

Drumuri minime între toate
perechile de vârfuri

Algoritmul Floyd–Warshall

Problema drumurilor minime între toate perechile de vârfuri

Se dă:

- un graf **orientat** ponderat $G = (V, E, w)$

Pentru oricare două vârfuri x și y al lui G să se determine distanța de la x la y și un drum minim de la x la y

Ponderile pot fi și negative dar **NU** există **circuite cu cost negativ** în G



► Soluția 1

Se aplică algoritmul lui Dijkstra pentru fiecare vârf x

► Soluția 1

Se aplică algoritmul lui Dijkstra pentru fiecare vârf x

!!!funcționează dacă ponderile sunt pozitive

Complexitate = $n * \text{complexitate Dijkstra}$

➤ Soluția 2

Algoritmul Floyd–Warshall

Floyd–Warshall

- ▶ Fie $W = (w_{ij})_{i,j=1..n}$ **matricea costurilor** grafului G :

$$w_{ij} = \begin{cases} 0, & \text{daca } i = j \\ w(i,j), & \text{daca } ij \in E \\ \infty, & \text{daca } ij \notin E \end{cases}$$

- ▶ Vrem să calculăm **matricea distanțelor** $D = (d_{ij})_{i,j=1..n}$:

$$d_{ij} = \delta(i, j)$$

Floyd–Warshall

- ▶ Fie $W = (w_{ij})_{i,j=1..n}$ **matricea costurilor** grafului G :

$$w_{ij} = \begin{cases} 0, & \text{daca } i = j \\ w(i,j), & \text{daca } ij \in E \\ \infty, & \text{daca } ij \notin E \end{cases}$$

- ▶ Vrem să calculăm **matricea distanțelor** $D = (d_{ij})_{i,j=1..n}$:

$$d_{ij} = \delta(i, j)$$

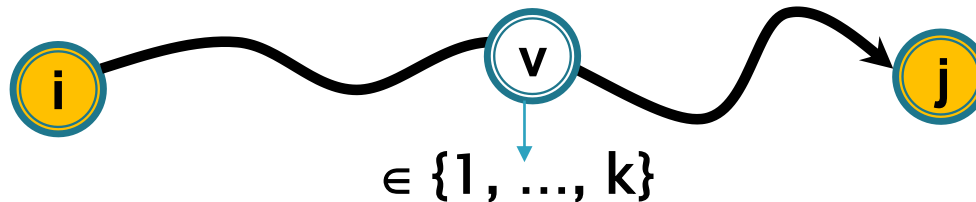
- ▶ **Observație:** w_{ij} = costul minim al unui i – j drum fără vârfuri intermediare (cu cel mult un arc)

Floyd–Warshall

► Ideea algoritmului Floyd–Warshall:

Pentru $k = 1, 2, \dots, n$ calculăm pentru oricare două vârfuri i, j :

costul minim al unui drum (elementar) de la i la j care are ca vârfuri intermediare doar vârfuri din mulțimea $\{1, 2, \dots, k\}$



Floyd–Warshall

► Ideea algoritmului Floyd–Warshall:

Astfel, pentru $k = 1, 2, \dots, n$ calculăm matricea

$$D^{(k)} = (d_{ij}^k)_{i,j=1..n} :$$

d_{ij}^k = costul minim al unui drum de la i la j care are
vârfurile intermediare în $\{1, 2, \dots, k\}$

• Inițializare:



Floyd–Warshall

► Ideea algoritmului Floyd–Warshall:

Astfel, pentru $k = 1, 2, \dots, n$ calculăm matricea

$$D^{(k)} = (d_{ij}^k)_{i,j=1..n} :$$

d_{ij}^k = costul minim al unui drum de la i la j care are vârfurile intermediare în $\{1, 2, \dots, k\}$

- Inițializare: $D^{(0)} = W$



Care este matricea distanțelor?

Floyd–Warshall

► Ideea algoritmului Floyd–Warshall:

Astfel, pentru $k = 1, 2, \dots, n$ calculăm matricea

$$D^{(k)} = (d_{ij}^k)_{i,j=1..n} :$$

d_{ij}^k = costul minim al unui drum de la i la j care are
vârfurile intermediare în $\{1, 2, \dots, k\}$

- Inițializare: $D^{(0)} = W$

- Avem $D^{(n)} = D$

Floyd–Warshall

- ▶ **Ideea algoritmului Floyd–Warshall:**

Pentru a reține și un drum minim



Floyd–Warshall

► Ideea algoritmului Floyd–Warshall:

Pentru a reține și un drum minim

– matrice de predecesori $P^{(k)} = (p_{ij}^k)_{i,j=1..n}$:

p_{ij}^k = predecesorul lui j pe drumul minim curent
găsit de la i la j care are vârfurile
intermediare în $\{1, 2, \dots, k\}$

Floyd–Warshall



- ▶ Cum calculăm elementele matricei $D^{(k)}$?

Floyd–Warshall

- ▶ Cum calculăm elementele matricei $D^{(k)}$?



