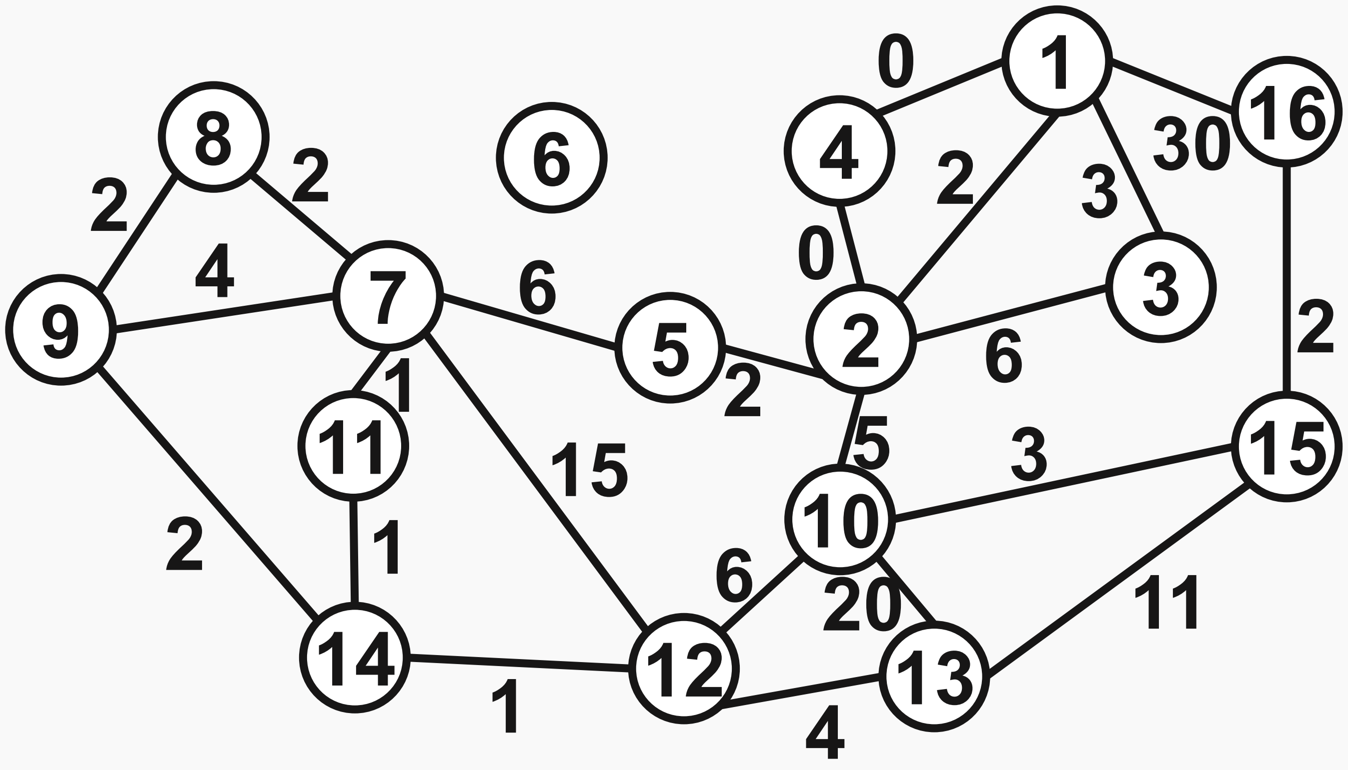
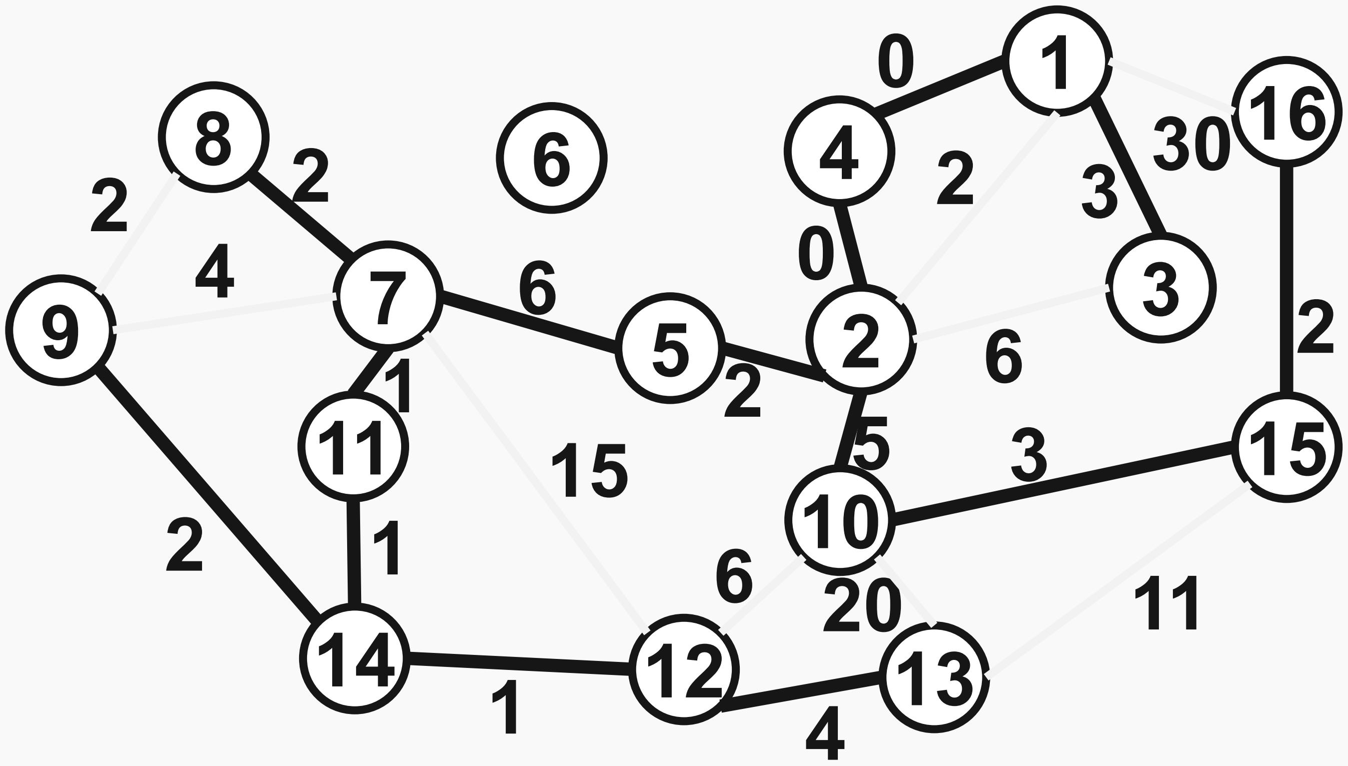
Structuri de date și algoritmi
Grafuri – Acoperire și Capacitate

Lect. Dr. Ing. Cristian Chilipirea – cristian.chilipirea@mta.ro

Arbori minimi de acoperire

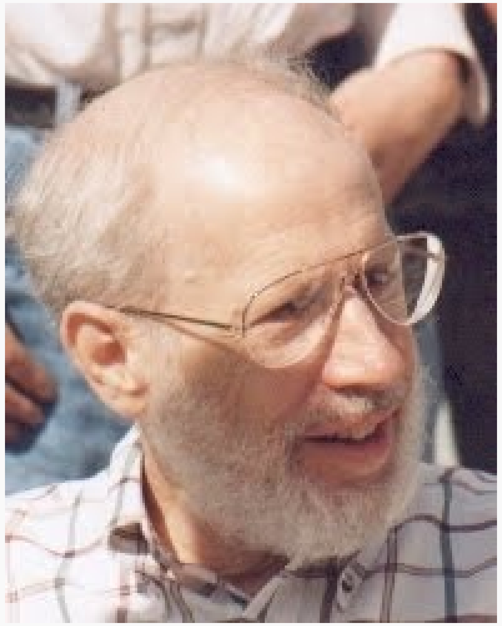


Arbori minimi de acoperire

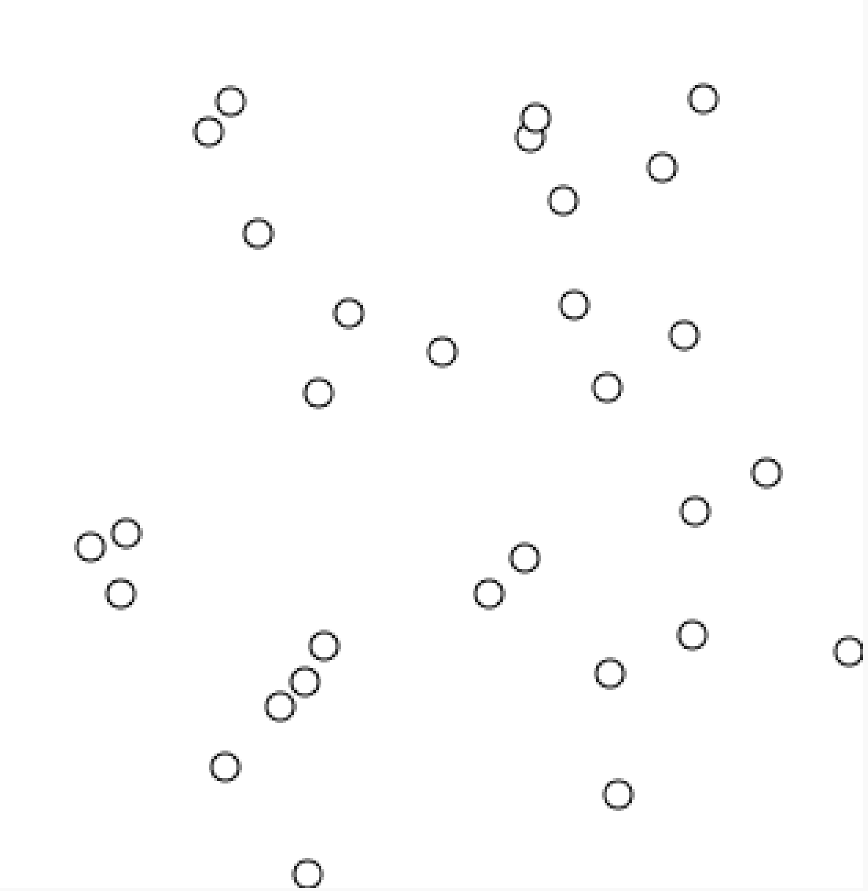


Algoritmul lui Kruskal (1956)

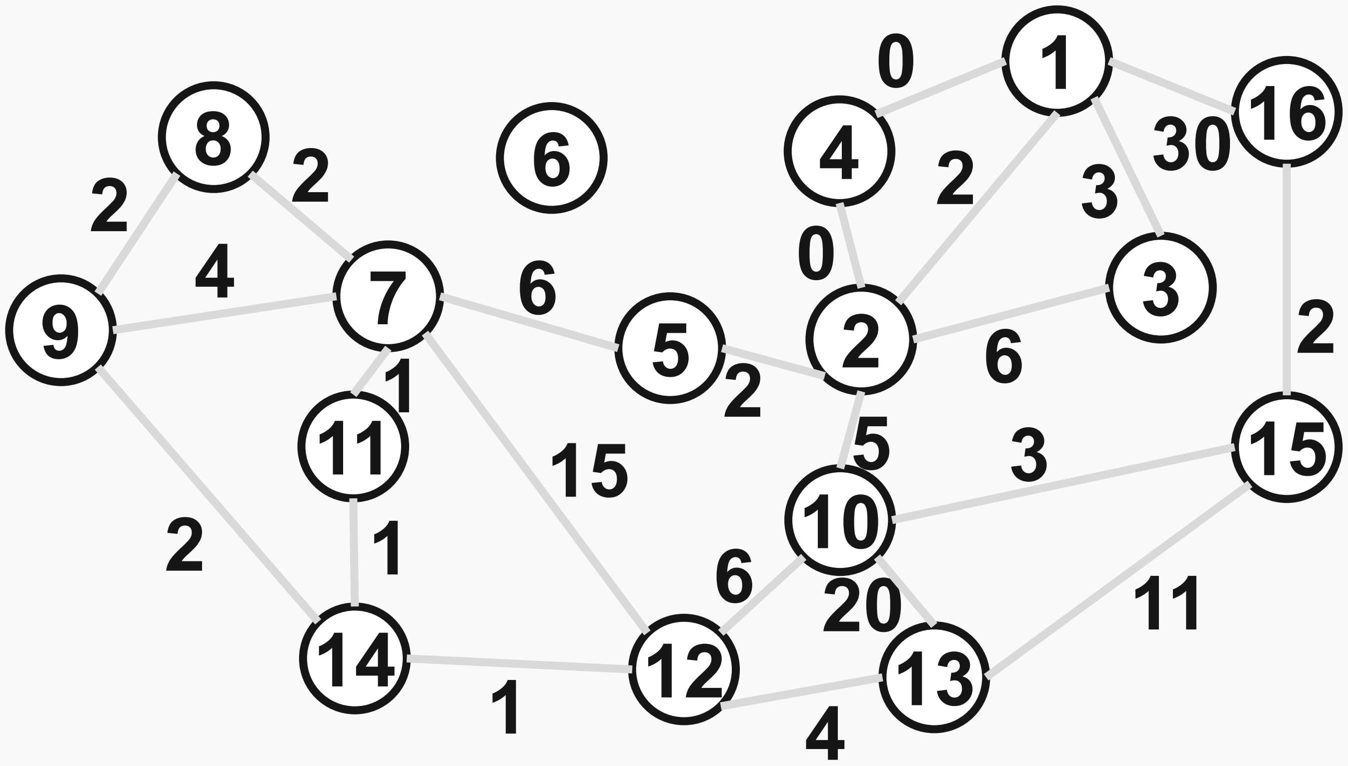
```
tree Kruskal(G) {
    sort(G.E); // sort by weight
    A = {};
    for each (node in G.V)
        Make_set(node);
    for each ((u, v) in G.E) {
        if (Find_set(u) != Find_set(v)) {
            A = A ∪ {(u, v)};
            Union(Find_set(u), Find_set(v));
        }
    }
    return A;
}
```



