Sortare topologică

Sortare topologică

- **Fie G = (V, E) graf orientat**
- **Sortare topologică a lui G =**

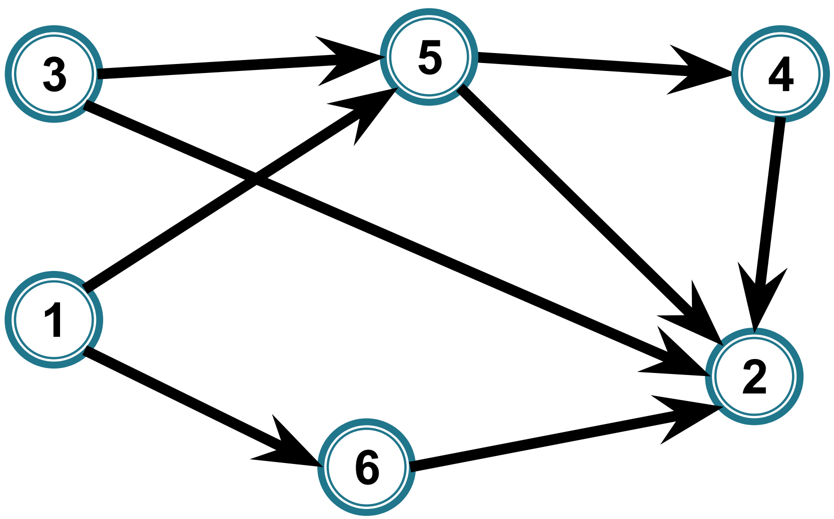
  ordonare a vârfurilor astfel încât dacă uv ∈ E atunci u se află înaintea lui v în ordonare

  - **Nu este neapărat unică**

Sortare topologică

- **Fie G = (V, E) graf orientat**
- **Sortare topologică a lui G =**

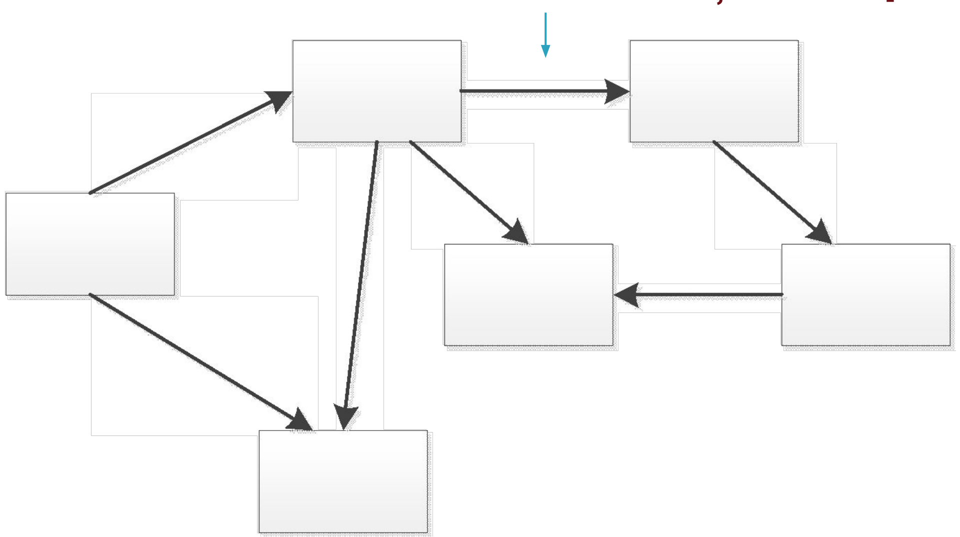
  ordonare a vârfurilor astfel încât dacă $uv \in E$ atunci u se află înaintea lui v în ordonare

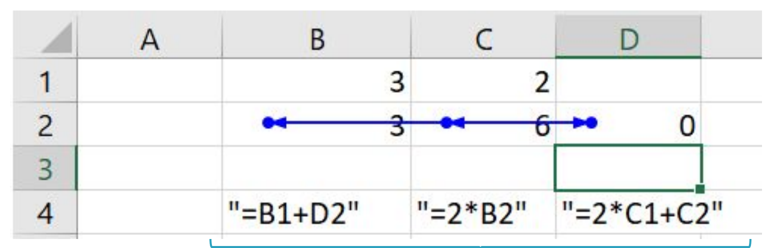


Sortare topologică

- **<u>Aplicații</u>**
  - **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





formulele din celulele B2..D2

În ce ordine trebuie executate activitățile?

