Structuri pentru mulțimi disjuncte

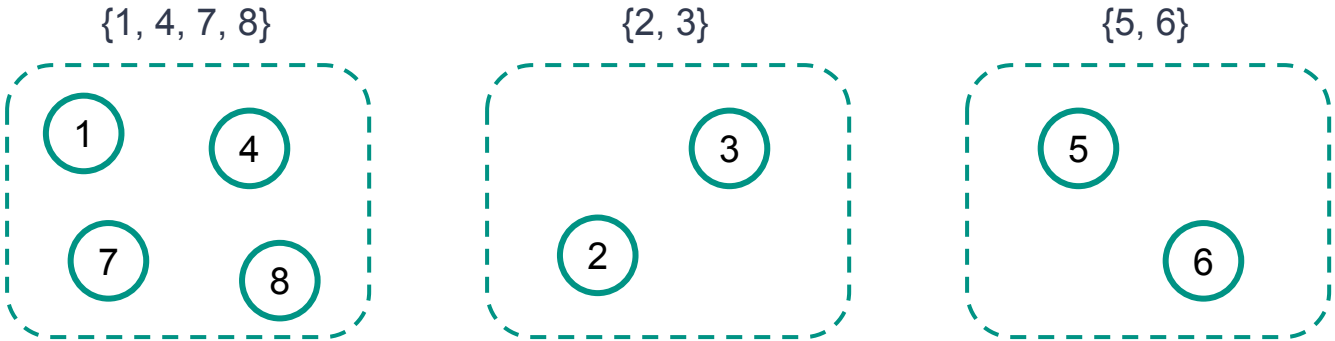
Operații cu mulțimi disjuncte

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 eficient submulțimile, astfel încât operațiile să se efectueze ”eficient”?



Operații cu mulțimi disjuncte

Soluții

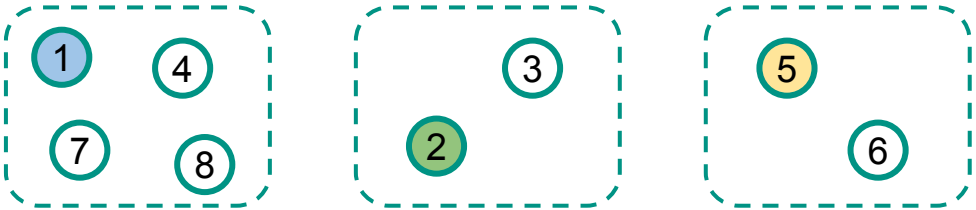
Asociem fiecărei submulțimi un reprezentant (o culoare).

Notăm operațiile:

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

Vector de reprezentanți

Varianta 1 - memorăm într-un vector r, pentru fiecare element x, reprezentantul mulțimii r[x] (v. Kruskal curs)



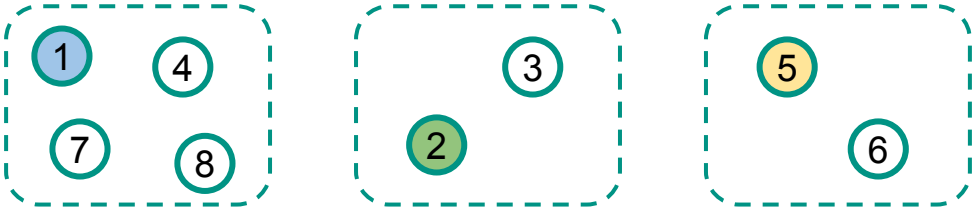
r:

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

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

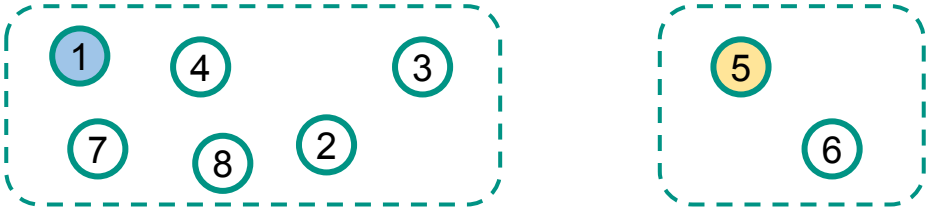
```
void Initializare(int u) { r[u] = u; }
int Reprez(int u) { return r[u]; }
void Reuneste(int u, int v) {
    r1 = Reprez(u);   // r1 = r[u]
    r2 = Reprez(v);   // r2 = r[v]
      for (k=1; k<=n; k++)
          if (r[k] == r2)
              r[k] = r1;
}
```

Vector de reprezentanți - Exemplu



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

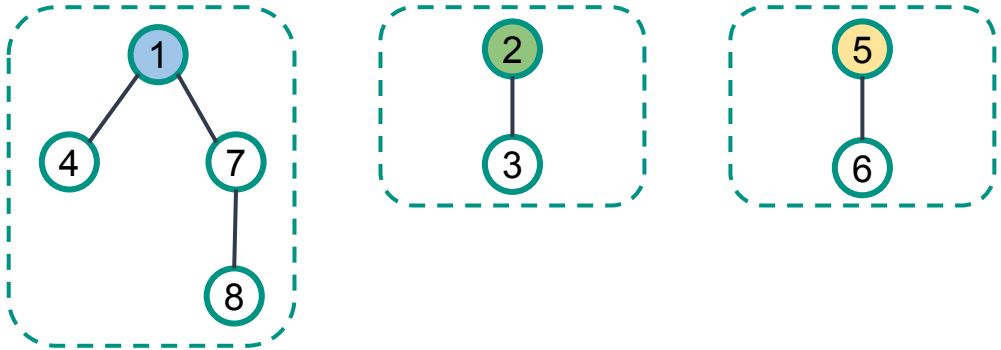
Reuneste(4, 3) ⇒



	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**

- **Inițializare(u)** - O(1)

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

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

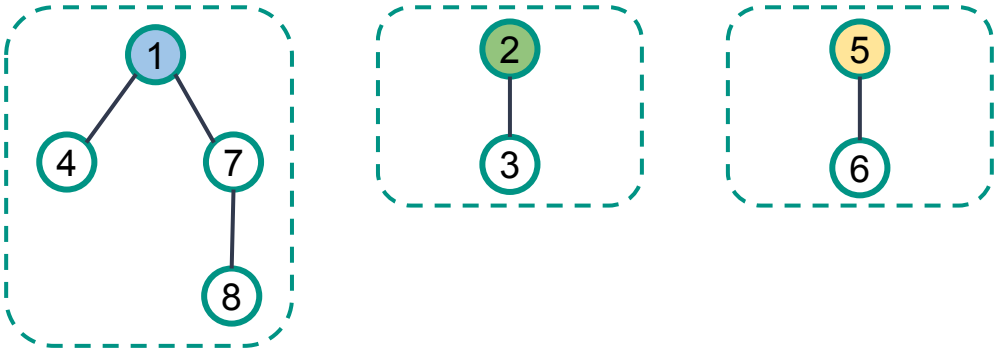
```
int Reprez(int u) {
     while (tata[u] != 0)
          u = tata[u];
     return u;
}
```

- **Reunește(u, v)**
  - reuniune ponderată
    în funcție de înălțimea arborilor
  - O(1) după determinarea reprezentanților lui u și v

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

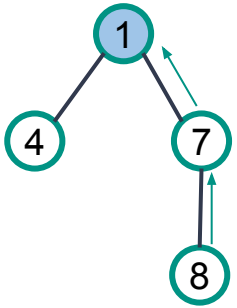
- **Arbori de înălțime logaritmică**

Păduri de mulțimi disjuncte - Exemplu



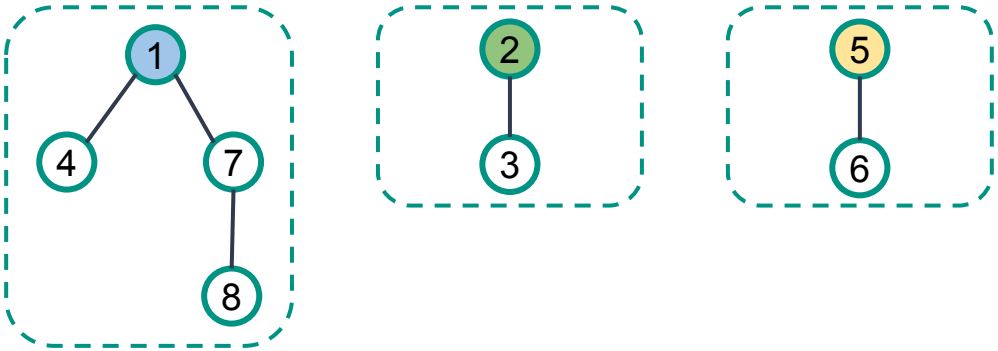
	1	2	3	4	5	6	7	8
tata:	0	0	2	1	0	5	1	7
h:	2	1	0	0	1	0	1	0

