

Sortare topologică

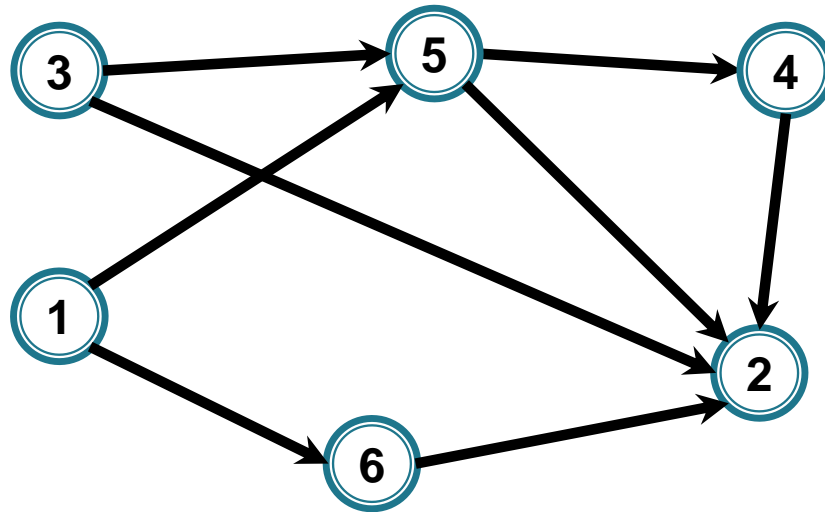


Sortare topologică

- ▶ Fie $G = (V, E)$ graf orientat
- ▶ Sortare topologică a lui G =
ordonare a vârfurilor astfel încât dacă $uv \in E$ atunci u
se află înaintea lui v în ordonare
 - **Nu este neapărat unică**
 - **!!! graf orientat**

Sortare topologică

- ▶ Fie $G = (V, E)$ graf orientat
- ▶ Sortare topologică a lui $G =$
ordonare a vârfurilor astfel încât dacă $uv \in E$ atunci u
se află înaintea lui v în ordonare

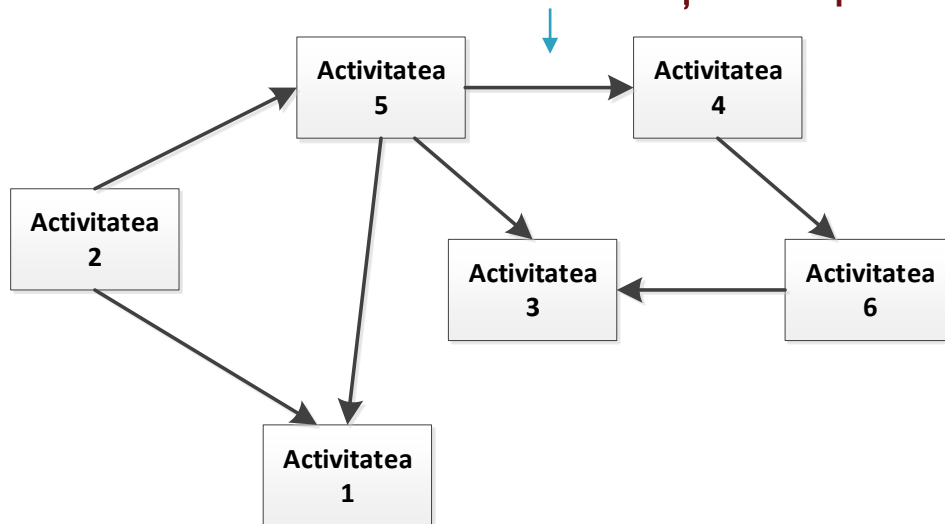


Sortare topologică

► Aplicații

- Ordinea de calcul în proiecte în care intervin relații de dependență / precedență (exp: calcul de formule, ordinea de compilare când clasele/pachetele depind unele de altele)
- Detecție de deadlock
- Determinarea de drumuri critice

Activitatea 4 depinde de 5, deci trebuie să se desfășoare după ea



În ce ordine trebuie executate activitățile?

