# Construcția de grafuri cu secvența gradelor dată

Buzatu Giulian
Ilie Dumitru
Coordonator: Prof. Dr. Dumitran Adrian Marius

# Problemă

- Se dă numărul de legături ale fiecărui nod. Este realizabilă o rețea de legături între noduri care să respecte numărul de legături ale fiecărui nod?
- Dacă da, să se construiască un model de rețea.

# Problemă

- Se dă numărul de legături ale fiecărui nod. Este realizabilă o rețea de legături între noduri care să respecte numărul de legături ale fiecărui nod?
- Dacă da, să se construiască un model de rețea.
- Exemplu: Într-o grupă de studenți, fiecare student este întrebat cu câți studenți a colaborat în timpul anilor de studii. Este realizabilă o rețea de colaborări care să corespundă răspunsurilor lor? Este posibil ca informațiile adunate să fie incorecte?

➢ Studentul 1 – 3 colaborări
➢ Studentul 2 – 3 colaborări
➢ Studentul 3 – 2 colaborări
➢ Studentul 4 – 3 colaborări
➢ Studentul 5 – 2 colaborări

# Problemă

- Dată o secvenţă de numere s, se poate construi un graf neorientat având secvenţa gradelor s?
- Dar un multigraf neorientat?
- Dar un arbore?

- **Condiții necesare?**
- **Condiții suficiente?**

# Aplicații

Aplicații ale construcției de grafuri pe baza secvenței gradelor:

- **Chimie** – studiul structurilor posibile a unor compuși cu formula chimică dată.
- **Biologie** - rețele metabolice, de interacțiuni între gene/proteine.
- **Studii epidemiologice** (vezi COVID19) - prin chestionare anonimă, persoanele declară numărul de interacțiuni umane avute.
- **Proiectarea de rețele**
- **Studii bazate pe simulări de rețele**

# Construcția de grafuri cu secvența gradelor dată

## Problemă

Fie $s_0$= $\{d_1,d_2,\ldots,d_n\}$ o secvență de numere naturale.
Să se construiască, dacă se poate, un graf neorientat G cu $s(G) = s_0$ (secvența gradelor lui G este $s_0$).

# Construcția de grafuri cu secvența gradelor dată

**Problemă**

Fie $s_0 = \{d_1, d_2, \ldots, d_n\}$ o secvență de numere naturale.
Să se construiască, dacă se poate, un graf neorientat G cu $s(G) = s_0$ (secvența gradelor lui G este $s_0$).

Condiții necesare pentru existența lui G:

- $d_1 + d_2 + \ldots + d_n$ – număr par
- $d_i \le n - 1, \forall\ 1 \le i \le n$

# Construcția de grafuri cu secvența gradelor dată

**Problemă**

Fie $s_0 = \{d_1, d_2, \ldots, d_n\}$ o secvență de numere naturale.
Să se construiescă, dacă se poate, un graf neorientat G cu $s(G) = s_0$ (secvența gradelor lui G este $s_0$).

Condiții necesare pentru existența lui G:

- $d_1 + d_2 + \ldots + d_n$ – număr par
- $d_i \leq n - 1, \forall\ 1 \leq i \leq n$

Încercați să găsiți un graf G pentru $s_0 = \{3, 3, 1, 1\}$.

# Construcția de grafuri cu secvența gradelor dată

**Problemă**

Fie $s_0 = \{d_1, d_2, \ldots, d_n\}$ o secvență de numere naturale.
Să se construiescă, dacă se poate, un graf neorientat G cu $s(G) = s_0$ (secvența gradelor lui G este $s_0$).

Condiții necesare pentru existența lui G:

- $d_1 + d_2 + \ldots + d_n$ – număr par
- $d_i \leq n - 1, \forall\ 1 \leq i \leq n$

Încercați să găsiți un graf G pentru $s_0 = \{3, 3, 1, 1\}$.

Hmm, nu se poate!
Deși sunt îndeplinite condițiile necesare, acestea nu sunt suficiente.
Totuși, se poate construi un multigraf.

# Algoritmul Havel-Hakimi

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

1. Începem construcția de la vârful cu gradul cel mai mare.

# Algoritmul Havel-Hakimi

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

1. Începem construcția de la vârful cu gradul cel mai mare.
2. Îi alegem ca vecini vârfurile cu gradele cele mai mari.

# Algoritmul Havel-Hakimi

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

1. Începem construcția de la vârful cu gradul cel mai mare.
2. Îi alegem ca vecini vârfurile cu gradele cele mai mari.
3. Actualizăm secvența $s_0$ și reluăm până când:
   3.1. Secvența conține doar 0, caz în care am obținut graful G.
   3.2. Graful nu are suficiente noduri incomplete pentru a satisface nodul cu grad maxim, deci nu se poate construi G prin acest procedeu.

# Algoritmul Havel-Hakimi

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

1. Începem construcția de la vârful cu gradul cel mai mare.
2. Îi alegem ca vecini vârfurile cu gradele cele mai mari.
3. Actualizăm secvența $s_0$ și reluăm până când:
   3.1. Secvența conține doar 0, caz în care am obținut graful G.
   3.2. Graful nu are suficiente noduri incomplete pentru a satisface nodul cu grad maxim, deci nu se poate construi G prin acest procedeu.

**Dacă algoritmul de mai sus spune că nu există un graf, se poate construi G prin altă metodă?**

# Algoritmul Havel-Hakimi

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

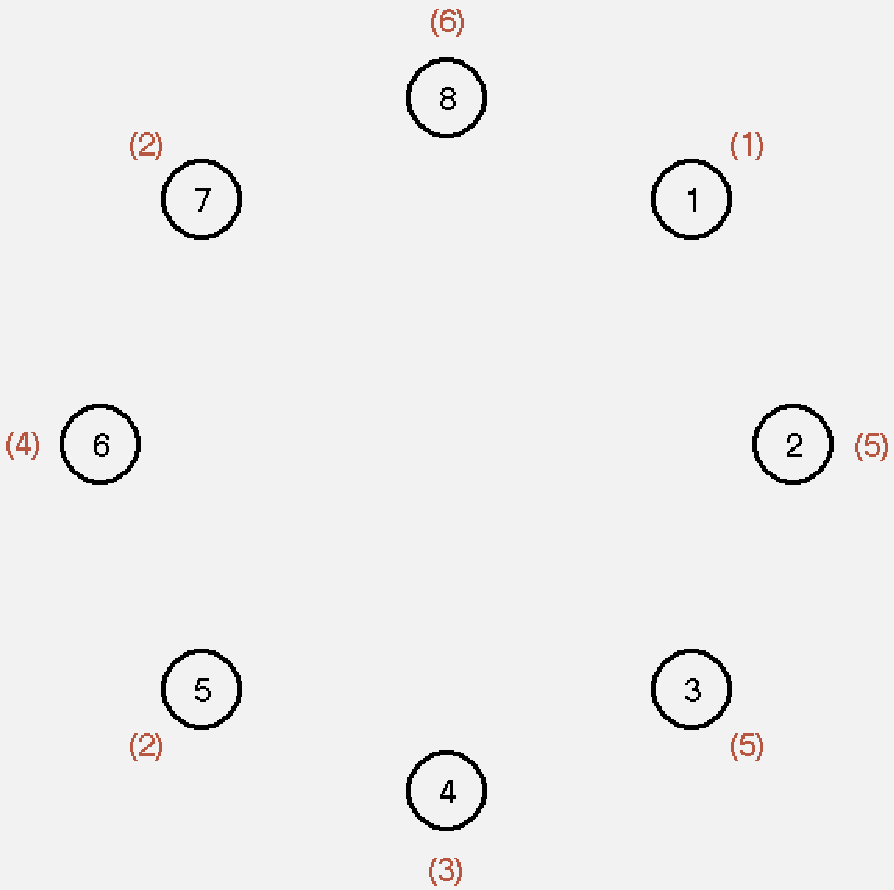
1. Începem construcția de la vârful cu gradul cel mai mare.
2. Îi alegem ca vecini vârfurile cu gradele cele mai mari.
3. Actualizăm secvența $s_0$ și reluăm până când:
   3.1. Secvența conține doar 0, caz în care am obținut graful G.
   3.2. Graful nu are suficiente noduri incomplete pentru a satisface nodul cu grad maxim, deci nu se poate construi G prin acest procedeu.

**Conform Teoremei Havel-Hakimi, nu este posibilă construcția grafului de la 3.2. prin altă metodă.**
**În concluzie, algoritmul descris anterior este complet.**

# Exemple aplicare algoritm

În ilustrațiile din cadrul exemplelor, următoarele observații sunt valabile:

- muchiile cu verde sunt cele adăugate înainte;
- muchiile cu portocaliu sunt cele nou adăugate;
- numerele de lângă noduri sunt gradele rămase după adăugarea muchiilor actuale (cu portocaliu cât timp încă trebuie să adăugăm muchii pentru ele, odată ce numărul devine 0, va fi reprezentat cu verde);



# Exemple aplicare algoritm

**Exemplul 1**

Fie $s_0$ = {1, 5, 5, 3, 2, 4, 2, 6}. Etichete noduri= {1, 2, 3, 4, 5, 6, 7, 8}.

- Alegem nodul cu gradul maxim.
- După aceea, selectăm ca vecini nodurile care au cele mai mari grade.

# Exemple aplicare algoritm

**Exemplul 1**

Fie $s_0$ = {1, 5, 5, 3, 2, 4, 2, 6}. Etichete noduri= {1, 2, 3, 4, 5, 6, 7, 8}.

- Alegem nodul cu gradul maxim.
- După aceea, selectăm ca vecini nodurile care au cele mai mari grade.

**Ar fi utilă sortarea descrescătoare a elementelor lui $s_0$.**

$s_0$= {6, 5, 5, 4, 3, 2, 2, 1}.

Etichete noduri = {8, 2, 3, 6, 4, 5, 7, 1}.

# Exemple aplicare algoritm

**Pasul 1**

$s_0$= {6, 5, 5, 4, 3, 2, 2, 1}.