Reprez(8) ⇒ returnează 1



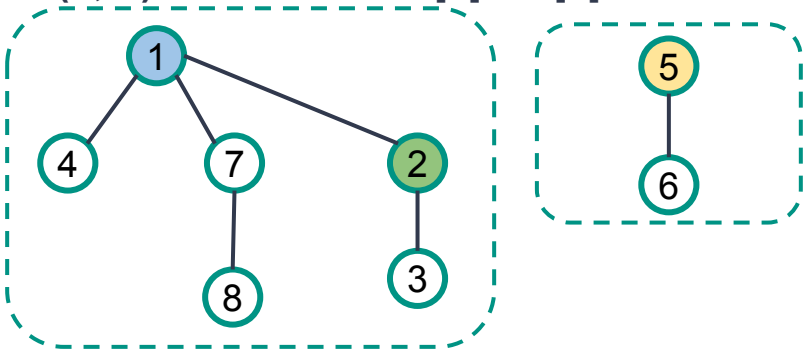
tata[8] = 7, tata[7] = 1, tata[1] = 0

Păduri de mulțimi disjuncte - Exemplu



	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) ⇒ 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 rădăcină

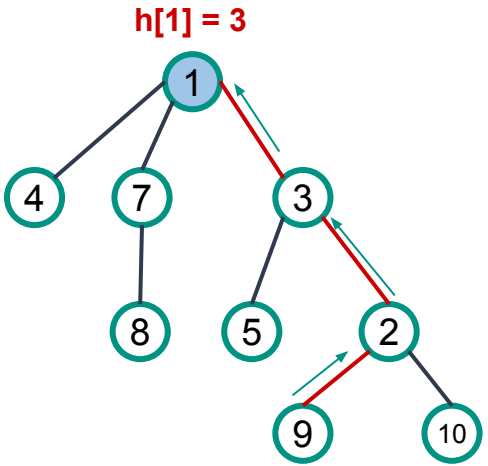
  (vârfurile de pe acest lanț, parcurs pentru a găsi reprezentantul lui u, vor deveni fii ai rădăcinii, pentru ca reprezentantul lor să fie găsit mai ușor în căutările ulterioare)

!! h nu se actualizează

De exemplu, după apelul **Reprez(9)** pentru arborele din dreapta,

rezultatul va fi 1, iar arborele devine

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



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 rădăcină

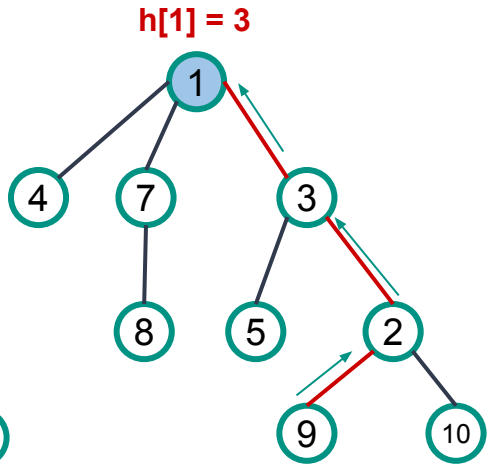
  (vârfurile de pe acest lanț, parcurs pentru a găsi reprezentantul lui u, vor deveni fii ai rădăcinii, pentru ca reprezentantul lor să fie găsit mai ușor în căutările ulterioare)

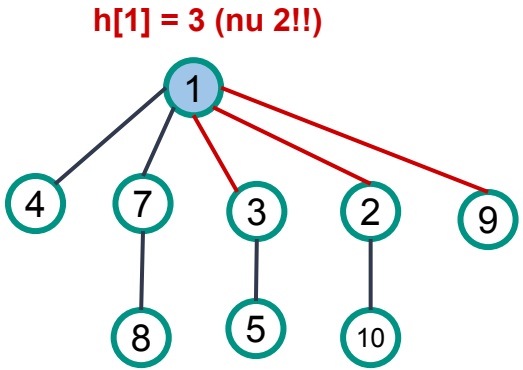
```
int Reprez(int u) {
     if (tata[u] == 0)
          return u;
     tata[u] = Reprez(tata[u]);
     return tata[u];
}
```

!! h nu se actualizează

De exemplu, după apelul **Reprez(9)** pentru arborele din dreapta,

rezultatul va fi 1, iar arborele devine





Algoritmul lui Kruskal

Implementare cu păduri disjuncte

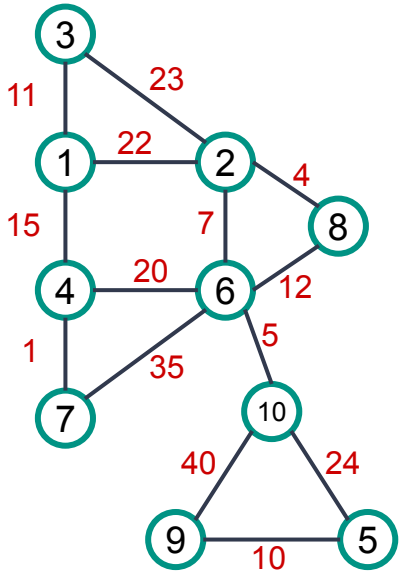
Kruskal – Pseudocod

```
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
}
```

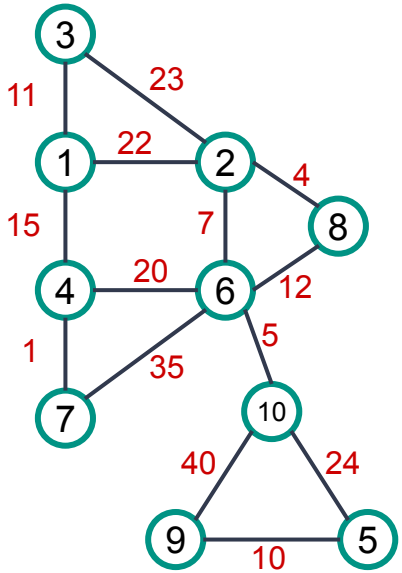


Ordine muchii

(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

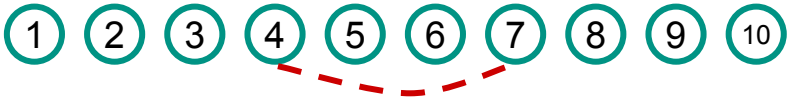
	1	2	3	4	5	6	7	8	9	10
tata	0	0	0	0	0	0	0	0	0	0
h	0	0	0	0	0	0	0	0	0	0



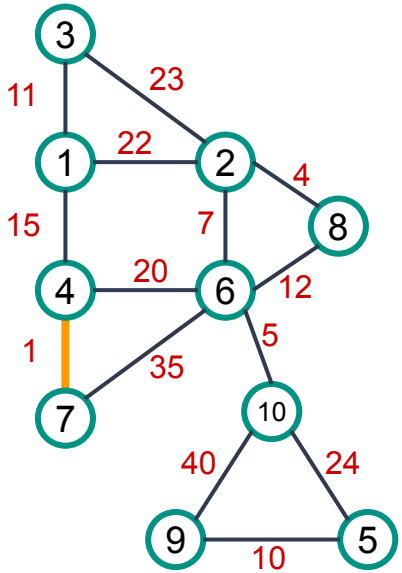
Ordine muchii

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

Pădurea de mulțimi disjuncte la pasul curent



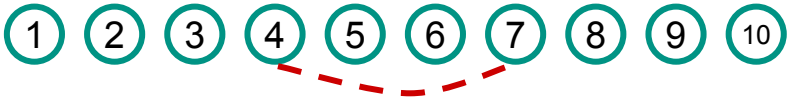
Muchia curentă
(4, 7):



Ordine muchii

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

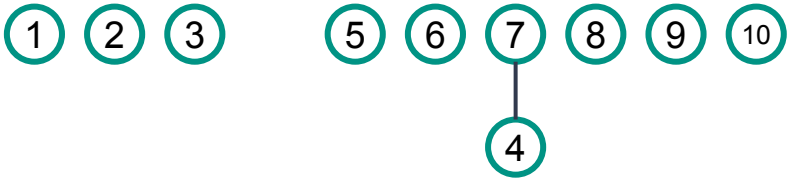
Pădurea de mulțimi disjuncte la pasul curent



