**Arbori**

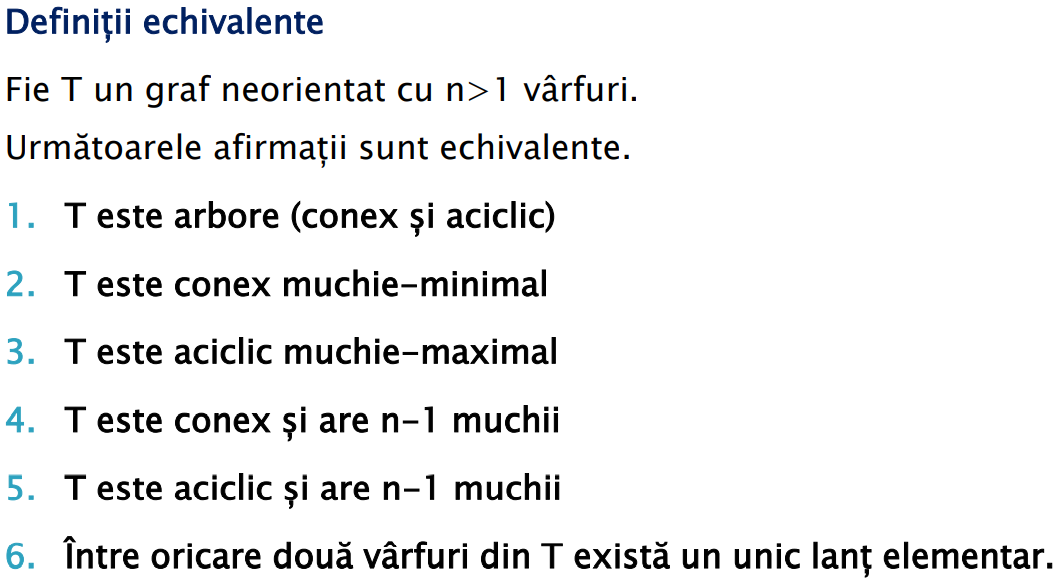
**Leme**

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

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

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

4. Fie G un graf neorientat conex și C un ciclu in G. Fie e ∈ E(C) o muchie din ciclul C. Atunci G-e este tot un graf conex.

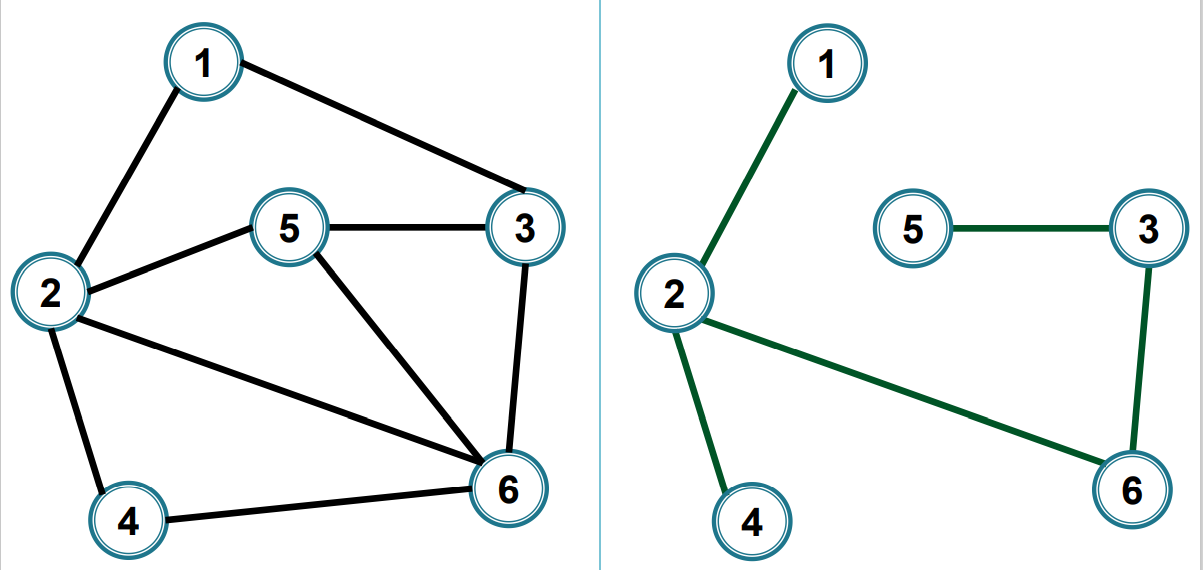


Graf complet cu n noduri => nr de apcm e n(n-2)