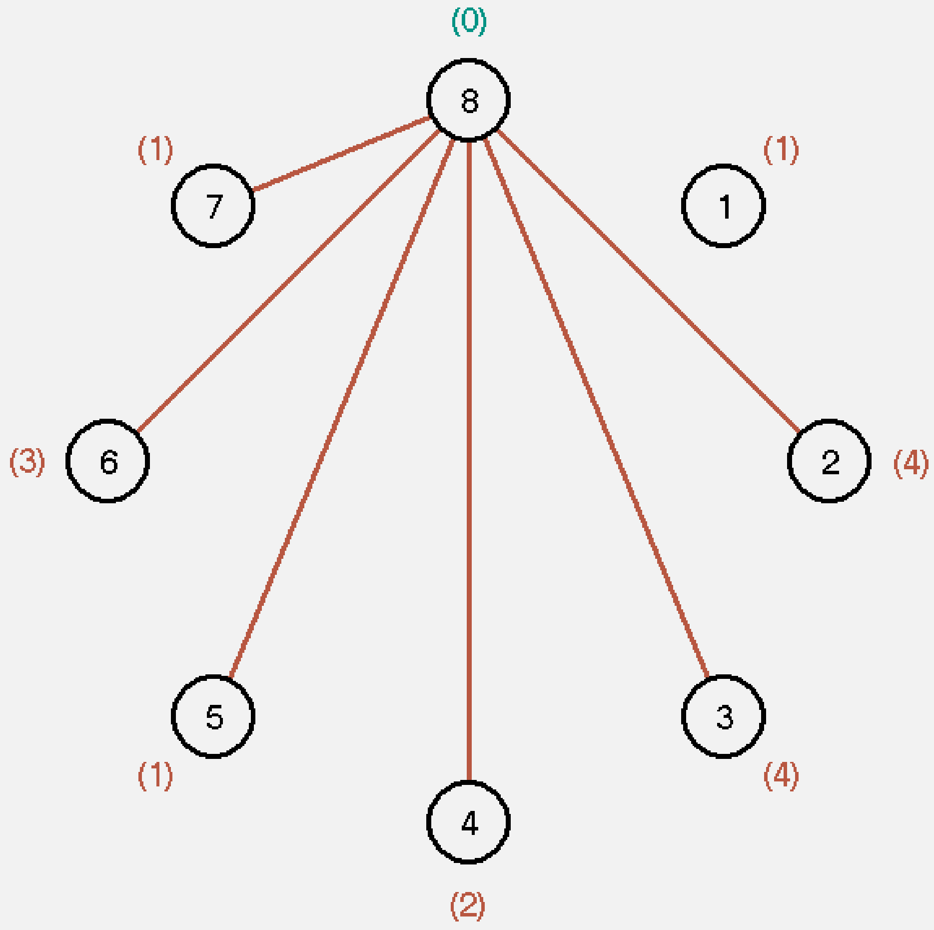
Etichete noduri = {8, 2, 3, 6, 4, 5, 7, 1}.

# Exemple aplicare algoritm

**Pasul 1**

$s_0$= {6, 5, 5, 4, 3, 2, 2, 1}.

Etichete noduri = {8, 2, 3, 6, 4, 5, 7, 1}.



# Exemple aplicare algoritm

**Pasul 1**

$s_0 = \{6, 5, 5, 4, 3, 2, 2, 1\}$.

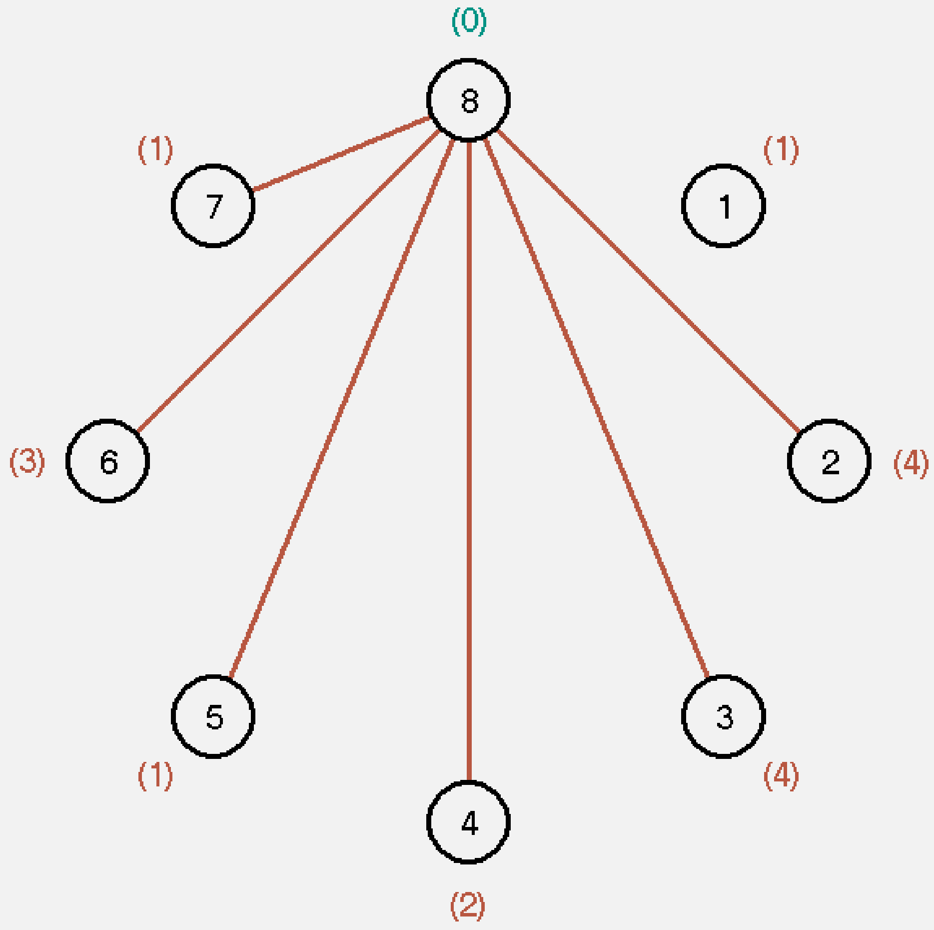
Etichete noduri = {8, 2, 3, 6, 4, 5, 7, 1}.

Secvența rămasă:

$s_1 = \{\ 4, 4, 3, 2, 1, 1, 1\}$.

Etichete noduri = { 2, 3, 6, 4, 5, 7, 1}.

(este ordonată descrescător)



# Exemple aplicare algoritm

**Pasul 2**

$s_1 = \{ \ 4, 4, 3, 2, 1, 1, 1\}.$

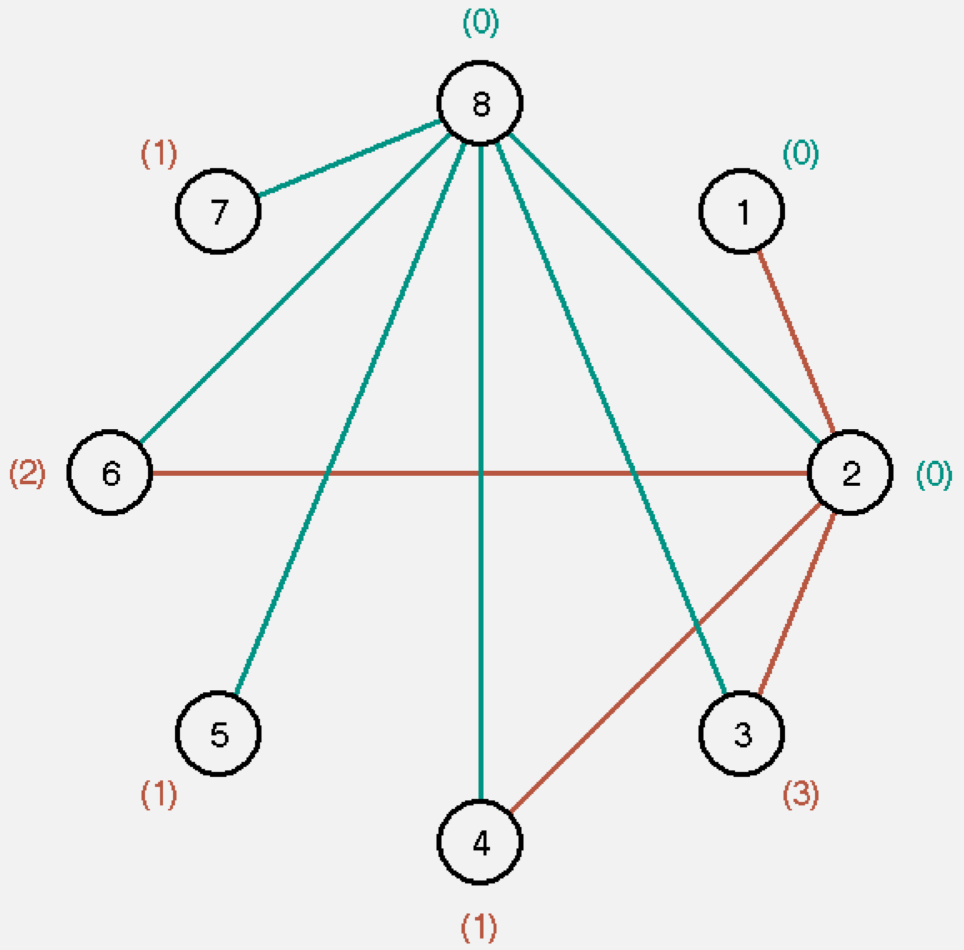
Etichete noduri = { 2, 3, 6, 4, 1, 5, 7}.

# Exemple aplicare algoritm

**Pasul 2**

$s_1$= {   4, 4, 3, 2, 1, 1, 1}.

Etichete noduri = {   2, 3, 6, 4, 1, 5, 7}.



# Exemple aplicare algoritm

**Pasul 2**

$s_1 = \{ \; 4, 4, 3, 2, 1, 1, 1\}.$

Etichete noduri = { 2, 3, 6, 4, 1, 5, 7}.