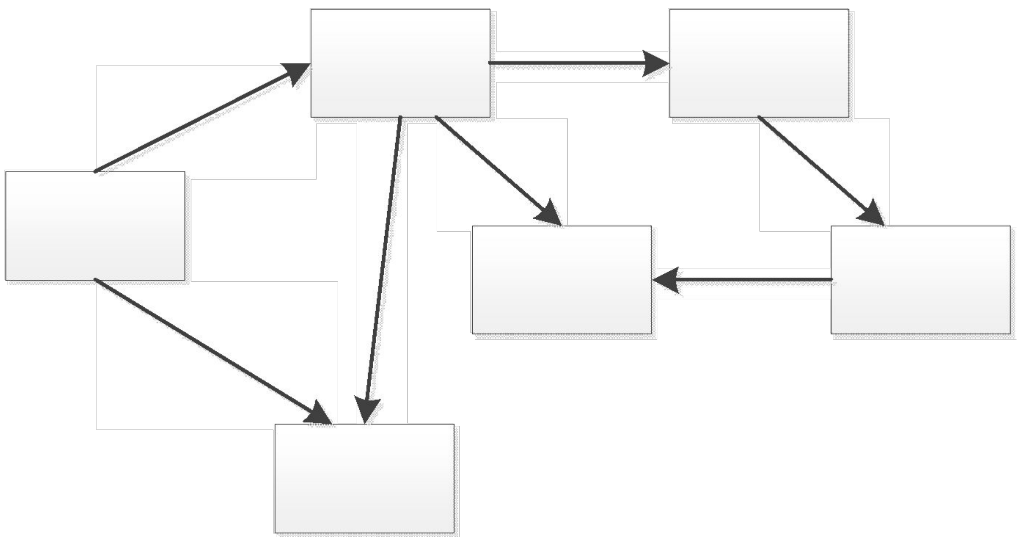
În ce ordine se evaluează formulele?

Probleme - dacă există dependențe circulare

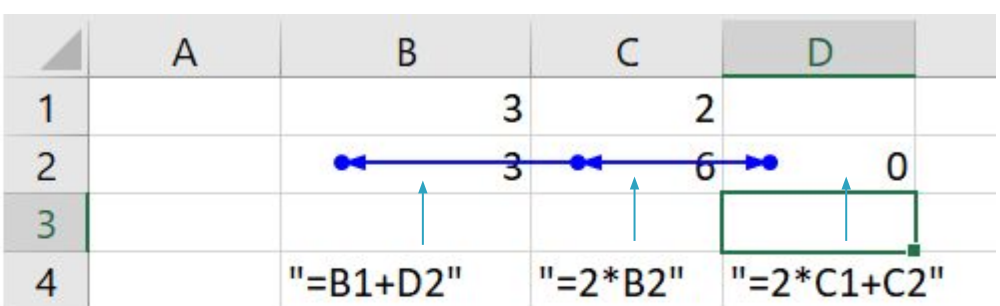
Sortare topologică

Aplicații

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



În ce ordine trebuie executate activitățile?



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

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

- **Propoziţie.** 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 ∈ E atunci u se află înaintea lui v în ordonare

- **Propoziţie.** Dacă G este aciclic atunci G are o sortare topologică
  - **Demonstrație ⇒ Algoritm?**

Sortare topologică

- **Fie G = (V, E) graf orientat**
- **Sortare topologică a lui G =**

  ordonare a vârfurilor astfel încât dacă uv ∈ E atunci u se află înaintea lui v în ordonare

- **Propoziţie.** Dacă G este aciclic atunci G are o sortare topologică

  - **Demonstrație ⇒ 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 puţin 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 puţin 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 in ordonare
    G  ← 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  ← G - v
```

- **Implementare - similar BF**
  - Pornim cu <u>toate</u> vârfurile cu grad intern 0 i 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  ← G - v
```

