta(s, x) \quad \forall x \notin Q$  (=deja selectat)
  - Primul vârf selectat este  $s$  și  $d[s] = 0 = \delta(s, x)$

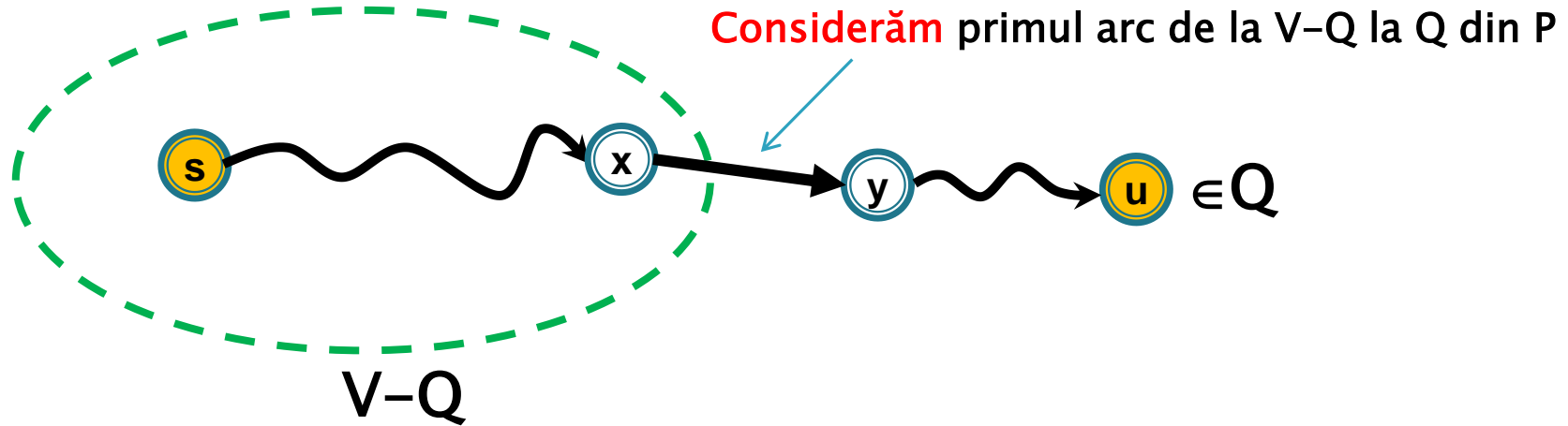


# Corectitudine

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

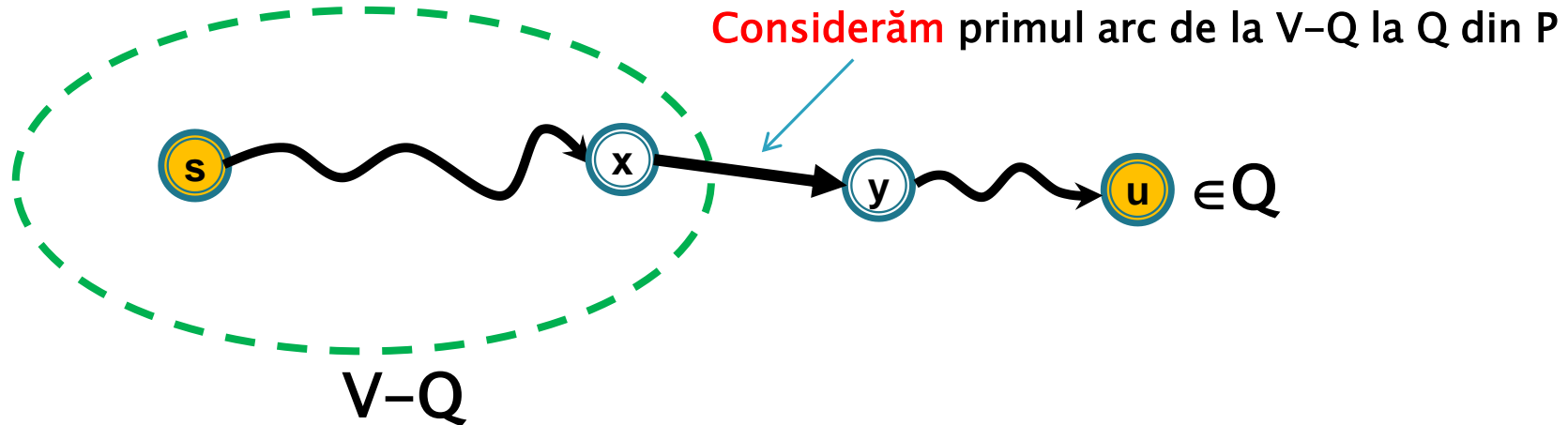
