

# Sortarea topologică



# Sortarea topologică

Fie  $G = (V, E)$  un **graf orientat**.

**Sortarea topologică a lui  $G$  =**

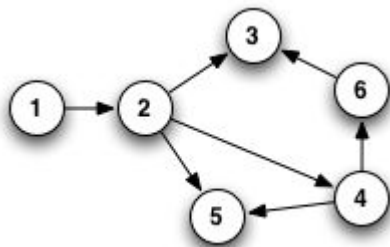
ordonarea vârfurilor astfel încât, dacă  $uv \in E$ , atunci  $u$  se află înaintea lui  $v$  în ordonare.

**Nu este neapărat unică!**

# Sortarea topologică – Aplicații

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

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



În ce ordine trebuie executate activitățile?

Formulele din celulele B2..D2

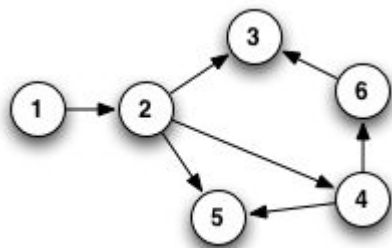
	A	B	C	D
1		3	2	
2		3	6	0
3				
4		"=B1+D2"	"=2*B2"	"=2*C1+C2"

În ce ordine se evaluează formulele?

**Probleme: dacă există dependențe circulare**

# Sortarea topologică – Aplicații

- planificarea de proiecte, ordinea de execuție a unor operații: compilarea pachetelor, ordinea de calcul a formulelor din excel etc
- determinarea de drumuri minime în grafuri fără circuite



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"

În ce ordine se evaluează formulele?

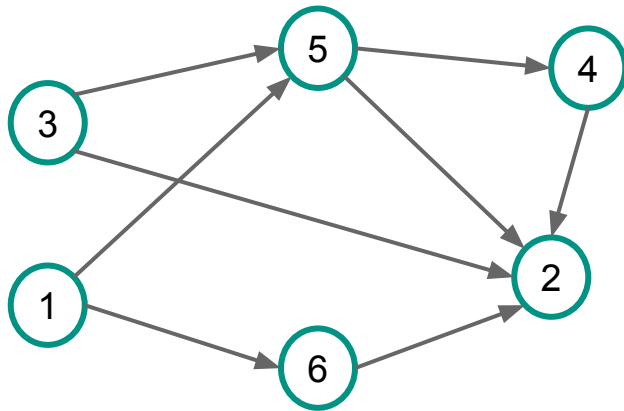
**Probleme: dacă există dependențe circulare**

# Sortarea topologică

Fie  $G = (V, E)$  un **graf orientat**.

**Sortarea topologică a lui  $G$  =**

ordonarea vârfurilor astfel încât, dacă  $uv \in E$ , atunci  $u$  se află înaintea lui  $v$  în ordonare.



# Sortarea topologică

Fie  $G = (V, E)$  un **graf orientat**.

**Sortarea topologică a lui  $G$  =**

ordonarea 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.

# Sortarea topologică

Fie  $G = (V, E)$  un **graf orientat**.

**Sortarea topologică a lui  $G$  =**

ordonarea 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?



# Sortarea topologică

Fie  $G = (V, E)$  un **graf orientat**.

**Sortarea topologică a lui  $G$  =**

ordonarea 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 poate fi primul nod în sortarea topologică?





# Sortarea topologică

Fie  $G = (V, E)$  **graf orientat**.

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

# Pseudocod



# Sortarea topologică – Pseudocod

cât timp  $|V(G)| > 0$  execută

    alege  $v$  cu  $d^-(v) = 0$

    adauga  $v$  în ordonare

$G \leftarrow G - v$

**Corectitudine:** rezultă din **Lemă** + inducție

# Sortarea topologică – Pseudocod

cât timp  $|V(G)| > 0$  execută

    alege  $v$  cu  $d^-(v) = 0$

    adauga  $v$  în ordonare

$G \leftarrow G - v$



**Implementare? Complexitate?**

# Sortarea topologică – Pseudocod

cât timp  $|V(G)| > 0$  execută

    alege  $v$  cu  $d^-(v) = 0$

    adauga  $v$  în ordonare

$G \leftarrow G - v$

## Implementare - similar BF

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

# Sortarea topologică – Pseudocod

cât timp  $|V(G)| > 0$  execută

    alege  $v$  cu  $d^-(v) = 0$

    adauga  $v$  în 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**)

