Structuri pentru mulțimi disjuncte

Operații cu mulțimi disjuncte

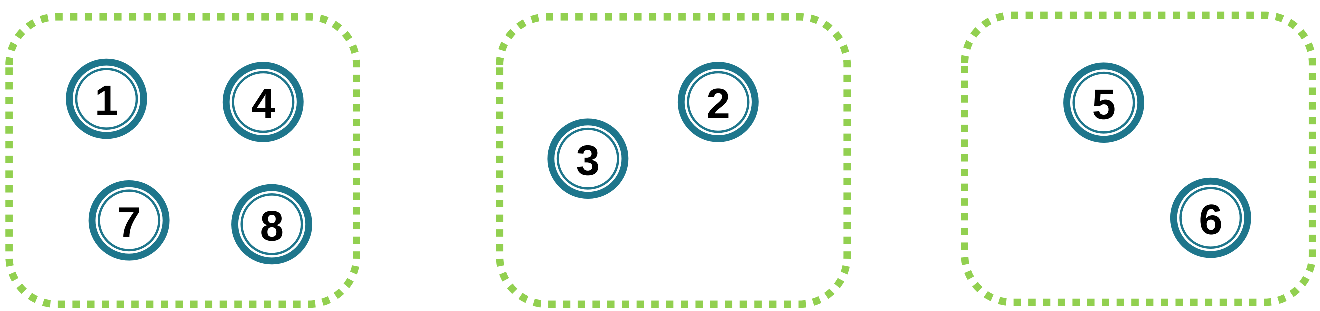
Problemă

Asupra unei partiții ale mulțimii {1,2,...,n} (în submulțimi disjuncte) se efectuează o succesiune de operații de tip

- reuniune
- test de apartenență

Cum putem memora submulțimile astfel încât operațiile să se efectueze "eficient"?

{1, 4, 7, 8} {2, 3} {5, 6}



Operații cu mulțimi disjuncte

Soluții

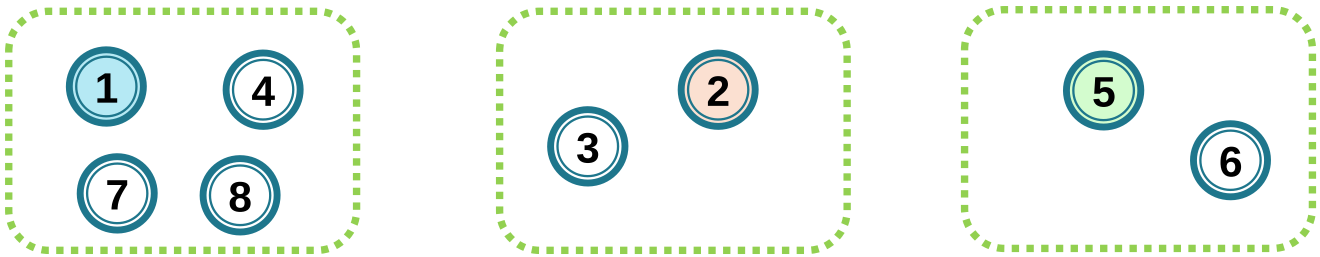
Asociem fiecărei submulțimi un reprezentant (culoare).

Notăm operațiile

- **Initializare(u)** – creează o mulțime cu un singur element u
- **Reprez(u)** – returnează reprezentantul mulțimii care conține pe u
- **Reuneste(u,v)** – unește mulțimea care conține u cu cea care conține v

Vectori de reprezentanți

Varianta 1 – Memorăm într-un vector r pentru fiecare element x reprezentantul mulțimii r[x] – v. Kruskal curs



	1	2	3	4	5	6	7	8
r:	1	2	2	1	5	5	1	1

- **Initializare(u)** – O(1)
- **Reprez(u)** – O(1)
- **Reuneste(u,v)** – O(n)

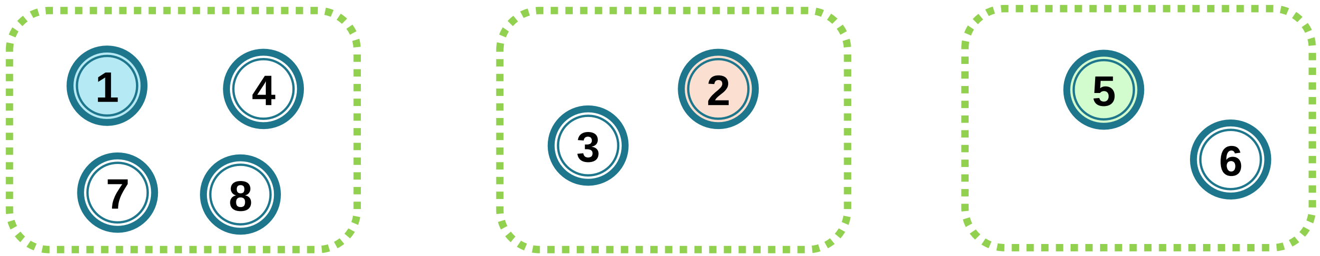
```
void Initializare(int u){ r[u]=u;}

int Reprez(int u){ return r[u]; }

void Reuneste(int u,int v){
  r1 = Reprez(u);//r1=r[u]
  r2 = Reprez(v);//r2=r[v]
  for(k=1;k<=n;k++)
    if(r[k]==r2)
      r[k] = r1;        }
```

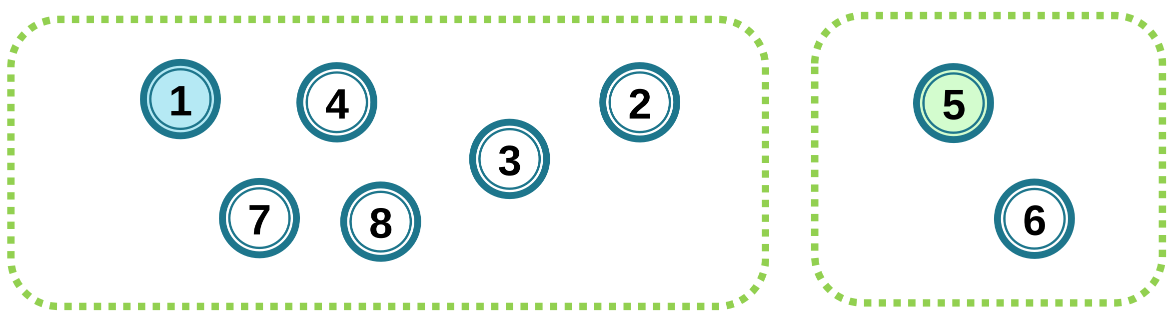
Vectori de reprezentanți

Exemplu



	1	2	3	4	5	6	7	8
r:	1	2	2	1	5	5	1	1

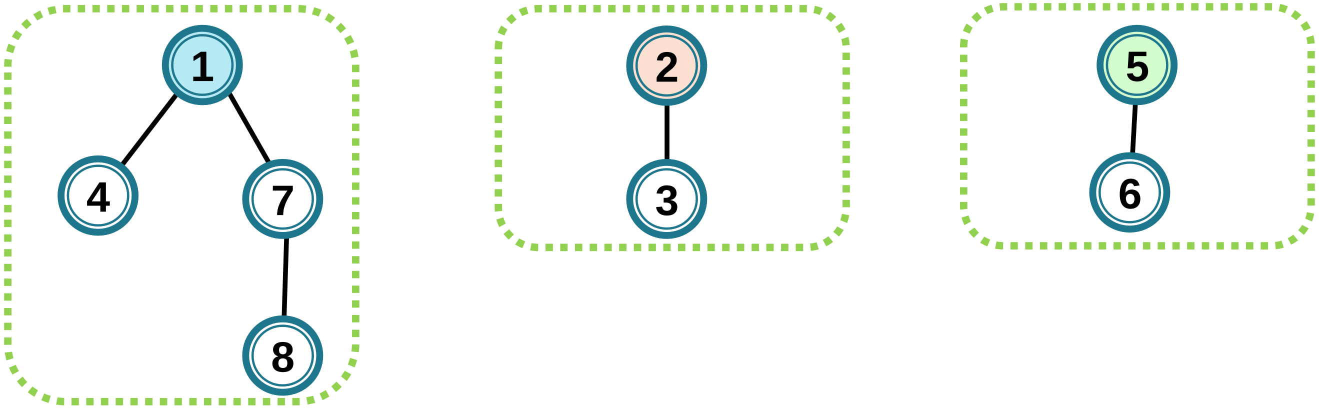
Reuneste(4, 3) $\Rightarrow$



	1	2	3	4	5	6	7	8
r:	1	1	1	1	5	5	1	1

Operații cu mulțimi disjuncte

Varianta 2 – Memorăm vârfurile fiecărei mulțimi ca un arbore (memorat cu tata), având ca **reprezentant rădăcina**



	1	2	3	4	5	6	7	8
tata:	0	0	2	1	0	5	1	7

Păduri de mulțimi disjuncte

Varianta 2 – Memorăm vârfurile fiecărei mulțimi ca un arbore (memorat cu tata), având ca **reprezentant rădăcina**

- **Initializare(u)** : O(1)

```
void Initializare(int u){ tata[u]=h[u]=0;}
```

- **Reprez(u)** – determinarea rădăcinii arborelui care conține u
  – **liniar în înălțimea arborelui**

```
int Reprez(int u){
    while(tata[u]!=0)
        u=tata[u];
    return u;       }
```

Păduri de mulțimi disjuncte

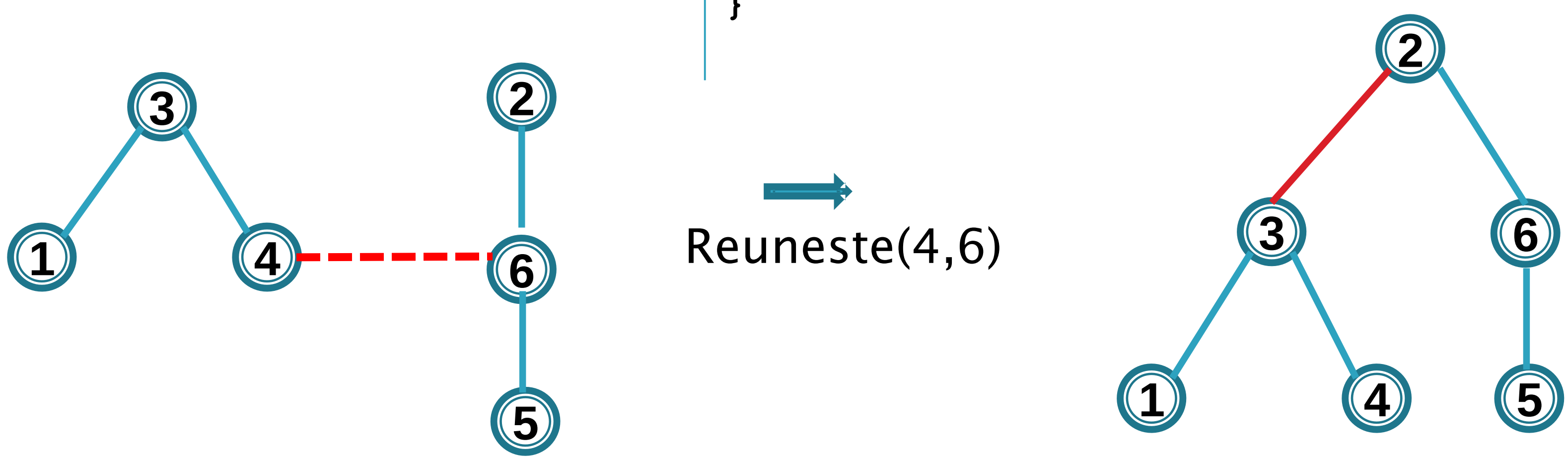
Varianta 2 – Memorăm vârfurile fiecărei mulțimi ca un arbore (memorat cu tata), având ca **reprezentant rădăcina**

- **Reuneste(u,v)** – reuniune ponderată **în funcţie de înălţimea arborilor**

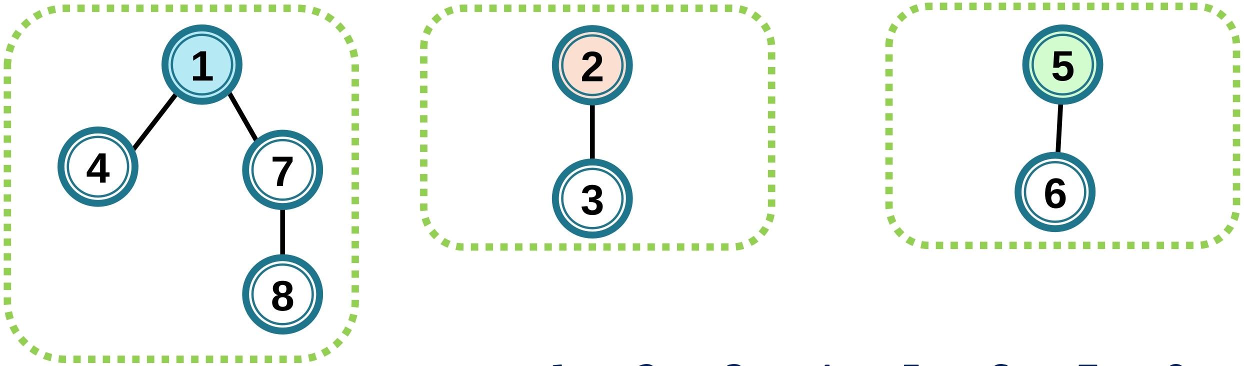
  –arborele cu înălţimea mai mică devine subarbore al rădăcinii celuilalt arbore