	A	B	C	D
1		3	2	
2		3	6	0
3				
4		"=B1+D2"	"=2*B2"	"=2*C1+C2"

formulele din celulele B2..D2

În ce ordine se evaluează formulele?

Probleme – dacă există dependențe circulare

Sortare topologică

- ▶ Fie $G = (V, E)$ graf orientat
- ▶ Sortare topologică a lui $G =$
ordonare a vârfurilor astfel încât dacă $uv \in E$ atunci u
se află înaintea lui v în ordonare
- ▶ **Propoziție.** Dacă G este aciclic atunci G are o sortare topologică

Sortare topologică

- ▶ Fie $G = (V, E)$ graf orientat
- ▶ Sortare topologică a lui $G =$
ordonare a vârfurilor astfel încât dacă $uv \in E$ atunci u se află înaintea lui v în ordonare
- ▶ Propoziție. Dacă G este aciclic atunci G are o sortare topologică
 - Demonstrație \Rightarrow Algoritm?



Care vârf poate fi primul în sortarea topologică?

Sortare topologică – Algoritm

- ▶ Fie $G = (V, E)$ graf orientat
- ▶ **Lemă.** Dacă G este aciclic, atunci G are cel puțin un vârf v cu gradul intern 0 ($d^-(v) = 0$).

Sortare topologică – Algoritm

- ▶ Fie $G = (V, E)$ graf orientat
- ▶ **Lemă.** Dacă G este aciclic, atunci G are cel puțin un vârf v cu gradul intern 0 ($d^-(v) = 0$).

Demonstrație: considerăm un drum elementar maxim.
Extremitatea inițială a sa are grad intern 0

(v. si dem. Proprietății: un arbore cu $n > 2$ vârfuri are minim 2 vârfuri terminale)

Sortare topologică – Algoritm

- ▶ Fie $G = (V, E)$ graf orientat
- ▶ **Lemă.** Dacă G este aciclic, atunci G are cel puțin un vârf v cu gradul intern 0 ($d^-(v) = 0$).
- ▶ **Algoritm**

```
cât timp  $|V(G)| > 0$  execută  
    alege  $v$  cu  $d^-(v) = 0$   
    adauga  $v$  în ordonare  
     $G \leftarrow G - v$ 
```

- ▶ Corectitudinea – rezultă din Lemă + inducție

Pseudocod

Sortare topologică – Algoritm

► Algoritm

```
cât timp  $|V(G)| > 0$  execută  
    alege  $v$  cu  $d^-(v) = 0$   
    adauga  $v$  in ordonare  
     $G \leftarrow G - v$ 
```



Implementare? Complexitate?

Sortare topologică – Algoritm

▶ Algoritm

```
cât timp  $|V(G)| > 0$  execută  
    alege  $v$  cu  $d^-(v) = 0$   
    adauga  $v$  in ordonare  
     $G \leftarrow G - v$ 
```

▶ Implementare – similar BF

- Pornim cu toate vârfurile cu grad intern 0 și le adăugăm într-o coadă
-

Sortare topologică – Algoritm

▶ Algoritm

```
cât timp  $|V(G)| > 0$  execută  
    alege  $v$  cu  $d^-(v) = 0$   
    adauga  $v$  in ordonare  
     $G \leftarrow G - v$ 
```

▶ Implementare – similar BF

- Pornim cu toate vârfurile cu grad intern 0 și le adăugăm într-o coadă
- Repetăm:
 - extragem un vârf din coadă
 - îl eliminăm din graf (= scădem gradele interne ale vecinilor, nu îl eliminăm din reprezentare)
 -