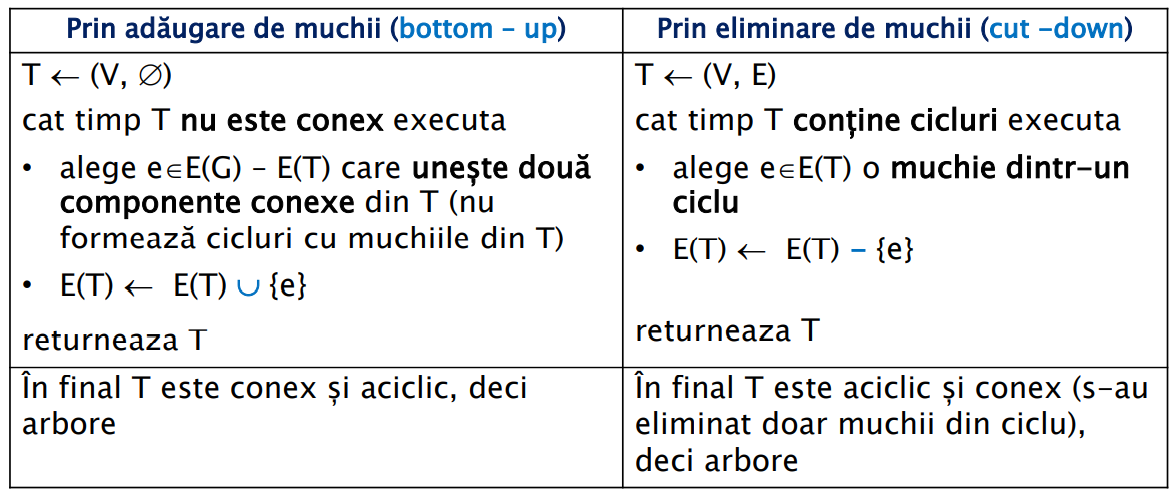
**Arbori partiali ai unui graf**

Proprietate:

**Orice graf neorientat conex conține cel putin un arbore parțial** (un graf parțial care este arbore).

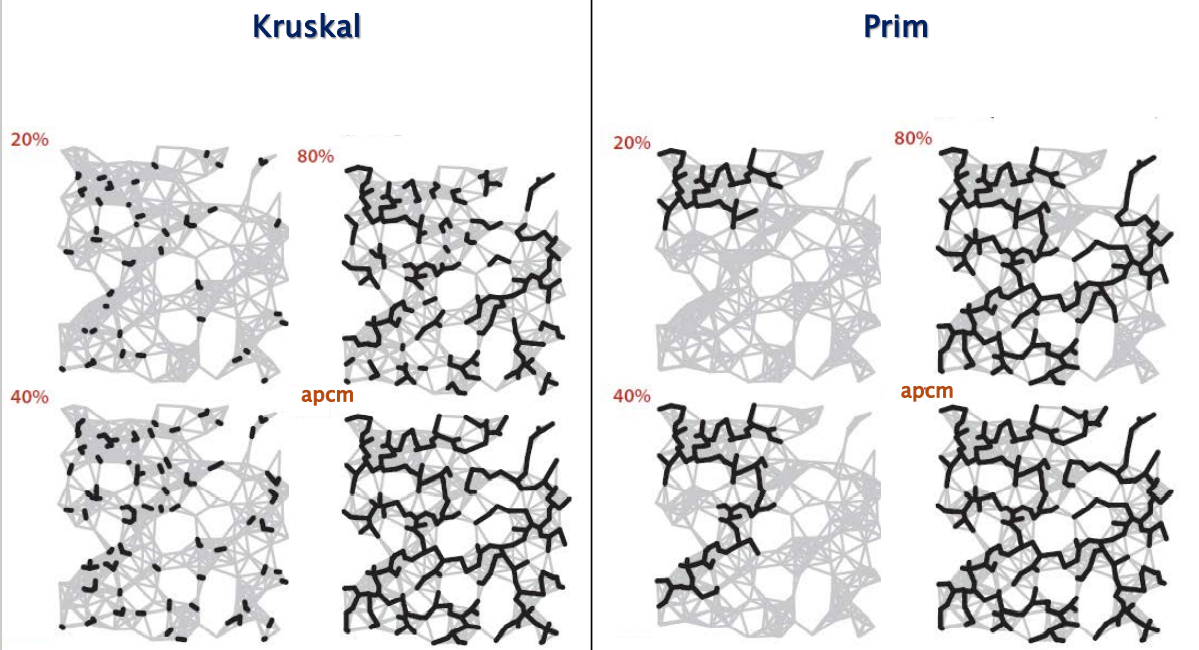


Exista două tipuri de algoritmi de construcție a unui arbore parțial al unui graf conex:

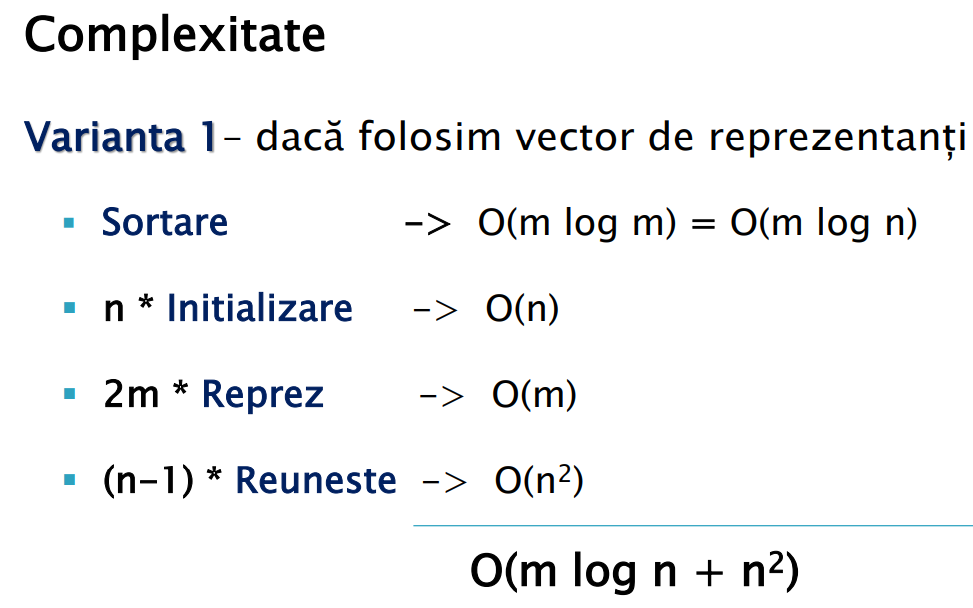
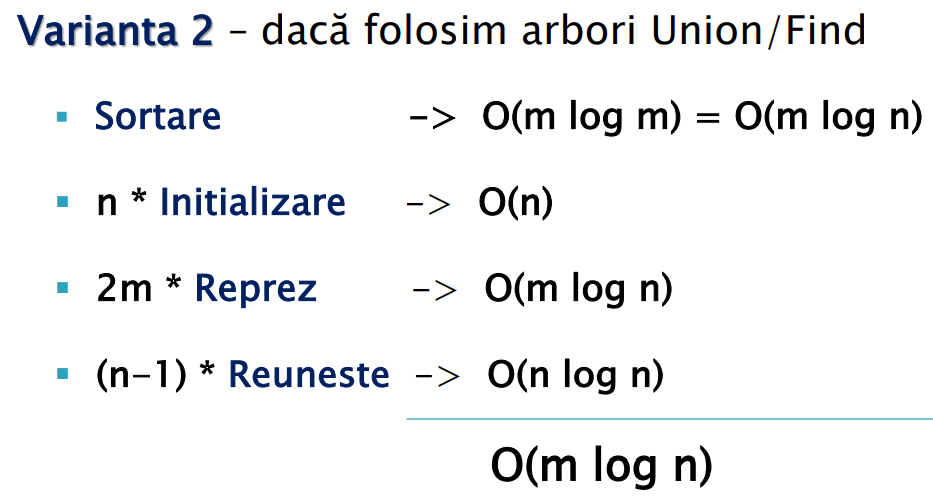