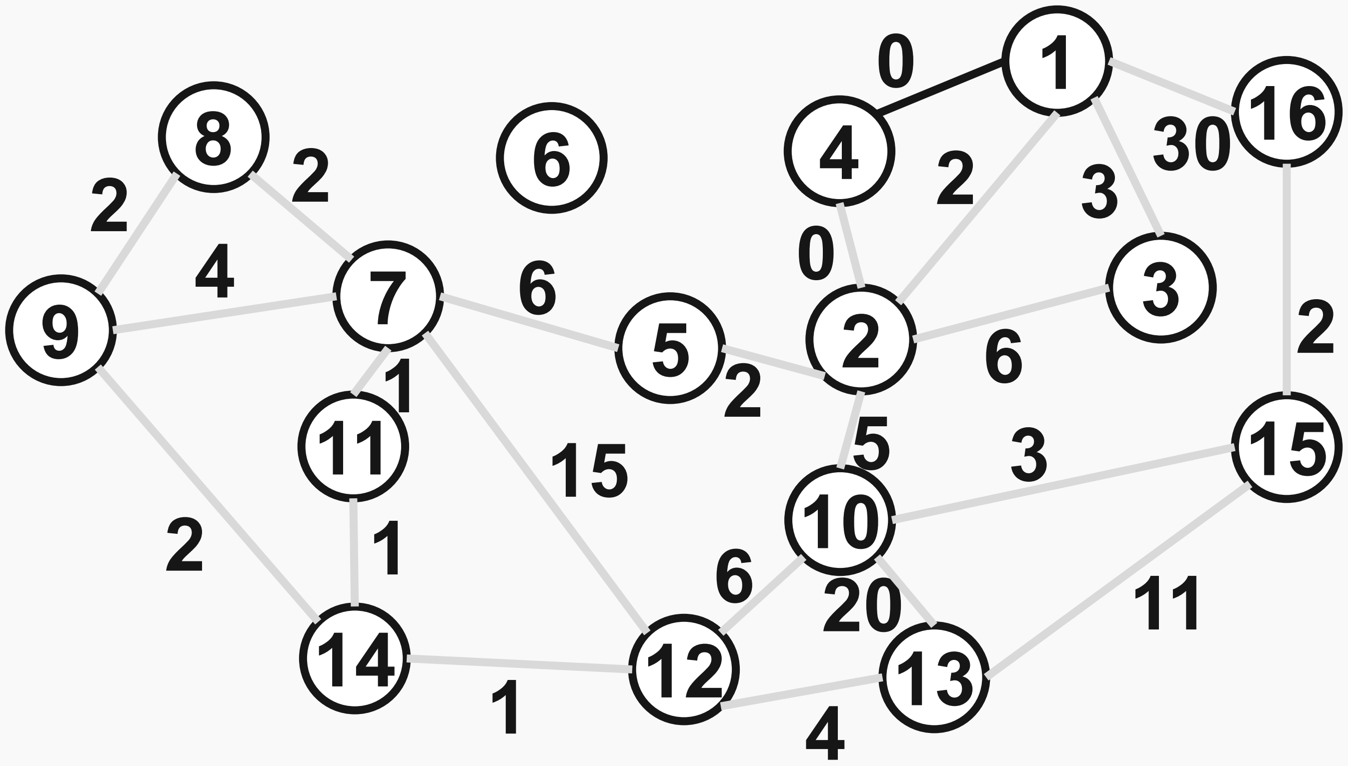
Algoritmul lui Kruskal



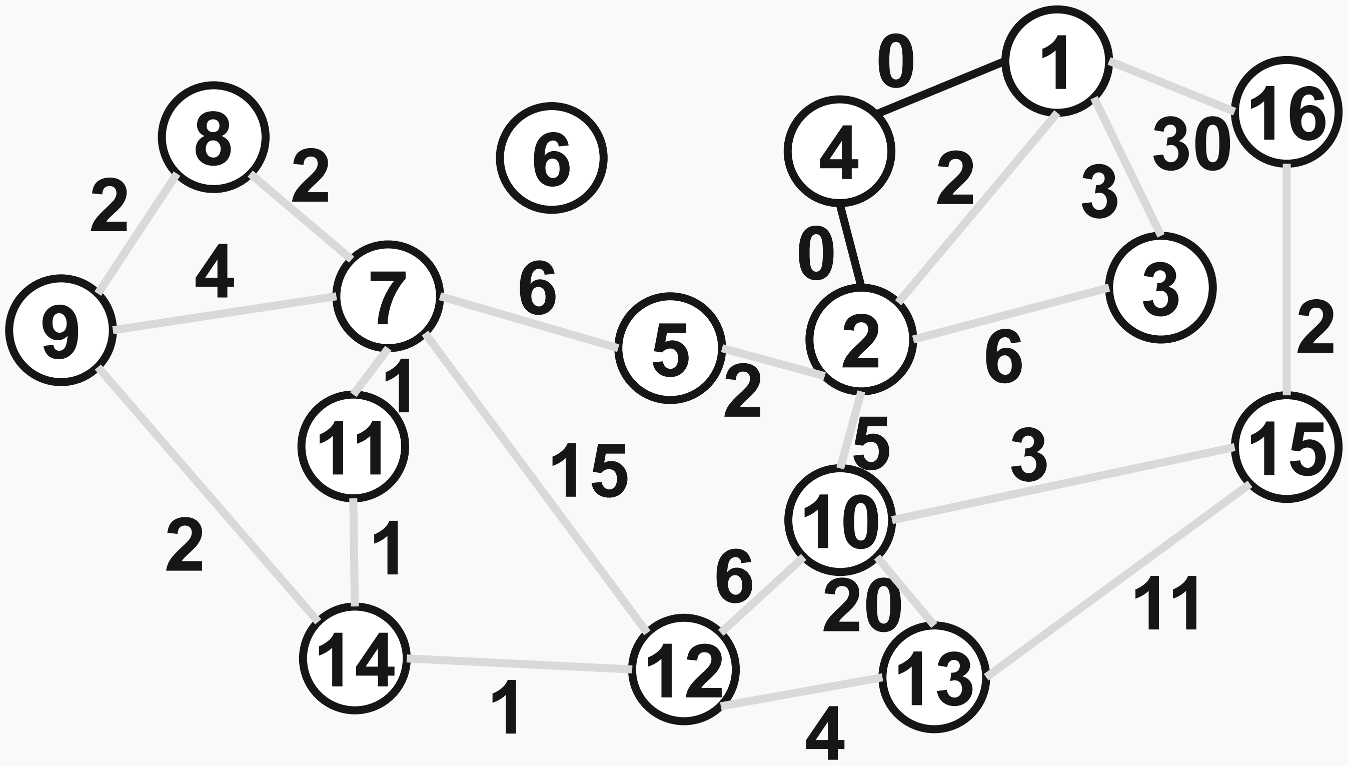
Algoritmul lui Kruskal



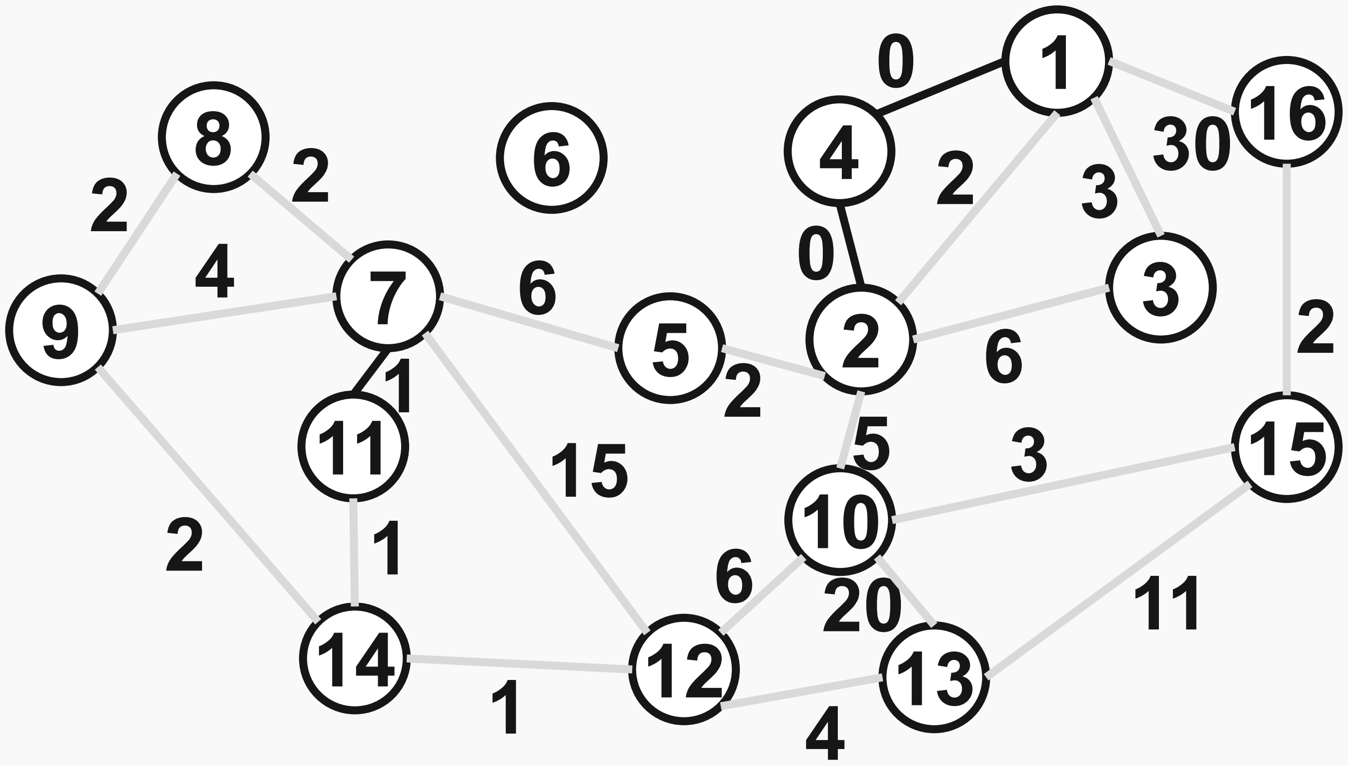
Algoritmul lui Kruskal



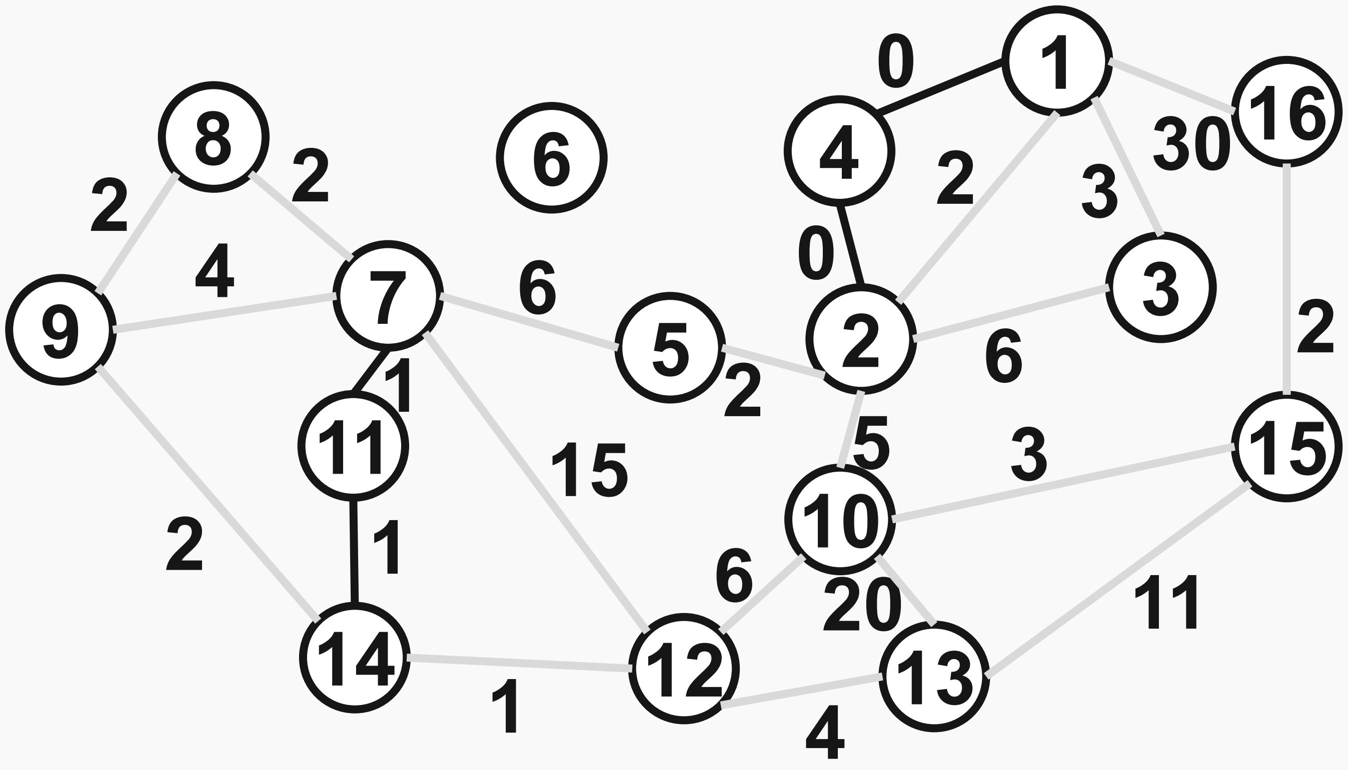
Algoritmul lui Kruskal



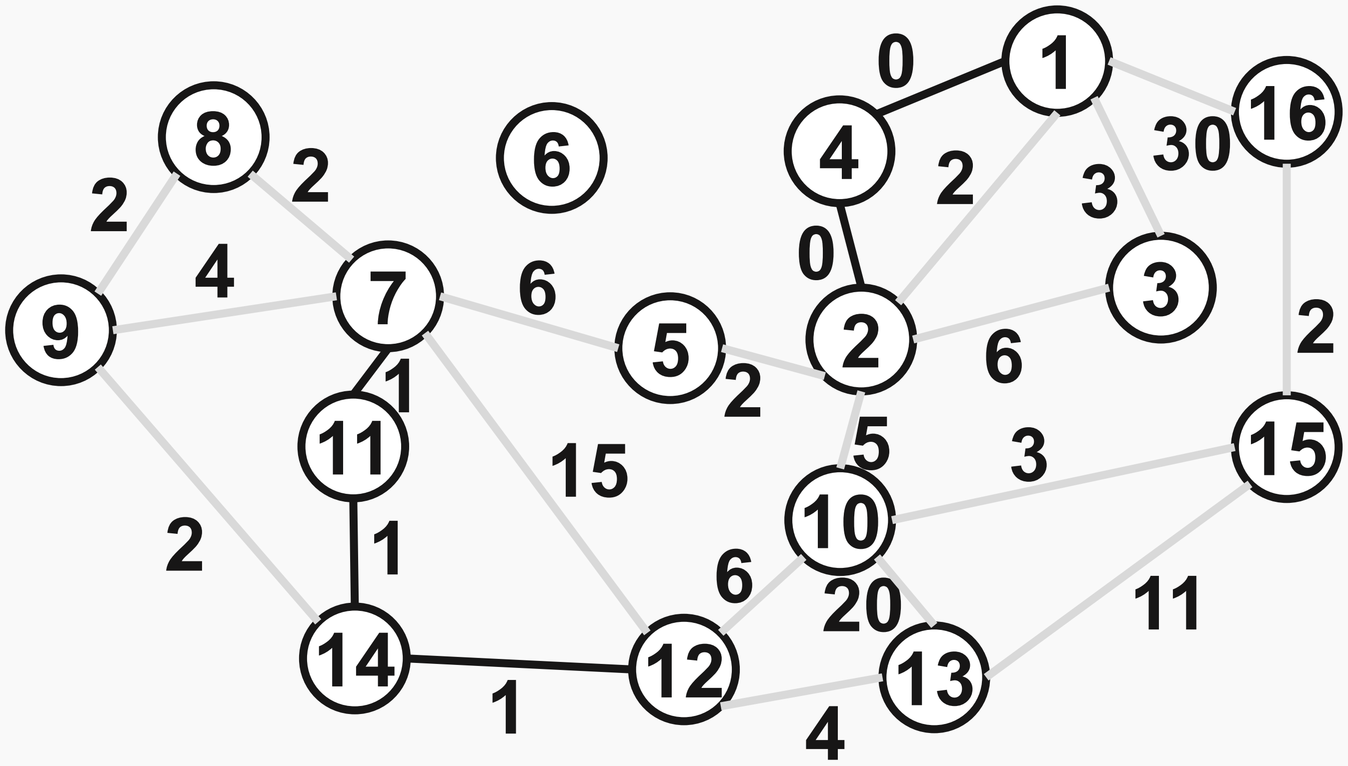
Algoritmul lui Kruskal



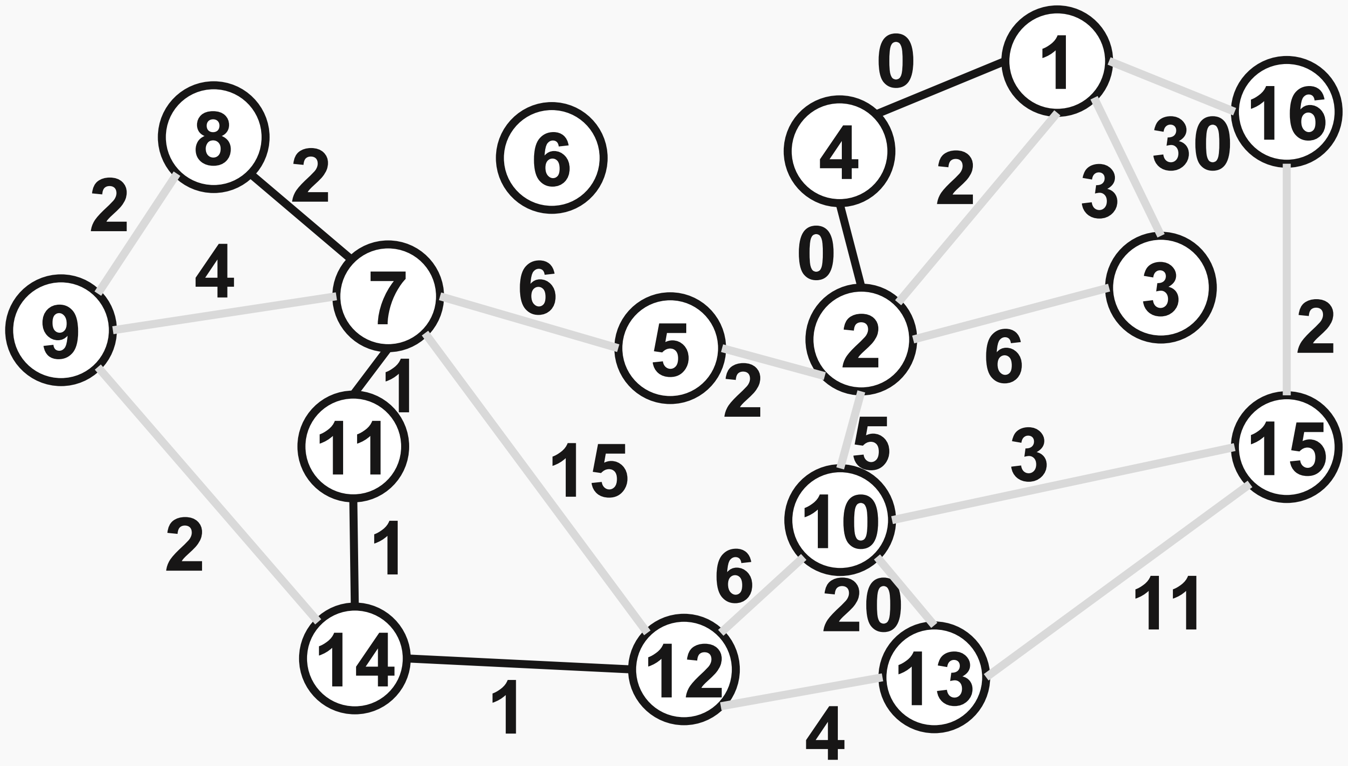
Algoritmul lui Kruskal



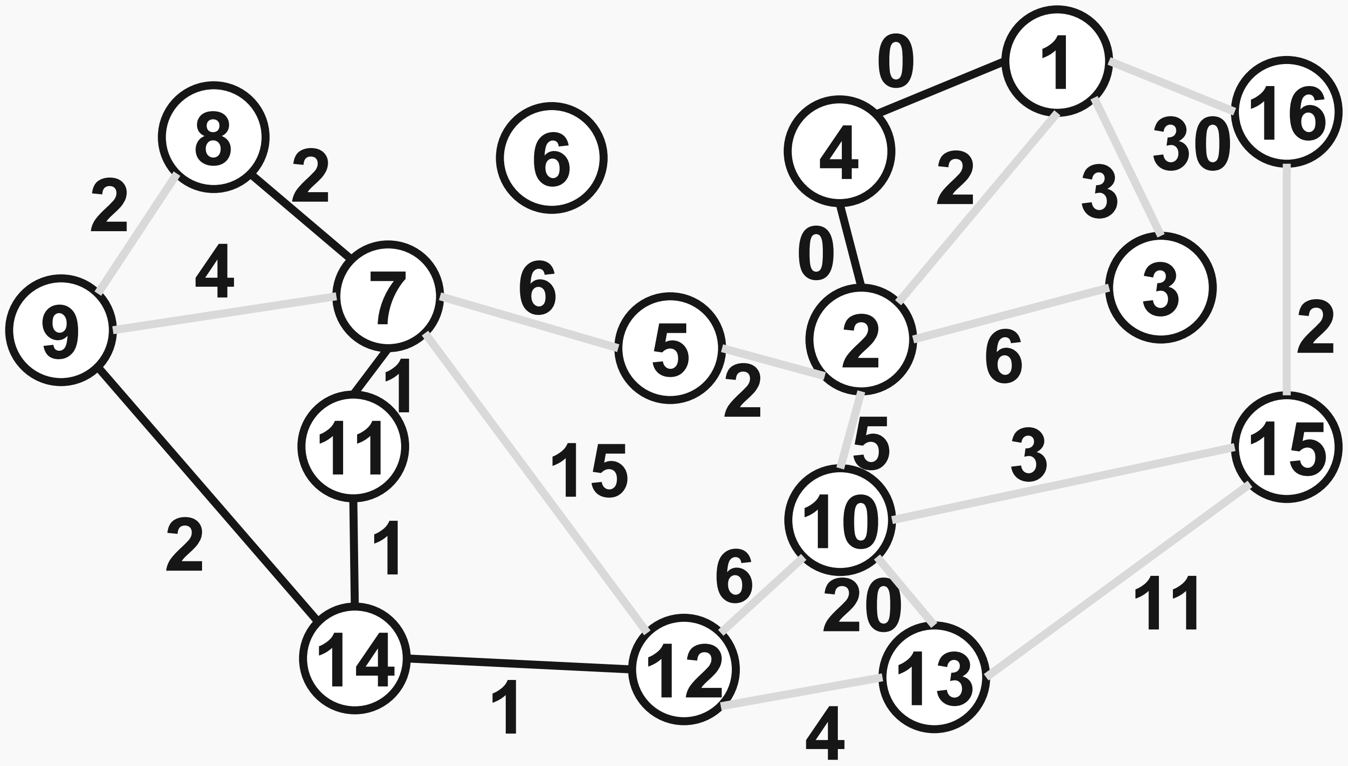
Algoritmul lui Kruskal



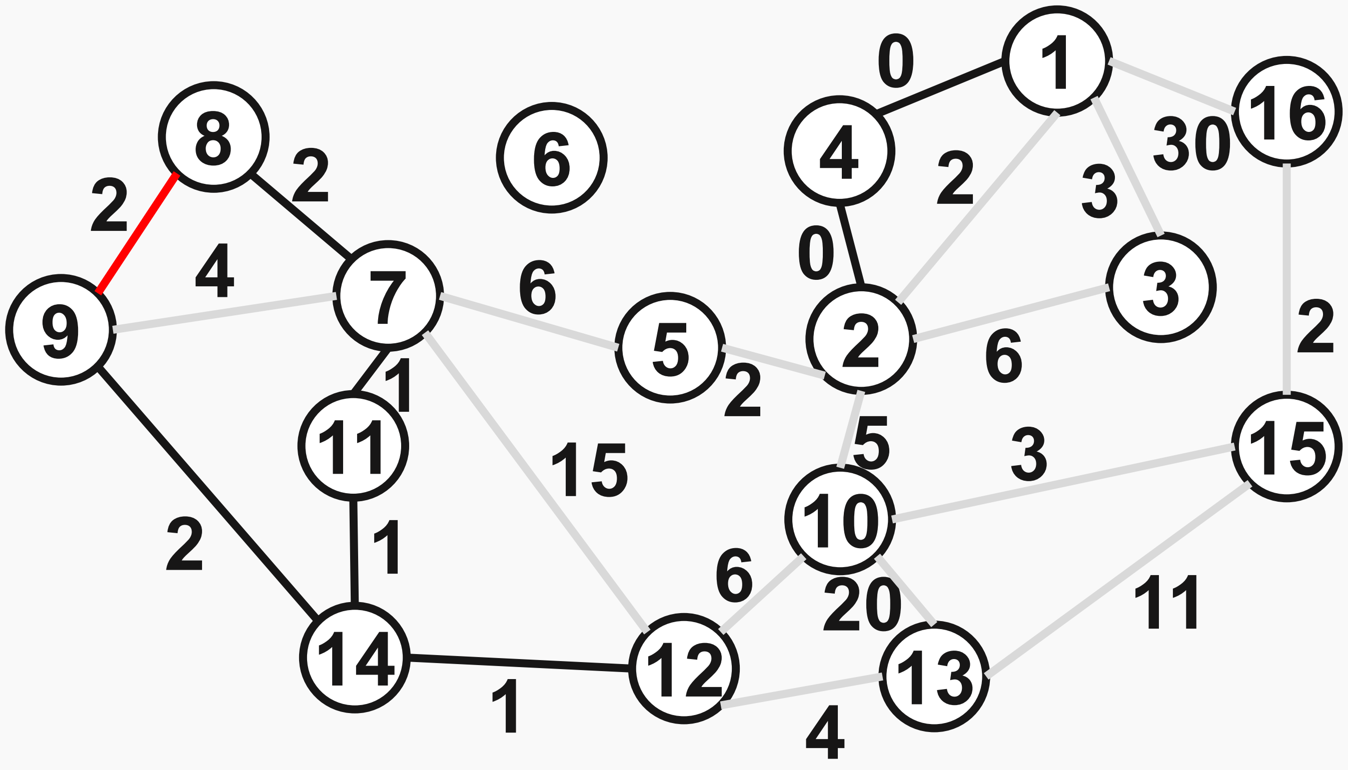
Algoritmul lui Kruskal



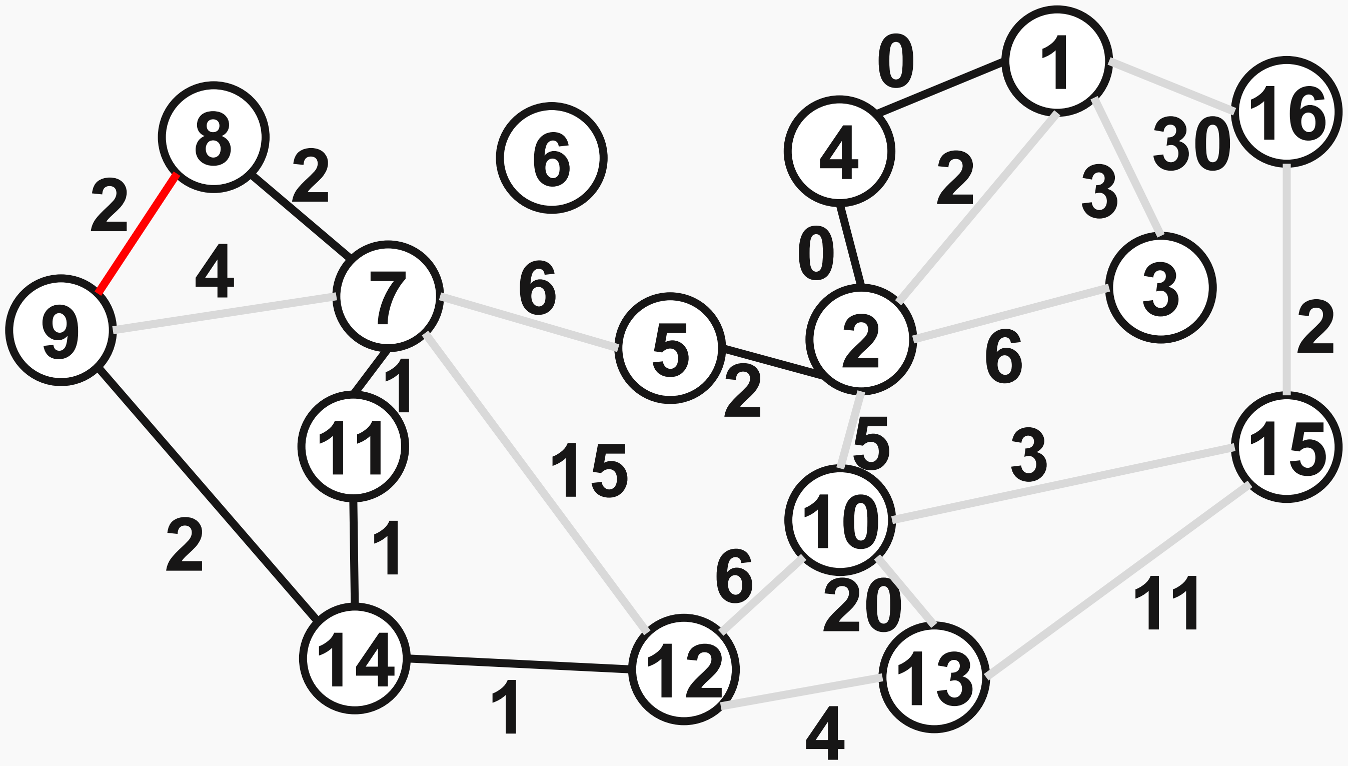
Algoritmul lui Kruskal



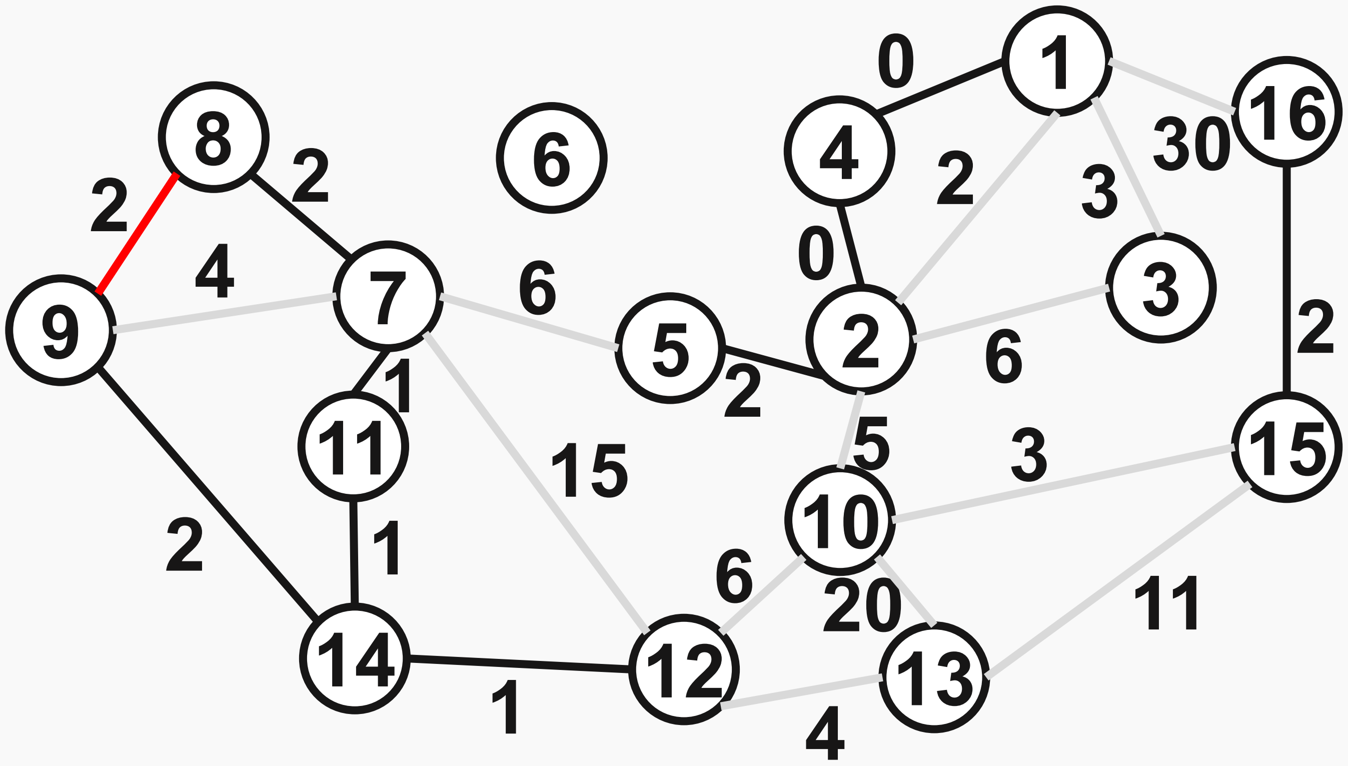
Algoritmul lui Kruskal



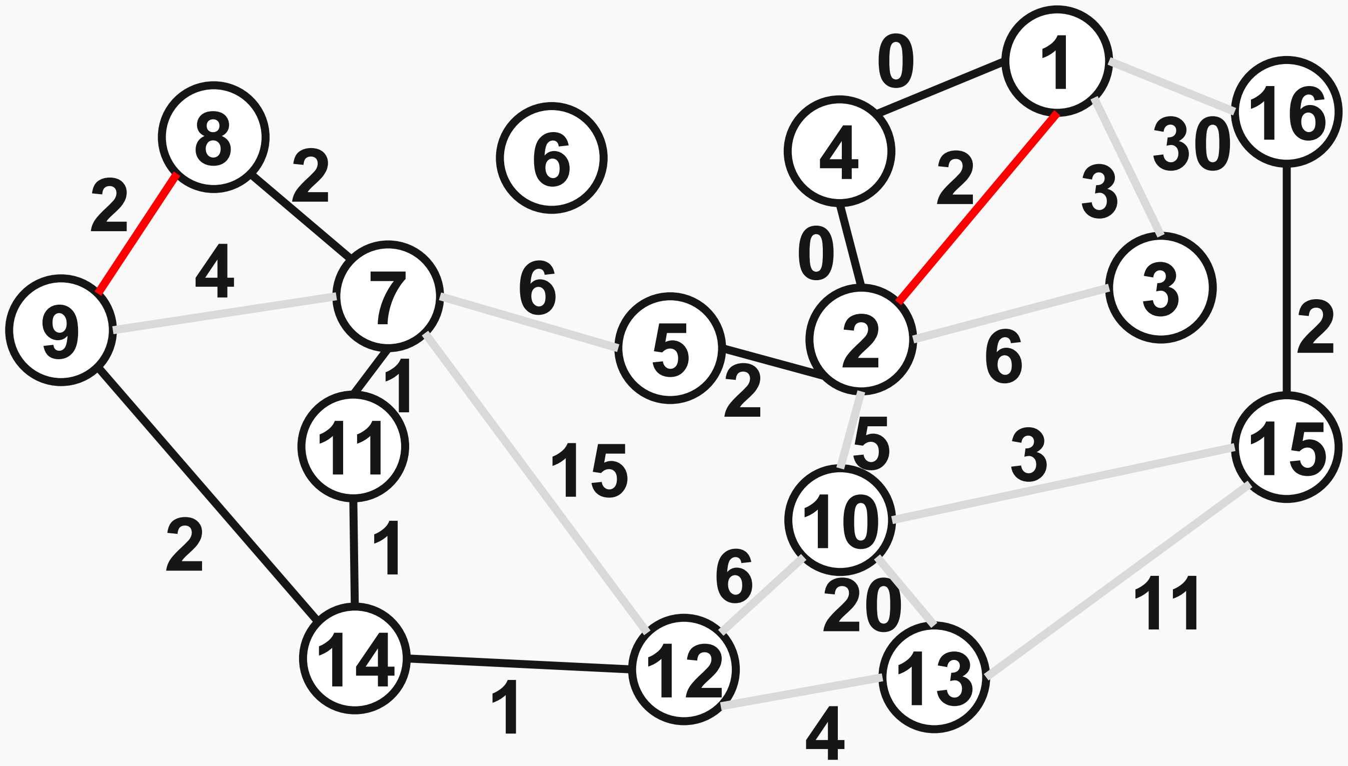
Algoritmul lui Kruskal