Sortare topologică – Algoritm

▶ Algoritm

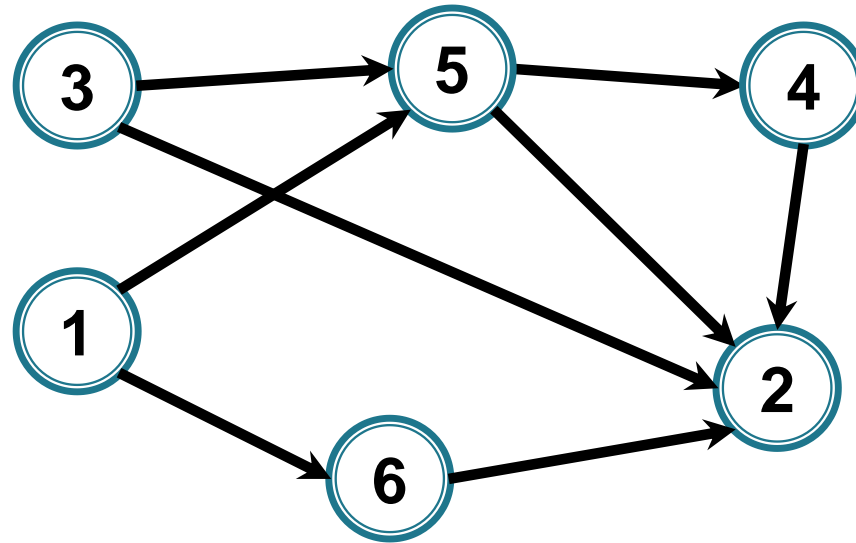
```
cât timp  $|V(G)| > 0$  execută  
    alege  $v$  cu  $d^-(v) = 0$   
    adauga  $v$  in ordonare  
     $G \leftarrow G - v$ 
```

▶ Implementare – similar BF

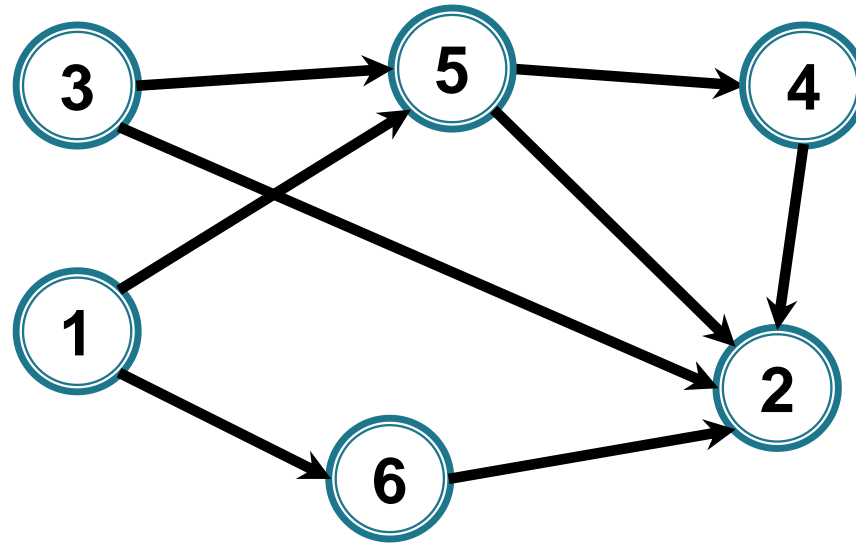
- Pornim cu toate vârfurile cu grad intern 0 și le adăugăm într-o coadă
- Repetăm:
 - extragem un vârf din coadă
 - îl eliminăm din graf (= scădem gradele interne ale vecinilor, nu îl eliminăm din reprezentare)
 - adăugăm în coadă vecinii al căror grad intern devine 0