**Arbore parțial de cost minim (APCM) al unui graf conex ponderat**

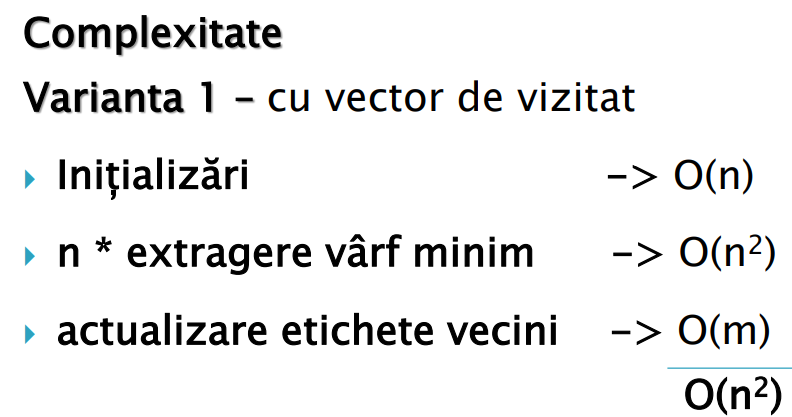
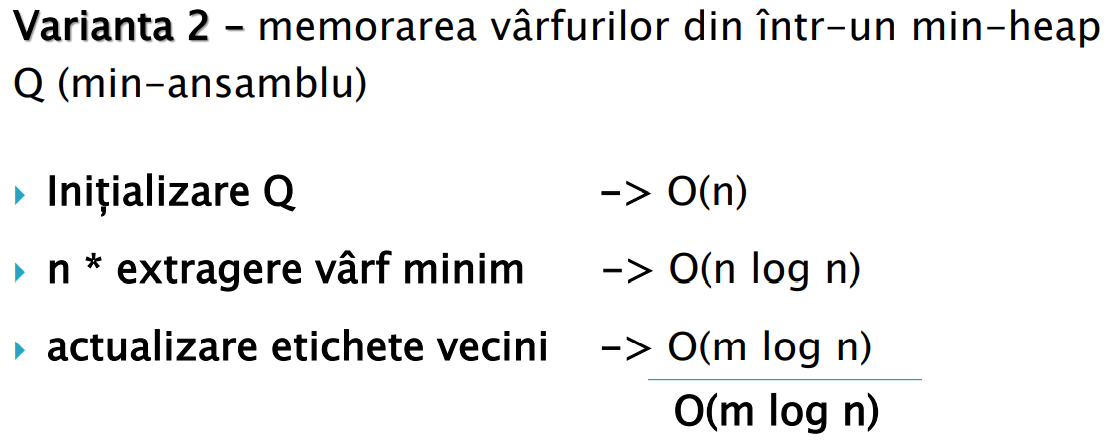
* **Kruskal**
* **Prim**
* **Boruvka**