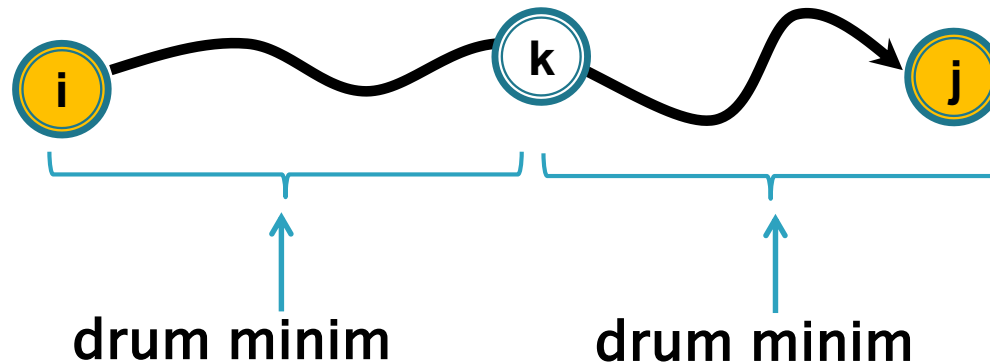
Relație de recurență

Floyd–Warshall

- **Ideea** de calcul al matricei $D^{(k)}$:

Fie P un drum (elementar) de cost minim de la i la j cu vârfurile intermediare în mulțimea $\{1, 2, \dots, k\}$

- Dacă vârful k este vârf intermediar al lui P

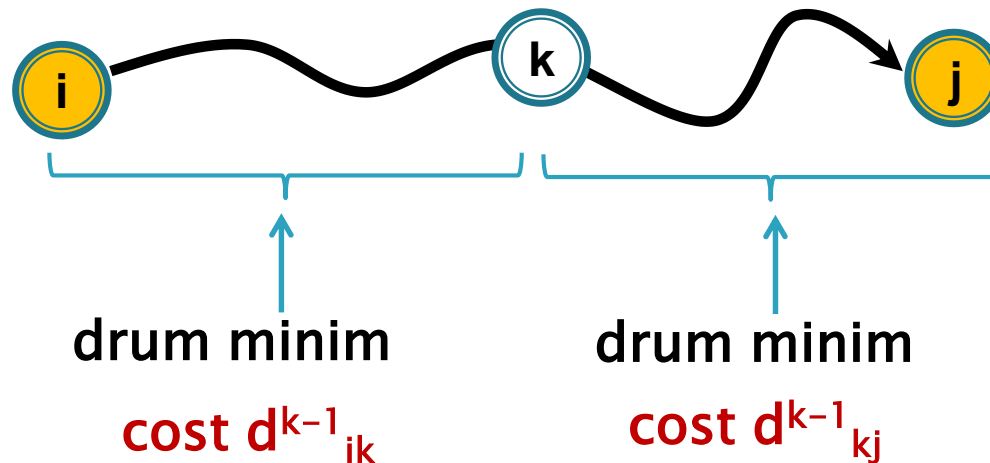


Floyd-Warshall

- **Ideea** de calcul al matricei $D^{(k)}$:

Fie P un drum de cost minim de la i la j cu vârfurile intermediare în mulțimea $\{1, 2, \dots, k\}$

- Dacă vârful k este vârf intermediar al lui P

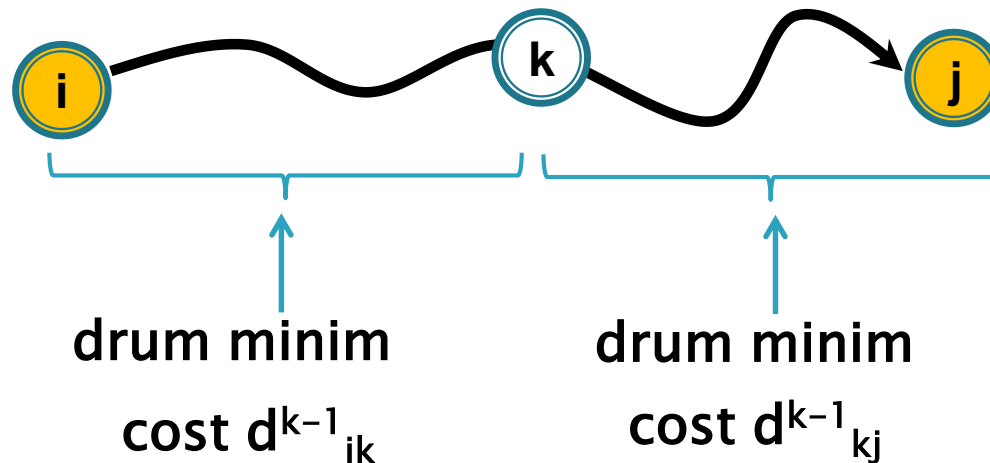


Floyd-Warshall

- **Ideea** de calcul al matricei $D^{(k)}$:

Fie P un drum de cost minim de la i la j cu vârfurile intermediare în mulțimea $\{1, 2, \dots, k\}$