- **Implementare - similar BF**
  - **Pornim cu toate vârfurile cu grad intern 0 i 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  ← G - v
```

- **Implementare - similar BF**
  - Pornim cu toate vârfurile cu grad intern 0 și le adăugăm într-o coadă
  - Repetăm:

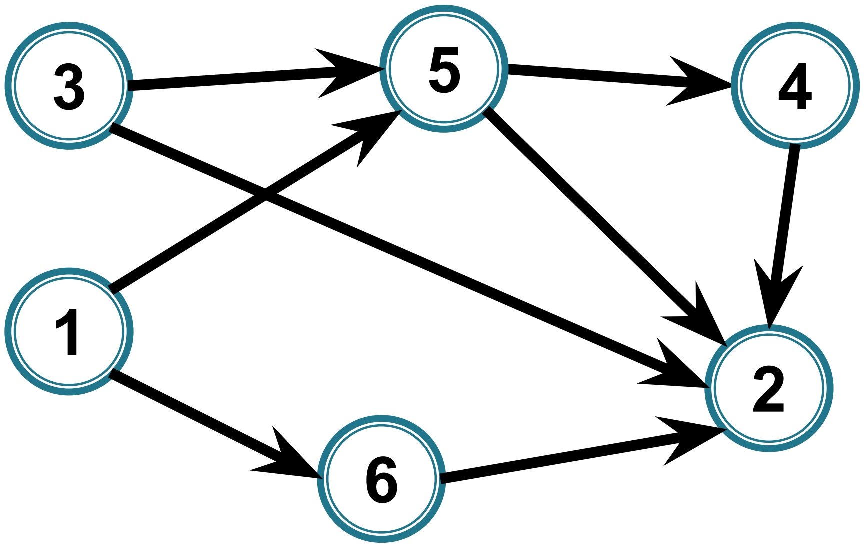
    \- extragem un vârf din coadă