Secvența rămasă:

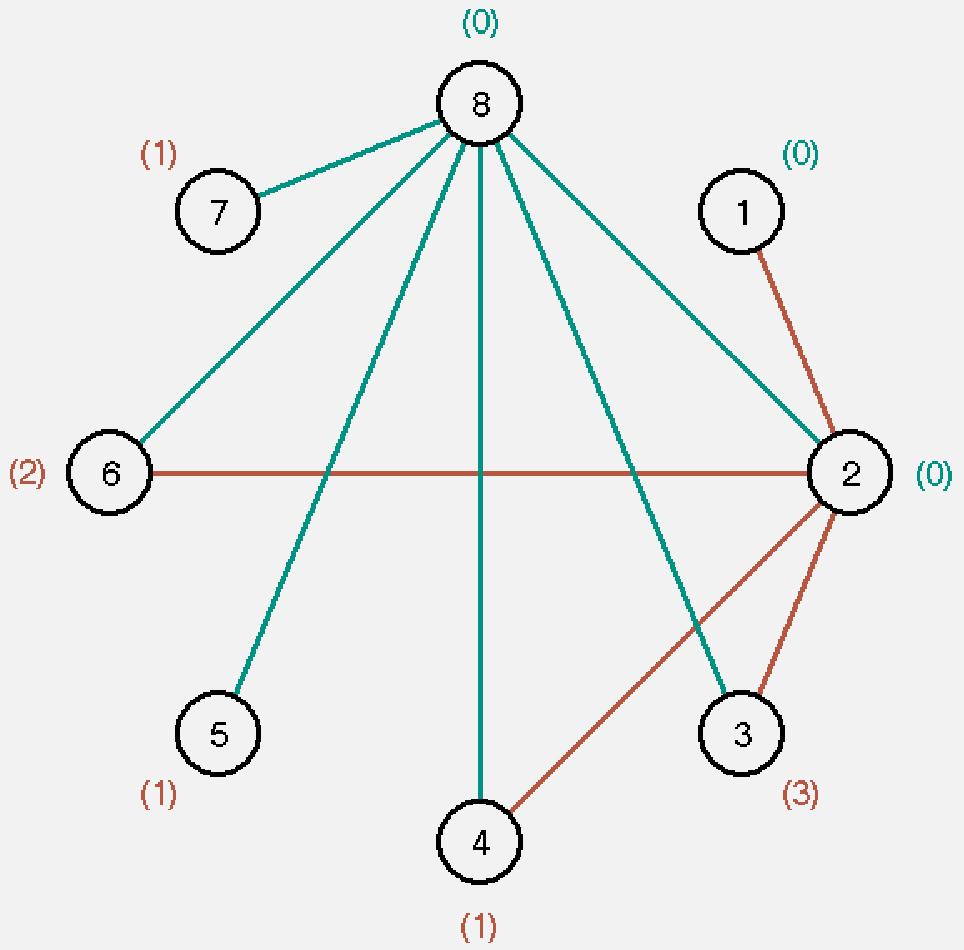
$s_2 = \{ \; 3, 2, 1, 0, 1, 1\}.$

Etichete noduri = { 3, 6, 4, 1, 5, 7}.

Secvența rămasă ordonată descrescător:

$s_2 = \{ \; 3, 2, 1, 1, 1, 0\}.$

Etichete noduri = { 3, 6, 4, 5, 7, 1}.



# Exemple aplicare algoritm

**Pasul 3**

$s_2 = \{ \quad 3, 2, 1, 1, 1, 0\}.$

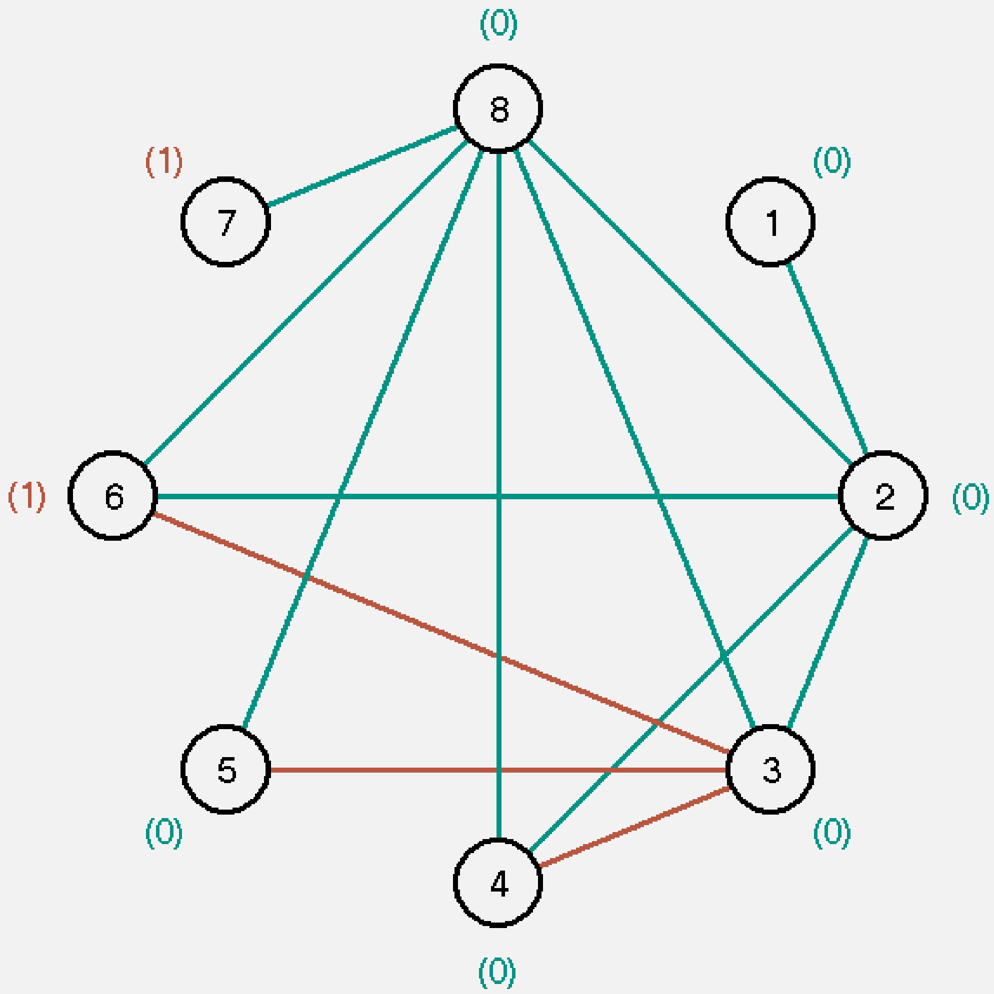
Etichete noduri = { 3, 6, 4, 5, 7, 1}.

# Exemple aplicare algoritm

**Pasul 3**

$s_2$= {      3, 2, 1, 1, 1, 0}.

Etichete noduri = {      3, 6, 4, 5, 7, 1}.



# Exemple aplicare algoritm

**Pasul 3**

$s_2 = \{ \quad 3, 2, 1, 1, 1, 0\}.$

Etichete noduri = { 3, 6, 4, 5, 7, 1}.

Secvența rămasă:

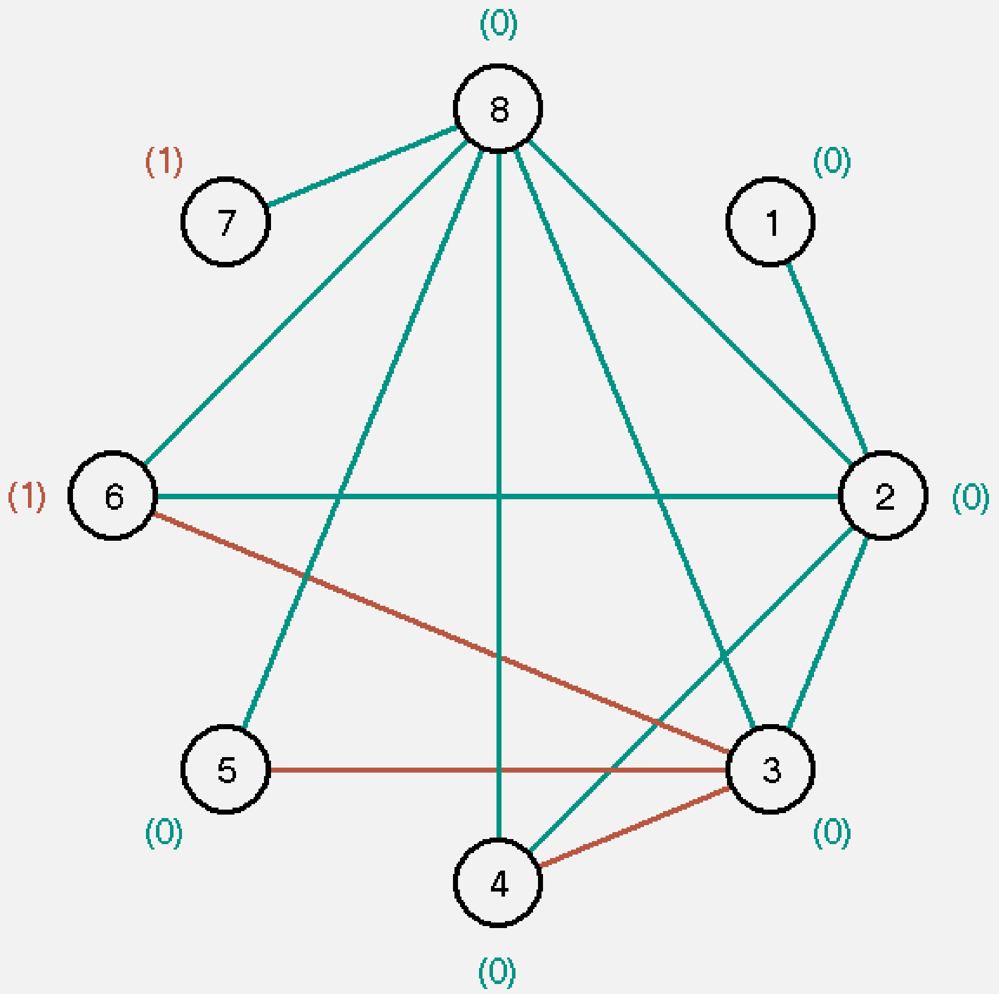
$s_3 = \{ \quad 1, 0, 0, 1, 0\}.$

Etichete noduri = { 6, 4, 5, 7, 1}.

Secvența rămasă ordonată descrescător:

$s_3 = \{ \quad 1, 1, 0, 0, 0\}.$

Etichete noduri = { 6, 7, 4, 5, 1}.



# Exemple aplicare algoritm

**Pasul 4**

$s_3$= { 1, 1, 0, 0, 0}.