- Dacă vârful k este vârf intermediar al lui P



$$\Rightarrow d^k_{ij} = \min\{d^{k-1}_{ij}, d^{k-1}_{ik} + d^{k-1}_{kj}\}$$

Floyd–Warshall

- ▶ Se obține astfel relația

$$d_{ij}^k = \min \{d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}\}$$

- ▶ Observații

- Avem

$$d_{ik}^k = d_{ik}^{k-1}$$

$$d_{kj}^k = d_{kj}^{k-1}$$

de aceea în implementarea algoritmului putem folosi o singură matrice

Floyd–Warshall

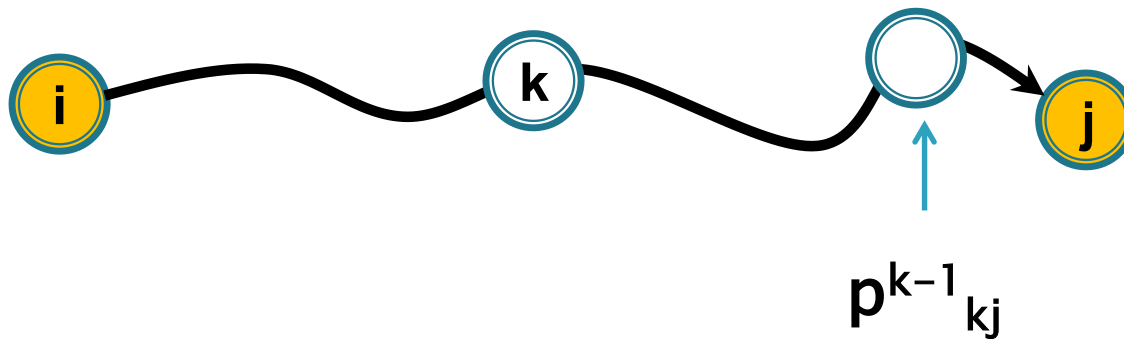
- ▶ Se obține astfel relația

$$d_{ij}^k = \min \{d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}\}$$

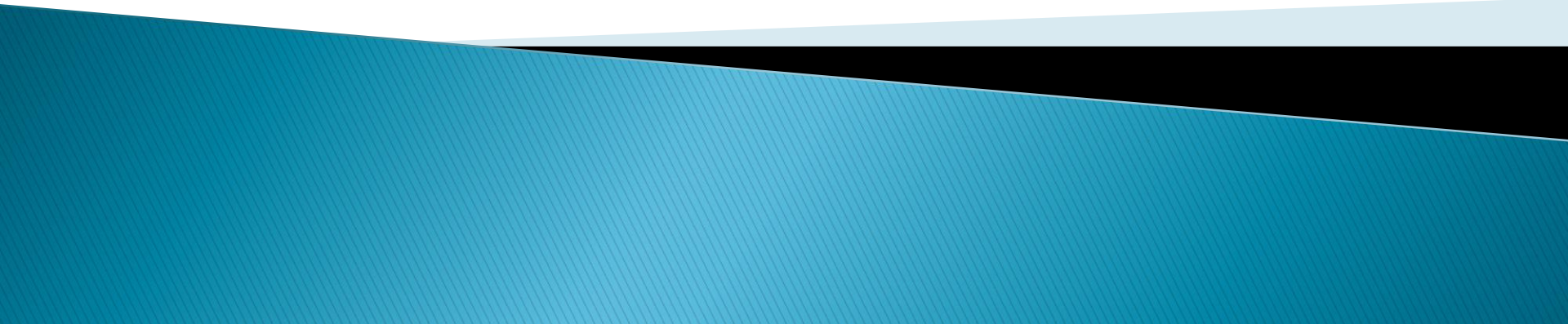
- ▶ Observații

Când de actualizează $d_{ij}^k = d_{ik}^{k-1} + d_{kj}^{k-1}$ trebuie actualizat și p_{ij}^k

$$p_{ij}^k = p_{kj}^{k-1}$$



Implementare



Floyd–Warshall

- ▶ Conform observațiilor anterioare, putem folosi o unică matrice D
- ▶ **Inițializare**

$$d[i][j] = w(i,j) - \text{costul arcului } (i,j)$$

$$p[i][j] = \begin{cases} i, & \text{daca } ij \in E \\ 0, & \text{altfel} \end{cases}$$

Floyd(G, w)

```
for(i=1;i<=n;i++)    //initial d = matricea costurilor
    for(j=1;j<=n;j++) {
        d[i][j]=w[i][j];
        if(w[i][j]==  $\infty$ )
            p[i][j]=0;
        else
            p[i][j]=i;
    }
```

Floyd(G, w)

```
for (i=1; i<=n; i++)
    for (j=1; j<=n; j++) {
        d[i][j]=w[i][j];
        if (w[i][j]==  $\infty$ )
            p[i][j]=0;
        else
            p[i][j]=i;
    }
for (k=1; k<=n; k++) //varfuri intermediare
```


Floyd(G, w)

```
for (i=1 ; i<=n ; i++)  
    for (j=1 ; j<=n ; j++) {  
        d[i][j]=w[i][j] ;  
        if (w[i][j]==  $\infty$ )  
            p[i][j]=0 ;  
        else  
            p[i][j]=i ;  
    }  
for (k=1 ; k<=n ; k++)  
    for (i=1 ; i<=n ; i++)  
        for (j=1 ; j<=n ; j++)
```

