

# Construcția de grafuri cu secvența gradelor dată

# Secvențe de grade



**Dată o formulă chimică, există un compus chimic cu această formulă? Dar unul aciclic? Ce structuri poate avea un astfel de compus?**

- $C_mH_n$  – poate exista moleculă **aciclică** cu această formulă?

# Secvențe de grade



Din studii empirice, chestionare, analize  $\Rightarrow$  informații despre numărul de interacțiuni ale unui nod

Este realizabilă o rețea de legături între noduri care să respecte numărul de legături?

Dacă da, să se construiască un model de rețea.

# Secvențe de grade



Din studii empirice, chestionare, analize  $\Rightarrow$  informații despre numărul de interacțiuni ale unui nod

**Este realizabilă o rețea de legături între noduri care să respecte numărul de legături? Dacă da, să se construiască un model de rețea.**

**Exemplu:** Într-o grupă de studenți, fiecare student este întrebat cu câți colegi a colaborat în timpul anilor de studii. Este realizabilă o rețea de colaborări care să corespundă răspunsurilor lor (sau este posibil ca informațiile adunate să fie incorecte)?

- Studentul 1 – cu 3
- Studentul 2 – cu 3
- Studentul 3 – cu 2
- Studentul 4 – cu 3
- Studentul 5 – cu 2

# Secvențe de grade

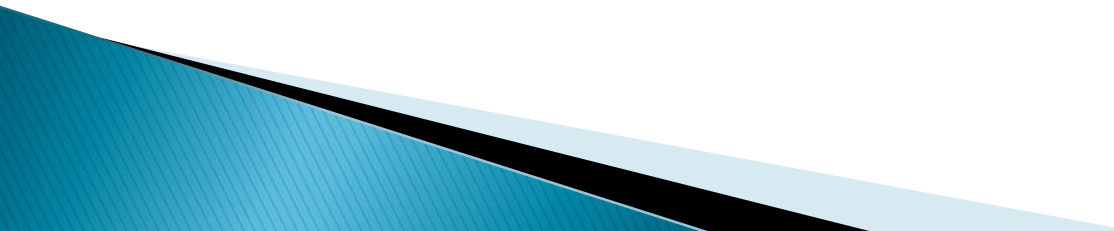


- Dată o secvență de numere  $s$ , se poate construi un graf neorientat având secvența gradelor  $s$ ?
  - Dar un multigraf neorientat?
  - Dar un arbore?
- 
- Condiții necesare
  - Condiții suficiente

# Secvențe de grade

- Construcția de grafuri cu secvența gradelor dată

## Aplicații:

- **chimie** – studiul structurii posibile a unor compuși cu formula chimică dată
  - **proiectare de rețele**
  - **biologie** – rețele metabolice, de interacțiuni între gene/proteine
  - **studii epidemiologice** – în care prin chestionare anonime persoanele declară numărul de persoane cu care au interacționat
  - studii bazate pe simulări de rețele...
- 

Construcția de grafuri neorientate cu  
secvența gradelor dată.

Algoritmul Havel–Hakimi

# Construcția de grafuri cu secvența gradelor dată

## Problemă

Fie  $s_0 = \{d_1, \dots, d_n\}$  o secvență de numere naturale.

Să se construiască, dacă se poate, un graf neorientat  $G$  cu  
 $s(G) = s_0$ .



# Construcția de grafuri cu secvența gradelor dată

## Problemă

Fie  $s_0 = \{d_1, \dots, d_n\}$  o secvență de numere naturale.

Să se construiască, dacă se poate, un graf neorientat  $G$  cu  
 $s(G) = s_0$ .

## Condiții necesare pentru existența lui $G$ :

- $d_1 + \dots + d_n$  – număr par
- $d_i \leq n - 1, \forall i$

# Construcția de grafuri cu secvența gradelor dată

## Problemă

Fie  $s_0 = \{d_1, \dots, d_n\}$  o secvență de numere naturale.

Să se construiască, dacă se poate, un graf neorientat  $G$  cu  
 $s(G) = s_0$ .

## Condiții necesare pentru existența lui $G$ :

- $d_1 + \dots + d_n$  – număr par
- $d_i \leq n - 1, \forall i$



Pentru  $s_0 = \{3, 3, 1, 1\}$  – nu există  $G$

$\Rightarrow$  **condițiile nu sunt și suficiente**

# Construcția de grafuri cu secvența gradelor dată



Idee algoritm de construcție a unui graf  $G$  cu  $s(G) = s_0$

- începem construcția de la vârful cu gradul cel mai mare
- îi alegem ca vecini vârfurile cu gradele cele mai mari

# Construcția de grafuri cu secvența gradelor dată



Idee algoritm de construcție a unui graf  $G$  cu  $s(G) = s_0$

- începem construcția de la vârful cu gradul cel mai mare
- îi alegem ca vecini vârfurile cu gradele cele mai mari
- actualizăm secvența  $s_0$  și reluăm până când
  - secvența conține doar 0  $\Rightarrow G$
  - secvența conține numere negative  $\Rightarrow$

# Construcția de grafuri cu secvența gradelor dată

Idee algoritm de construcție a unui graf  $G$  cu  $s(G) = s_0$

- începem construcția de la vârful cu gradul cel mai mare
- îi alegem ca vecini vârfurile cu gradele cele mai mari
- actualizăm secvența  $s_0$  și reluăm până când
  - secvența conține doar 0  $\Rightarrow G$
  - secvența conține numere negative  $\Rightarrow$

**$G$  nu se poate construi prin acest procedeu**



**Se poate construi  $G$  altfel?**

# Construcția de grafuri cu secvența gradelor dată

Idee algoritm de construcție a unui graf  $G$  cu  $s(G) = s_0$

- începem construcția de la vârful cu gradul cel mai mare
- îi alegem ca vecini vârfurile cu gradele cele mai mari
- actualizăm secvența  $s_0$  și reluăm până când
  - secvența conține doar 0  $\Rightarrow G$
  - secvența conține numere negative  $\Rightarrow$

**$G$  nu se poate construi prin acest procedeu**



**Teorema Havel–Hakimi  $\Rightarrow$  NU**

**$\Rightarrow$  Algoritmul anterior = Algoritmul Havel–Havimi**

# Exemplu algoritm Havel–Hakimi

$$s_0 = \{ 3, 4, 2, 1, 3, 4, 1, 2 \}$$

etichete vârfuri  $x_1$   $x_2$   $x_3$   $x_4$   $x_5$   $x_6$   $x_7$   $x_8$

- Pasul 1** – construim muchii pentru vârful de gradul maxim =  $x_2$
- alegem ca vecini următoarele vârfuri cu cele mai mari grade

# Exemplu algoritm Havel–Hakimi

$$s_0 = \{ 3, 4, 2, 1, 3, 4, 1, 2 \}$$

etichete vârfuri  $x_1$   $x_2$   $x_3$   $x_4$   $x_5$   $x_6$   $x_7$   $x_8$

- Pasul 1** – construim muchii pentru vârful de gradul maxim =  $x_2$
- alegem ca vecini următoarele vârfuri cu cele mai mari grade
- ⇒ ar fi utilă sortarea descrescătoare a elementelor lui  $s_0$

$$s_0 = \{ 4, 4, 3, 3, 2, 2, 1, 1 \}$$

etichete vârfuri  $x_2$   $x_6$   $x_1$   $x_5$   $x_3$   $x_8$   $x_4$   $x_7$



# Exemplu algoritm Havel–Hakimi

Pasul 1.

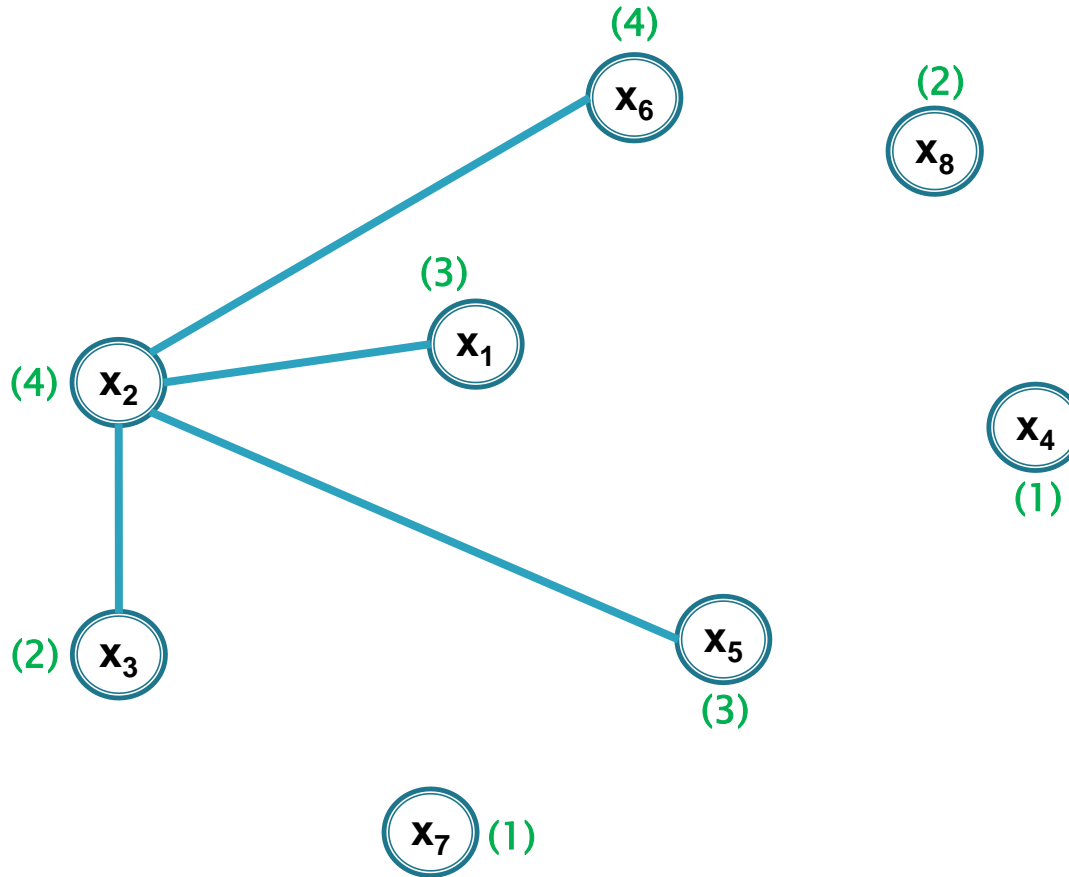
$$s_0 = \{ 4, 4, 3, 3, 2, 2, 1, 1 \}$$

etichete vârfuri  $x_2$   $x_6$   $x_1$   $x_5$   $x_3$   $x_8$   $x_4$   $x_7$



► **Muchii construite:**  $x_2x_6, x_2x_1, x_2x_5, x_2x_3$

# Exemplu algoritm Havel–Hakimi



# Exemplu algoritm Havel–Hakimi

Pasul 1.

$$s_0 = \{ 4, 4, 3, 3, 2, 2, 1, 1 \}$$

etichete vârfuri  $x_2$   $x_6$   $x_1$   $x_5$   $x_3$   $x_8$   $x_4$   $x_7$

► **Muchii construite:**  $x_2x_6, x_2x_1, x_2x_5, x_2x_3$

► **Secvența rămasă:**

$$s'_0 = \{ \quad 3, \quad 2, \quad 2, \quad 1, \quad 2, \quad 1, \quad 1 \}$$

etichete vârfuri  $x_6$   $x_1$   $x_5$   $x_3$   $x_8$   $x_4$   $x_7$

# Exemplu algoritm Havel–Hakimi

Pasul 1.

$$s_0 = \{ 4, 4, 3, 3, 2, 2, 1, 1 \}$$

etichete vârfuri  $x_2$   $x_6$   $x_1$   $x_5$   $x_3$   $x_8$   $x_4$   $x_7$

► Muchii construite:  $x_2x_6, x_2x_1, x_2x_5, x_2x_3$

► Secvența rămasă:

$$s'_0 = \{ 3, 2, 2, 1, 2, 1, 1 \}$$

etichete vârfuri  $x_6$   $x_1$   $x_5$   $x_3$   $x_8$   $x_4$   $x_7$

Secvența rămasă ordonată descrescător:


$$s'_0 = \{ 3, 2, 2, 2, 1, 1, 1 \}$$

etichete vârfuri  $x_6$   $x_1$   $x_5$   $x_8$   $x_3$   $x_4$   $x_7$

# Exemplu algoritm Havel–Hakimi

Pasul 2.