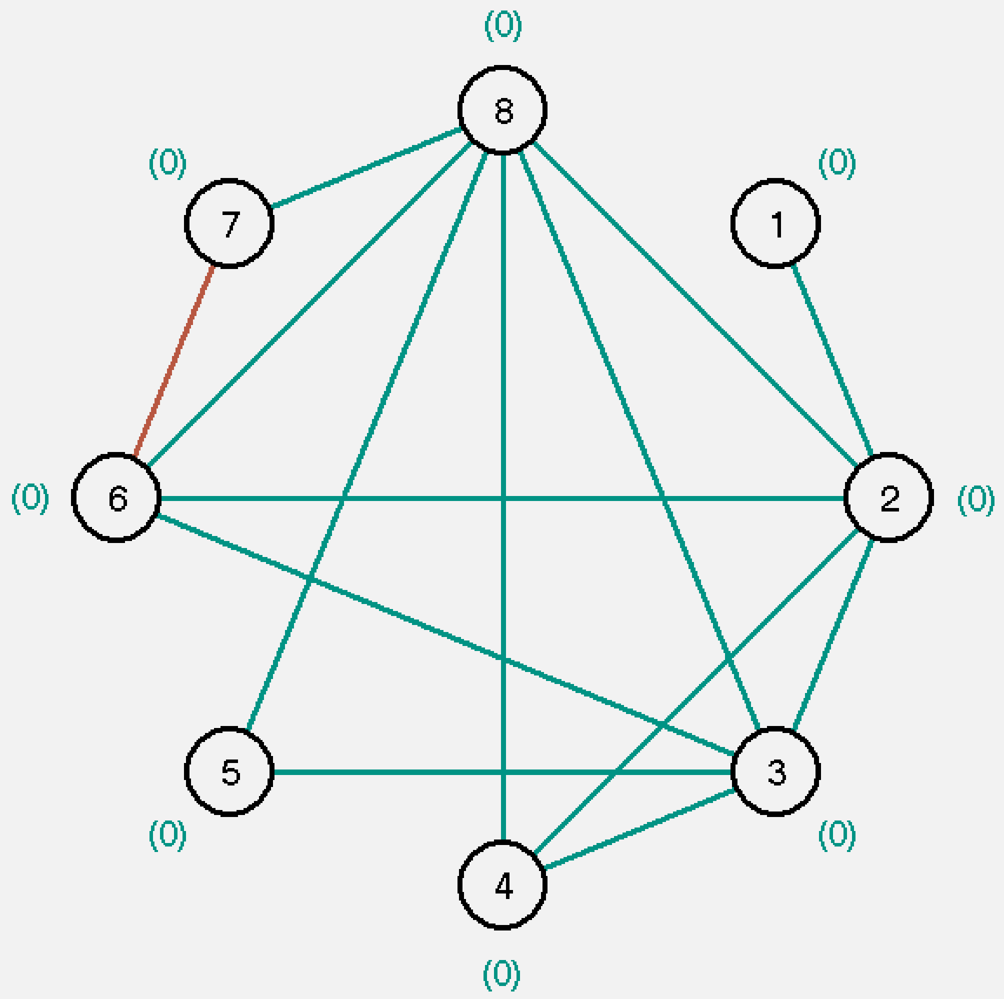
Etichete noduri = { 6, 7, 4, 5, 1}.

# Exemple aplicare algoritm

**Pasul 4**

$s_3$= { 1, 1, 0, 0, 0}.

Etichete noduri = { 6, 7, 4, 5, 1}.



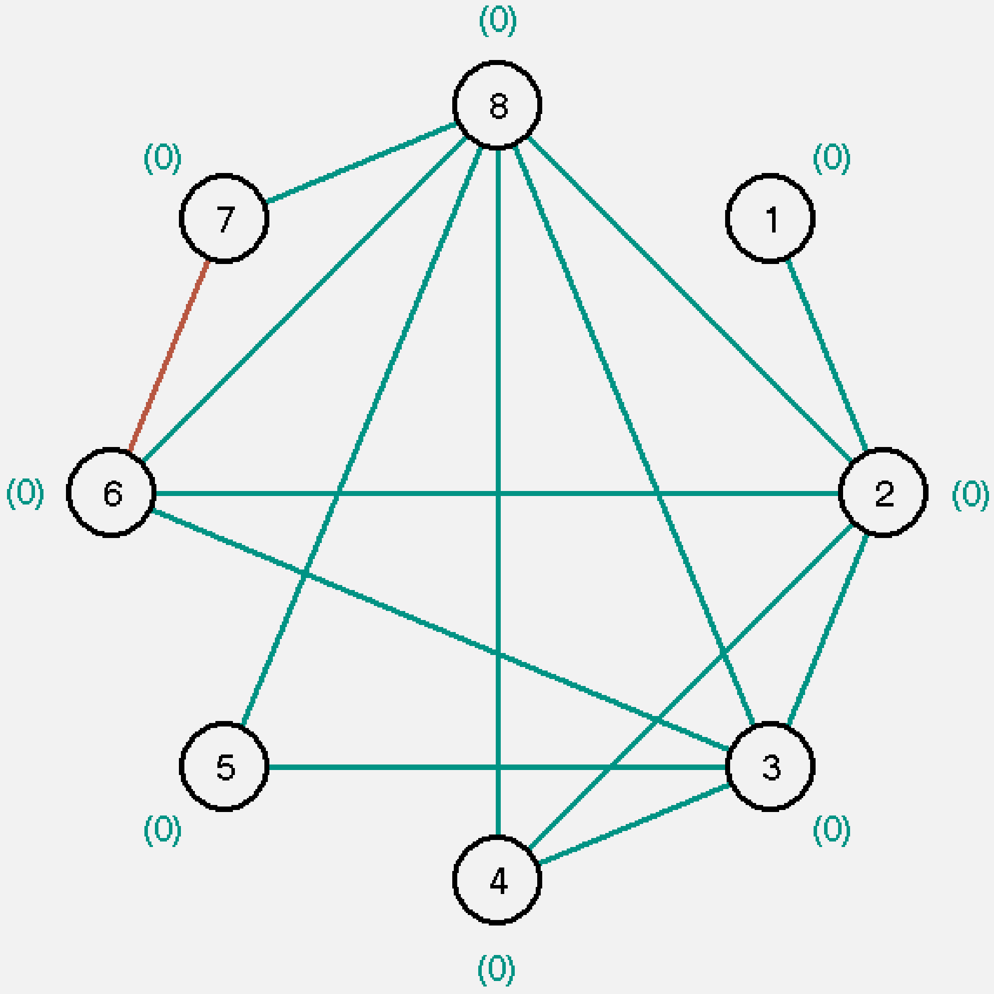
# Exemple aplicare algoritm

**Pasul 4**

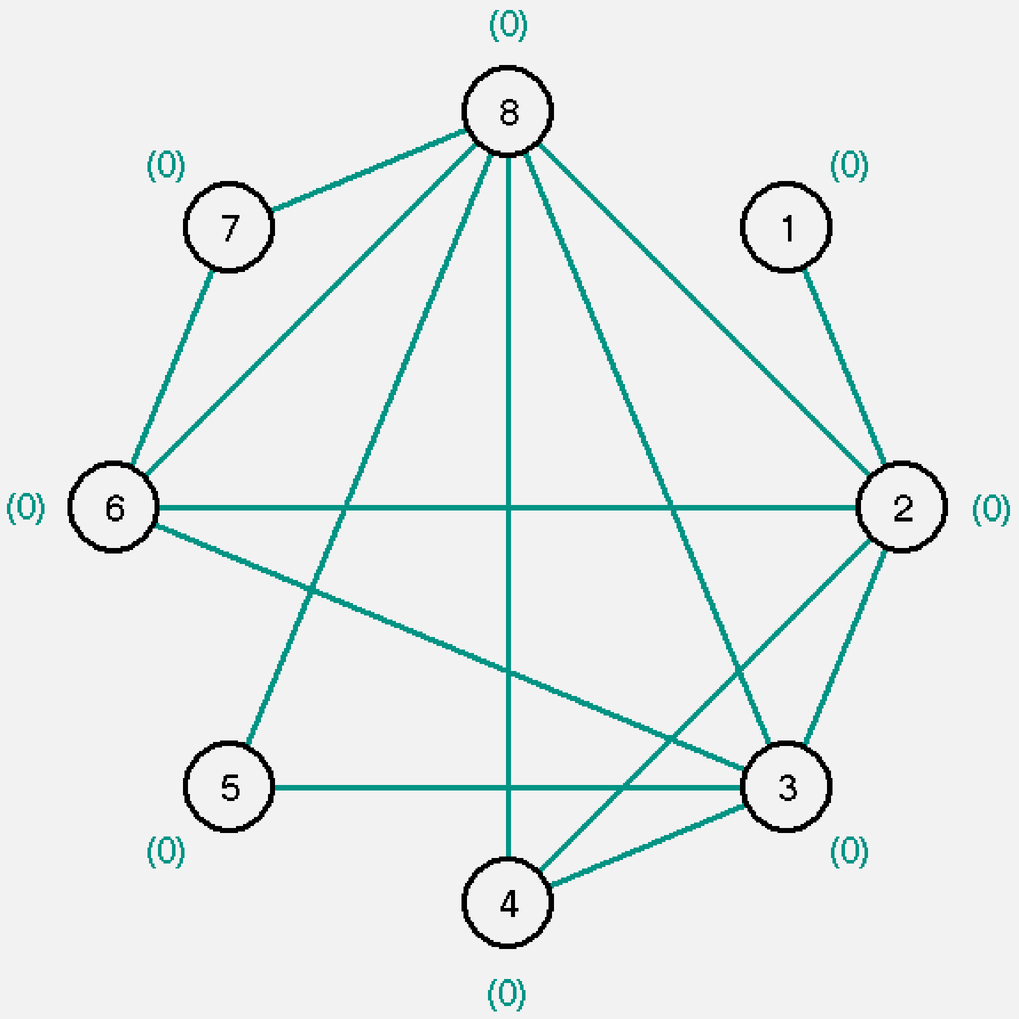
$s_3$= { 1, 1, 0, 0, 0}.
Etichete noduri = { 6, 7, 4, 5, 1}.

Secvența rămasă:
$s_4$= { 0, 0, 0, 0}.
Etichete noduri = { 7, 4, 5, 1}.

**STOP! Am găsit o soluție!**



# Exemple aplicare algoritm



# Exemple aplicare algoritm

**Exemplul 2**

Fie $s_0$ = {5, 4, 4, 2, 4, 1}. Etichete noduri= {1, 2, 3, 4, 5, 6}.

- După sortarea descrescătoare, avem:

$$s_0 = \{5, 4, 4, 4, 2, 1\}.$$

Etichete noduri = {1, 2, 3, 5, 4, 6}.

# Exemple aplicare algoritm

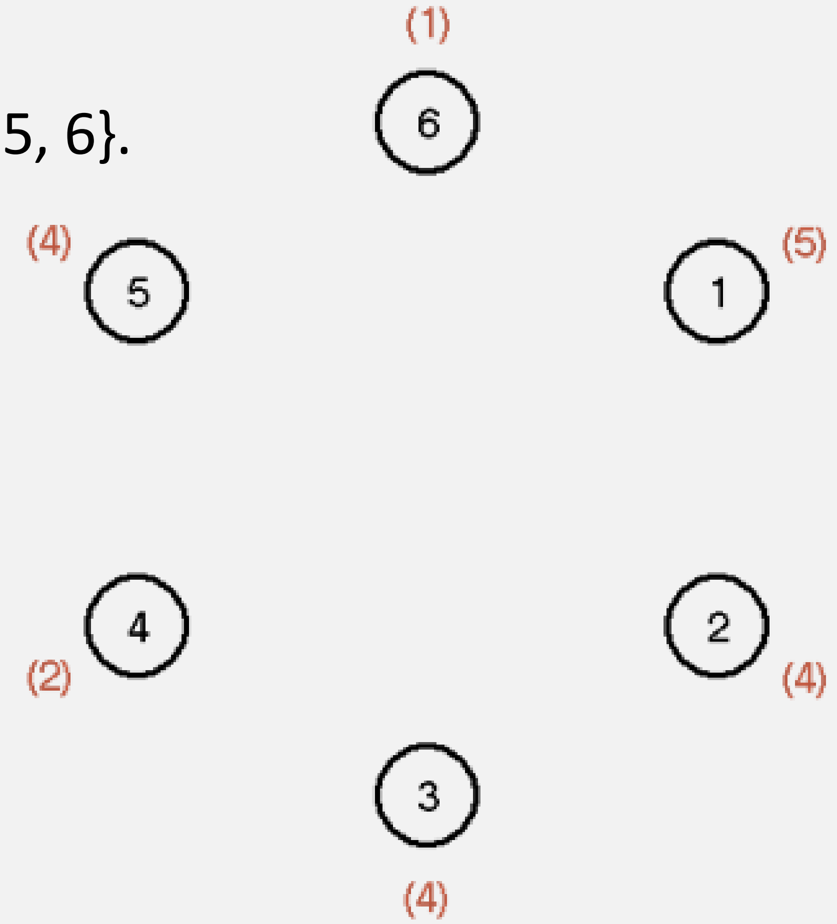
**Exemplul 2**

Fie $s_0$ = {5, 4, 4, 2, 4, 1}. Etichete noduri= {1, 2, 3, 4, 5, 6}.