Floyd(G, w)

```
for (i=1 ; i<=n ; i++)
    for (j=1 ; j<=n ; j++) {
        d[i][j]=w[i][j] ;
        if (w[i][j]==  $\infty$ )
            p[i][j]=0 ;
        else
            p[i][j]=i ;
    }
for (k=1 ; k<=n ; k++)
    for (i=1 ; i<=n ; i++)
        for (j=1 ; j<=n ; j++)
            if (d[i][j]>d[i][k]+d[k][j]) {
                d[i][j]=d[i][k]+d[k][j] ;
                p[i][j]=p[k][j] ;
            }
```

Floyd–Warshall

- ▶ **leșire:** matricea d = **matricea distanțelor minime**
- ▶ Afișarea unui drum de la i la j , **daca** $d[i][j] < \infty$, se face folosind matricea p

Floyd-Warshall

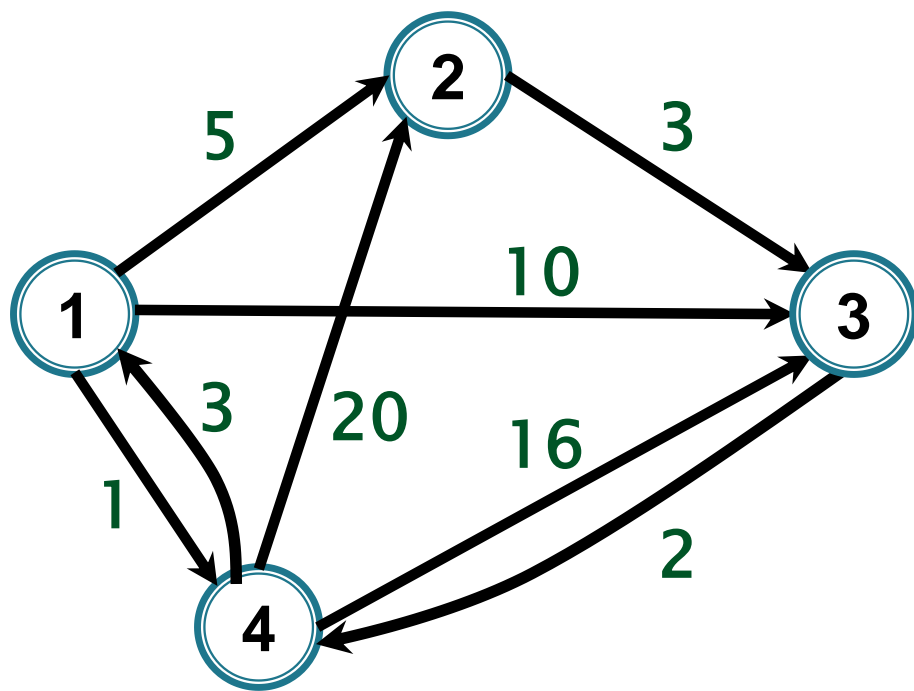
- ▶ **leșire:** matricea d = **matricea distanțelor minime**
- ▶ Afișarea unui drum de la i la j , **daca** $d[i][j] < \infty$, se face folosind matricea p

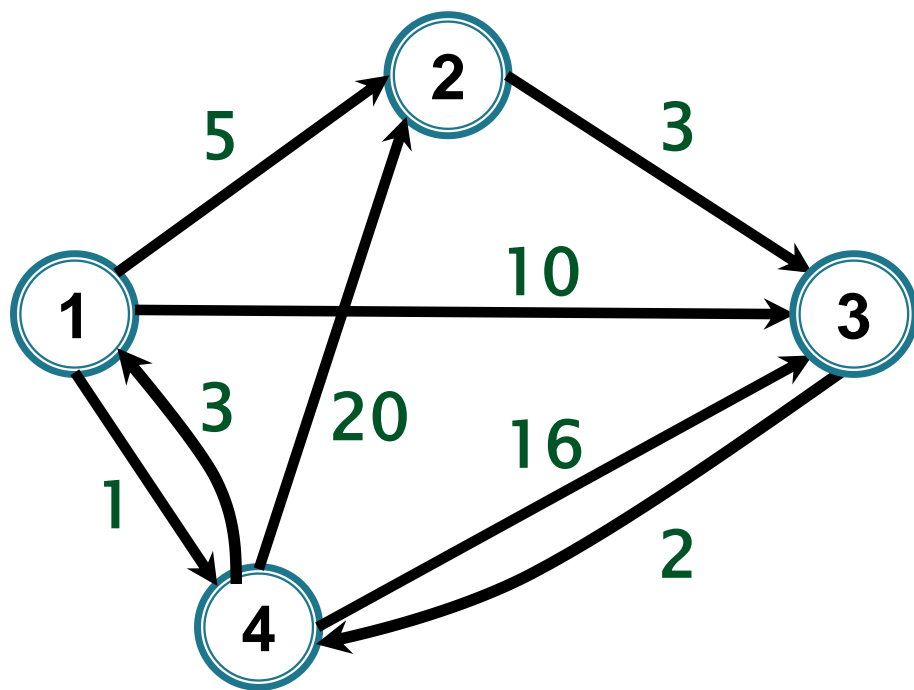
```
void drum(int i,int j){  
    if(i!=j)  
        drum(i,p[i][j]);  
    cout<<j<<" ";  
}
```