  - O(1) după determinarea reprezentanților lui u și v
  - **arbori de înălțime logaritmică**

```
void Reuneste(int u,int v){
    int ru=Reprez(u); int rv=Reprez(v);
    if (h[ru]>h[rv])
            tata[rv]=ru;
    else{ tata[ru]=rv;
            if(h[ru]==h[rv])
                  h[rv]=h[rv]+1;
         }
}
```

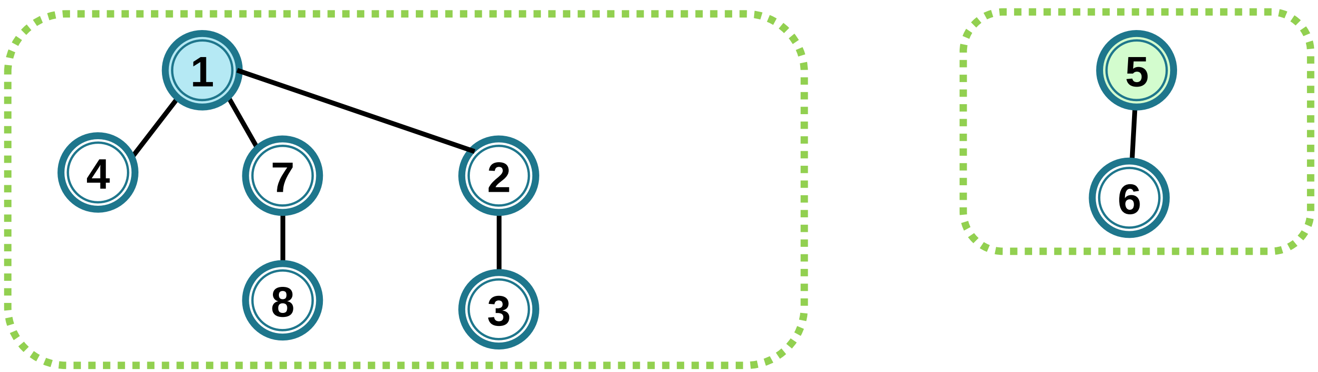


Păduri de mulțimi disjuncte



	1	2	3	4	5	6	7	8
tata:	0	0	2	1	0	5	1	7
h:	2	1	0	0	1	0	1	0

Reuneste(4, 3) $\Rightarrow$ deoarece h[1] > h[2], se va seta tata[2] = 1
(h nu se modifică)



	1	2	3	4	5	6	7	8
tata:	0	1	2	1	0	5	1	7

Păduri de mulțimi disjuncte

Reprez(u) Optimizare – compresie de cale

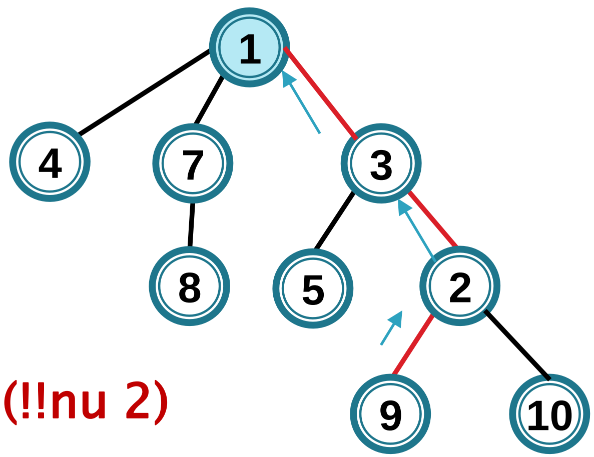
– tatăl vârfurilor de pe lanțul de la u la rădăcină se va seta ca fiind rădăcina

(vârfurile de pe acest lanț, parcurs pentru a găsi reprezentantul lui u, vor deveni fii ai rădăcinii, pentru ca reprezentantul lor să fie găsit mai ușor în căutările ulterioare)

```
int Reprez(int u){
   if (tata[u]==0)
         return u;
   tata[u]=Reprez(tata[u]);
   return tata[u];
}
```

!! h nu se actualizează

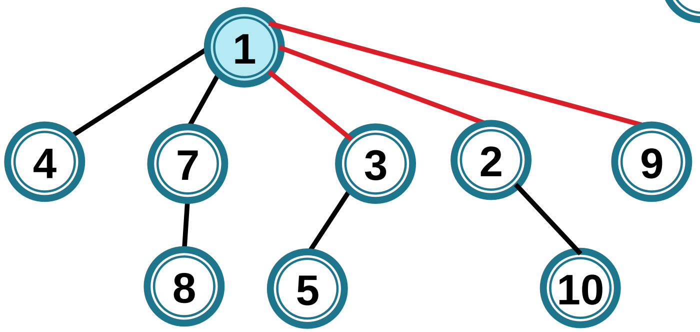
h[1]=3



De exemplu, după apelul Reprez(9) pentru arborele

h[1]=3 (!!nu 2)

rezultatul va fi 1, iar arborele devine

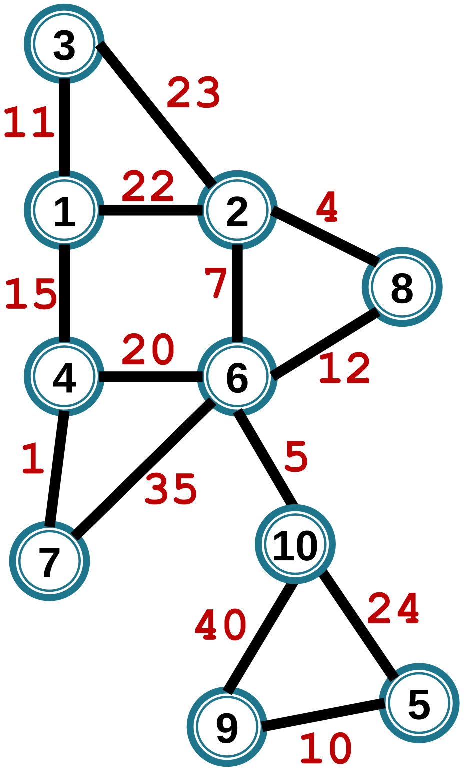


Algoritmul lui Kruskal

Implementare cu păduri disjuncte

Kruskal – Pseudocod

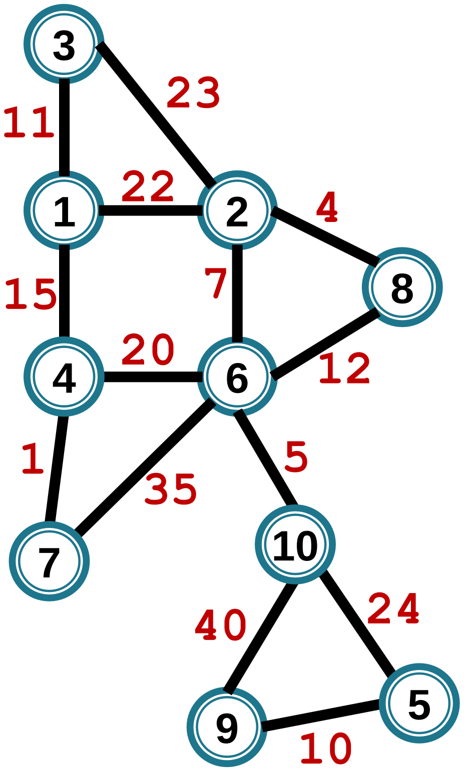
```
sorteaza(E)
for(v=1;v<=n;v++)
     Initializare(v);
nrmsel=0
for(uv ∈ E)
    if(Reprez(u)!=Reprez(v))
    {
           E(T) = E(T) ∪ {uv};
           Reuneste(u,v);
           nrmsel=nrmsel+1;
           if(nrmsel==n-1)
                STOP; //break;
    }
```



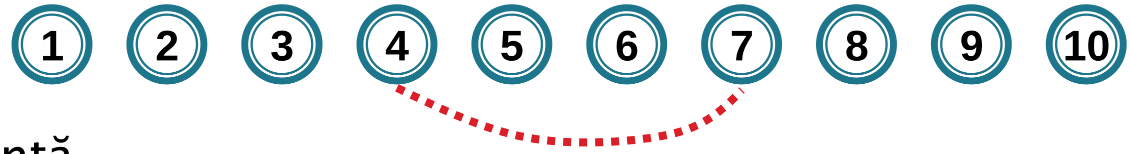
Ordine muchii
(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent

	1	2	3	4	5	6	7	8	9	10
tata	0	0	0	0	0	0	0	0	0	0
h	0	0	0	0	0	0	0	0	0	0



Pădurea de mulțimi disjuncte la pasul curent



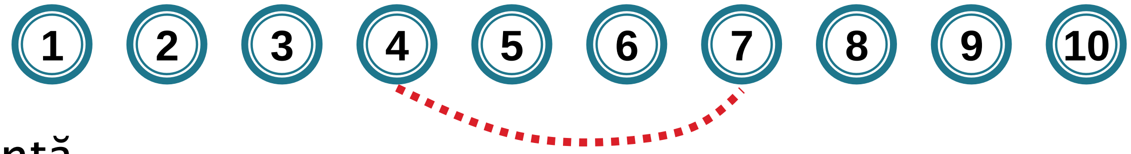
Muchia curentă

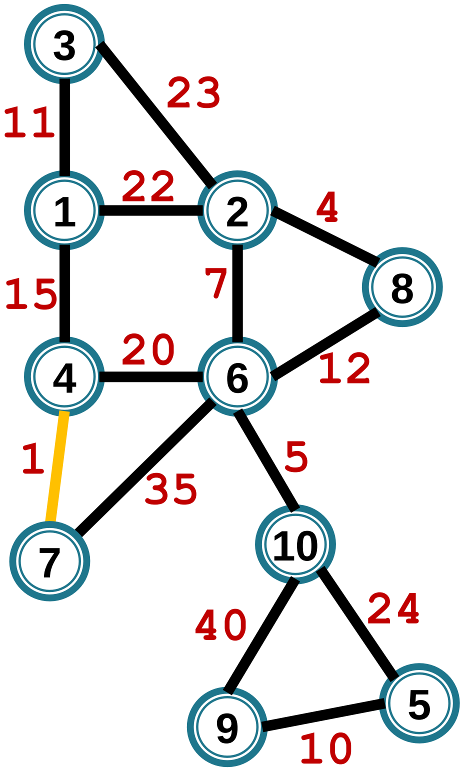
(4,7):

Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent



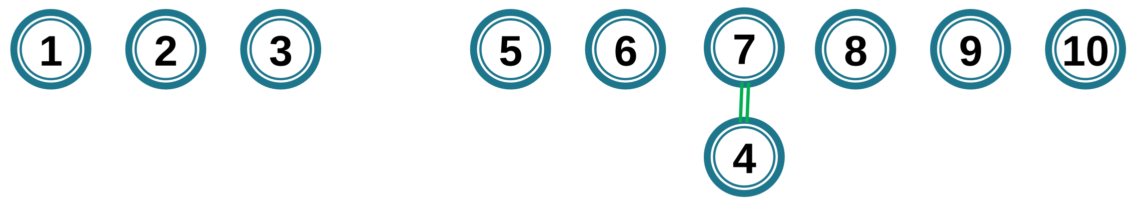


Muchia curentă

(4,7):

Reprez(4)≠ Reprez(7)

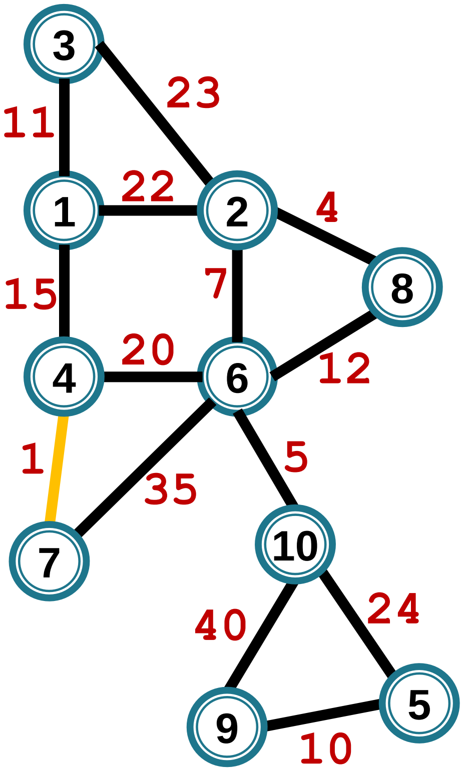
Reuneste(4,7)



Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

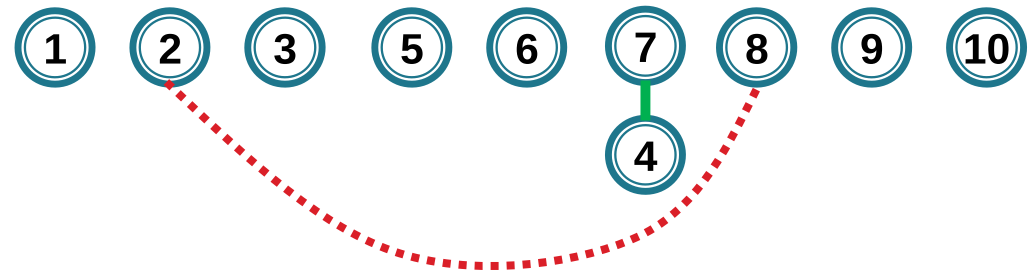
	1	2	3	4	5	6	7	8	9	10
tata	0	0	0	7	0	0	0	0	0	0
h	0	0	0	0	0	0	1	0	0	0



Ordine muchii

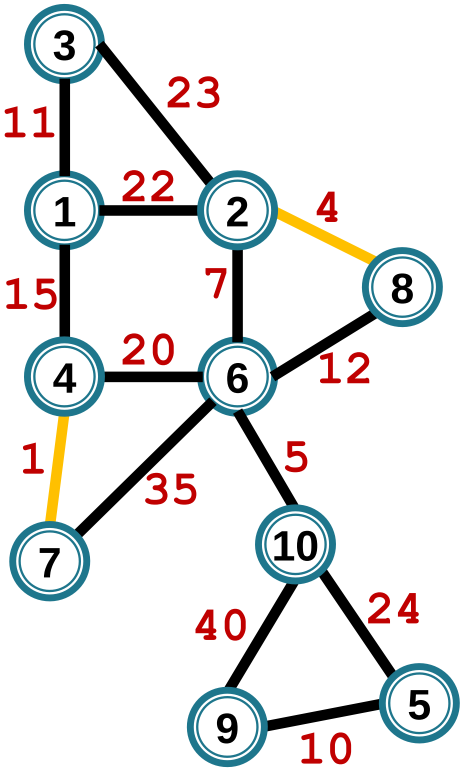
(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent



Muchia curentă

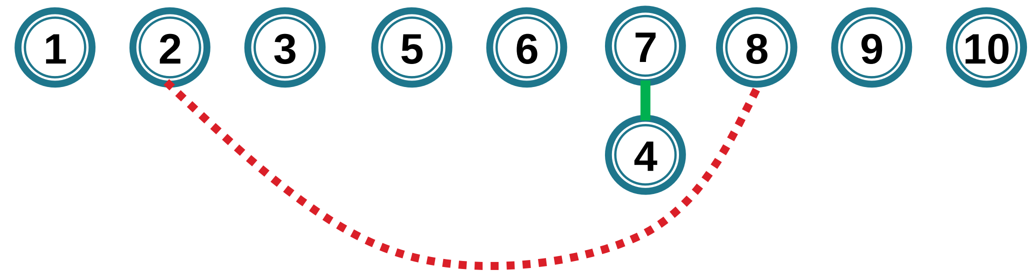
(2,8):



Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent

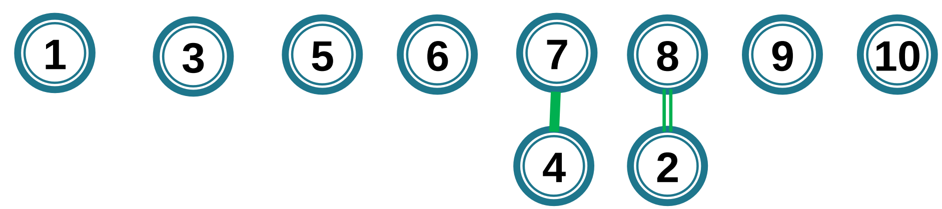


Muchia curentă

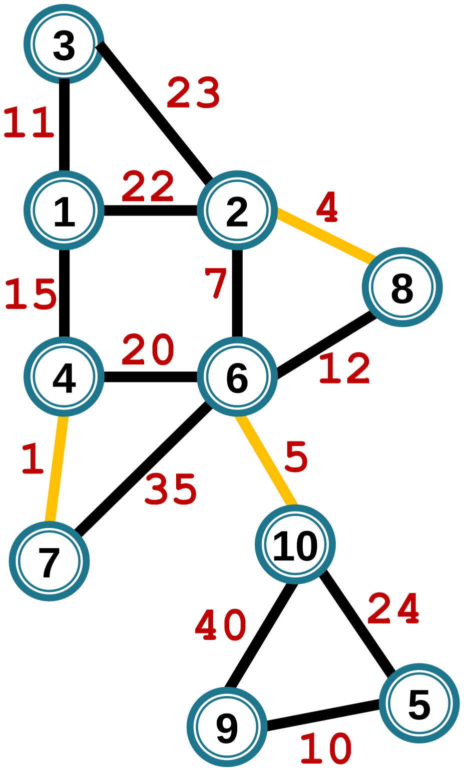
(2,8):

Reprez(2)≠ Reprez(8)

Reuneste(2,8)



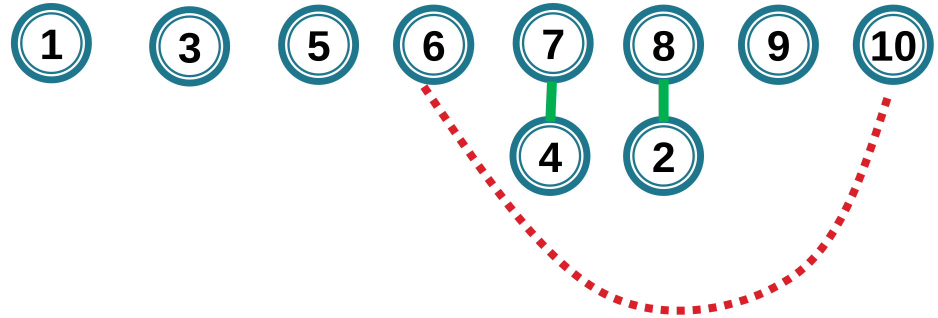
	1	2	3	4	5	6	7	8	9	10
tata	0	8	0	7	0	0	0	0	0	0
h	0	0	0	0	0	0	1	1	0	0



Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent

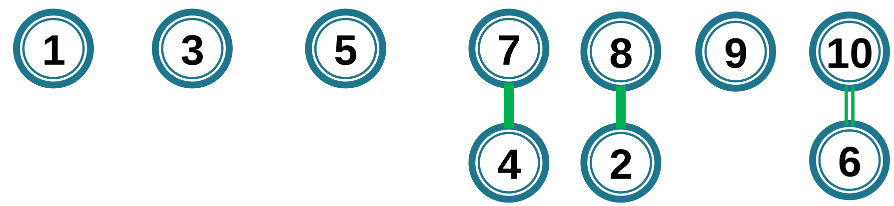


Muchia curentă

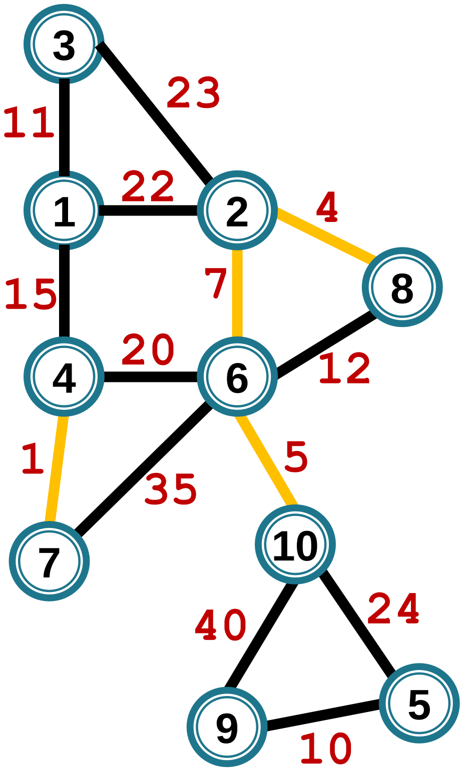
(6,10):

Reprez(6)≠ Reprez(10)

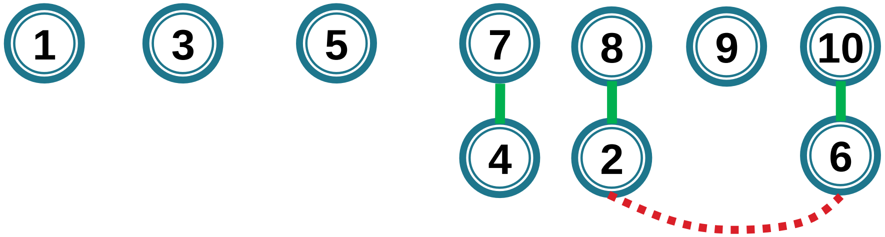
Reuneste(6,10)



	1	2	3	4	5	6	7	8	9	10
tata	0	8	0	7	0	10	0	0	0	0
h	0	0	0	0	0	0	1	1	0	1



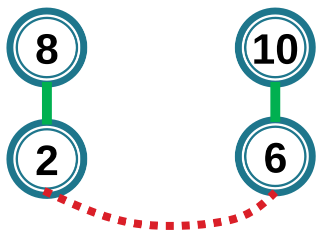
Pădurea de mulțimi disjuncte la pasul curent



Muchia curentă

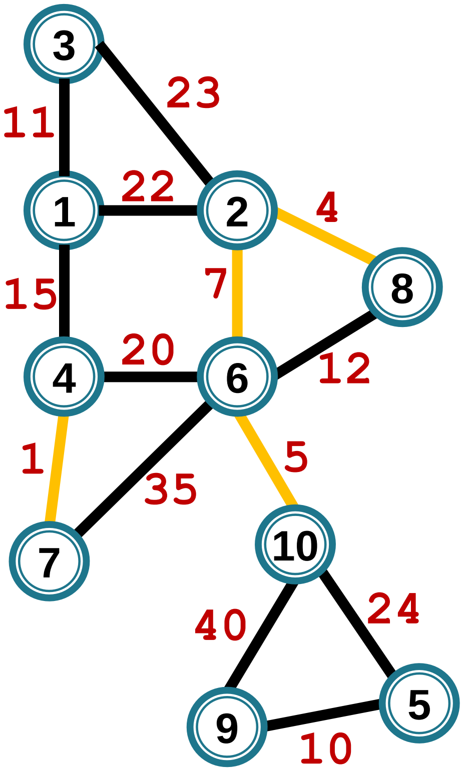
(2,6):

Reprez(2)≠ Reprez(6)

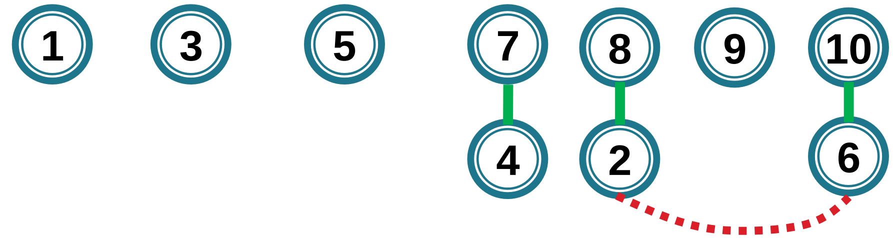


Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)



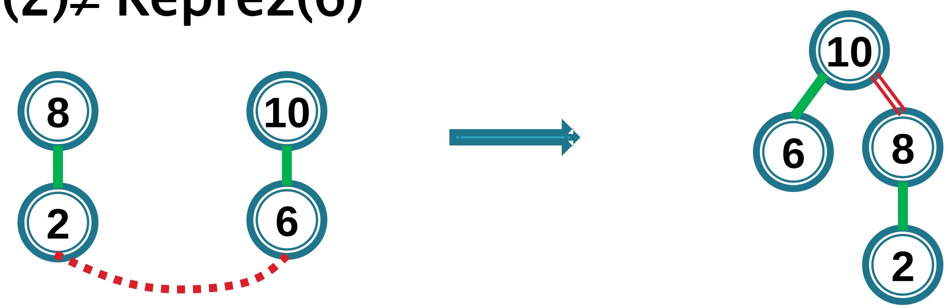
Pădurea de mulțimi disjuncte la pasul curent