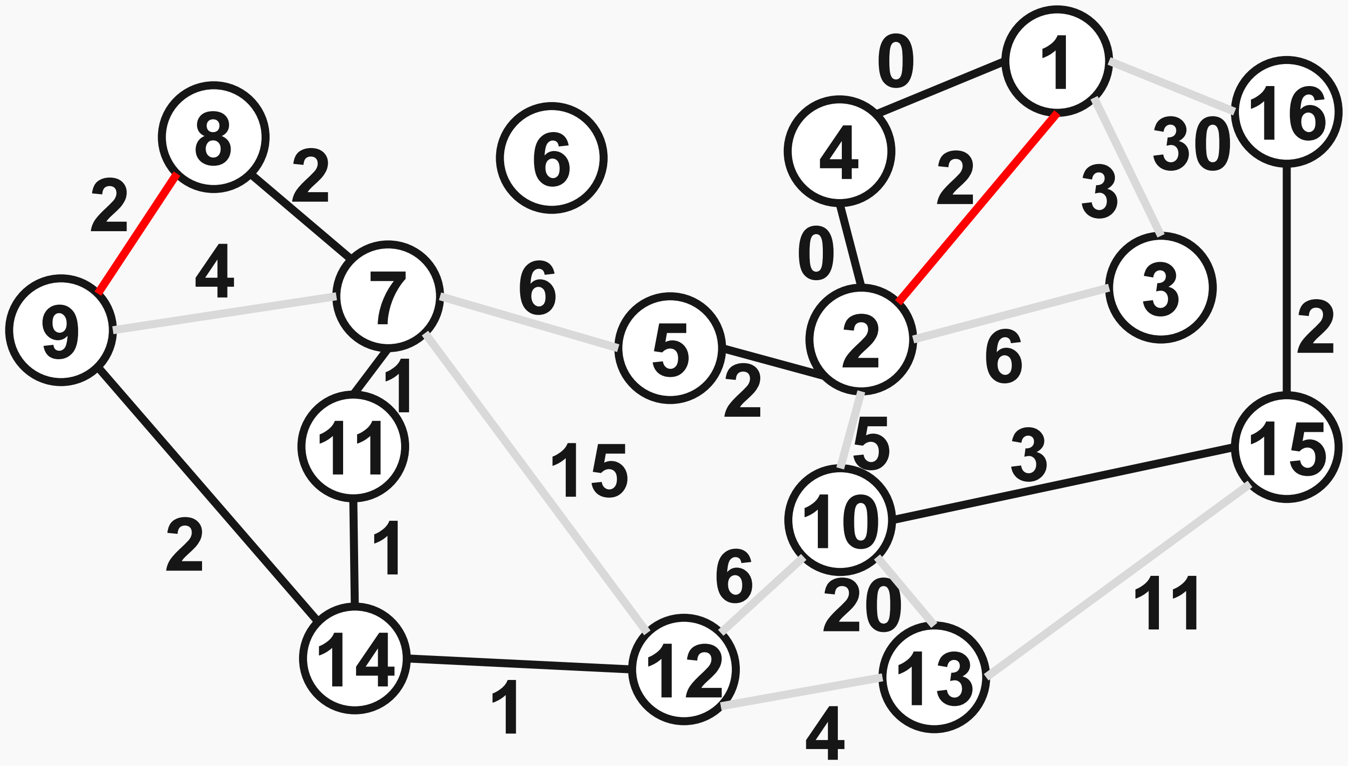
Algoritmul lui Kruskal



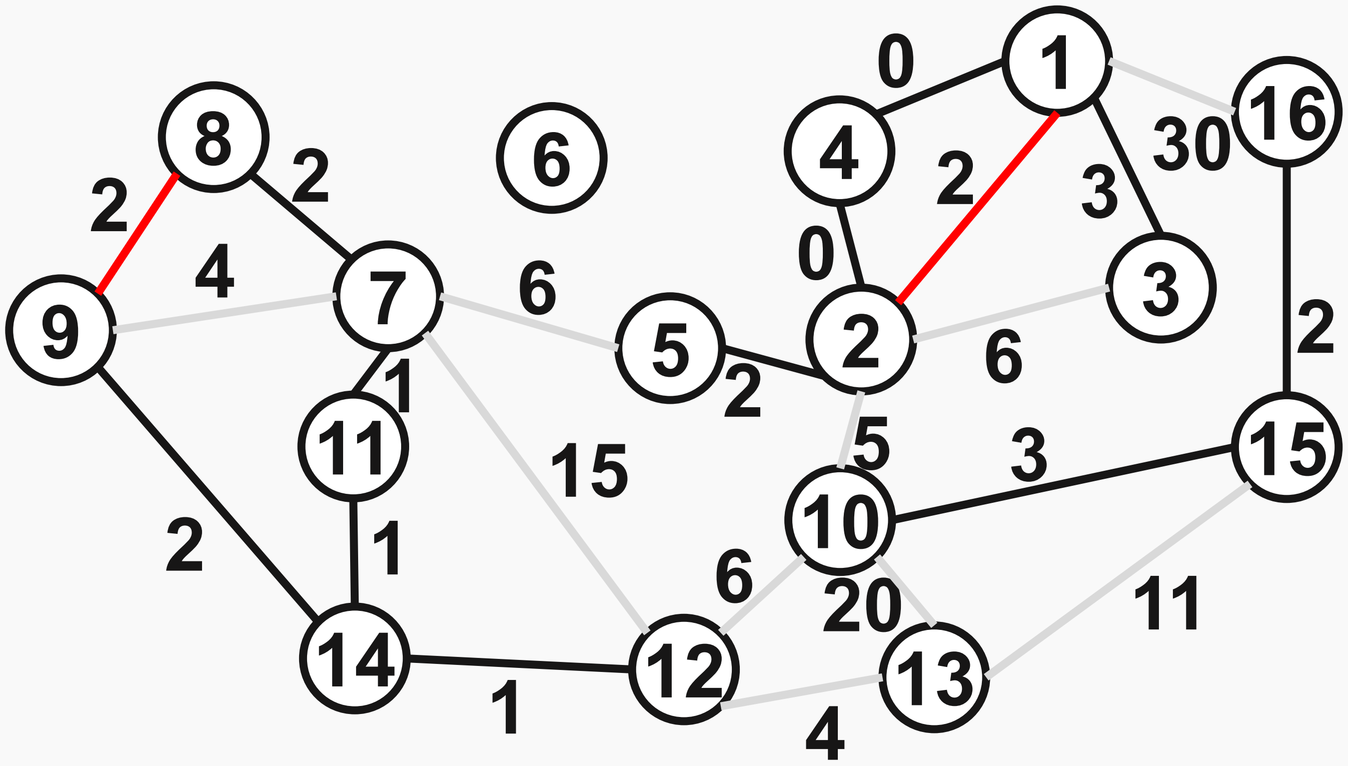
Algoritmul lui Kruskal



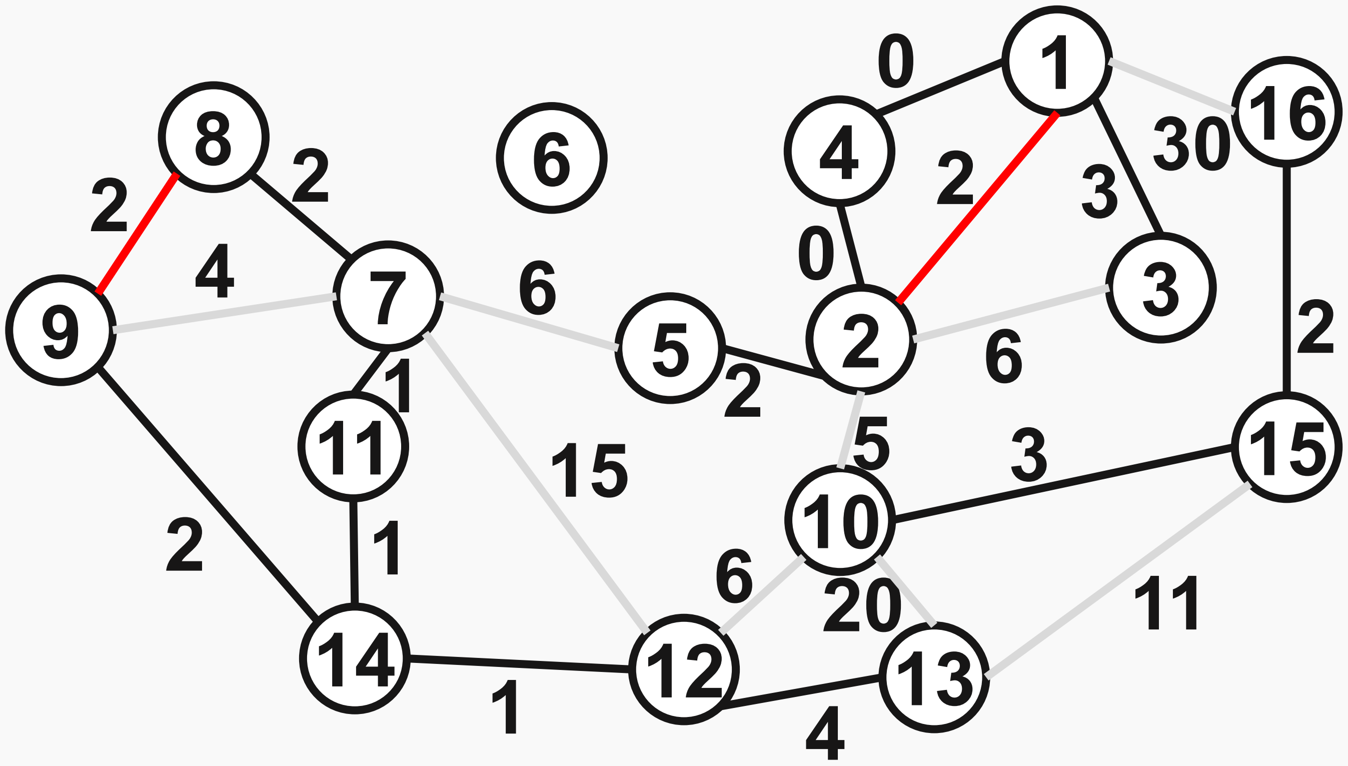
Algoritmul lui Kruskal



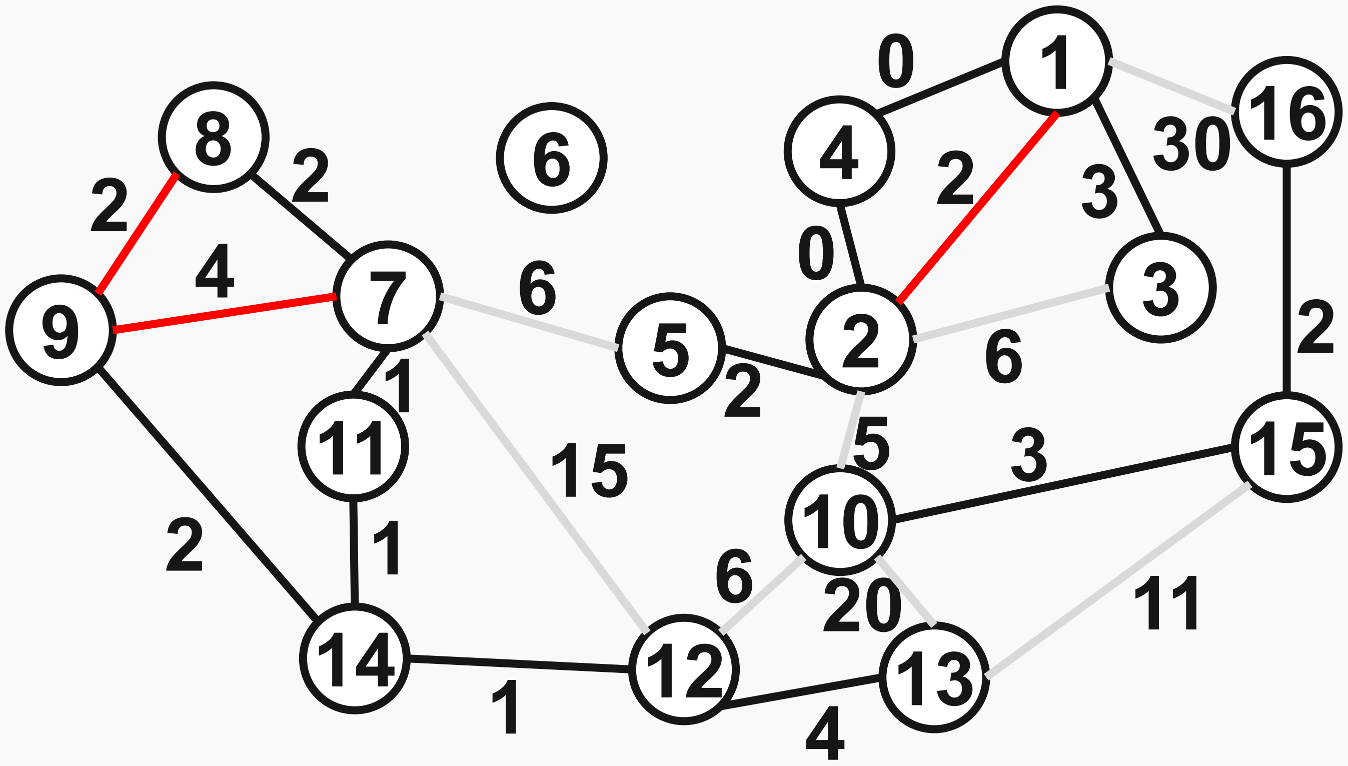
Algoritmul lui Kruskal



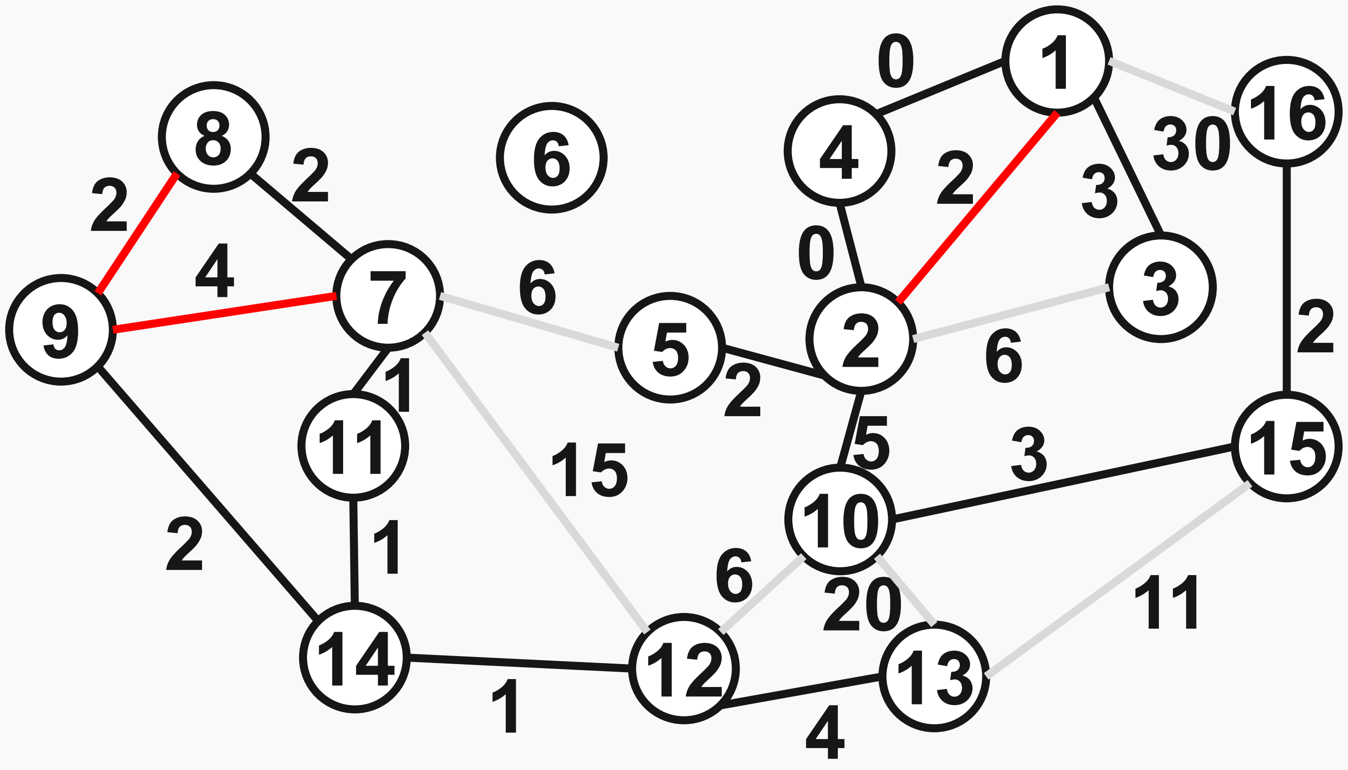
Algoritmul lui Kruskal



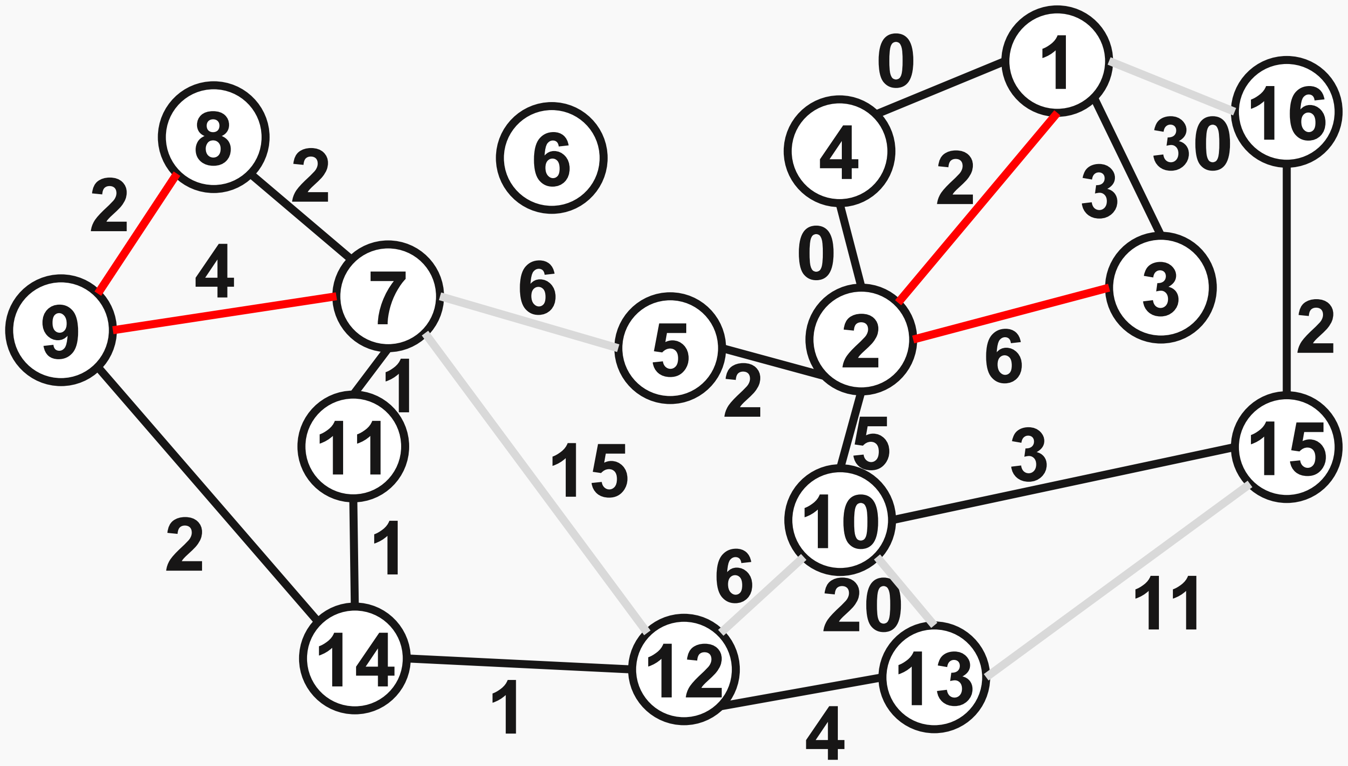
Algoritmul lui Kruskal



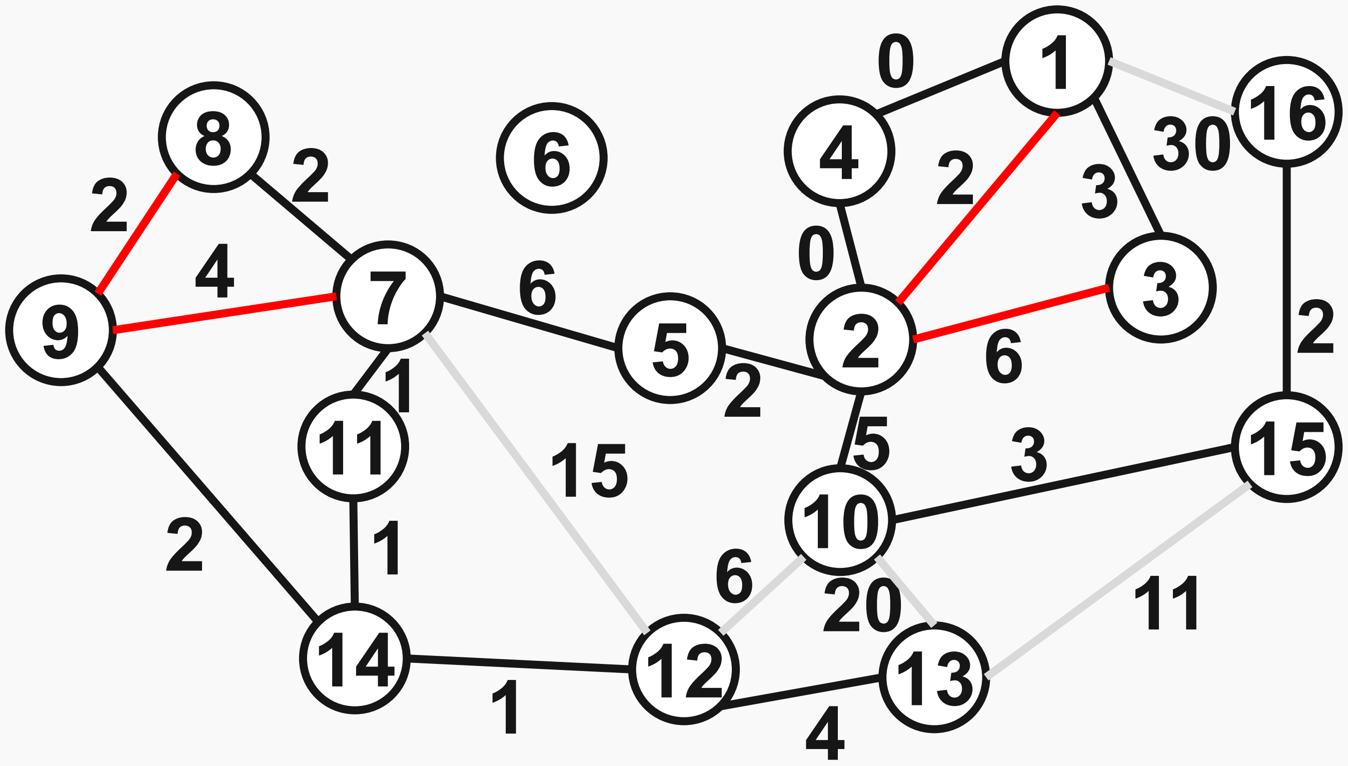
Algoritmul lui Kruskal



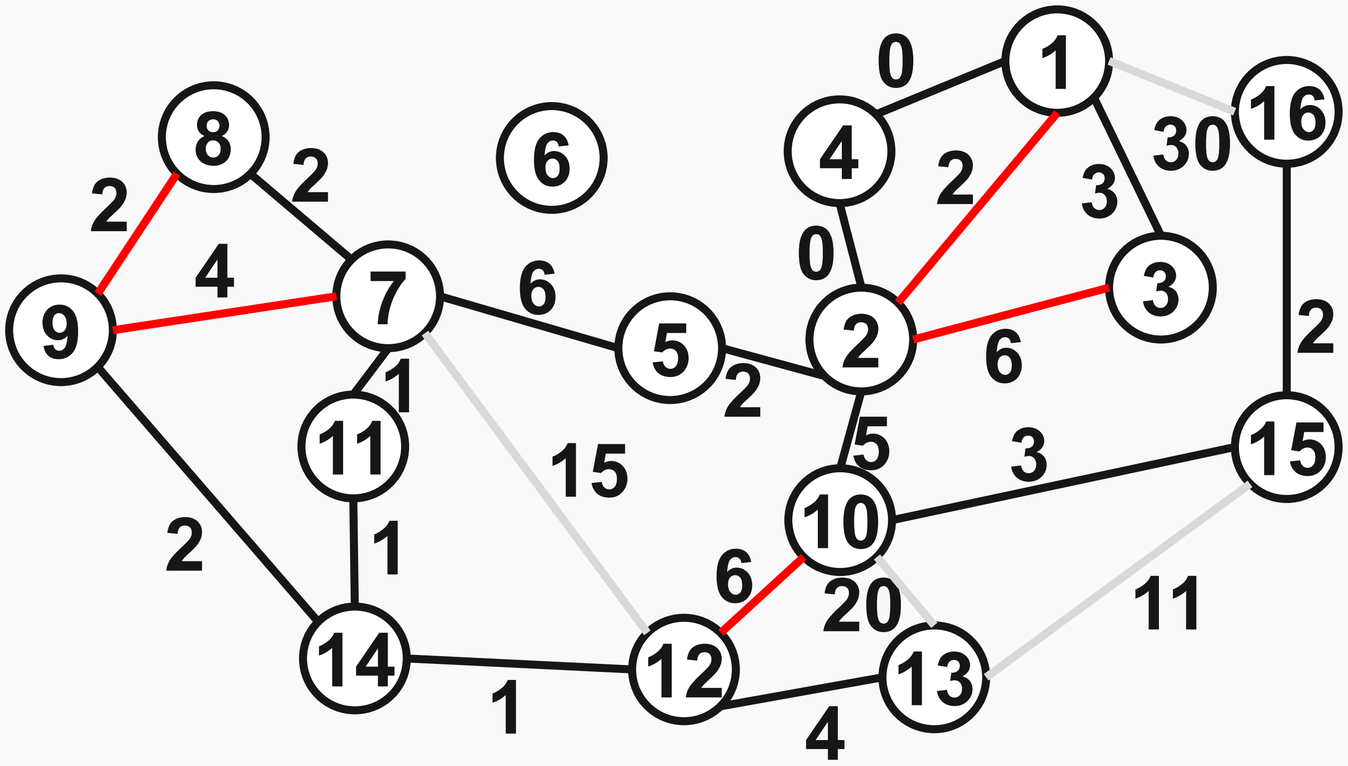
Algoritmul lui Kruskal



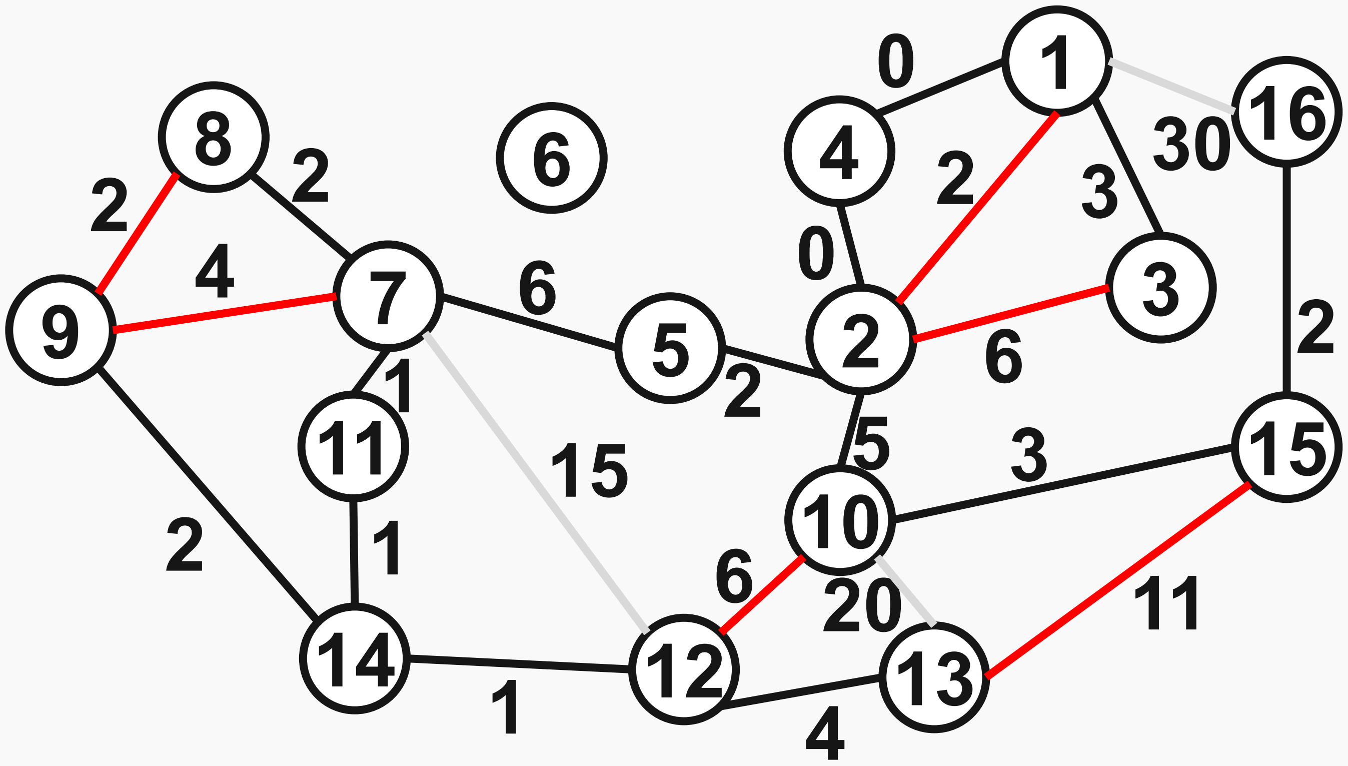
Algoritmul lui Kruskal



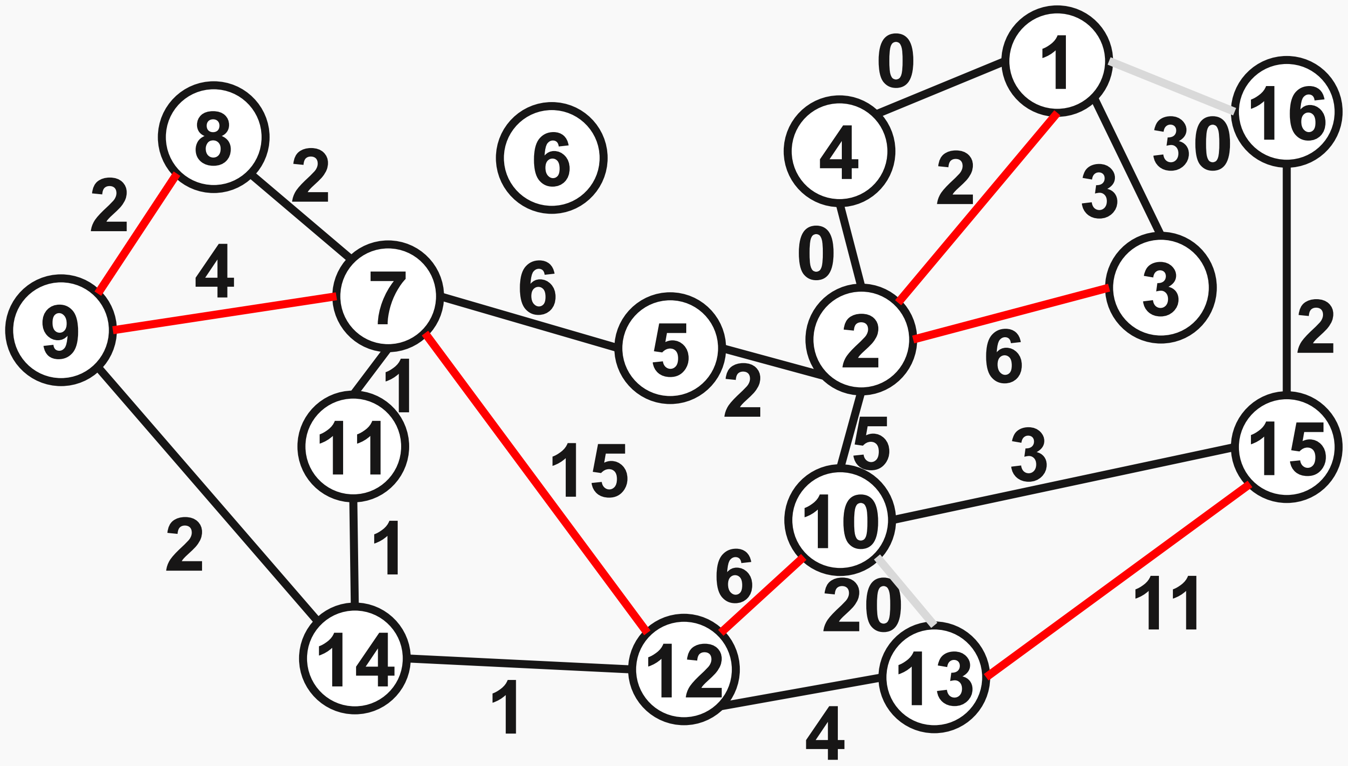
Algoritmul lui Kruskal



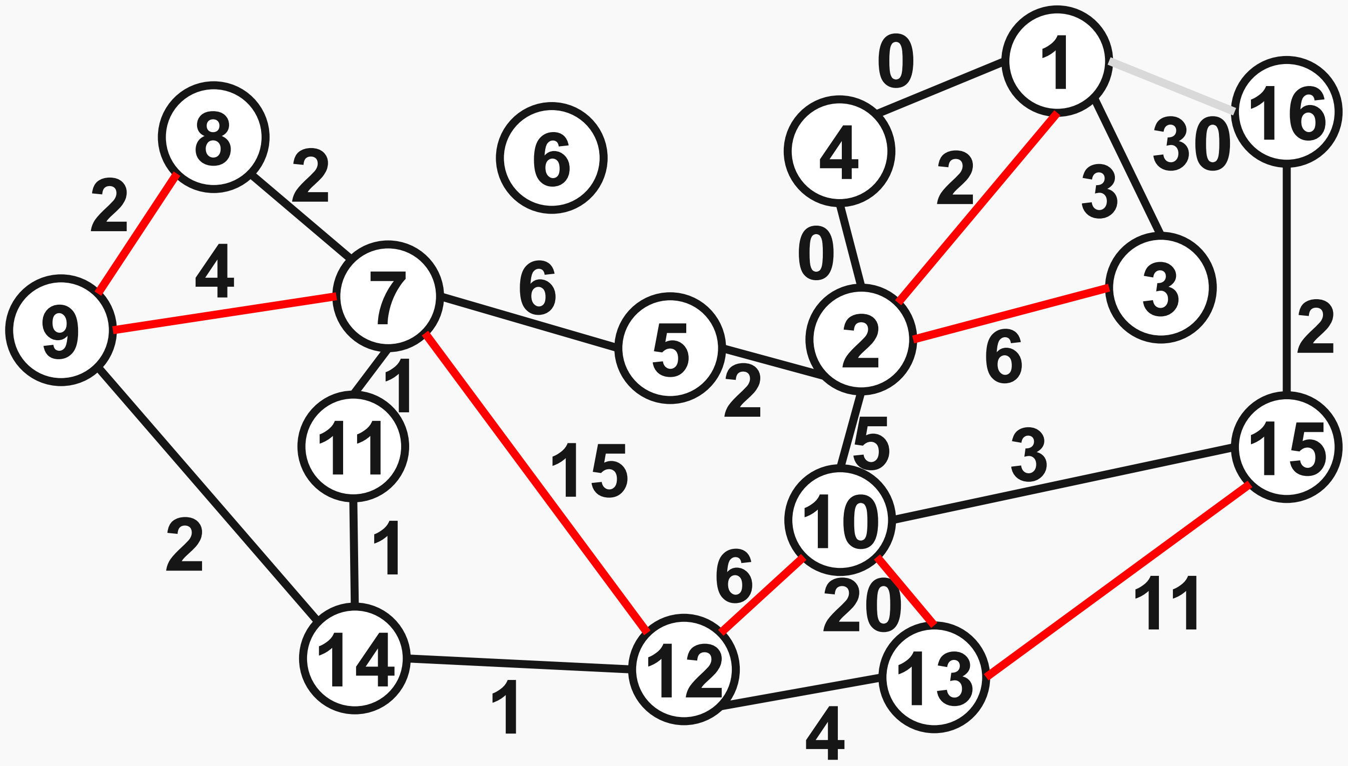
Algoritmul lui Kruskal