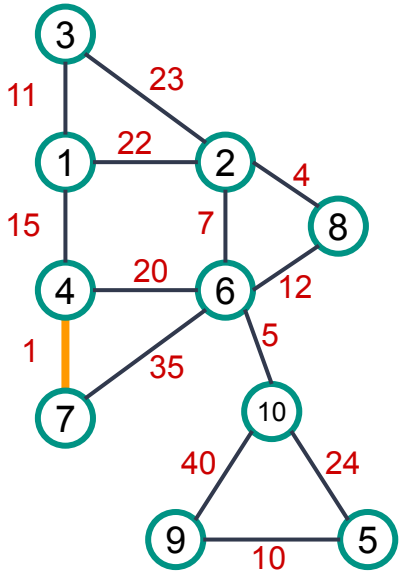
Muchia curentă
(4, 7):

Reprez(4) ≠ Reprez(7)

Reuneste(4, 7)



	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

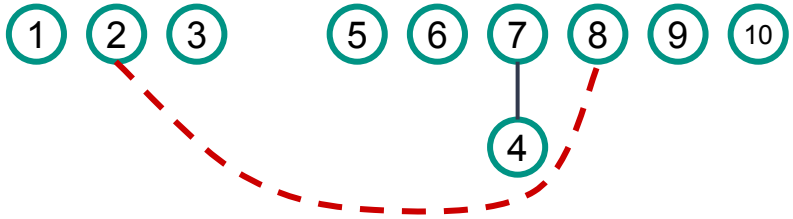


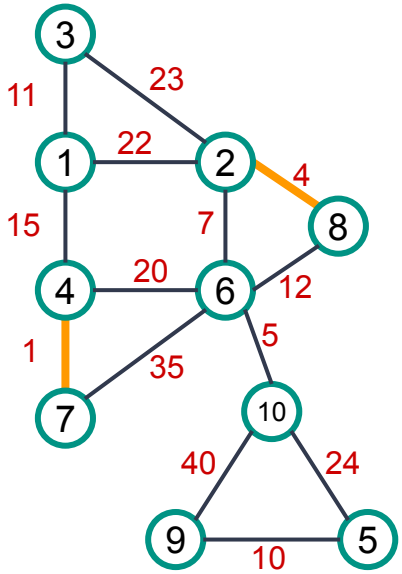
Ordine muchii

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

Muchia curentă
(2, 8):

Pădurea de mulțimi disjuncte la pasul curent

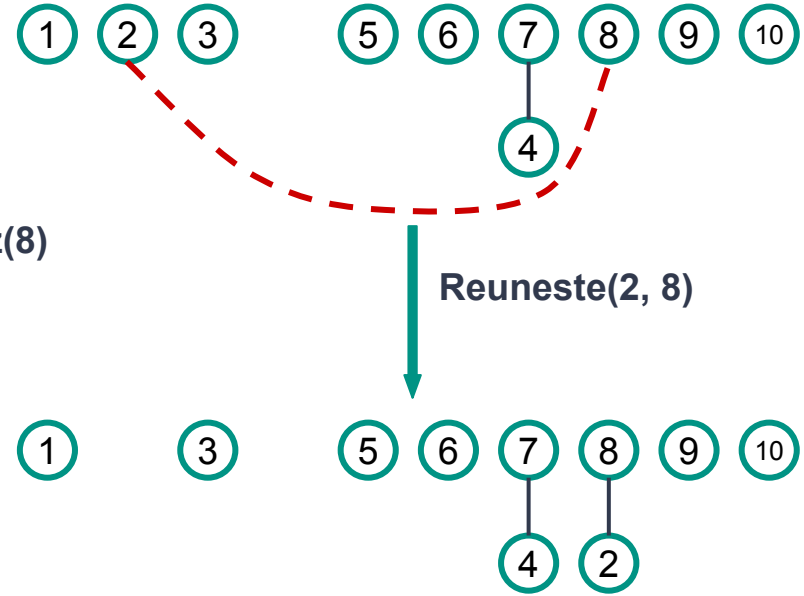




Ordine muchii

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

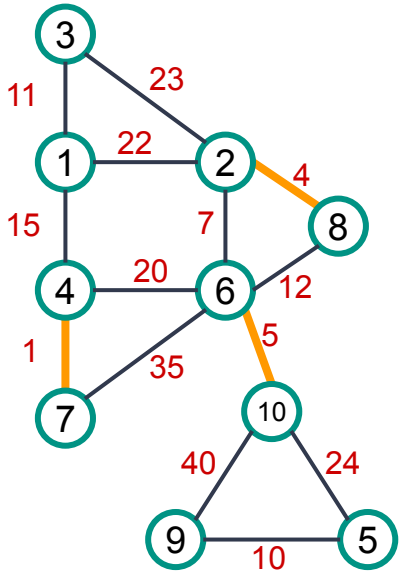
Pădurea de mulțimi disjuncte la pasul curent



Muchia curentă
(2, 8):

Reprez(2) ≠ Reprez(8)

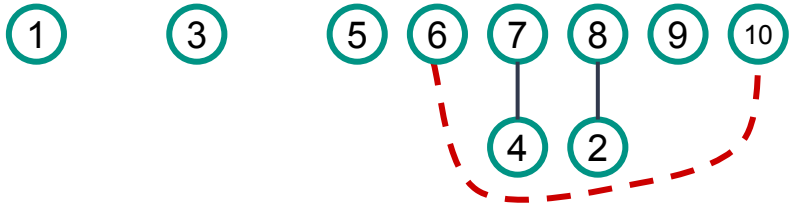
	1	2	3	4	5	6	7	8	9	10
tata	0	**8**	0	7	0	0	0	0	0	0
h	0	0	0	0	0	0	1	**1**	0	0



Ordine muchii

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

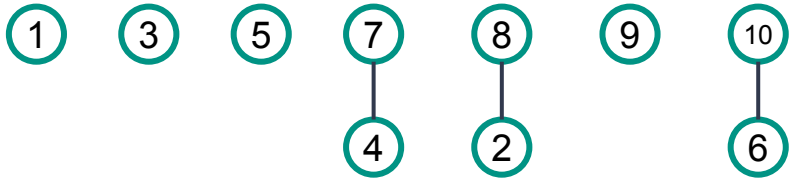
Pădurea de mulțimi disjuncte la pasul curent



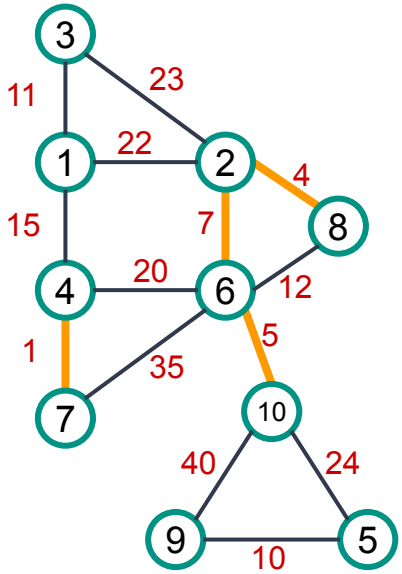
Muchia curentă
(6, 10):

Reprez(6) ≠ Reprez(10)

Reuneste(6, 10)



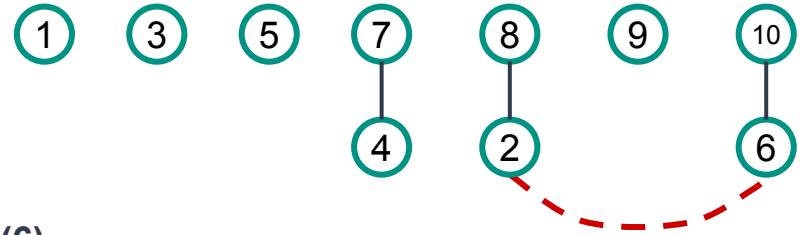
	1	2	3	4	5	6	7	8	9	10
tata	0	8	0	7	0	**10**	0	0	0	0
h	0	0	0	0	0	0	1	1	0	**1**



Ordine muchii

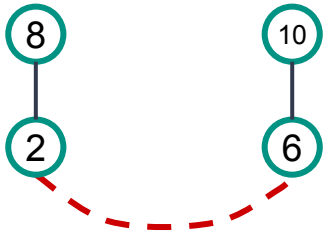
(4, 7)	(4, 6)
(2, 8)	(1, 2)
(6, 10)	(2, 3)
(2, 6)	(5, 10)
(5, 9)	(6, 7)
(1, 3)	(9 ,10)
(6, 8)	
(1, 4)	

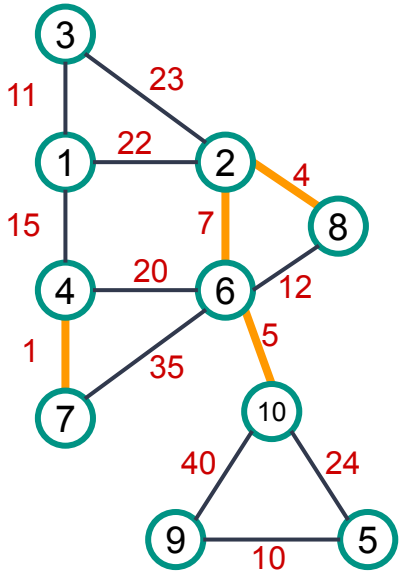
Pădurea de mulțimi disjuncte la pasul curent