Muchia curentă

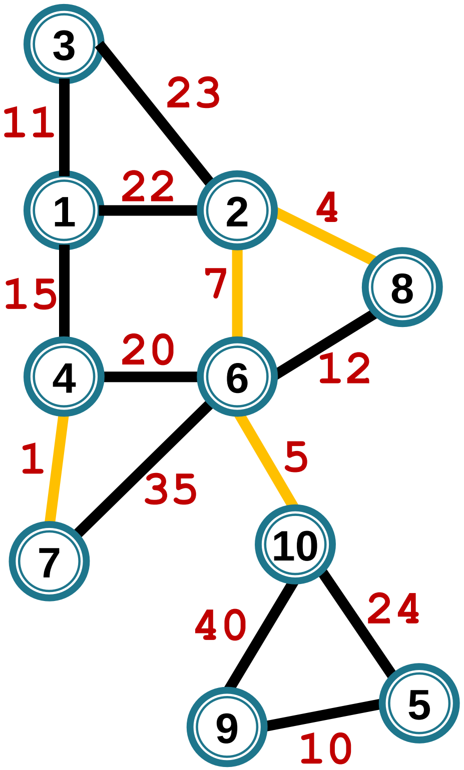
(2,6):

Reprez(2)≠ Reprez(6)



Ordine muchii

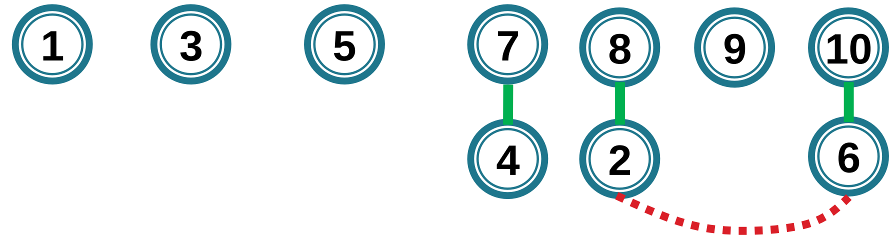
(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)



Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent

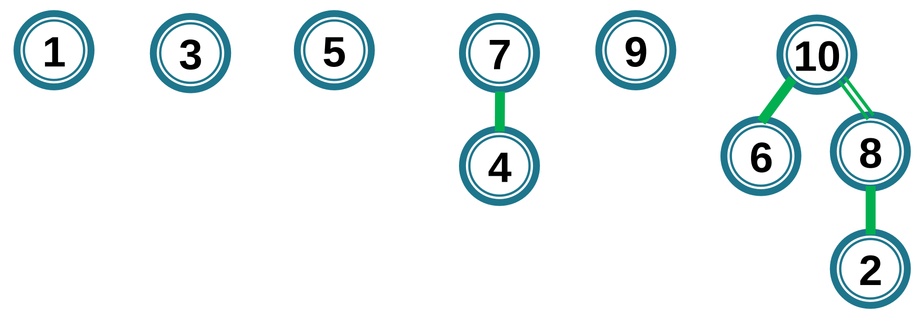


Muchia curentă

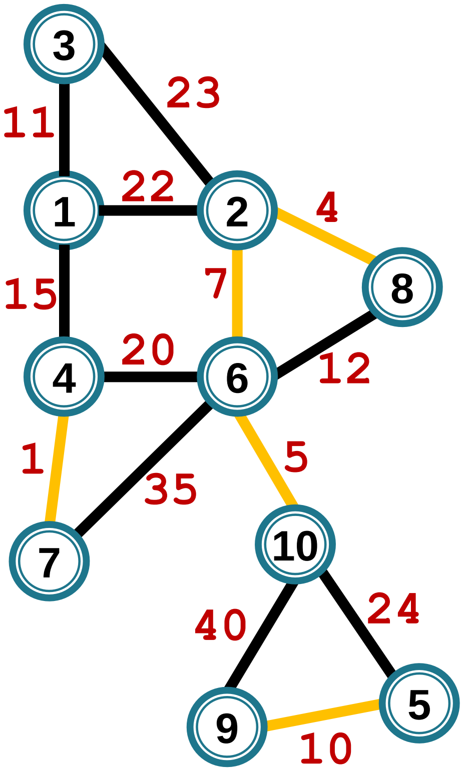
(2,6):

Reprez(2)≠ Reprez(6)

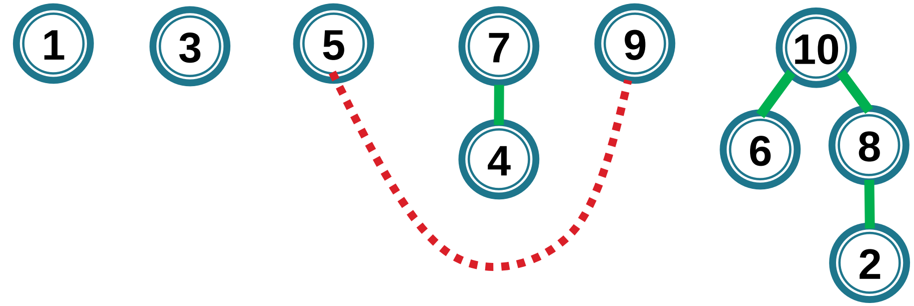
Reuneste(2, 6)



	1	2	3	4	5	6	7	8	9	10
tata	0	8	0	7	0	10	0	10	0	0
h	0	0	0	0	0	0	1	1	0	2



Pădurea de mulțimi disjuncte la pasul curent

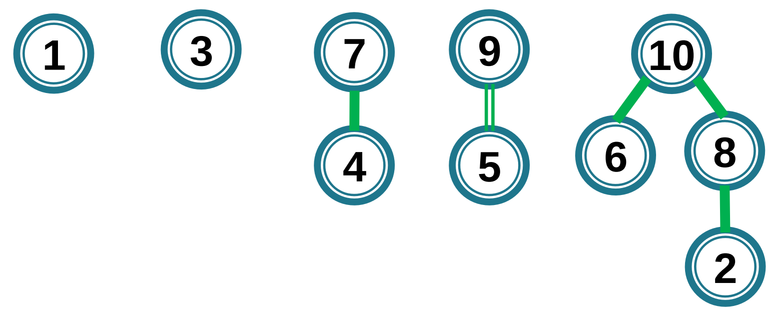


Muchia curentă

(5,9):

Reprez(5)≠ Reprez(5)

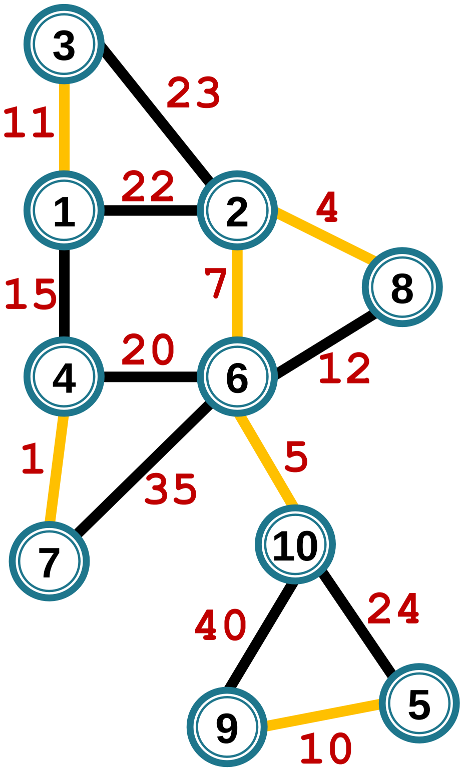
Reuneste(5, 9)



Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

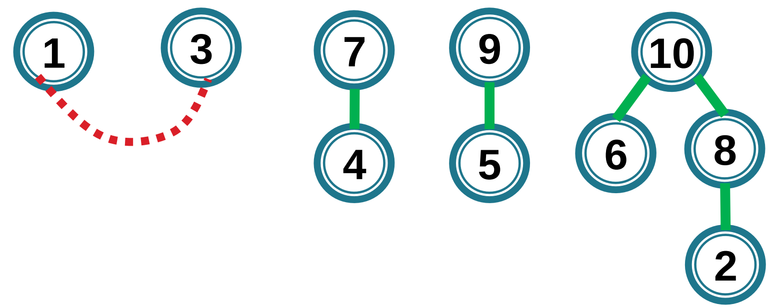
	1	2	3	4	5	6	7	8	9	10
tata	0	8	0	7	9	10	0	10	0	0
h	0	0	0	0	0	0	1	1	1	2



Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent

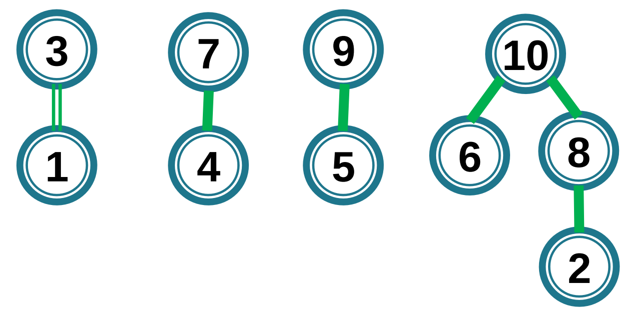


Muchia curentă

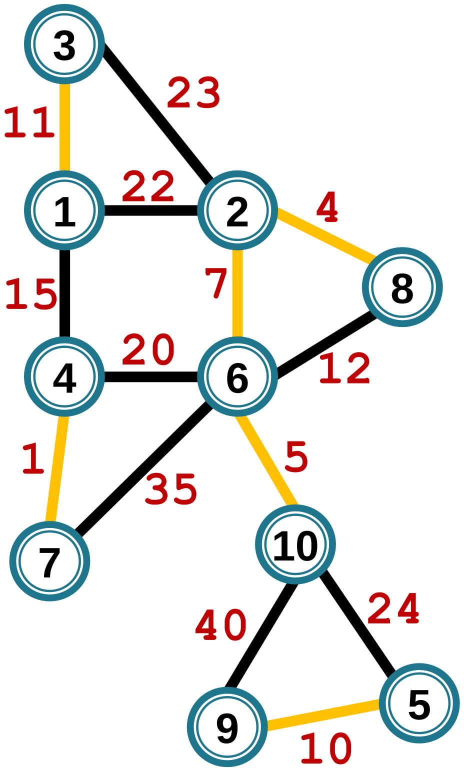
(1,3):

Reprez(1)≠ Reprez(3)

Reuneste(1, 3)



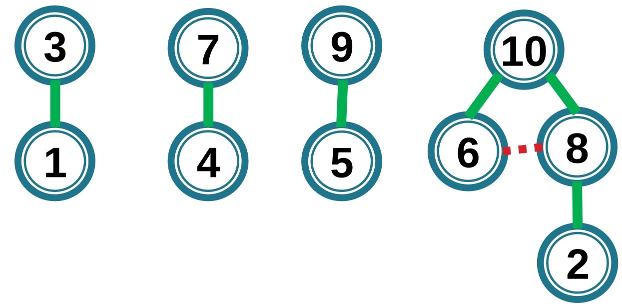
	1	2	3	4	5	6	7	8	9	10
tata	3	8	0	7	9	10	0	10	0	0
h	0	0	1	0	0	0	1	1	1	2



Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent



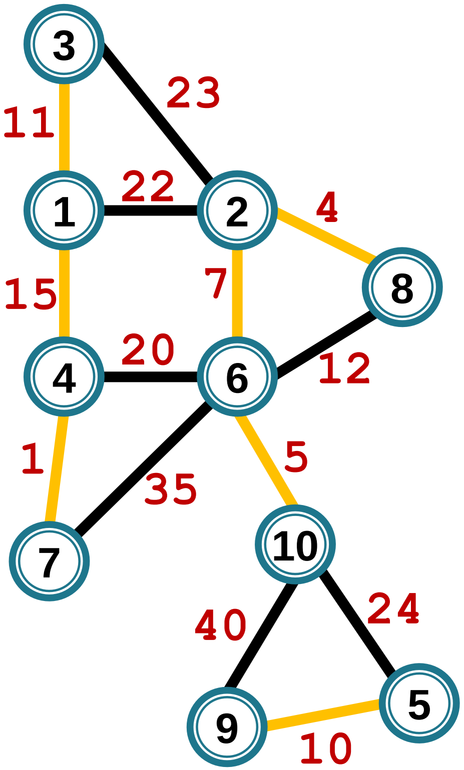
Muchia curentă

(6,8):

Reprez(6) = Reprez(8) ⇒ nu este selectată

Observație: Până acum în funcția Reprez nu a fost modificat vectorul tata prin compresie de cale, deoarece vârfurile erau la distanță cel mult 1 față de rădăcină

