

Construcția de grafuri cu secvența gradelor dată

Secvențe de grade



Data o formulă chimică, există un compus chimic care are această formulă?

Dar unul aciclic?

Ce structuri poate avea un astfel de compus?

- ☐ $C_m H_n$ - 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



Data 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

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

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 suficiente**

Totuși, putem crea un multigraf

Construcția de grafuri cu secvența gradelor dată



Idee de algoritm de construcție a unui graf 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

Construcția de grafuri cu secvența gradelor dată

Exemplu

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

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
- ☐ alegem ca vecini următoarele vârfuri cu cele mai mari grade

Construcția de grafuri cu secvența gradelor dată



Idee de algoritm de construcție a unui graf 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
 - secvența conține doar 0 $\Rightarrow G$
 - secvența conține numere negative \Rightarrow

Construcția de grafuri cu secvența gradelor dată

Idee de algoritm de construcție a unui graf 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
 - 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 de algoritm de construcție a unui graf 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
 - 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-Hakimi**

Exemplu algoritm Havel–Hakimi

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

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

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

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

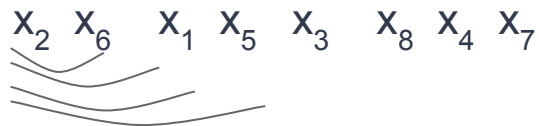
x_2 x_6 x_1 x_5 x_3 x_8 x_4 x_7

Exemplu algoritm Havel-Hakimi

Pasul 1

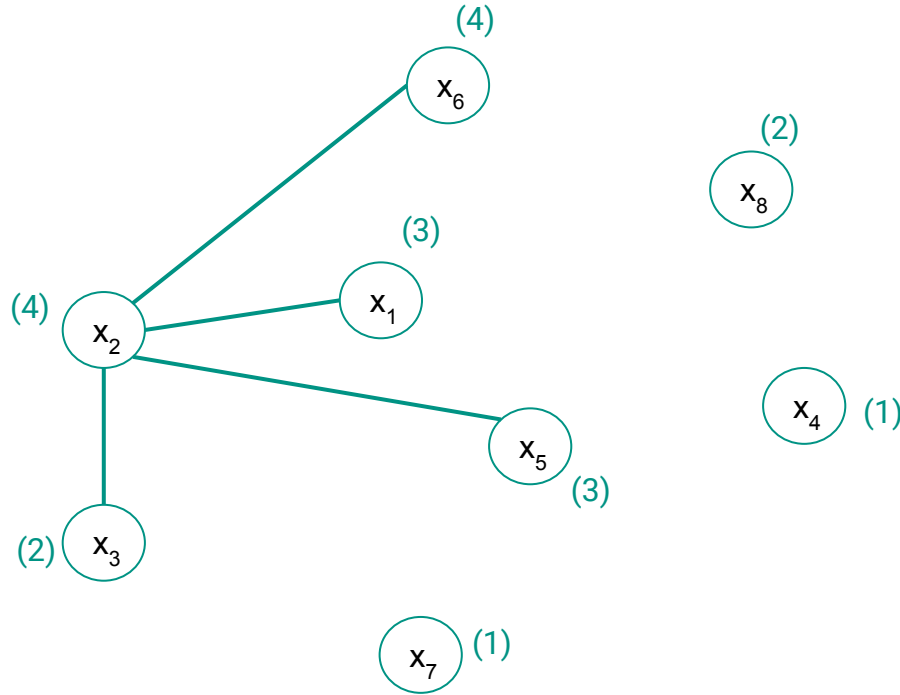
$$s_0 = \{4, 4, 3, 3, 2, 2, 1, 1\}$$

etichete vârfuri



Muchii construite: x_2x_6 , x_2x_1 , x_2x_5 , x_2x_3

Exemplu algoritm Havel-Hakimi

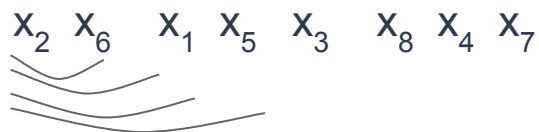


Exemplu algoritm Havel–Hakimi

Pasul 1

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

etichete vârfuri



Muchii construite: x_2x_6 , x_2x_1 , x_2x_5 , x_2x_3

Secvența rămasă:

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

etichete vârfuri

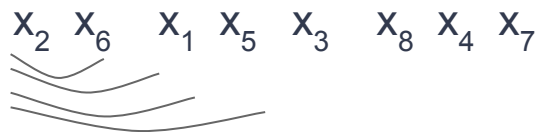
x_6 x_1 x_5 x_3 x_8 x_4 x_7

Exemplu algoritm Havel–Hakimi

Pasul 1

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

etichete vârfuri



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

Secvența rămasă:

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

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

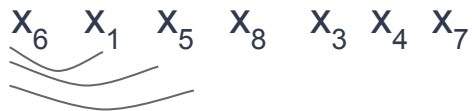
$x_6 \ x_1 \ x_5 \ x_8 \ x_3 \ x_4 \ x_7$

Exemplu algoritm Havel–Hakimi

Pasul 2

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

etichete vârfuri



Muchii construite: x_6x_1, x_6x_5, x_6x_8

Secvența rămasă:

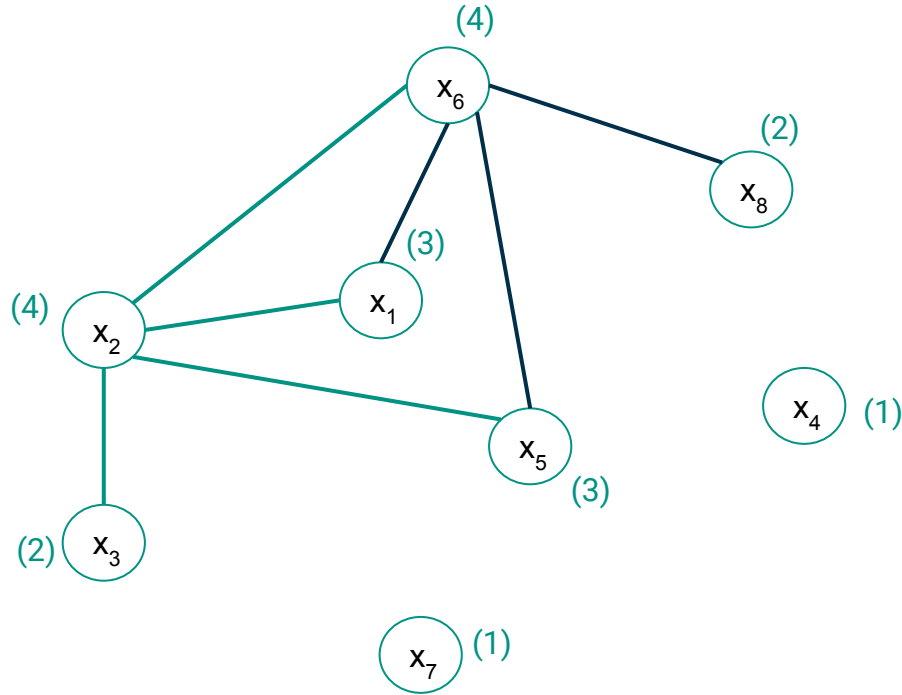
$$s''_0 = \{ \textcolor{red}{1}, \textcolor{red}{1}, \textcolor{red}{1}, 1, 1, 1 \}$$

etichete vârfuri

$x_1, x_5, x_8, x_3, x_4, x_7$

(este ordonată descrescător)

Exemplu algoritm Havel-Hakimi



Exemplu algoritm Havel–Hakimi

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

etichete vârfuri $x_1 \quad x_5 \quad x_8 \quad x_3 \quad x_4 \quad x_7$



Muchii construite: x_1x_5

Secvența rămasă:

$s'''_0 = \{ 0, 1, 1, 1, 1 \}$

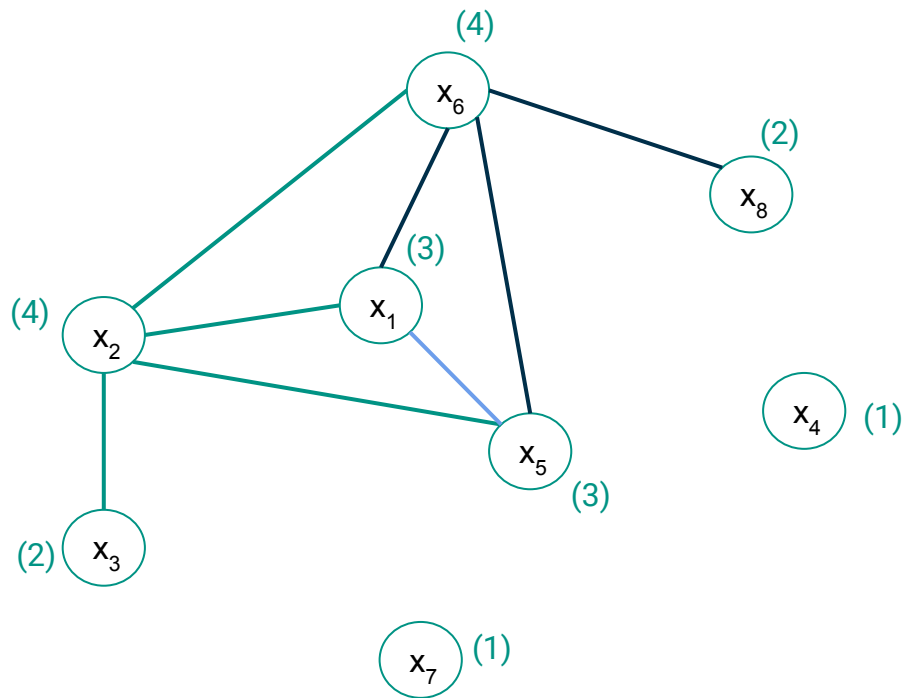
etichete vârfuri $x_5 \quad x_8 \quad x_3 \quad x_4 \quad x_7$

Secvența rămasă ordonată descrescător:


$s'''_0 = \{ 1, 1, 1, 1, 0 \}$

etichete vârfuri $x_7 \quad x_3 \quad x_4 \quad x_8 \quad x_5$

Exemplu algoritm Havel-Hakimi



Exemplu algoritm Havel–Hakimi

Pasul 4 $s'''_0 = \{ 1, 1, 1, 1, 0 \}$
etichete vârfuri $x_7 \ x_3 \ x_4 \ x_8 \ x_5$


Muchii construite: x_7x_3

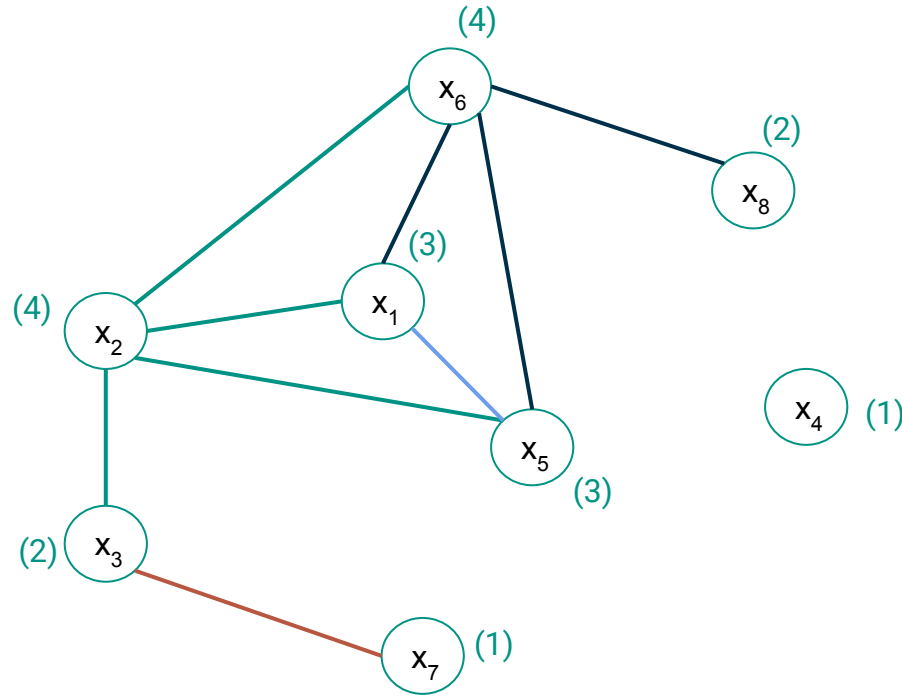
Secvența rămasă:

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

Secvența rămasă ordonată descrescător:

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

Exemplu algoritm Havel-Hakimi



Exemplu algoritm Havel–Hakimi

Pasul 5

$$s''''_0 = \{ \quad 1, \quad 1, \quad 0, \quad 0 \}$$

etichete vârfuri

$$\underbrace{x_4 \quad x_8} \quad x_3 \quad x_5$$

Muchii construite: $x_4 x_8$

Secvența rămasă:

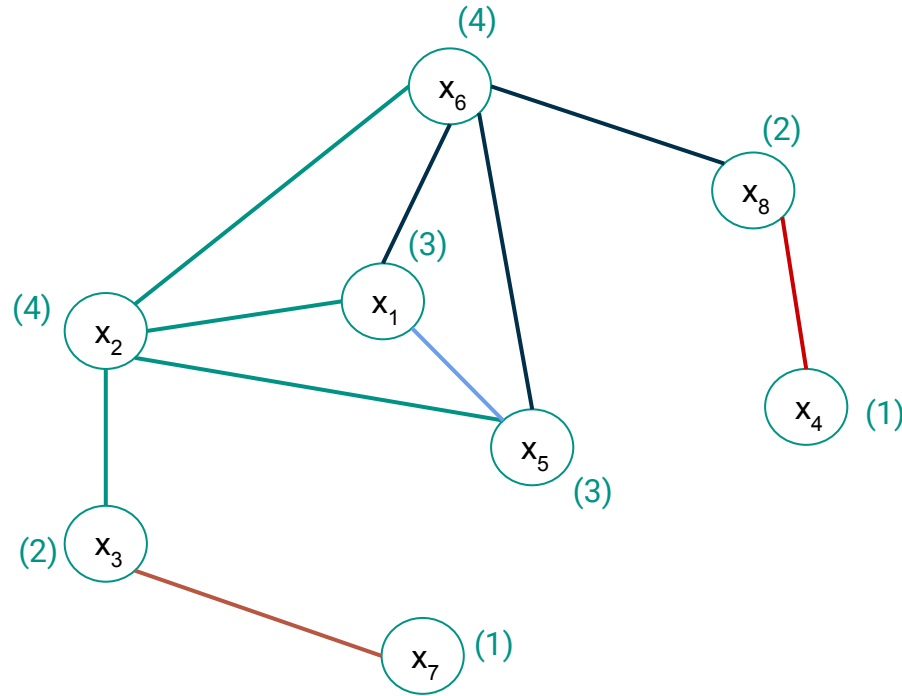
$$s''''_0 = \{ \quad 0, \quad 0, \quad 0 \}$$

etichete vârfuri

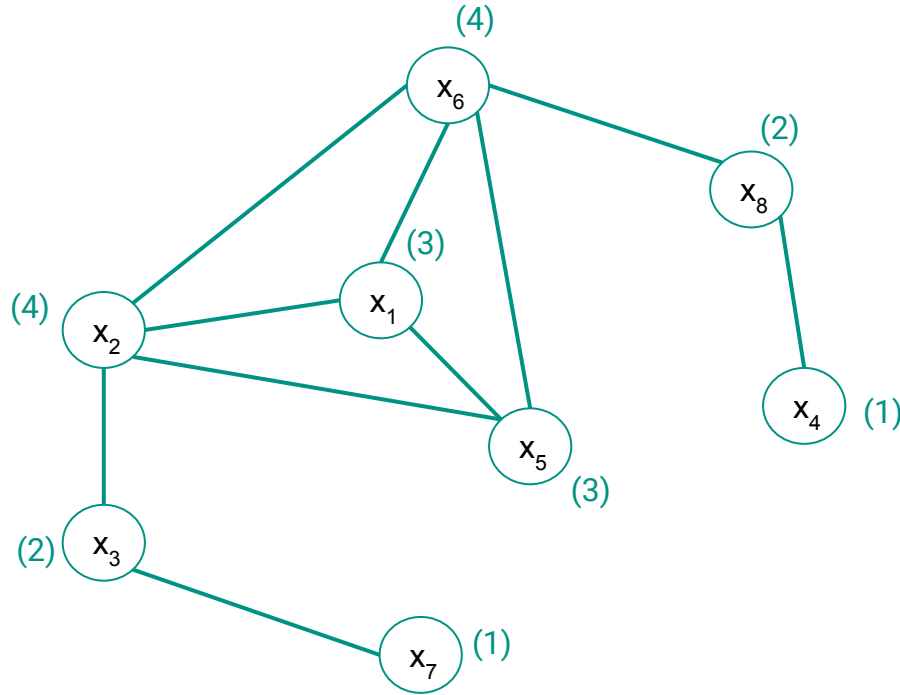
$$x_8 \quad x_3 \quad x_5$$

STOP

Exemplu algoritm Havel-Hakimi



Exemplu algoritm Havel-Hakimi



Algoritm 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

Algoritm 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_k} **cele mai mari d_k numere din s_0**

Algoritm 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

Algoritm 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 muchia $x_k x_j$ la G

înlocuiește d_j în secvența s_0 cu **$d_j - 1$**

dacă $d_j - 1 < 0$, atunci scrie NU, STOP

Algoritm 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_k} **cele mai mari d_k numere** din s_0

pentru $j \in \{i_1, \dots, i_k\}$ execută

adaugă muchia $x_k x_j$ la G

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.

Algoritm 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_k} **cele mai mari d_k numere** din s_0

pentru $j \in \{i_1, \dots, i_k\}$ execută

adaugă muchia $x_k x_j$ la G

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).

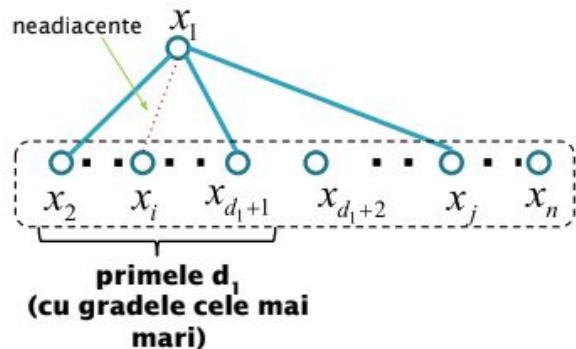
Observatie: Secvența $s'_0 = \{d_2 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n\}$ se obține din s_0 **eliminând primul element (d_1)** și scăzând 1 din primele d_1 elemente rămase – acestea au indicii $2, 3, \dots, d_1+1$.

Algoritm Havel-Hakimi - Corectitudine

Teorema Havel-Hakimi - Demonstrație

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

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

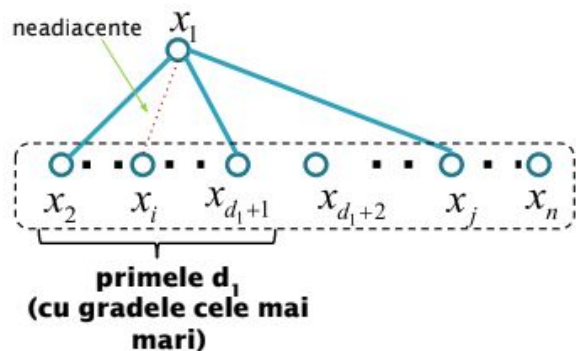


Algoritm Havel-Hakimi - Corectitudine

Teorema Havel-Hakimi - Demonstrație

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

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

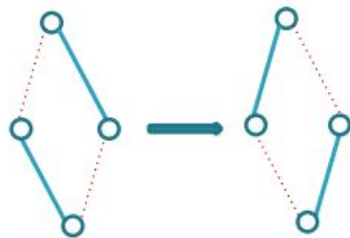
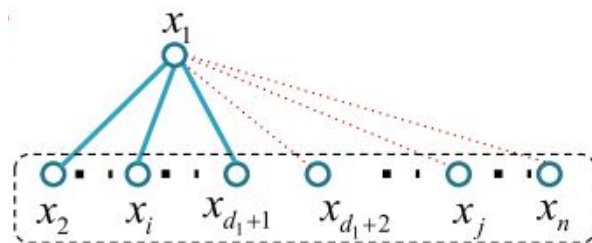


transformare
t pe pătrat



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

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

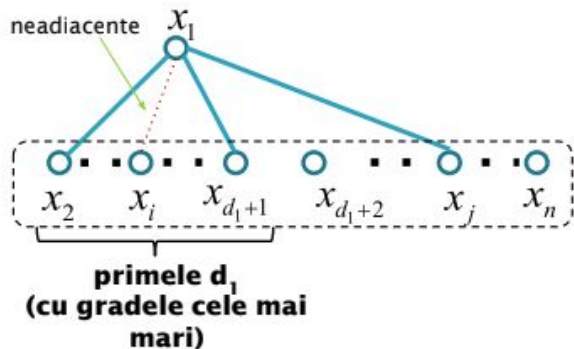


Algoritm Havel-Hakimi - Corectitudine

Teorema Havel-Hakimi - Demonstrație

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

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

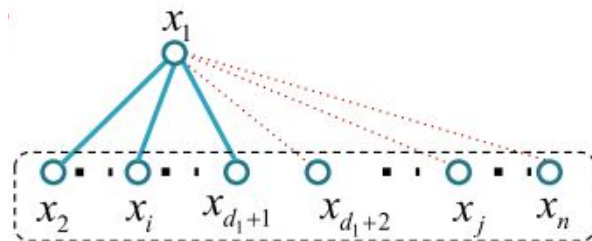


transformare
t pe pătrat

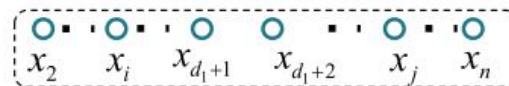


$$G^*, s(G^*) = s_0$$

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



elimin
 x_1



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

Algoritm Havel-Hakimi - Corectitudine

Teorema Havel-Hakimi - Demonstrație

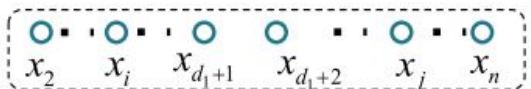
$$s_0 = \{d_1 \geq \dots \geq d_n\} \quad \Leftarrow \quad s'_0 = \{d_2 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n\}$$

Algoritm Havel-Hakimi - Corectitudine

Teorema Havel-Hakimi - Demonstrație

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

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

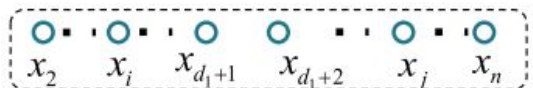
Teorema Havel-Hakimi - Demonstrație

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

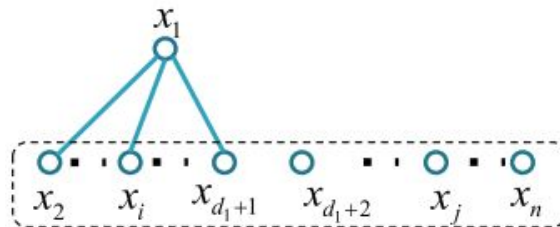
Fie G' cu $s(G') = s'_0$

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

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



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 mai mici sau egale cu $n-1$ și fie $i \in \{1, \dots, n\}$ fixat.
Fie 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 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

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 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

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 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

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 teoremei 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, are 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
- ☐ $d_1 + \dots + d_k \leq k(k-1) + \sum_{i=k+1}^n \min\{d_i, k\}, \forall 1 \leq k \leq n$

Muchii critice



Muchii critice

O muchie este **critică** \Leftrightarrow nu este conținută într-un ciclu.

Găsirea unui ciclu - parcurgere DF

- ☐ **muchii de avansare** - ale arborelui DF (memorat cu un vector de tați), prin care se descoperă vârfuri noi
- ☐ **muchii de întoarcere** - închid ciclu, nu pot fi critice

Muchii critice



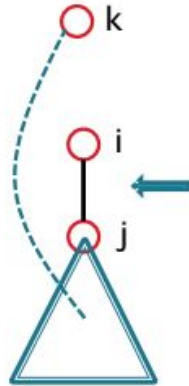
Cum testăm dacă o muchie de avansare (i, j) este critică?

Muchii critice



Cum testăm dacă o muchie de avansare (i, j) este critică?

- nu este conținută într-un ciclu închis de o muchie de întoarcere



Muchii critice

O muchie de avansare (i, j) este critică



nu este conținută într-un ciclu închis de o muchie de întoarcere



nu există nicio muchie de întoarcere cu

- ☐ o extremitate în **j sau într-un descendent al lui j**
- ☐ cealaltă extremitate în **i sau într-un ascendent al lui i** (într-un vârf de pe **un nivel mai mic sau egal** cu nivelul lui i)



Muchii critice

Memorăm, pentru fiecare vârf i :

$niv_min[i]$ = nivelul minim al unui vârf care este extremitate a unei muchii de întoarcere din i sau dintr-un descendent al lui i

= nivelul minim la care se închide un ciclu elementar care conține vârful i (printr-o muchie de întoarcere)



Muchii critice

- **nivel[i]** = nivelul lui *i* în arborele DF
- **niv_min[i]** = $\min \{ \text{nivel}[i], A, B \}$
 - $A = \min \{ \text{nivel}[k] \mid ik \text{ muchie de întoarcere} \}$
 - $B = \min \{ \text{nivel}[k] \mid j \text{ descendent al lui } i, jk \text{ muchie de întoarcere} \}$



Muchii critice

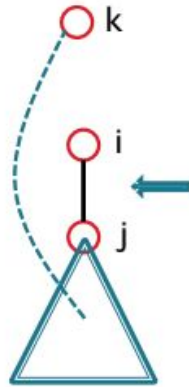
O muchie de avansare ij este critică

\Leftrightarrow



Muchii critice

O muchie de avansare ij este critică $\Leftrightarrow \text{niv_min}[j] > \text{nivel}[i]$



Muchii critice



Cum calculăm eficient $\text{niv_min}[i]$?

- $\text{niv_min}[i] = \min \{ \text{nivel}[i], A, B \}$
 - $A = \min \{ \text{nivel}[k] \mid ik \text{ muchie de întoarcere} \}$
 - $B = \min \{ \text{nivel}[k] \mid j \text{ descendent al lui } i, jk \text{ muchie de întoarcere} \}$

Muchii critice

Cum calculăm eficient $\text{niv_min}[i]$?

- $\text{niv_min}[i] = \min \{ \text{nivel}[i], A, B \}$
 - $A = \min \{ \text{nivel}[k] \mid i \text{ k muchie de întoarcere} \}$
 - $B = \min \{ \text{nivel}[k] \mid j \text{ descendent al lui } i, j \text{ k muchie de întoarcere} \}$



B se poate calcula recursiv

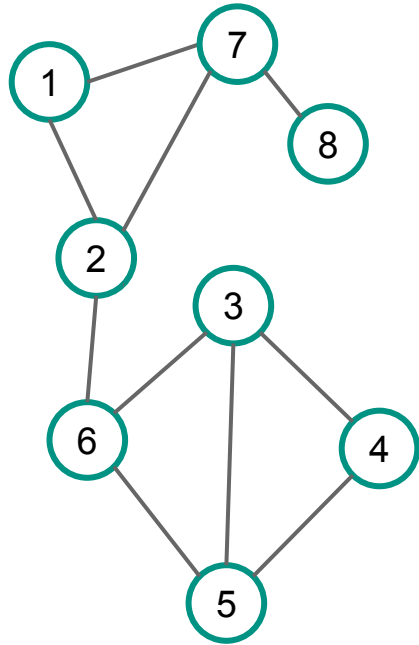
Muchii critice

Cum calculăm eficient $\text{niv_min}[i]$?