$s'_0 = \{ \quad 3, \quad 2, \quad 2, \quad 2, \quad 1, \quad 1, \quad 1 \}$   
etichete vârfuri  $x_6 \quad x_1 \quad x_5 \quad x_8 \quad x_3 \quad x_4 \quad x_7$



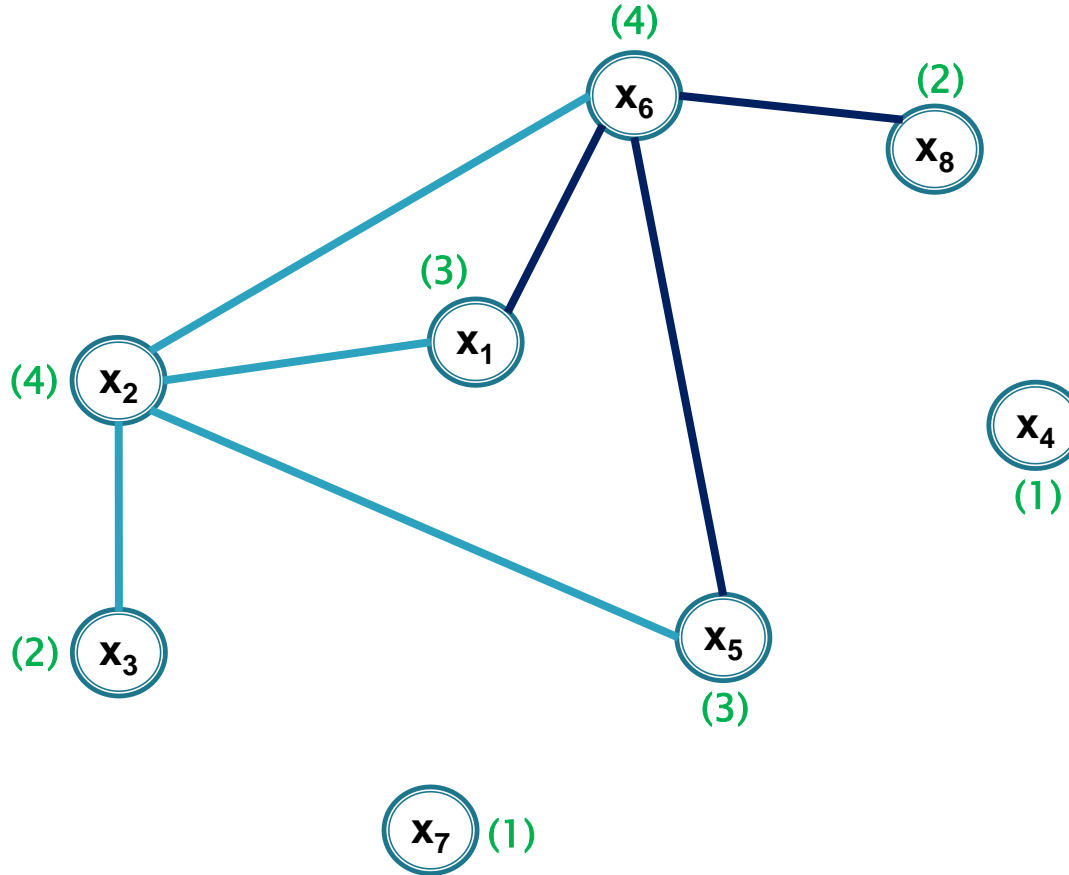
► Muchii construite:  $x_6x_1, x_6x_5, x_6x_8$

► Secvența rămasă:

$s''_0 = \{ \quad 1, \quad 1, \quad 1, \quad 1, \quad 1, \quad 1 \}$   
etichete vârfuri  $x_1 \quad x_5 \quad x_8 \quad x_3 \quad x_4 \quad x_7$

(este ordonată descrescător)

# Exemplu algoritm Havel–Hakimi



# Exemplu algoritm Havel–Hakimi

Pasul 3.

$s''_0 = \{$                       1,   1,   1,   1,   1,   1}  
etichete vârfuri             $x_1$   $x_5$     $x_8$     $x_3$     $x_4$     $x_7$

► Muchii construite:  $x_1 x_5$

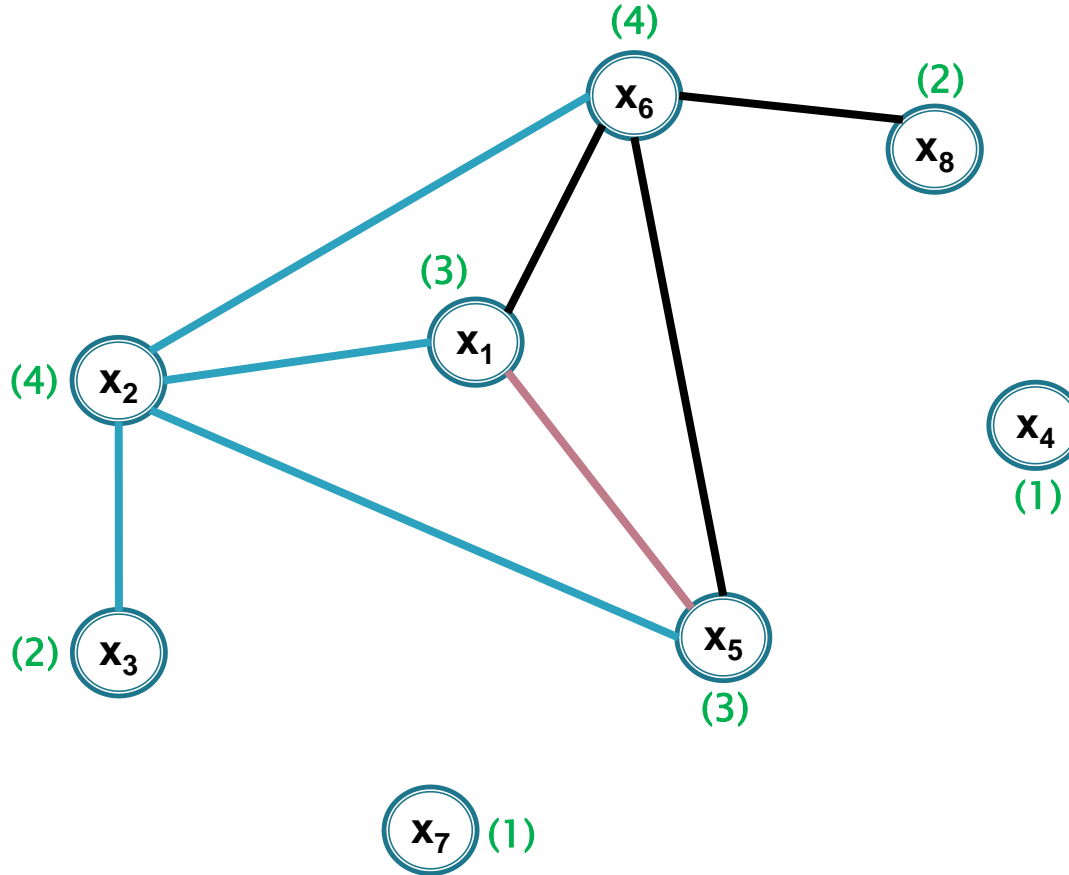
► Secvența rămasă:

$s'''_0 = \{$                       0,   1,   1,   1,   1}  
etichete vârfuri             $x_5$     $x_8$     $x_3$     $x_4$     $x_7$

Secvența rămasă ordonată descrescător:

$s'''_0 = \{$                       1,   1,   1,   1,   0}  
etichete vârfuri             $x_7$     $x_3$     $x_4$     $x_8$     $x_5$

# Exemplu algoritm Havel–Hakimi





# Exemplu algoritm Havel–Hakimi

Pasul 4.

$s'''_0 = \{$   
etichete vârfuri

1,	1,	1,	1,	0}
$x_7$	$x_3$	$x_4$	$x_8$	$x_5$



► Muchii construite:  $x_7x_3$

► Secvența rămasă:

$s^{iv}_0 = \{$   
etichete vârfuri

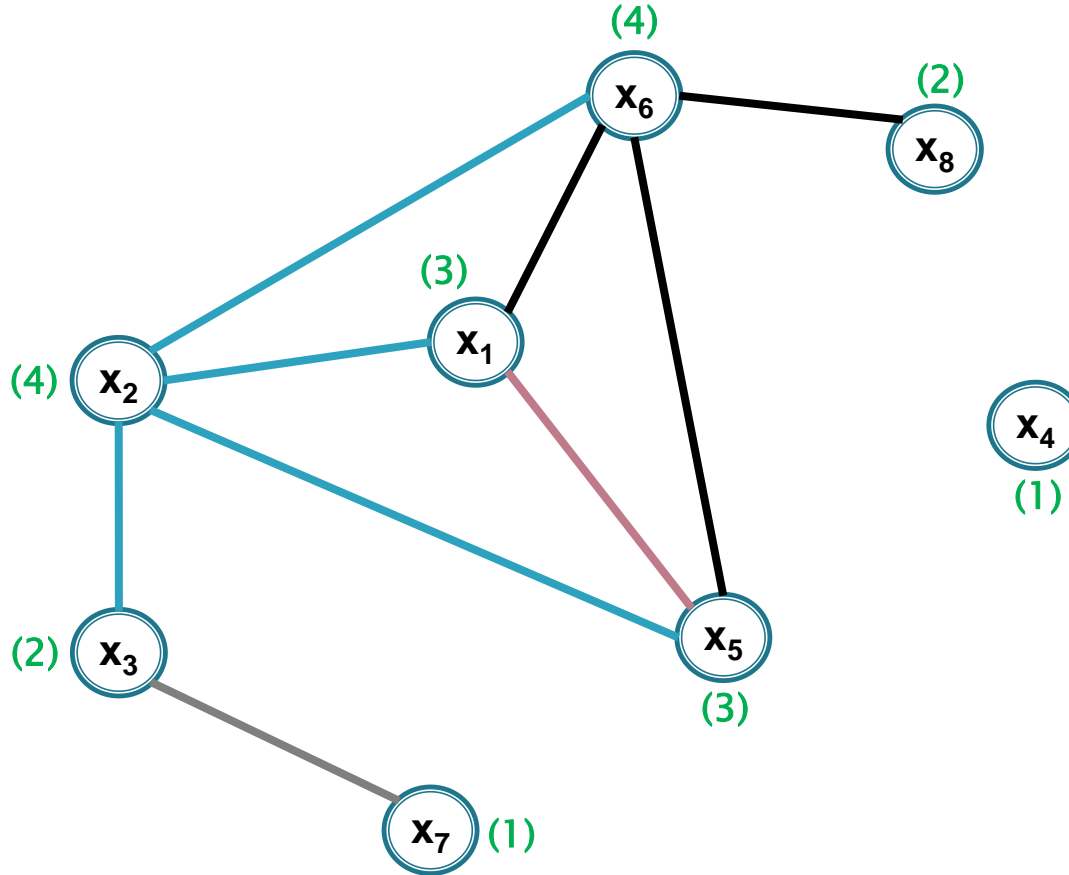
0,	1,	1,	0}
$x_3$	$x_4$	$x_8$	$x_5$

Secvența rămasă ordonată descrescător:

$s'''_0 = \{$   
etichete vârfuri

1,	1,	0,	0}
$x_4$	$x_8$	$x_3$	$x_5$

# Exemplu algoritm Havel–Hakimi




# Exemplu algoritm Havel–Hakimi

Pasul 5.