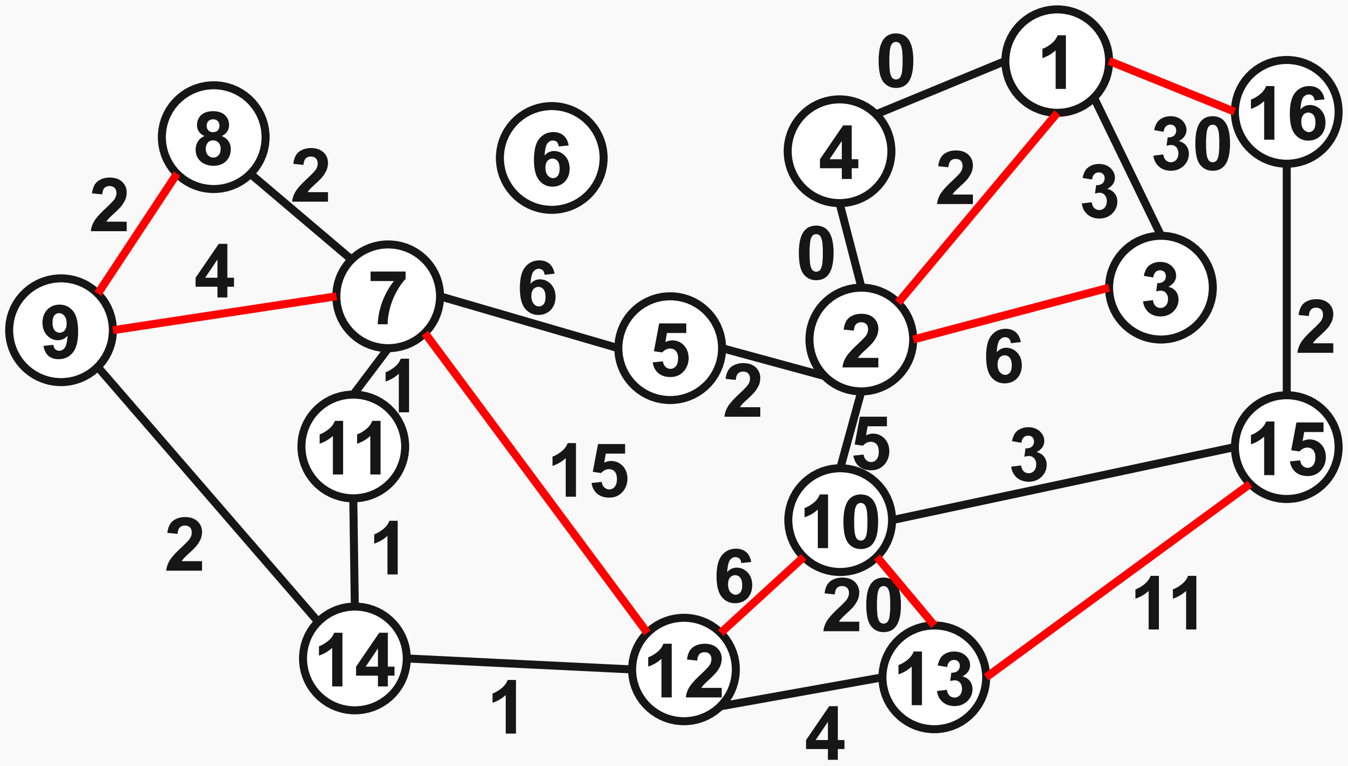
Algoritmul lui Kruskal



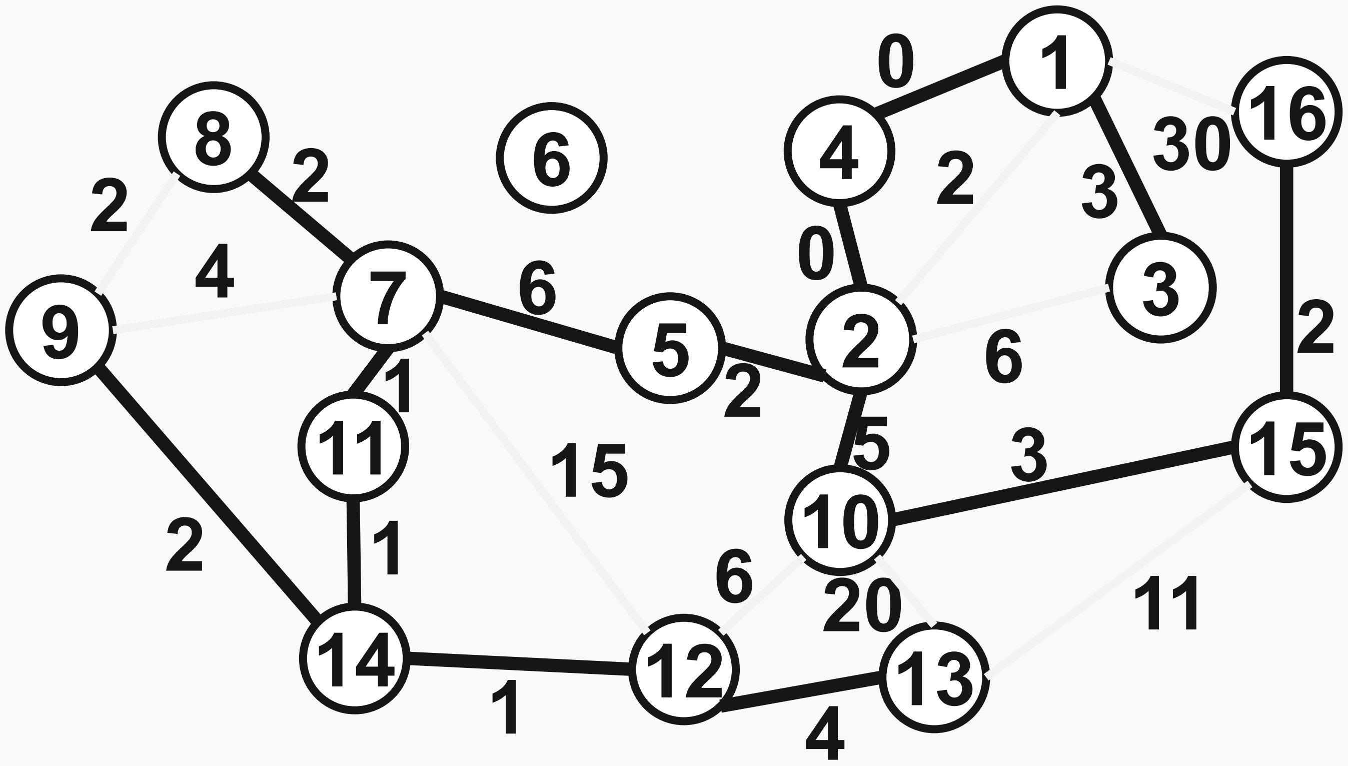
Algoritmul lui Kruskal



Algoritmul lui Kruskal



Algoritmul lui Kruskal



Complexitate?

```
tree Kruskal(G) {
    sort(G.E); // sort by weight
    A = {};
    for each (node in G.V)
        Make_set(node);
    for each ((u, v) in G.E) {
        if (Find_set(u) != Find_set(v)) {
            A = A ∪ {(u, v)};
            Union(Find_set(u), Find_set(v));
        }
    }
    return A;
}
```

Complexitate?

$$O(E\log(E))$$

Flux maxim

Graf capacitate

- În general valoarea de pe muchie reprezintă o distanță.

  - O distanță mai mare face muchia mai greu de parcurs.

- Valoarea muchiei poate reprezenta o capacitate.