Muchia curentă
(2, 6):

Reprez(2) ≠ Reprez(6)

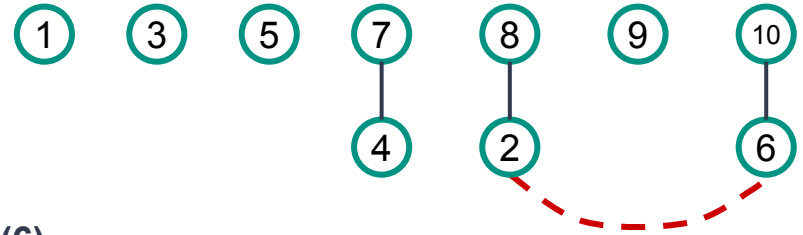




Ordine muchii

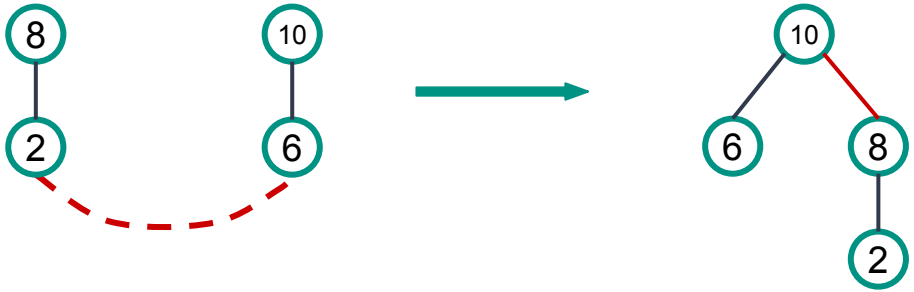
(4, 7)	(4, 6)
(2, 8)	(1, 2)
(6, 10)	(2, 3)
(2, 6)	(5, 10)
(5, 9)	(6, 7)
(1, 3)	(9 ,10)
(6, 8)	
(1, 4)	

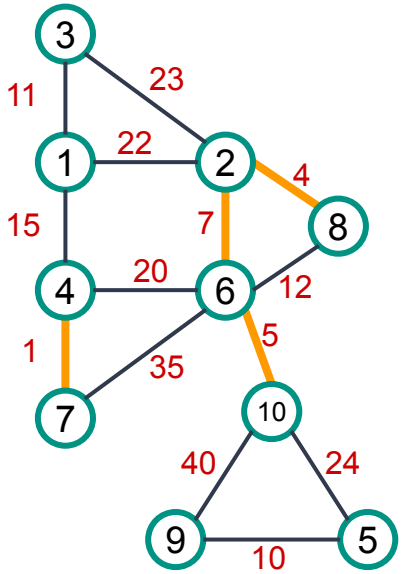
Pădurea de mulțimi disjuncte la pasul curent



Muchia curentă
(2, 6):

Reprez(2) ≠ Reprez(6)

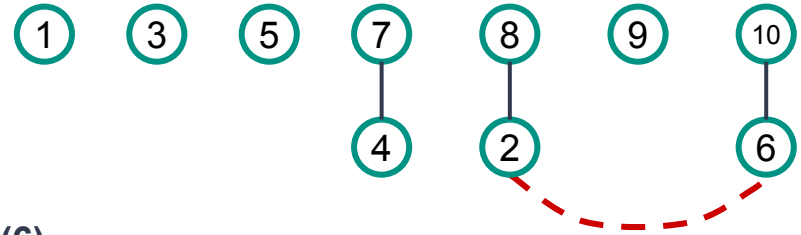




Ordine muchii

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

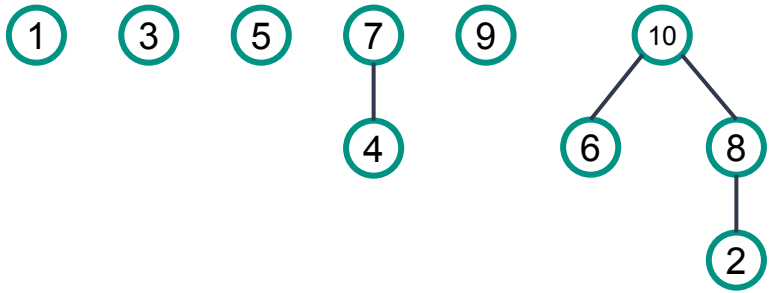
Pădurea de mulțimi disjuncte la pasul curent



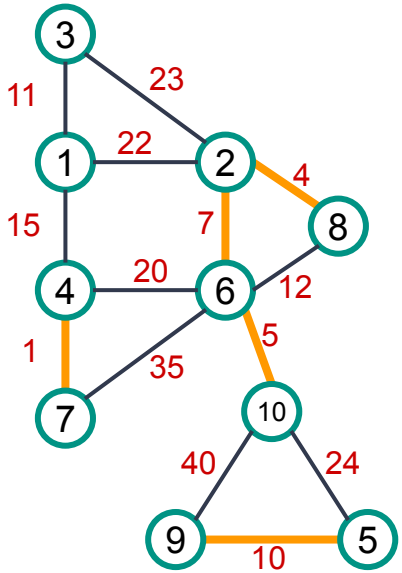
Muchia curentă
(2, 6):

Reprez(2) ≠ Reprez(6)

Reuneste(2, 6)



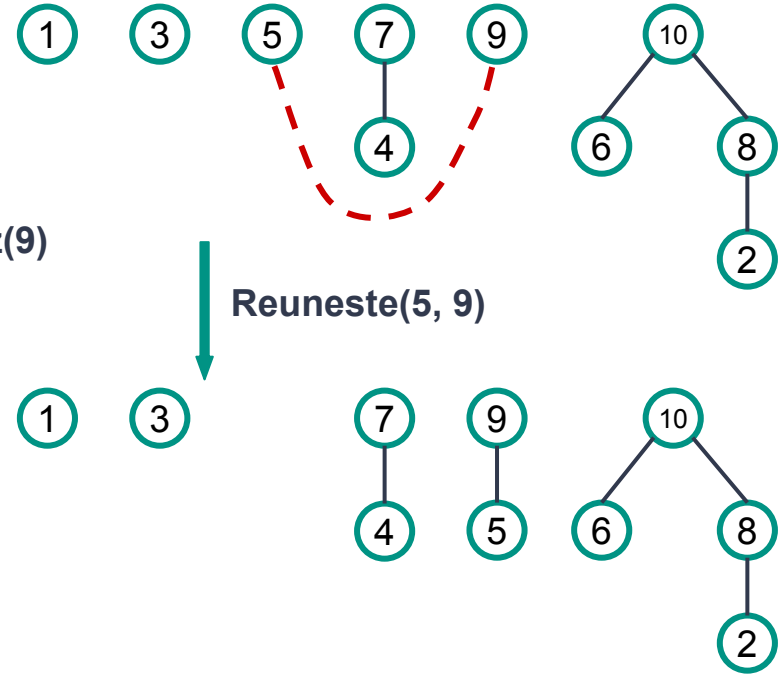
	1	2	3	4	5	6	7	8	9	10
tata	0	8	0	7	0	10	0	**10**	0	0
h	0	0	0	0	0	0	1	1	0	**1**



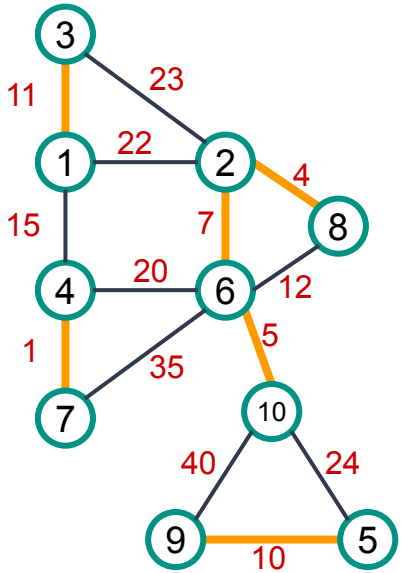
Ordine muchii

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

Pădurea de mulțimi disjuncte la pasul curent



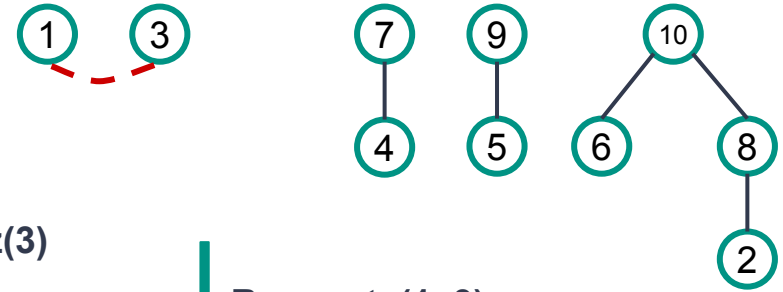
	1	2	3	4	5	6	7	8	9	10
tata	0	8	0	7	**9**	10	0	10	0	0
h	0	0	0	0	0	0	1	1	**1**	1



