

Drumuri minime DAG-uri

Problema

- Se dau n activitati numerotate de la 1 la n .
- Aceste activitati nu se pot desfasura in mod independent unele de altele.
- Se dau perechi de forma (x,y) cu semnificatia ca activitatea x trebuie sa se fi desfasurat pentru ca activitatea y sa poata sa inceapa.

Problema

- Se dau n activitati numerotate de la 1 la n .
- Aceste activitati nu se pot desfasura in mod independent unele de altele.
- Sa se determine (daca este posibil) o ordine de desfasurare a acestor activitati care sa respecte regulile de dependenta.

Modelare



- **Q: Cum modelam aceasta problema?**
- **A: Grafuri orientate**

Modelare



- **Q: Mai exact?**

Modelare



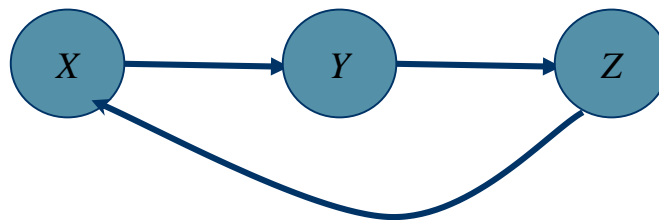
- **A: Grafuri orientate;**
- **A: Daca doua evenimente, x si y , sunt direct dependente: x trebuie sa se desfasoare inaintea lui y , atunci**



Modelare



- Q: In ce conditii exista solutii?
- A: Daca trei evenimente: x , y si z cu proprietatea



Acestea nu pot fi programate!

Mai exact, **nu trebuie sa existe circuite!**

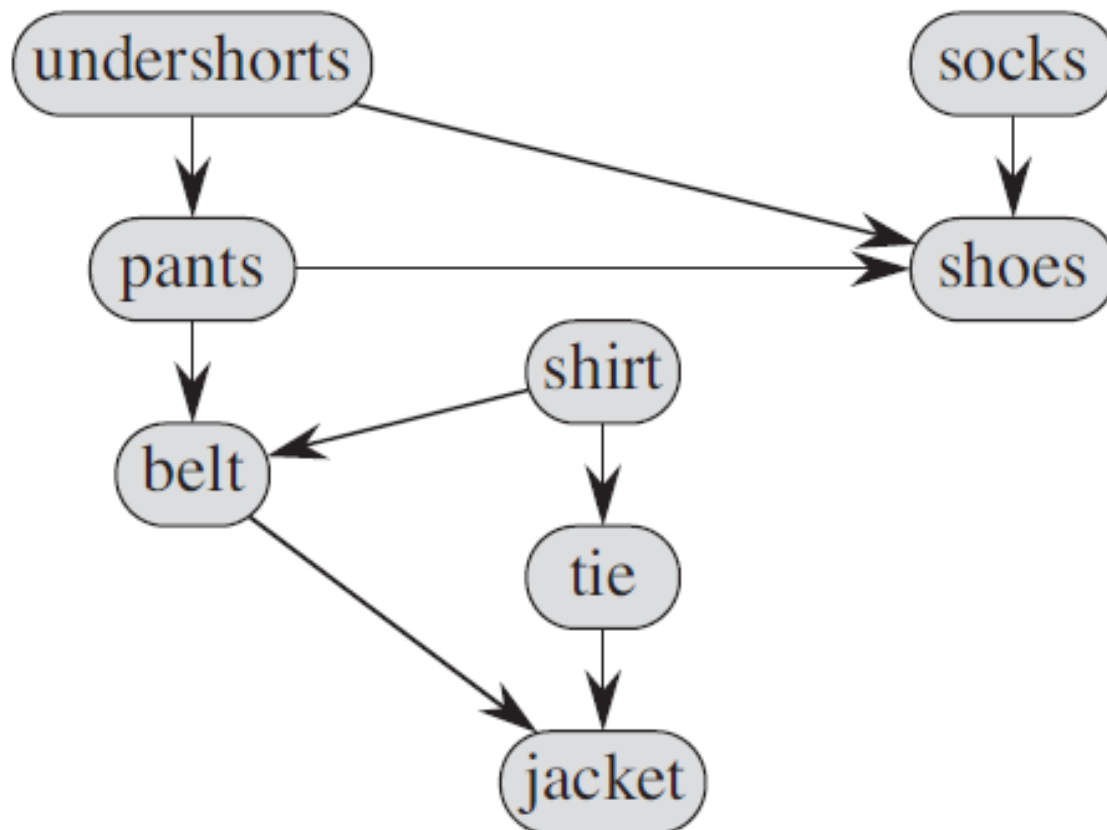
DAG = Directed Acyclic Graph

Modelare

- **Exemplu:** Ordinea in care ne imbracam inainte de plecare

Modelare - Exemplu

Ordinea in care ne imbracam inainte de plecare - Solutii?



Sortare Topologica

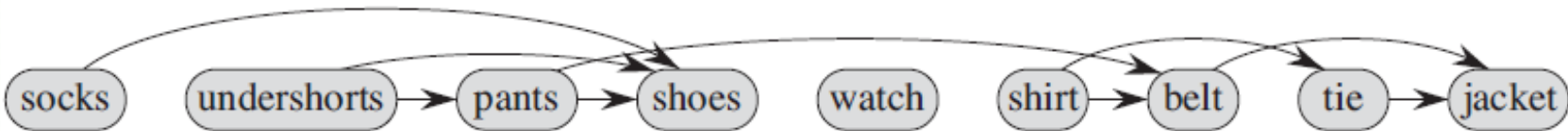
Ordinea in care ne imbracam inainte de plecare



Rezultatul...

Sortare Topologica

Ordinea in care ne imbracam inainte de plecare



Algoritmul

TOPOLOGICAL-SORT ($G=(V,E)$)



COMPLEXITATE?

1. Top-Sort=NULL
2. Cat timp am noduri cu gradul incident = 0
 - a. Identificam P - multimea nodurilor cu gradul incident = 0
 - b. Top-Sort.right_append(P)
 - c. $V=V\setminus P$
 - d. Reactualizez gradele nodurilor ramase in V
3. Daca V nu este NULL
 - a. Afisez "Sortarea nu se poate face"
4. altfel
 - a. Afisez Top-Sort;

Algoritmul

TOPOLOGICAL-SORT ($G=(V,E)$)



$O(|V|+|E|)$

1. Top-Sort=NULL
2. Cat timp am noduri cu gradul incident = 0
 - a. Identificam P - multimea nodurilor cu gradul incident = 0
 - b. Top-Sort.right_append(P)
 - c. $V=V\setminus P$
 - d. Reactualizez gradele nodurilor ramase in V
3. Daca V nu este NULL
 - a. Afisez "Sortarea nu se poate face"
4. altfel
 - a. Afisez Top-Sort;

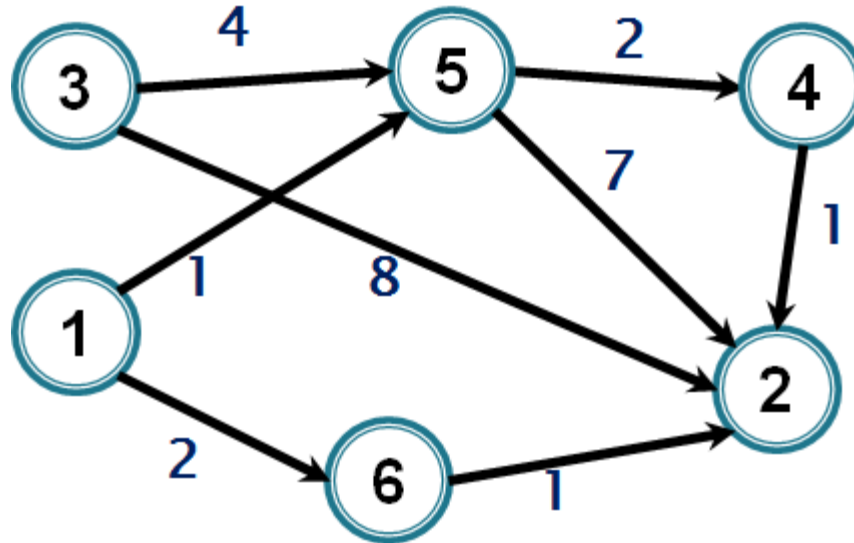
Algoritmul - Corectitudine

TOPOLOGICAL-SORT ($G=(V,E)$)

1. De ce atunci cand exista solutii, solutia furnizata de algoritm este corecta?
2. De ce atunci cand nu exista solutii (exista un circuit), algoritmul semnaleaza corect acest lucru?