$s^{iv}_0 = \{$   
etichete vârfuri

1, 1, 0, 0}  
 $x_4$   $x_8$   $x_3$   $x_5$



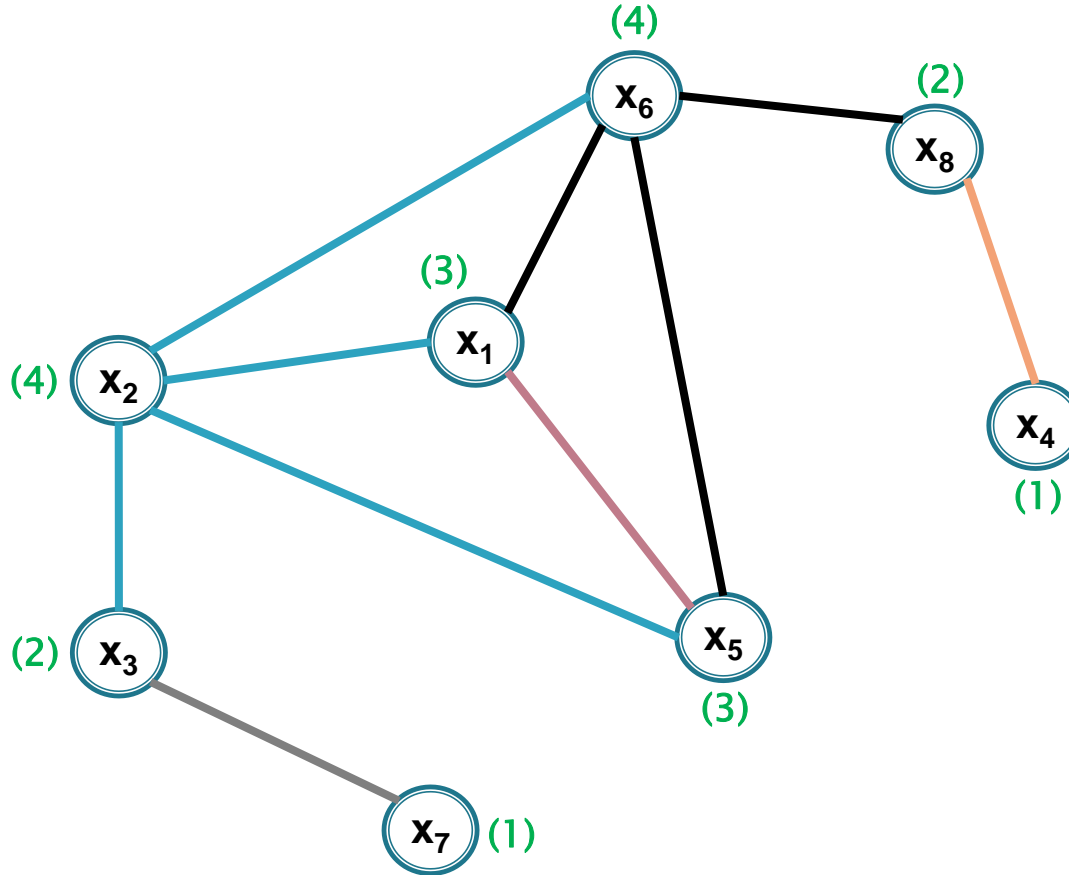
- ▶ Muchii construite:  $x_4x_8$
- ▶ Secvența rămasă:

$s^{iv}_0 = \{$   
etichete vârfuri

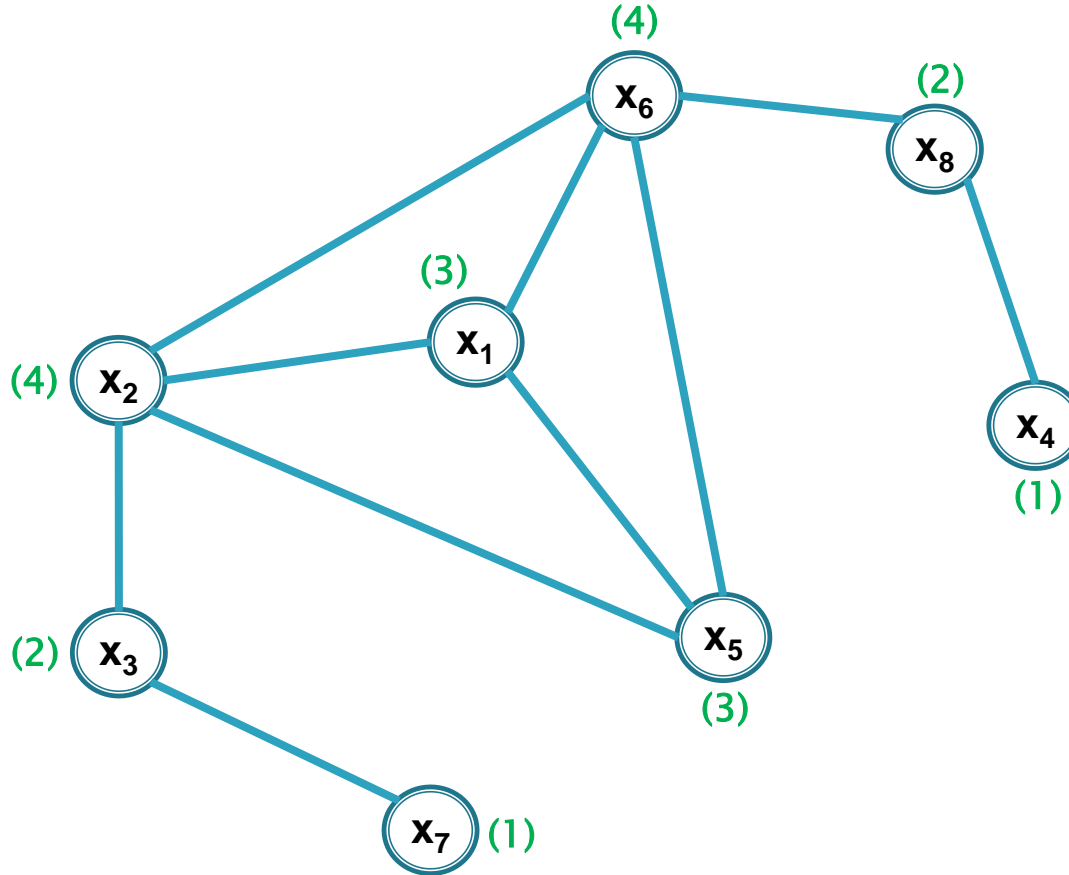
0, 0, 0}  
 $x_8$   $x_3$   $x_5$

**STOP**

# Exemplu algoritm Havel–Hakimi



# Exemplu algoritm Havel–Hakimi



# Algorithm Havel–Hakimi

1. Dacă  $d_1 + \dots + d_n$  este impar sau există în  $s_0$  un  $d_i > n-1$ , atunci scrie NU, STOP.
2. cât timp  $s_0$  conține valori nenule execută
  - alege  $d_k$  **cel mai mare număr** din secvența  $s_0$
  - elimină  $d_k$  din  $s_0$
  - fie  $d_{i_1}, \dots, d_{i_{d_k}}$  **cele mai mari  $d_k$  numere** din  $s_0$

# Algorithm Havel–Hakimi

1. Dacă  $d_1 + \dots + d_n$  este impar sau există în  $s_0$  un  $d_i > n-1$ , atunci scrie NU, STOP.
2. cât timp  $s_0$  conține valori nenule execută
  - alege  $d_k$  **cel mai mare număr** din secvența  $s_0$
  - elimină  $d_k$  din  $s_0$
  - fie  $d_{i_1}, \dots, d_{i_{d_k}}$  **cele mai mari  $d_k$  numere** din  $s_0$
  - pentru  $j \in \{i_1, \dots, i_{d_k}\}$  execută:

# Algorithm Havel–Hakimi

1. Dacă  $d_1 + \dots + d_n$  este impar sau există în  $s_0$  un  $d_i > n-1$ , atunci scrie NU, STOP.
2. cât timp  $s_0$  conține valori nenule execută
  - alege  $d_k$  **cel mai mare număr** din secvența  $s_0$
  - elimină  $d_k$  din  $s_0$
  - fie  $d_{i_1}, \dots, d_{i_{d_k}}$  **cele mai mari  $d_k$  numere** din  $s_0$
  - pentru  $j \in \{i_1, \dots, i_{d_k}\}$  execută:
    - adaugă la  $G$  muchia  $x_k x_j$
    - înlocuiește  $d_j$  în secvența  $s_0$  cu  $d_j - 1$
    - dacă  $d_j - 1 < 0$ , atunci scrie NU, STOP.



# Algorithm Havel–Hakimi

1. Dacă  $d_1 + \dots + d_n$  este impar sau există în  $s_0$  un  $d_i > n-1$ , atunci scrie NU, STOP.
2. cât timp  $s_0$  conține valori nenule execută
  - alege  $d_k$  **cel mai mare număr** din secvența  $s_0$
  - elimină  $d_k$  din  $s_0$
  - fie  $d_{i_1}, \dots, d_{i_{d_k}}$  **cele mai mari  $d_k$  numere** din  $s_0$
  - pentru  $j \in \{i_1, \dots, i_{d_k}\}$  execută:
    - adaugă la  $G$  muchia  $x_k x_j$
    - înlocuiește  $d_j$  în secvența  $s_0$  cu  $d_j - 1$
    - dacă  $d_j - 1 < 0$ , atunci scrie NU, STOP.

**Observație.** Pentru a determina ușor care este cel mai mare număr din secvență și care sunt cele mai mari valori care îi urmează, **este util ca pe parcursul algoritmului secvența  $s_0$  să fie ordonată descrescător.**

**Complexitate?**

# Algoritm Havel–Hakimi – Corectitudine

## Teorema Havel–Hakimi

O secvență de  $n \geq 2$  numere naturale

$$s_0 = \{d_1 \geq \dots \geq d_n\}$$

cu  $d_1 \leq n-1$  este secvența gradelor unui graf neorientat (cu  $n$  vârfuri)

$\Leftrightarrow$  secvența

$$s'_0 = \{d_2 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n\}$$

este secvența gradelor unui graf neorientat (cu  $n-1$  vârfuri).

# Algoritm Havel–Hakimi – Corectitudine

## Teorema Havel–Hakimi

O secvență de  $n \geq 2$  numere naturale

$$s_0 = \{d_1 \geq \dots \geq d_n\}$$

cu  $d_1 \leq n-1$  este secvența gradelor unui graf neorientat (cu  $n$  vârfuri)

$\Leftrightarrow$  secvența

$$s'_0 = \{d_2 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n\}$$

este secvența gradelor unui graf neorientat (cu  $n-1$  vârfuri).