- $\text{niv_min}[i] = \min \{ \text{nivel}[i], A, B \}$
 - $A = \min \{ \text{nivel}[k] \mid ik \text{ muchie de întoarcere} \}$
 - $B = \min \{ \text{nivel}[k] \mid j \text{ descendent al lui } i, jk \text{ muchie de întoarcere} \}$

$$B = \min \{ \text{niv_min}[j] \mid j \text{ fiu al lui } i \}$$

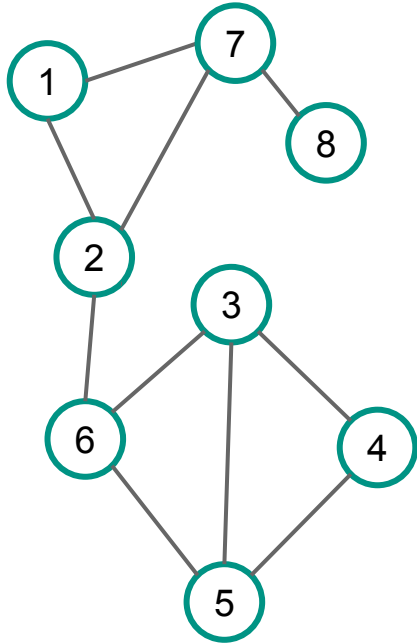
Muchii critice – Exemplu



Muchii critice – Exemplu

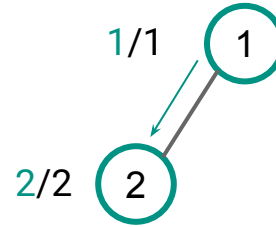
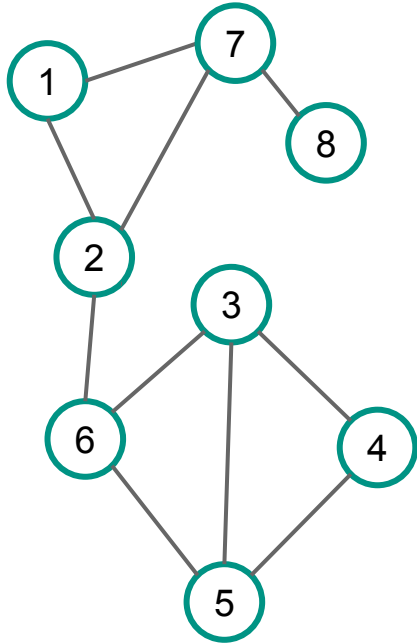
nivel/niv_min

1/1



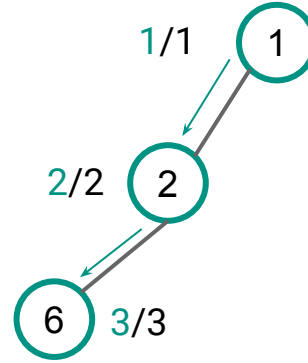
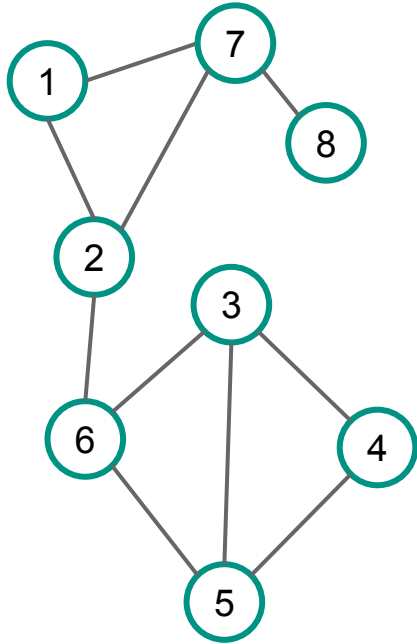
Muchii critice – Exemplu

nivel/niv_min



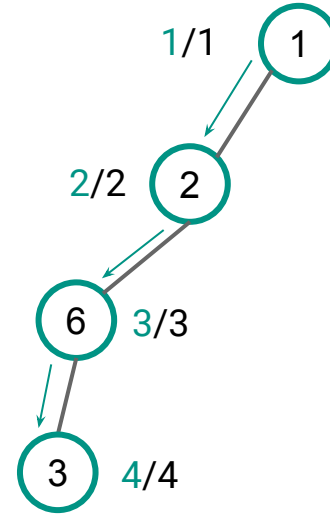
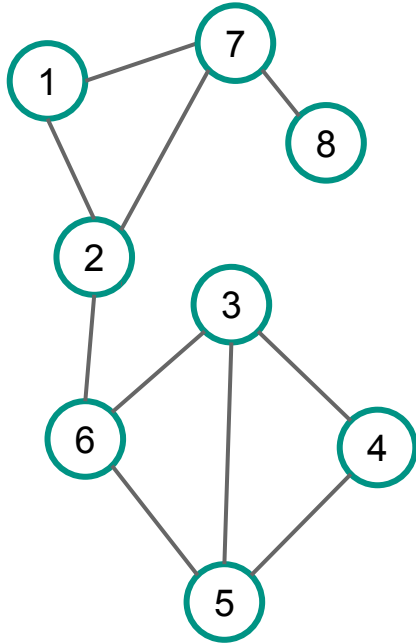
Muchii critice – Exemplu

nivel/niv_min



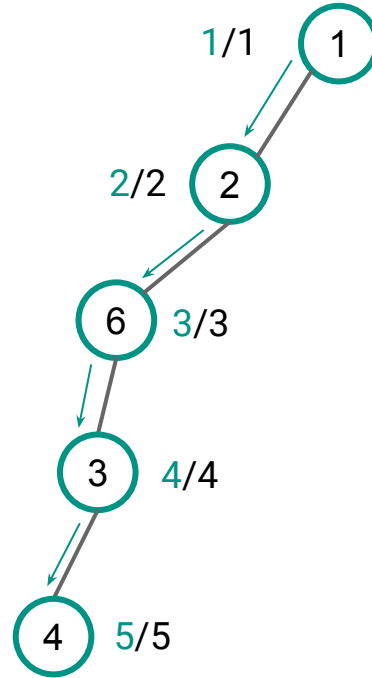
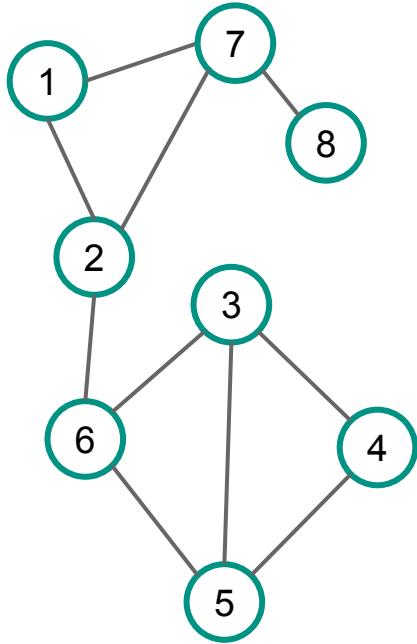
Muchii critice – Exemplu

nivel/niv_min



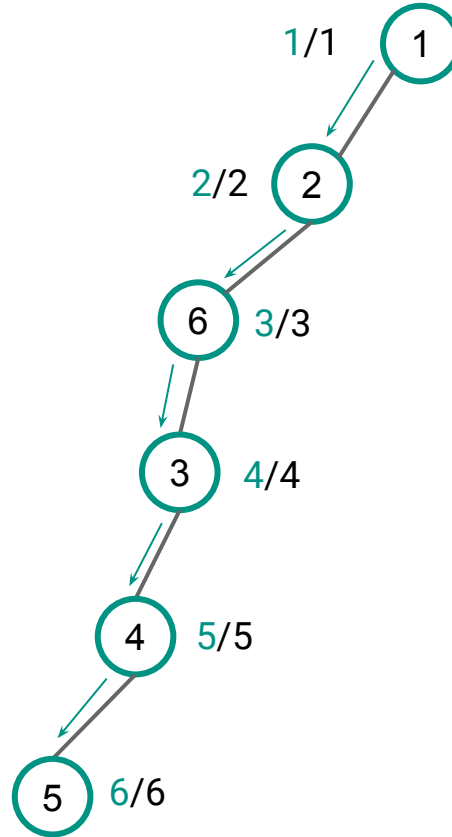
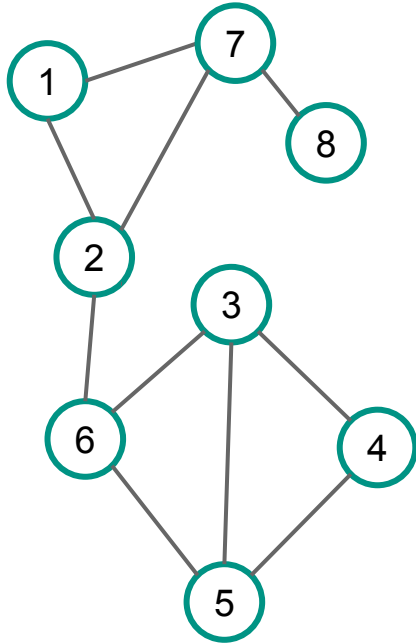
Muchii critice – Exemplu

nivel/niv_min



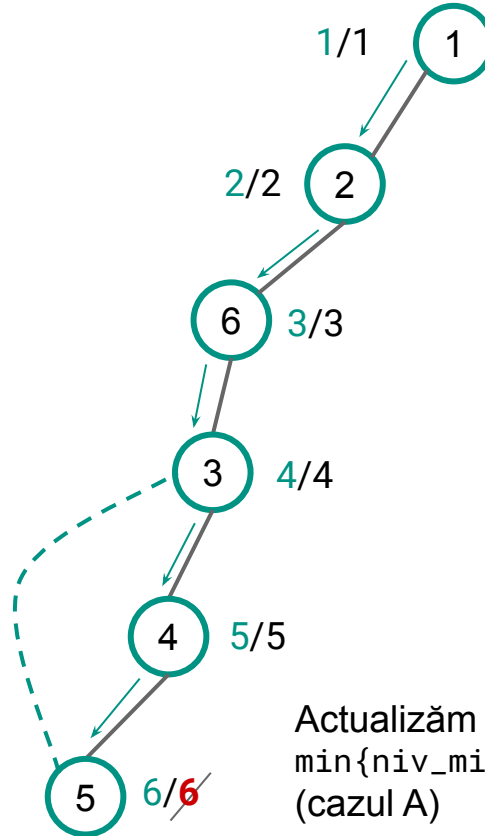
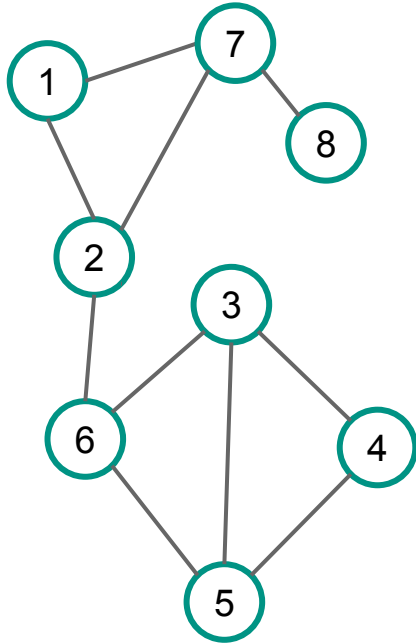
Muchii critice – Exemplu

nivel/niv_min



Muchii critice – Exemplu

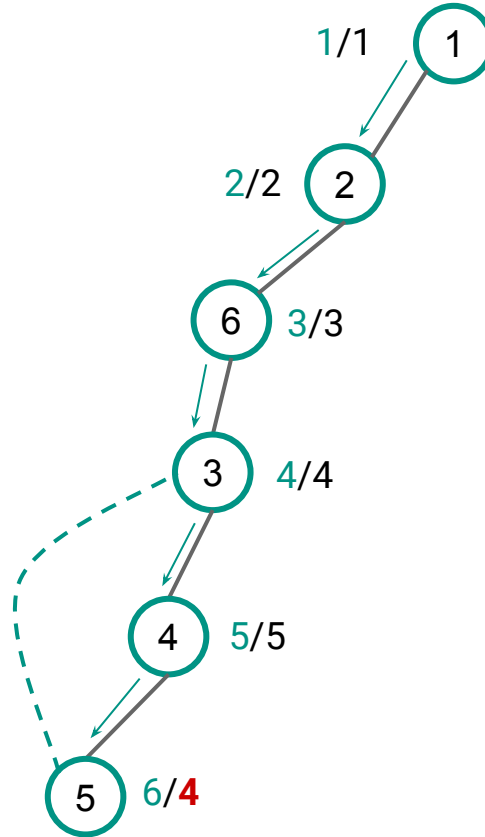
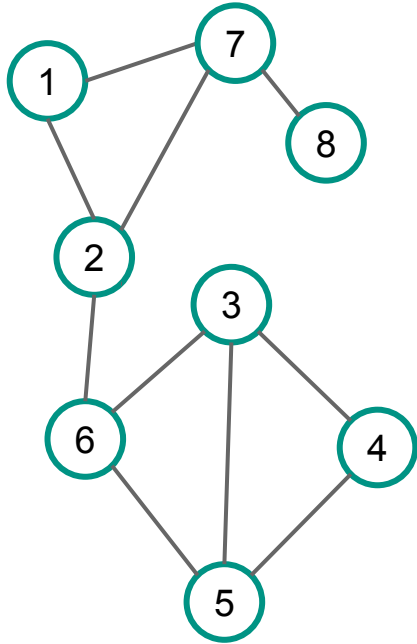
nivel/niv_min



Actualizăm $\text{niv_min}[5] = \min\{\text{niv_min}[5], \text{nivel}[3]\}$
(cazul A)