# Sortarea topologică – Pseudocod

cât timp  $|V(G)| > 0$  execută

    alege  $v$  cu  $d^-(v) = 0$

    adauga  $v$  în ordonare

$G \leftarrow G - v$

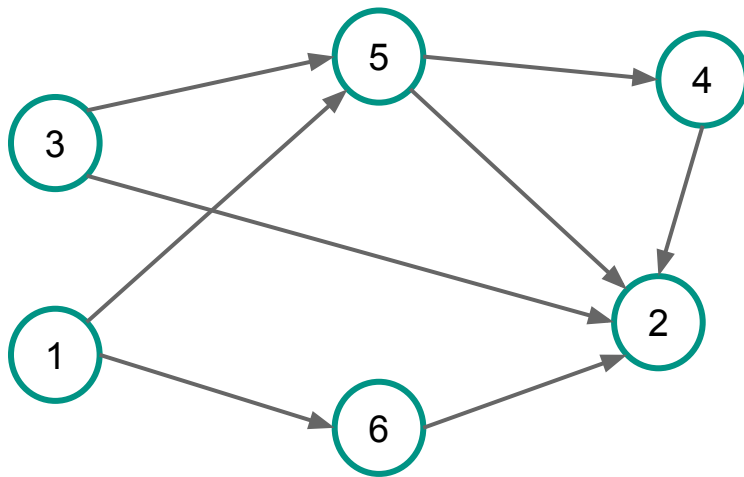
## Implementare - similar BF

- ☐ 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

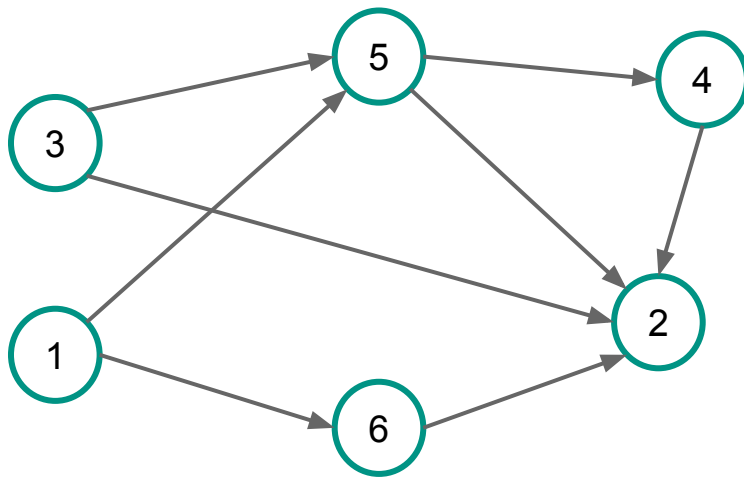
# Exemplu



# Sortare topologică – Exemplu

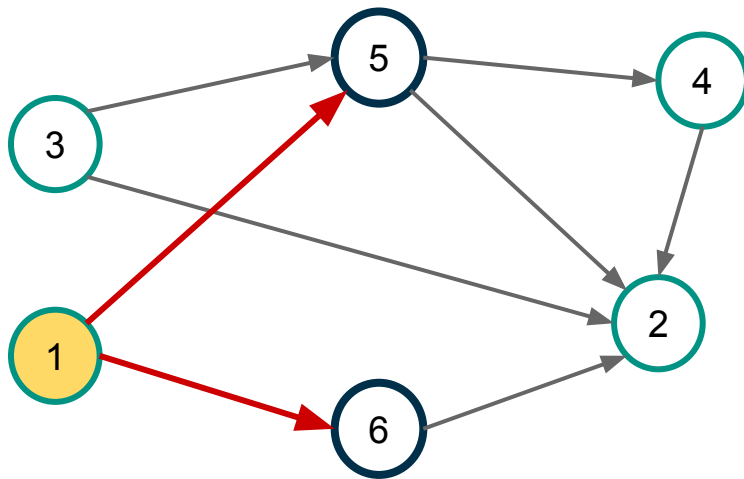


# Sortare topologică – Exemplu



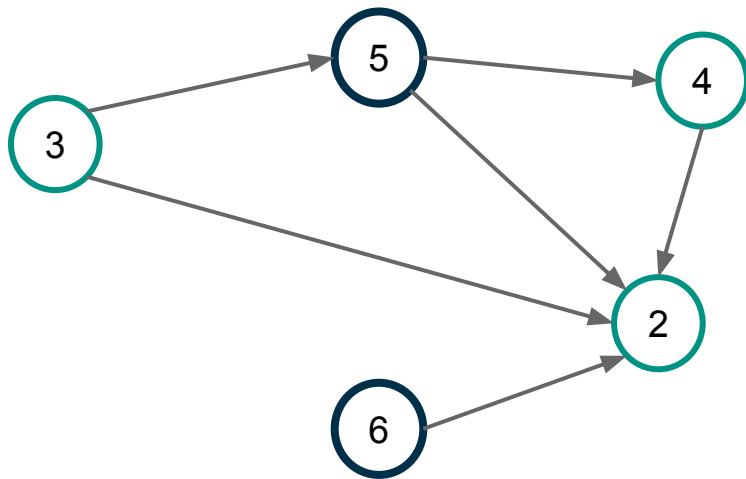
C: 1 3