    \- îl eliminăm din graf (= scădem gradele interne ale vecinilor, nu îl eliminăm din reprezentare)

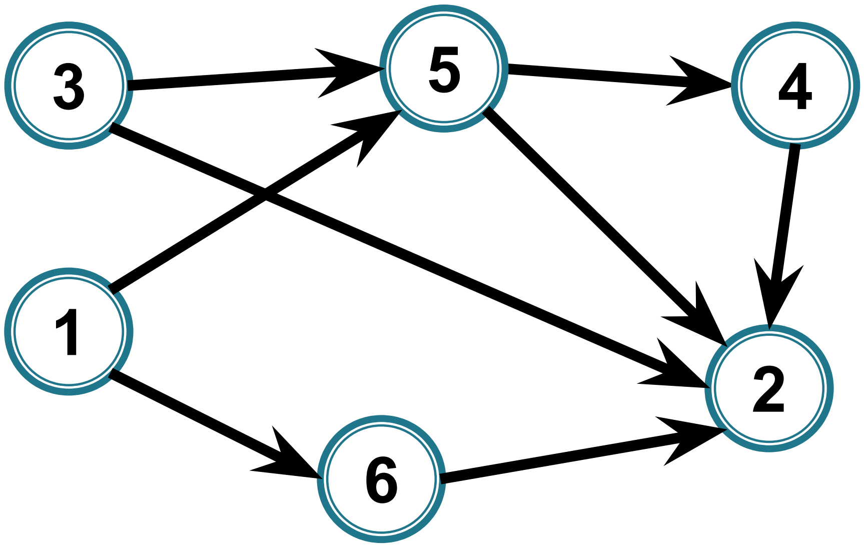
    \- adăugăm în coadă vecinii al căror grad intern devine 0

Exemplu

Sortare topologică

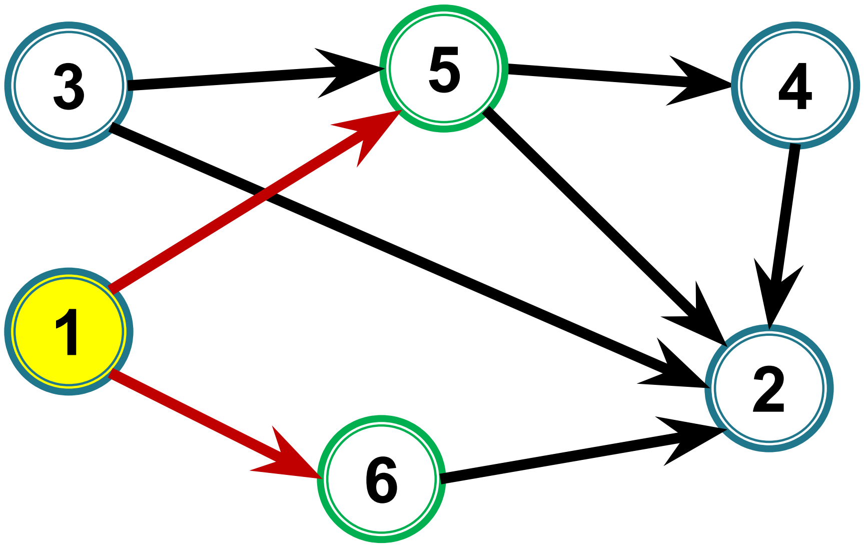


Sortare topologică



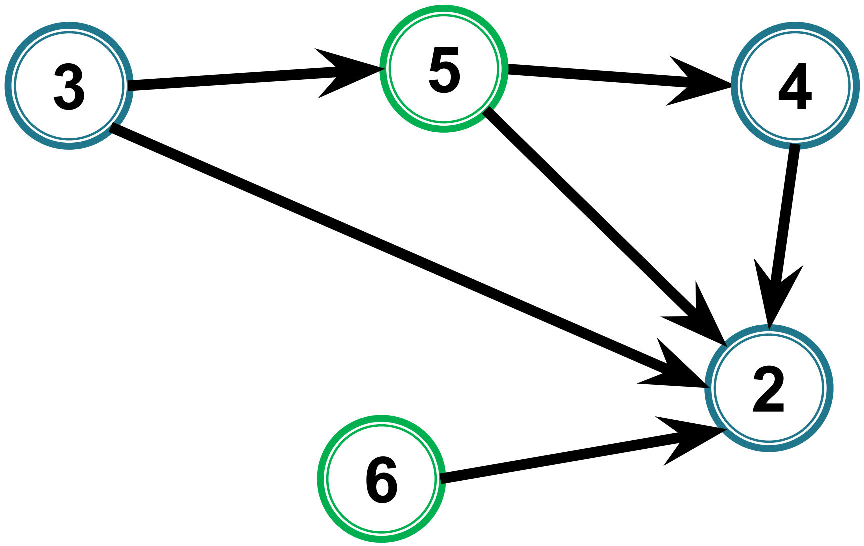
C: 1 3

Sortare topologică



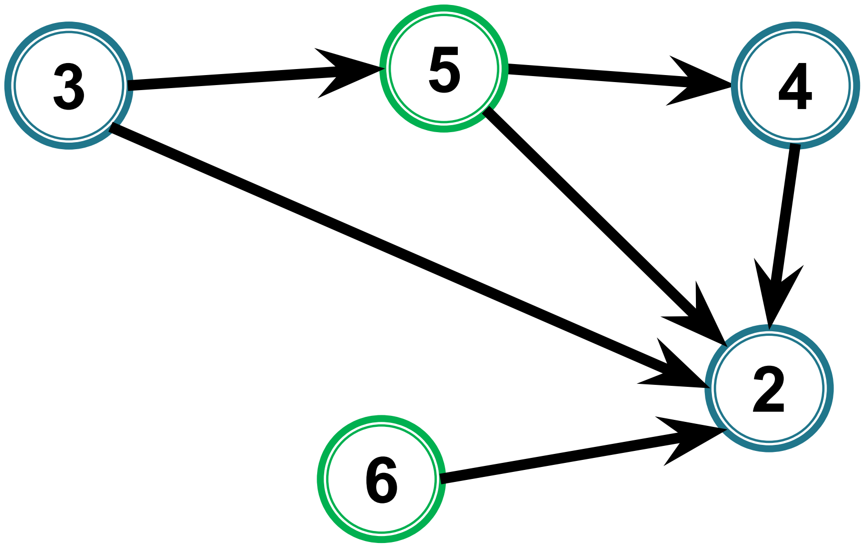