Ordine muchii

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

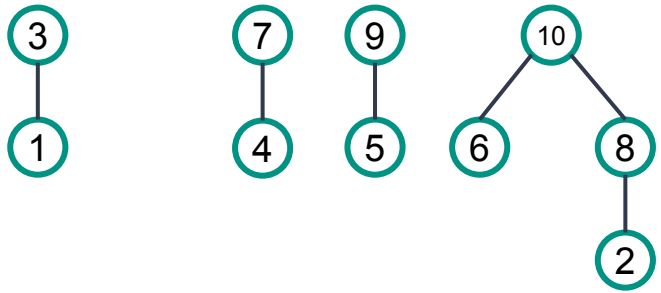
Pădurea de mulțimi disjuncte la pasul curent



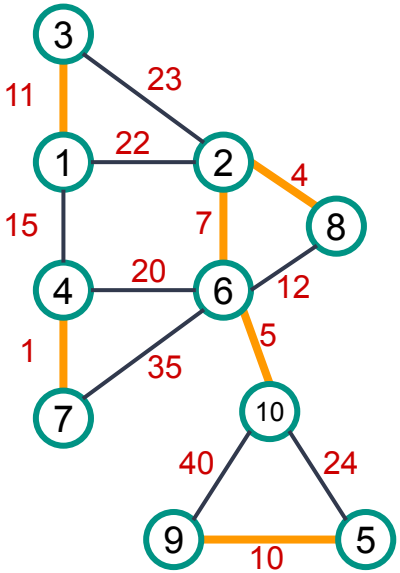
Muchia curentă
(1, 3):

Reprez(1) ≠ Reprez(3)

Reuneste(1, 3)



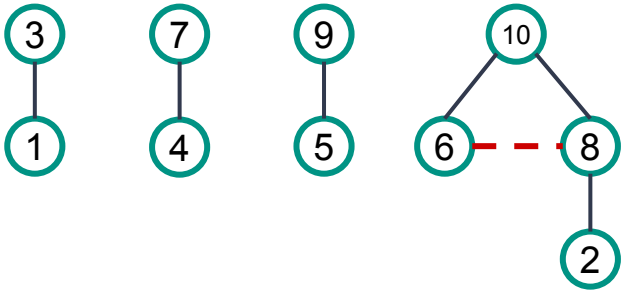
	1	2	3	4	5	6	7	8	9	10
tata	**3**	8	0	7	9	10	0	10	0	0
h	0	0	**1**	0	0	0	1	1	1	1



Ordine muchii

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

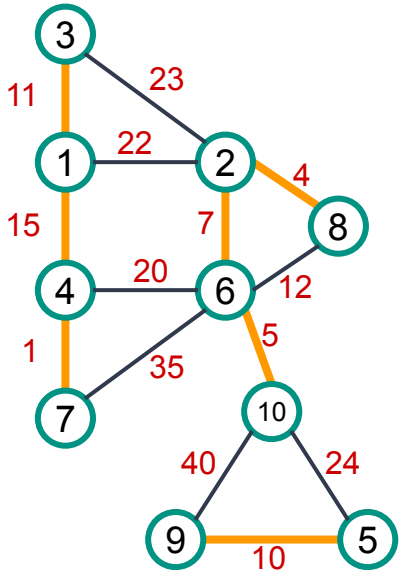
Pădurea de mulțimi disjuncte la pasul curent



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	1



Ordine muchii

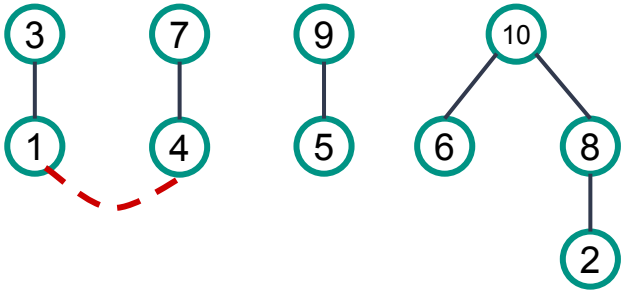
(4, 7)	(4, 6)
(2, 8)	(1, 2)
(6, 10)	(2, 3)
(2, 6)	(5, 10)
(5, 9)	(6, 7)
(1, 3)	(9 ,10)
(6, 8)	

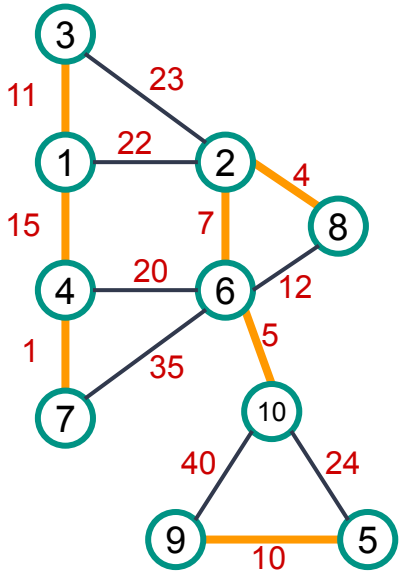
(1, 4)

Muchia curentă
(1, 4):

Reprez(1) ≠ Reprez(4)

Pădurea de mulțimi disjuncte la pasul curent

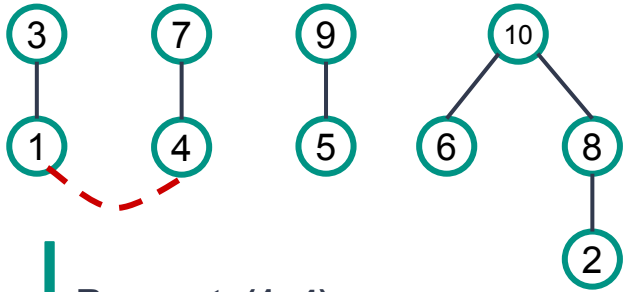




Ordine muchii

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

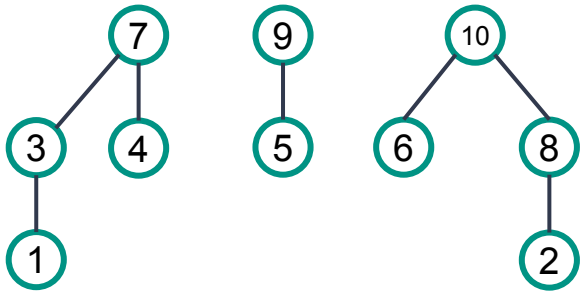
Pădurea de mulțimi disjuncte la pasul curent



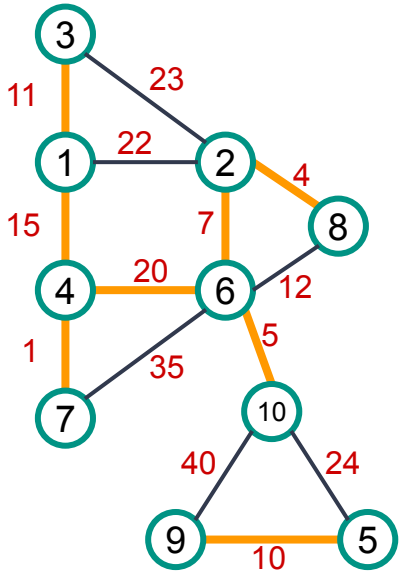
Muchia curentă
(1, 4):

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	1

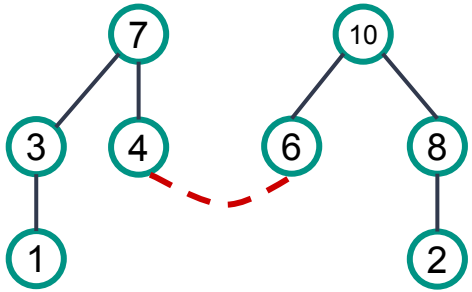


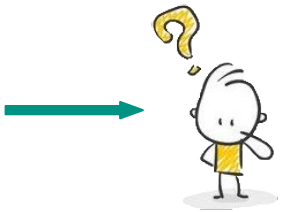
Ordine muchii

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

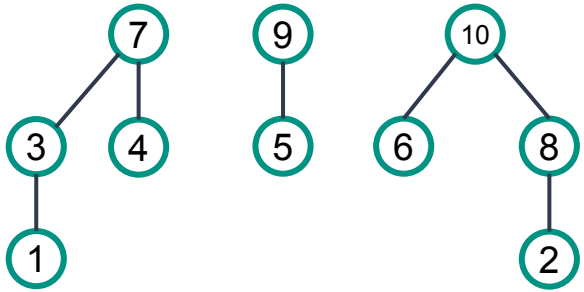
Muchia curentă
(4, 6):