Floyd–Warshall

Complexitate – $O(n^3)$





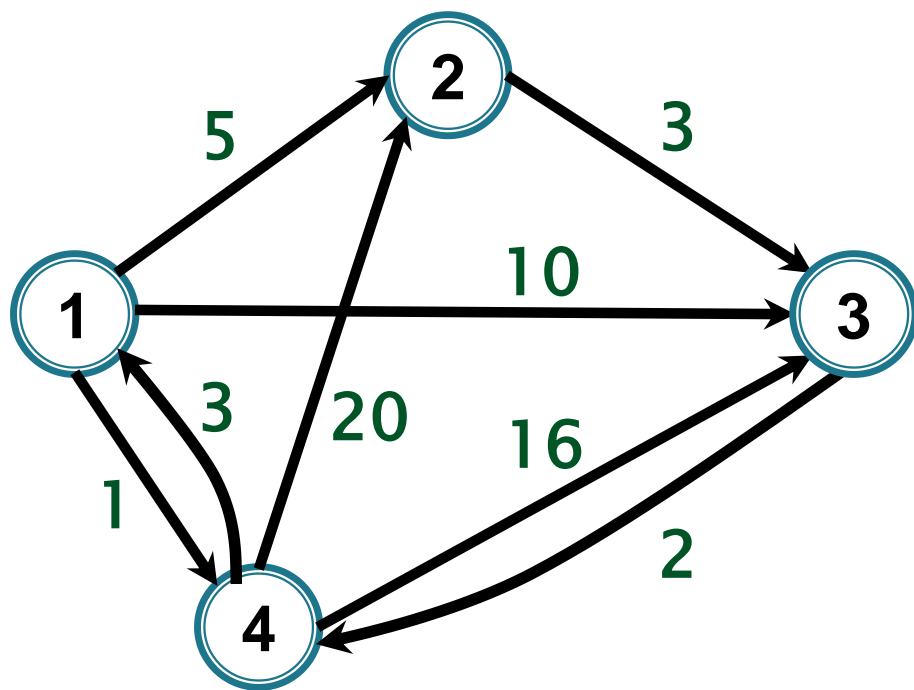


W=d=

0	5	10	1
∞	0	3	∞
∞	∞	0	2
3	20	16	0

p=

0	1	1	1
0	0	2	0
0	0	0	3
4	4	4	0



$W=d=$

0	5	10	1
∞	0	3	∞
∞	∞	0	2
3	20	16	0

$p=$

0	1	1	1
0	0	2	0
0	0	0	3
4	4	4	0

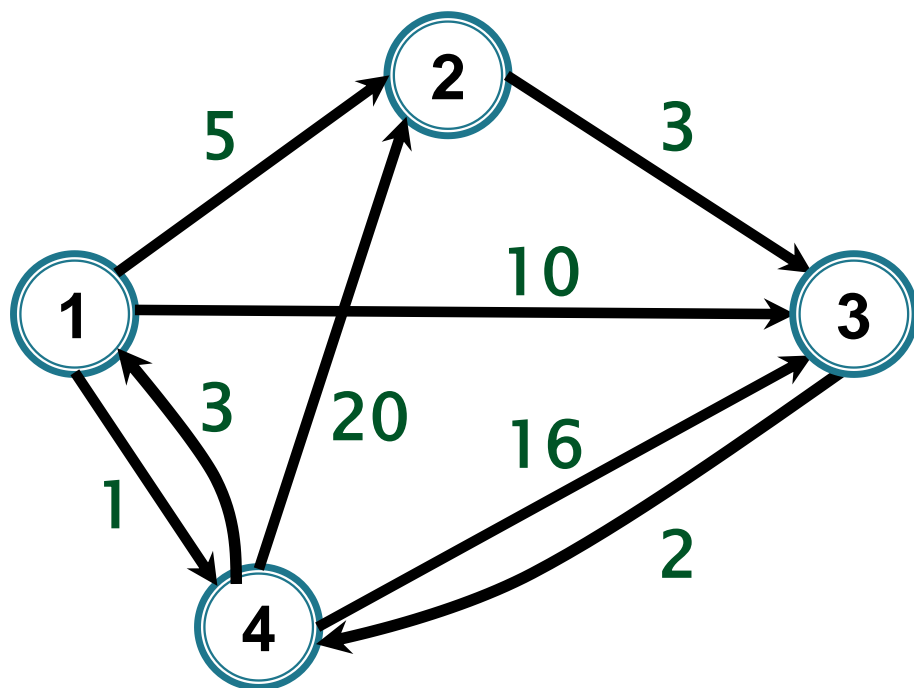
$k=1$

$d=$

0	5	10	1
∞	0	3	∞
∞	∞	0	2
3	8	13	0

$p=$

0	1	1	1
0	0	2	0
0	0	0	3
4	1	1	0



$W=d=$

0	5	10	1
∞	0	3	∞
∞	∞	0	2
3	20	16	0

$p=$

0	1	1	1
0	0	2	0
0	0	0	3
4	4	4	0

$k=1$

$d=$

0	5	10	1
∞	0	3	∞
∞	∞	0	2
3	8	13	0

$k=2$

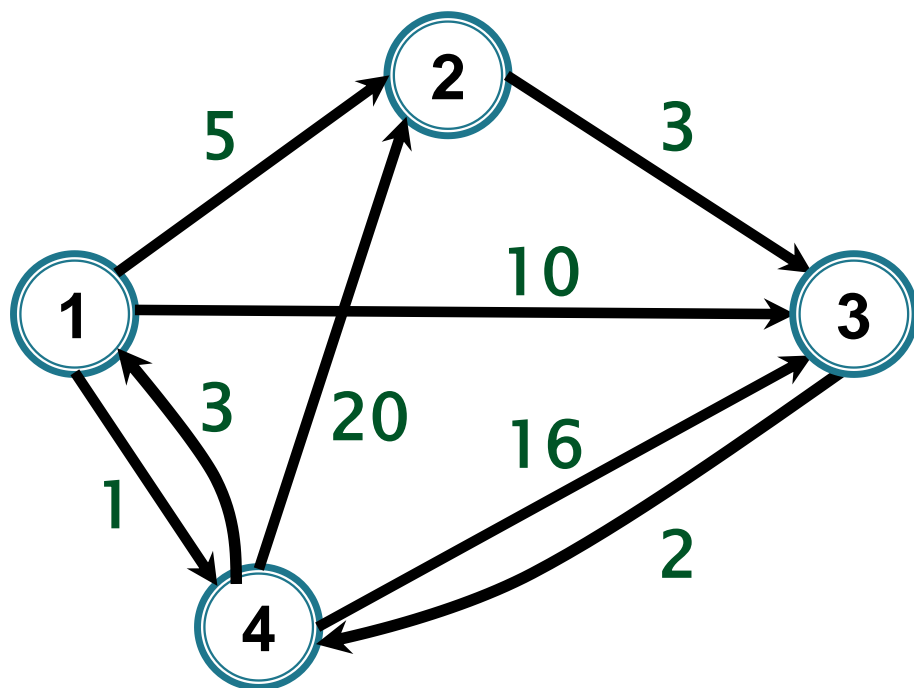
0	5	8	1
∞	0	3	∞
∞	∞	0	2
3	8	11	0

$d=$

$p=$

0	1	1	1
0	0	2	0
0	0	0	3
4	1	1	0

0	1	2	1
0	0	2	0
0	0	0	3
4	1	2	0



$W=d=$

0	5	10	1
∞	0	3	∞
∞	∞	0	2
3	20	16	0

$p=$

0	1	1	1
0	0	2	0
0	0	0	3