Exemplu

Sortare topologică

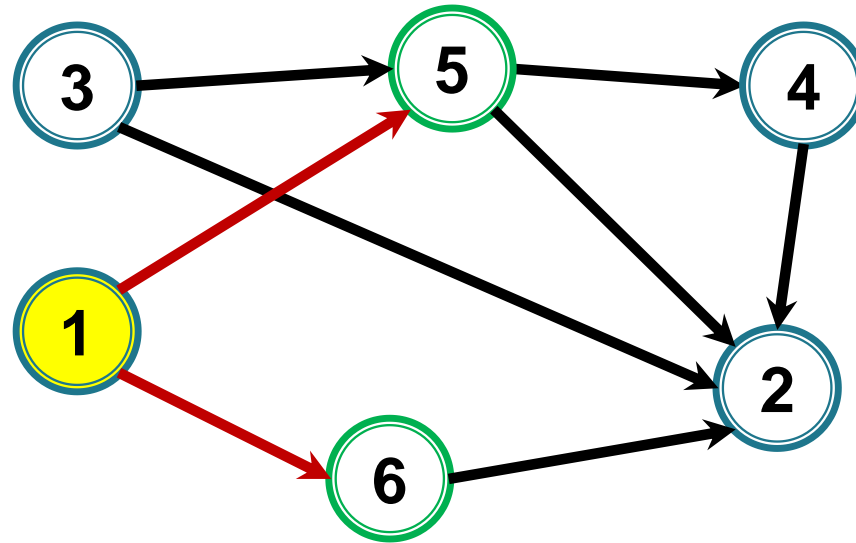


Sortare topologică



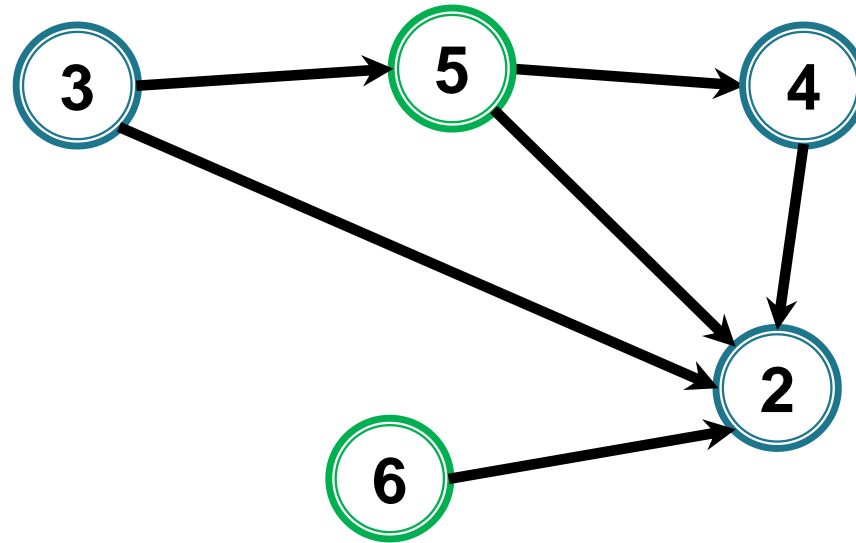
C: 1 3

Sortare topologică



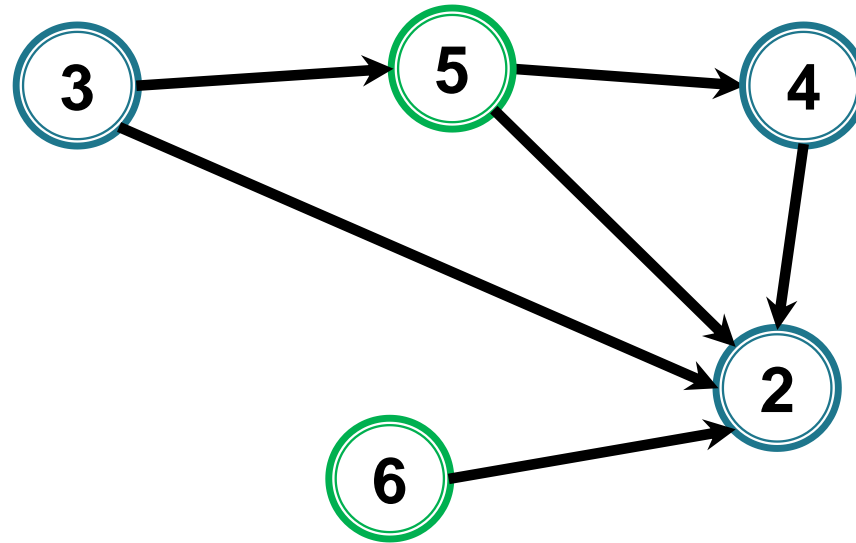
C: **1** 3

Sortare topologică



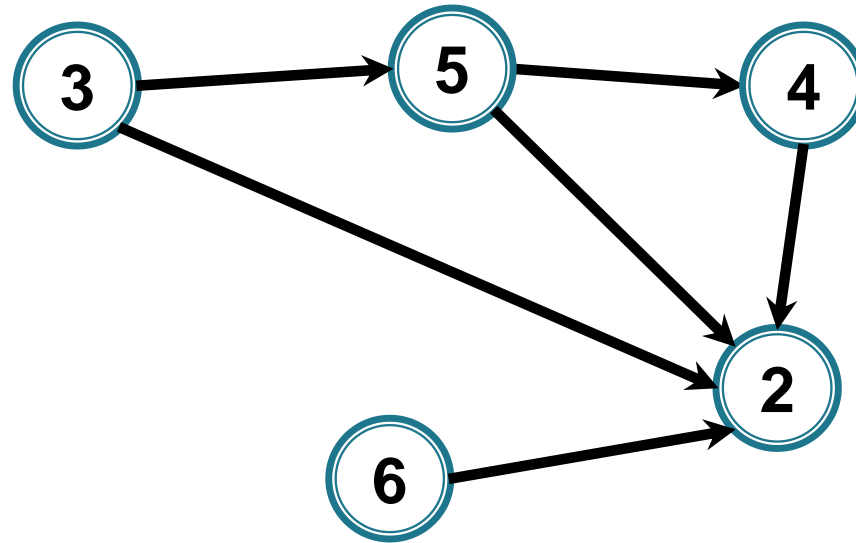
C: **1** 3

Sortare topologică



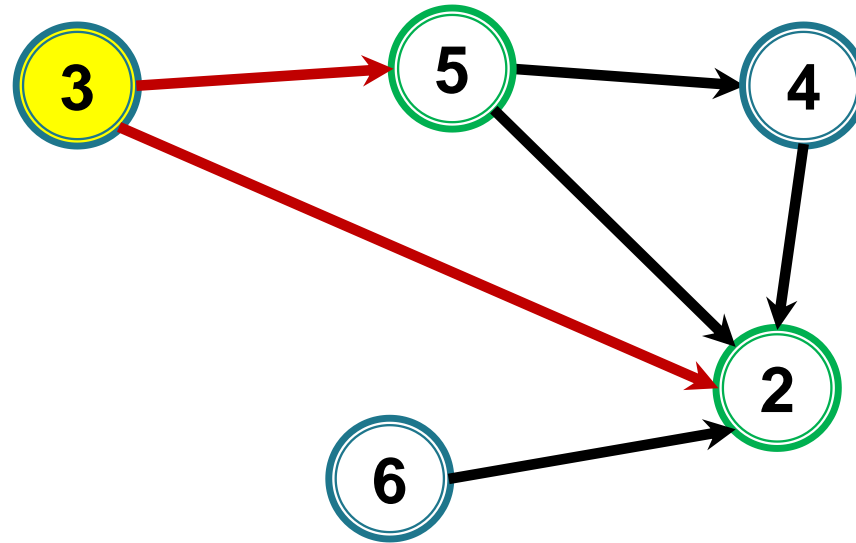
C: **1** 3 6

Sortare topologică



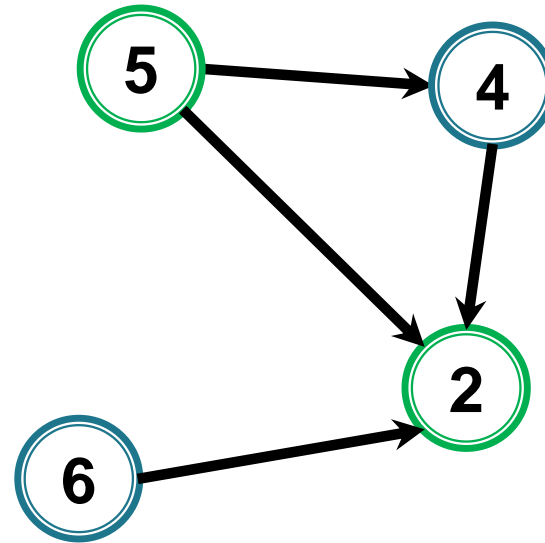
C: **1** 3 6

Sortare topologică



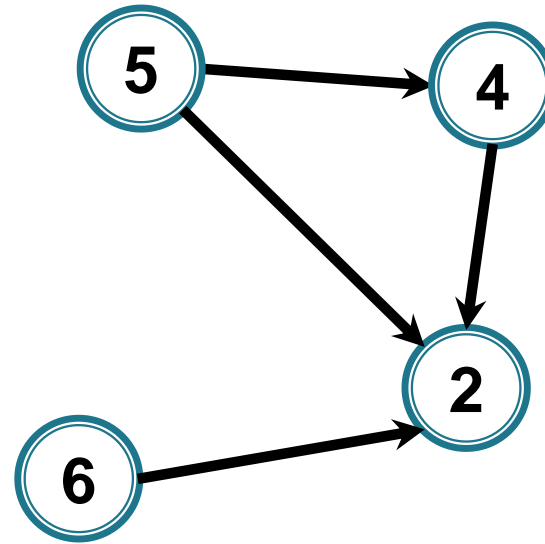
C: 1 3 6

Sortare topologică



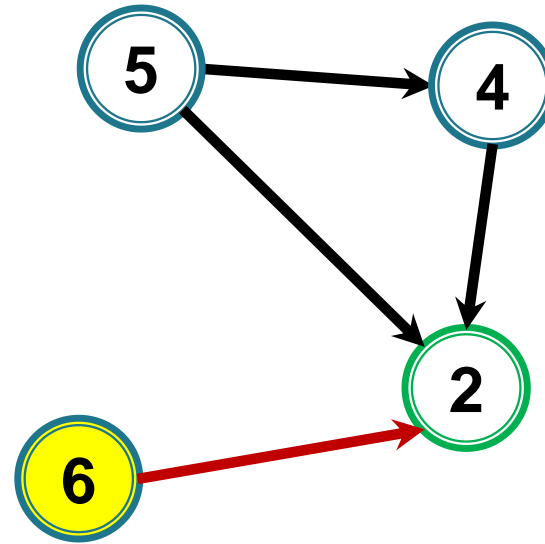
C: **1** **3** 6

Sortare topologică



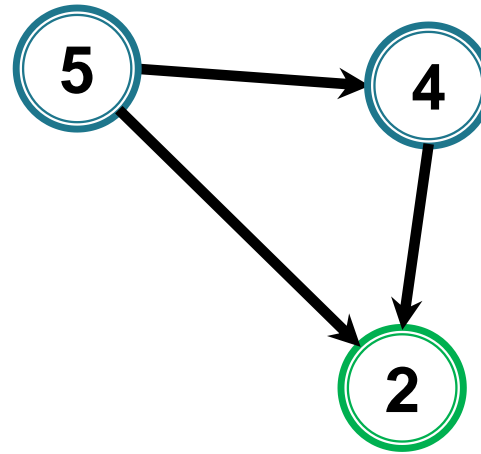
C: **1** **3** 6 5

Sortare topologică



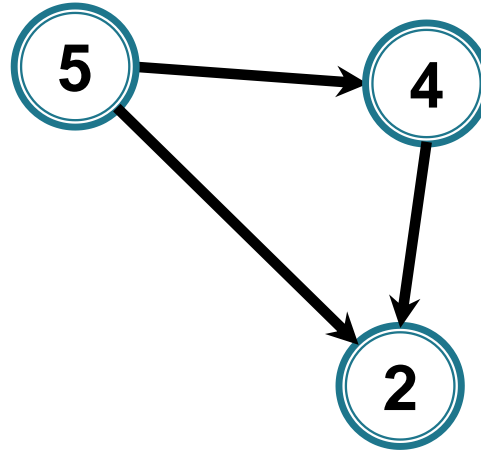
C: 1 3 6 5

Sortare topologică