# Sortare topologică – Exemplu



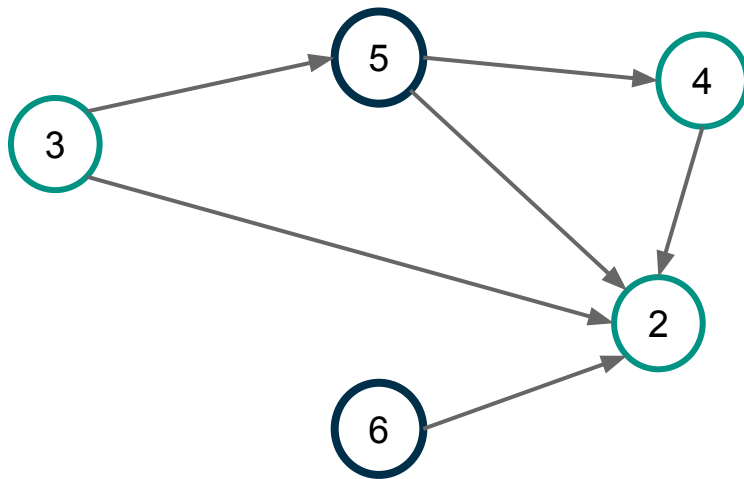
C: **1** 3

# Sortare topologică – Exemplu



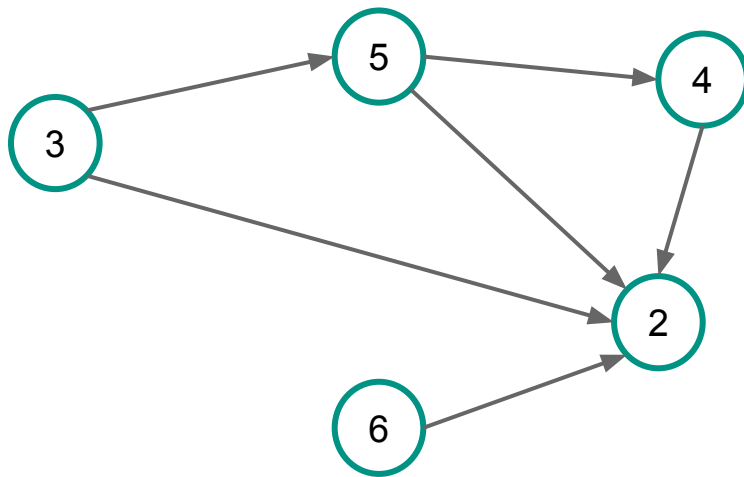
C: **1** 3

# Sortare topologică – Exemplu



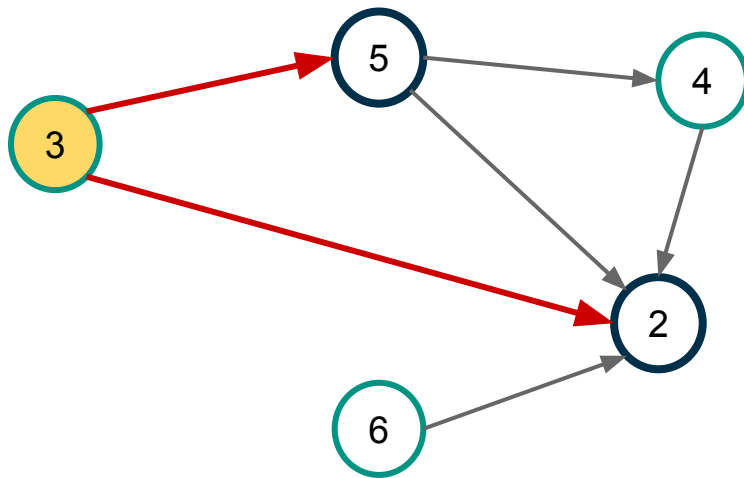
C: **1** 3 6

# Sortare topologică – Exemplu



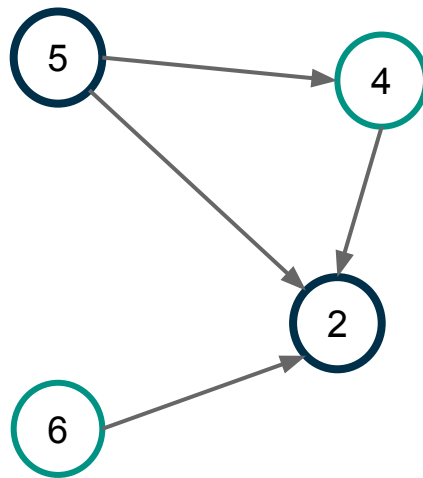
C: **1** 3 6

# Sortare topologică – Exemplu



C: 1 3 6

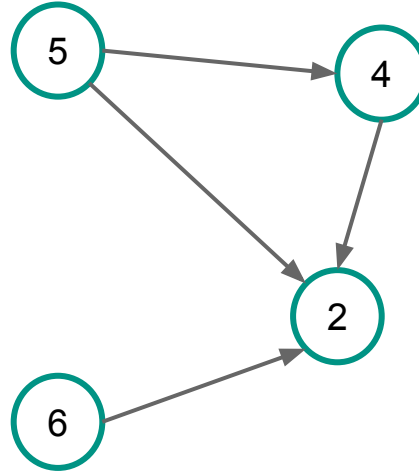
# Sortare topologică – Exemplu



C: 1 3 6

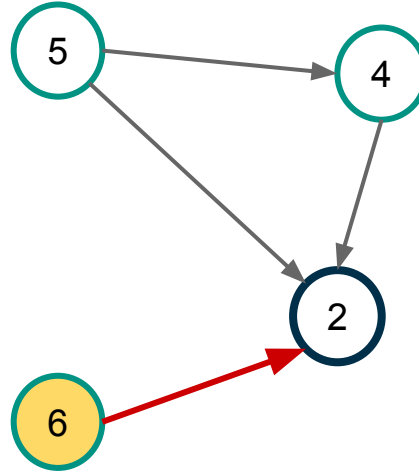