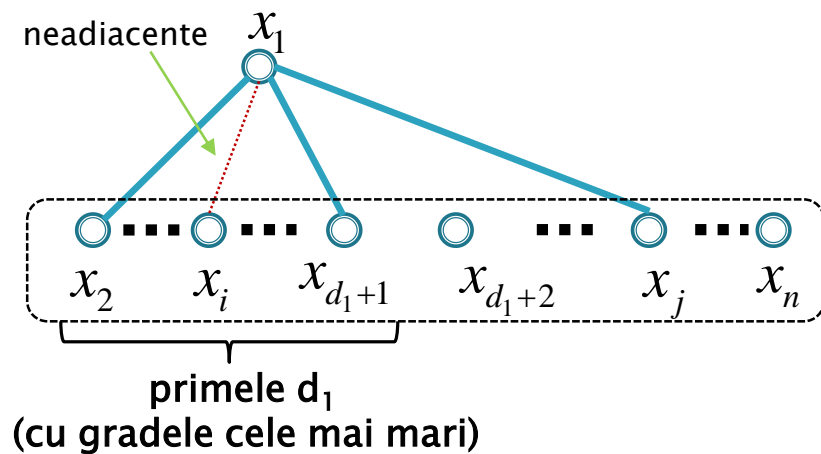
**Observație:** Secvența  $s'_0 = \{d_2 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n\}$  se obține din  $s_0$  **eliminând primul element (adică  $d_1$ )** și scăzând 1 din primele  $d_1$  elemente rămase – acestea au indicii **2, 3, ...,  $d_1+1$**

# Algoritm Havel–Hakimi – Corectitudine

## Teorema Havel–Hakimi – Demonstrație

$$s_0 = \{d_1 \geq \dots \geq d_n\} \Rightarrow s'_0 = \{d_2 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n\}$$

$$G, s(G) = s_0$$

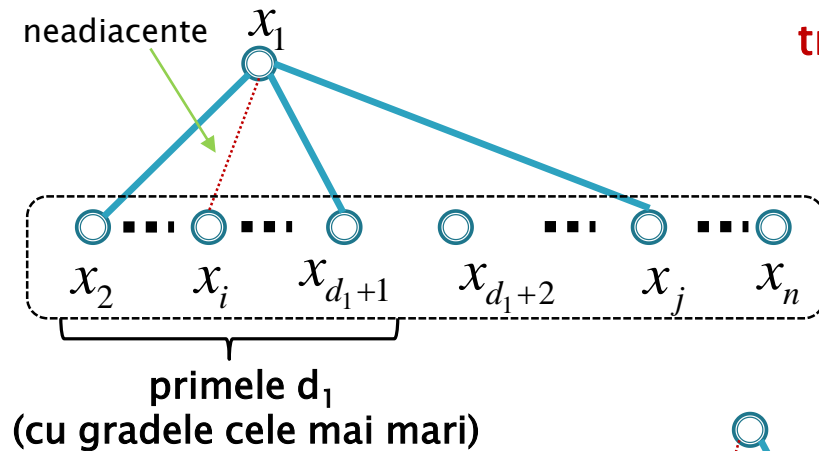


# Algoritm Havel-Hakimi – Corectitudine

## Teorema Havel-Hakimi – Demonstrație

$$s_0 = \{d_1 \geq \dots \geq d_n\} \Rightarrow s'_0 = \{d_2 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n\}$$

$$G, s(G) = s_0$$

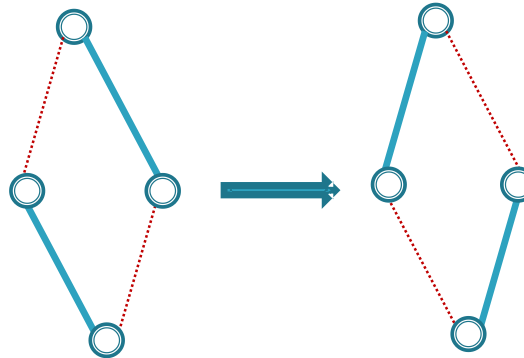
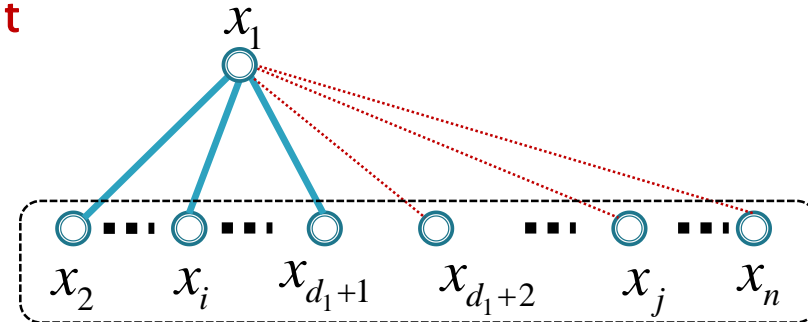


transformare  $t$   
pe pătrat



$$G^*, s(G^*) = s'_0$$

$$N_{G^*}(x_1) = \{x_2, \dots, x_{d_1+1}\}$$

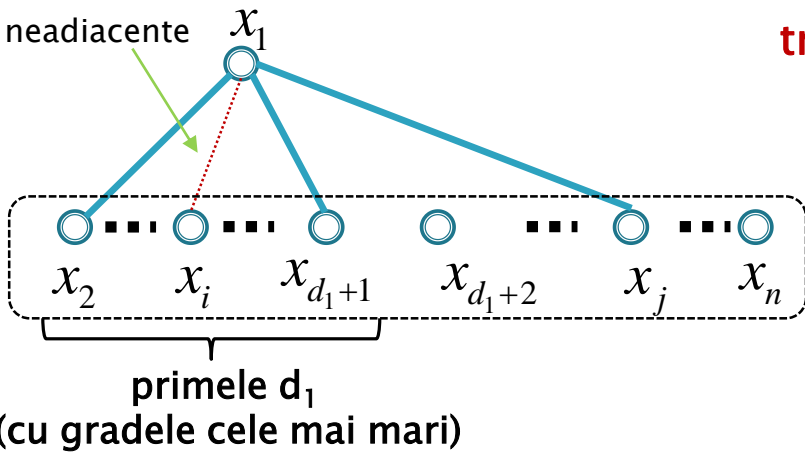


# Algoritm Havel–Hakimi – Corectitudine

## Teorema Havel–Hakimi – Demonstrație

$$s_0 = \{d_1 \geq \dots \geq d_n\} \Rightarrow s'_0 = \{d_2 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n\}$$

$$G, s(G) = s_0$$

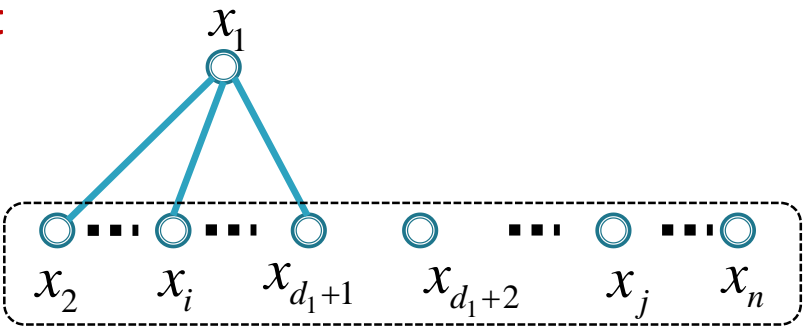


transformare  $t$   
pe pătrat

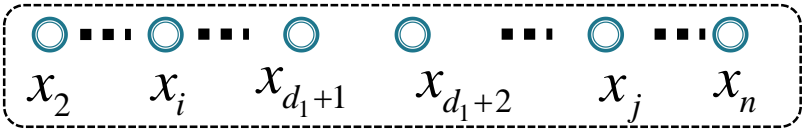


$$G^*, s(G^*) = s_0$$

$$N_{G^*}(x_1) = \{x_2, \dots, x_{d_1+1}\}$$



elimin  $x_1$

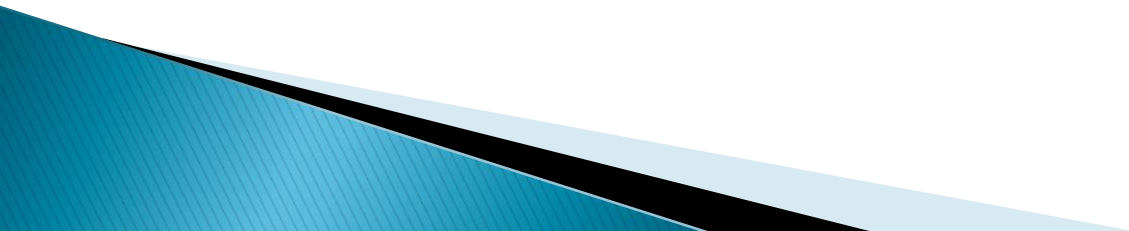


$$G' = G^* - x_1, s(G') = s'_0$$

# Algoritm Havel–Hakimi – Corectitudine

## Teorema Havel–Hakimi – Demonstrație

$$s_0 = \{d_1 \geq \dots \geq d_n\} \iff s'_0 = \{d_2 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n\}$$

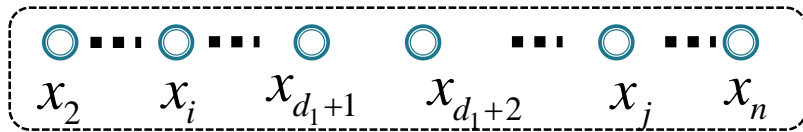


# Algoritm Havel–Hakimi – Corectitudine

## Teorema Havel–Hakimi – Demonstrație

$$s_0 = \{d_1 \geq \dots \geq d_n\} \iff s'_0 = \{d_2 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n\}$$

Fie  $G'$  cu  $s(G') = s'_0 = \{d_2 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n\}$





# Algoritm Havel–Hakimi – Corectitudine

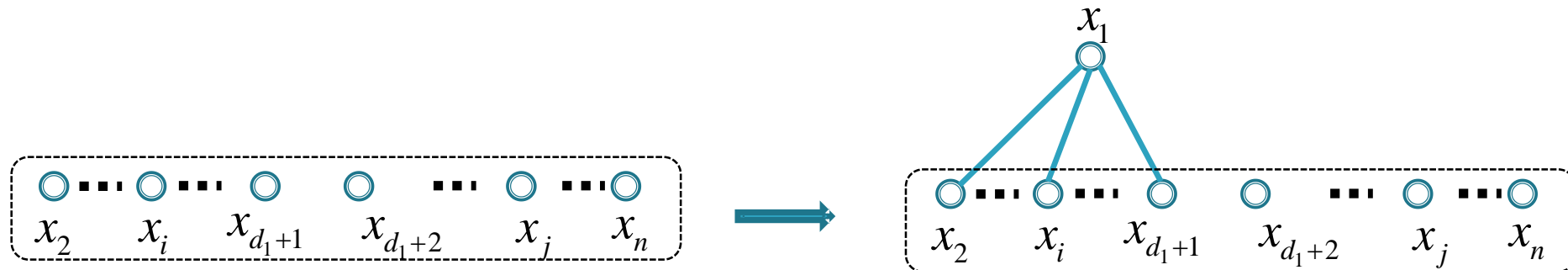
## Teorema Havel–Hakimi – Demonstrație

$$s_0 = \{d_1 \geq \dots \geq d_n\} \iff s'_0 = \{d_2 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n\}$$

Fie  $G'$  cu  $s(G') = s'_0$

$$G: V(G) = V(G') \cup \{x_1\}$$

$$E(G) = E(G') \cup \{x_1x_2, \dots, x_1x_{d_1+1}\}$$



adăugăm un vârf  $x_1$   
pe care îl unim cu  
 $x_2, \dots, x_{d_1+1}$

Avem  $s(G) = s_0$ .