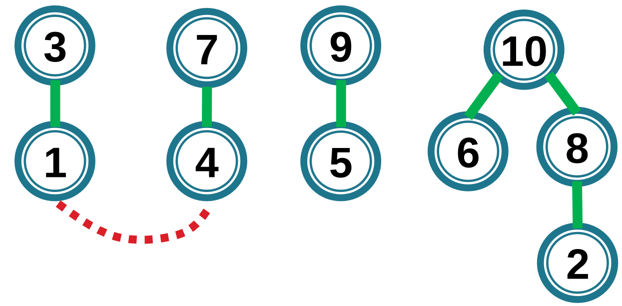
	1	2	3	4	5	6	7	8	9	10
tata	3	8	0	7	9	10	0	10	0	0
h	0	0	1	0	0	0	1	1	1	2



Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent

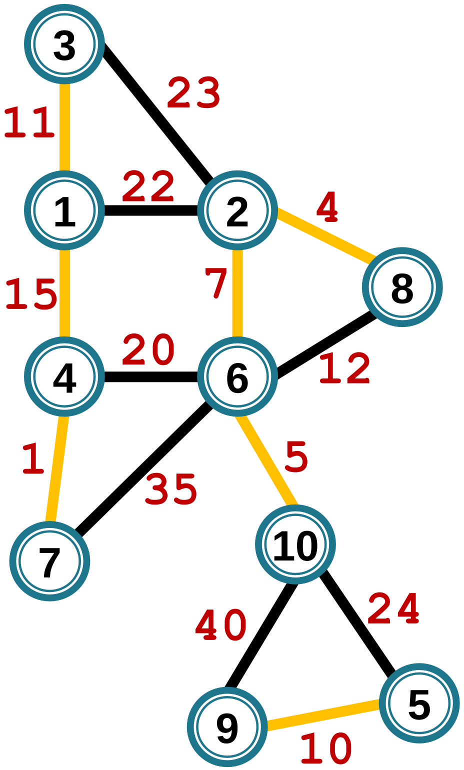


Muchia curentă

(1,4):

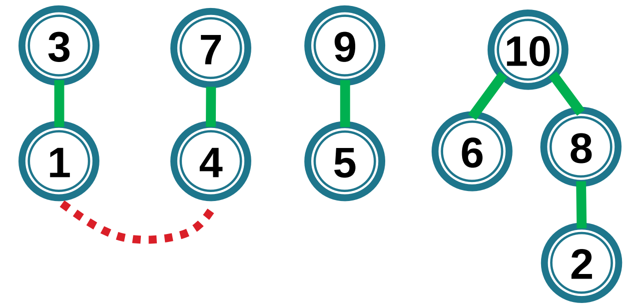
Reprez(1) $\neq$ Reprez(4)

Pădurea de mulțimi disjuncte la pasul curent



Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

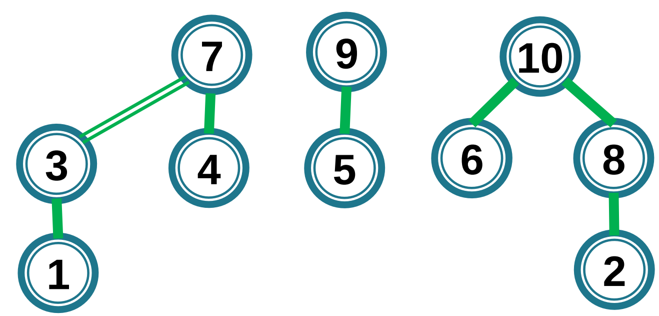


Muchia curentă

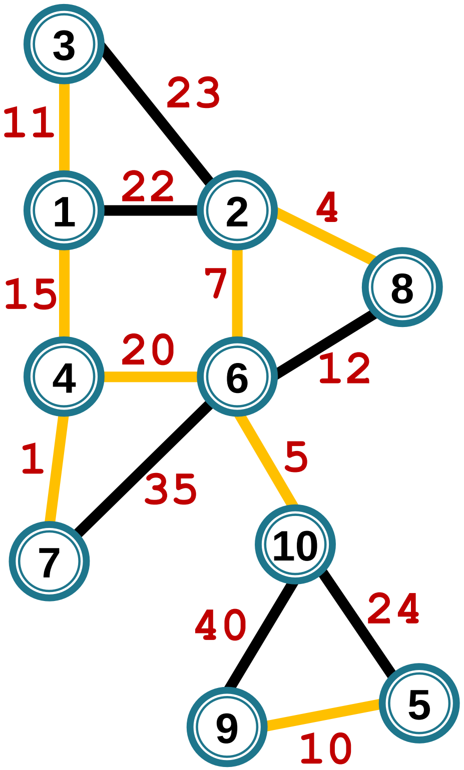
(1,4):

Reprez(1) ≠ Reprez(4)

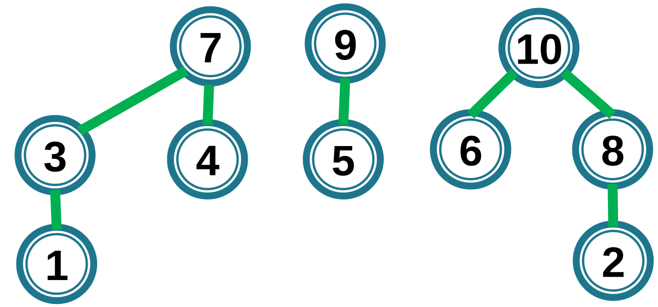
Reuneste(1, 4)



	1	2	3	4	5	6	7	8	9	10
tata	3	8	7	7	9	10	0	10	0	0
h	0	0	1	0	0	0	2	1	1	2



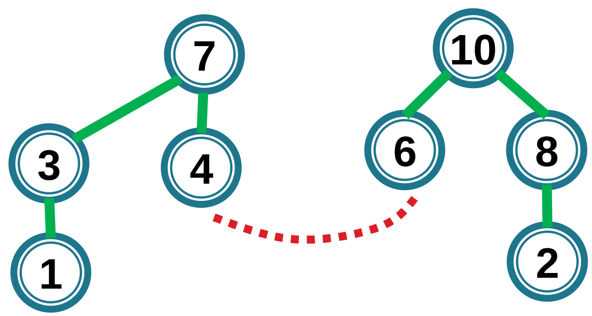
Pădurea de mulțimi disjuncte la pasul curent

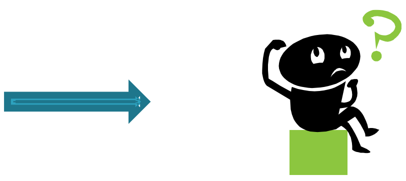


Muchia curentă

(4,6):

Reprez(4) ≠ Reprez(6)

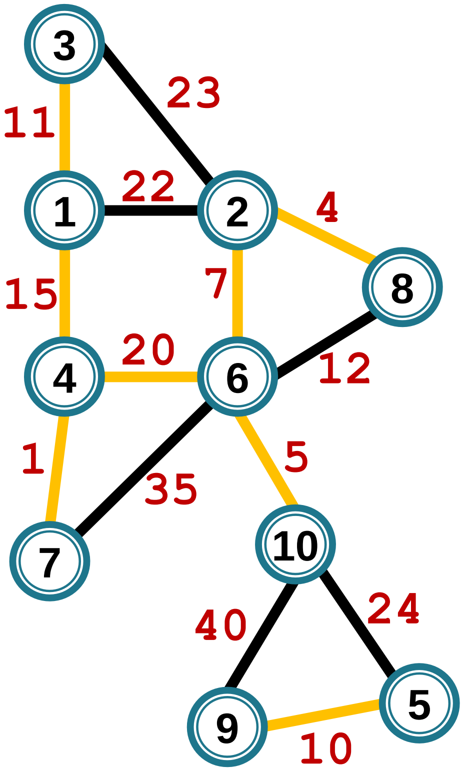


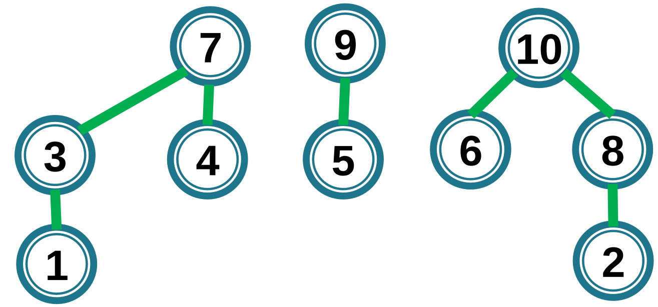


Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent

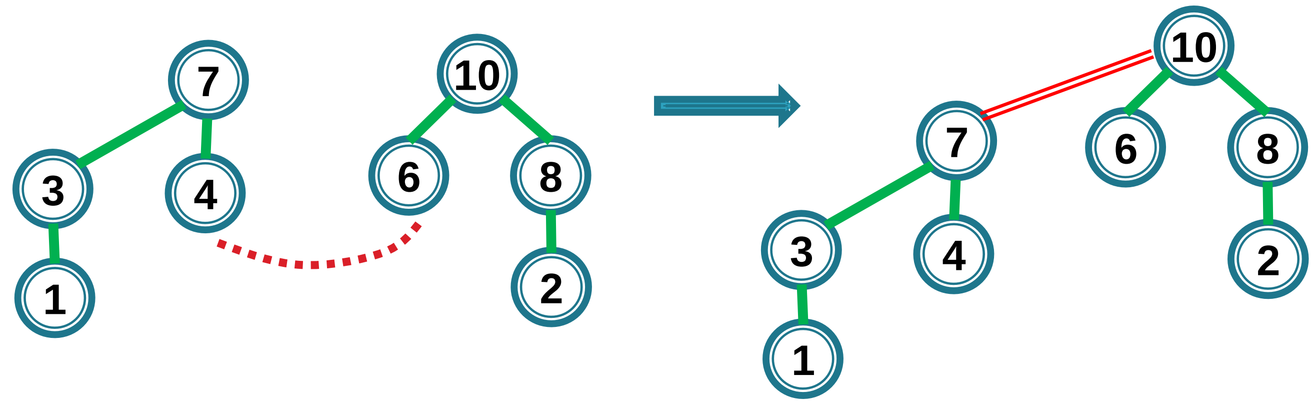




Muchia curentă

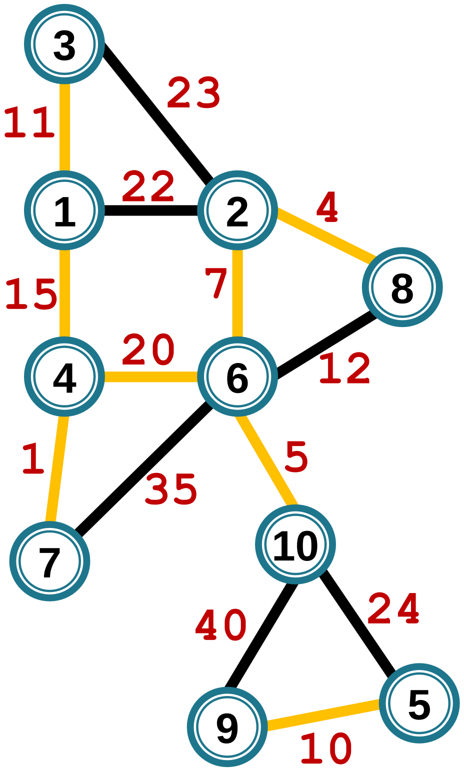
(4,6):

Reprez(4) ≠ Reprez(6)



Ordine muchii

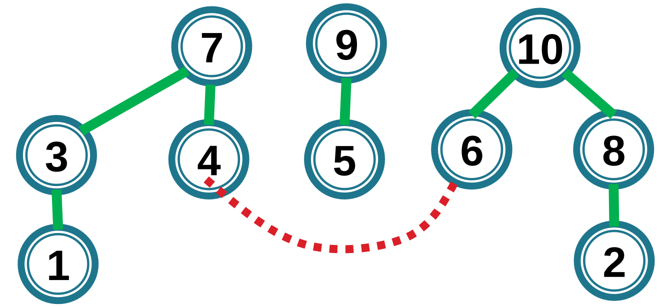
(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)



Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent

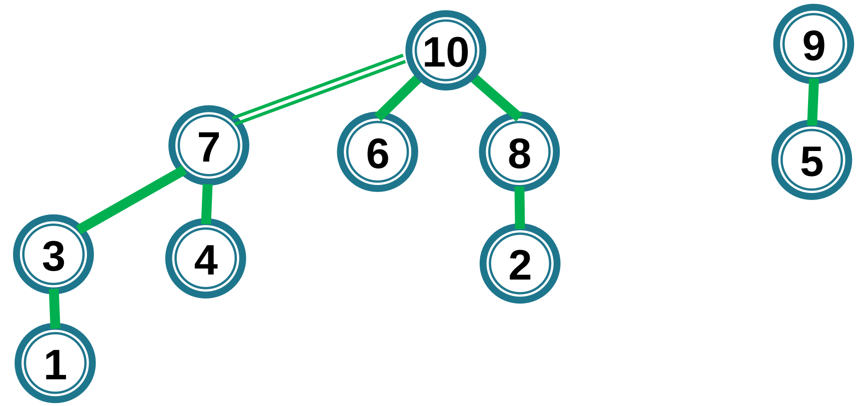


Muchia curentă

(4,6):