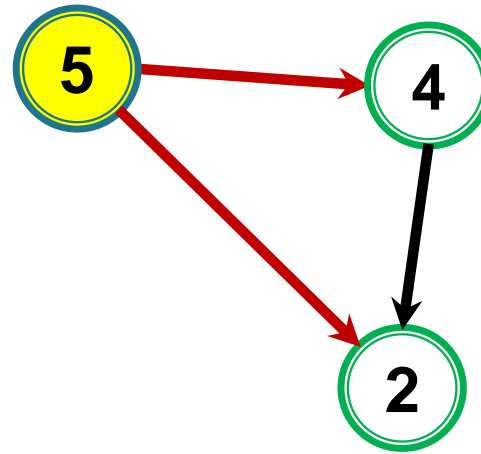
C: **1** **3** **6** 5

Sortare topologică



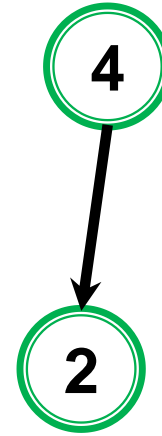
C: **1** **3** **6** 5

Sortare topologică



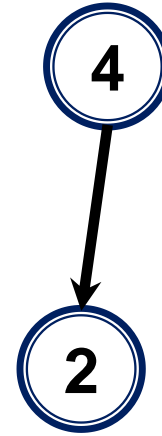
C: 1 3 6 5

Sortare topologică



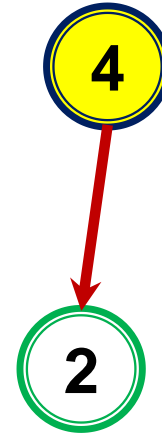
C: 1 3 6 5

Sortare topologică



C: 1 3 6 5 4

Sortare topologică



C: 1 3 6 5 4

Sortare topologică

2

C: 1 3 6 5 4

Sortare topologică

2

C: 1 3 6 5 4 2

Sortare topologică

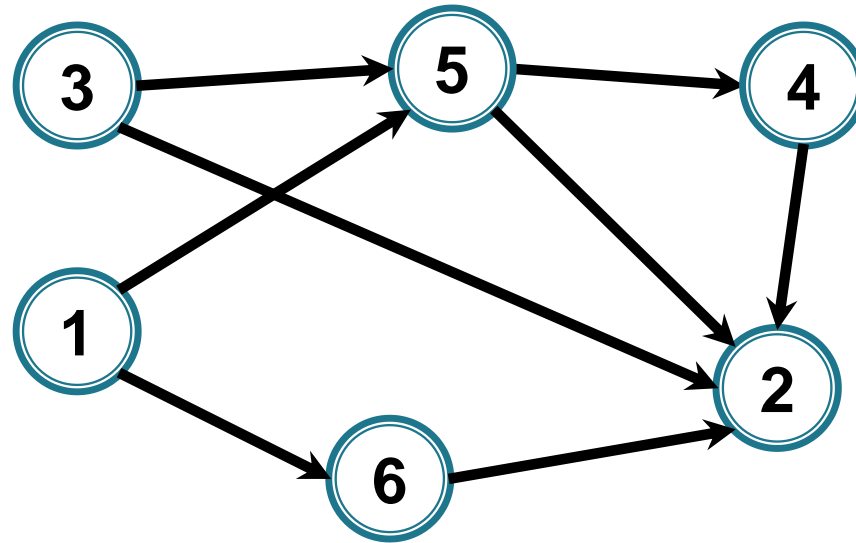


C: 1 3 6 5 4 2

Sortare topologică

C: 1 3 6 5 4 2

Sortare topologică



Sortare topologică: **1 3 6 5 4 2**

Sortare topologică – Algoritm

coada $C = \emptyset;$

adauga in C toate vârfurile v cu $d^-[v]=0$

Sortare topologică – Algoritm

coada $C = \emptyset$;

adauga in C toate vârfurile v cu $d^-[v]=0$