# Sortare topologică – Exemplu



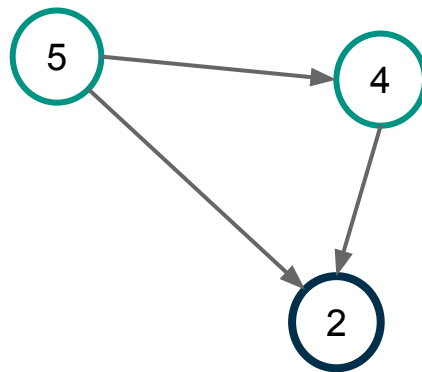
C: 1 3 6 5

# Sortare topologică – Exemplu



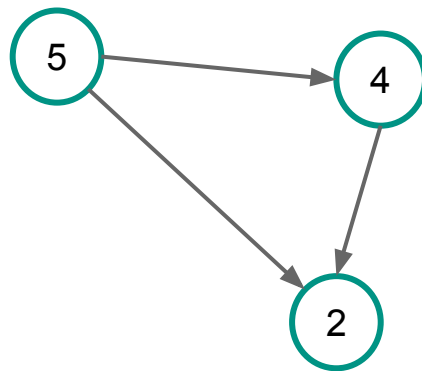
C: 1 3 6 5

# Sortare topologică - Exemplu



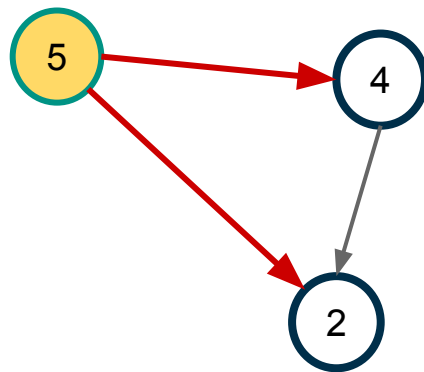
C: 1 3 6 5

# Sortare topologică - Exemplu



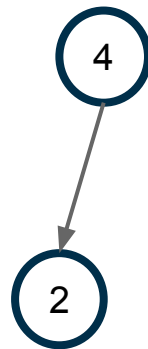
C: 1 3 6 5

# Sortare topologică - Exemplu



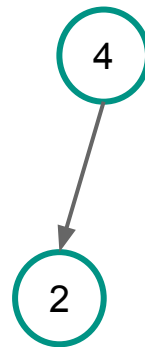
C: 1 3 6 5

# Sortare topologică - Exemplu



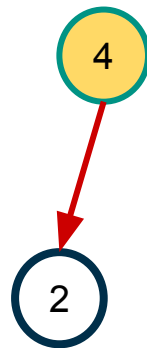
C: 1 3 6 5

# Sortare topologică - Exemplu



C: 1 3 6 5 4