4	4	4	0

$k=1$

$d=$

0	5	10	1
∞	0	3	∞
∞	∞	0	2
3	8	13	0

$k=2$

0	5	8	1
∞	0	3	∞
∞	∞	0	2
3	8	11	0

$k=3$

0	5	8	1
∞	0	3	5
∞	∞	0	2
3	8	11	0

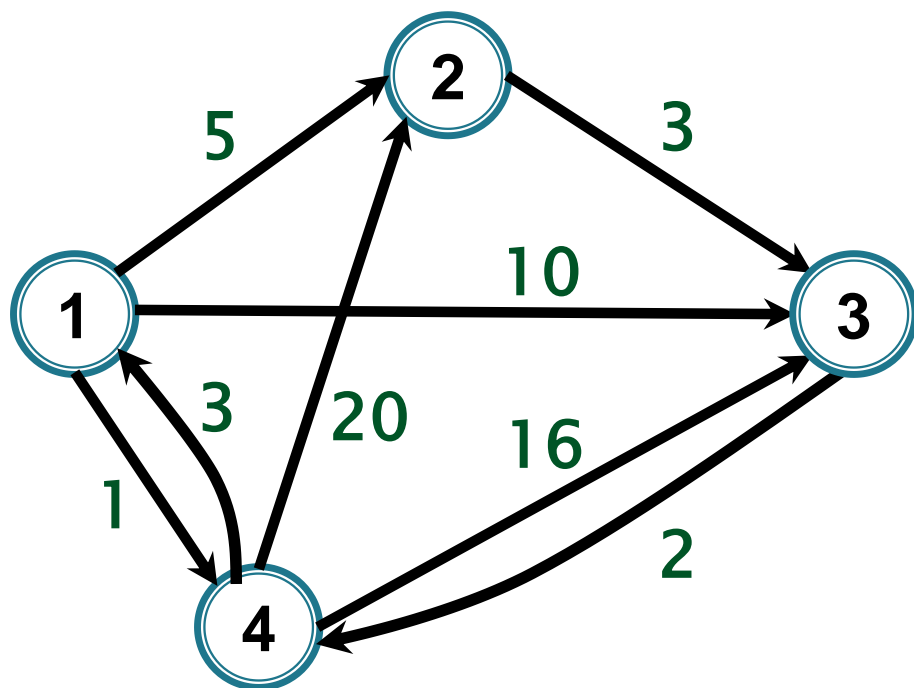
$d=$

$p=$

0	1	1	1
0	0	2	0
0	0	0	3
4	1	1	0

0	1	2	1
0	0	2	0
0	0	0	3
4	1	2	0

0	1	2	1
0	0	2	3
0	0	0	3
4	1	2	0



$W=d=$

0	5	10	1
∞	0	3	∞
∞	∞	0	2
3	20	16	0

$p=$

0	1	1	1
0	0	2	0
0	0	0	3
4	4	4	0

$k=1$

0	5	10	1
∞	0	3	∞
∞	∞	0	2
3	8	13	0

$k=2$

0	5	8	1
∞	0	3	∞
∞	∞	0	2
3	8	11	0

$k=3$

0	5	8	1
∞	0	3	5
∞	∞	0	2
3	8	11	0

$k=4$

0	5	8	1
8	0	3	5
5	10	0	2
3	8	11	0

$d=$

0	1	1	1
0	0	2	0
0	0	0	3
4	1	1	0

0	1	2	1
0	0	2	0
0	0	0	3
4	1	2	0

0	1	2	1
0	0	2	3
0	0	0	3
4	1	2	0

0	1	2	1
4	0	2	3
4	1	0	3
4	1	2	0

$p=$

Floyd–Warshall