  - Similar apei/curentului, cu cât capacitatea e mai mare cu atât e mai ușor de parcurs.

- Șoselele au și distanță și capacitate (număr benzi/viteză max)

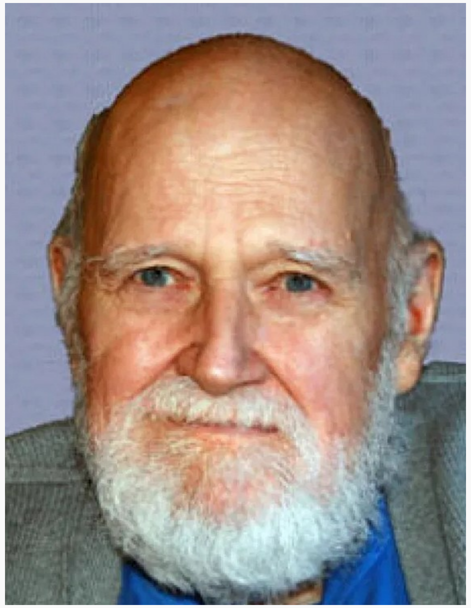
Flux maxim Algoritmul Ford-Fulkerson

- c capacitate muchie
- f flow muchie. Capacitate folosită
- $c_f(u, v) = c(u, v) - f(u, v)$ diferența de capacitate
- G_f graful cu muchii c_f

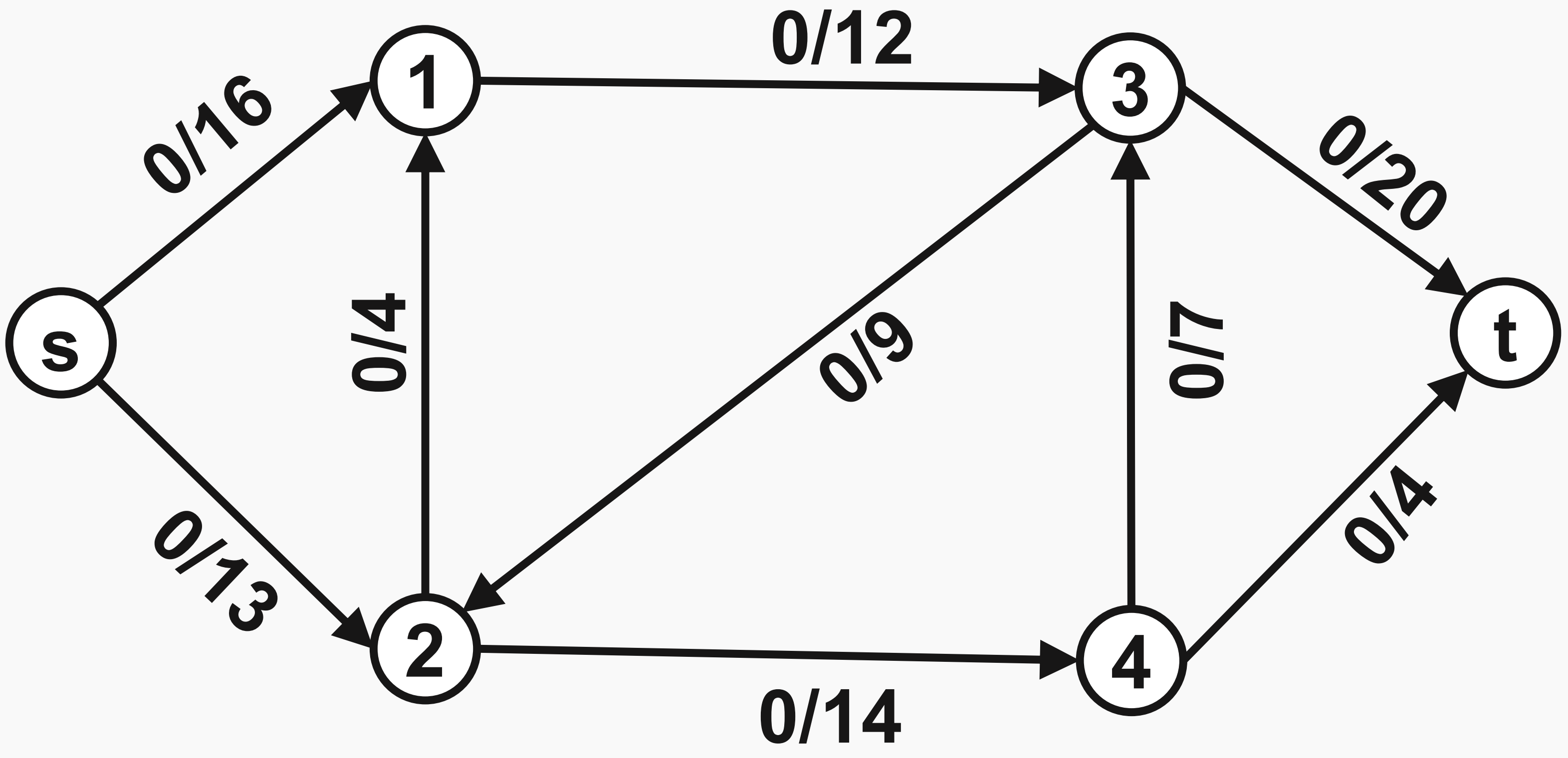
FORD-FULKERSON (G, s, t)
for each edge $(u, v) \in G.E$
    $(u, v).f = 0$
while there exists a path p from s to t in the residual network G_f
    $c_f(p) = \min\{c_f(u, v): (u, v) \ is \ in \ p\}$
    for each edge (u, v) in p
        if $(u, v) \in G.E$
            $(u, v).f = (u, v).f + c_f(p)$
        else
            $(v, u).f = (v, u).f - c_f(p)$