# Corectitudine

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



# Corectitudine

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

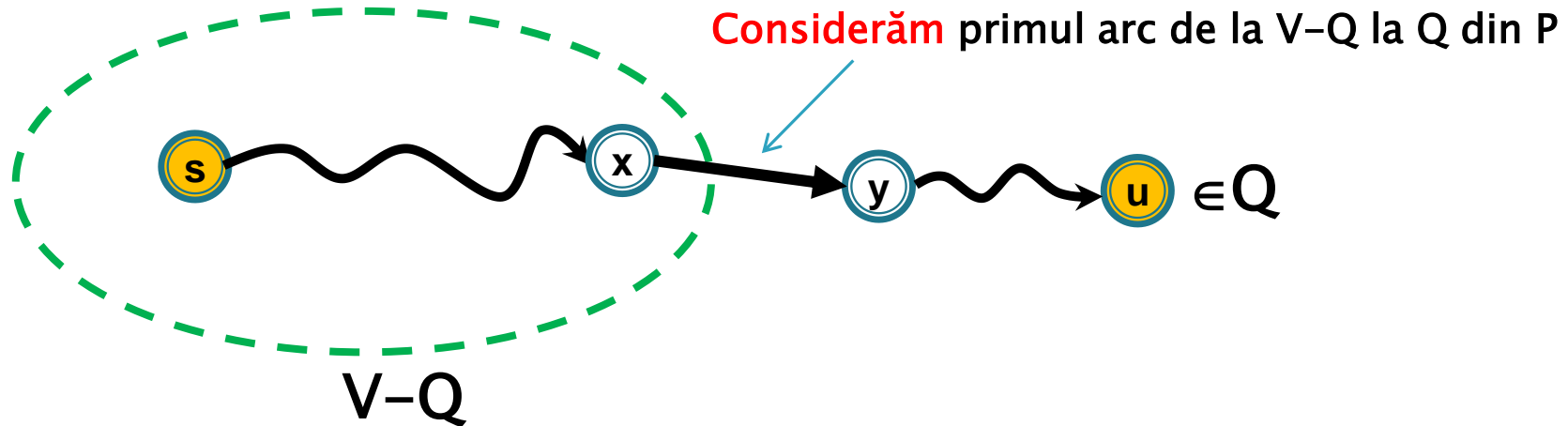


din modul în care este ales  $u$  (cu  $d$  minim)