Drumuri minime de sursa unica in DAG-uri

Dat la intrare un DAG - cu ponderi pe arce - si un nod de start s , sa se determine drumurile de cost minim de la s la toate celelalte noduri.



Drumuri minime de sursa unica in DAG-uri

Dat la intrare un DAG - cu ponderi pe arce - si un nod de start s , sa se determine drumurile de cost minim de la s la toate celelalte noduri.

Observatie 1: Arcele pot avea si cost negativ.

Observatie 2: Cand consideram un varf v , pentru a calcula $d(s,v)$ ar fi util sa stim $d(s,u)$ pentru orice u,v - arc;



Idee: Sortarea Topologica



Pseudocod

- Considerăm vârfurile în ordinea dată de sortarea topologică, începând cu vârful s
- Pentru fiecare vârf u relaxăm arcele uv către vecinii săi (pentru a găsi drumuri noi către aceștia)

Pseudocod

s - vârful de start

//initializam distante - ca la Dijkstra

pentru fiecare $u \in V$ executa

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

$d[s] = 0$

//determinăm o sortare topologică a vârfurilor

//este suficient sa pastrăm vârfurile din sortare începând
cu s

SortTop = Topological-Sort(G)

pentru fiecare $u \in \text{SortTop}$

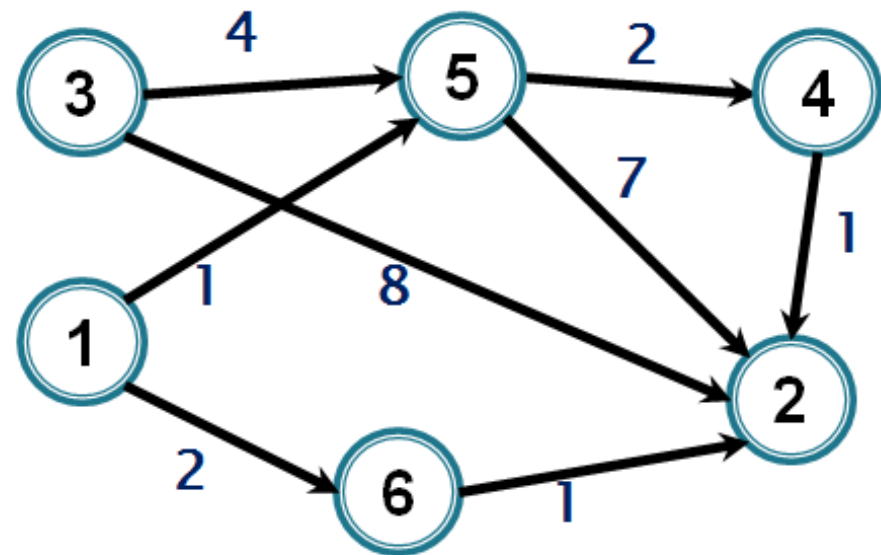
pentru fiecare $uv \in E$ executa

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

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

tata[v] = u

EXEMPLU



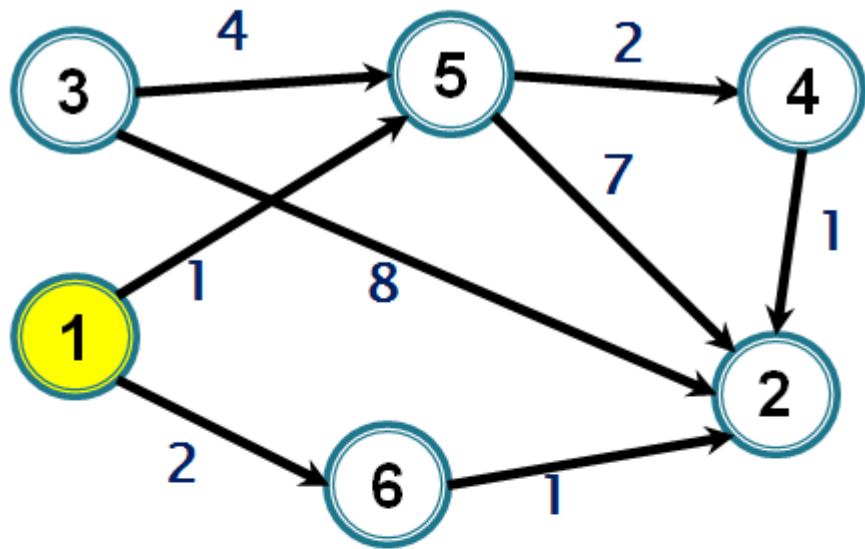
Sortare topologică

1, 3, 6, 5, 4, 2

s=3 - vârf de start

Ordine de calcul distanțe:

1, 3, 6, 5, 4, 2



Sortare topologică

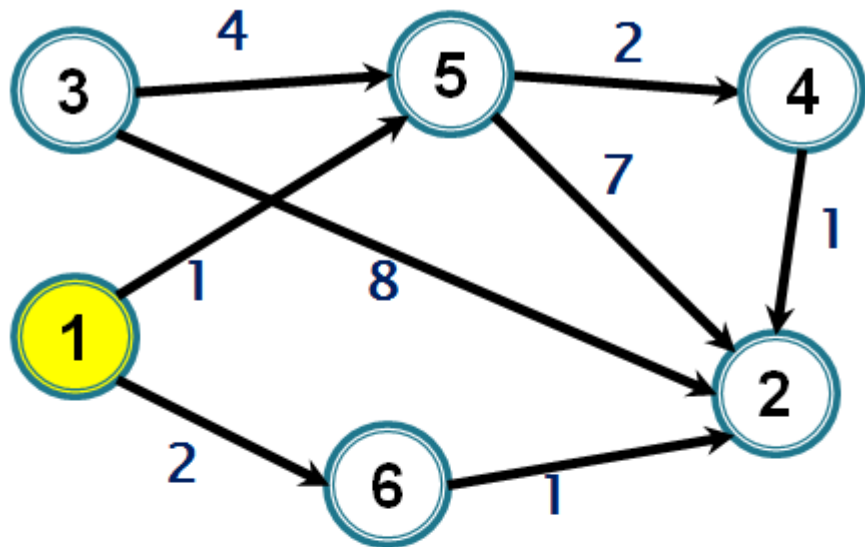
1, 3, 6, 5, 4, 2

s=3 – vârf de start

Ordine de calcul distanțe:

1, 3, 6, 5, 4, 2

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



Sortare topologică

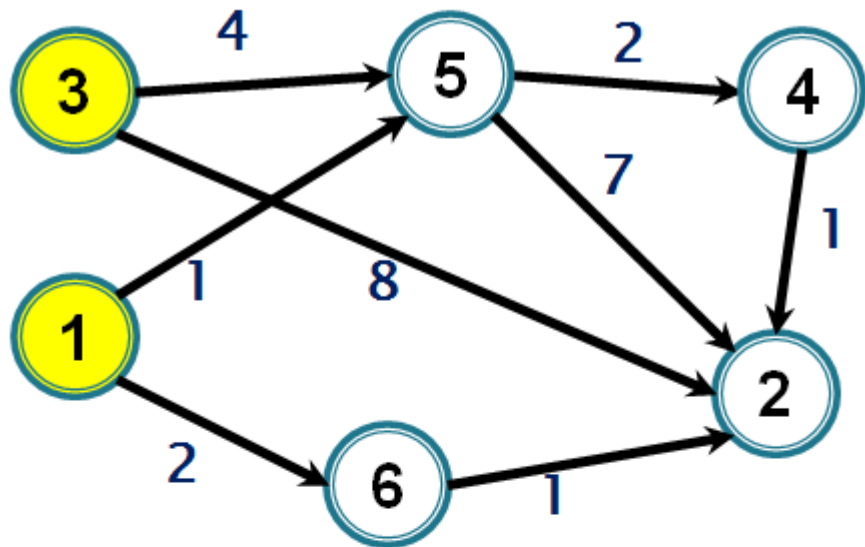
1, 3, 6, 5, 4, 2

$s=3$ – vârf de start

Ordine de calcul distanțe:

1, 3, 6, 5, 4, 2

d/tata	1	2	3	4	5	6
	$\infty/0,$	$\infty/0,$	$0/0,$	$\infty/0,$	$\infty/0,$	$\infty/0$
$u = 1:$	$\infty/0,$	$\infty/0,$	$0/0,$	$\infty/0,$	$\infty/0,$	$\infty/0$



Sortare topologică

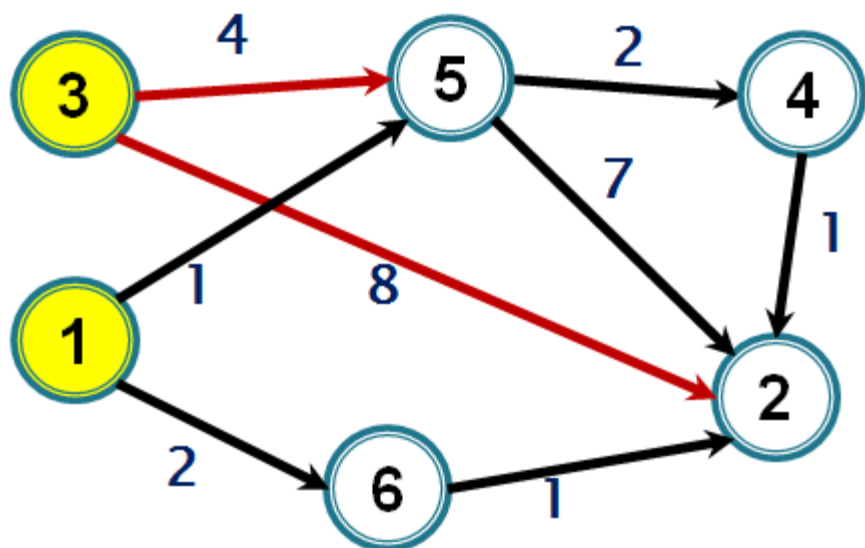
1, 3, 6, 5, 4, 2

$s=3$ – vârf de start

Ordine de calcul distanțe:

1, 3, 6, 5, 4, 2

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



Sortare topologică

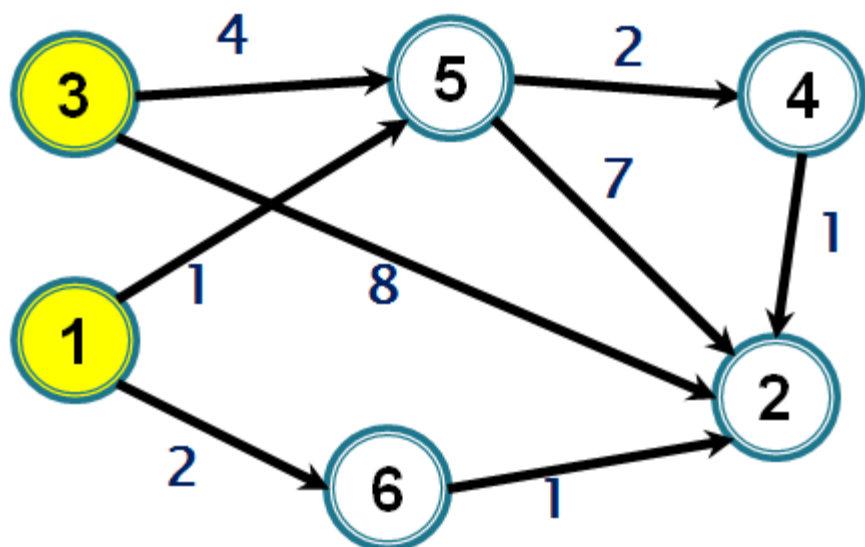
1, 3, 6, 5, 4, 2

$s=3$ – vârf de start

Ordine de calcul distanțe:

1, 3, 6, 5, 4, 2

d/tata	1	2	3	4	5	6
	$\infty/0,$	$\infty/0,$	$0/0,$	$\infty/0,$	$\infty/0,$	$\infty/0$
$u = 1:$	$\infty/0,$	$\infty/0,$	$0/0,$	$\infty/0,$	$\infty/0,$	$\infty/0$
$u = 3:$						



Sortare topologică

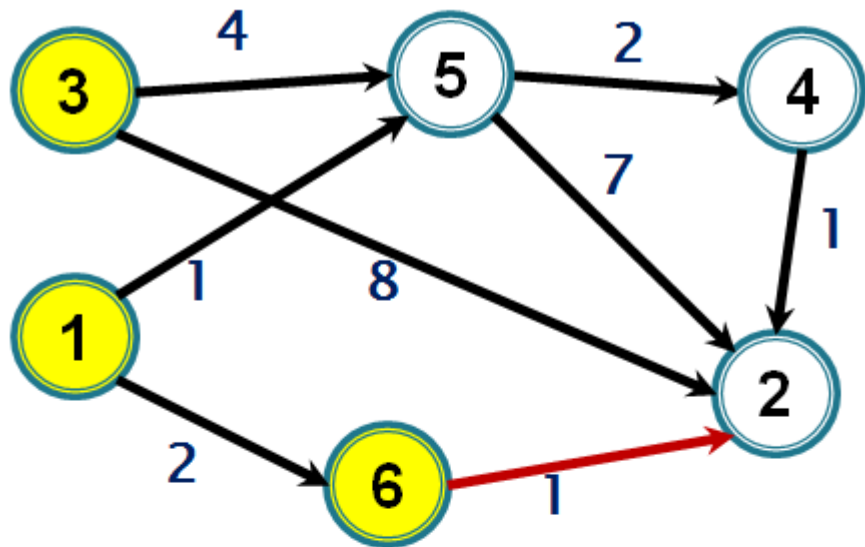
1, 3, 6, 5, 4, 2

$s=3$ – vârf de start

Ordine de calcul distanțe:

1, 3, 6, 5, 4, 2

d/tata	1	2	3	4	5	6
	$\infty/0,$	$\infty/0,$	$0/0,$	$\infty/0,$	$\infty/0,$	$\infty/0$
$u = 1:$	$\infty/0,$	$\infty/0,$	$0/0,$	$\infty/0,$	$\infty/0,$	$\infty/0$
$u = 3:$	$\infty/0,$	8/3,	$0/0,$	$\infty/0,$	4/3,	$\infty/0$