Flux maxim Algoritmul Ford-Fulkerson (1956)

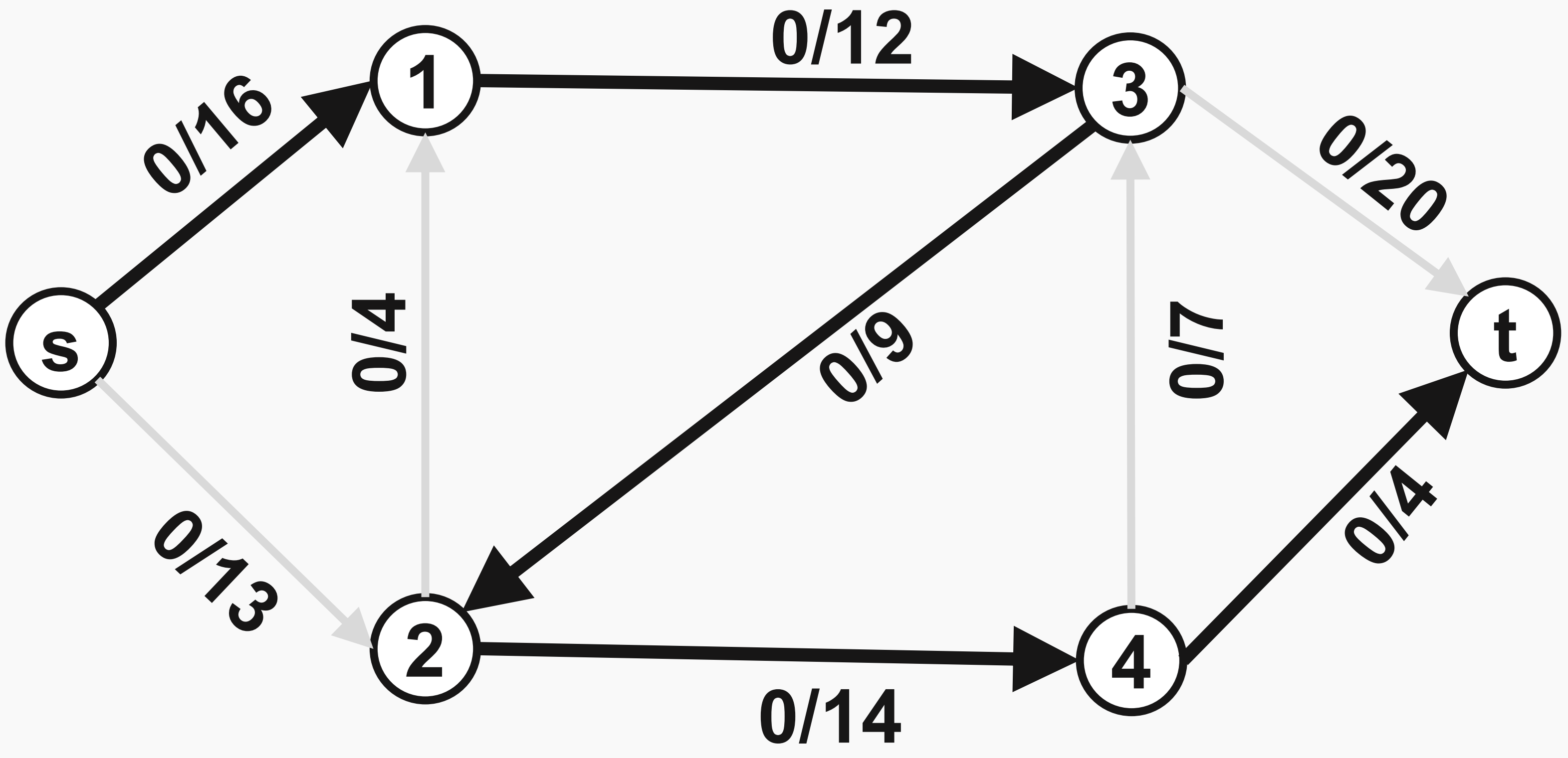
- c capacitate muchie
- f flow muchie. Capacitate folosită
- $c_f(u,v) = c(u,v) - f(u,v)$ diferența de capacitate
- G_f graful cu muchii c_f