C: 1 3

Sortare topologică



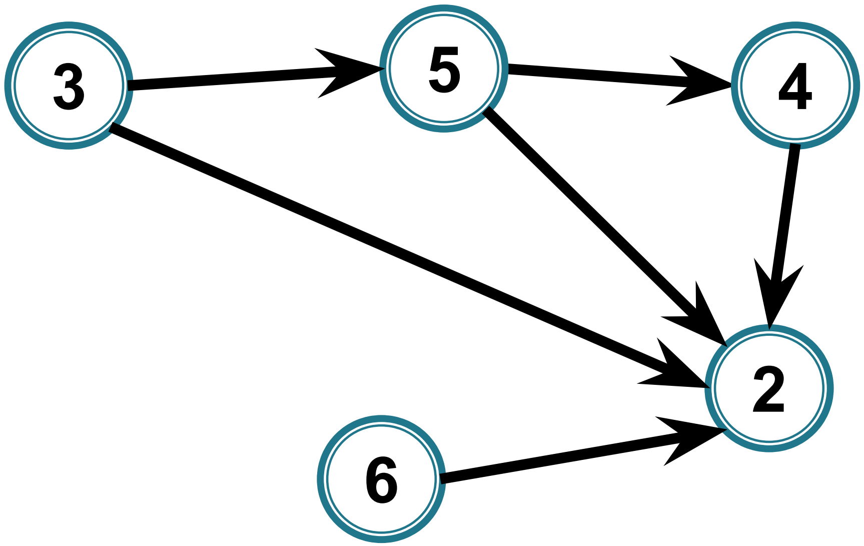
C: 1 3

Sortare topologică



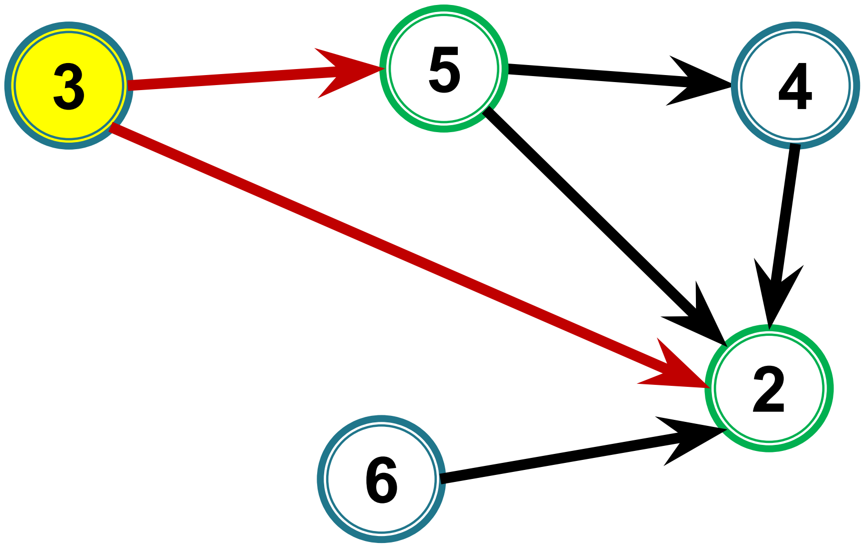
C: 1 3 6

Sortare topologică



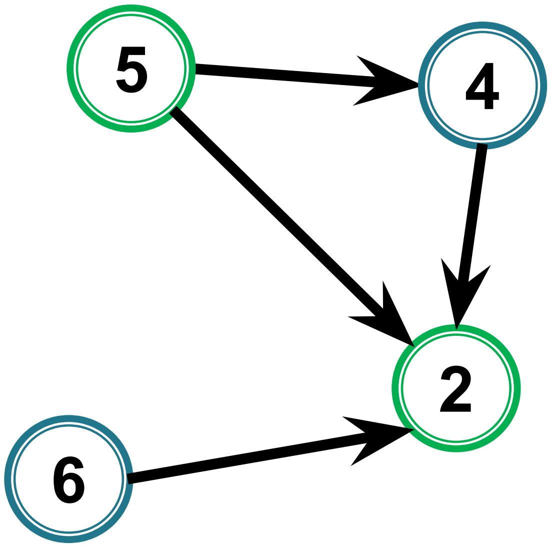
C: 1 3 6

Sortare topologică



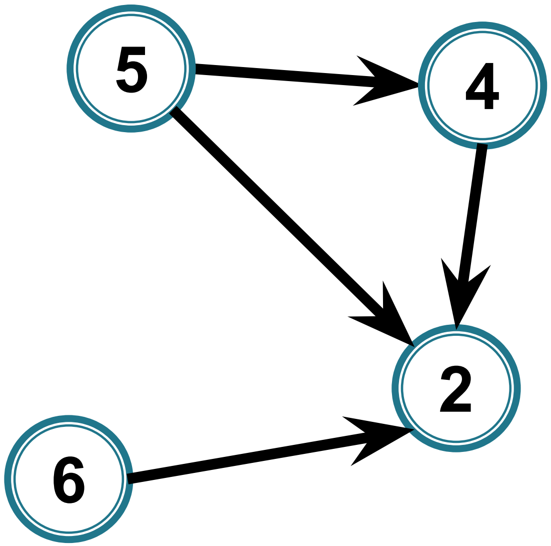
C: 1 3 6

Sortare topologică



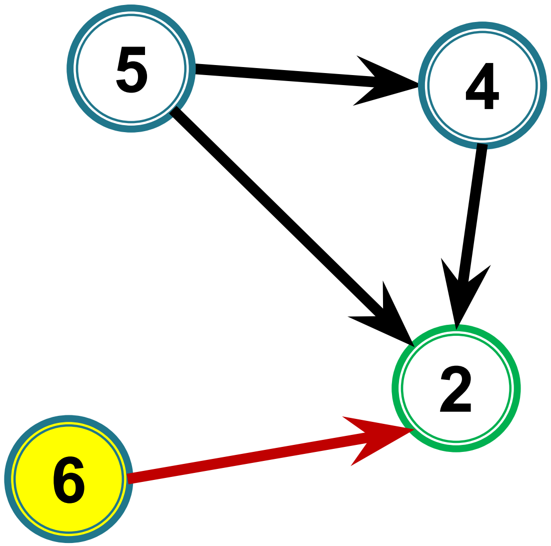
C: 1 3 6