Sortare topologică

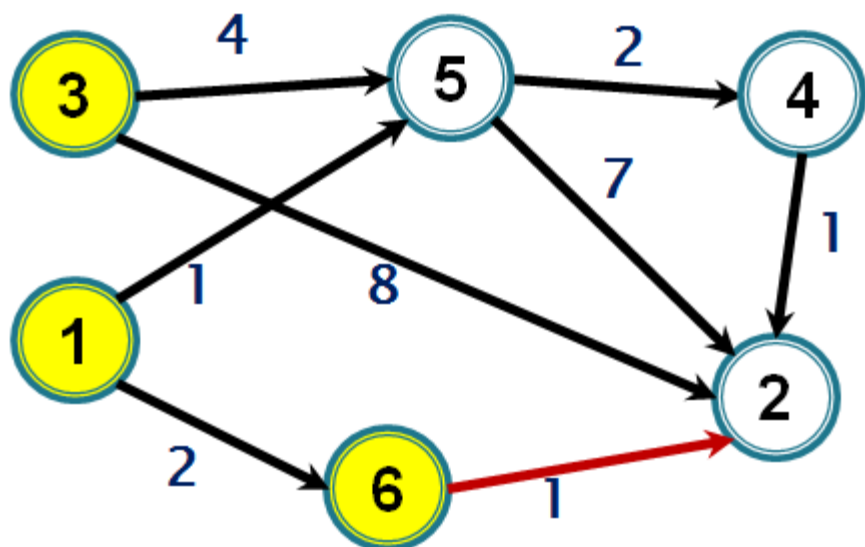
1, 3, 6, 5, 4, 2

$s=3$ – vârf de start

Ordine de calcul distanțe:

1, 3, 6, 5, 4, 2

d/tata	¹ [$\infty/0$,	² [$\infty/0$,	³ [$0/0$,	⁴ [$\infty/0$,	⁵ [$\infty/0$,	⁶ [$\infty/0$]
$u = 1$:	[$\infty/0$,	[$\infty/0$,	[$0/0$,	[$\infty/0$,	[$\infty/0$,	[$\infty/0$]
$u = 3$:	[$\infty/0$,	[$8/3$,	[$0/0$,	[$\infty/0$,	[$4/3$,	[$\infty/0$]
$u = 6$:						



Sortare topologică

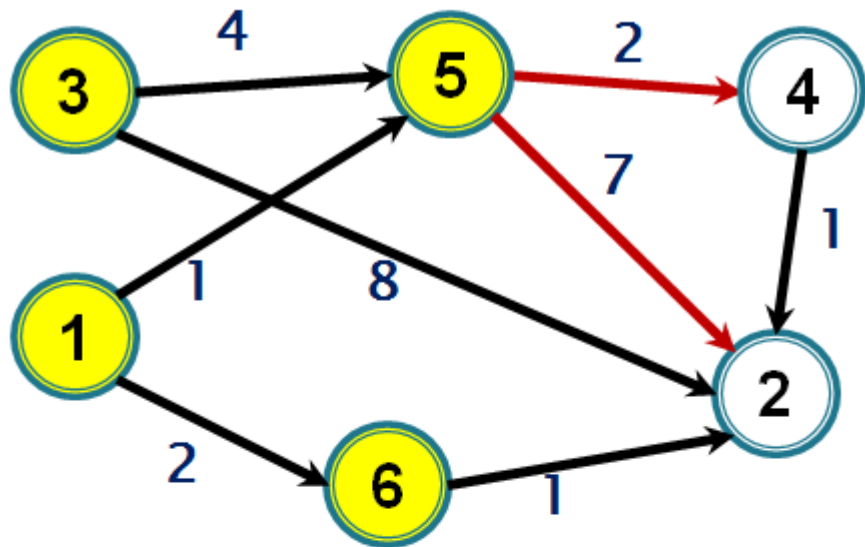
1, 3, 6, 5, 4, 2

$s=3$ – vârf de start

Ordine de calcul distanțe:

1, 3, 6, 5, 4, 2

d/tata	1	2	3	4	5	6
	$\infty/0,$	$\infty/0,$	$0/0,$	$\infty/0,$	$\infty/0,$	$\infty/0$
$u = 1:$	$\infty/0,$	$\infty/0,$	$0/0,$	$\infty/0,$	$\infty/0,$	$\infty/0$
$u = 3:$	$\infty/0,$	8/3,	0/0,	$\infty/0,$	4/3,	$\infty/0$
$u = 6:$	$\infty/0,$	8/3,	0/0,	$\infty/0,$	4/3,	$\infty/0$



Sortare topologică

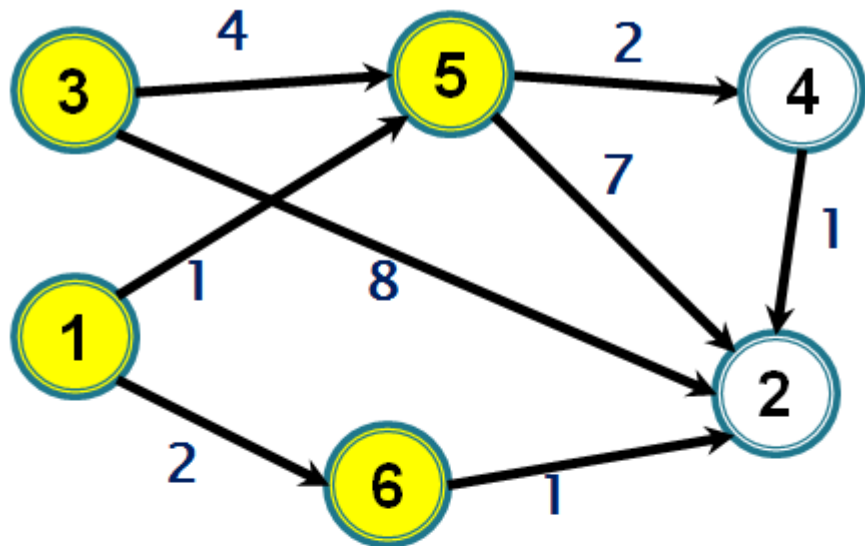
1, 3, 6, 5, 4, 2

$s=3$ – vârf de start

Ordine de calcul distanțe:

1, 3, 6, 5, 4, 2

d/tata	¹ [$\infty/0$,	² [$\infty/0$,	³ [$0/0$,	⁴ [$\infty/0$,	⁵ [$\infty/0$,	⁶ [$\infty/0$]
u = 1:	[$\infty/0$,	[$\infty/0$,	[$0/0$,	[$\infty/0$,	[$\infty/0$,	[$\infty/0$]
u = 3:	[$\infty/0$,	[$8/3$,	[$0/0$,	[$\infty/0$,	[$4/3$,	[$\infty/0$]
u = 6:	[$\infty/0$,	[$8/3$,	[$0/0$,	[$\infty/0$,	[$4/3$,	[$\infty/0$]
u = 5:						



Sortare topologică

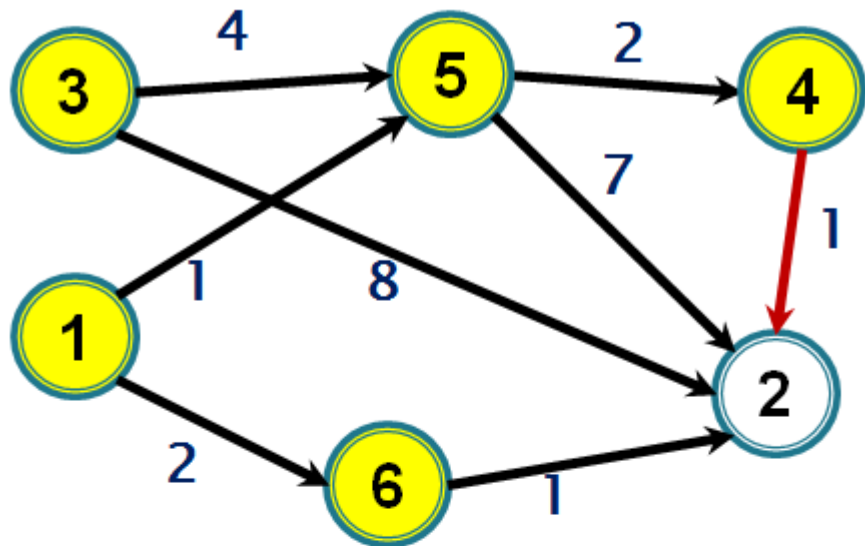
1, 3, 6, 5, 4, 2

$s=3$ – vârf de start

Ordine de calcul distanțe:

1, 3, 6, 5, 4, 2

d/tata	1	2	3	4	5	6
	$\infty/0,$	$\infty/0,$	$0/0,$	$\infty/0,$	$\infty/0,$	$\infty/0$
$u = 1:$	$\infty/0,$	$\infty/0,$	$0/0,$	$\infty/0,$	$\infty/0,$	$\infty/0$
$u = 3:$	$\infty/0,$	$8/3,$	$0/0,$	$\infty/0,$	$4/3,$	$\infty/0$
$u = 6:$	$\infty/0,$	$8/3,$	$0/0,$	$\infty/0,$	$4/3,$	$\infty/0$
$u = 5:$	$\infty/0,$	$8/3,$	$0/0,$	$6/5,$	$4/3,$	$\infty/0$



Sortare topologică

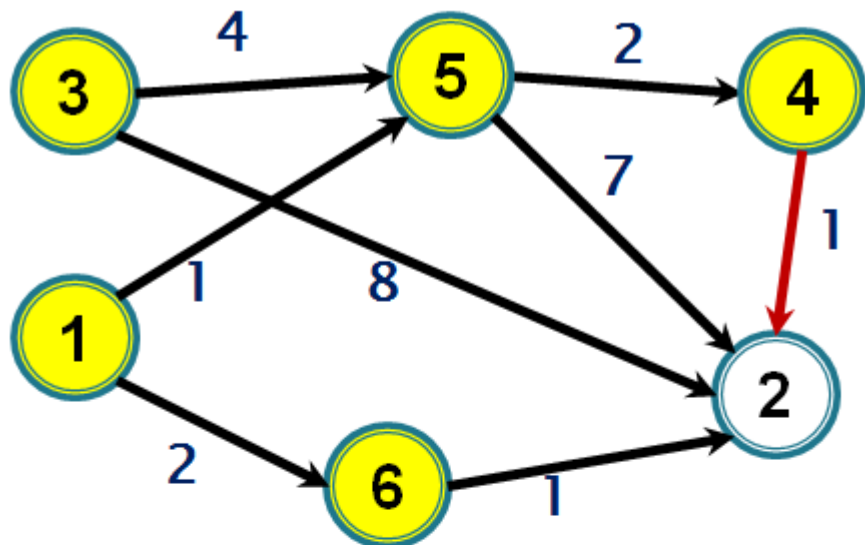
1, 3, 6, 5, 4, 2

$s=3$ – vârf de start

Ordine de calcul distanțe:

1, 3, 6, 5, 4, 2

d/tata	¹ [$\infty/0$,	² [$\infty/0$,	³ [$0/0$,	⁴ [$\infty/0$,	⁵ [$\infty/0$,	⁶ [$\infty/0$]
u = 1:	[$\infty/0$,	[$\infty/0$,	[$0/0$,	[$\infty/0$,	[$\infty/0$,	[$\infty/0$]
u = 3:	[$\infty/0$,	[$8/3$,	[$0/0$,	[$\infty/0$,	[$4/3$,	[$\infty/0$]
u = 6:	[$\infty/0$,	[$8/3$,	[$0/0$,	[$\infty/0$,	[$4/3$,	[$\infty/0$]
u = 5:	[$\infty/0$,	[$8/3$,	[$0/0$,	[$6/5$,	[$4/3$,	[$\infty/0$]
u = 4:						



Sortare topologică

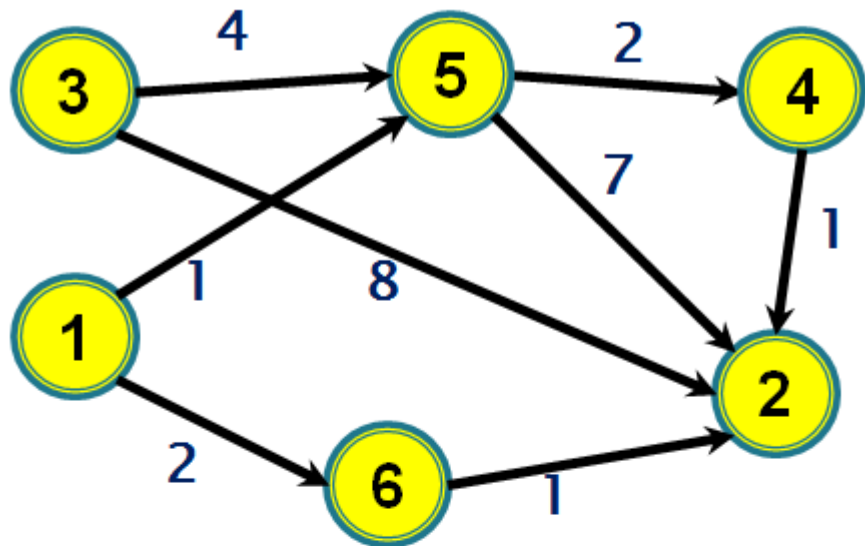
1, 3, 6, 5, 4, 2

$s=3$ – vârf de start

Ordine de calcul distanțe:

1, 3, 6, 5, 4, 2

d/tata	1	2	3	4	5	6
$u = 1:$	$\infty/0$	$\infty/0$	$0/0$	$\infty/0$	$\infty/0$	$\infty/0$
$u = 3:$	$\infty/0$	$8/3$	$0/0$	$\infty/0$	$4/3$	$\infty/0$
$u = 6:$	$\infty/0$	$8/3$	$0/0$	$\infty/0$	$4/3$	$\infty/0$
$u = 5:$	$\infty/0$	$8/3$	$0/0$	$6/5$	$4/3$	$\infty/0$
$u = 4:$	$\infty/0$	$7/4$	$0/0$	$6/5$	$4/3$	$\infty/0$



Sortare topologică

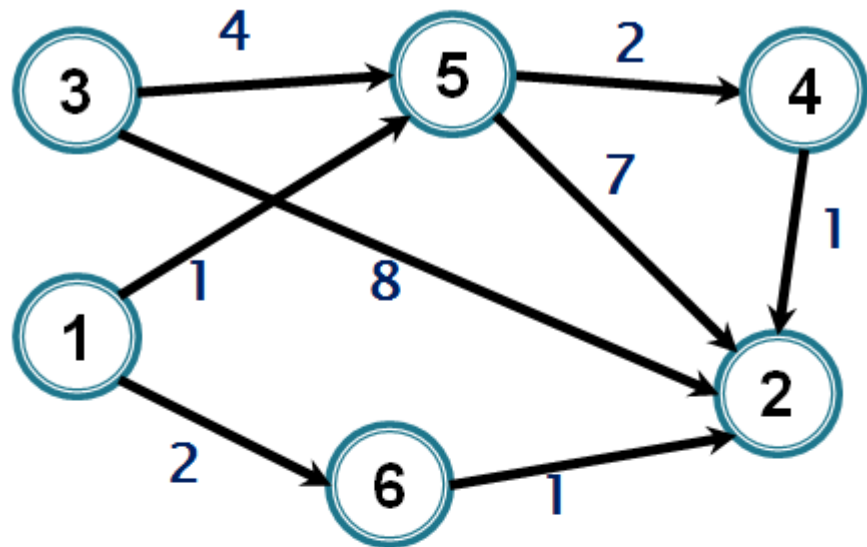
1, 3, 6, 5, 4, 2

$s=3$ – vârf de start

Ordine de calcul distanțe:

1, 3, 6, 5, 4, 2

d/tata	¹ [$\infty/0$,	² [$\infty/0$,	³ [$0/0$,	⁴ [$\infty/0$,	⁵ [$\infty/0$,	⁶ [$\infty/0$]
u = 1:	[$\infty/0$,	$\infty/0$,	$0/0$,	$\infty/0$,	$\infty/0$,	$\infty/0$]
u = 3:	[$\infty/0$,	8/3,	$0/0$,	$\infty/0$,	4/3,	$\infty/0$]
u = 6:	[$\infty/0$,	8/3,	$0/0$,	$\infty/0$,	4/3,	$\infty/0$]
u = 5:	[$\infty/0$,	8/3,	$0/0$,	6/5,	4/3,	$\infty/0$]
u = 4:	[$\infty/0$,	7/4,	$0/0$,	6/5,	4/3,	$\infty/0$]
u = 2:						