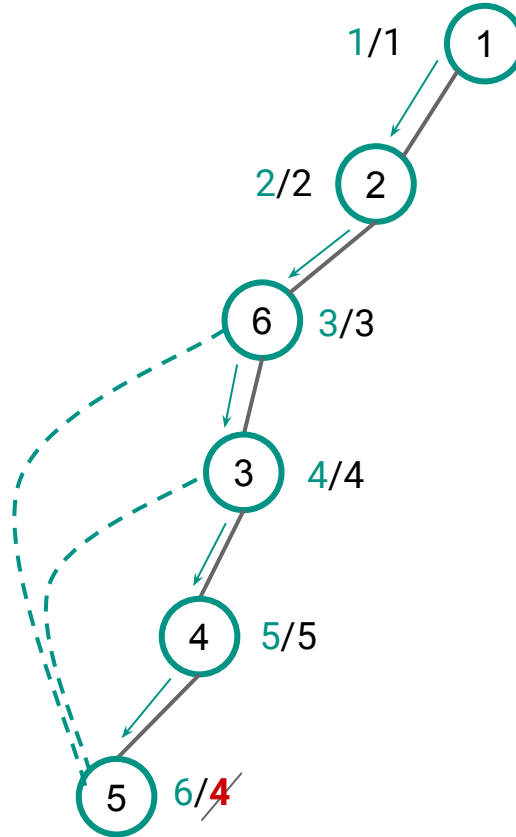
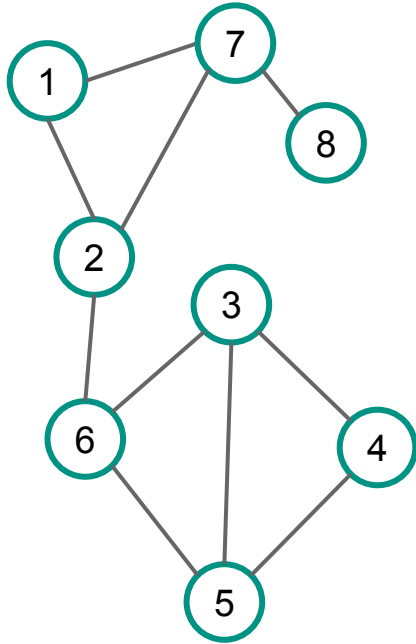
Muchii critice – Exemplu

nivel/niv_min



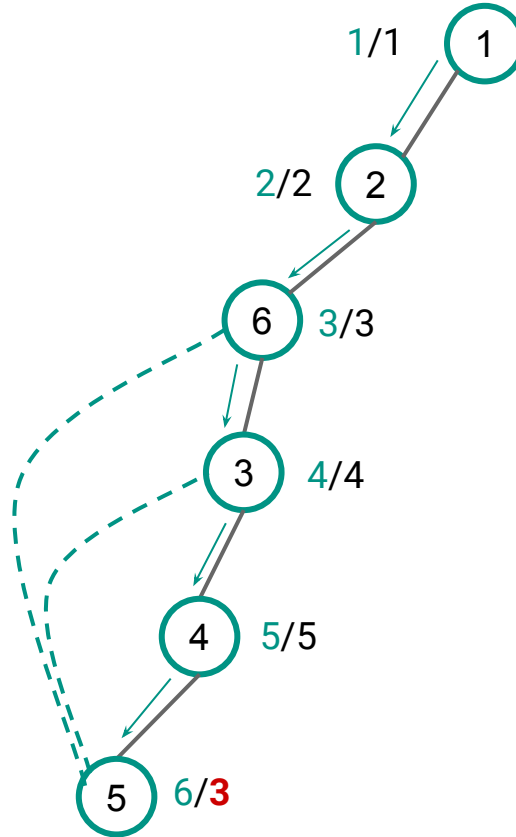
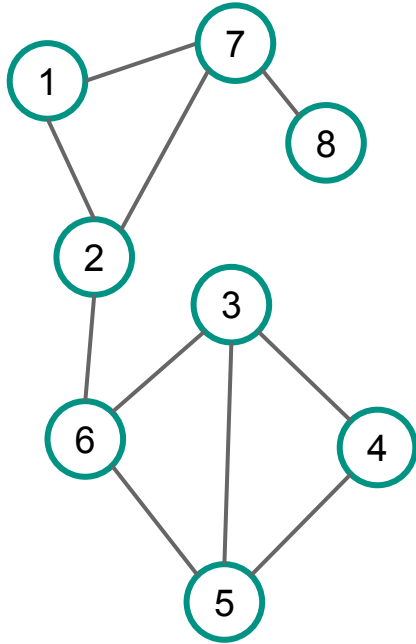
Muchii critice – Exemplu

nivel/niv_min

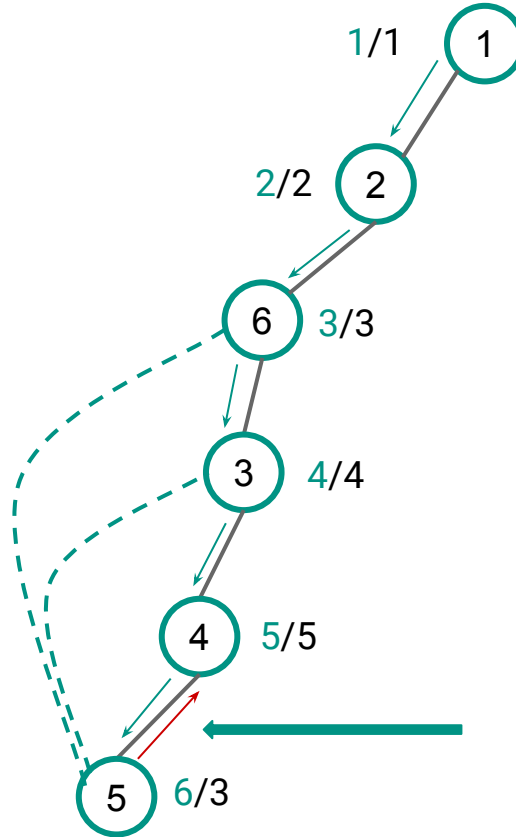
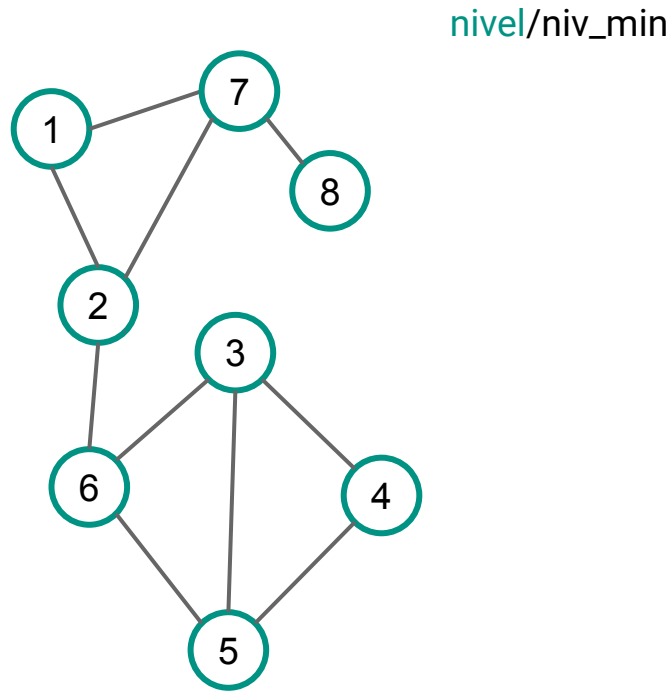


Muchii critice – Exemplu

nivel/niv_min

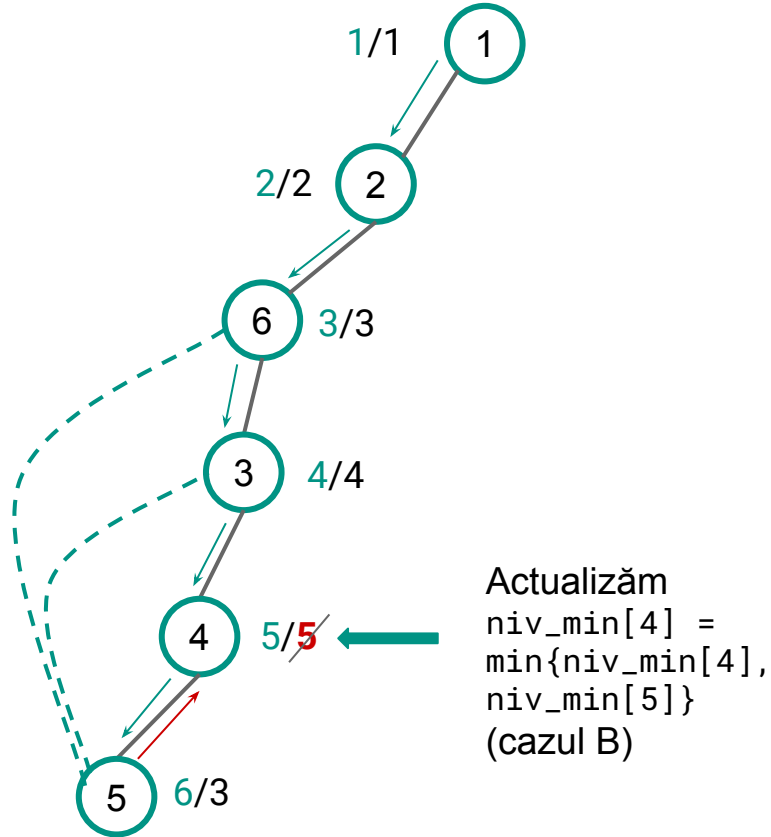
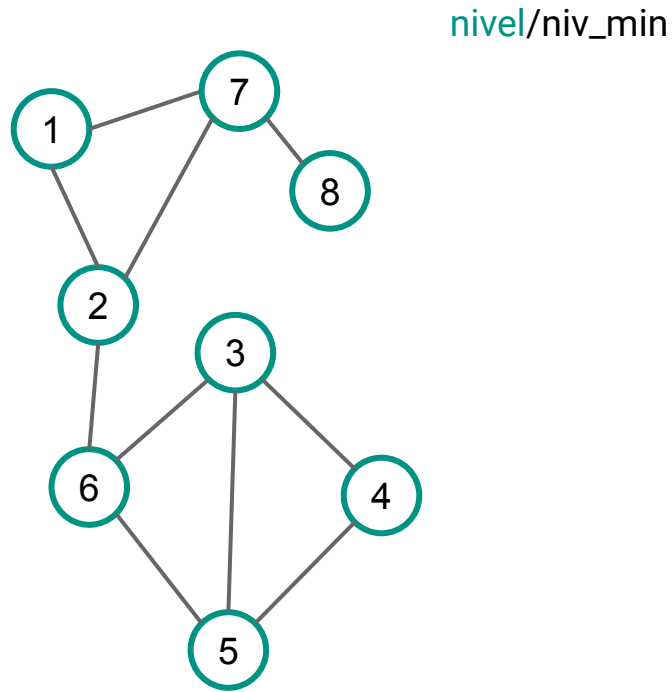


Muchii critice – Exemplu



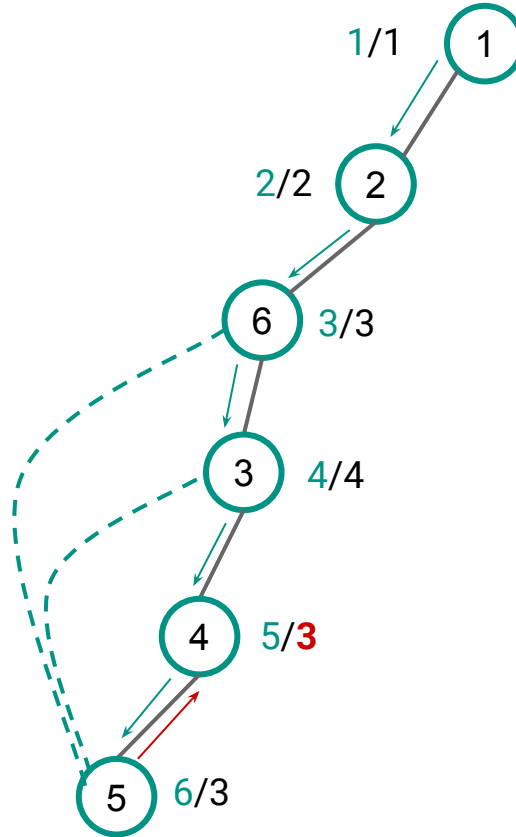
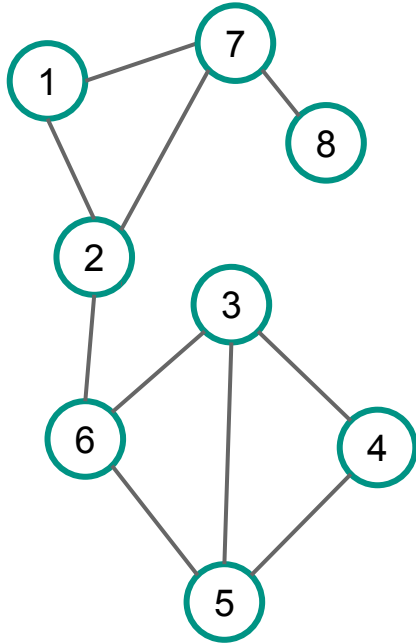
Test muchie critică:
 $\text{niv_min}[5] = 3 < \text{nivel}[4] = 5$
 $\Rightarrow \text{NU}$