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

# Corectitudine

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

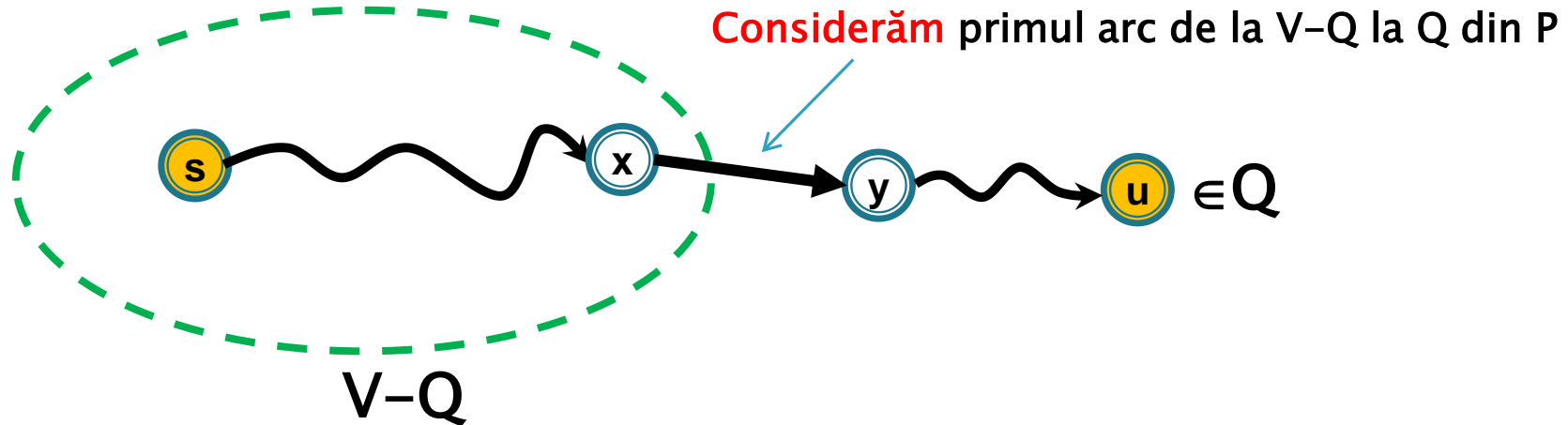


din modul în care este ales  $u$       dupa relaxarea lui  $xy$  (!!x a fost deja selectat)

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

# Corectitudine

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

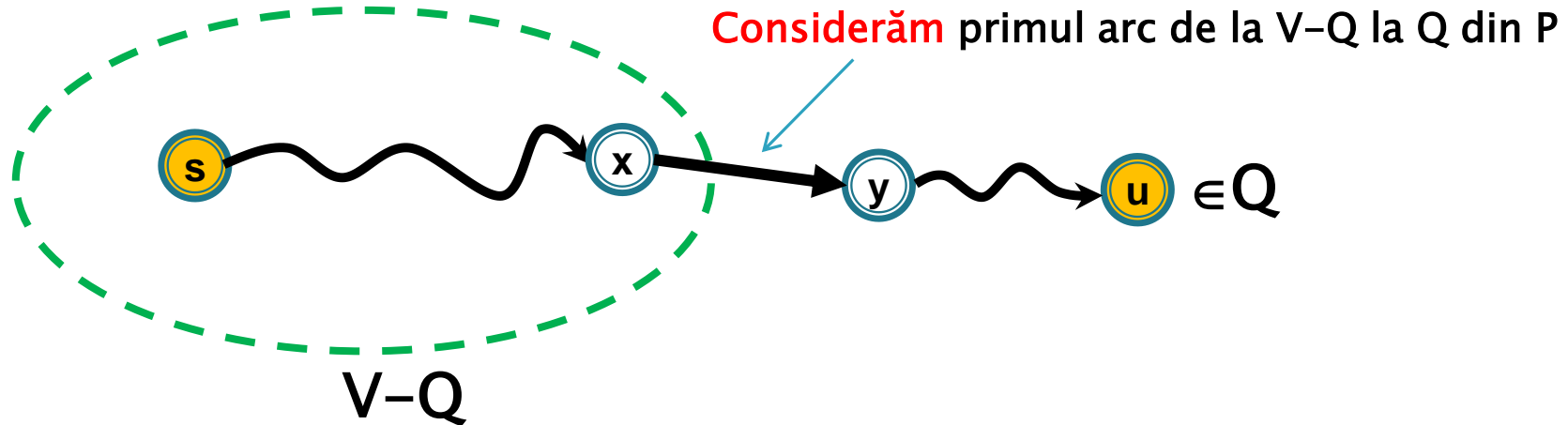


din modul în care este ales  $u$       dupa relaxarea lui  $xy$  (!!x a fost deja selectat)