Sortare topologică

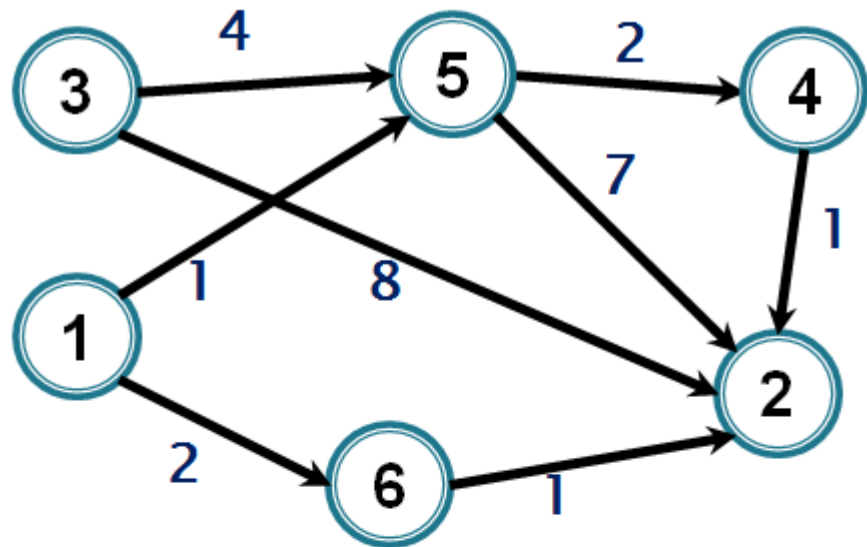
1, 3, 6, 5, 4, 2

$s=3$ – vârf de start

Ordine de calcul distanțe:

1, 3, 6, 5, 4, 2

d/tata	1	2	3	4	5	6
	$\infty/0$,	$\infty/0$,	$0/0$,	$\infty/0$,	$\infty/0$,	$\infty/0$
$u = 1$:	$\infty/0$,	$\infty/0$,	$0/0$,	$\infty/0$,	$\infty/0$,	$\infty/0$
$u = 3$:	$\infty/0$,	$8/3$,	$0/0$,	$\infty/0$,	$4/3$,	$\infty/0$
$u = 6$:	$\infty/0$,	$8/3$,	$0/0$,	$\infty/0$,	$4/3$,	$\infty/0$
$u = 5$:	$\infty/0$,	$8/3$,	$0/0$,	$6/5$,	$4/3$,	$\infty/0$
$u = 4$:	$\infty/0$,	$7/4$,	$0/0$,	$6/5$,	$4/3$,	$\infty/0$
$u = 2$:	$\infty/0$,	$7/4$,	$0/0$,	$6/5$,	$4/3$,	$\infty/0$



Sortare topologică

1, 3, 6, 5, 4, 2

$s=3$ – vârf de start

Ordine de calcul distanțe:

1, 3, 6, 5, 4, 2

d/tata

1

2

3

4

5

6

Soluție [$\infty/0$, 7/4, 0/0, 6/5, 4/3, $\infty/0$]

Un drum minim de la 3 la 2?

Corectitudine

Observatie 1:

Toate nodurile situate la “stanga” nodului de start in sortarea topologica vor avea distanta catre ele ∞

Observatie 2:

Pentru orice alt nod, corectitudinea rezultatului obtinut se bazeaza pe corectitudinea rezultatului obtinut pentru nodurile anterioare in sortarea topologica

Aplicatie: Drumuri Critice

Se cunosc pentru un proiect cu n activități, numerotate $1, \dots, n$:

- **durata** fiecărei activități
- **perechi (i, j)** = activitatea i trebuie să se încheie **înainte** să înceapă j
- activitățile se pot desfășura și **în paralel**

Se cere: **timpul minim de finalizare a proiectului** (dacă momentul de start este ora 0) + **planificarea activităților**

Drumuri Critice: Exemplu

$n = 6$

- **Activitatea 1 - durata 7**
- **Activitatea 2 - durata 4**
- **Activitatea 3 - durata 30**
- **Activitatea 4 - durata 12**
- **Activitatea 5 - durata 2**
- **Activitatea 6 - durata 5**
- **(1, 2)**
- **(2, 3)**
- **(3, 6)**
- **(4, 3)**
- **(2, 6)**
- **(3, 5)**

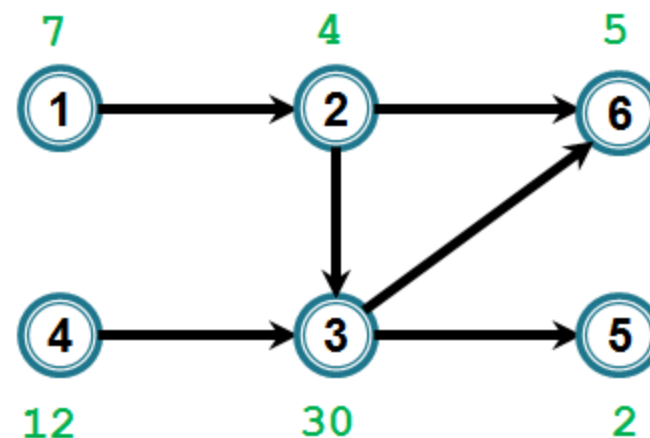


Modelare?

Drumuri Critice: Exemplu

$n = 6$

- Activitatea 1 - durata 7
- Activitatea 2 - durata 4
- Activitatea 3 - durata 30
- Activitatea 4 - durata 12
- Activitatea 5 - durata 2
- Activitatea 6 - durata 5
- (1, 2)
- (2, 3)
- (3, 6)
- (4, 3)
- (2, 6)
- (3, 5)