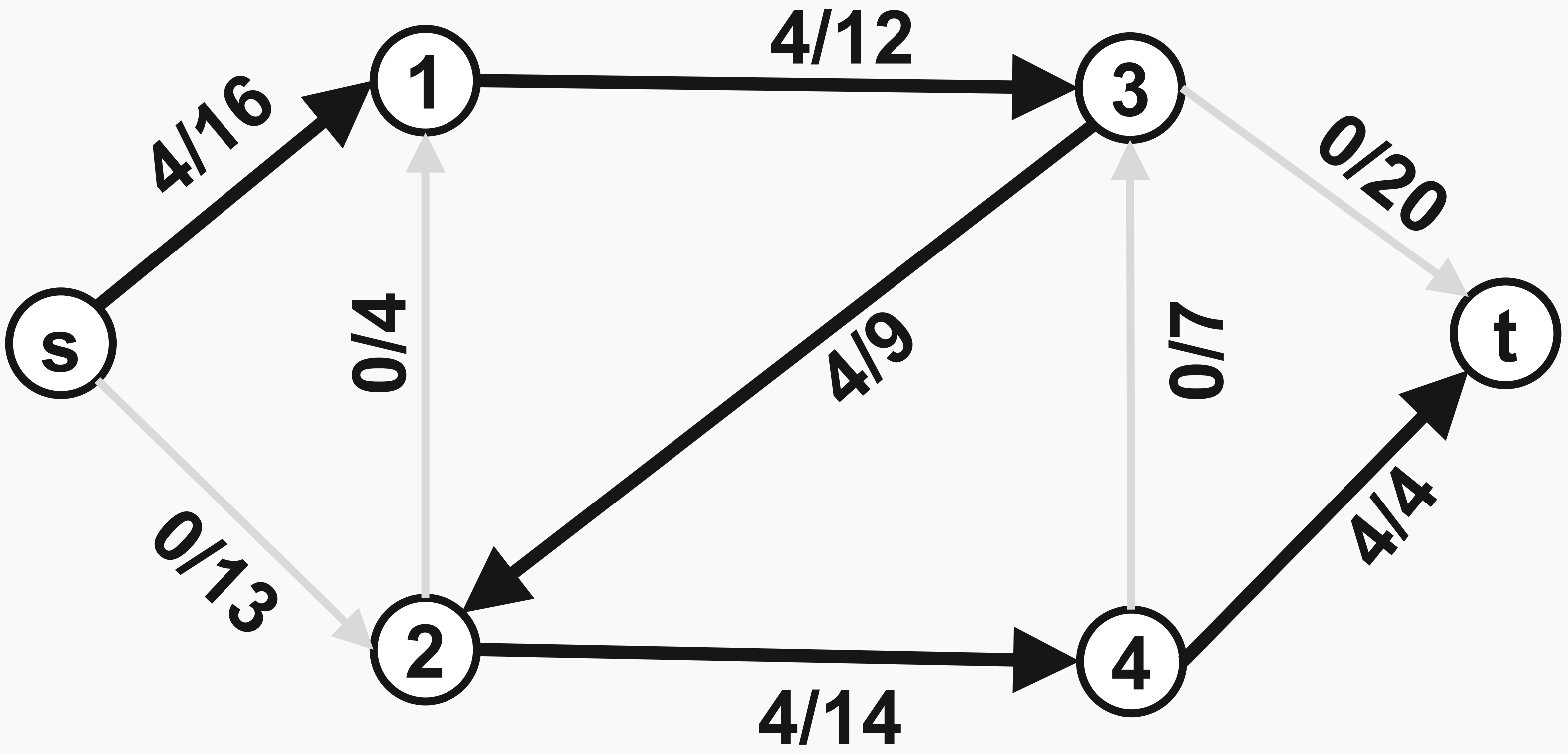
Algoritmul Ford-Fulkerson



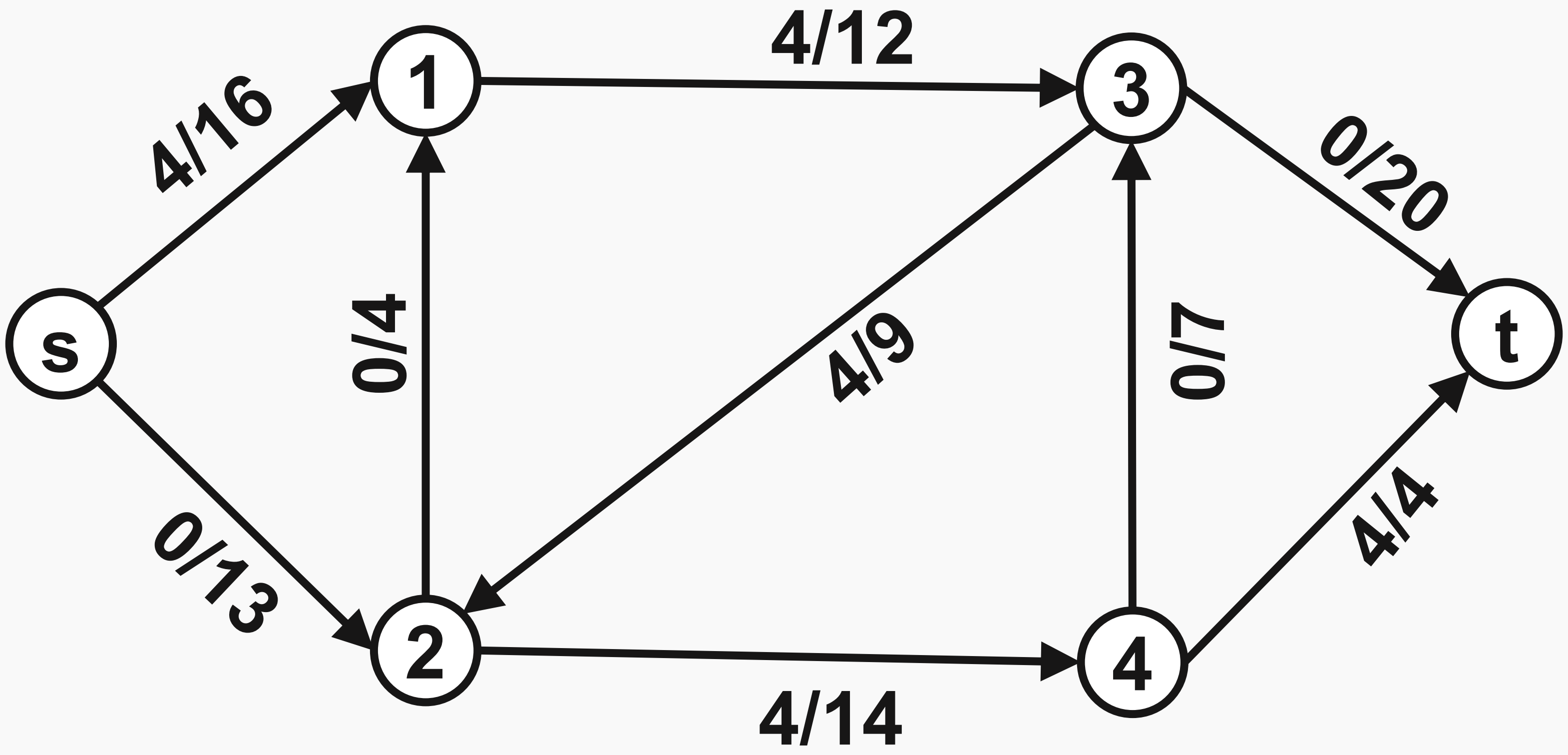
Algoritmul Ford-Fulkerson



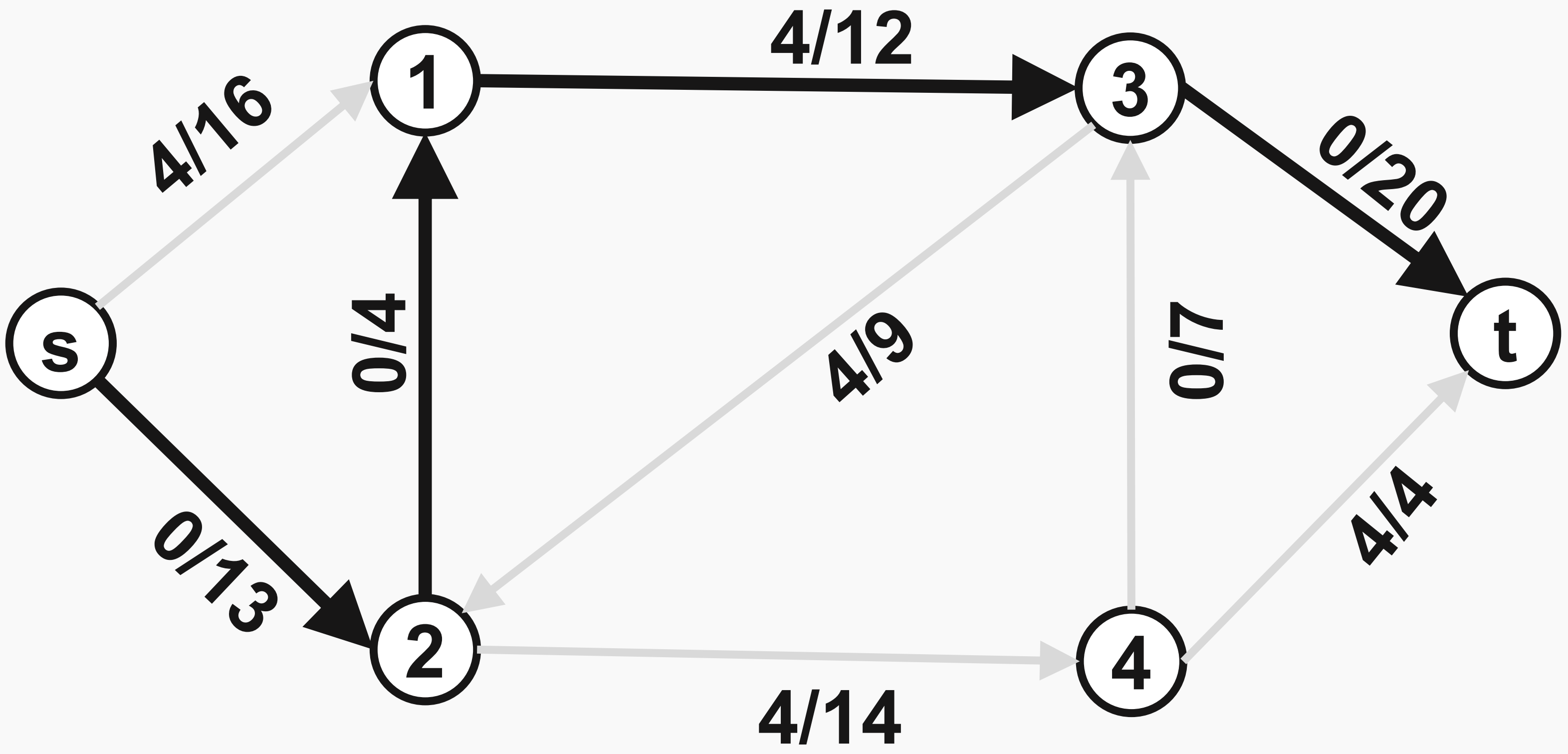
Algoritmul Ford-Fulkerson



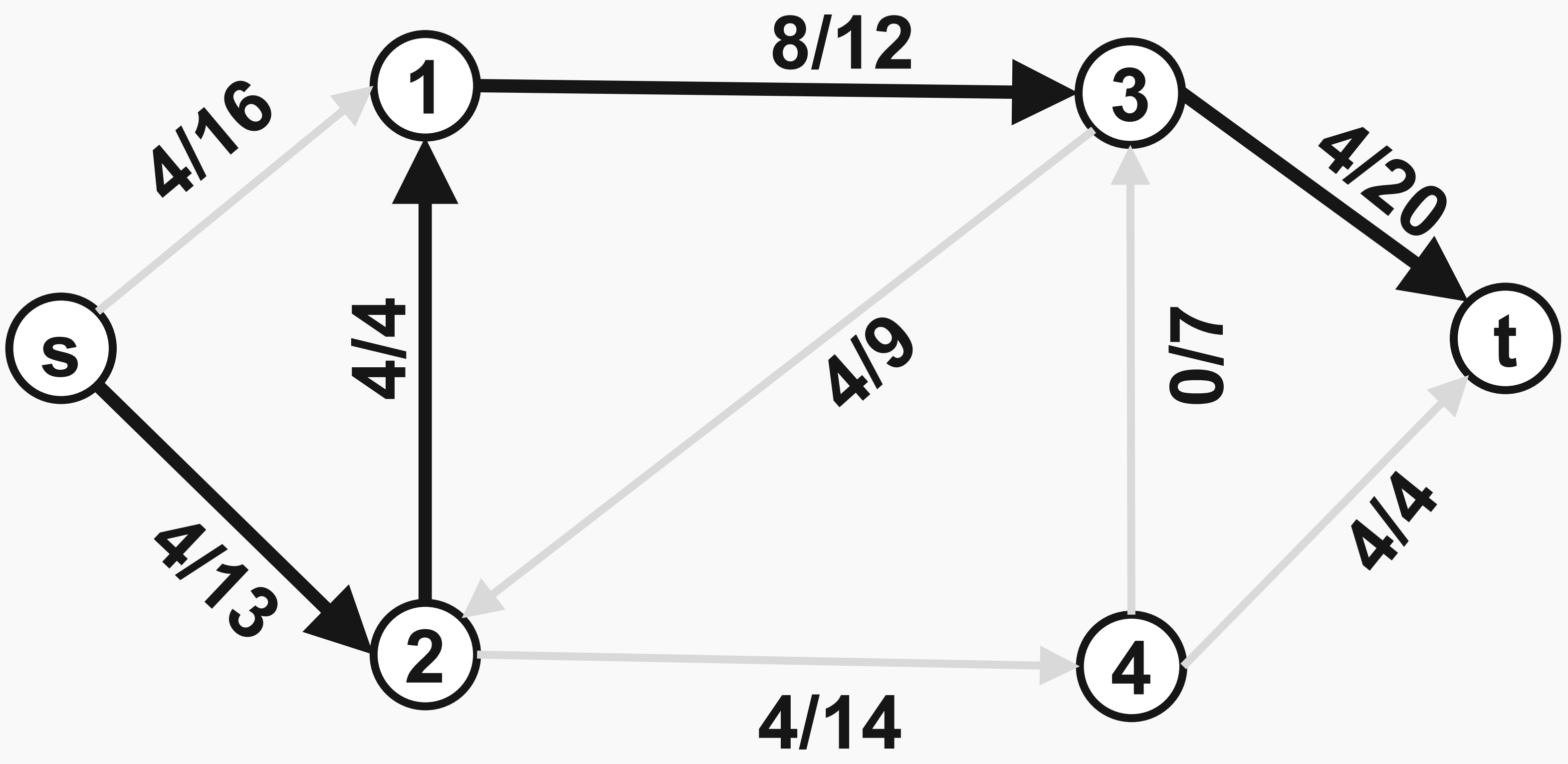
Algoritmul Ford-Fulkerson



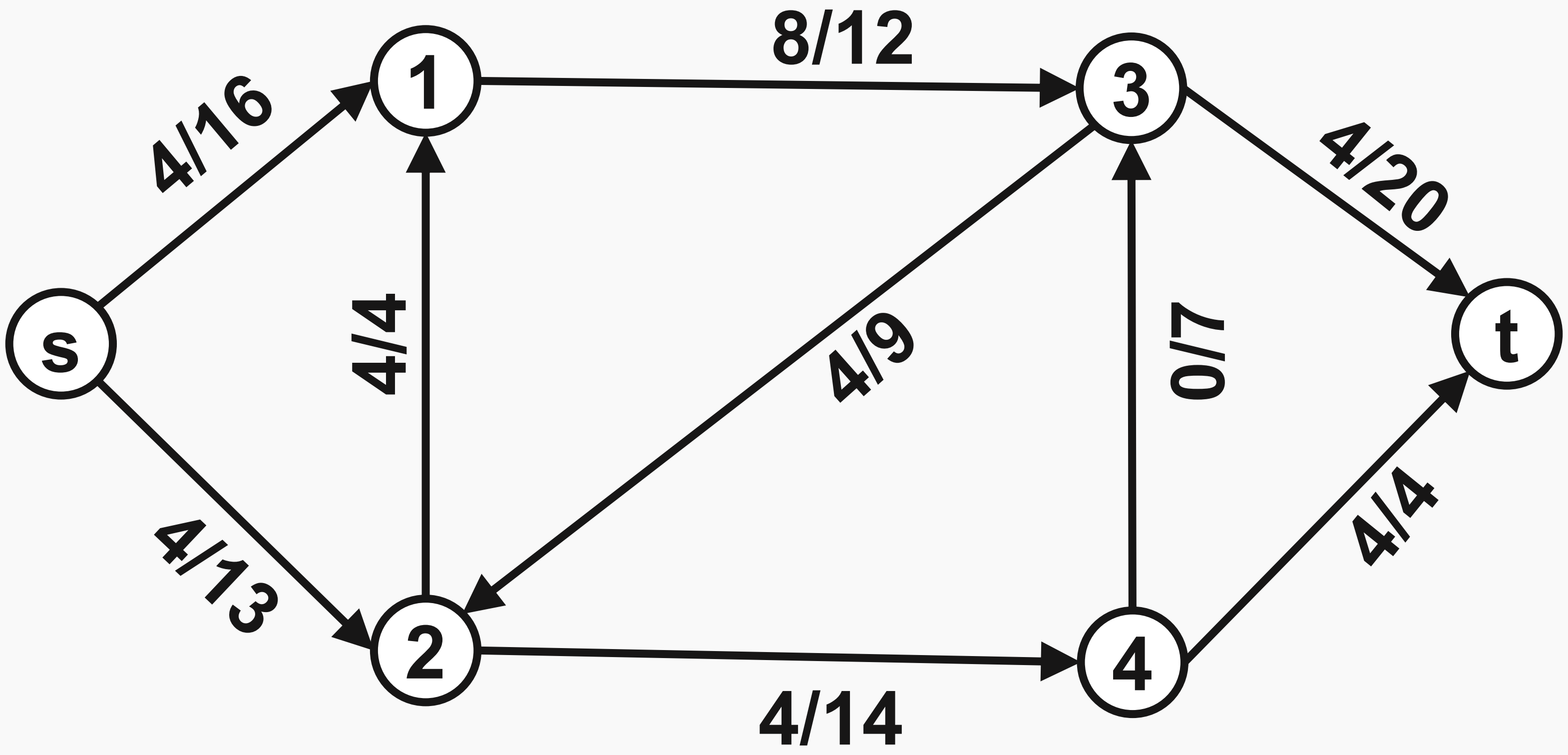
Algoritmul Ford-Fulkerson



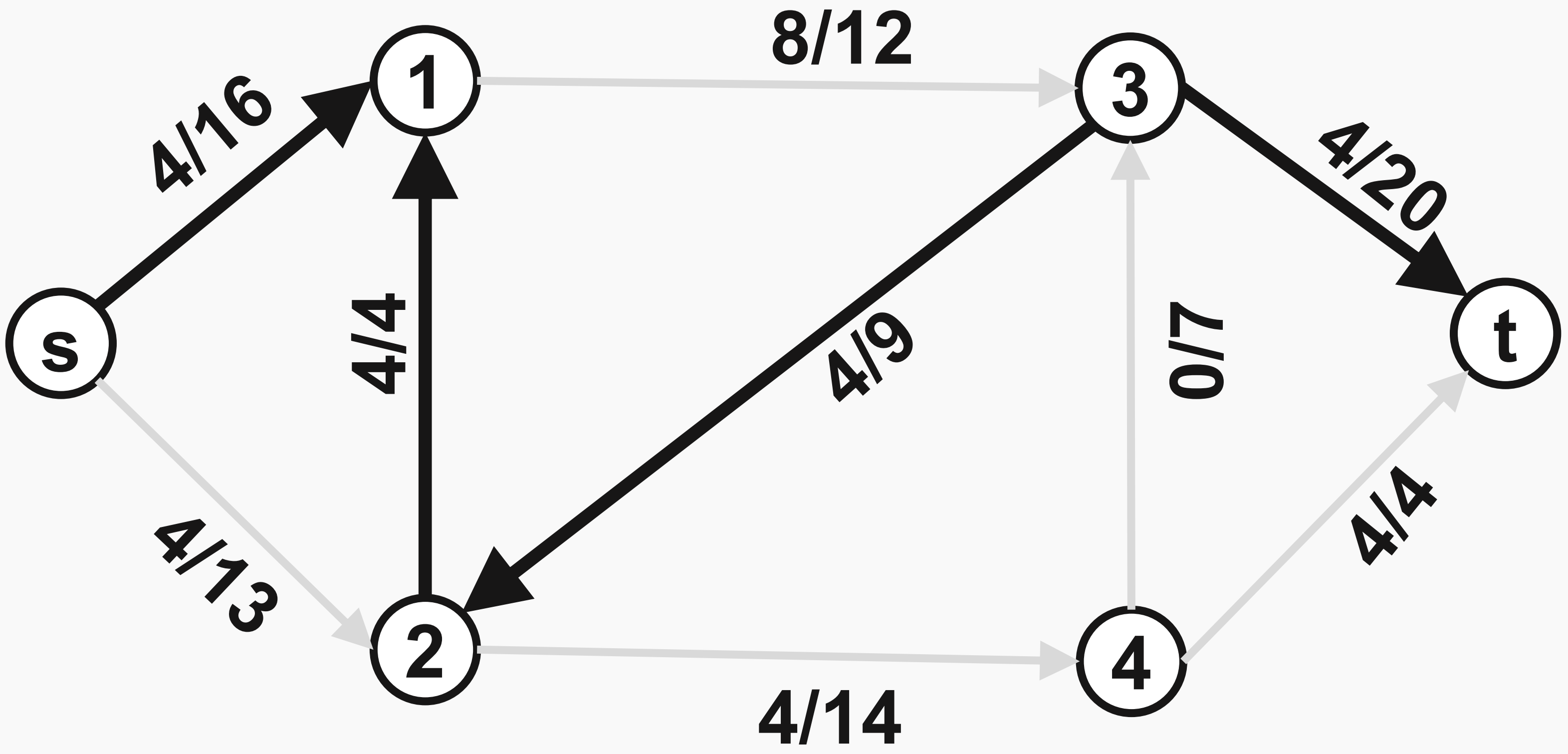
Algoritmul Ford-Fulkerson



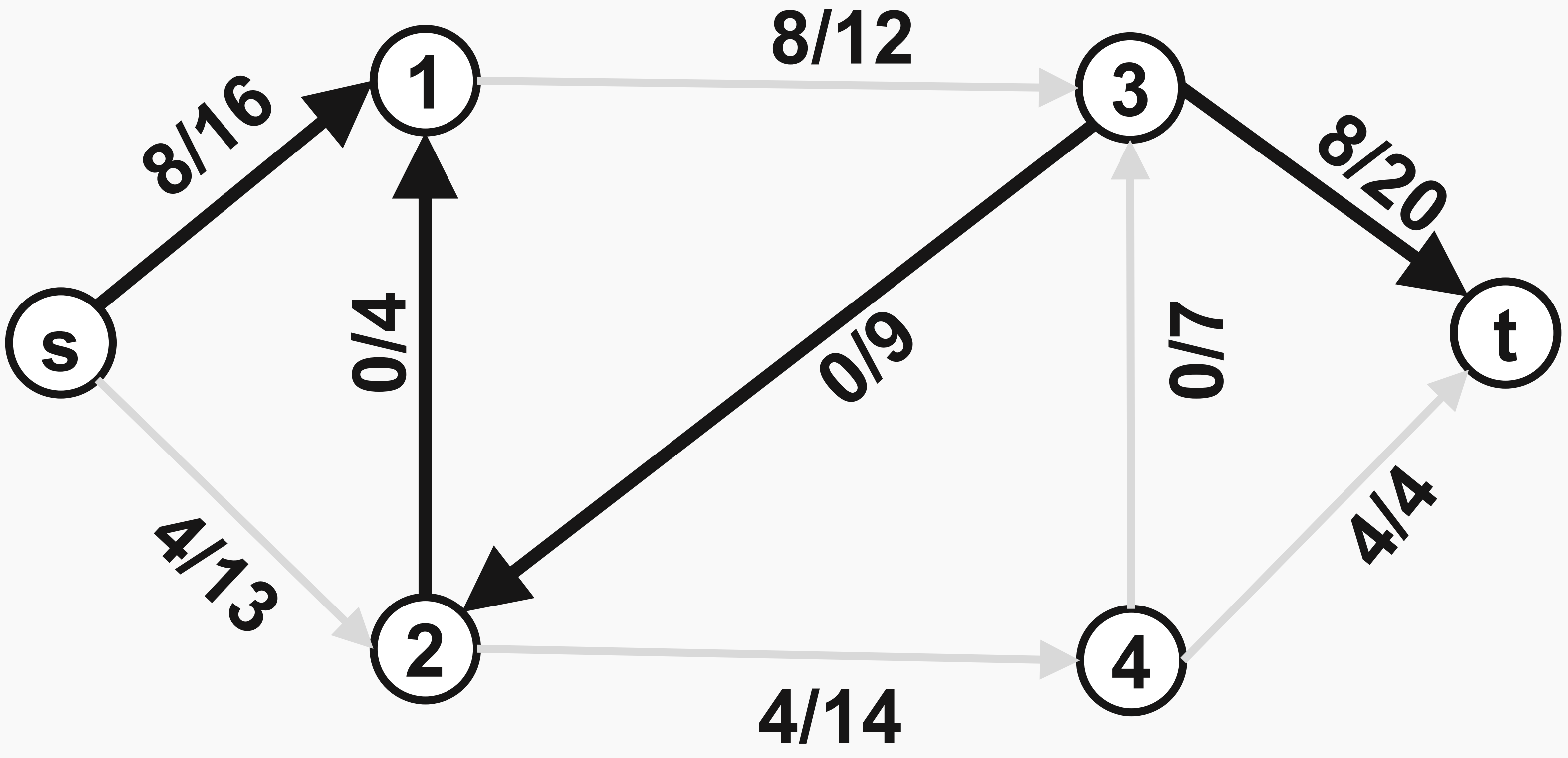
Algoritmul Ford-Fulkerson



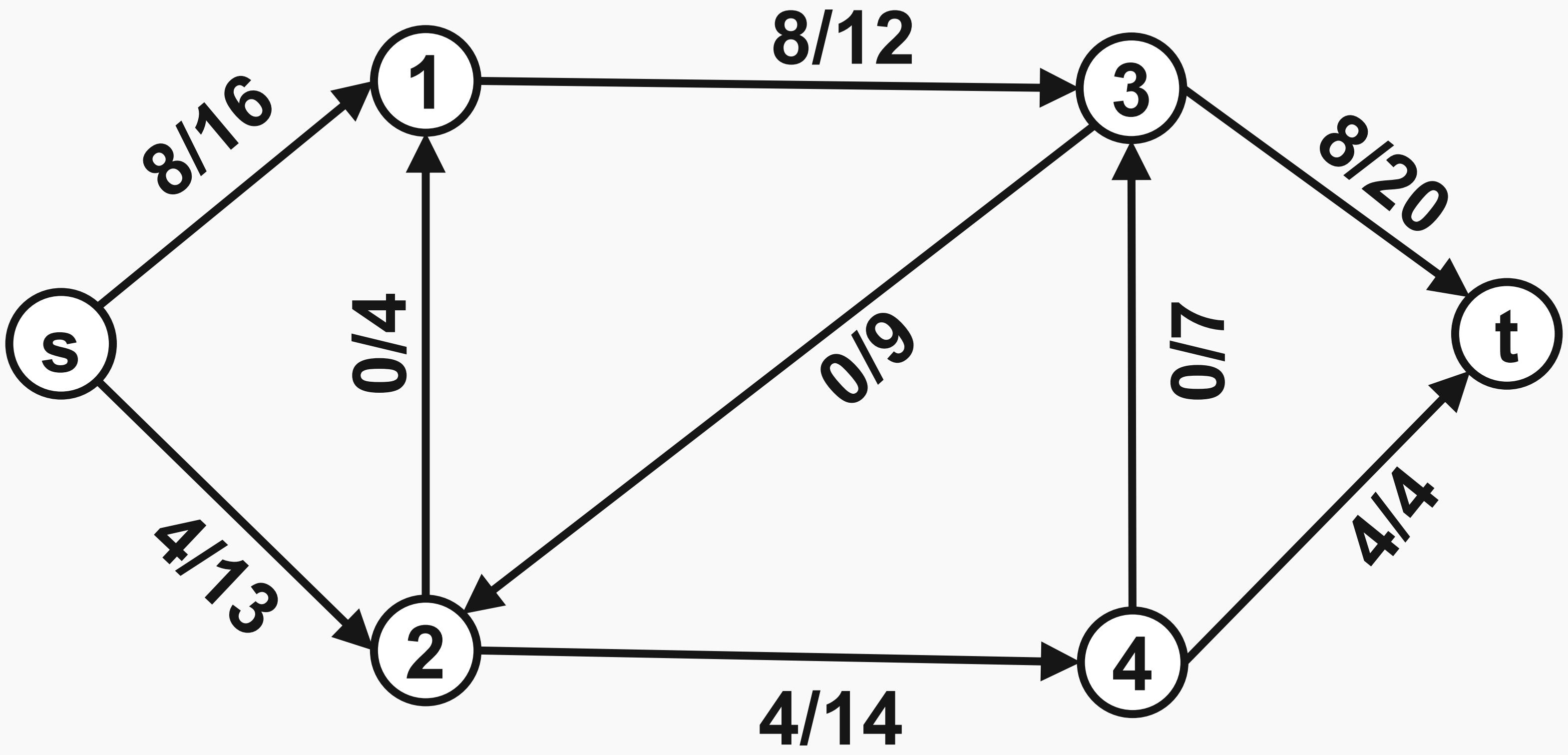