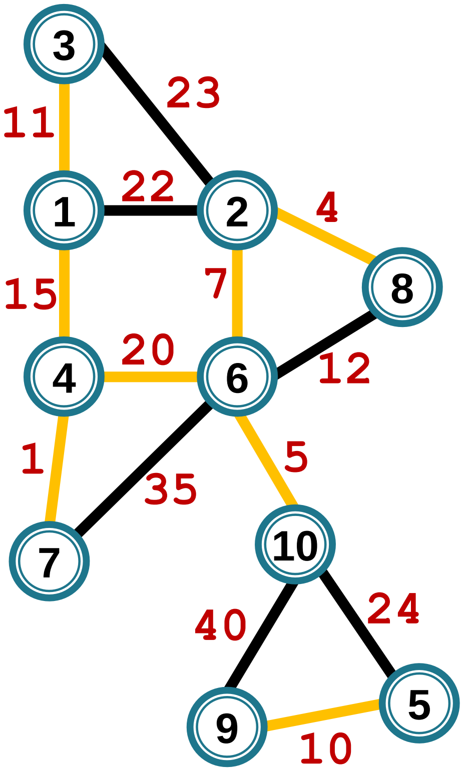
Reprez(4) ≠ Reprez(6)

Reuneste(4, 6)



	1	2	3	4	5	6	7	8	9	10
tata	3	8	7	7	9	10	10	10	0	0
h	0	0	1	0	0	0	2	1	1	3

Pădurea de mulțimi disjuncte la pasul curent



Ordine muchii
(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Muchia curentă

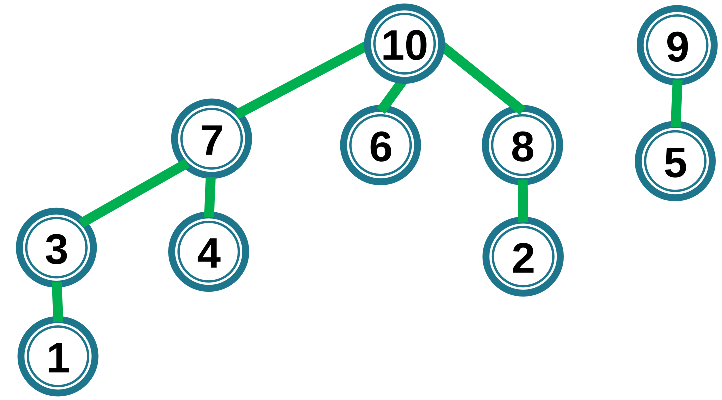
(1,2):

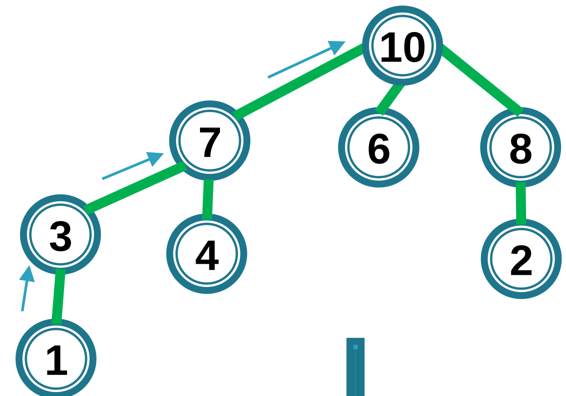
Reprez(1): $\Rightarrow$ 10+

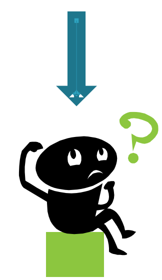
compresie de cale

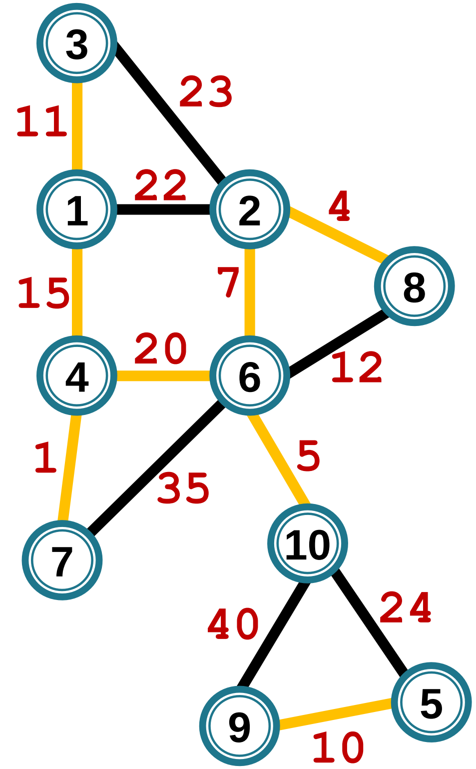
!!h nu se modifica

(h[7] rămâne 2)

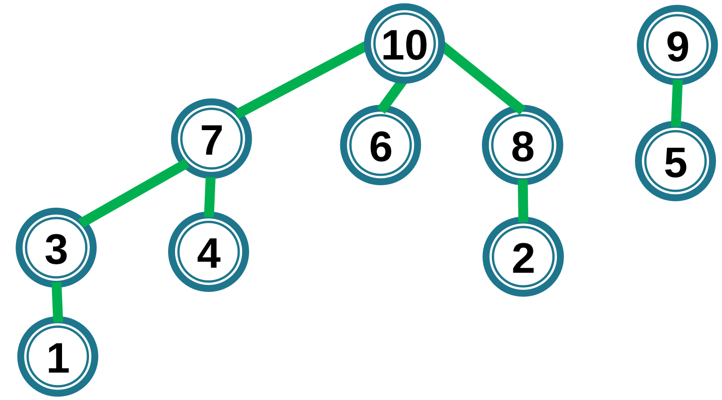








Pădurea de mulțimi disjuncte la pasul curent



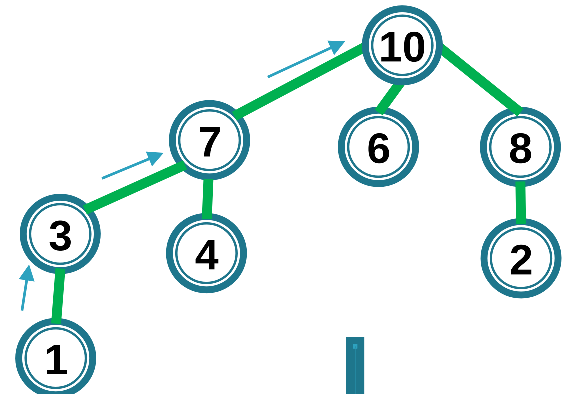
Muchia curentă

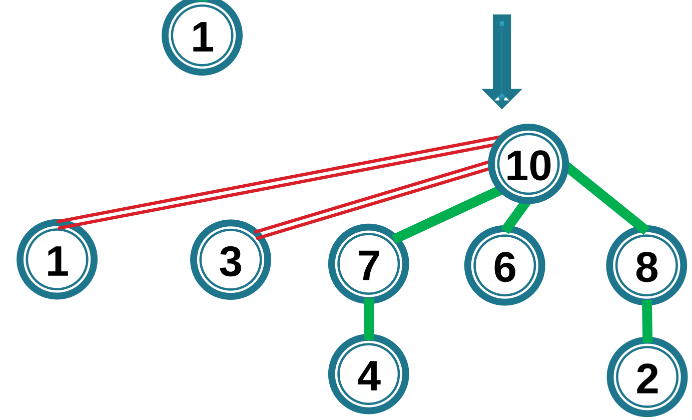
(1,2):

Reprez(1): ⇒ 10+

compresie de cale

!!h nu se modifica
(h[7] rămâne 2)

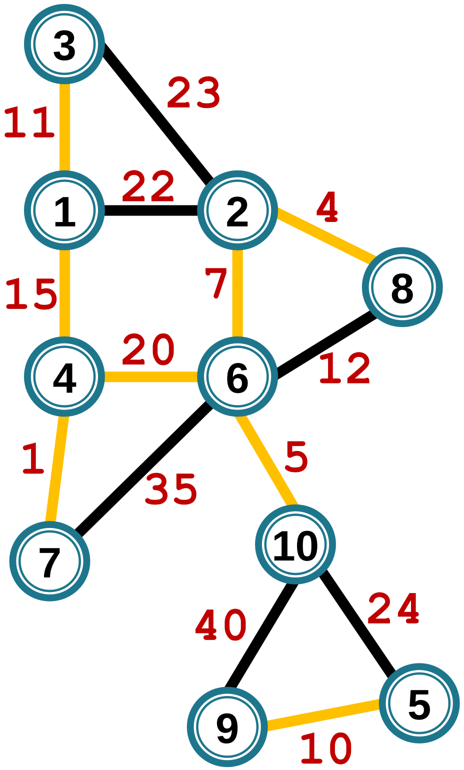




Ordine muchii

(4, 7)	(2, 3)
(2, 8)	(5, 10)
(6, 10)	(6, 7)
(2, 6)	(9, 10)
(5, 9)	
(1, 3)	
(6, 8)	
(1, 4)	
(4, 6)	
(1, 2)	

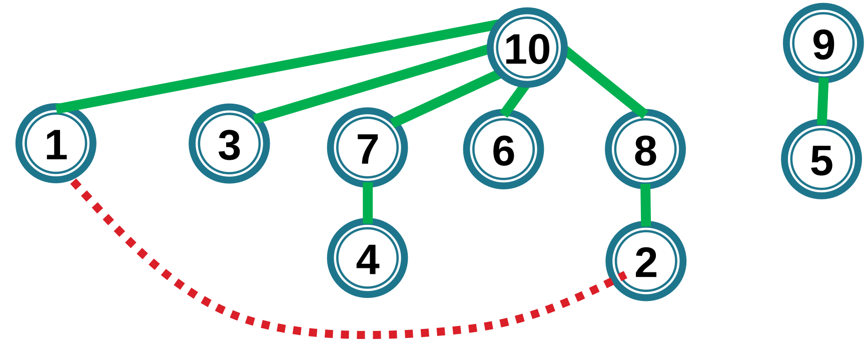
	1	2	3	4	5	6	7	8	9	10
tata	10	8	10	7	9	10	10	10	0	0
h	0	0	1	0	0	0	2	1	1	3



Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent

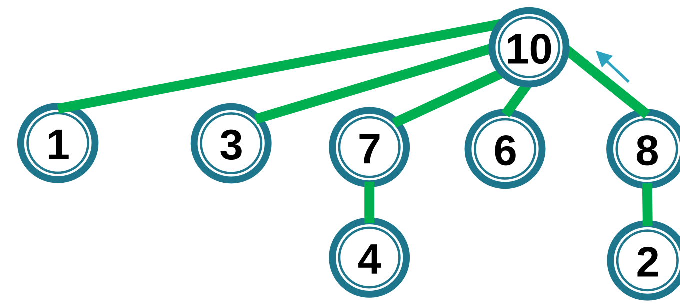


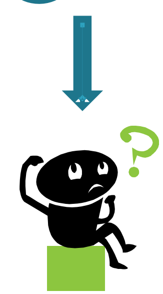
Muchia curentă

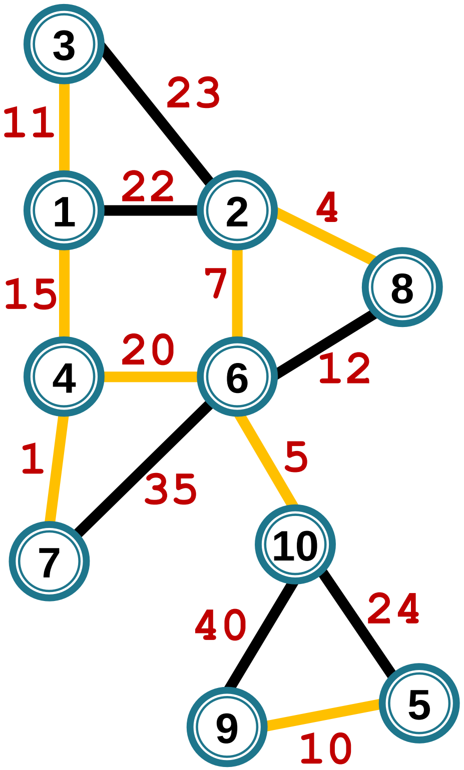
(1,2):