Muchii critice – Exemplu

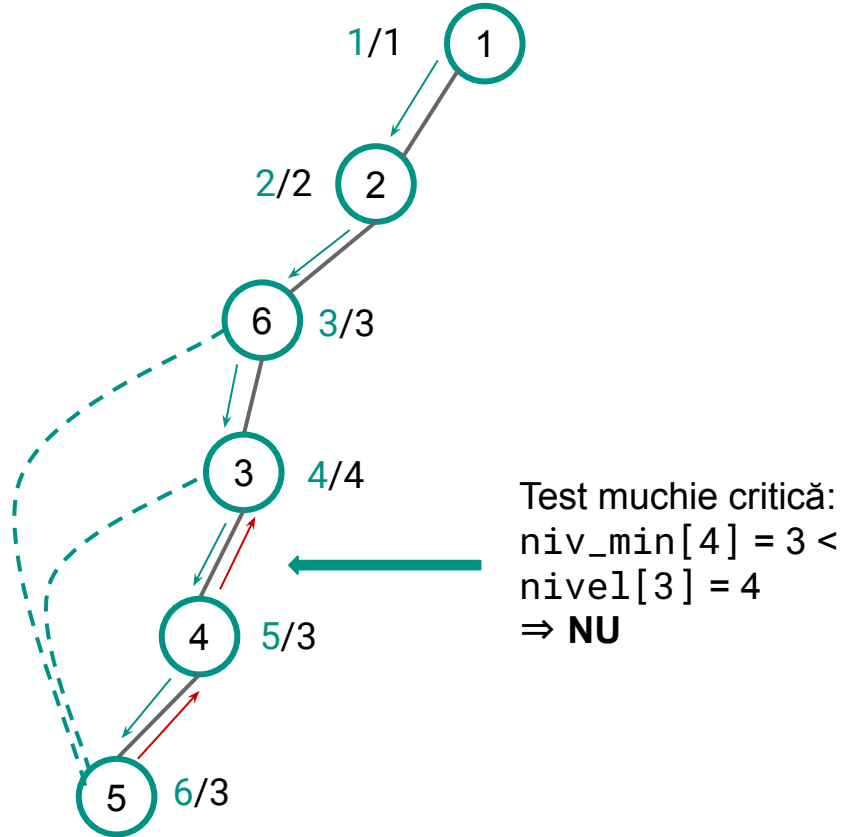
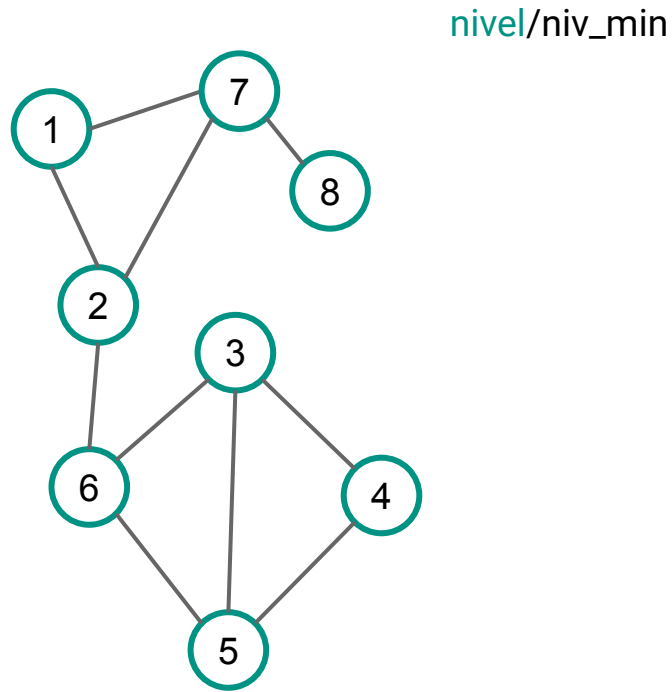


Muchii critice – Exemplu

nivel/niv_min

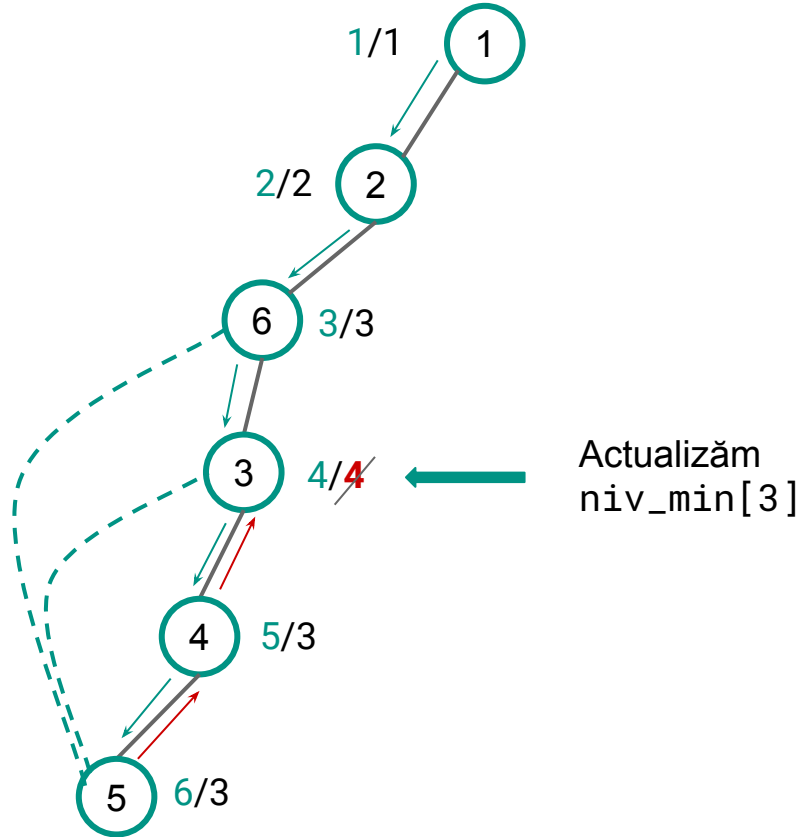
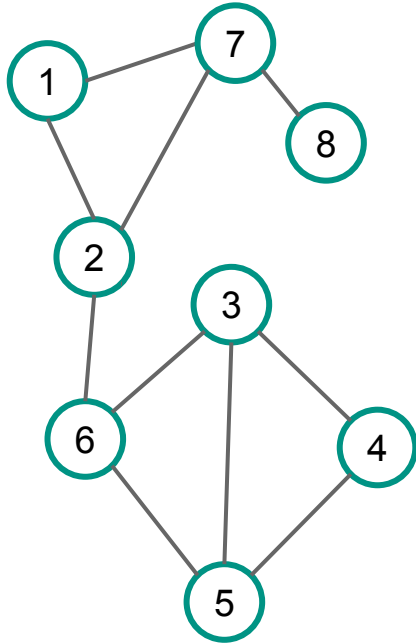


Muchii critice – Exemplu



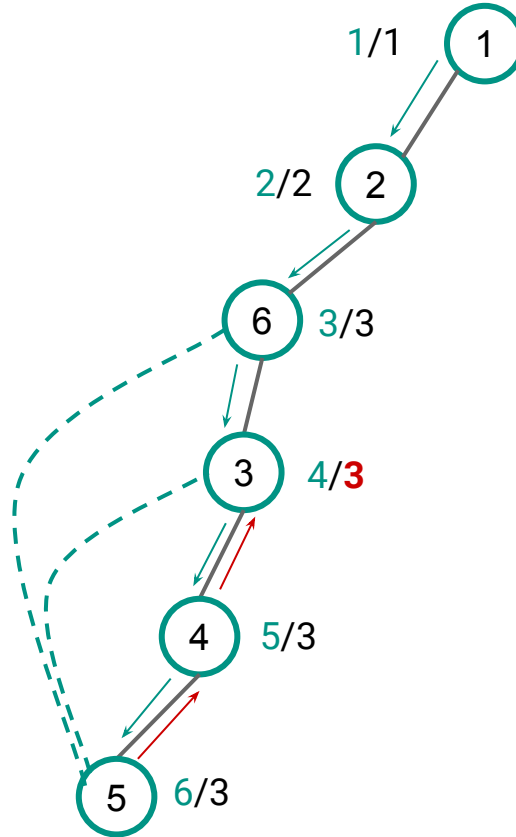
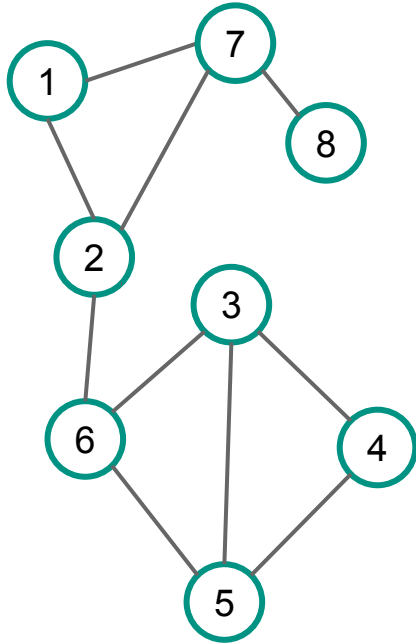
Muchii critice – Exemplu

nivel/niv_min

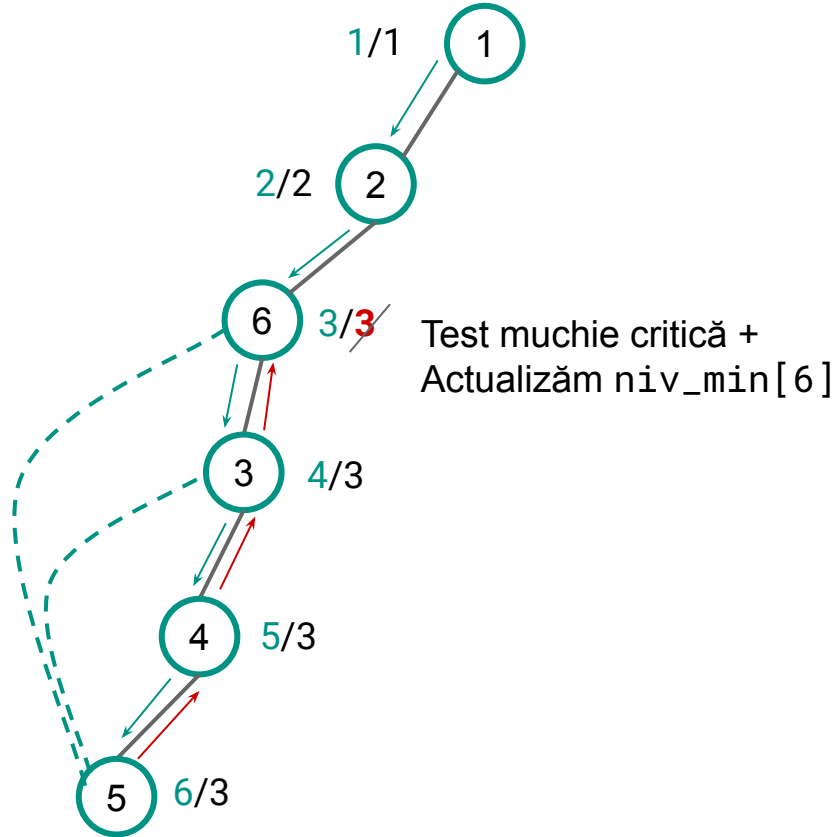
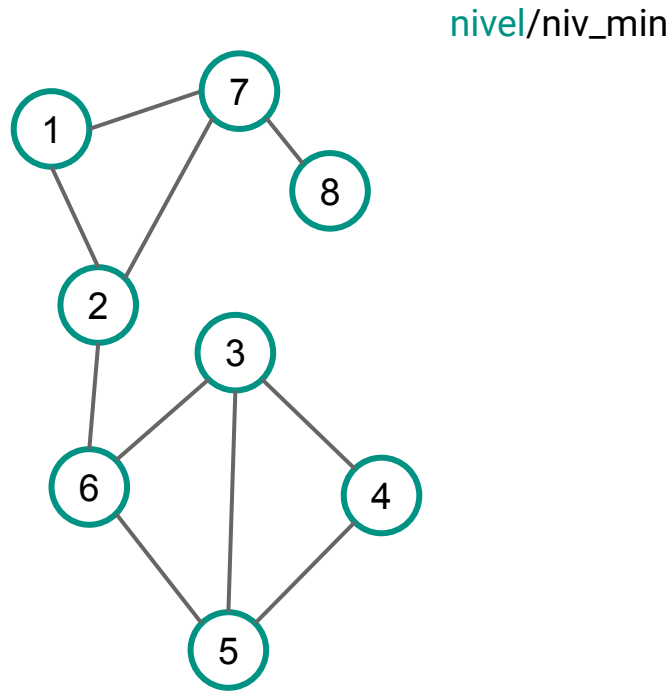


Muchii critice – Exemplu

nivel/niv_min

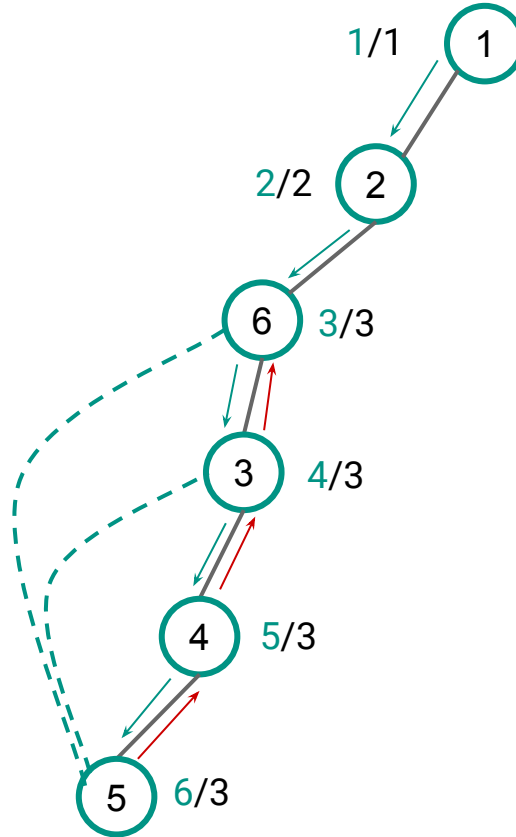
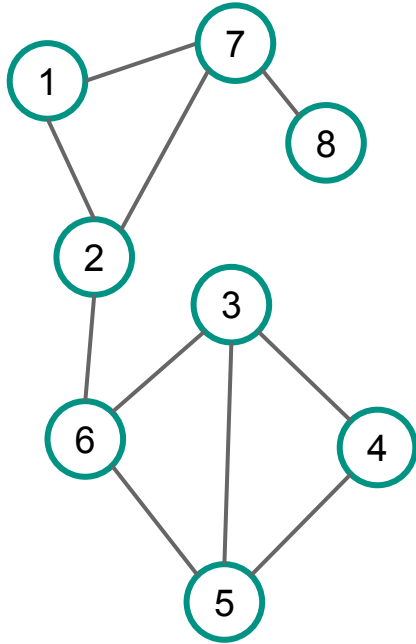