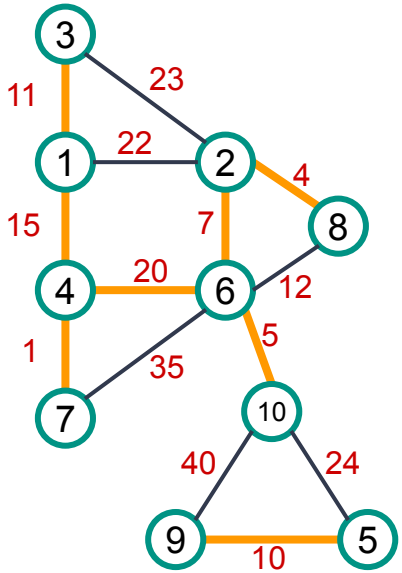
Reprez(4) ≠ Reprez(6)





Pădurea de mulțimi disjuncte la pasul curent





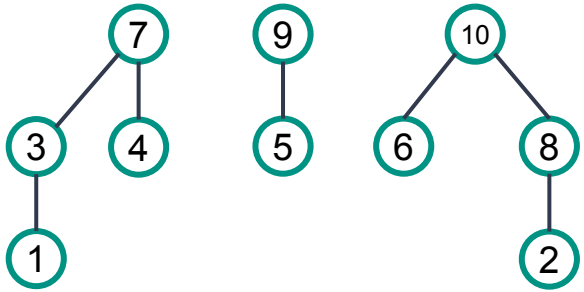
Ordine muchii

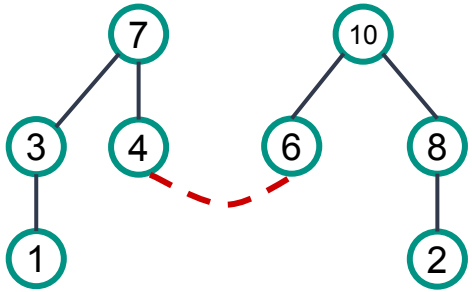
(4, 7)	**(4, 6)**
(2, 8)	(1, 2)
(6, 10)	(2, 3)
(2, 6)	(5, 10)
(5, 9)	(6, 7)
(1, 3)	(9 ,10)
(6, 8)	
(1, 4)	

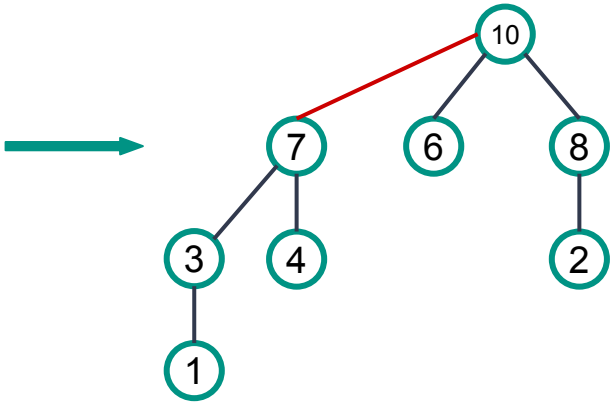
Muchia curentă
(4, 6):

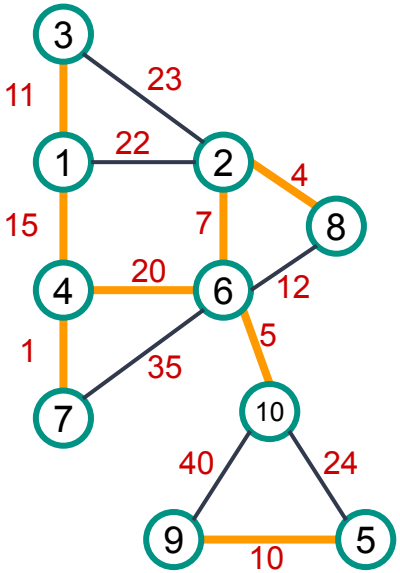
Reprez(4) ≠ Reprez(6)

Pădurea de mulțimi disjuncte la pasul curent





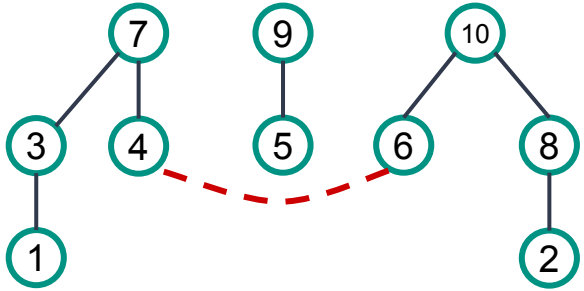




Ordine muchii

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

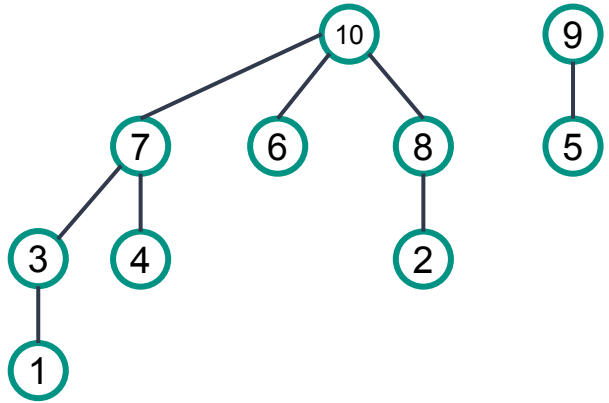
Pădurea de mulțimi disjuncte la pasul curent



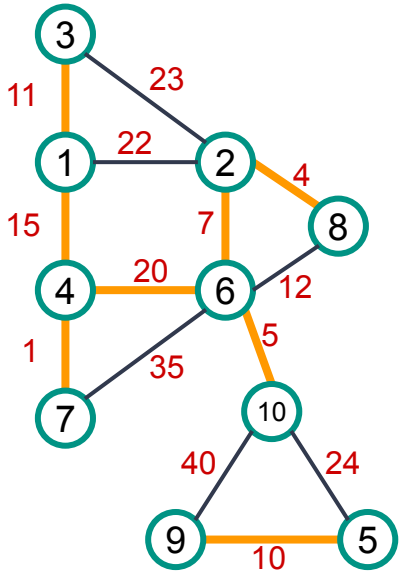
Muchia curentă
(4, 6):

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**



Ordine muchii

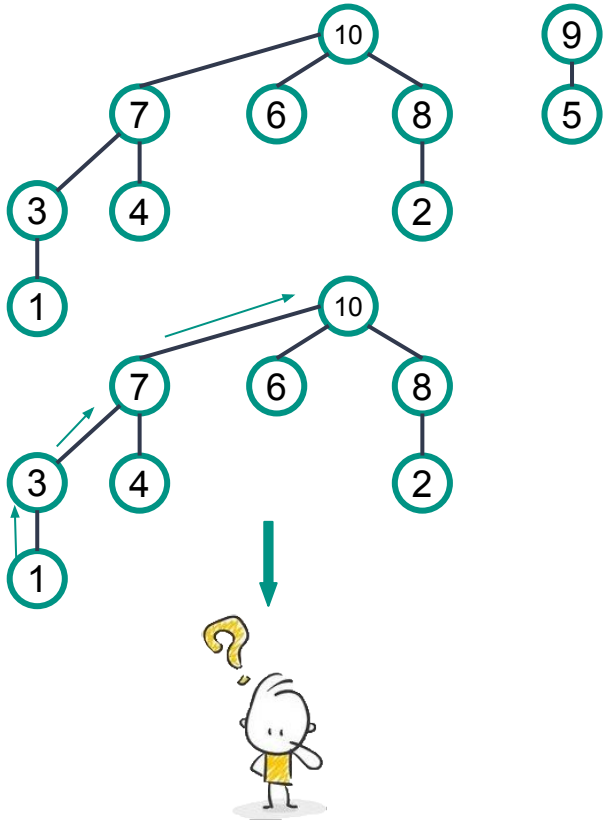
(4, 7)	(4, 6)
(2, 8)	**(1, 2)**
(6, 10)	(2, 3)
(2, 6)	(5, 10)
(5, 9)	(6, 7)
(1, 3)	(9 ,10)
(6, 8)	
(1, 4)	