Reprez(2): $\Rightarrow$ 10+

compresie de cale



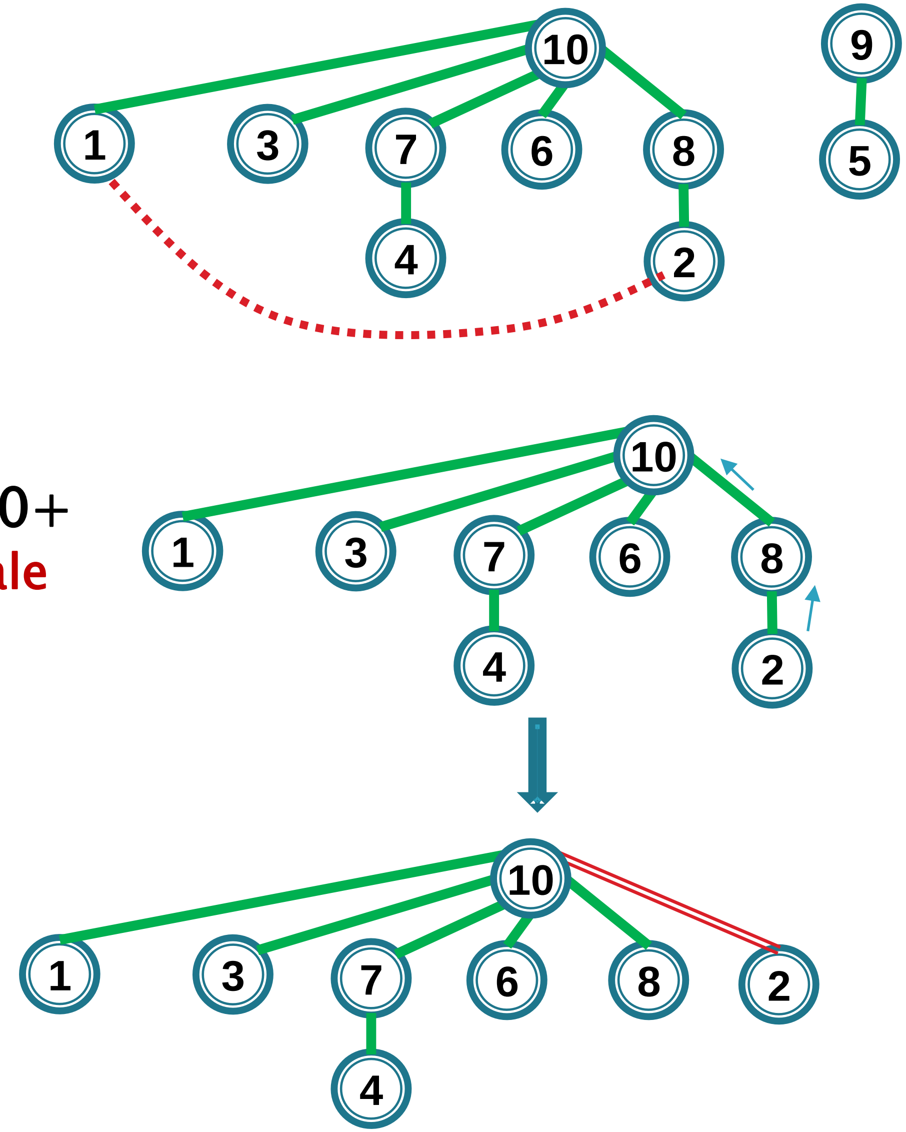




Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent



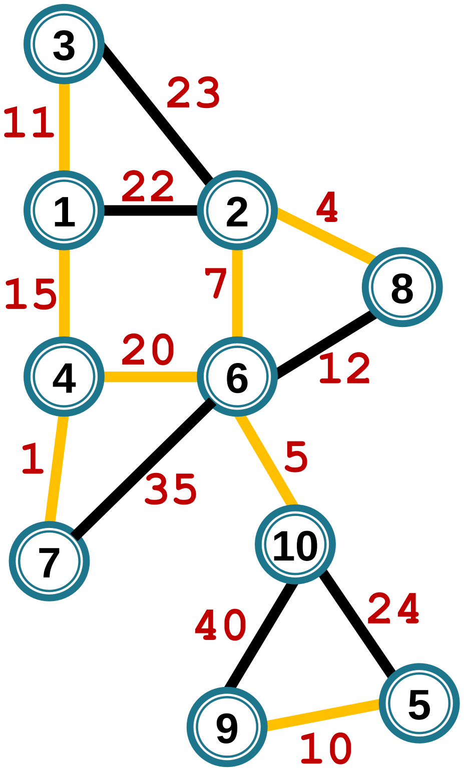
Muchia curentă

(1,2):

Reprez(2): ⇒ 10+

compresie de cale

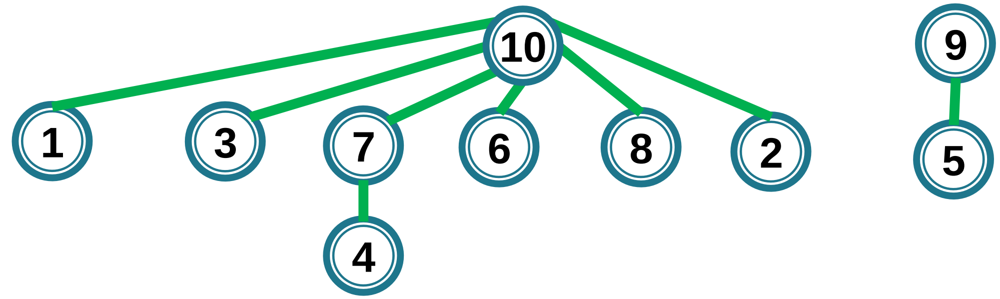
	1	2	3	4	5	6	7	8	9	10
tata	10	10	10	7	9	10	10	10	0	0
h	0	0	1	0	0	0	2	1	1	3



Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent



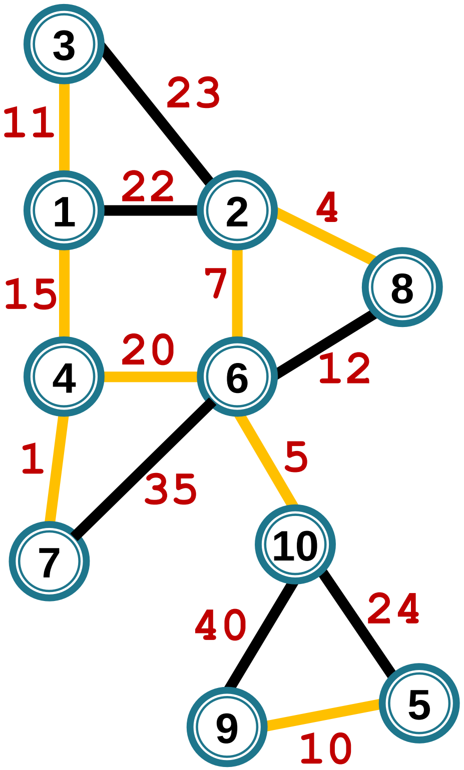
Muchia curentă

(1,2):

Reprez(1) = 10

Reprez(2) = 10 ⇒ nu este selectată

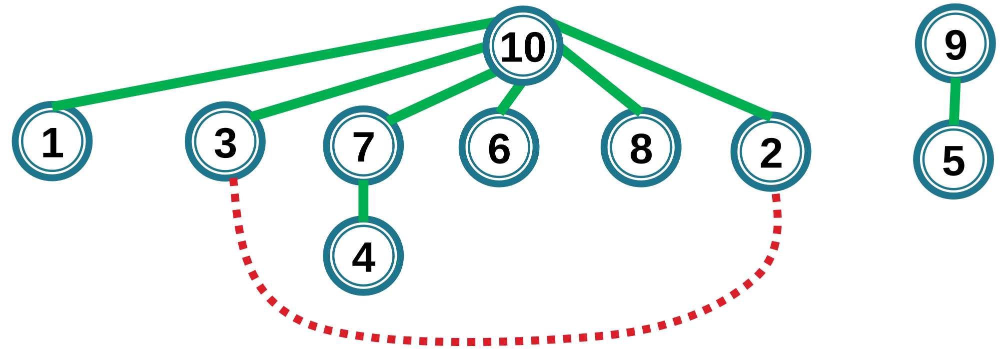
	1	2	3	4	5	6	7	8	9	10
tata	10	10	10	7	9	10	10	10	0	0
h	0	0	1	0	0	0	2	1	1	3



Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent



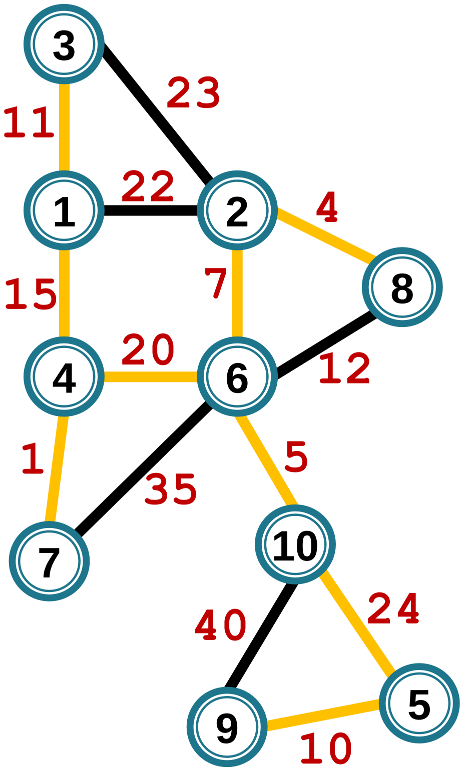
Muchia curentă

(2,3):

Reprez(2) = Reprez(3) ⇒ nu este selectată

- **2 și 3 sunt fii ai rădăcinii, compresia de cale nu modifică vectorul tata**

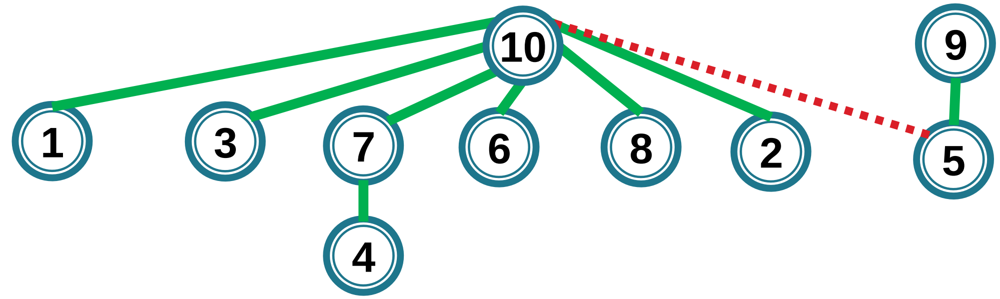
	1	2	3	4	5	6	7	8	9	10
tata	10	10	10	7	9	10	10	10	0	0
h	0	0	1	0	0	0	2	1	1	3



Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent



Muchia curentă

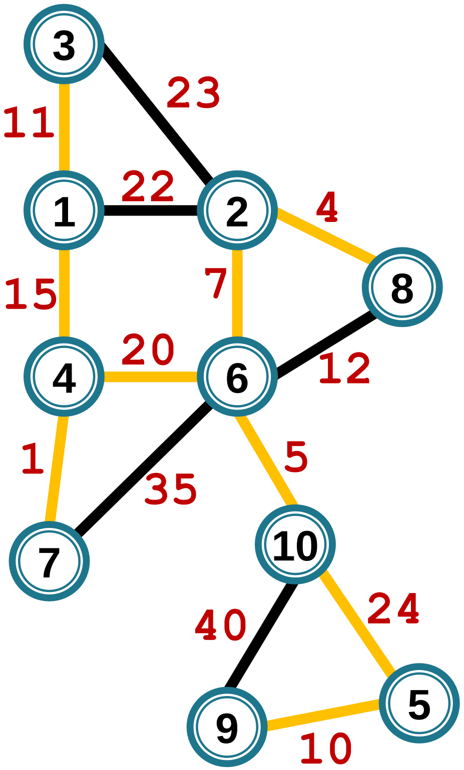
(5,10):

Reprez(5) ≠ Reprez(10)

Reuneste(5, 10)

reuniune ponderată

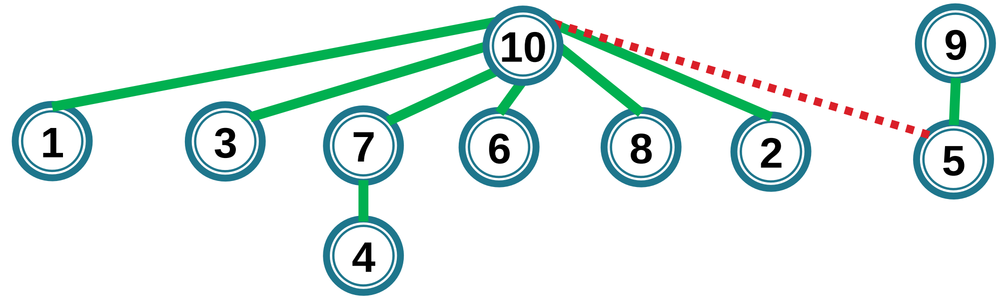
	1	2	3	4	5	6	7	8	9	10
tata	10	10	10	7	9	10	10	10	0	0
h	0	0	1	0	0	0	2	1	1	3



Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent



Muchia curentă

(5,10):

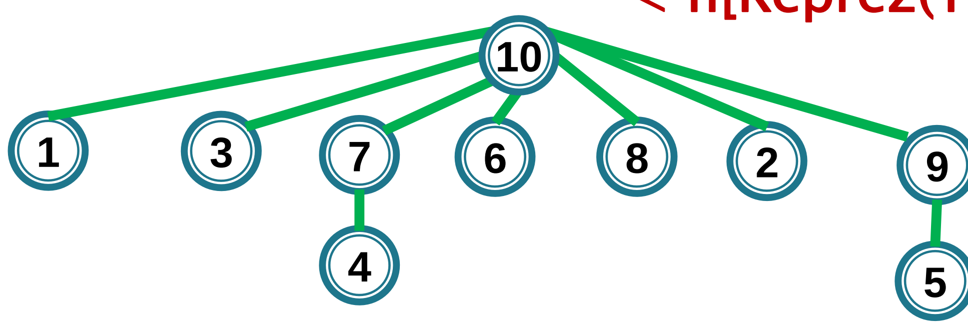
Reprez(5) ≠ Reprez(10)

Reuneste(5, 10)

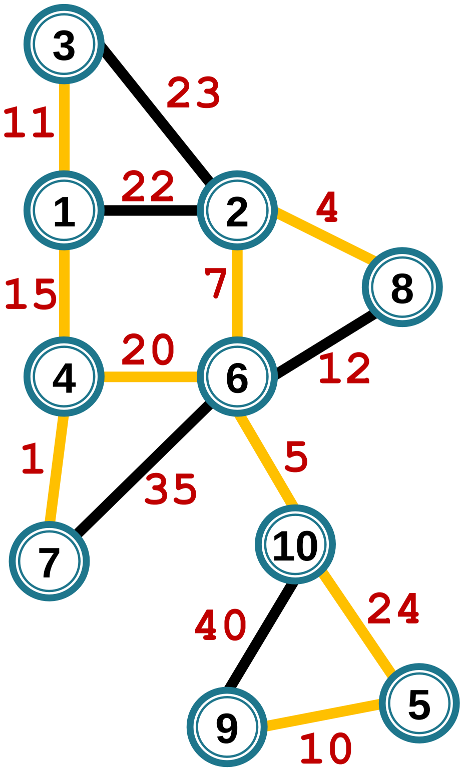
reuniune ponderată

h[Reprez(5)] = h[9] =1

< h[Reprez(10)] = h[10]=3



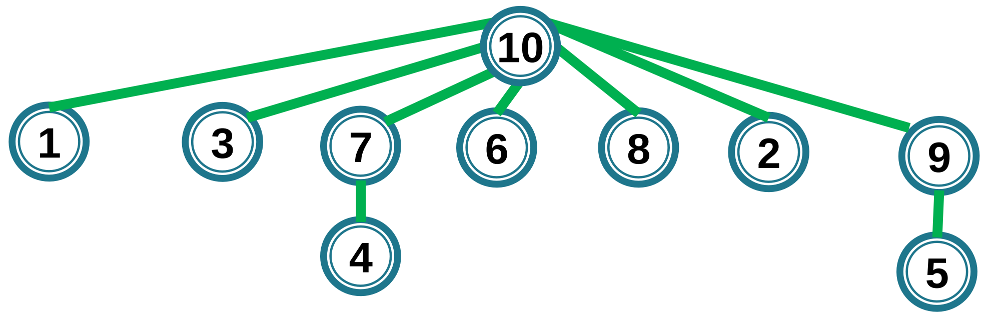
	1	2	3	4	5	6	7	8	9	10
tata	10	10	10	7	9	10	10	10	10	0
h	0	0	1	0	0	0	2	1	1	3



Ordine muchii

(4, 7)
(2, 8)
(6, 10)
(2, 6)
(5, 9)
(1, 3)
(6, 8)
(1, 4)
(4, 6)
(1, 2)
(2, 3)
(5, 10)
(6, 7)
(9, 10)

Pădurea de mulțimi disjuncte la pasul curent



STOP – au fost selectate n-1 muchii

Muchii apcm ≠ muchiile din pădurea de mulțimi disjuncte finală (formată dintr-un singur arbore)

Complexitate operații arbori păduri disjuncte

n elemente

Un șir de $m \geq n$ operații asupra celor n elemente de tip:

- **Initializare**
- **Reprez(u)**
- **Reuneste(u,v)**

=> **Complexitatea?**

Complexitate operații arbori păduri disjuncte

Proprietatea 1

Notăm cu size[x] dimensiunea subarborelui de rădăcină x

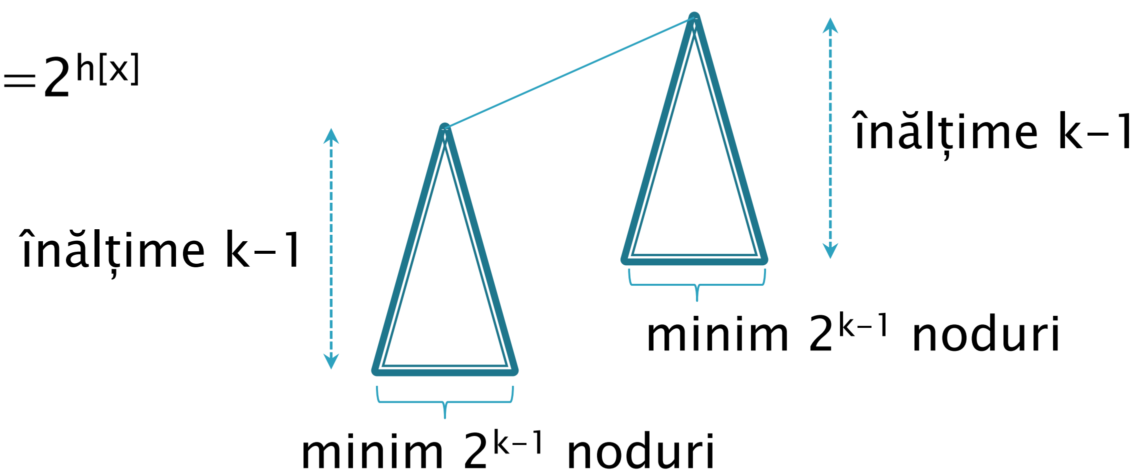
Avem

$$\mathbf{size[x] \geq 2^{h[x]}}$$

Demonstrație. Inducție după k = h[x]

Un nod x de înălțime h[x]=k se obține doar din reuniunea a doi subarbori înălțime k−1. Pentru ele se aplică ipoteza de inducție:

$size[x] \geq 2^{k-1}+2^{k-1}=2^k=2^{h[x]}$



Complexitate operații arbori păduri disjuncte

Proprietatea 2

$h[x] < h[tata[x]]$

(valabilă și cu compresie de cale – h creste pe o cale ce duce către rădăcină)

Proprietatea 3

$h[x] \leq \lg(n)$ pentru orice x

Complexitate operații arbori păduri disjuncte

Teorema 1

Pentru o mulțime cu n elemente și un șir de $m \geq n$ operații de tip **Initializare**, **Reprez**, **Reuneste** cu **reuniune ponderată (după înălțime)**

- complexitatea unei operații de tip **Reprez** sau **Reuneste** este O(log(n)),
- complexitatea șirului de operații este O(m log(n))

Demonstrație – Complexitatea unei operații – dată de înălțimea arborelui

Complexitate operații arbori păduri disjuncte

Teorema 2

Pentru o mulțime cu n elemente și un șir de $m \geq n$ operații de tip **Initializare**, **Reprez**, **Reuneste** cu **reuniune ponderată (după înălțime)** $+$ **compresie de cale**

- complexitatea șirului de operații este O(**m log*(n)**)

unde log*(n) = de câte ori se aplică log lui n pentru a obține o valoare ≤ 1

$$\log^*(n) = \begin{cases} 0, & \text{dacă } n \leq 1 \\ 1 + \log^*\left(\log(n)\right), & \text{dacă } n > 1 \end{cases}$$

Pentru valorile lui n care apar în practică log* n ≤ 5 => O(m)

Complexitate operații arbori păduri disjuncte

Teorema 2

Pentru o mulțime cu n elemente și un șir de $m \geq n$ operații de tip **Initializare**, **Reprez**, **Reuneste** cu **reuniune ponderată (după înălțime)** + **compresie de cale**

- complexitatea șirului de operații este O(**m log*(n)**)

unde log*(n) = de câte ori se aplică log lui n pentru a obține o valoare ≤ 1

$$\log^*(n) = \begin{cases} 0, & \text{dacă } n \leq 1 \\ 1 + \log^*\left(\log(n)\right), & \text{dacă } n > 1 \end{cases}$$

Pentru valorile lui n care apar în practică log* n ≤ 5 => O(m)

https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/UnionFind.pdf

https://en.wikipedia.org/wiki/Disjoint-set_data_structure

Complexitate operații arbori păduri disjuncte

Teorema 2

Pentru o mulțime cu n elemente și un șir de $m \geq n$ operații de tip **Initializare**, **Reprez**, **Reuneste** cu **reuniune ponderată (după înălțime)** + **compresie de cale**

- complexitatea șirului de operații este O(**m log*(n)**)

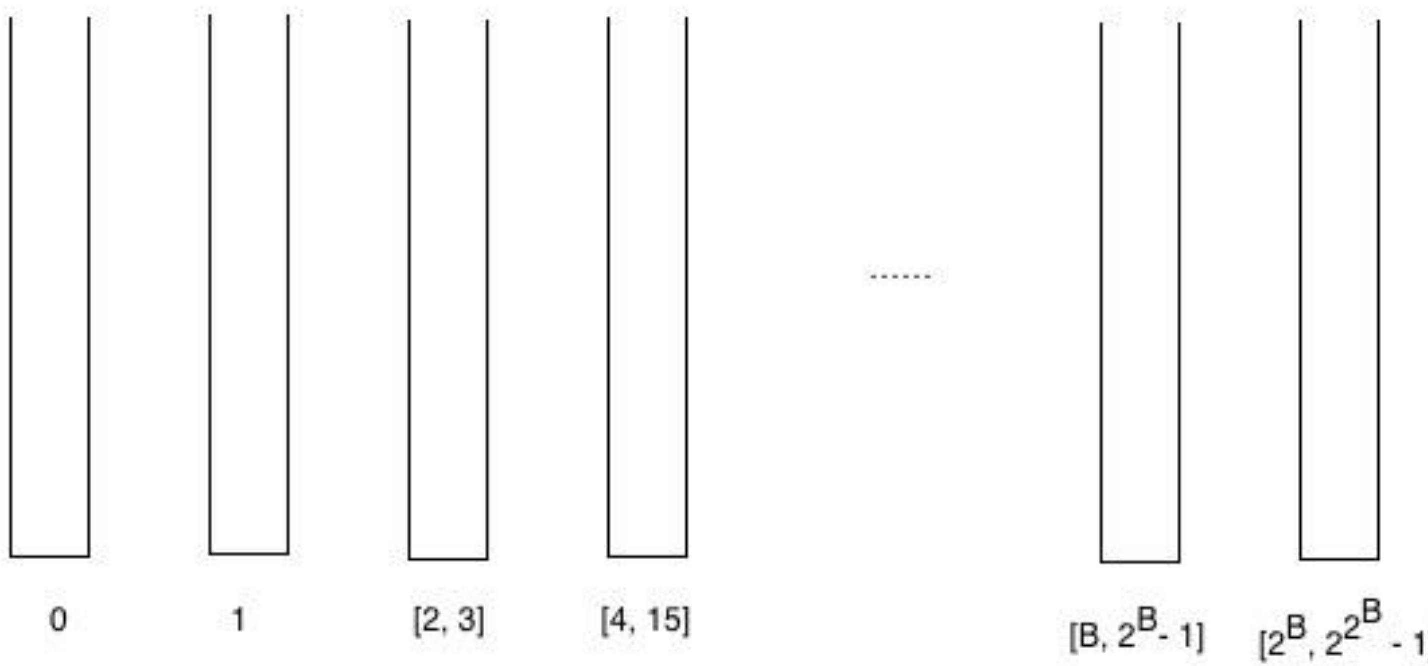
Demonstrație – Temă suplimentară

Idei:

Complexitate operații arbori păduri disjuncte

Numărul maxim de noduri x cu $h[x]=k$ este cel mult $n/2^k$ (deoarece $size[x] \geq 2^k$)

Împărțim vârfurile în ≤ **log*(n)** clase după înălțime



Complexitate operații arbori păduri disjuncte

Determinăm o limită pentru numărul total de muchii parcurse în șirul de operații (în Reprez), în funcție de tipul lor:

-muchie către rădăcină

-muchie în aceeași clasă

-muchie de la o clasă la alta

(h crește strict când urcăm în arbore)