- După sortarea descrescătoare, avem:

$$s_0 = \{5, 4, 4, 4, 2, 1\}.$$

Etichete noduri = {1, 2, 3, 5, 4, 6}.



# Exemple aplicare algoritm

**Pasul 1**

$s_0$= {5, 4, 4, 4, 2, 1}.

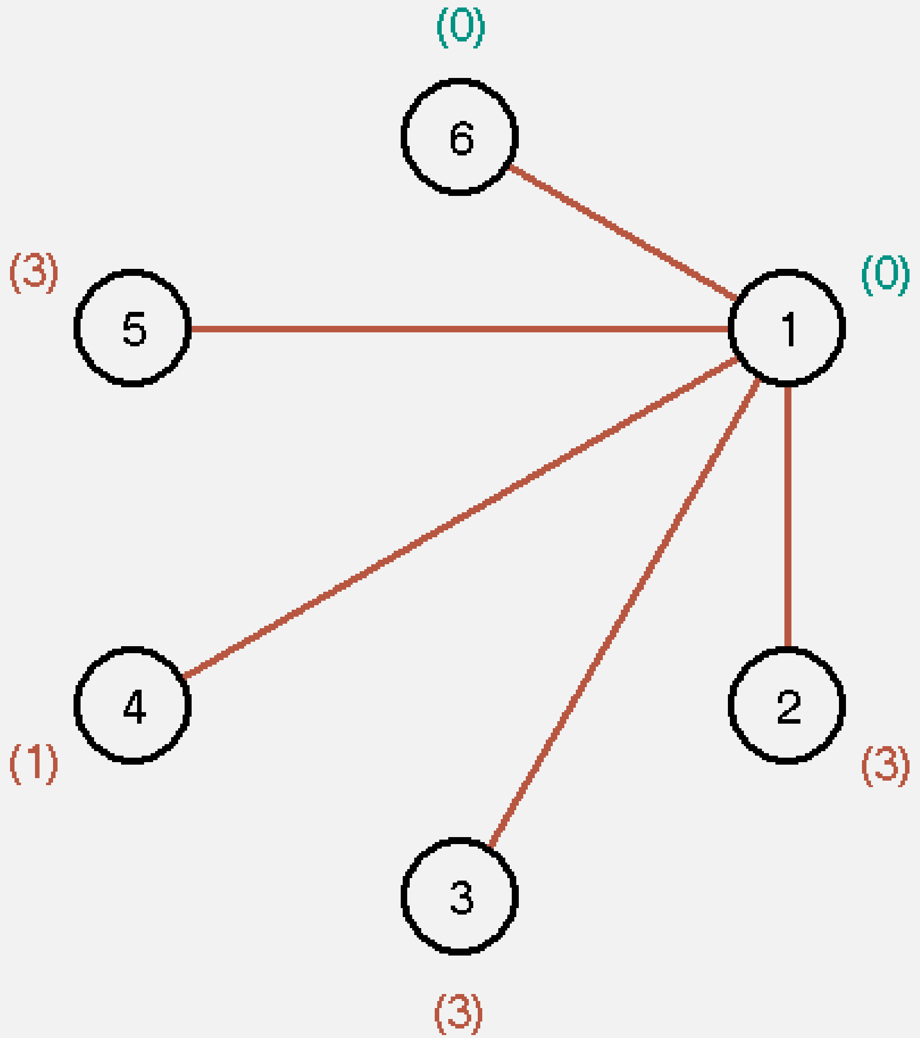
Etichete noduri = {1, 2, 3, 5, 4, 6}.

# Exemple aplicare algoritm

**Pasul 1**

$s_0$= {5, 4, 4, 4, 2, 1}.

Etichete noduri = {1, 2, 3, 5, 4, 6}.

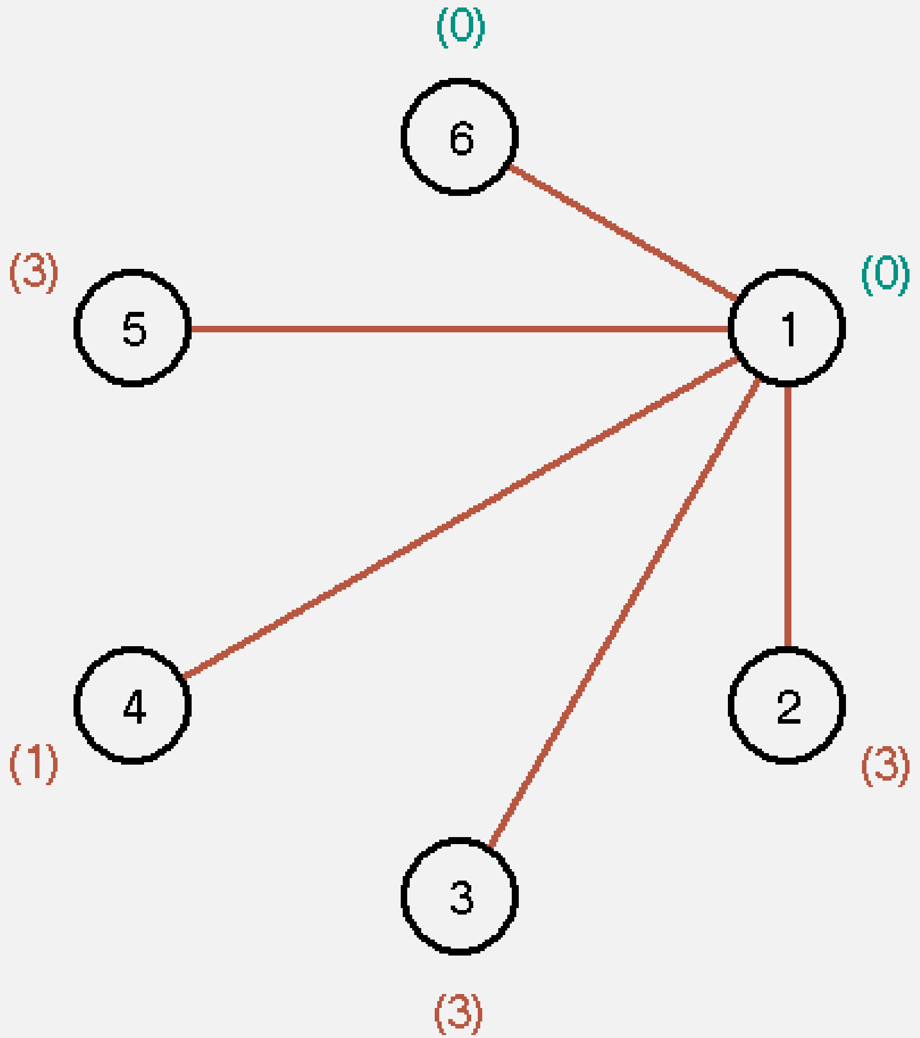


# Exemple aplicare algoritm

**Pasul 1**

$s_0$= {5, 4, 4, 4, 2, 1}.

Etichete noduri = {1, 2, 3, 5, 4, 6}.




Secvența rămasă:

$s_1$= {   3, 3, 3, 1, 0}.

Etichete noduri = {   2, 3, 5, 4, 6}.

(este ordonată descrescător)

# Exemple aplicare algoritm

**Pasul 2**

$s_1 = \{\ 3, 3, 3, 1, 0\}.$

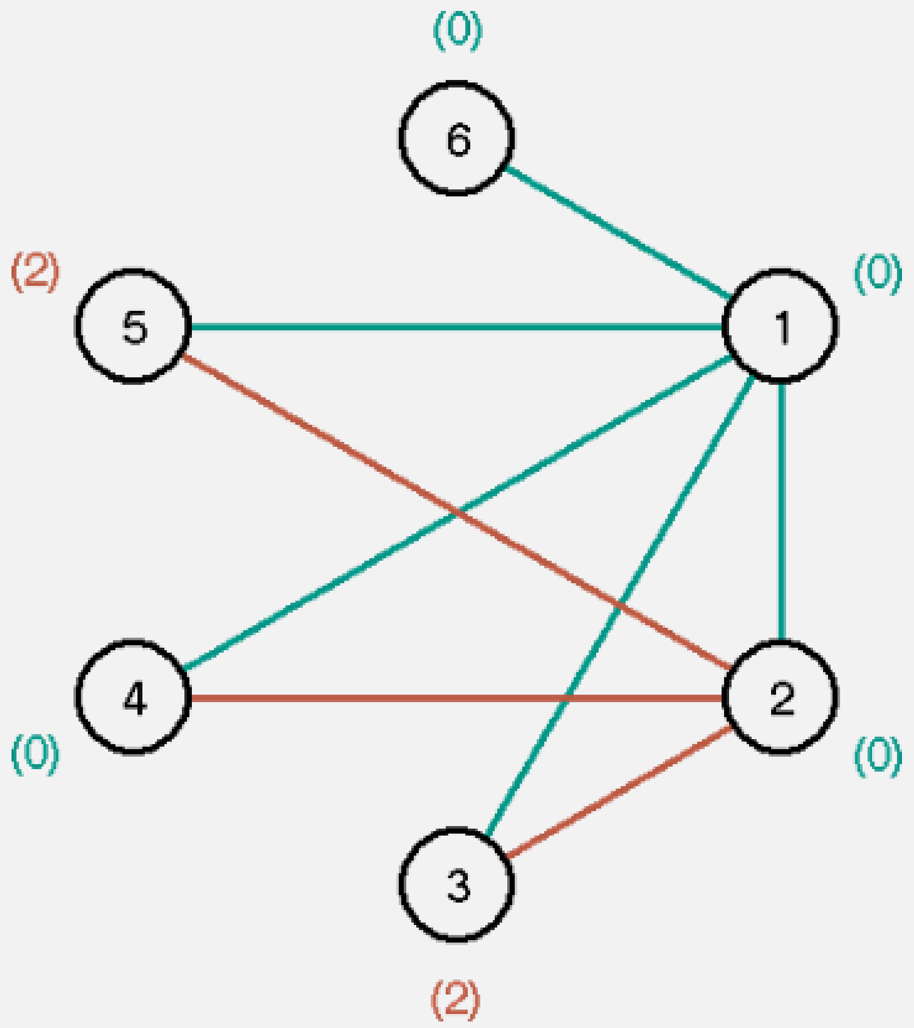
Etichete noduri = { 2, 3, 5, 4, 6}.