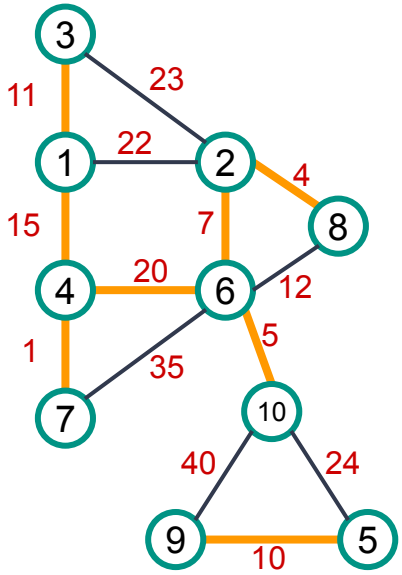
Muchia curentă
(1, 2):

Reprez(1): ⇒ 10 +
compresie de cale

!! h nu se modifică
(h[7] rămâne 2)

Pădurea de mulțimi disjuncte la pasul curent





Ordine muchii

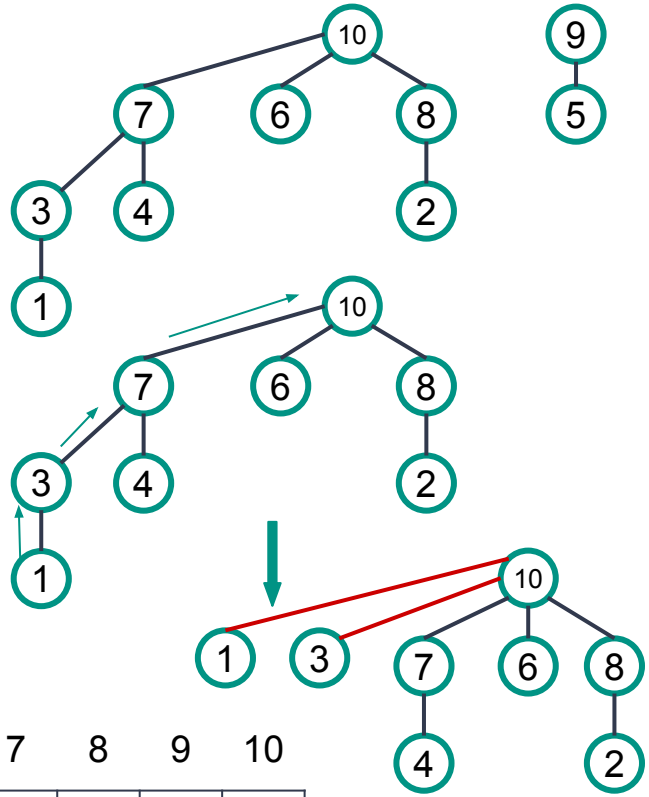
(4, 7)	(4, 6)
(2, 8)	**(1, 2)**
(6, 10)	(2, 3)
(2, 6)	(5, 10)
(5, 9)	(6, 7)
(1, 3)	(9 ,10)
(6, 8)	
(1, 4)	

Muchia curentă
(1, 2):

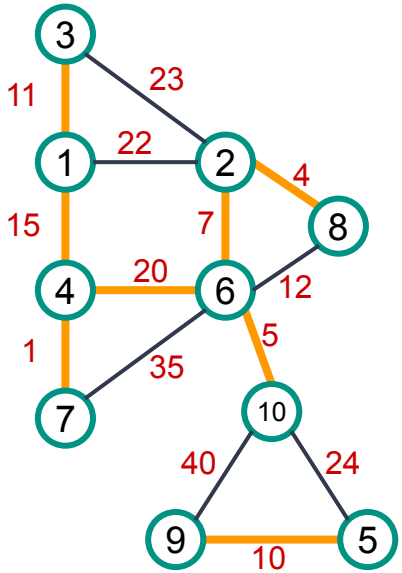
Reprez(1): ⇒ 10 +
compresie de cale

!! h nu se modifică
(h[7] rămâne 2)

Pădurea de mulțimi disjuncte la pasul curent



	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

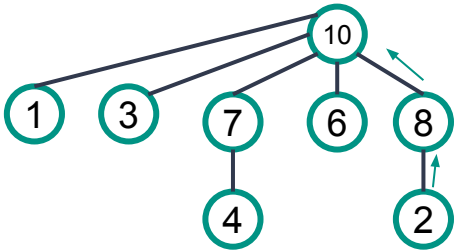


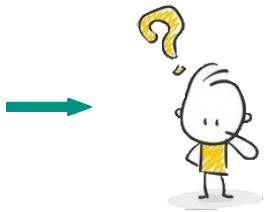
Ordine muchii

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

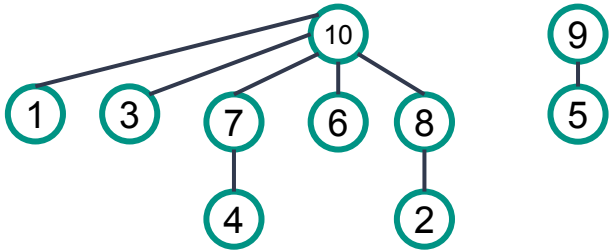
Muchia curentă
(1, 2):

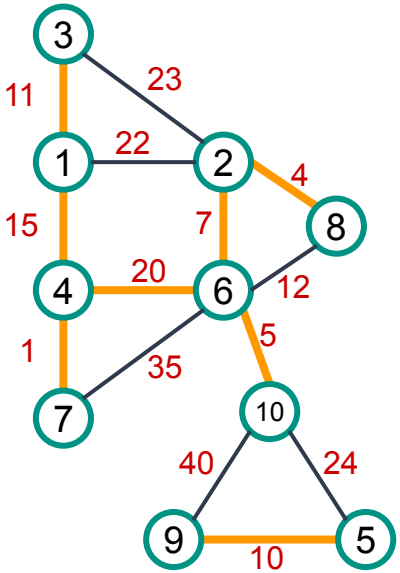
Reprez(2): ⇒ 10 +
compresie de cale





Pădurea de mulțimi disjuncte la pasul curent

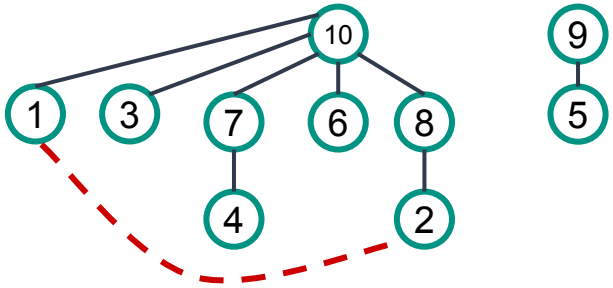




Ordine muchii

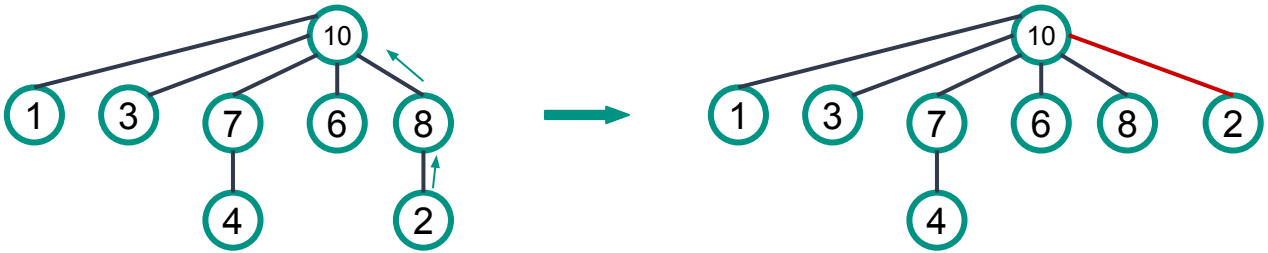
(4, 7)	(4, 6)
(2, 8)	**(1, 2)**
(6, 10)	(2, 3)
(2, 6)	(5, 10)
(5, 9)	(6, 7)
(1, 3)	(9 ,10)
(6, 8)	
(1, 4)	

Pădurea de mulțimi disjuncte la pasul curent

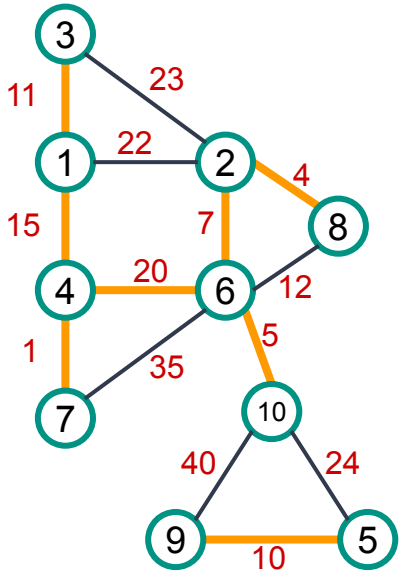


Muchia curentă
(1, 2):