- ▶ Algoritmul funcționează corect chiar dacă arcele au și costuri negative (dar graful nu are circuite negative)



- ▶ Cum putem detecta pe parcursul algoritmului existența unui circuit negativ (\Rightarrow datele de intrare nu sunt corecte) ?

Aplicație
Închiderea tranzitivă a unui
graf orientat
Algoritmul Roy–Warhsall

Roy-Warshall

- ▶ **Aplicație:** Închiderea tranzitivă a unui graf orientat $G=(V, E)$ (**!!!neponderat**):

$$G^* = (V, E^*), \text{ unde}$$

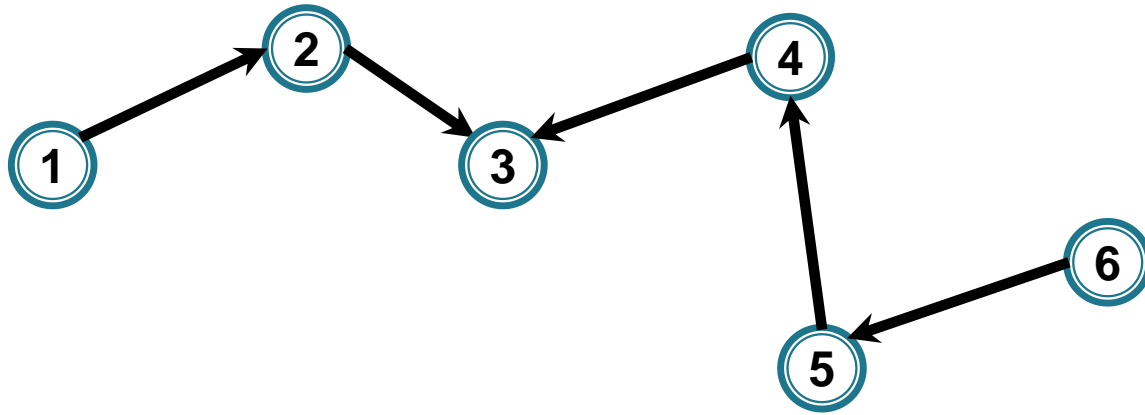
$$E^* = \{(i, j) \mid \text{există drum (de lungime minim 1) de la } i \text{ la } j \text{ în } G\}$$

- **Utilitate:**

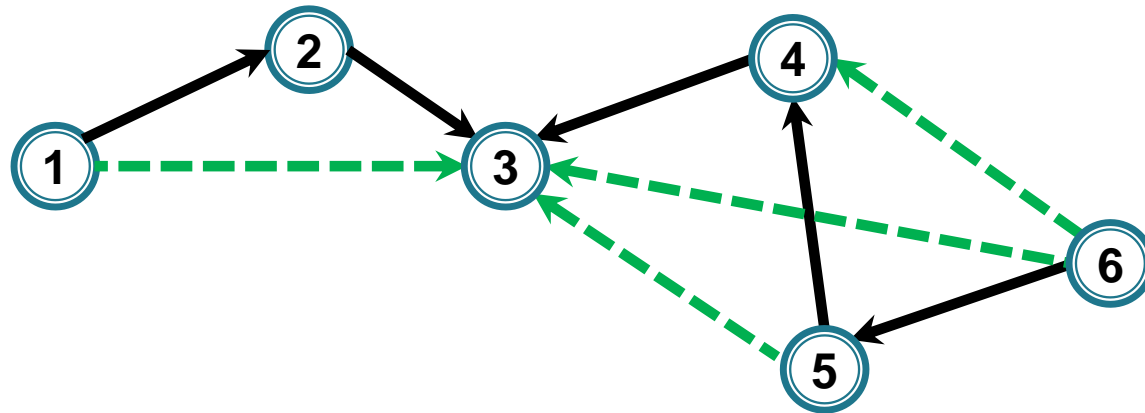
- grupări de obiecte aflate în relație (directă sau indirectă): optimizări în baze de date, analize în rețele, logică