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

# Corectitudine

- ▶ Presupunem la pasul curent  $d[x] = \delta(s, x) \quad \forall x \notin Q$  (=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 fost deja selectat)

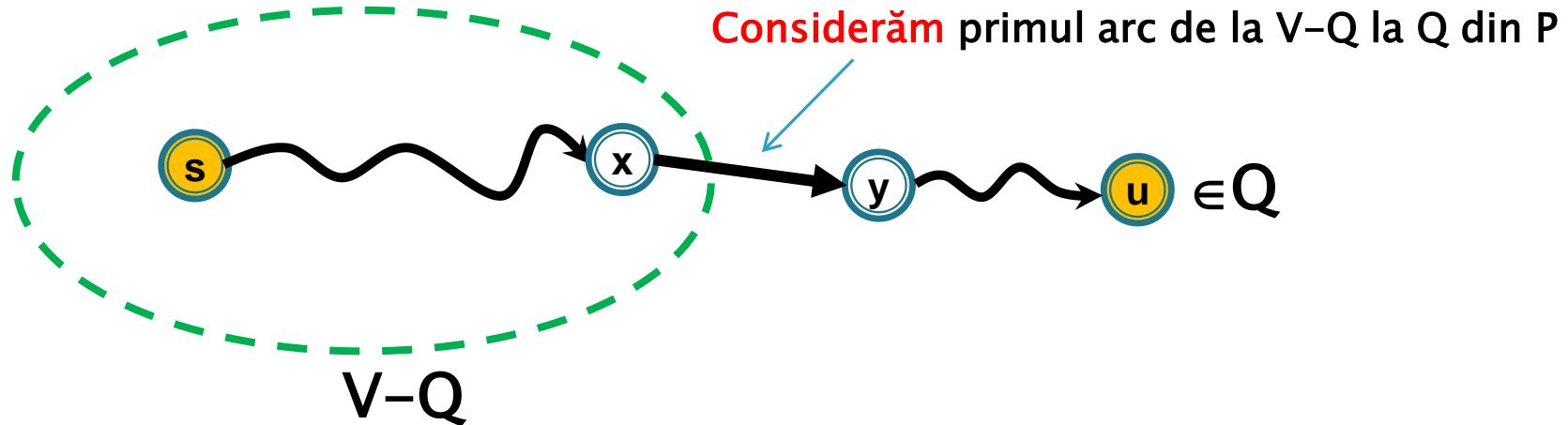
$$d[u] \leq d[y] \leq d[x] + w(x, y) = \delta(s, x) + w(x, y) = w(s \xrightarrow{P} y)$$

$\leq$

ipoteza de inductie pentru  $x$

# Corectitudine

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



din modul în care este ales  $u$       după relaxarea lui  $xy$  (!! $x$  a fost deja selectat)

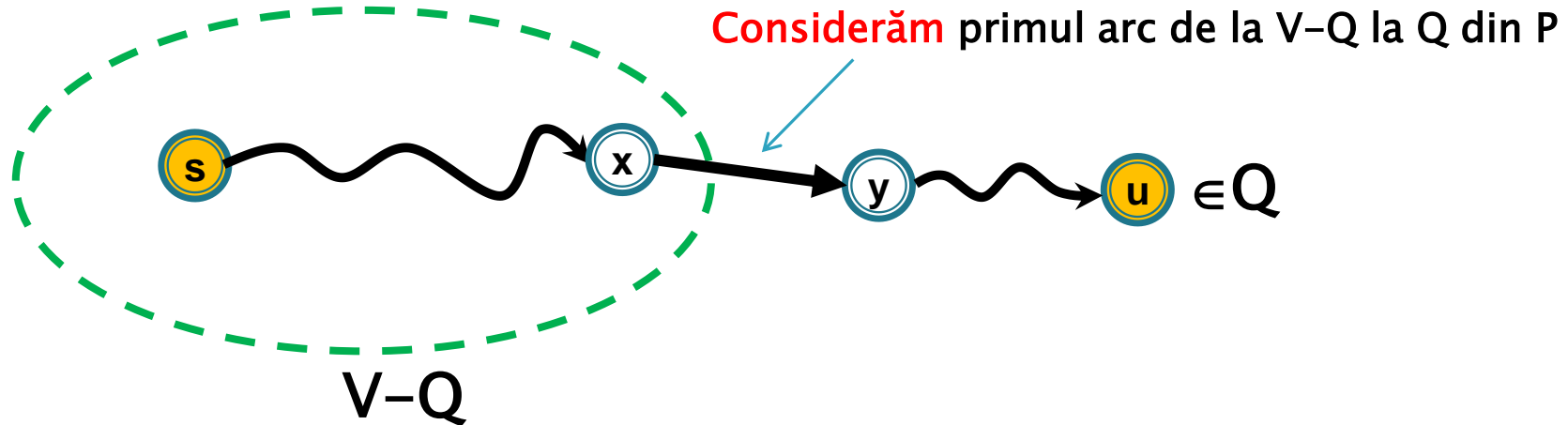
$$d[u] \leq d[y] \leq d[x] + w(x, y) = \delta(s, x) + w(x, y) = w(s \xrightarrow{P} y)$$