Reprez(2): ⇒ 10 +
compresie de cale



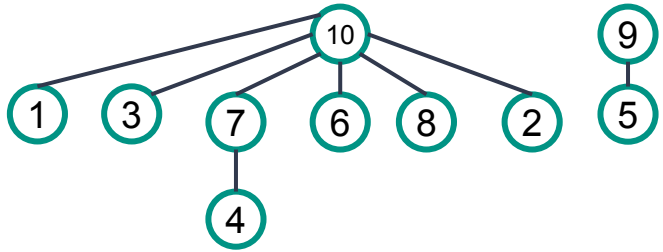
	1	2	3	4	5	6	7	8	9	10
tata	10	**10**	10	7	9	10	10	10	0	0
h	0	0	1	0	0	0	2	1	1	3



Ordine muchii

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

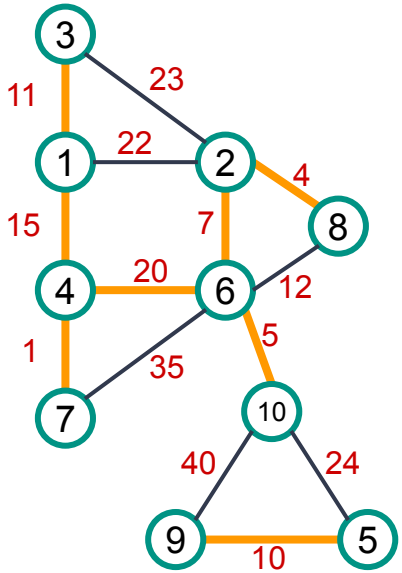
Pădurea de mulțimi disjuncte la pasul curent



Muchia curentă
(1, 2):

Reprez(1) = 10
Reprez(2) = 10 ⇒ nu este selectată

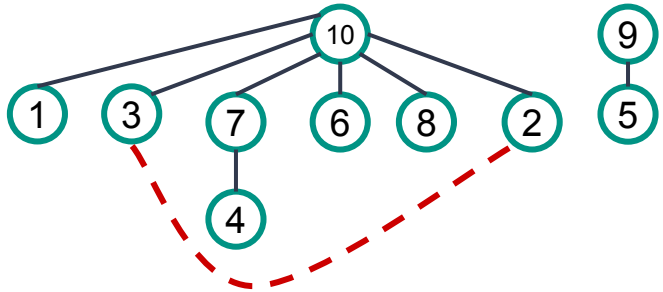
	1	2	3	4	5	6	7	8	9	10
tata	10	10	10	7	9	10	10	10	0	0
h	0	0	1	0	0	0	2	1	1	3



Ordine muchii

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

Pădurea de mulțimi disjuncte la pasul curent

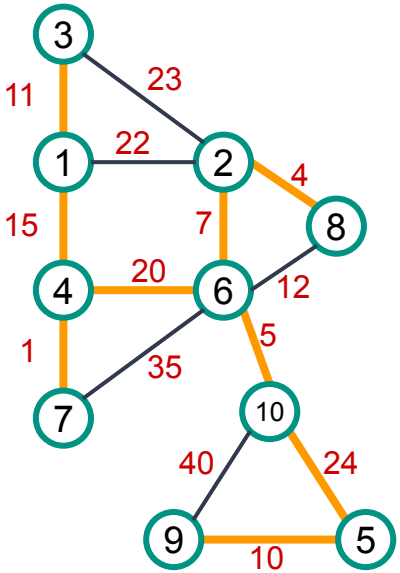


Muchia curentă
(2, 3):

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

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

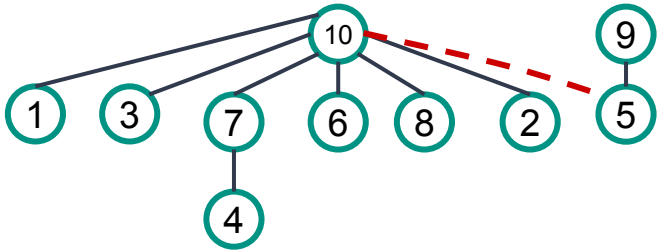
	1	2	3	4	5	6	7	8	9	10
tata	10	10	10	7	9	10	10	10	0	0
h	0	0	1	0	0	0	2	1	1	3



Ordine muchii

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

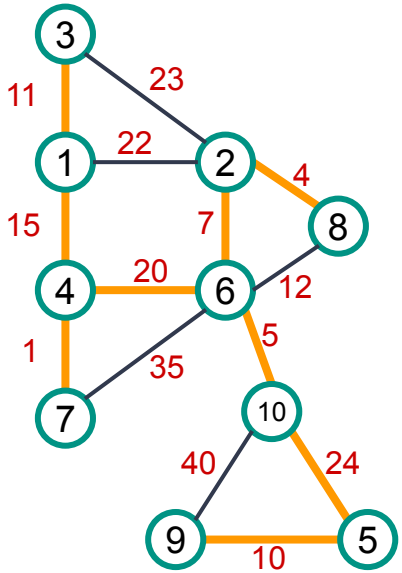
Pădurea de mulțimi disjuncte la pasul curent



Muchia curentă
(5, 10):
Reprez(5) ≠ Reprez(10)

Reuneste(5, 10)
reuniune ponderată

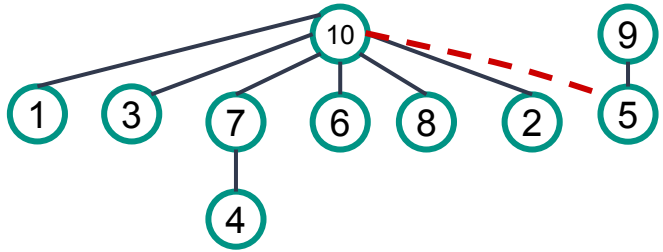
	1	2	3	4	5	6	7	8	9	10
tata	10	10	10	7	9	10	10	10	0	0
h	0	0	1	0	0	0	2	1	1	3



Ordine muchii

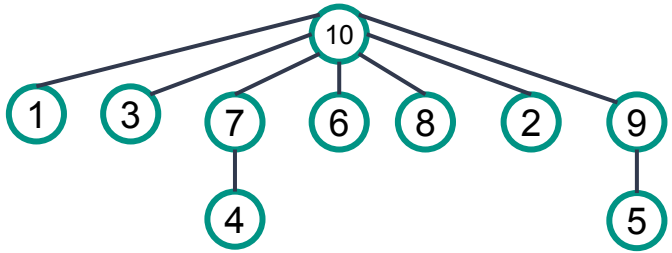
(4, 7)	(4, 6)
(2, 8)	(1, 2)
(6, 10)	(2, 3)
(2, 6)	**(5, 10)**
(5, 9)	(6, 7)
(1, 3)	(9 ,10)
(6, 8)	
(1, 4)	

Pădurea de mulțimi disjuncte la pasul curent

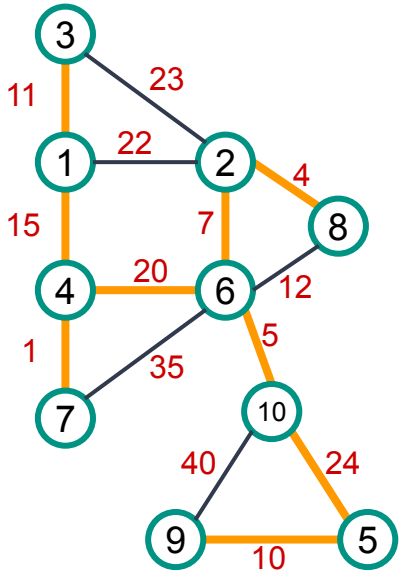


Muchia curentă
(5, 10):
Reprez(5) ≠ Reprez(10)

Reuneste(5, 10)
reuniune ponderată
h[Reprez(5)] = h[9] = 1
< h[Reprez(10)] = h[10] = 3



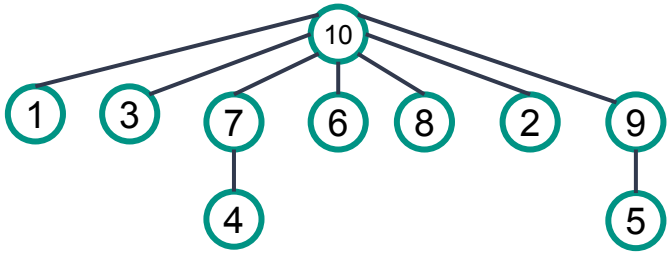
	1	2	3	4	5	6	7	8	9	10
tata	10	10	10	7	9	10	10	10	**10**	0
h	0	0	1	0	0	0	2	1	1	3



Ordine muchii

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

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)

!