# Algoritm Havel–Hakimi – Corectitudine

## Teorema Havel–Hakimi



Unde intervine în demonstrație faptul că  $d_1$  este maxim?

# Algoritm Havel–Hakimi – Corectitudine

## Teorema Havel–Hakimi

Unde intervine în demonstrație faptul că  $d_1$  este maxim?



Se poate renunța la această ipoteză  $\Rightarrow$

Extindere a Teoremei Havel–Hakimi

# Algoritm Havel–Hakimi – Corectitudine

## Teorema Havel–Hakimi

Unde intervine în demonstrație faptul că  $d_1$  este maxim?



**Se poate renunța la această ipoteză  $\Rightarrow$**

## Extindere a Teoremei Havel–Hakimi

Fie  $s_0 = \{d_1, \dots, d_n\}$  o secvență de  $n \geq 2$  numere naturale cu mai mici sau egale cu  $n-1$  și fie  $i \in \{1, \dots, n\}$  fixat. Fie  $s_0^{(i)}$  secvența obținută din  $s_0$  astfel:

# Algoritm Havel–Hakimi – Corectitudine

## Teorema Havel–Hakimi

Unde intervine în demonstrație faptul că  $d_1$  este maxim?



**Se poate renunța la această ipoteză  $\Rightarrow$**

## Extindere a Teoremei Havel–Hakimi

Fie  $s_0 = \{d_1, \dots, d_n\}$  o secvență de  $n \geq 2$  numere naturale cu mai mici sau egale cu  $n-1$  și fie  $i \in \{1, \dots, n\}$  fixat. Fie  $s_0^{(i)}$  secvența obținută din  $s_0$  astfel:

- eliminăm elementul  $d_i$

# Algoritm Havel–Hakimi – Corectitudine

## Teorema Havel–Hakimi

Unde intervine în demonstrație faptul că  $d_1$  este maxim?



**Se poate renunța la această ipoteză  $\Rightarrow$**

## Extindere a Teoremei Havel–Hakimi

Fie  $s_0 = \{d_1, \dots, d_n\}$  o secvență de  $n \geq 2$  numere naturale cu mai mici sau egale cu  $n-1$  și fie  $i \in \{1, \dots, n\}$  fixat. Fie  $s_0^{(i)}$  secvența obținută din  $s_0$  astfel:

- eliminăm elementul  $d_i$
- scădem o unitate din primele  $d_i$  componente în ordine descrescătoare ale secvenței rămase.

# Algoritm Havel–Hakimi – Corectitudine

## Teorema Havel–Hakimi

Unde intervine în demonstrație faptul că  $d_1$  este maxim?



**Se poate renunța la această ipoteză  $\Rightarrow$**

## Extindere a Teoremei Havel–Hakimi

Fie  $s_0 = \{d_1, \dots, d_n\}$  o secvență de  $n \geq 2$  numere naturale cu mai mici sau egale cu  $n-1$  și fie  $i \in \{1, \dots, n\}$  fixat. Fie  $s_0^{(i)}$  secvența obținută din  $s_0$  astfel:

- eliminăm elementul  $d_i$
- scădem o unitate din primele  $d_i$  componente în ordine descrescătoare ale secvenței rămase.

Are loc echivalența:

$s_0$  este secvența gradelor unui graf neorientat  $\Leftrightarrow$   
 $s_0^{(i)}$  este secvența gradelor unui graf neorientat

# Algoritm Havel–Hakimi – Corectitudine

## Teorema Havel–Hakimi

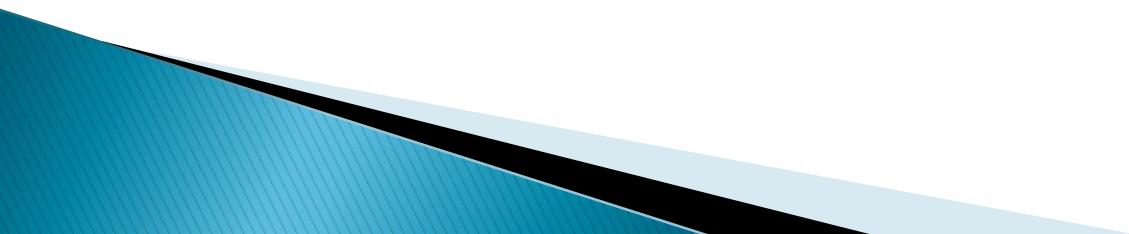
Unde intervine în demonstrație faptul că  $d_1$  este maxim?

Se poate renunța la această ipoteză  $\Rightarrow$

## Extindere a Algoritmului Havel–Hakimi

La un pas vârful poate fi ales arbitrar (nu neapărat cel corespunzător elementului maxim).

Se păstrează însă criteriul de alegere al vecinilor (cu gradele cele mai mari)





# Construcția de grafuri cu secvența gradelor dată

- ▶ Cu ajutorul transformării  $t$  pe pătrat putem obține pornind de la un graf  $G$  toate grafurile cu secvența gradelor  $s(G)$  (și mulțimea vârfurilor  $V(G)$ )



# Construcția de grafuri cu secvența gradelor dată

- ▶ Cu ajutorul transformării  $t$  pe pătrat putem obține pornind de la un graf  $G$  toate grafurile cu secvența gradelor  $s(G)$  (și mulțimea vârfurilor  $V(G)$ )

- ▶ Mai exact, ar loc următorul rezultat (exercițiu):

Fie  $G_1$  și  $G_2$  două grafuri neorientate cu mulțimea vârfurilor  $V=\{1, \dots, n\}$ .

Atunci  $s(G_1)=s(G_2) \Leftrightarrow$  există un șir de transformări  $t$  de interschimbare pe pătrat prin care se poate obține graful  $G_2$  din  $G_1$ .



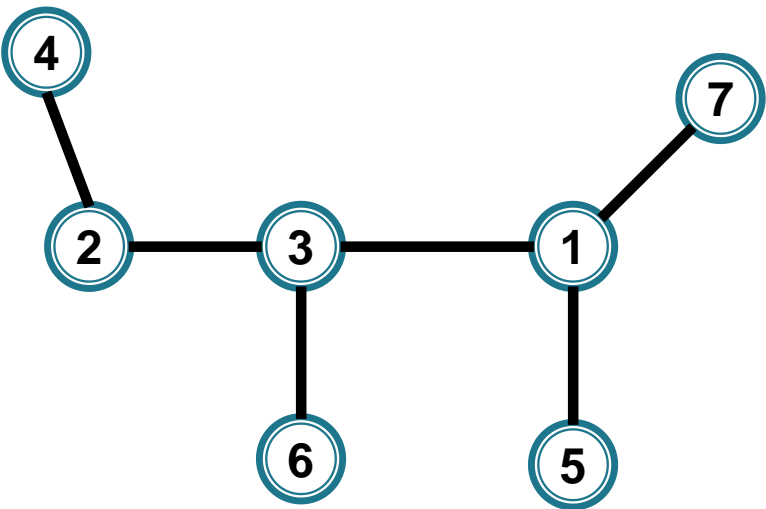
# Construcția de grafuri cu secvența gradelor dată

## Teorema Erdős – Gallai (suplimentar)

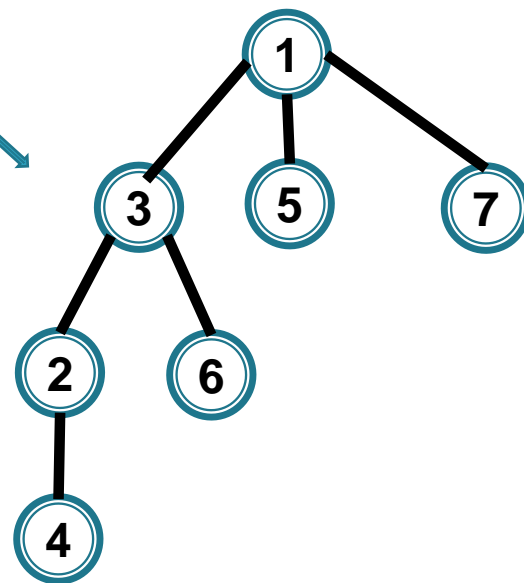
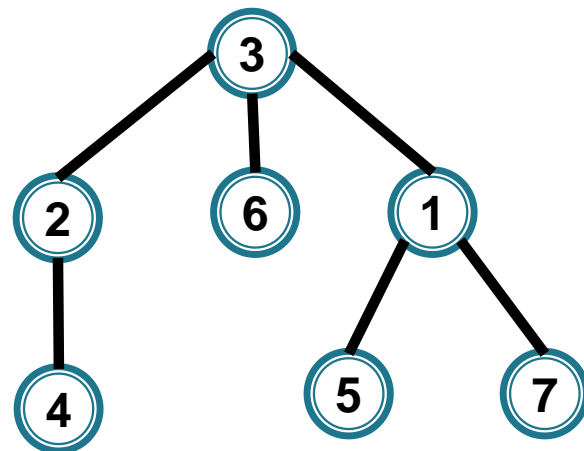
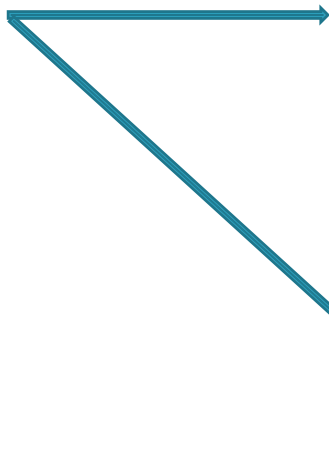
O secvență de  $n \geq 2$  numere naturale  $s_0 = \{d_1 \geq \dots \geq d_n\}$  este secvența gradelor unui graf neorientat  $\Leftrightarrow$

- $d_1 + \dots + d_n$  par și
- $d_1 + \dots + d_k \leq k(k-1) + \sum_{i=k+1}^n \min\{d_i, k\}, \forall 1 \leq k \leq n$

# Arbori



Arbore = conex + aciclic



Arbore cu rădăcină

# Arbori

- **Arbore** = graf neorientat **conex** și **aciclic**

# Arbori

➤ **Arbore** = graf neorientat **conex** și **aciclic**

➤ **Arbori filogenetici** – ilustrează evoluții

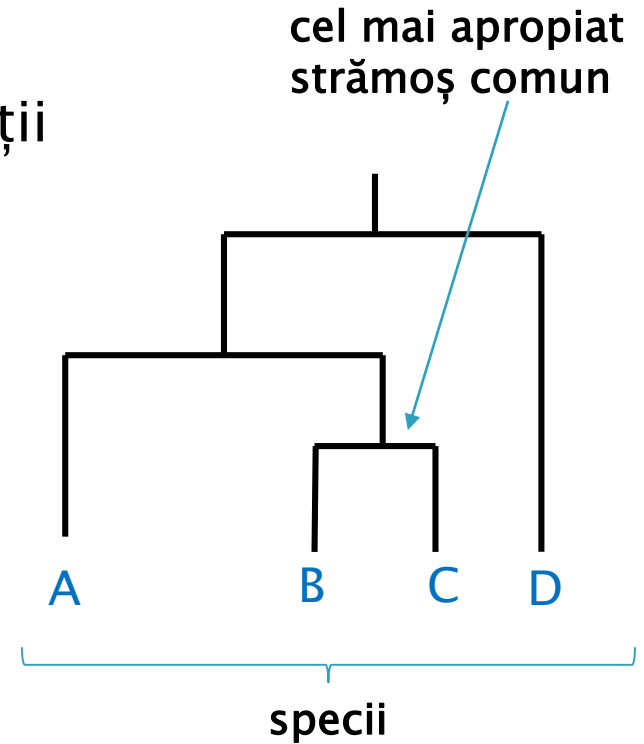
➤ **Arbori de dependențe, de joc**

➤ **Probleme de rutare**

➤ **Arbori aleatorii**

➤ **Arbori economici (cu costul minim)**

➤ **Structuri de date...**



# Arbori

## Leme

1. Orice arbore  $T$  cu  $n > 1$  are cel puțin două vârfuri terminale (de grad 1)



# Arbori

## Leme

1. Orice arbore  $T$  cu  $n > 1$  are cel puțin două vârfuri terminale (de grad 1)



Fie  $P$  un lanț elementar maxim în  $T$

Extremitățile lui  $P$  sunt vârfuri terminale, altfel:

# Arbori

## Leme

1. Orice arbore  $T$  cu  $n > 1$  are cel puțin două vârfuri terminale (de grad 1)



Fie  $P$  un lanț elementar maxim în  $T$

Extremitățile lui  $P$  sunt vârfuri terminale, altfel:

– putem extinde lanțul cu o muchie 

sau

# Arbori

## Leme

1. Orice arbore  $T$  cu  $n > 1$  are cel puțin două vârfuri terminale (de grad 1)



Fie  $P$  un lanț elementar maxim în  $T$

Extremitățile lui  $P$  sunt vârfuri terminale, altfel:

– putem extinde lanțul cu o muchie



sau

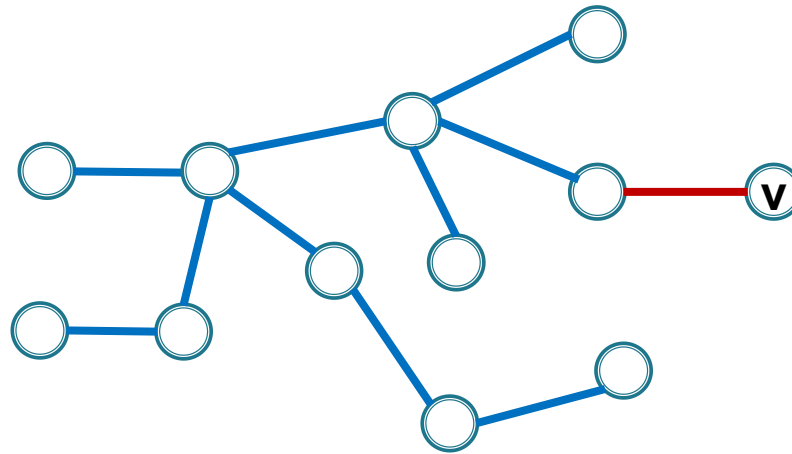
– se închide un ciclu în  $T$



# Arbori

## Leme

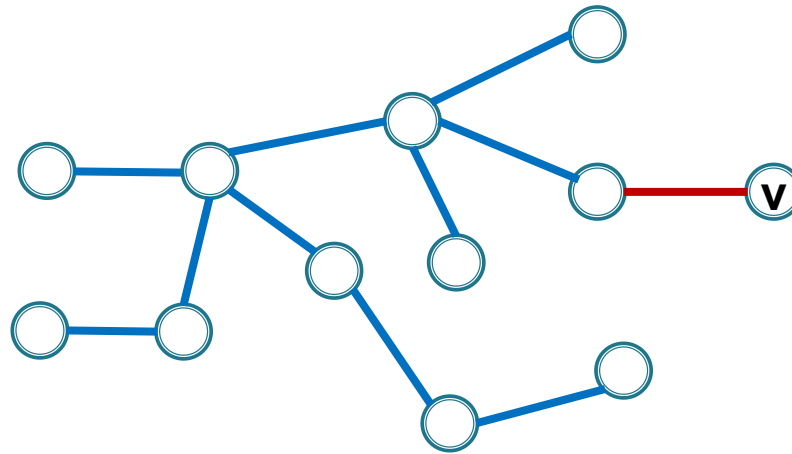
2. Fie  $T$  un arbore cu  $n > 1$  vârfuri și  $v$  un vârf terminal în  $T$ .  
Atunci  $T - v$  este arbore.



# Arbori

## Leme

2. Fie  $T$  un arbore cu  $n > 1$  vârfuri și  $v$  un vârf terminal în  $T$ .  
Atunci  $T - v$  este arbore.



Rezultă din definiția conexității + un vârf terminal nu poate fi vârf intern al unui lanț elementar

# Arbori

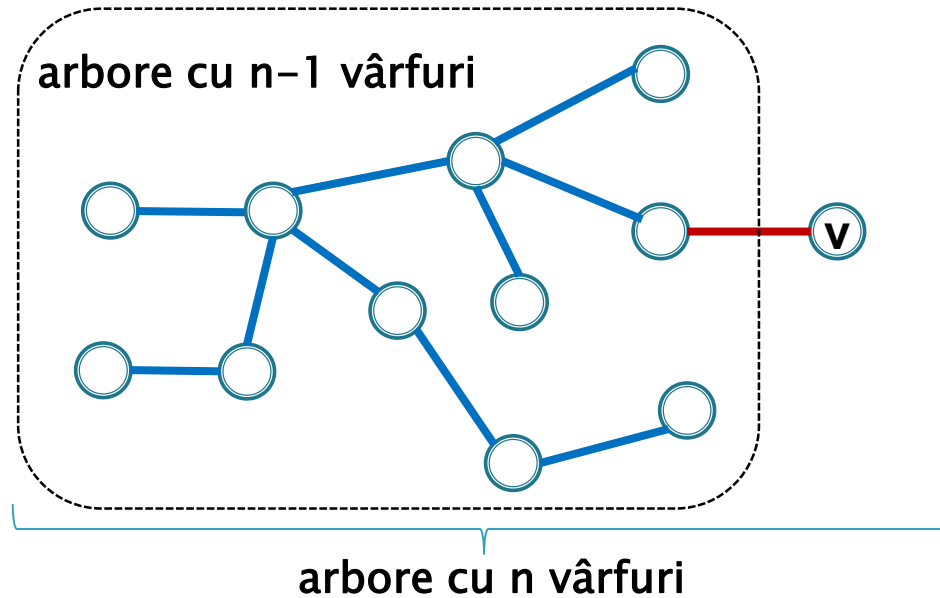
## Leme

3. Un arbore cu  $n$  vârfuri are  $n-1$  muchii.

# Arbori

## Leme

3. Un arbore cu  $n$  vârfuri are  $n-1$  muchii.

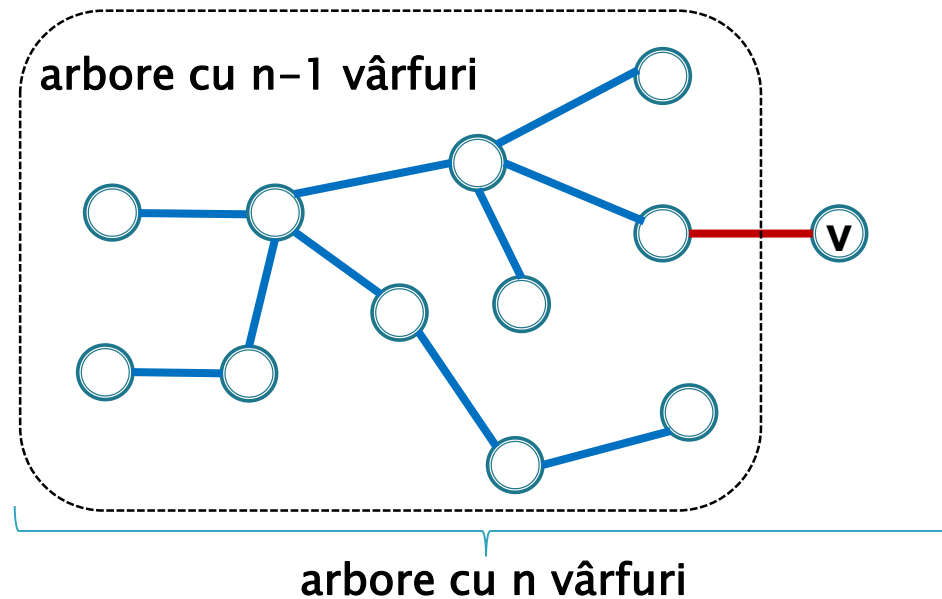


Inducție după  $n$

# Arbori

## Leme

3. Un arbore cu  $n$  vârfuri are  $n-1$  muchii.



### Inducție după $n$

- Dacă  $T$  este un arbore cu  $n$  vârfuri și  $v$  este vârf terminal în  $T$ , atunci  $T - v$  este arbore cu  $n-1$  vârfuri și  $|E(T-v)| = |E(T)| - 1$
- Aplicăm ipoteza de inducție pentru  $T-v$



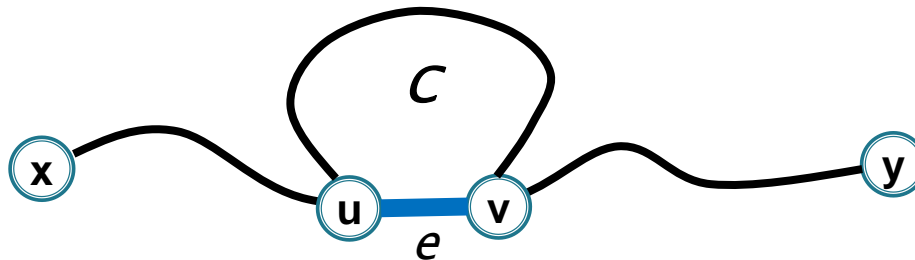
# Arbori

## Observație

Fie  $G$  un graf neorientat conex și  $C$  un ciclu în  $G$ .

Fie  $e \in E(C)$  o muchie din ciclul  $C$ .

Atunci  $G - e$  este tot un graf conex.



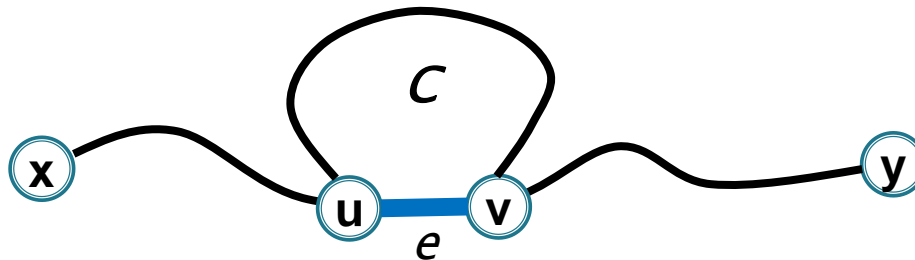
# Arbori

## Observație

Fie  $G$  un graf neorientat conex și  $C$  un ciclu în  $G$ .

Fie  $e \in E(C)$  o muchie din ciclul  $C$ .

Atunci  $G-e$  este tot un graf conex.



Rezultă din definiția conexității + observația:

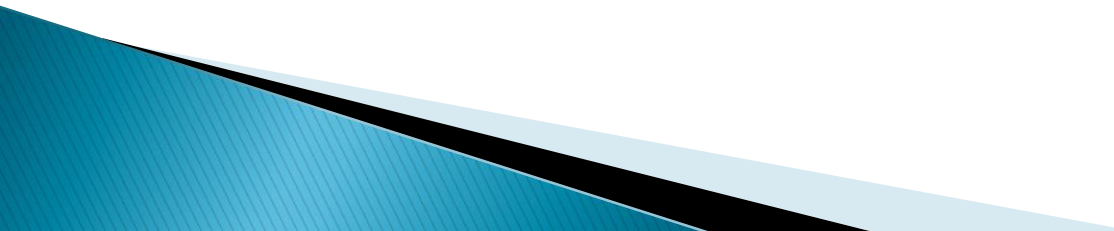
- dintr-un  $x$ - $y$  lanț în  $G$  care conține muchia  $e$  se poate obține un  $x$ - $y$  lanț în  $G-e$  înlocuind muchia  $e$  cu lanțul  $C-e$ .