# Sortare topologică - Exemplu



C: 1 3 6 5 4



# Sortare topologică – Exemplu



C: 1 3 6 5 4

# Sortare topologică – Exemplu

2

C: 1 3 6 5 4 2

# Sortare topologică – Exemplu

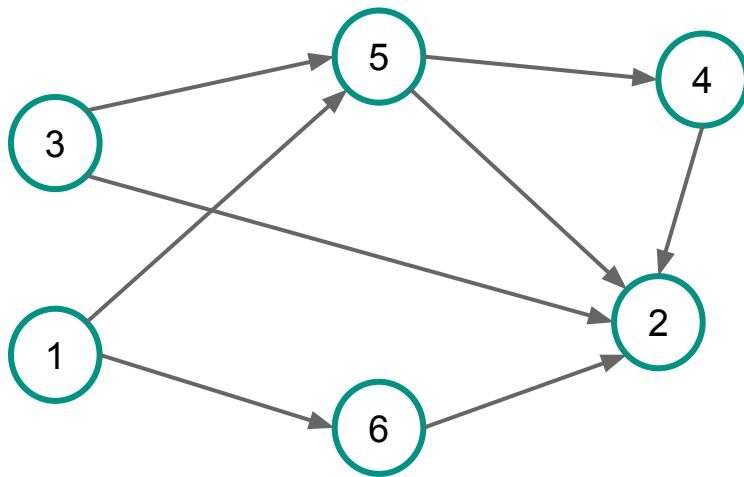


C: 1 3 6 5 4 2

# Sortare topologică – Exemplu

C: 1 3 6 5 4 2

# Sortare topologică – Exemplu



**SORTARE TOPOLOGICĂ: 1 3 6 5 4 2**

# Algorithm

A dark blue diagonal gradient bar that starts at the bottom left and extends towards the top right, covering the lower half of the slide.