Sortare topologică



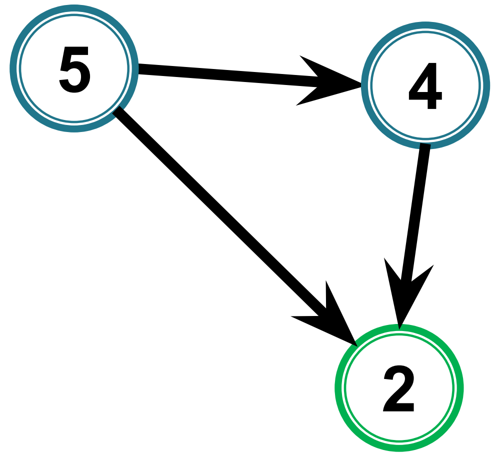
C: 1 3 6
5

Sortare topologică



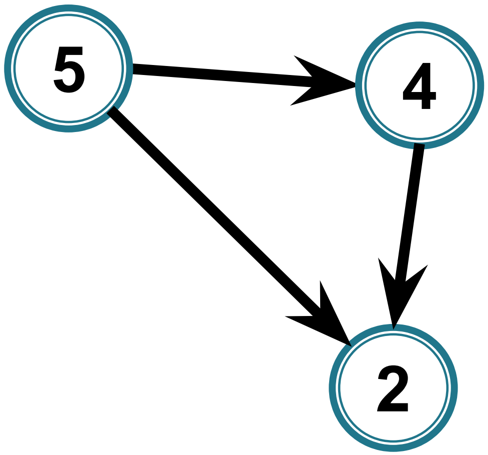
C: 1 3 6 5

Sortare topologică



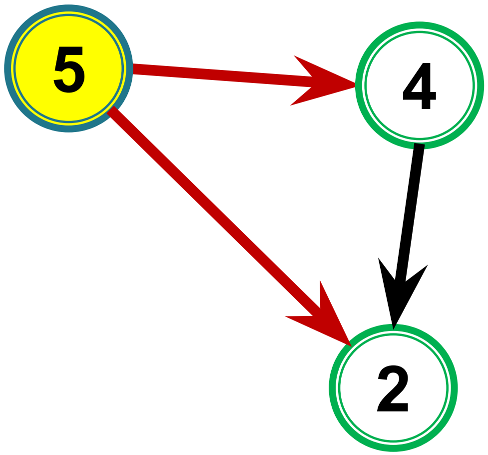
C: 1 3 6 5

Sortare topologică



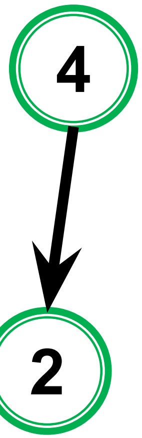
C: 1 3 6 5

Sortare topologică



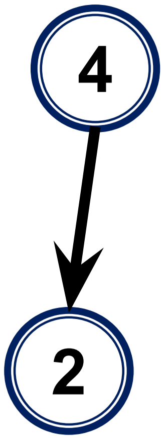
C: 1 3 6 5