# Arbori

## Definiții echivalente

Fie  $T$  un graf neorientat cu  $n > 1$  vârfuri.

Următoarele afirmații sunt echivalente.

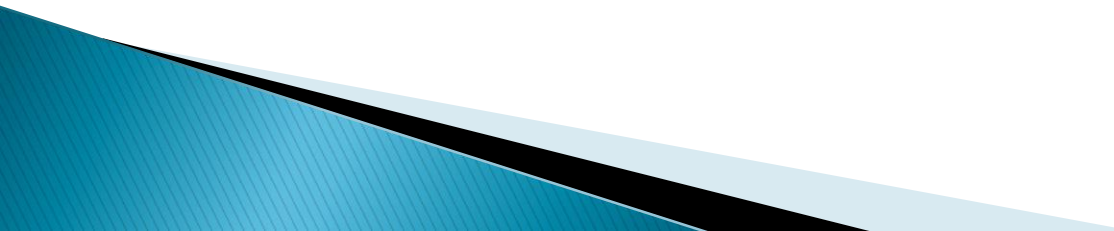
1.  $T$  este arbore (conex și aciclic)
  - 2.
  - 3.
  - 4.
  - 5.
  - 6.
- 

# Arbori

## Definiții echivalente

Fie  $T$  un graf neorientat cu  $n > 1$  vârfuri.

Următoarele afirmații sunt echivalente.

1.  $T$  este arbore (conex și aciclic)
  2.  $T$  este conex muchie-minimal
  3.  $T$  este aciclic muchie-maximal
  - 4.
  - 5.
  - 6.
- 

# Arbori

## Definiții echivalente

Fie  $T$  un graf neorientat cu  $n > 1$  vârfuri.

Următoarele afirmații sunt echivalente.

1.  $T$  este arbore (conex și aciclic)
2.  $T$  este conex muchie-minimal
3.  $T$  este aciclic muchie-maximal
4.  $T$  este conex și are  $n-1$  muchii
5.  $T$  este aciclic și are  $n-1$  muchii
- 6.

# Arbori

## Definiții echivalente

Fie  $T$  un graf neorientat cu  $n > 1$  vârfuri.

Următoarele afirmații sunt echivalente.

1.  $T$  este arbore (conex și aciclic)
2.  $T$  este conex muchie-minimal
3.  $T$  este aciclic muchie-maximal
4.  $T$  este conex și are  $n-1$  muchii
5.  $T$  este aciclic și are  $n-1$  muchii
6. Între oricare două vârfuri din  $T$  există un unic lanț elementar.

# Construcția de arbori cu secvența gradelor dată

Fie  $s_0 = \{d_1, \dots, d_n\}$ .



- Condiții necesare pentru ca  $s_0$  să fie secvența gradelor unui **arbore**?

- Idei de algoritm – cu ce vârf începem construcția?

# Construcția de arbori cu secvența gradelor dată

Fie  $s_0 = \{d_1, \dots, d_n\}$ .

- Condiții necesare pentru ca  $s_0$  să fie secvența gradelor unui **arbore**?

$$d_1 + \dots + d_n = 2(n-1)$$

- Idei de algoritm – cu ce vârf începem construcția?
  - cu vârf terminal



# Construcția de arbori cu secvența gradelor dată

## Teoremă

O secvență de  $n \geq 2$  numere naturale  $s_0 = \{d_1, \dots, d_n\}$  este secvența gradelor unui arbore  $\Leftrightarrow$

$$d_1 + \dots + d_n = 2(n-1)$$

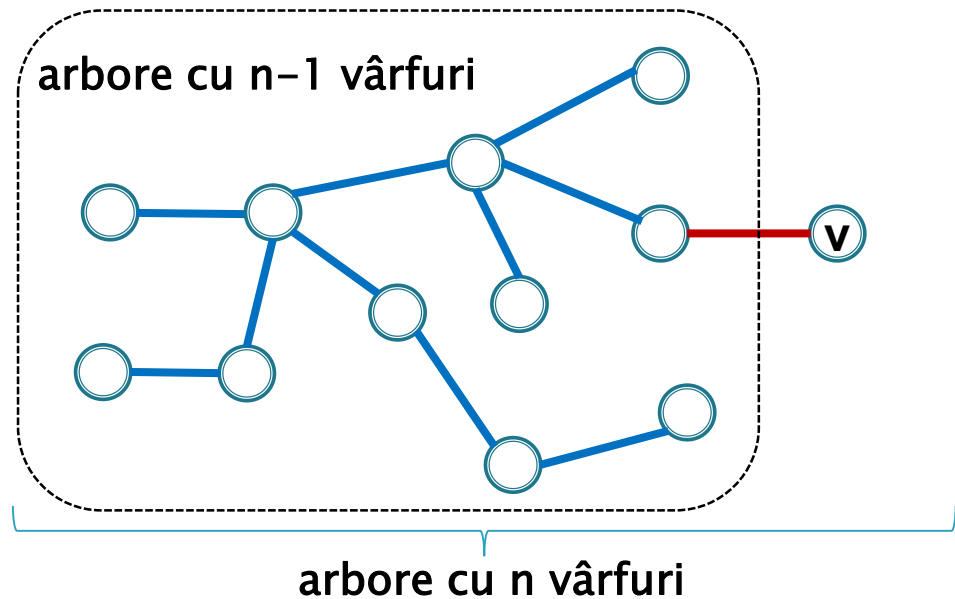
# Construcția de arbori cu secvența gradelor dată

## Teoremă

O secvență de  $n \geq 2$  numere naturale  $s_0 = \{d_1, \dots, d_n\}$  este secvența gradelor unui arbore  $\Leftrightarrow$

$$d_1 + \dots + d_n = 2(n-1)$$

Demonstrație – Inducție după  $n - v$ . curs



# Construcția de arbori cu secvența gradelor dată

## Algoritm

**Idee** – rezultă din demonstrația inductivă:

La un pas unim un vârf de grad 1 cu un vârful de grad maxim ( $>1$ )



Unde intervine în demonstrație faptul că  $d_1$  este maxim?

# Construcția de arbori cu secvența gradelor dată

## Algoritm

**Idee** – rezultă din demonstrația inductivă:

La un pas unim un vârf de grad 1 cu un vârful de grad maxim ( $>1$ )

Unde intervine în demonstrație faptul că  $d_1$  este maxim?



**Nu intervine, este doar pentru a simplifica scrierea demonstrației**

**$\Rightarrow$  putem uni un vârf de grad 1 cu orice vârf de grad  $> 1$**

# Construcția de arbori cu secvența gradelor dată

## Algoritm

**Idee** – rezultă din demonstrația inductivă + observația anterioară

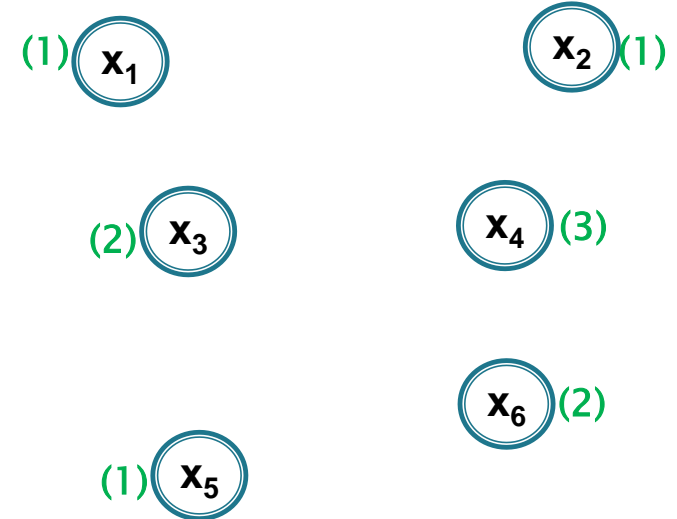
- ▶ La un pas unim un vârf de grad 1 cu **un vârf** de grad  $>1$  și actualizăm secvența  $s_0$
- ▶ Se repetă de  $n-2$  ori
- ▶ În final rămân în secvență două vârfuri de grad 1, care se unesc printr-o muchie

# Construcția de arbori cu secvența gradelor dată

## Algoritm – Exemplu

$$s_0 = \{\underset{x_1}{1}, \underset{x_2}{1}, \underset{x_3}{2}, \underset{x_4}{3}, \underset{x_5}{1}, \underset{x_6}{2}\} \text{ – are suma } 2(n-1)$$

Pasul 1



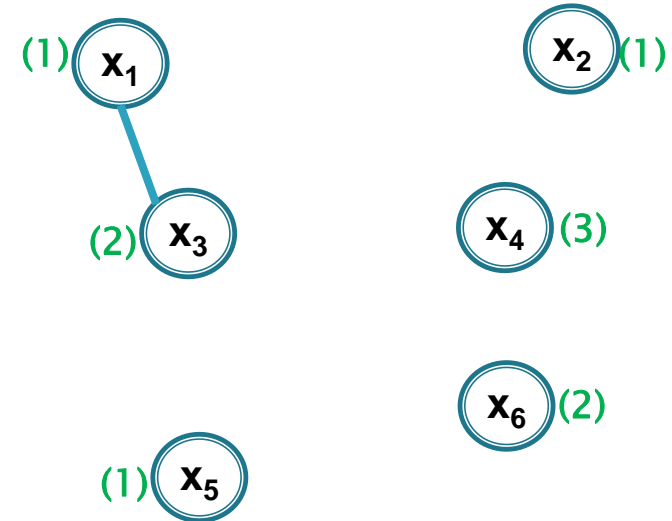
# Construcția de arbori cu secvența gradelor dată

## Algoritm – Exemplu

$$s_0 = \{\underset{x_1}{1}, \underset{x_2}{1}, \underset{x_3}{2}, \underset{x_4}{3}, \underset{x_5}{1}, \underset{x_6}{2}\} \quad \text{– are suma } 2(n-1)$$

Pasul 1 – unim  $x_1$  cu  $x_3$

$$s'_0 = \{\underset{x_2}{1}, \underset{x_3}{\color{red}1}, \underset{x_4}{3}, \underset{x_5}{1}, \underset{x_6}{2}\}$$



# Construcția de arbori cu secvența gradelor dată

## Algoritm – Exemplu

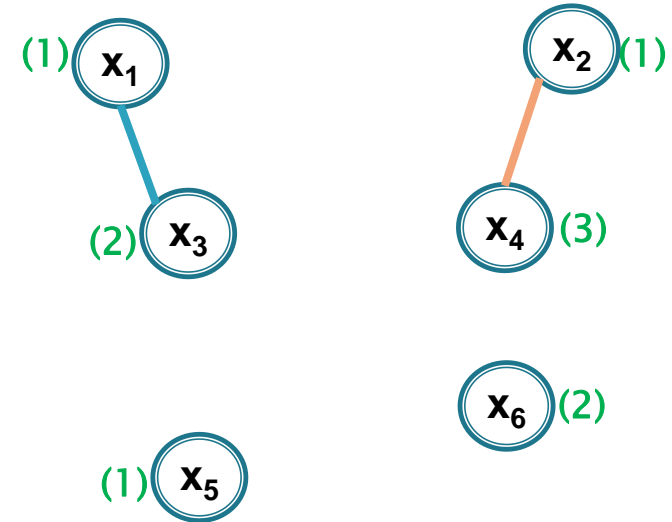
$$s_0 = \{\underset{x_1}{1}, \underset{x_2}{1}, \underset{x_3}{2}, \underset{x_4}{3}, \underset{x_5}{1}, \underset{x_6}{2}\} \quad - \text{ are suma } 2(n-1)$$