$\leq$       ipoteza de inducție pentru  $x$

costurile arcelor sunt nenegative

# Corectitudine

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



din modul în care este ales  $u$       după relaxarea lui  $xy$  (!! $x$  a fost deja selectat)

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

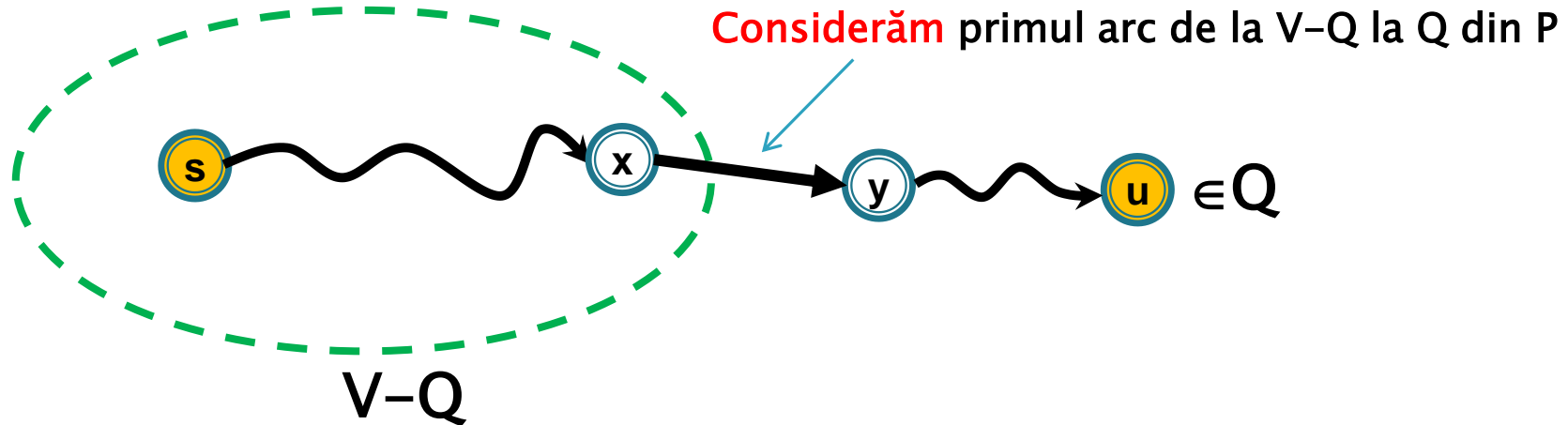
$\leq w(P)$       ipoteza de inducție pentru  $x$

costurile arcelor sunt nenegative



# Corectitudine

- ▶ Presupunem la pasul curent  $d[x] = \delta(s, x) \quad \forall x \notin Q$  (=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 fost deja selectat)