# Exemple aplicare algoritm

**Pasul 2**

$s_1 = \{\ 3, 3, 3, 1, 0\}.$

Etichete noduri = { 2, 3, 5, 4, 6}.



# Exemple aplicare algoritm

**Pasul 2**

$s_1 = \{\ \ 3, 3, 3, 1, 0\}.$

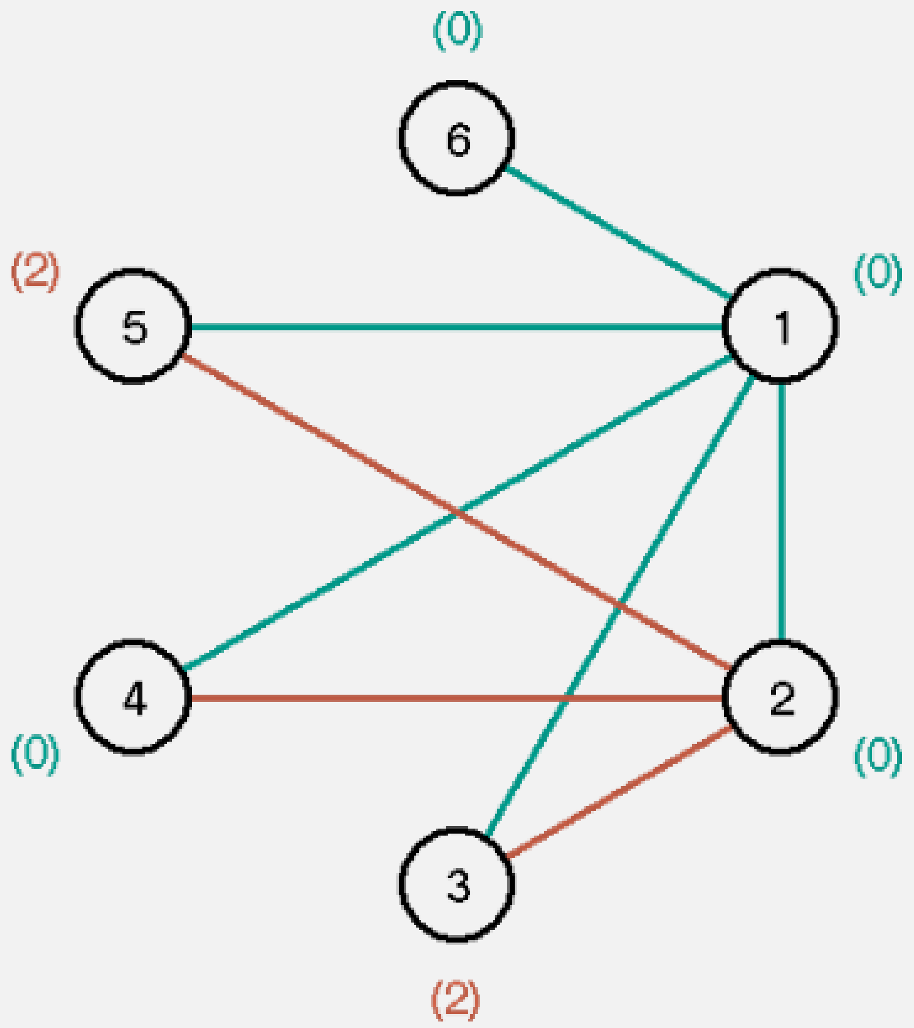
Etichete noduri = {   2, 3, 5, 4, 6}.

Secvența rămasă:

$s_2 = \{\ \ \ 2, 2, 0, 0\}.$

Etichete noduri = {      3, 5, 4, 6}.

(este ordonată descrescător)



# Exemple aplicare algoritm

**Pasul 3**

$s_2 = \{ \quad 2, 2, 0, 0\}.$

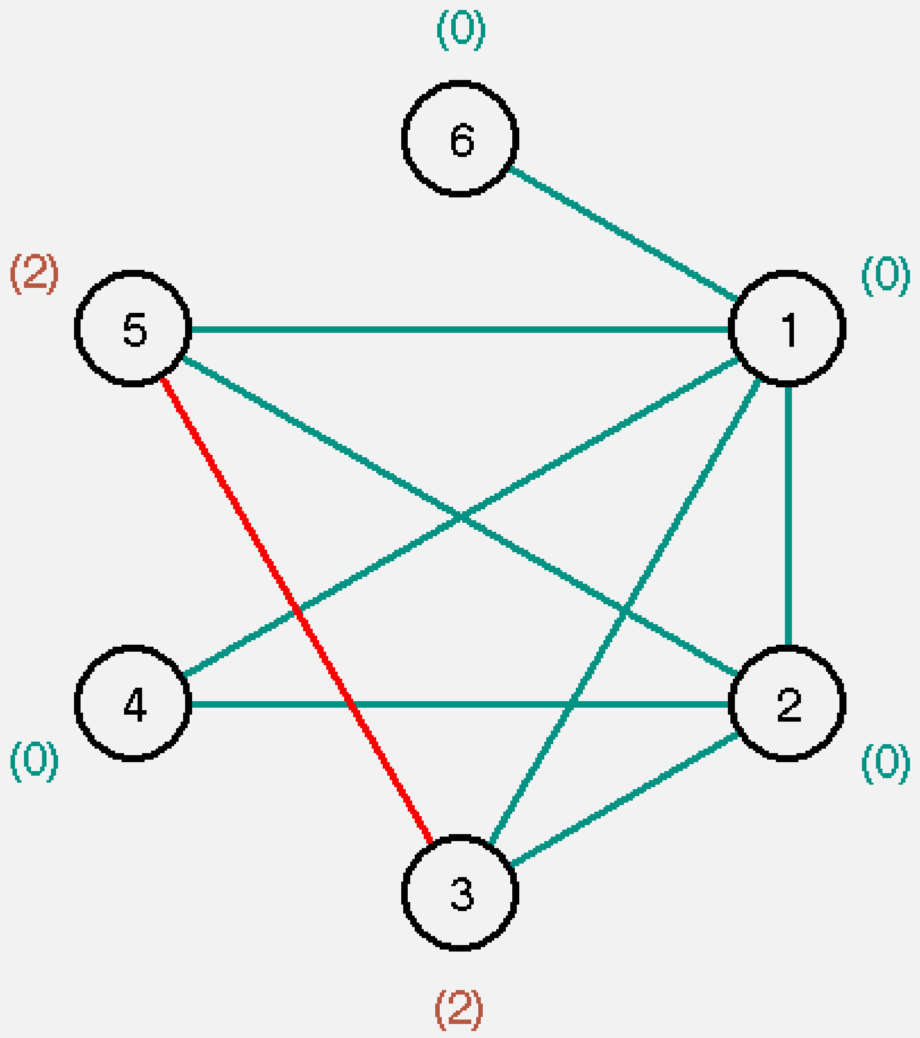
Etichete noduri = { 3, 5, 4, 6}.

# Exemple aplicare algoritm

**Pasul 3**

$s_2 = \{ \quad 2, 2, 0, 0\}.$

Etichete noduri = { 3, 5, 4, 6}.

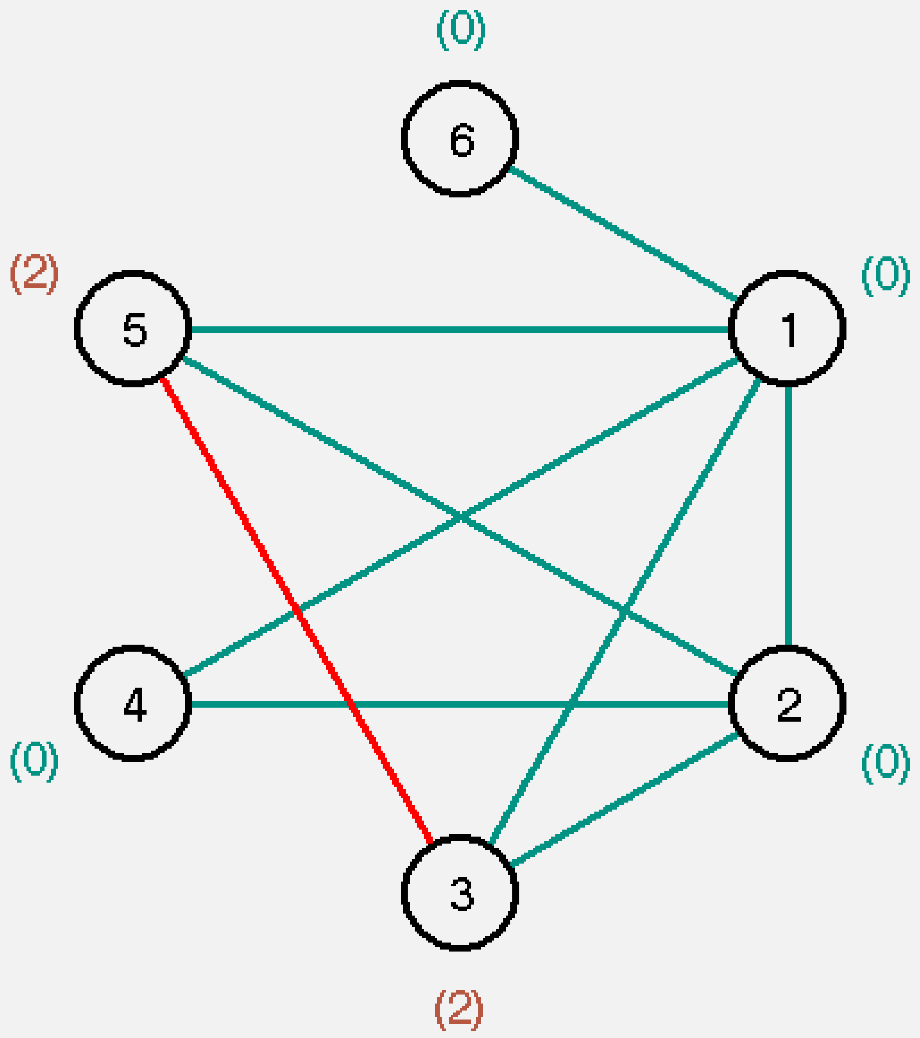


# Exemple aplicare algoritm

**Pasul 3**

$s_2$= {      2, 2, 0, 0}.