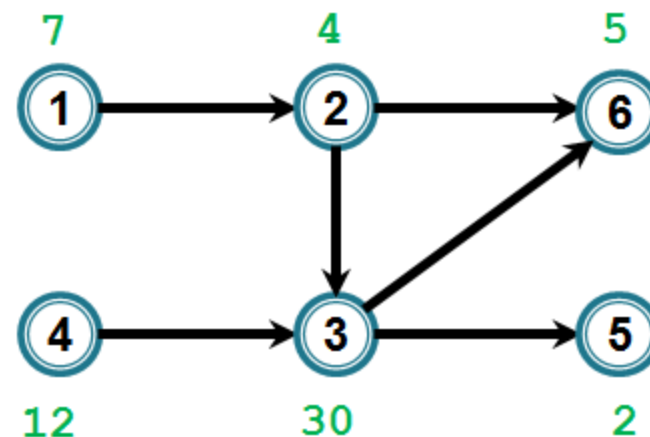
Drumuri Critice: Exemplu

$n = 6$

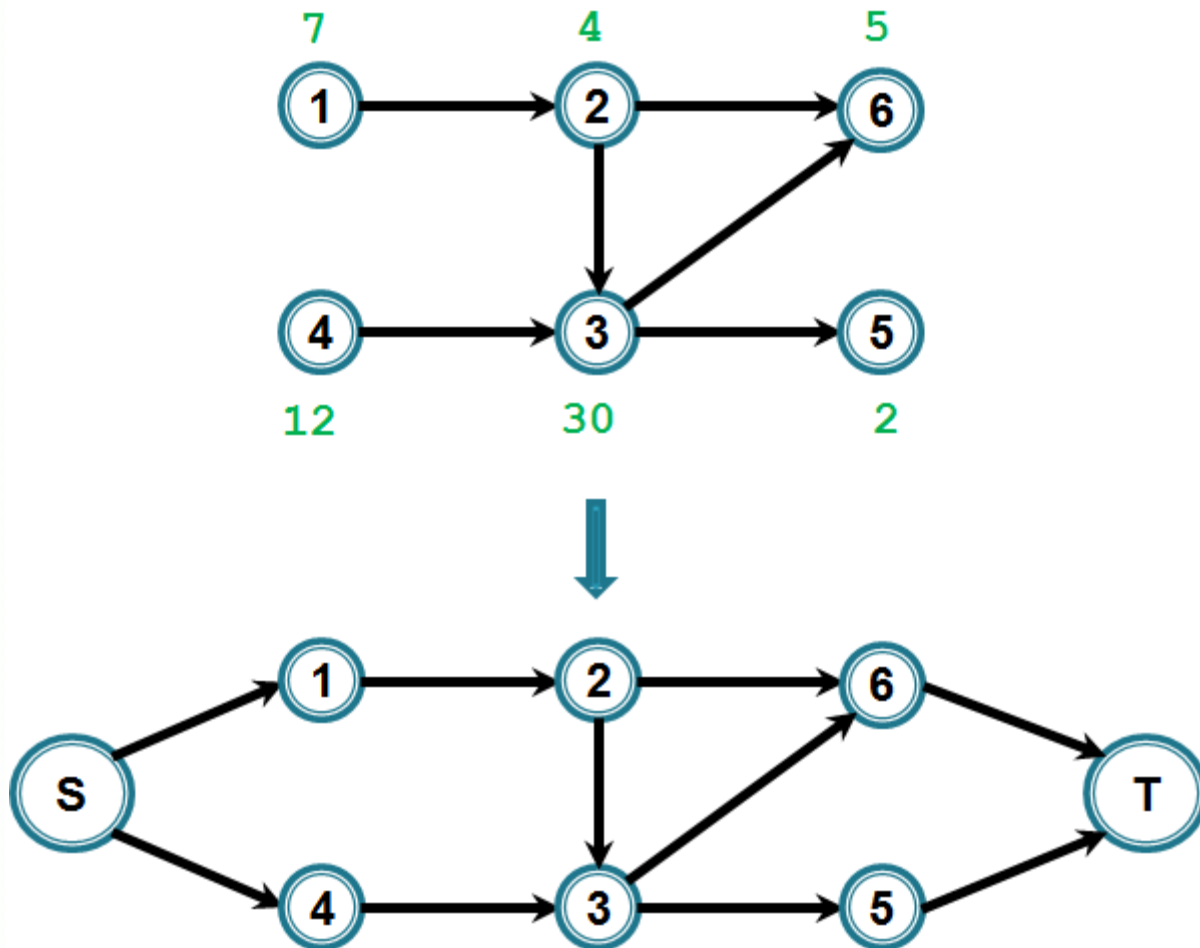
- Activitatea 1 - durata 7
- Activitatea 2 - durata 4
- Activitatea 3 - durata 30
- Activitatea 4 - durata 12
- Activitatea 5 - durata 2
- Activitatea 6 - durata 5
- (1, 2)
- (2, 3)
- (3, 6)
- (4, 3)
- (2, 6)
- (3, 5)



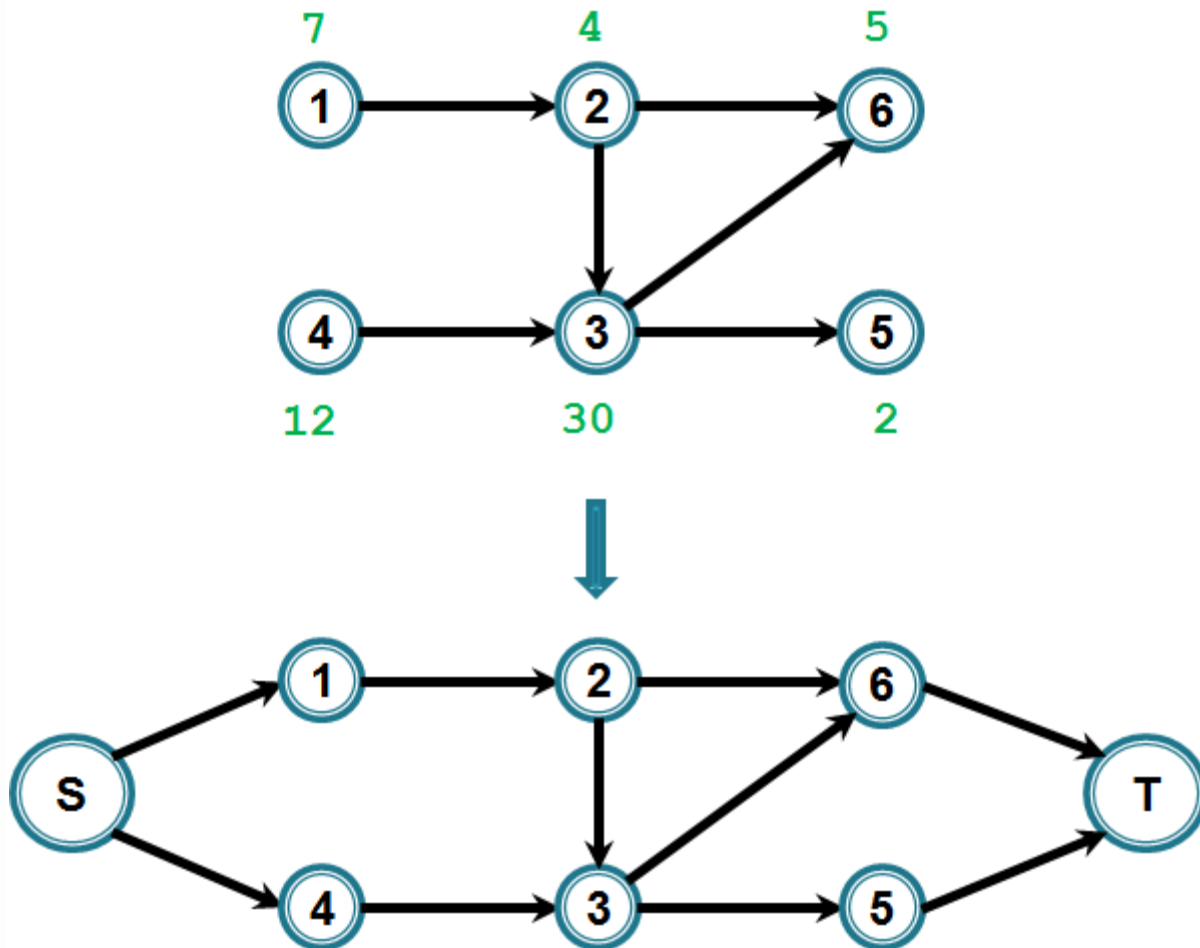
Ponderile?