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

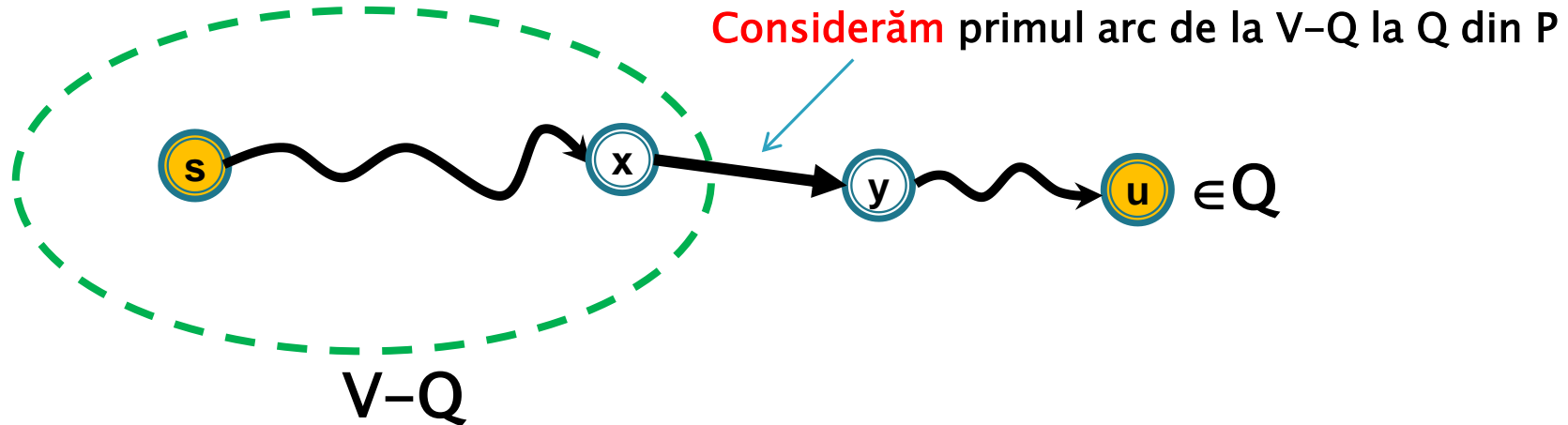
$\leq w(P) \leq d[u]$

ipoteza de inductie pentru  $x$

Din Lema,  $d[u] = \text{costul unui drum de la } s \text{ la } u$ ,  
deci este  $\geq \text{costul drumului minim de la } s \text{ la } u$

# Corectitudine


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



$$\begin{aligned} d[u] &\leq d[y] \leq d[x] + w(x, y) = \delta(s, x) + w(x, y) = w(s \xrightarrow{P} y) \\ &\leq w(P) \leq d[u] \end{aligned}$$

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

# Drumuri maxime

 Putem modifica algoritmul lui Dijkstra de determinare de drumuri minime în grafuri (nu neapărat aciclice) a.î. să determine drumuri maxime (elementare) de la s la celelalte vârfuri?

# Drumuri maxime

Putem modifica algoritmul lui Dijkstra de determinare de drumuri minime în grafuri (nu neapărat aciclice) a.î. să determine drumuri maxime (elementare) de la  $S$  la celelalte vârfuri

- Modificăm astfel doar inițializarea distanțelor (cu  $-\infty$  în loc de  $+\infty$ ) și inversăm condiția de la relaxarea arcelor pentru a calcula maxim în loc de minim



**Corectitudine** - probabil similar cu Dijkstra?!!

# Drumuri maxime

Putem modifica algoritmul lui Dijkstra de determinare de drumuri minime în grafuri (nu neapărat aciclice) a.î. să determine drumuri maxime (elementare) de la  $S$  la celelalte vârfuri

- Modificăm astfel doar inițializarea distanțelor (cu  $-\infty$  în loc de  $+\infty$ ) și inversăm condiția de la relaxarea arcelor pentru a calcula maxim în loc de minim
- **Corectitudine** - probabil similar cu Dijkstra?!!