Etichete noduri = {      3, 5, 4, 6}.




**STOP! Nu există soluție.**

# Havel-Hakimi implementare

1. Dacă $d_1 + \ldots + d_n$ este impar sau există în $s_0$ un $d_i > n-1$, atunci scrie NU, STOP

# Havel-Hakimi implementare

1. Dacă $d_1 + \ldots + d_n$ este impar sau există în $s_0$ un $d_i > n\text{-}1$, 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 $d_{i1}, \ldots, d_{idk}$ **cele mai mari $d_k$ numere** din $s_0$

# Havel-Hakimi implementare

1. Dacă $d_1 + \ldots + d_n$ 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 $d_{i1}, \ldots, d_{idk}$ **cele mai mari $d_k$ numere** din $s_0$
   - Pentru $j \in \{i_1, \ldots, i_{dk}\}$ execută
     - Adaugă muchia $x_k x_j$ la G
     - Înlocuiește $d_j$ în secvența $s_0$ cu $d_j - 1$
     - Dacă $d_j < 0$, atunci scrie NU, STOP

# Havel-Hakimi implementare

**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.

# Havel-Hakimi implementare

**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.

- **Idei?**
- **Complexitate?**

# Havel-Hakimi complexitate algoritm

- O soluție se folosește de heapuri. Această soluție are complexitatea $O((N+M)*\log(N))$.

# Havel-Hakimi complexitate algoritm

- O soluție se folosește de heapuri. Această soluție are complexitatea O((N+M)*log(N)).
- O altă soluție, care nu ne oferă mulțimea muchiilor, se folosește de treapuri și de proprietatea de spargere a unui treap. Complexitatea acestei soluții este O(N*log(N)).

# Havel-Hakimi complexitate algoritm

- O soluție se folosește de heapuri. Această soluție are complexitatea O((N+M)*log(N)).
- O altă soluție, care nu ne oferă mulțimea muchiilor, se folosește de treapuri și de proprietatea de spargere a unui treap. Complexitatea acestei soluții este O(N*log(N)).
- O altă soluție se folosește de counting sort și poate atinge complexitatea O(N+M), oferind în același timp și mulțimea muchiilor.

# Havel-Hakimi complexitate algoritm

- O soluție se folosește de heapuri. Această soluție are complexitatea $O((N+M)*\log(N))$.
- O altă soluție, care nu ne oferă mulțimea muchiilor, se folosește de treapuri și de proprietatea de spargere a unui treap. Complexitatea acestei soluții este $O(N*\log(N))$.
- O altă soluție se folosește de counting sort și poate atinge complexitatea $O(N+M)$, oferind în același timp și mulțimea muchiilor. Această soluție sortează descrescător gradele și reține pentru fiecare grad ultima apariție. Scăderea cu unu a primelor x grade poate rezulta în pierderea proprietății de monotonie a șirului.

# Havel-Hakimi complexitate algoritm

- O altă soluție se folosește de counting sort și poate atinge complexitatea O(N+M), oferind în același timp și mulțimea muchiilor. Această soluție sortează descrescător gradele și reține pentru fiecare grad ultima apariție. Scăderea cu unu a primelor x grade poate rezulta în pierderea proprietății de monotonie a șirului. Pentru a nu face asta solutia folosește ultima poziție a fiecărui grad pentru a verifica dacă poate scădea cu 1 din toate nodurile cu acel grad. Dacă nu, acesta va scădea cu 1 doar ultimele x poziții ale acelui grad.

# Havel-Hakimi complexitate algoritm

- O altă soluție se folosește de counting sort și poate atinge complexitatea O(N+M), oferind în același timp și mulțimea muchiilor. Această soluție sortează descrescător gradele și reține pentru fiecare grad ultima apariție. Scăderea cu unu a primelor x grade poate rezulta în pierderea proprietății de monotonie a șirului. Pentru a nu face asta solutia folosește ultima poziție a fiecărui grad pentru a verifica dacă poate scădea cu 1 din toate nodurile cu acel grad. Dacă nu, acesta va scădea cu 1 doar ultimele x poziții ale acelui grad. În timp ce scade gradele algoritmul va actualiza valorile corespunzătoare ultimelor poziții ale gradelor.

# Teorema Havel-Hakimi demonstrație corectitudine

Pentru a demonstra că algoritmul precedent este complet și corect vom folosi dubla incluziune.

# Teorema Havel-Hakimi demonstrație corectitudine

Pentru a demonstra că algoritmul precedent este complet și corect vom folosi dubla incluziune.

"$\Leftarrow$": Dacă secvența $s_1 = \{d_2-1, \ldots, d_x-1, d_{x+1}-1, d_{x+2}, \ldots, d_{n-1}, d_n\}$ este grafică (se poate forma un graf cu aceste grade), evident și secvența $s_0 = \{d_1 = x, d_2, \ldots, d_x, d_{x+1}, d_{x+2}, \ldots, d_{n-1}, d_n\}$ este grafică.

# Teorema Havel-Hakimi demonstrație corectitudine

Pentru a demonstra că algoritmul precedent este complet și corect vom folosi dubla incluziune.

"$\Leftarrow$": Dacă secvența $s_1 = \{d_2-1, \dots, d_x-1, d_{x+1}-1, d_{x+2}, \dots, d_{n-1}, d_n\}$ este grafică (se poate forma un graf cu aceste grade), evident și secvența $s_0 = \{d_1 = x, d_2, \dots, d_x, d_{x+1}, d_{x+2}, \dots, d_{n-1}, d_n\}$ este grafică.

"$\Rightarrow$": Dacă secvența $s_0 = \{d_1 = x, d_2, \dots, d_x, d_{x+1}, d_{x+2}, \dots, d_{n-1}, d_n\}$ este grafică atunci și secvența $s_1 = \{d_2-1, \dots, d_x-1, d_{x+1}-1, d_{x+2}, \dots, d_{n-1}, d_n\}$ este grafică.

# Teorema Havel-Hakimi demonstrație corectitudine

"$\Rightarrow$": Dacă secvența $s_0 = \{d_1 = x, d_2, \ldots, d_x, d_{x+1}, d_{x+2}, \ldots, d_{n-1}, d_n\}$ este grafică atunci și secvența $s_1 = \{d_2-1, \ldots, d_x-1, d_{x+1}-1, d_{x+2}, \ldots, d_{n-1}, d_n\}$ este grafică.

# Teorema Havel-Hakimi demonstrație corectitudine

"$\Rightarrow$": Dacă secvența $s_0 = \{d_1 = x, d_2, \ldots, d_x, d_{x+1}, d_{x+2}, \ldots, d_{n-1}, d_n\}$ este grafică atunci și secvența $s_1 = \{d_2-1, \ldots, d_x-1, d_{x+1}-1, d_{x+2}, \ldots, d_{n-1}, d_n\}$ este grafică.

Fie S mulțimea grafurilor ce au secvența gradelor egală cu $s_0$. Fie $G_1 \in S$, astfel încât suma gradelor vecinilor nodului 1 este maximă.

# Teorema Havel-Hakimi demonstrație corectitudine

"⇒": Dacă secvența $s_0 = \{d_1 = x, d_2, \ldots, d_x, d_{x+1}, d_{x+2}, \ldots, d_{n-1}, d_n\}$ este grafică atunci și secvența $s_1 = \{d_2-1, \ldots, d_x-1, d_{x+1}-1, d_{x+2}, \ldots, d_{n-1}, d_n\}$ este grafică.

Fie S mulțimea grafurilor ce au secvența gradelor egală cu $s_0$. Fie $G_1 \in S$, astfel încât suma gradelor vecinilor nodului 1 este maximă.

Să considerăm că acesta nu conectează nodul 1 cu nodurile cu cele mai mari grad.

# Teorema Havel-Hakimi demonstrație corectitudine

"⇒": Dacă secvența $s_0 = \{d_1 = x, d_2, \ldots, d_x, d_{x+1}, d_{x+2}, \ldots, d_{n-1}, d_n\}$ este grafică atunci și secvența $s_1 = \{d_2-1, \ldots, d_x-1, d_{x+1}-1, d_{x+2}, \ldots, d_{n-1}, d_n\}$ este grafică.

Fie S mulțimea grafurilor ce au secvența gradelor egală cu $s_0$. Fie $G_1 \in S$, astfel încât suma gradelor vecinilor nodului 1 este maximă.

Să considerăm că acesta nu conectează nodul 1 cu nodurile cu cele mai mari grad. Asta înseamnă că există un nod u care este conectat cu 1, și un nod v care nu este conectat cu 1, și care are gradul mai mare decât u ($d_u < d_v$).

# Teorema Havel-Hakimi demonstrație corectitudine

"⇒": Dacă secvența $s_0 = \{d_1 = x, d_2, \dots, d_x, d_{x+1}, d_{x+2}, \dots, d_{n-1}, d_n\}$ este grafică atunci și secvența $s_1 = \{d_2-1, \dots, d_x-1, d_{x+1}-1, d_{x+2}, \dots, d_{n-1}, d_n\}$ este grafică.

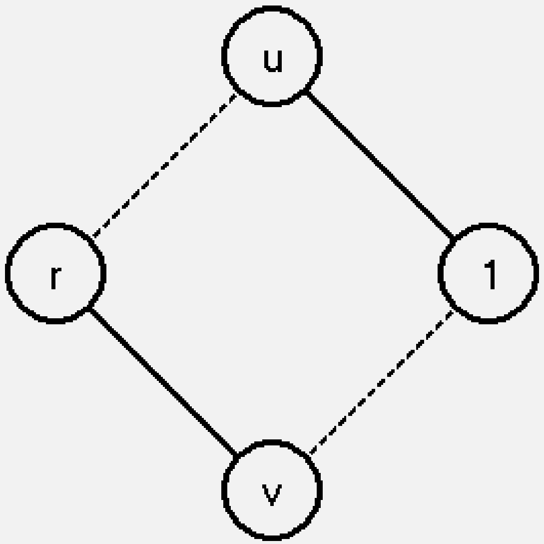
Fie S mulțimea grafurilor ce au secvența gradelor egală cu $s_0$. Fie $G_1 \in S$, astfel încât suma gradelor vecinilor nodului 1 este maximă.

Să considerăm că acesta nu conectează nodul 1 cu nodurile cu cele mai mari grad. Asta înseamnă că există un nod u care este conectat cu 1, și un nod v care nu este conectat cu 1, și care are gradul mai mare decât u ($d_u < d_v$).

Deci, există un nod r care este conectat cu nodul v, dar nu este conectat cu nodul u.