Sortare topologică



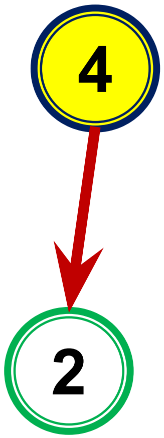
C: 1 3 6 5

Sortare topologică



C: 1 3 6 5 4

Sortare topologică



C: 1 3 6 5 4

Sortare topologică

C: **1 3 6 5 4**

Sortare topologică

C: 1 3 6 5 4 2

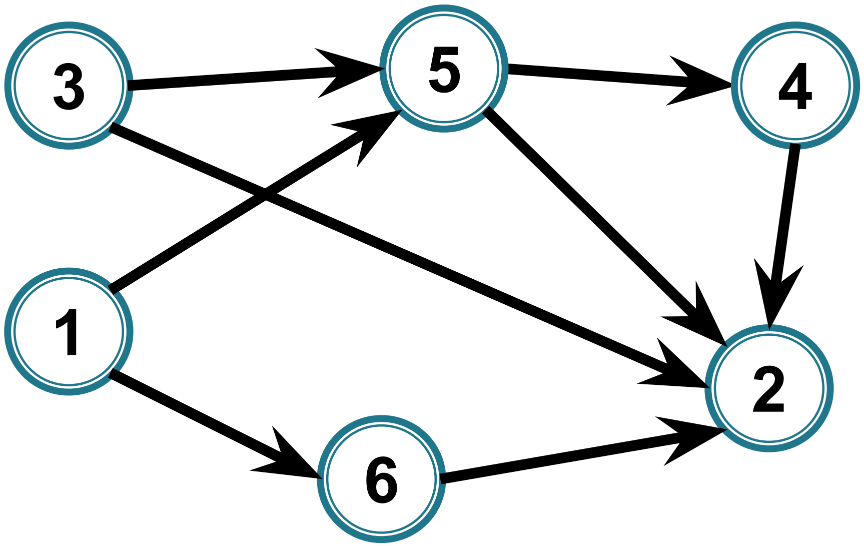
Sortare topologică

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 ← ∅;
adauga in C toate vârfurile v cu d⁻[v]=0
```