# Sortare topologică – Algoritm

**coada**  $C \leftarrow \emptyset$

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

# Sortare topologică – Algoritm

**coada**  $C \leftarrow \emptyset$

**adauga** in  $C$  toate varfurile  $v$  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**  $C \leftarrow \emptyset$

**adauga** in  $C$  toate varfurile  $v$  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**  $C \leftarrow \emptyset$

**adauga** in  $C$  toate varfurile  $v$  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 – Algoritm



**Ce se întâmplă dacă graful conține, totuși, circuite?**

**Cum detectăm acest lucru pe parcursul algoritmului?**

# Alt algorithm

# Sortare topologică – Alt algoritm

## Suplimentar

Există un algoritm bazat pe DF, pornind de la următoarea **observație**:

- dacă **final[u]** = momentul la care a fost **finalizat** vârful  $u$  în parcurgerea DF, avem:
  - $uv \in E \Rightarrow f[u] > f[v]$

# Sortare topologică – Alt algoritm

## Suplimentar

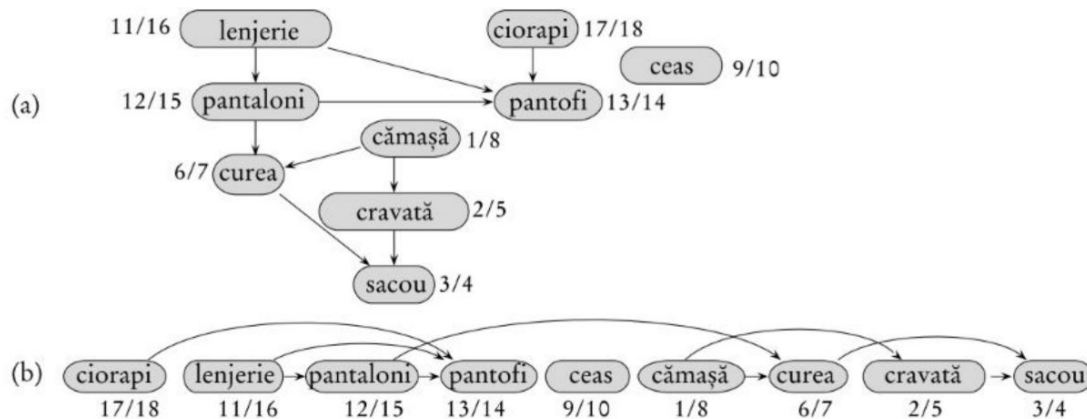
Există un algoritm bazat pe DF, pornind de la următoarea **observație**:

- dacă **final[u]** = momentul la care a fost **finalizat** vârful  $u$  în parcurgerea DF, avem:
  - $uv \in E \Rightarrow f[u] > f[v]$
- atunci sortarea topologică = sortare descrescătoare în raport cu **final**

# Sortare topologică – Alt algoritm

Dacă **final[u]** = momentul la care a fost **finalizat** vârful  $u$  în parcurgerea DF, avem:  $uv \in E \Rightarrow f[u] > f[v]$

□ atunci sortarea topologică = sortare descrescătoare în raport cu **final**



**Figura 23.7** (a) Profesorul Bumstead își sortează topologic îmbrăcămintea când se îmbracă. Fiecare muchie  $(u, v)$  înseamnă că articolul  $u$  trebuie îmbrăcat înaintea articolului  $v$ . Timpii de descoperire și de terminare dintr-o căutare în adâncime sunt prezentați alături de fiecare vârf. (b) Același graf sortat topologic. Vârfurile lui sunt aranjate de la stânga la dreapta în ordinea descrescătoare a timpului de terminare. Se observă că toate muchiile orientate merg de la stânga la dreapta.

# Sortare topologică – Alt algoritm

Dacă **final[u]** = momentul la care a fost **finalizat** vârful  $u$  în parcurgerea DF, avem:  $uv \in E \Rightarrow f[u] > f[v]$

- atunci sortarea topologică = sortare descrescătoare în raport cu **final**

## Sortare\_Topologică(G)

1. apelează  $CA(G)$  pentru a calcula timpii de terminare  $f[v]$  pentru fiecare vârf  $v$
2. pe măsură ce fiecare vârf este terminat, inserează în capul unei liste înlănțuite
3. returnează lista înlănțuită de vârfuri