Roy-Warshall

Exemplul 1

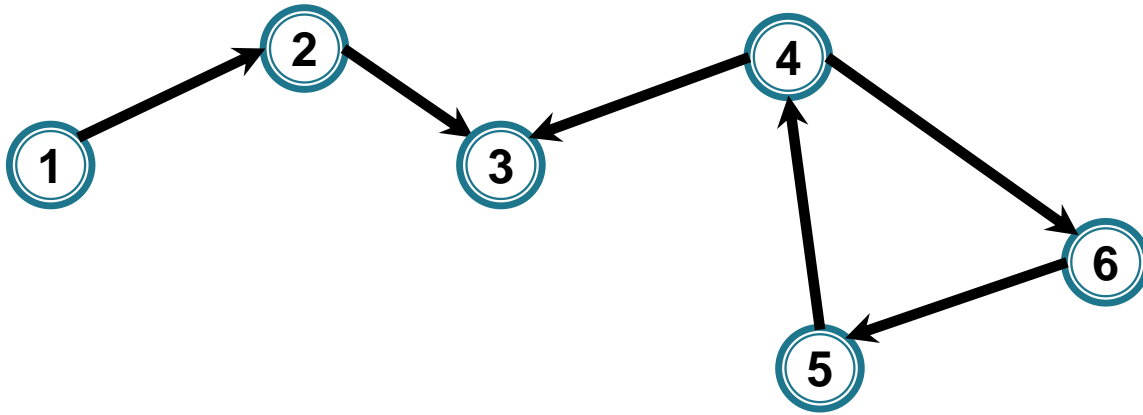


Închiderea tranzitivă

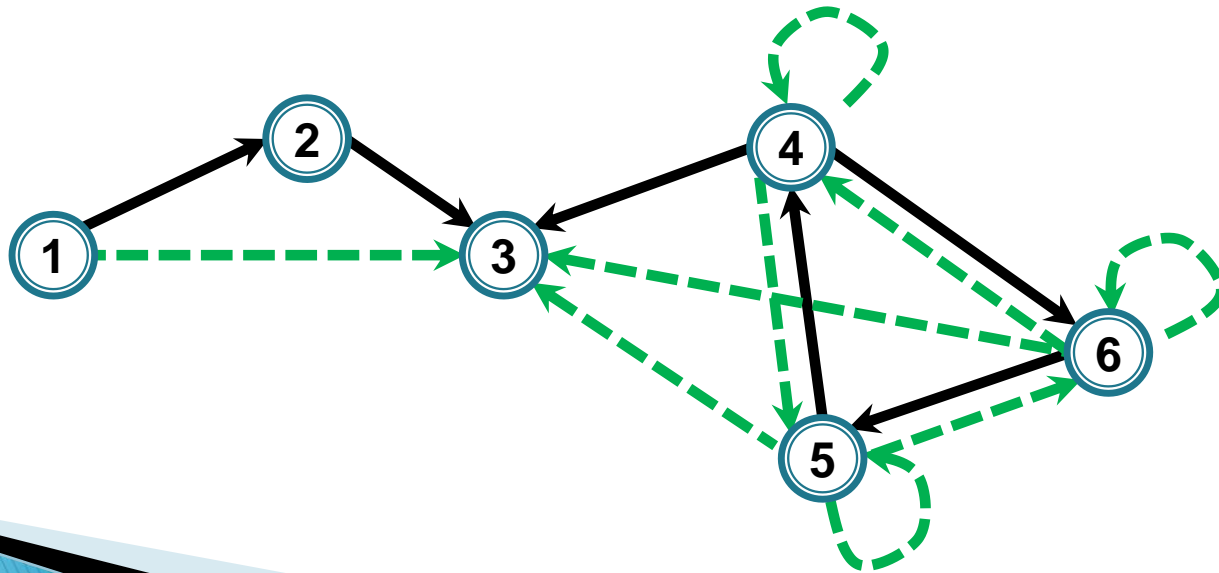


Roy-Warshall

Exemplul 2



Închiderea tranzitivă



Roy-Warshall

- ▶ Închiderea tranzitivă \Leftrightarrow calculăm matricea existenței drumurilor (matricea de adiacență a închiderii tranzitive)

$$D = (d_{ij})_{i,j=1..n} :$$

$d_{ij} = 1$, dacă există drum nevid de la i la j
0, altfel

Roy-Warshall

```
initial d = matricea de adiacenta
```

```
for (k=1 ; k<=n ; k++)
```

```
    for (i=1 ; i<=n ; i++)
```

```
        for (j=1 ; j<=n ; j++)
```

```
            d[i][j] = d[i][j]  $\vee$  (d[i][k]  $\wedge$  d[k][j]) ;
```

SAU

SI

Roy-Warshall

► Observație

Dacă A este matricea de adiacență a unui graf și

$A^k = (a^k_{ij})_{i,j=1..n}$: **puterea k a matricei** ($k < n$)

atunci a^k_{ij} = **numărul de drumuri distincte de lungime k de la i la j (!nu neapărat elementare)**

Demonstrație – Inducție. Temă (suplimentar)

Roy-Warshall

► Observație

Dacă A este matricea de adiacență a unui graf și

$$A^k = (a^k_{ij})_{i,j=1..n} : \text{puterea } k \text{ a matricei } (k < n)$$

atunci a^k_{ij} = numărul de drumuri distincte de lungime k de la i la j (!nu neapărat elementare)

◦ Consecință

$$D = A \vee A^2 \vee \dots \vee A^{n-1}$$

unde o valoare diferită de 0 se interpretează ca true