Muchii critice – Exemplu

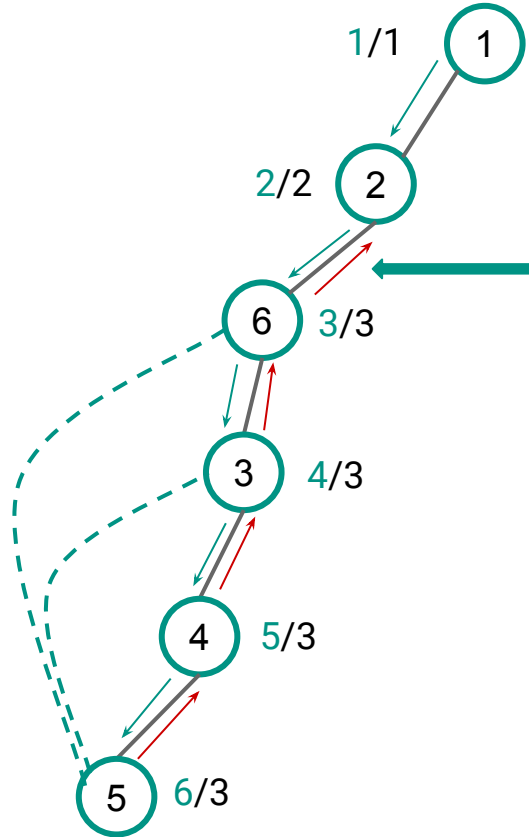
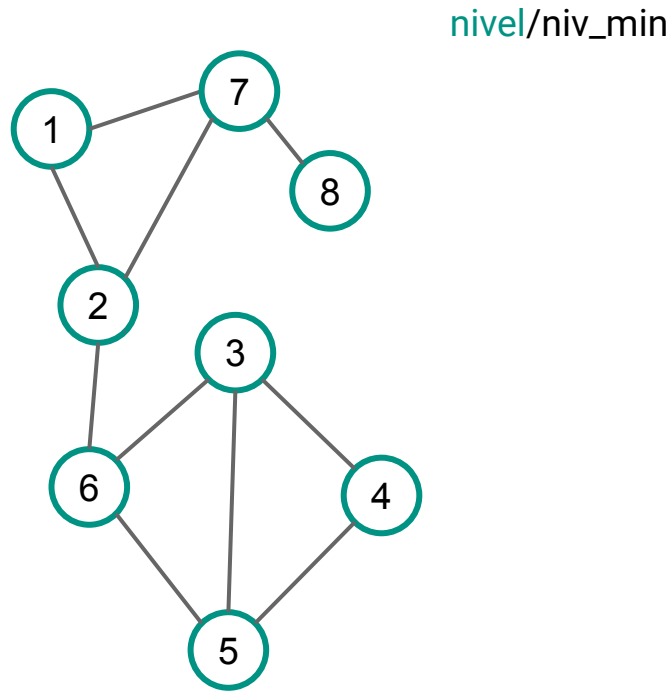


Muchii critice – Exemplu

nivel/niv_min



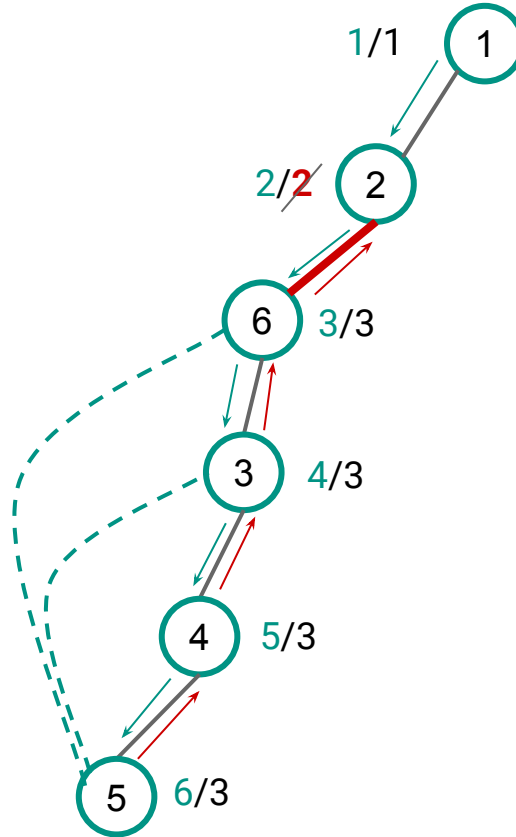
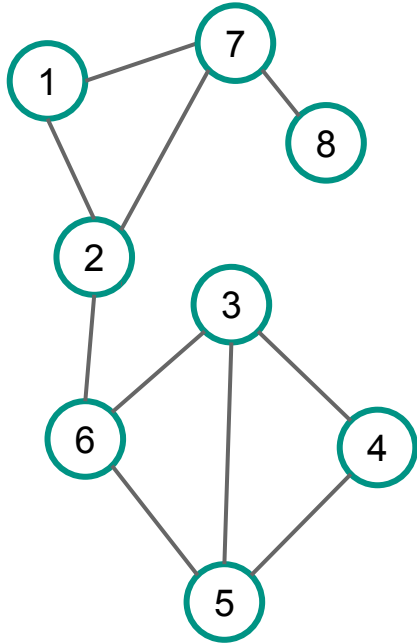
Muchii critice – Exemplu



Test muchie critică:
 $\text{niv_min}[6] = 3 >$
 $\text{nivel}[2] = 2$
 $\Rightarrow \mathbf{DA}$

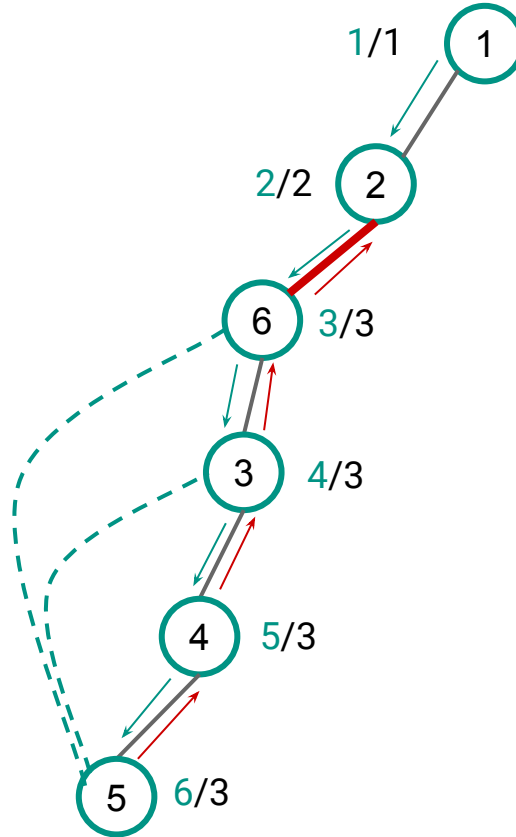
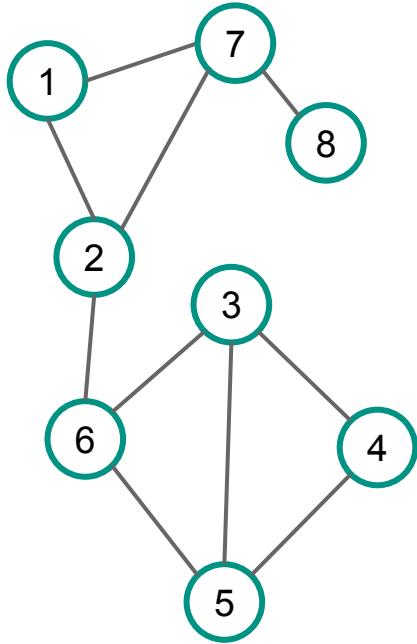
Muchii critice – Exemplu

nivel/niv_min



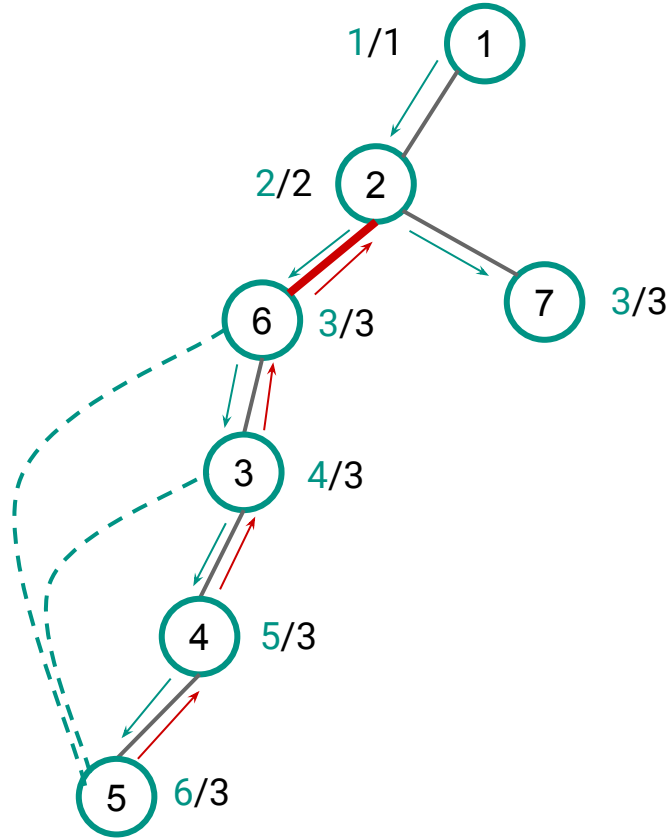
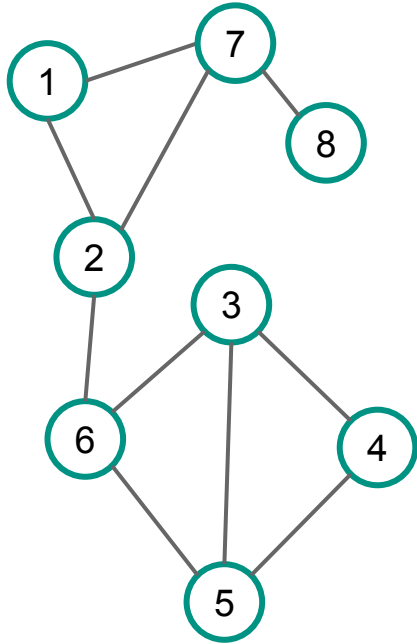
Muchii critice – Exemplu

nivel/niv_min



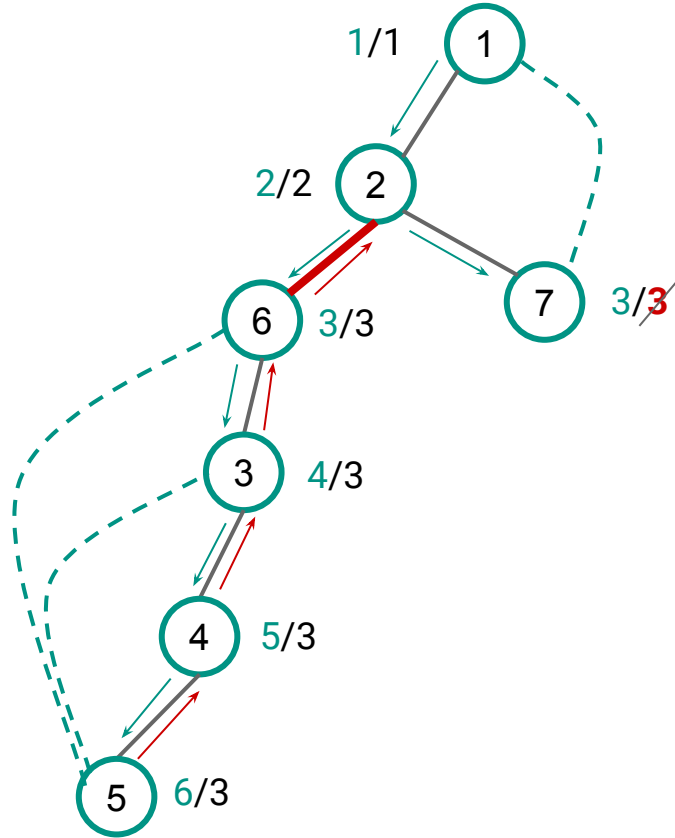
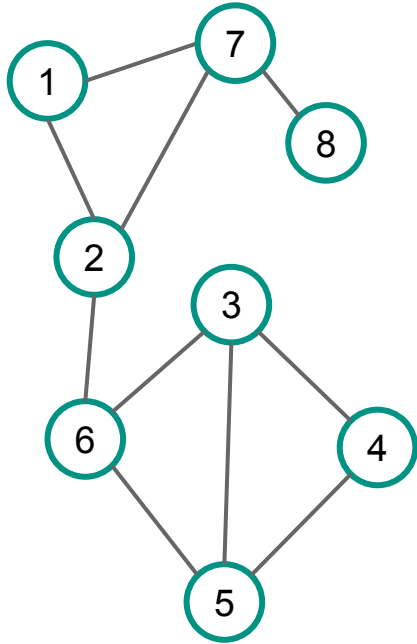
Muchii critice – Exemplu

nivel/niv_min



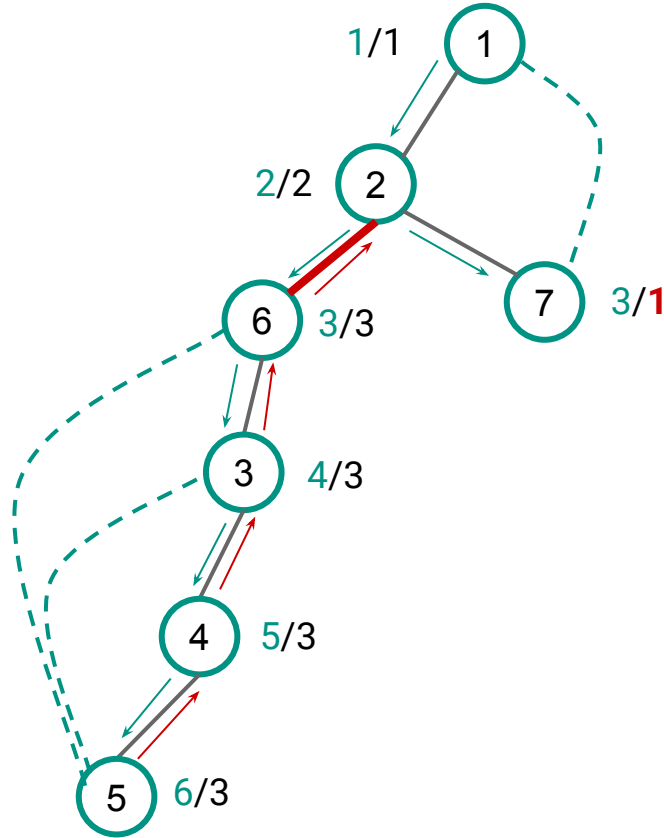
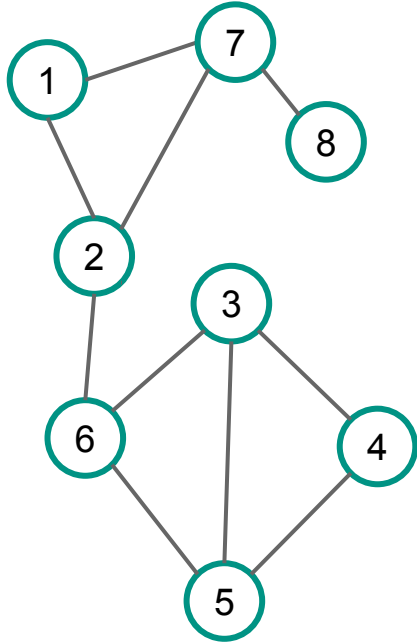
Muchii critice – Exemplu