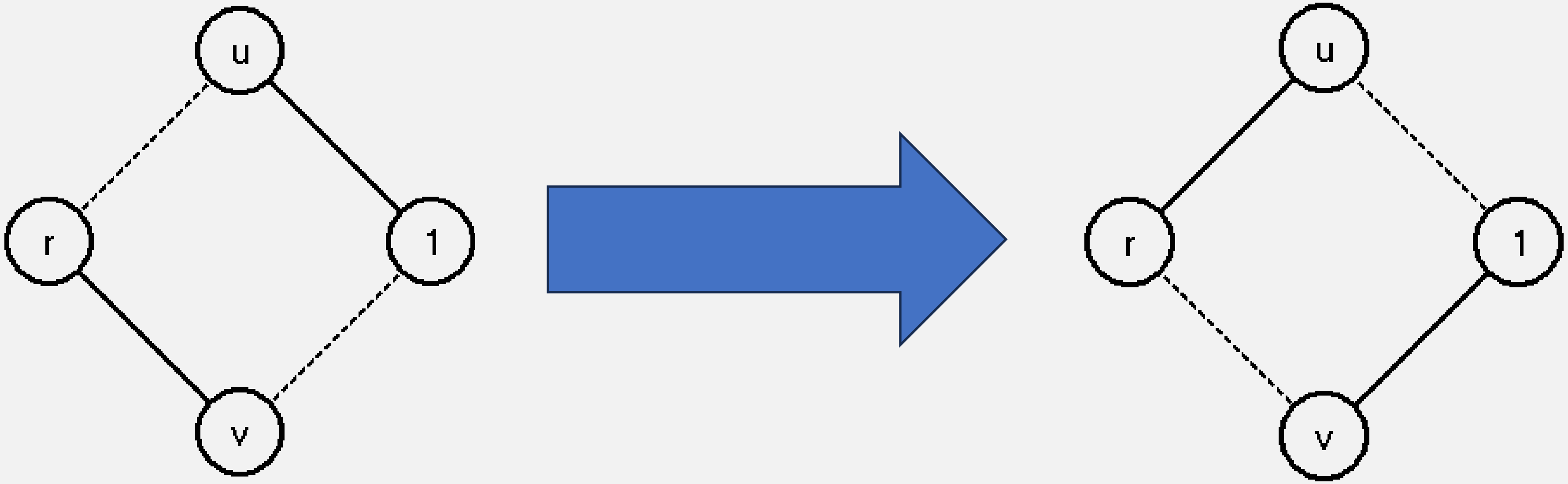
# Teorema Havel-Hakimi demonstrație corectitudine

Deci, există un nod r care este conectat cu nodul v, dar nu este conectat cu nodul u. (vezi imaginea de mai jos)



# Teorema Havel-Hakimi demonstrație corectitudine

Deci, există un nod r care este conectat cu nodul v, dar nu este conectat cu nodul u. Dacă am elimina muchiile (1, u) și (r, v) și am adăuga muchiile (u, r) și (1, v) am obține un graf cu aceeași secvență de grade, dar care să aibă suma gradelor vecinilor lui 1 mai mare decât maximul. (Contradicție)

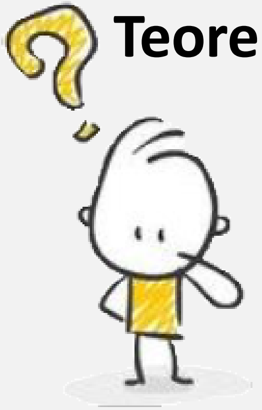


# Teorema Havel-Hakimi demonstrație corectitudine

Deci, există un nod r care este conectat cu nodul v, dar nu este conectat cu nodul u. Dacă am elimina muchiile (1, u) și (r, v) și am adăuga muchiile (u, r) și (1, v) am obține un graf cu aceeași secvență de grade, dar care să aibă suma gradelor vecinilor lui 1 mai mare decât maximul. (Contradicție)

Deci presupunerea inițială este falsă ⇒ Graful maximal cu suma gradelor vecinilor nodului 1 este cel în care conectăm nodul 1 cu nodurile cu grad maxim ⇒ dacă există soluție, algoritmul o găsește întotdeauna.

# Teorema Havel-Hakimi



**Teorema Havel-Hakimi**

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

# Teorema Havel-Hakimi

**Teorema Havel-Hakimi**

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

**Nu intervine! Se poate renunța la această ipoteză.** $\Rightarrow$

# Teorema Havel-Hakimi - Extindere

**Teorema Havel-Hakimi**

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

**Nu intervine! Se poate renunța la această ipoteză.** $\Rightarrow$

$\Rightarrow$ **Extindere a teoremei Havel-Hakimi**

# Teorema Havel-Hakimi - Extindere

**Teorema Havel-Hakimi**

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

**Nu intervine! Se poate renunța la această ipoteză.** ⇒

⇒ **Extindere a teoremei Havel-Hakimi**

Fie $s_0 = \{d_1, \ldots, d_n\}$ o secvență de $n \geq 2$ numere naturale mai mici sau egale cu $n-1$ și fie $i \in \{1, \ldots, n\}$ fixat.

# Teorema Havel-Hakimi - Extindere

**Teorema Havel-Hakimi**

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

**Nu intervine! Se poate renunța la această ipoteză.** $\Rightarrow$

$\Rightarrow$ **Extindere a teoremei Havel-Hakimi**

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

Fie secvența obținută din $s_0$ astfel:

- Eliminăm elementul $d_i$
- Scădem o unitate din primele $d_i$ componente, în ordine descrescătoare a secvenței rămase

# Teorema Havel-Hakimi - Extindere

Fie secvența obținută din $s_0$ astfel:

- Eliminăm elementul $d_i$
- Scădem o unitate din primele $d_i$ componente, în ordine descrescătoare a secvenței rămase

**Are loc echivalența:**

$s_0$ este secvența gradelor unui graf neorientat $\Leftrightarrow$

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

(unde $s_0^{(i)}$ reprezintă secvența după eliminarea nodului i)

# Teorema Havel-Hakimi - Extindere

Fie secvența obținută din $s_0$ astfel:

- Eliminăm elementul $d_i$
- Scădem o unitate din primele $d_i$ componente, în ordine descrescătoare a secvenței rămase

**Are loc echivalența:**

$s_0$ este secvența gradelor unui graf neorientat $\Leftrightarrow$

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

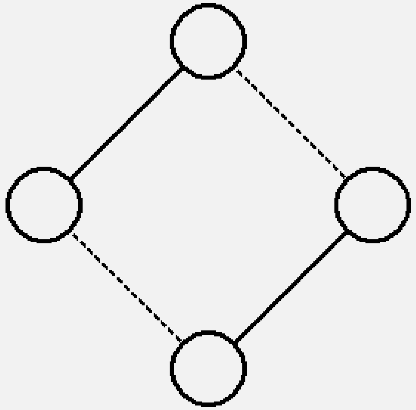
(unde $s_0^{(i)}$ reprezintă secvența după eliminarea nodului i)

**Concluzie:**

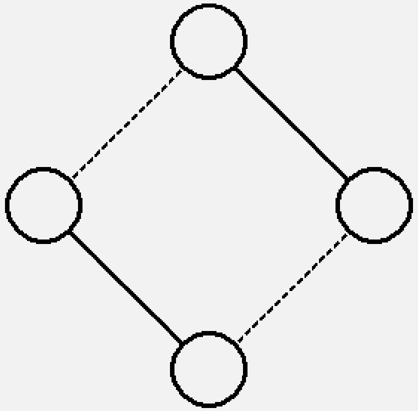
**La un pas, nodul poate fi ales arbitrar (nu neapărat cel corespunzător elementului maxim). Totuși, se păstrează criteriul de alegere al vecinilor (cei cu gradele cele mai mari).**

# Extindere

Cu ajutorul transformării pe următorul pătrat, putem obține, pornind de la un graf G, toate grafurile cu secvența gradelor s(G) și mulțimea nodurilor V(G).



$\Rightarrow$

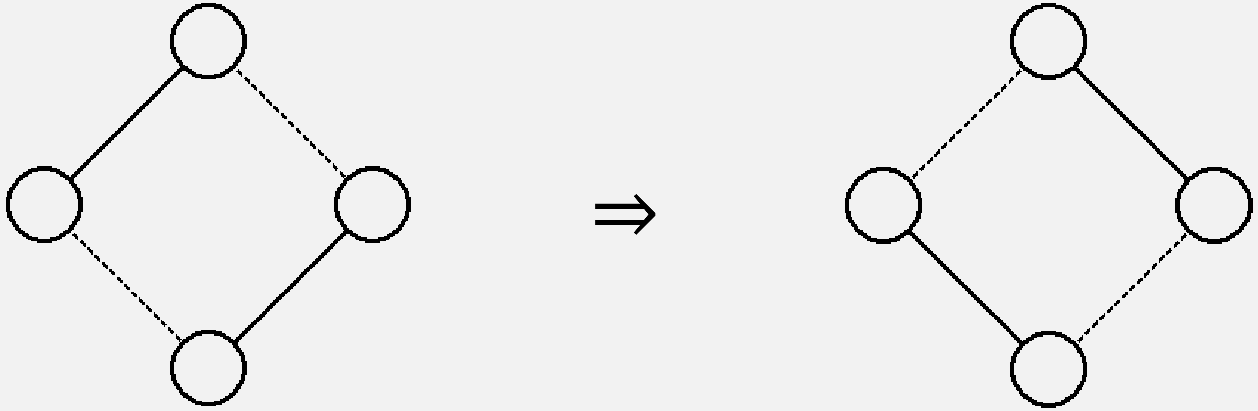


# Extindere

Cu ajutorul transformării pe următorul pătrat, putem obține, pornind de la un graf G, toate grafurile cu secvența gradelor s(G) și mulțimea nodurilor V(G).

Mai exact, are loc următorul rezultat:

- Fie $G_1$ și $G_2$ două grafuri neorientate cu mulțimea nodurilor $V = \{1, \ldots, n\}$.
- Atunci $s(G_1) = s(G_2) \Leftrightarrow$ există un șir de transformări de interschimbare pe pătrat, prin care se poate obține graful $G_2$ din $G_1$.



# Teorema Erdös-Gallai (suplimentar)

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

$$\Leftrightarrow \begin{cases} d_1 + \ldots + d_n \text{ număr par} \\ d_1 + \ldots + d_k \leq k * (k-1) + \sum_{i=k+1}^{n} min\{d_i, k\}, \forall 1 \leq k \leq n \end{cases}$$

# Teorema Erdös-Gallai (suplimentar)

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

$$\Leftrightarrow \begin{cases} d_1 + \ldots + d_n \text{ număr par} \\ d_1 + \ldots + d_k \leq k * (k-1) + \sum_{i=k+1}^{n} min\{d_i, k\}, \forall 1 \leq k \leq n \end{cases}$$

Aceste n+1 condiții pot fi verificate împreună in complexitatea O(N) pentru a verifica dacă acea secvență este grafică. De menționat din nou că acest algoritm NU ne oferă și graful.