cat timp $C \neq \emptyset$ executa

$i \leftarrow \text{extrage}(C)$;

 adauga i in sortare

 pentru $ij \in E$ executa

Sortare topologică – Algoritm

coada $C = \emptyset$;

adauga in C toate vârfurile v cu $d^-[v]=0$

cat timp $C \neq \emptyset$ executa

$i \leftarrow \text{extrage}(C)$;

 adauga i in sortare

 pentru $ij \in E$ executa

$d^-[j] = d^-[j] - 1$

Sortare topologică – Algoritm

coada $C = \emptyset$;

adauga in C toate vârfurile v cu $d^-[v]=0$

cat timp $C \neq \emptyset$ executa

$i \leftarrow \text{extrage}(C)$;

 adauga i in sortare

 pentru $ij \in E$ executa

$d^-[j] = d^-[j] - 1$

 daca $d^-[j]==0$ atunci

 adauga(j , C)

Sortare topologică



- ▶ Ce se întâmplă dacă graful conține totuși circuite?
- ▶ Cum detectăm acest lucru pe parcursul algoritmului?

Alt algorithm

Sortare topologică – Alt algoritm

- ▶ Există un algoritm bazat pe DF, pornind de la următoarea **observație**:
 - Dacă $\text{fin}[u]$ = momentul la care a fost finalizat vârful u în parcurgerea DF avem:
$$uv \in E \Rightarrow \text{fin}[u] > \text{fin}[v]$$
(deoarece uv nu poate fi arc de întoarcere)

Sortare topologică – Alt algoritm

- ▶ Există un algoritm bazat pe DF, pornind de la următoarea **observație**:
 - Dacă $\text{fin}[u]$ = momentul la care a fost finalizat vârful u în parcurgerea DF avem:
$$uv \in E \Rightarrow \text{fin}[u] > \text{fin}[v]$$
 - Atunci sortare descrescătoare în raport cu fin \Rightarrow sortare topologică

Sortare topologică – Alt algoritm

- ▶ Există un algoritm bazat pe DF, pornind de la următoarea **observație**:
 - Dacă $\text{fin}[u]$ = momentul la care a fost finalizat vârful u în parcurgerea DF avem:
$$uv \in E \Rightarrow \text{fin}[u] > \text{fin}[v]$$
 - Atunci sortare descrescătoare în raport cu fin \Rightarrow sortare topologică
 - \Rightarrow Idee algoritm: **Memorăm vârfurile într-o stivă pe măsura finalizării lor**; ordinea în care sunt scoase din stivă = sortare topologică

Sortare topologică – Alt algoritm

Stack S

DFS(i)

 viz[i] = 1

 pentru ij ∈ E

 daca viz[j]==0 atunci

 DFS(j)

 //i este finalizat

Sortare topologică – Alt algoritm

Stack S

DFS(i)

 viz[i] = 1

 pentru ij \in E

 daca viz[j]==0 atunci

 DFS(j)

 //i este finalizat

 push(S, i)

pentru i \in V executa

 daca viz[i]==0 atunci

 DFS(i)

cat timp S este nevida executa

 u = pop(S)

 adauga u in sortare