nivel/niv_min



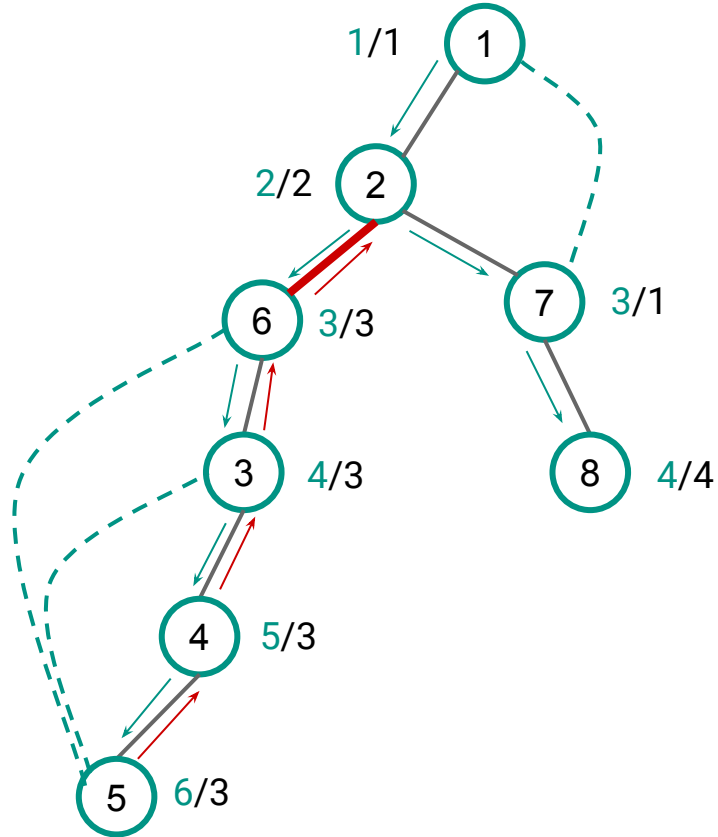
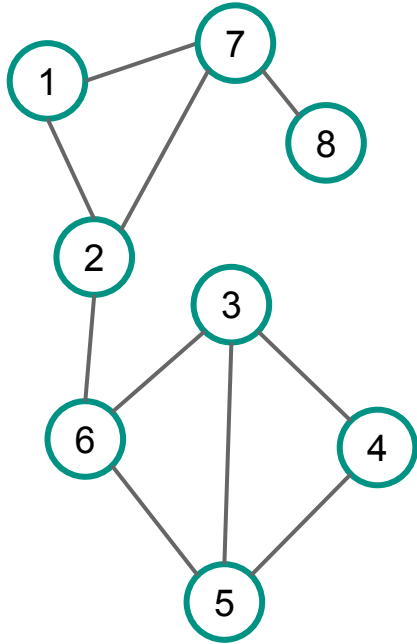
Muchii critice – Exemplu

nivel/niv_min

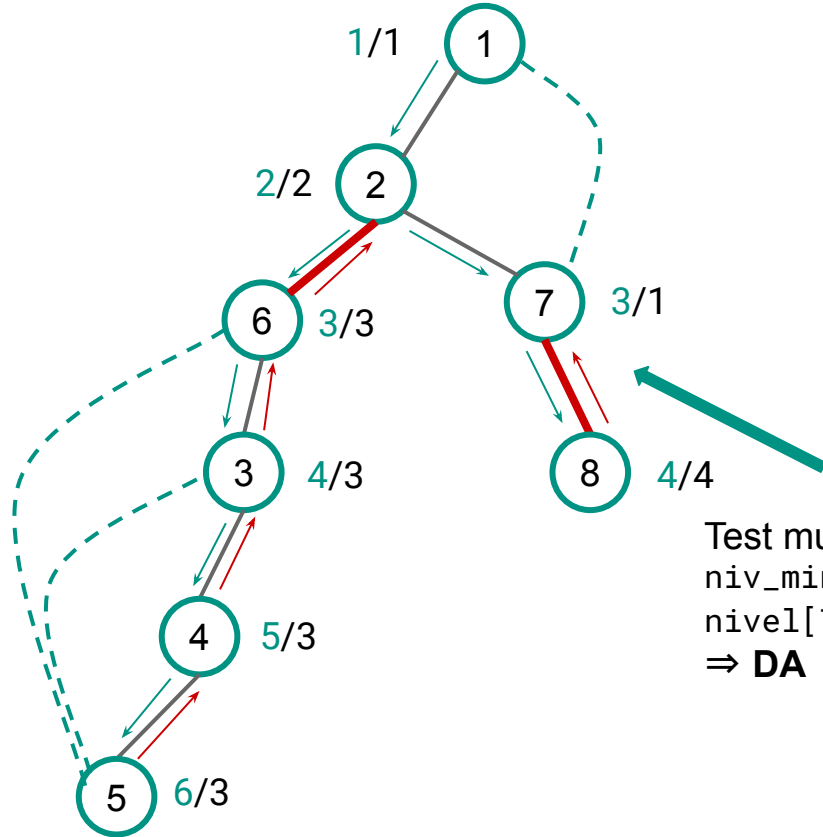
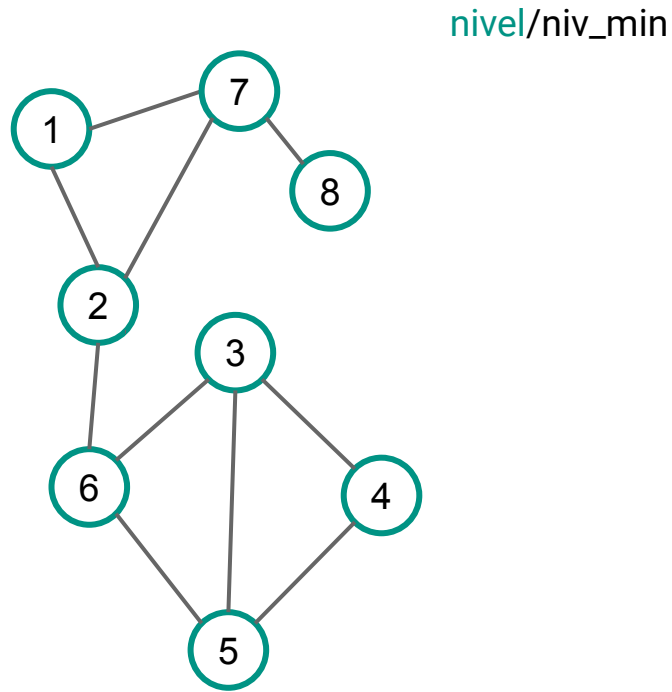


Muchii critice – Exemplu

nivel/niv_min

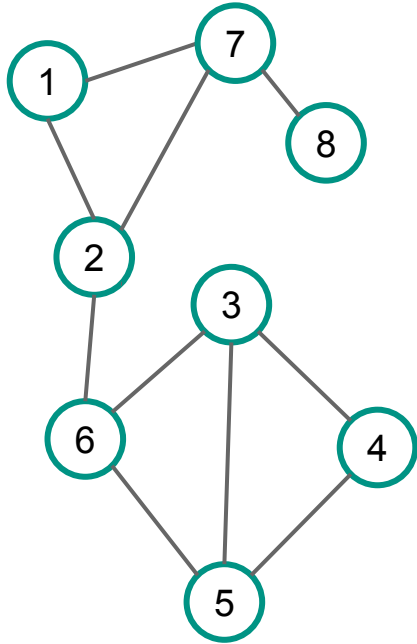


Muchii critice – Exemplu

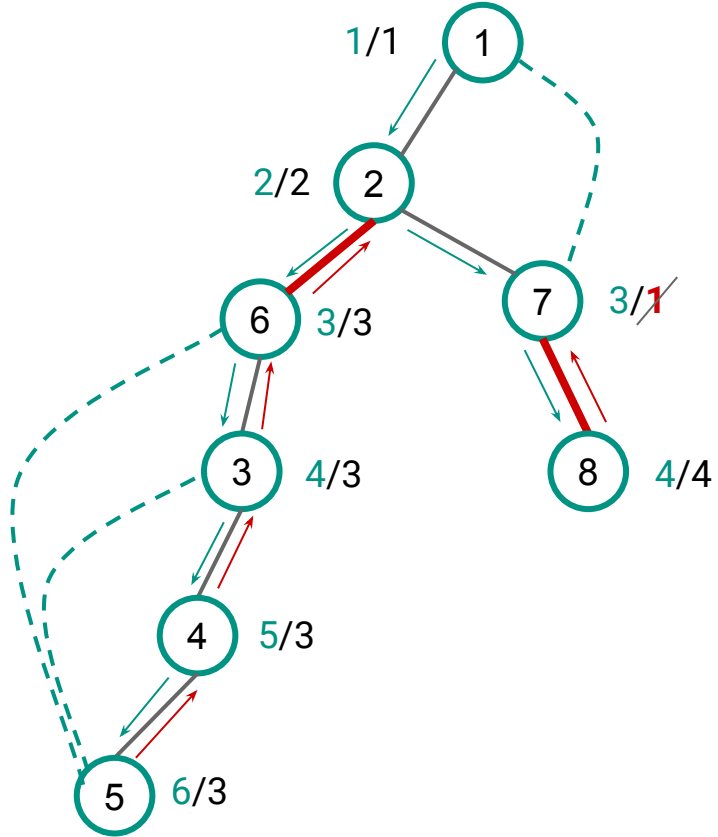


Test muchie critică:
 $niv_min[8] = 4 >$
 $nivel[7] = 3$
 \Rightarrow **DA**

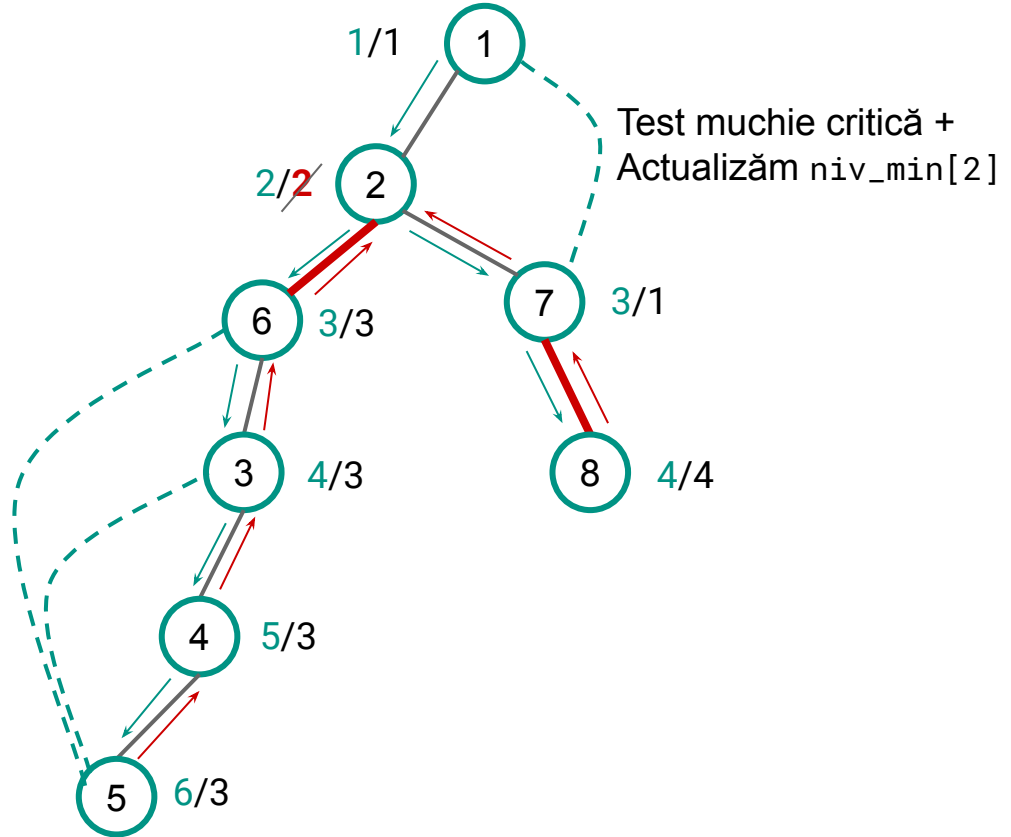
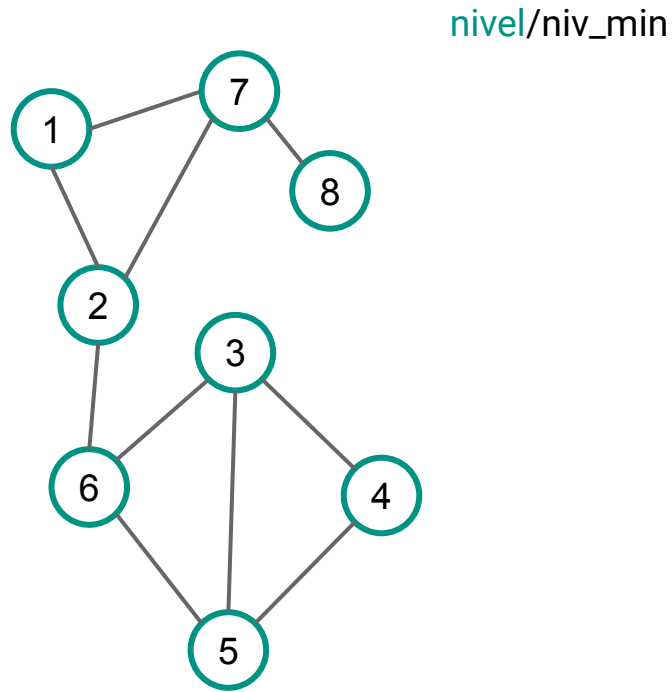
Muchii critice – Exemplu