Pasul 1 – unim  $x_1$  cu  $x_3$

$$s'_0 = \{\quad \underset{x_2}{1}, \underset{x_3}{1}, \underset{x_4}{3}, \underset{x_5}{1}, \underset{x_6}{2}\}$$

Pasul 2 – unim  $x_2$  cu  $x_4$

$$s''_0 = \{\quad \underset{x_3}{1}, \underset{x_4}{2}, \underset{x_5}{1}, \underset{x_6}{2}\}$$





# Construcția de arbori cu secvența gradelor dată

## Algoritm – Exemplu

$$s_0 = \{\underset{x_1}{1}, \underset{x_2}{1}, \underset{x_3}{2}, \underset{x_4}{3}, \underset{x_5}{1}, \underset{x_6}{2}\} \quad - \text{ are suma } 2(n-1)$$

Pasul 1 – unim  $x_1$  cu  $x_3$

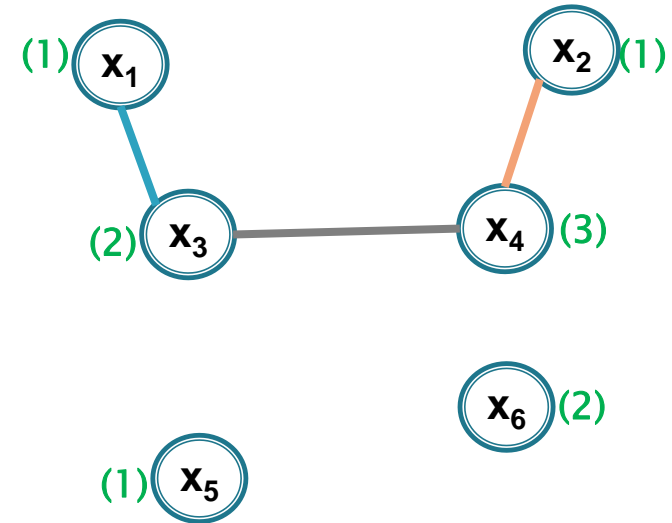
$$s'_0 = \{\quad \underset{x_2}{1}, \underset{x_3}{1}, \underset{x_4}{3}, \underset{x_5}{1}, \underset{x_6}{2}\}$$

Pasul 2 – unim  $x_2$  cu  $x_4$

$$s''_0 = \{\quad \quad \underset{x_3}{1}, \underset{x_4}{2}, \underset{x_5}{1}, \underset{x_6}{2}\}$$

Pasul 3 – unim  $x_3$  cu  $x_4$

$$s'''_0 = \{\quad \quad \quad \underset{x_4}{1}, \underset{x_5}{1}, \underset{x_6}{2}\}$$



# Construcția de arbori cu secvența gradelor dată

## Algoritm – Exemplu

$$s_0 = \{\underset{x_1}{1}, \underset{x_2}{1}, \underset{x_3}{2}, \underset{x_4}{3}, \underset{x_5}{1}, \underset{x_6}{2}\} \quad - \text{ are suma } 2(n-1)$$

Pasul 1 – unim  $x_1$  cu  $x_3$

$$s'_0 = \{\quad \underset{x_2}{1}, \underset{x_3}{1}, \underset{x_4}{3}, \underset{x_5}{1}, \underset{x_6}{2}\}$$

Pasul 2 – unim  $x_2$  cu  $x_4$

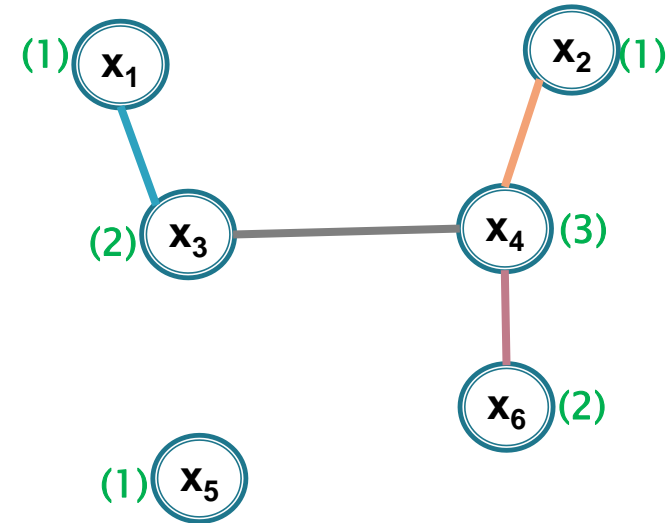
$$s''_0 = \{\quad \quad \underset{x_3}{1}, \underset{x_4}{2}, \underset{x_5}{1}, \underset{x_6}{2}\}$$

Pasul 3 – unim  $x_3$  cu  $x_4$

$$s'''_0 = \{\quad \quad \quad \underset{x_4}{1}, \underset{x_5}{1}, \underset{x_6}{2}\}$$

Pasul 4 – unim  $x_4$  cu  $x_6$

$$s^{iv}_0 = \{\quad \quad \quad \underset{x_5}{1}, \underset{x_6}{1}\}$$



# Construcția de arbori cu secvența gradelor dată

## Algoritm – Exemplu

$$s_0 = \{\underset{x_1}{1}, \underset{x_2}{1}, \underset{x_3}{2}, \underset{x_4}{3}, \underset{x_5}{1}, \underset{x_6}{2}\} \quad - \text{ are suma } 2(n-1)$$

Pasul 1 – unim  $x_1$  cu  $x_3$

$$s'_0 = \{\quad \underset{x_2}{1}, \underset{x_3}{1}, \underset{x_4}{3}, \underset{x_5}{1}, \underset{x_6}{2}\}$$

Pasul 2 – unim  $x_2$  cu  $x_4$

$$s''_0 = \{\quad \quad \underset{x_3}{1}, \underset{x_4}{2}, \underset{x_5}{1}, \underset{x_6}{2}\}$$

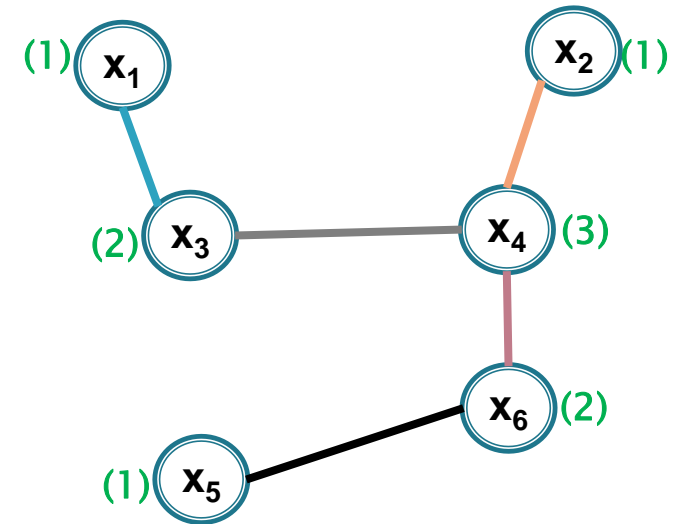
Pasul 3 – unim  $x_3$  cu  $x_4$

$$s'''_0 = \{\quad \quad \quad \underset{x_4}{1}, \underset{x_5}{1}, \underset{x_6}{2}\}$$

Pasul 4 – unim  $x_4$  cu  $x_6$

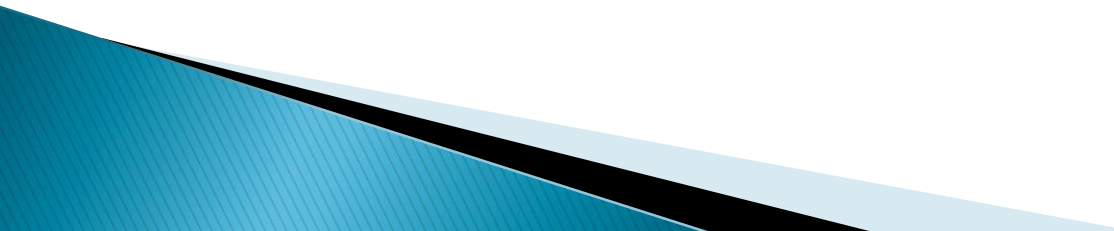
$$s^{iv}_0 = \{\quad \quad \quad \underset{x_5}{1}, \underset{x_6}{1}\}$$

– au mai rămas în secvență două vârfuri de grad 1, pe care le unim



# Construcția de arbori cu secvența gradelor dată

## Algoritm – Pseudocod

1. Dacă  $d_1 + \dots + d_n \neq 2(n-1)$ , atunci scrie NU, STOP.
  2. Cât timp  $s_0$  conține valori mai mari decât 1 execută //pentru  $i=1, n-2$ 
    - alege un număr  $d_k > 1$  și un număr  $d_t = 1$  din secvență  $s_0$
    - adaugă la T muchia  $x_k x_t$ .
- 

# Construcția de arbori cu secvența gradelor dată

## Algoritm – Pseudocod

1. Dacă  $d_1 + \dots + d_n \neq 2(n-1)$ , atunci scrie NU, STOP.
2. Cât timp  $s_0$  conține valori mai mari decât 1 execută //pentru  $i=1, n-2$ 
  - alege un număr  $d_k > 1$  și un număr  $d_t = 1$  din secvență  $s_0$
  - adaugă la  $T$  muchia  $x_k x_t$ .
  - elimină  $d_t$  din  $s_0$
  - înlocuiește  $d_k$  în secvența  $s_0$  cu  $d_k - 1$

# Construcția de arbori cu secvența gradelor dată

## Algoritm – Pseudocod

1. Dacă  $d_1 + \dots + d_n \neq 2(n-1)$ , atunci scrie NU, STOP.
2. Cât timp  $s_0$  conține valori mai mari decât 1 execută //pentru  $i=1, n-2$ 
  - alege un număr  $d_k > 1$  și un număr  $d_t = 1$  din secvența  $s_0$
  - adaugă la T muchia  $x_k x_t$ .
  - elimină  $d_t$  din  $s_0$
  - înlocuiește  $d_k$  în secvența  $s_0$  cu  $d_k - 1$
3. fie  $d_k, d_t$  unicele elemente nenule (egale cu 1) din  $s_0$ ;  
adaugă la T muchia  $x_k x_t$

# Construcția de arbori cu secvența gradelor dată

## Algoritm – Pseudocod

1. Dacă  $d_1 + \dots + d_n \neq 2(n-1)$ , atunci scrie NU, STOP.
2. Cât timp  $s_0$  conține valori mai mari decât 1 execută //pentru  $i=1, n-2$ 
  - alege un număr  $d_k > 1$  și un număr  $d_t = 1$  din secvența  $s_0$
  - adaugă la  $T$  muchia  $x_k x_t$ .
  - elimină  $d_t$  din  $s_0$
  - înlocuiește  $d_k$  în secvența  $s_0$  cu  $d_k - 1$
3. fie  $d_k, d_t$  unicele elemente nenule (egale cu 1) din  $s_0$ ;  
adaugă la  $T$  muchia  $x_k x_t$

**Complexitate:  $O(n)$**