Drumuri Critice: Exemplu

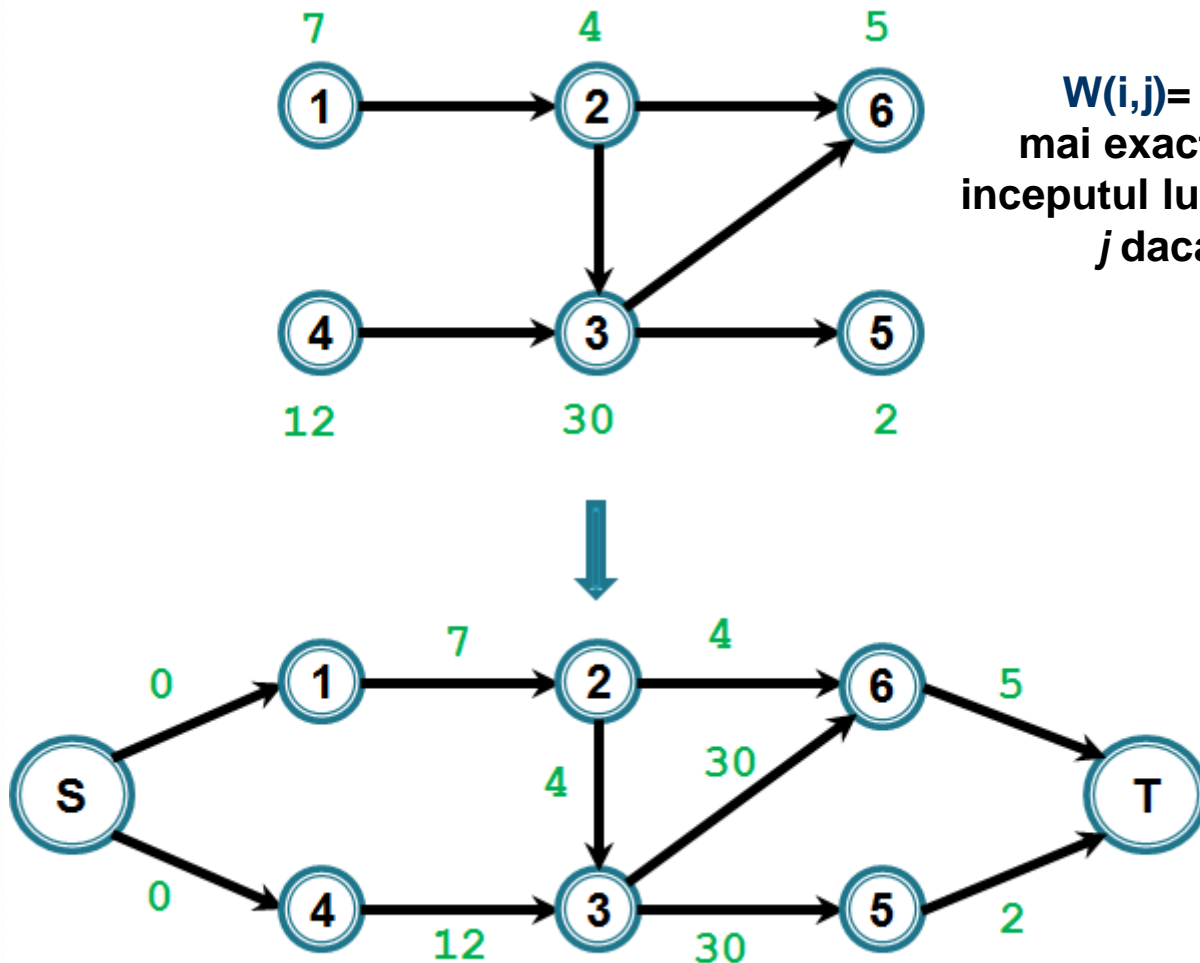


Drumuri Critice: Exemplu



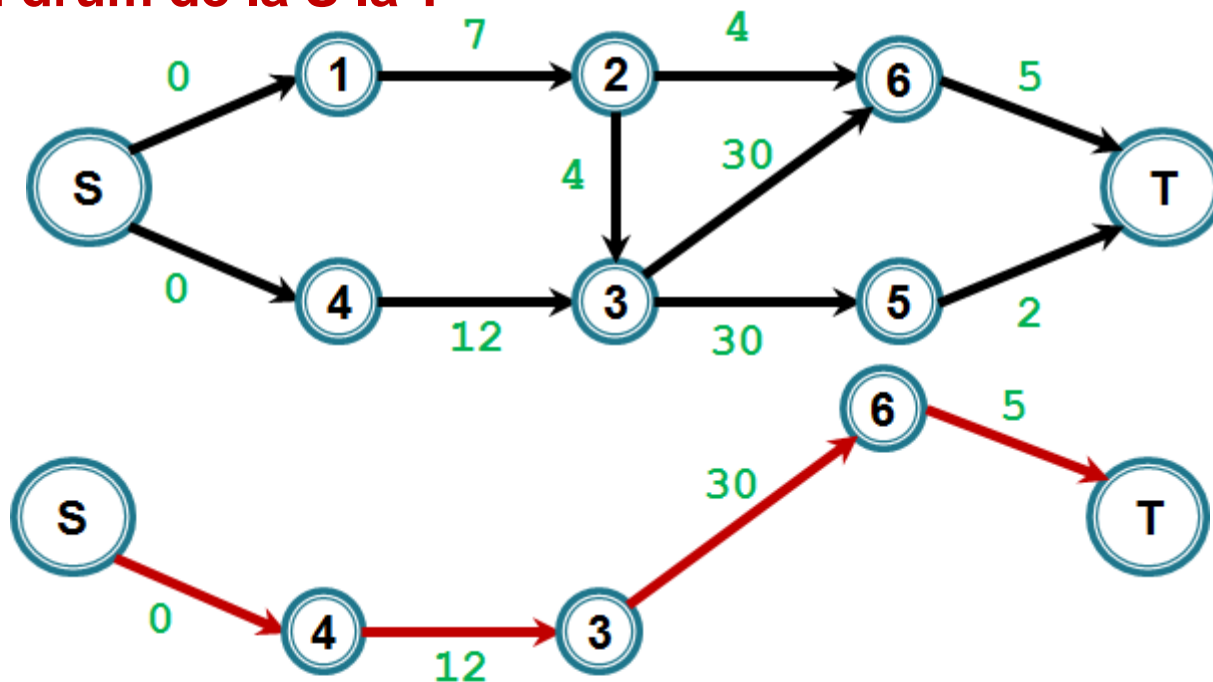
$W(i,j)=?$

Drumuri Critice: Exemplu



Drumuri Critice: Exemplu

Timpul minim de finalizare a proiectului = **costul maxim al unui drum de la S la T**



Drum CRITIC

Analiza...



Putem modifica algoritmul de determinare de drumuri minime în grafuri aciclice a.î. să determine drumuri maxime (de cost maxim) de la S la celelalte vârfuri?

Analiza...



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

Drumuri minime cu mai multe puncte de start

Problema

Dandu-se un graf (preferabil fara circuite de cost negativ), se pune problema gasirii in mod eficient a drumurilor de cost minimim de la oricare nod la oricare alt nod



Q: Cum retin costul drumurilor dintre i si j ?

A: Matricea $D[i][j]$ = costul drumului minim de la i la j

Problema

Dandu-se un graf (preferabil fara circuite de cost negativ), se pune problema gasirii in mod eficient a drumurilor de cost minimim de la oricare nod la oricare alt nod



Q: Cum retin efectiv drumul dintre i si j ?

A: Matricea $T[i][j]$ = Predecesorul nodului j in drumul de cost minim de la i la j

Solutia: Algoritmul Floyd-Warshall

$n = |V|$

//initializam distante si predecesori

pentru fiecare $i, j \in V$ executa

$D[i][i] = 0; D[i][j] = w[i][j];$

$T[i][i] = 0;$

$T[i][j] = i$ - daca $ij \in E;$

$T[i][j] = \text{NULL}$ - altfel;

Solutia: Algoritmul Floyd-Warshall

```
//actualizare distante si predecesori
```

```
    pentru k de la 1 la n
```

```
        pentru i de la 1 la n
```

```
            pentru j de la 1 la n
```

```
                 $D'[i][j] = \min(D[i][j], D[i][k] + D[k][j])$ 
```

```
                    daca  $D'[i][j] = D[i][j]$ 
```

```
                         $T'[i][j] = T[i][j]$ 
```

```
                    altfel
```

```
                         $T'[i][j] = T[k][j]$ 
```

```
                 $T = T'$  ;  $D = D'$ 
```



Complexitate?

Corectitudine

- Algoritmul incerca sa insereze in drumul minim contruit toate nodurile k ; un nod k este folosit in constructia unui drum de cost minim doar daca ajuta la reducerea costului.
- Ordinea in care sunt construite drumurile?
- Ce se intampla in cazul circuitelor de cost negativ?

The end