Sortare topologică - Algoritm

```
coada C ← ∅;
adauga in C toate vârfurile v cu d⁻[v]=0

cat timp C ≠ ∅ executa
    i ←  extrage(C);
    adauga i in sortare
    pentru ij ∈ E executa
```

Sortare topologică - Algoritm

```
coada C ← ∅;
adauga in C toate vârfurile v cu d⁻[v]=0

cat timp C ≠ ∅ executa
    i ←  extrage(C);
    adauga i in sortare
    pentru ij ∈ E executa
        d⁻[j] = d⁻[j] - 1
```

Sortare topologică - Algoritm

```
coada C ← ∅;
adauga in C toate vârfurile v cu d⁻[v]=0

cat timp C ≠ ∅ executa
   i ←  extrage(C);
   adauga i in sortare
   pentru ij ∈ 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 algoritm

Sortare topologică – Alt algoritm

- **Suplimentar** - există un algoritm bazat pe DF, pornind de la următoarea **observație:**

  - Dacă **f**[u] = momentul la care a fost <u>**finalizat**</u> vârful u în parcurgerea DF avem:

    uv ∈ E ⇒ 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 <u>**finalizat**</u> vârful u în parcurgerea DF avem:

    $$uv \in E \Rightarrow f[u] > f[v]$$

  - Atunci sortare 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 ∈ E ⇒ f[u] > f[v]`**

  - Atunci sortare topologică = sortare descrescătoare în raport cu final

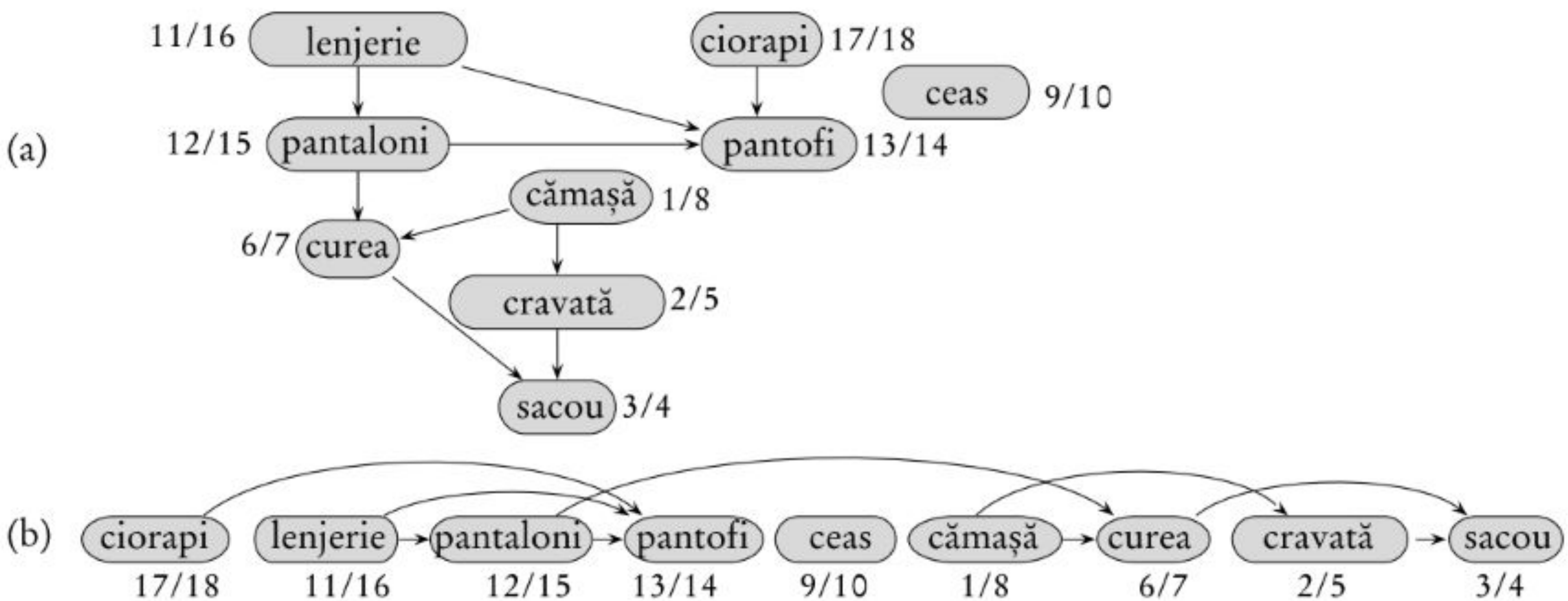


Figura 23.7 **(a)** Profesorul Bumstead își sortează topologic îmbrăcămintea când se îmbracă. Fiecare muchie (u, v) înseamnă că articolul u trebuie îmbrăcat înaintea articolului v. Timpii de descoperire și de terminare dintr-o căutare în adâncime sunt prezentați alături de fiecare vârf. **(b)** Același 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 sortare 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 înlănţuite

3: returnează lista înlănţuită de vârfuri