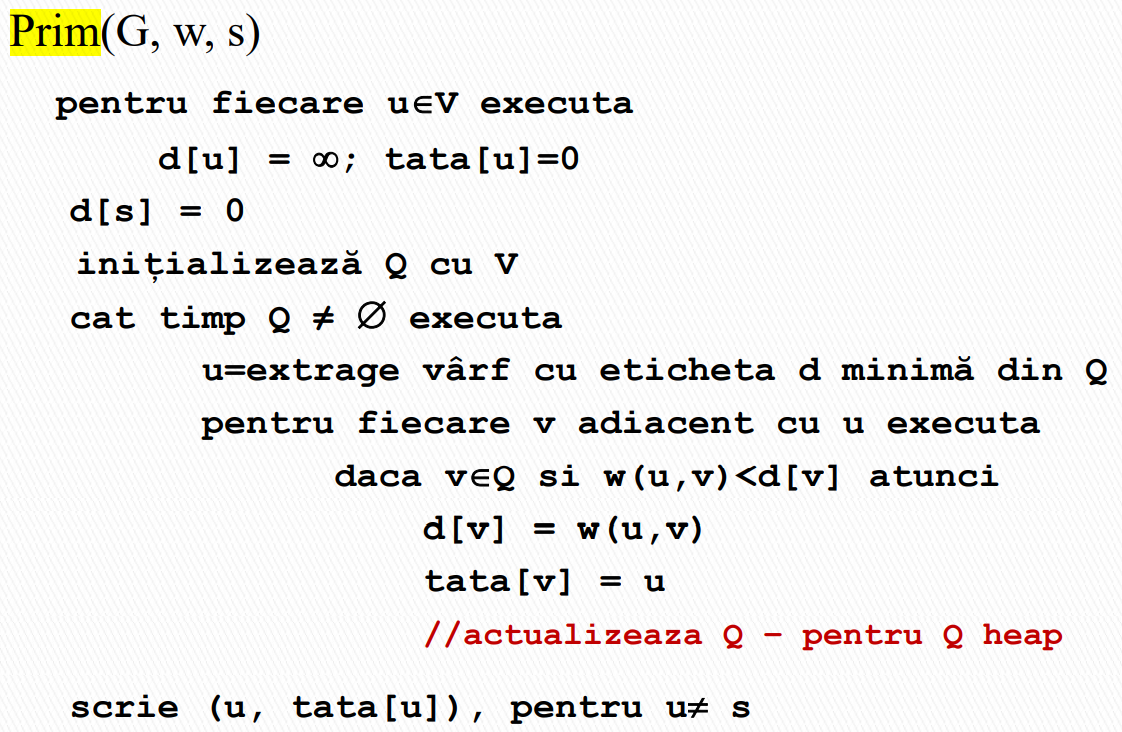


* **Kruskal**

* **Prim**



Observație: Comparatie cu algoritmul Kruskal

Dacă graful este complet (spre exemplu dacă toate punctele se pot conecta și distanța dintre puncte este distanța euclidiană) m = n(n-1)/2 este de ordin n2 ⇒ O(n2) mai eficient

◦ Kruskal: O(m log n).

◦ Prim: O(n2).

𝑛 − 1 ≤ 𝑚 ≤ 𝑛2. Daca 𝑒 ≈ 𝑛2  algoritmul **Kruskal** este asimptotic **mai slab** **decat** algoritmul **Prim**.

* **Boruvka** -> O(m log n) (seamana cu Prim, dar poate produce un apcm diferit)

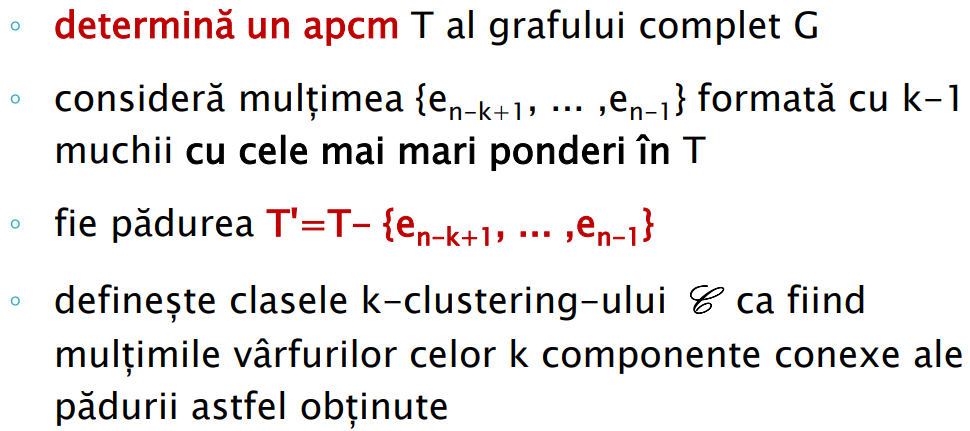
= [greedy algorithm](https://en.wikipedia.org/wiki/Greedy_algorithm) for finding a [minimum spanning tree](https://en.wikipedia.org/wiki/Minimum_spanning_tree) in a graph, or a minimum spanning forest in the case of a graph that is not connected. Input: graf neorientat ponderat

* se init toate vf ca fiind comp conexe
* init apcm ca fiind vid
* cat timp exista mai mult de o comp conexa
  + adauga muchia de cost min care uneste aceasta comp cu alta (daca muchia nu e deja adaugata)
* return acm

**Aplicatii Clustering**:

n obiecte (noduri), muchiile reprez distanta dintr ele, vrem k clustere

Echivalent cu:



* vrem k comp conexe (asa punem stop in padurea din alg lui Kruskal)
* nu merg pana la n-1 ca la Kruskal, ci pana la n-k
* La un pas determinăm cele mai asemănătoare (apropiate) două obiecte aflate în clase diferite (cu distanţa cea mai mică între ele) şi unim clasele lor
* muchiile ramase vor fi distante intre clustere (muchiile cu costul cel mai mare)