nivel/niv_min

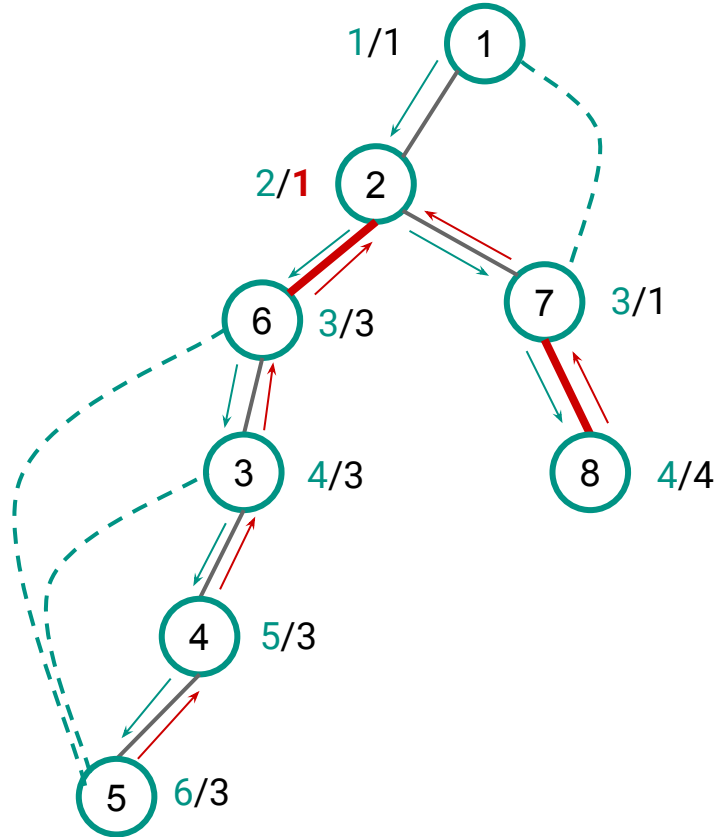
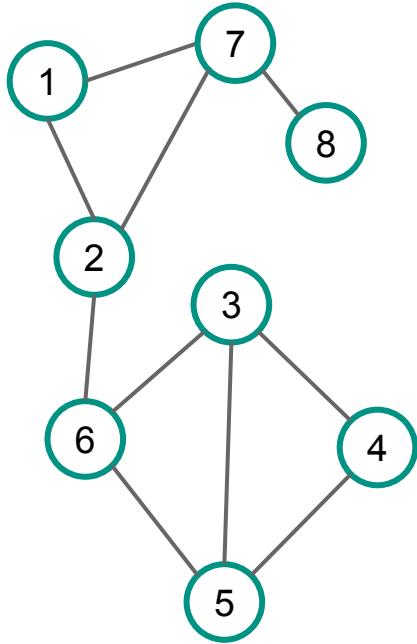


Muchii critice – Exemplu

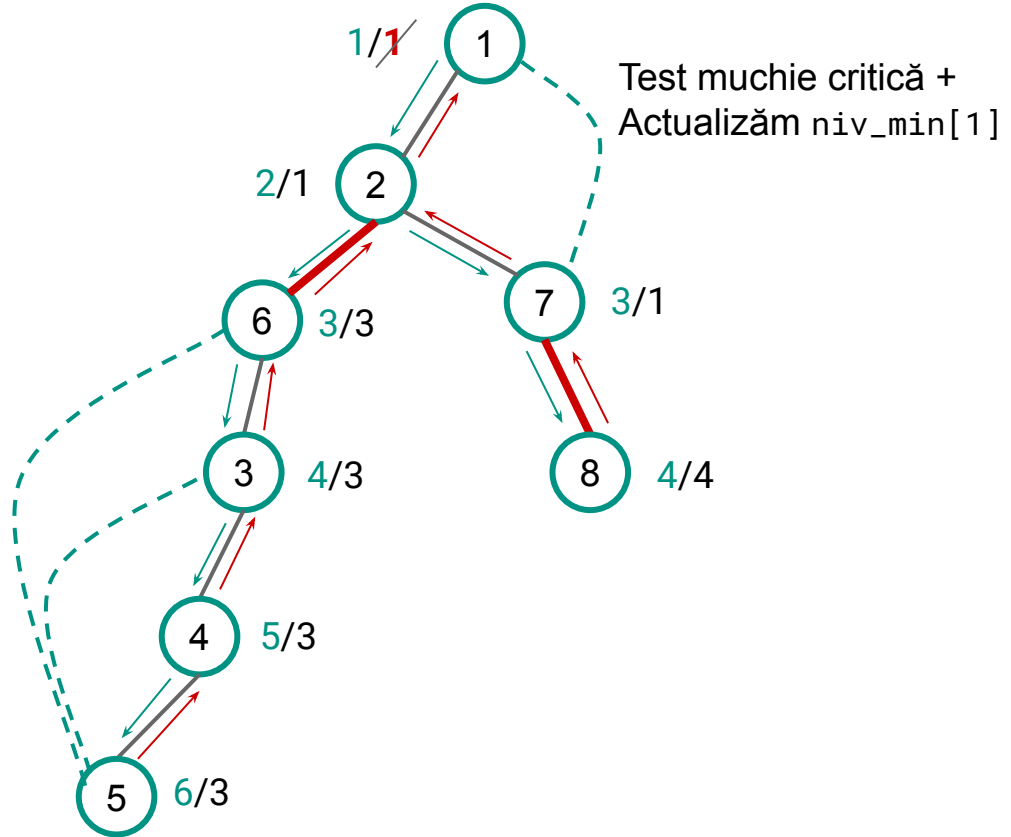
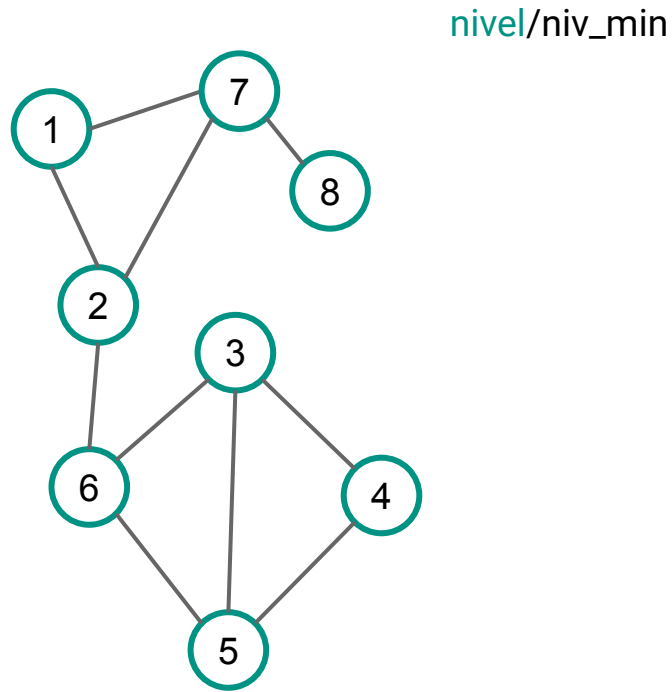


Muchii critice – Exemplu

nivel/niv_min

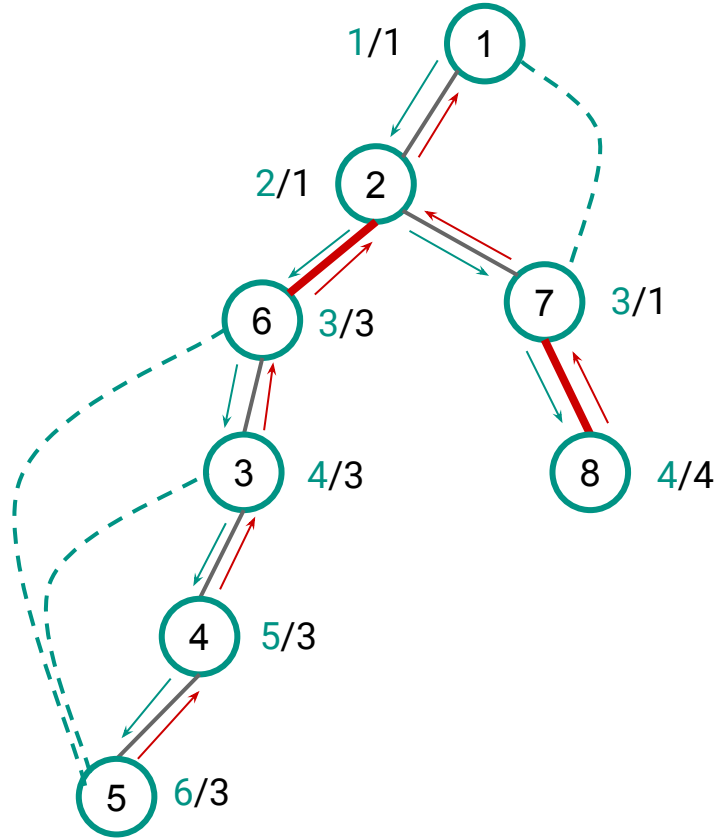
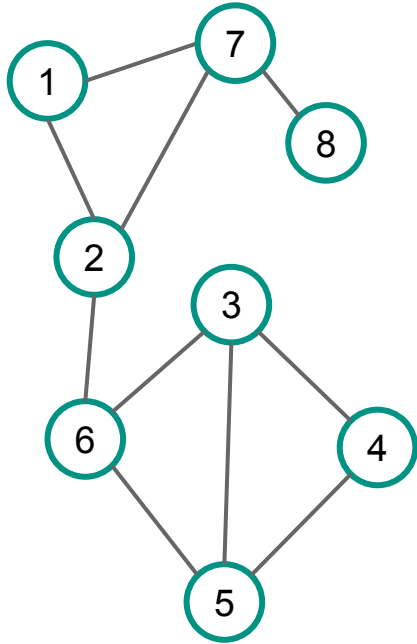


Muchii critice – Exemplu



Muchii critice – Exemplu

nivel/niv_min



Indicații de implementare

```
void df(int i) {
    viz[i] = 1;
    niv_min[i] = nivel[i];
    for (j vecin al lui i)
        if (viz[j] == 0) {           //ij muchie de avansare
            nivel[j] = nivel[i] + 1;
            df(j);
            //actualizare niv_min[i] - formula B
            niv_min[i] = min{ niv_min[i], niv_min[j] }
            //test ij este muchie critica
            ...
        }
    else
        if(nivel[j] < nivel[i] - 1)   //ij muchie de intoarcere
            //actualizare niv_min[i] - formula A
            ...
}
```

Indicații de implementare

```
void df(int i) {
    viz[i] = 1;
    niv_min[i] = nivel[i];
    for (j vecin al lui i)
        if (viz[j] == 0) { //ij muchie de avansare
            nivel[j] = nivel[i] + 1;
            df(j);
            //actualizare niv_min[i] - formula B
            niv_min[i] = min{ niv_min[i], niv_min[j] }
            //test ij este muchie critica
            if (niv_min[j] > nivel[i]) scrie muchia ij
        }
    else
        if(nivel[j] < nivel[i] - 1) //ij muchie de intoarcere
            //actualizare niv_min[i] - formula A
            niv_min[i] = min{ niv_min[i], niv[j] }
}
```

Puncte critice



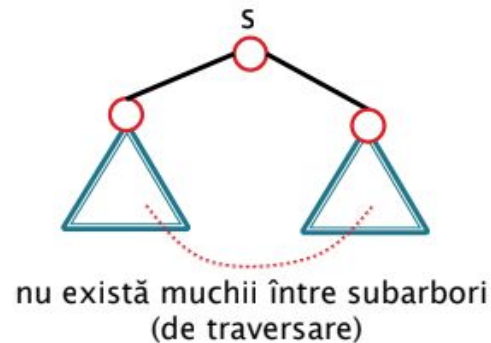
Puncte critice

Un vârf este **punct critic** \Leftrightarrow există două vârfuri $x, y \neq v$ astfel încât aparține oricărui x, y -lanț.

Arborele DF

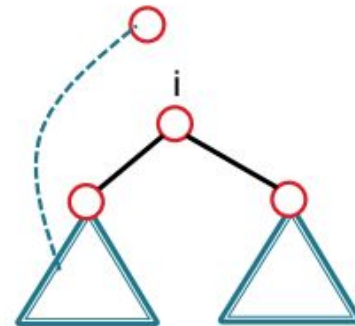
☐ **rădăcina** este punct critic

\Leftrightarrow



☐ **un alt vârf i** din arbore este critic

\Leftrightarrow



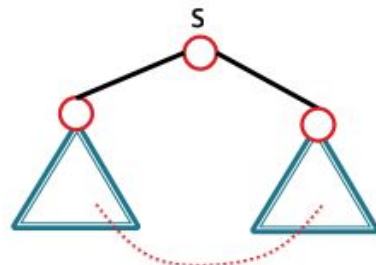
Puncte critice

Un vârf este **punct critic** \Leftrightarrow există două vârfuri $x, y \neq v$ astfel încât aparține oricărui x, y -lanț.

Arborele DF

- **rădăcina** este punct critic \Leftrightarrow

are cel puțin 2 fii în arborele DF



nu există muchii între subarbori
(de traversare)

- **un alt vârf i** din arbore este critic \Leftrightarrow

are cel puțin un fiu j cu $\text{niv_min}[j] \geq \text{nivel}[i]$