Algoritmul Ford-Fulkerson



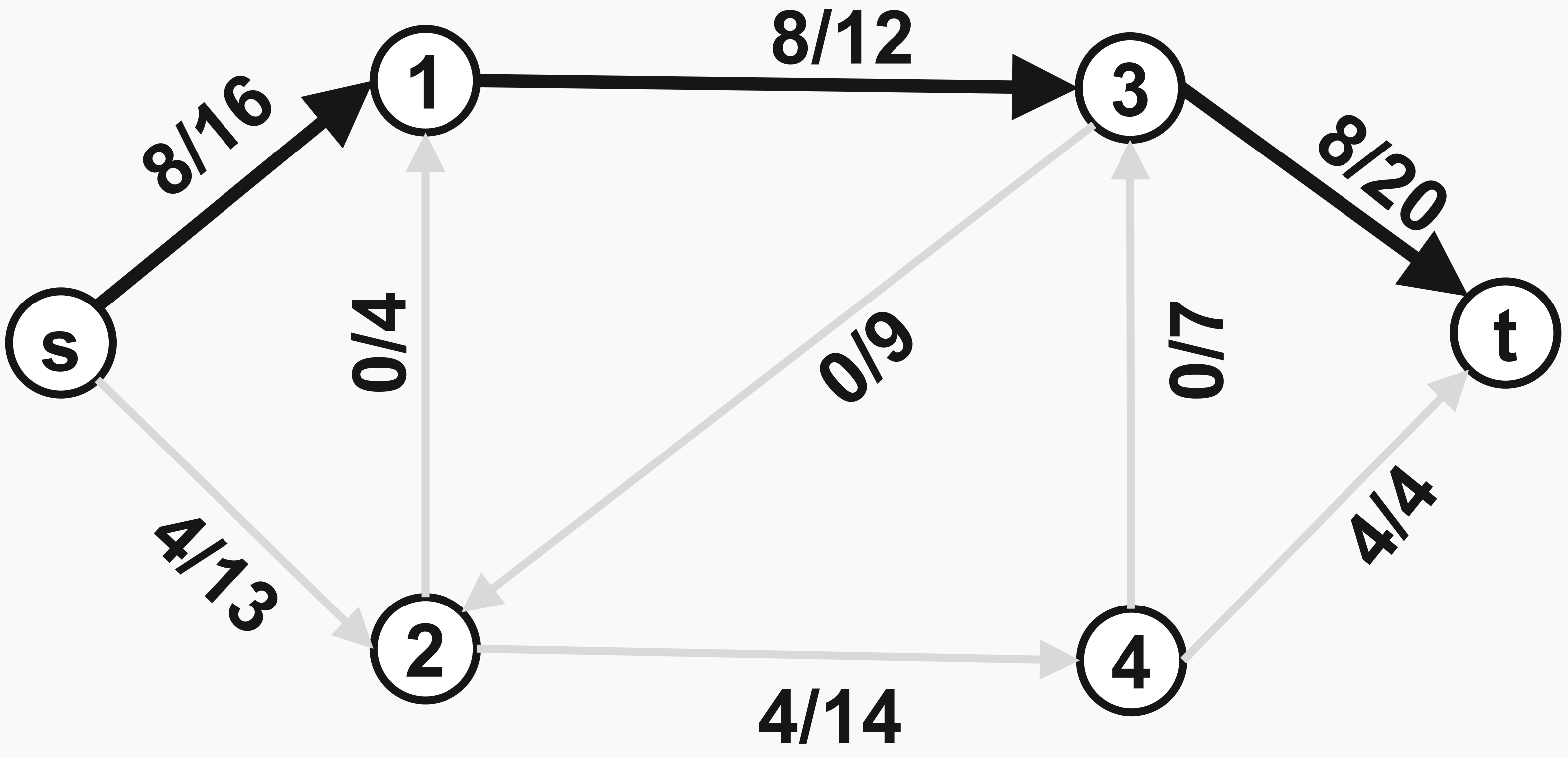
Algoritmul Ford-Fulkerson



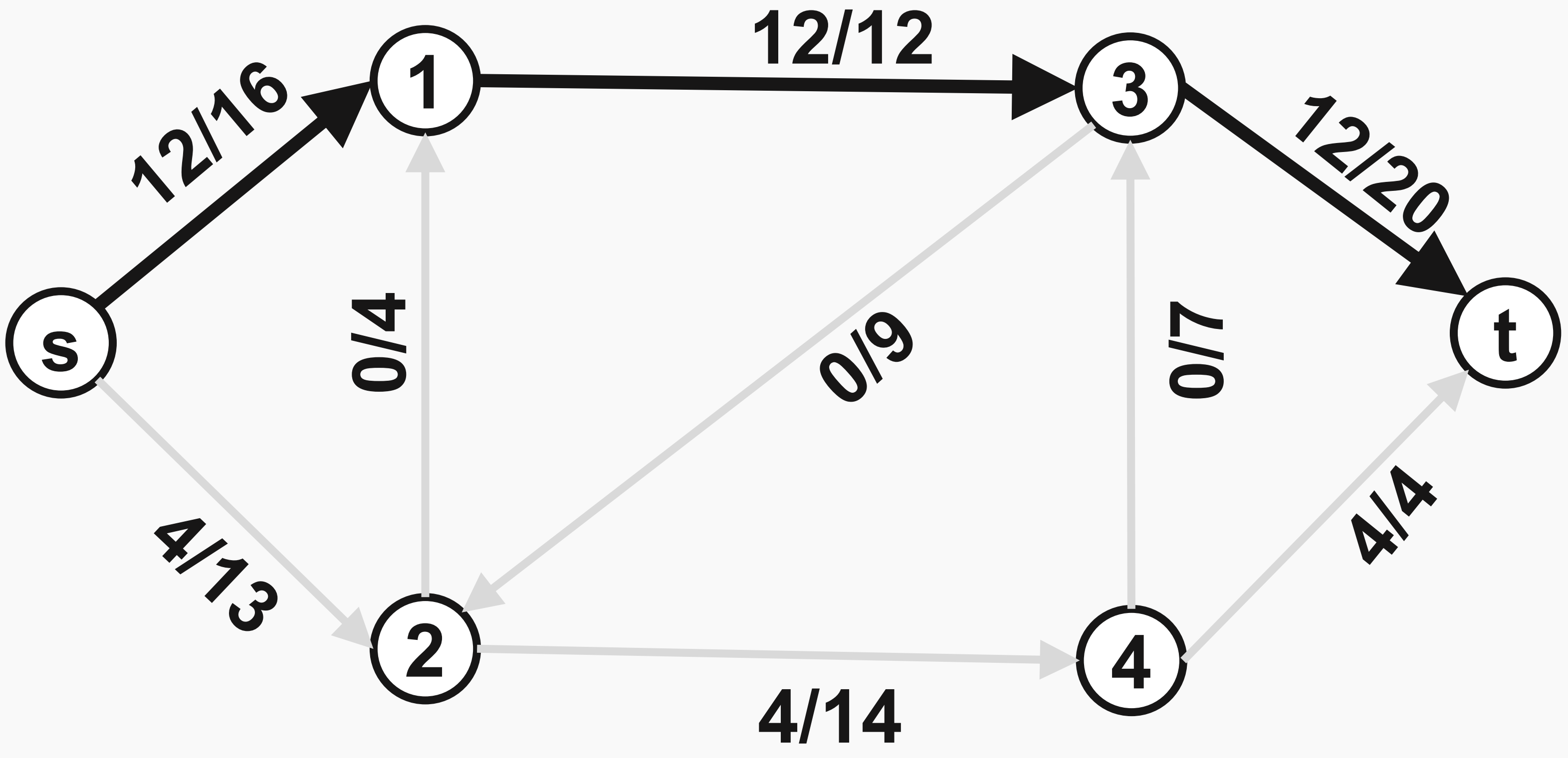
Algoritmul Ford-Fulkerson



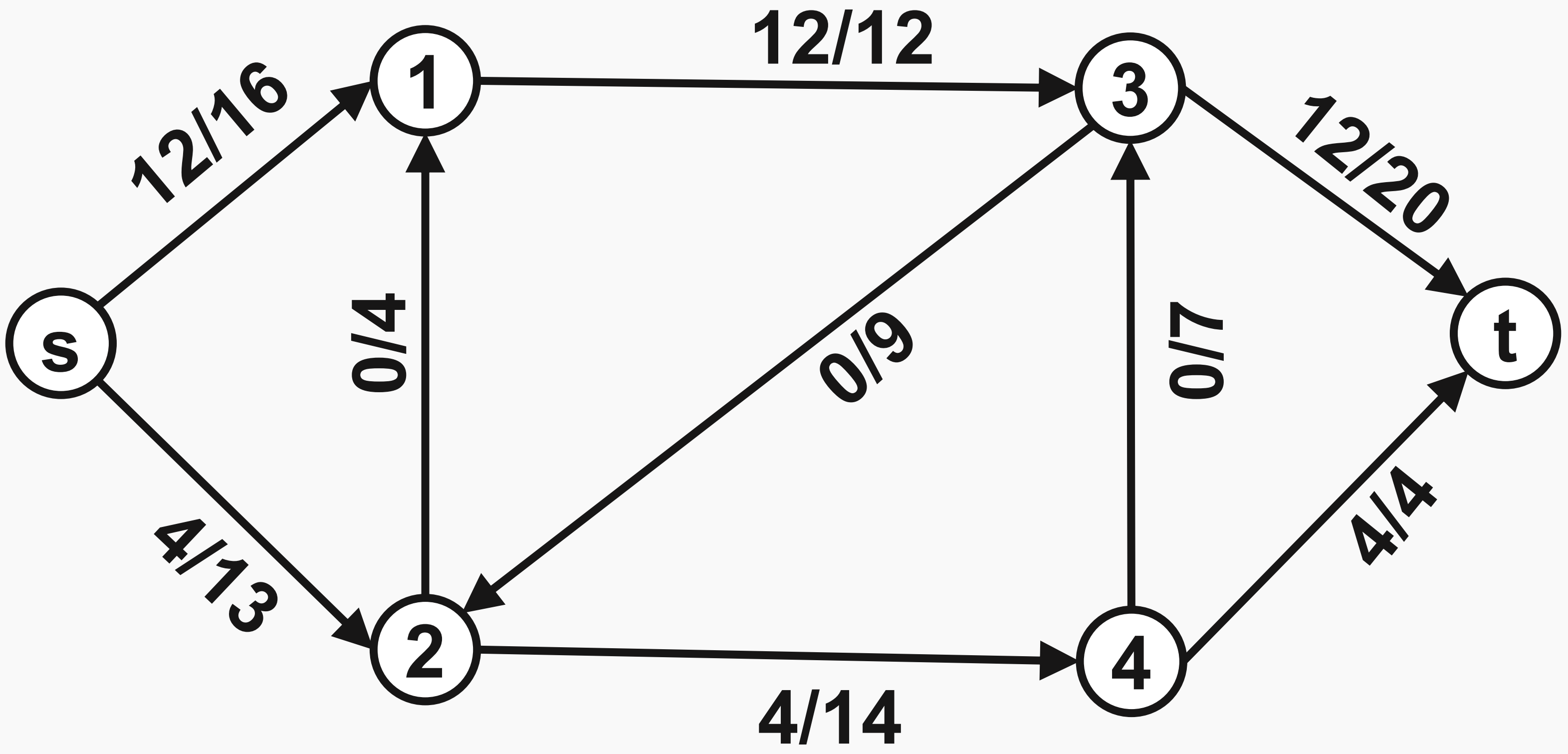
Algoritmul Ford-Fulkerson



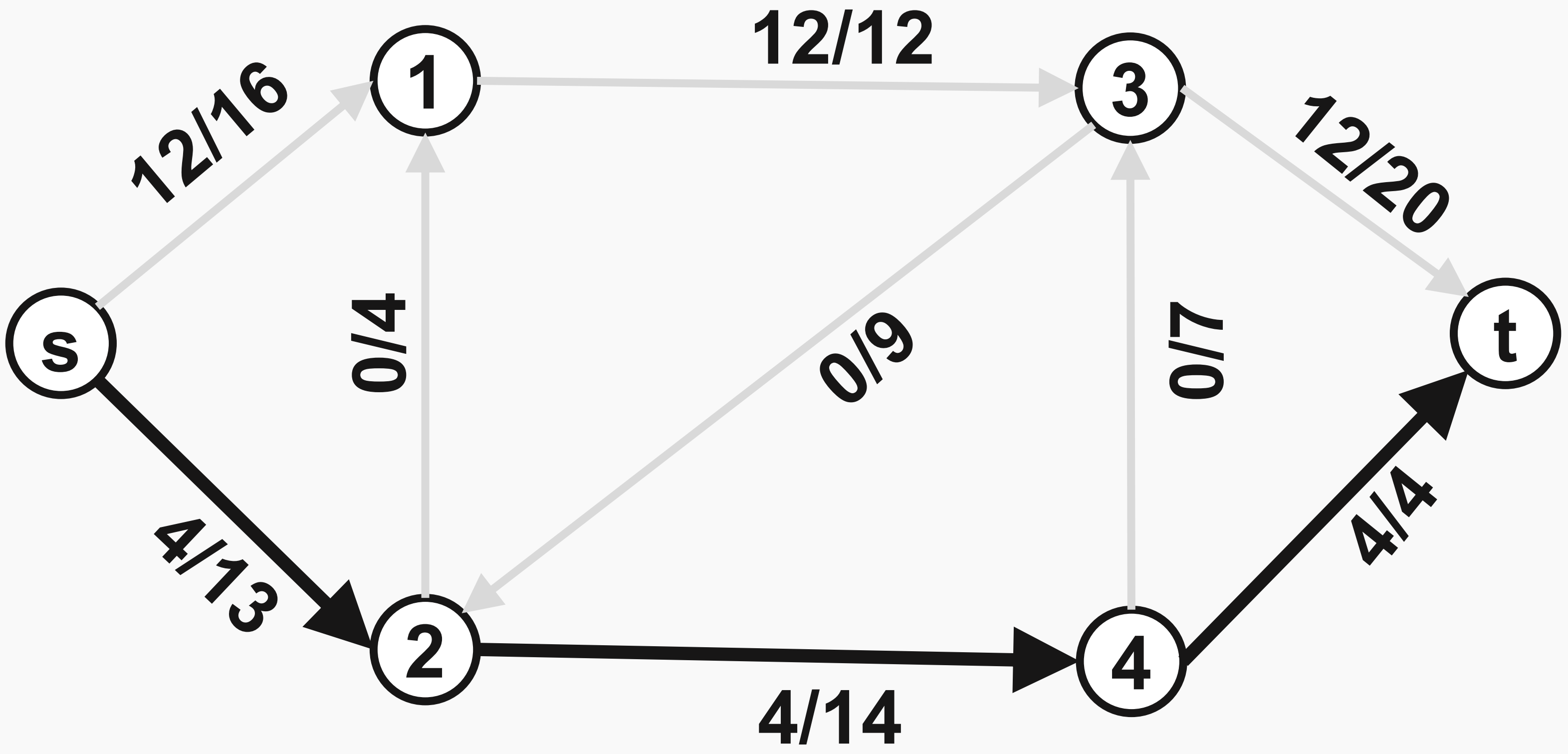
Algoritmul Ford-Fulkerson



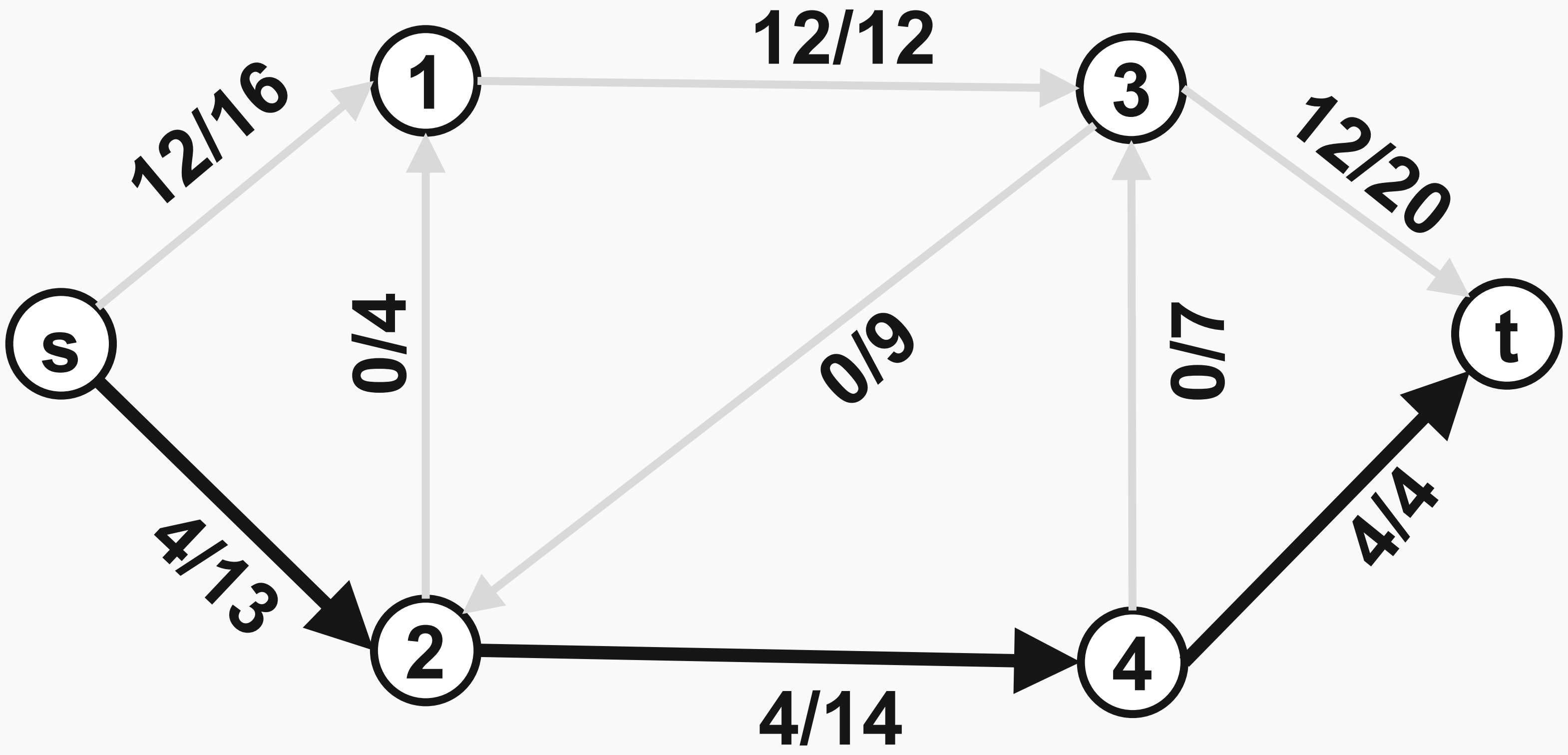
Algoritmul Ford-Fulkerson



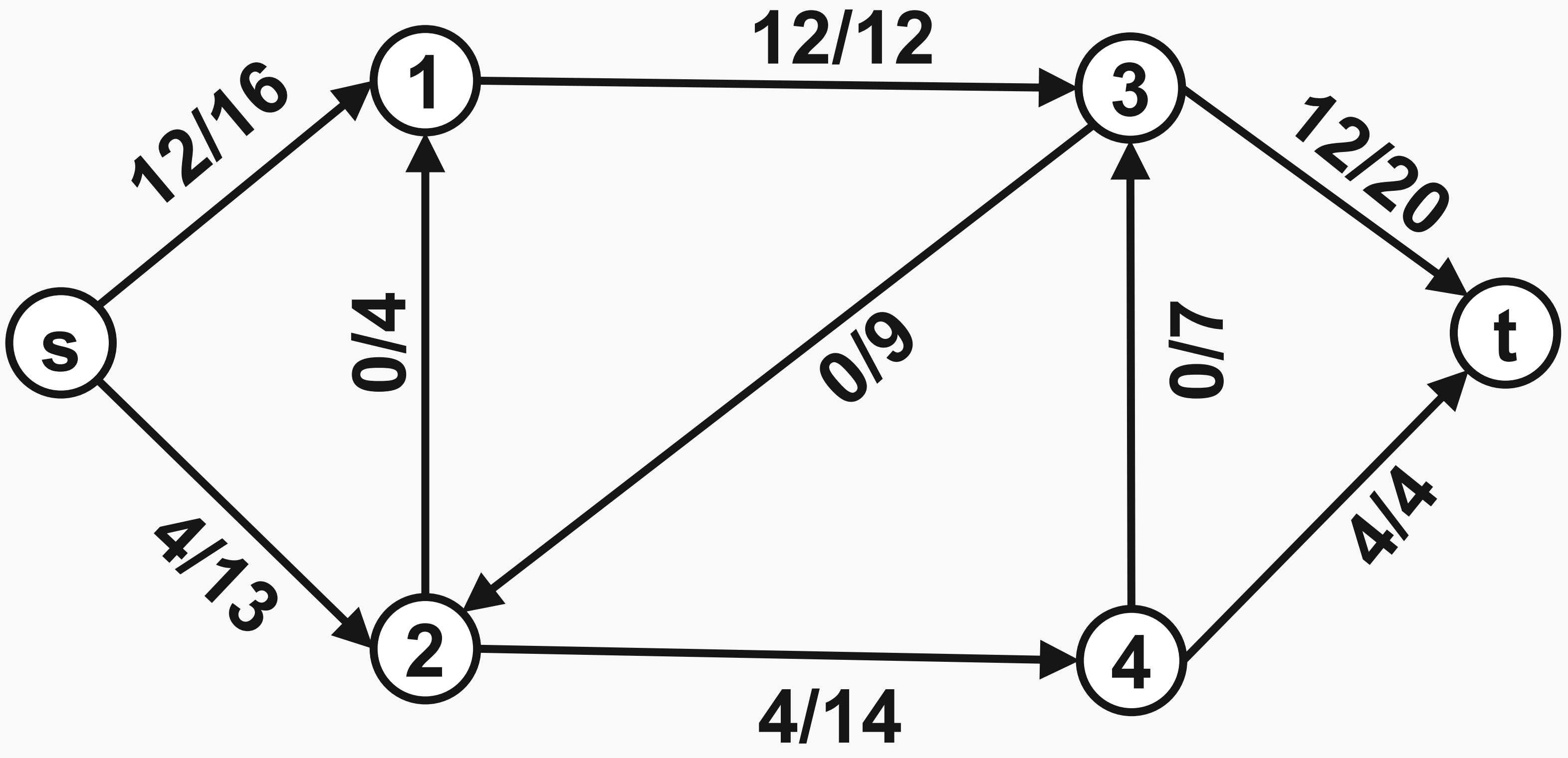
Algoritmul Ford-Fulkerson



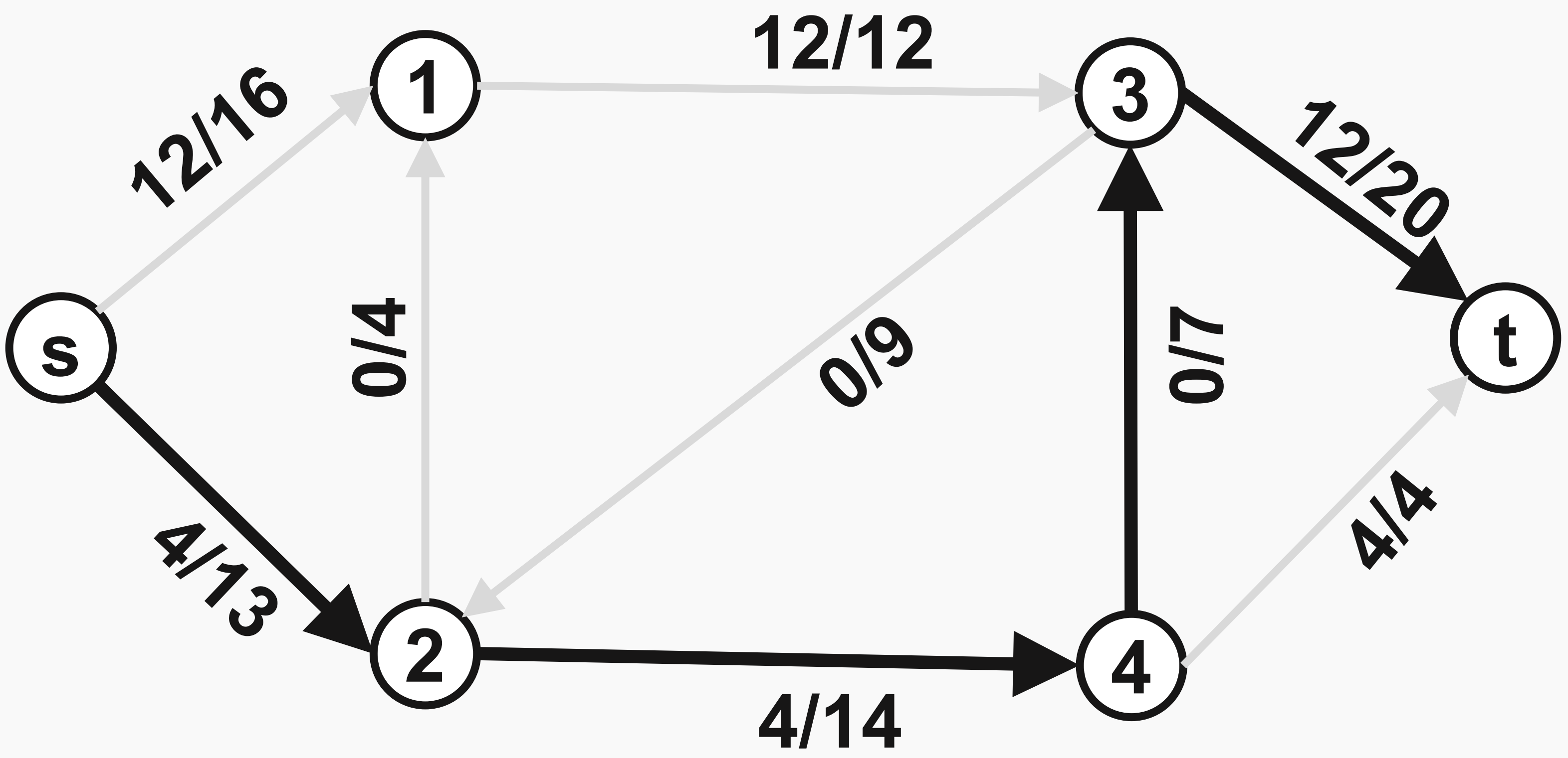
Algoritmul Ford-Fulkerson



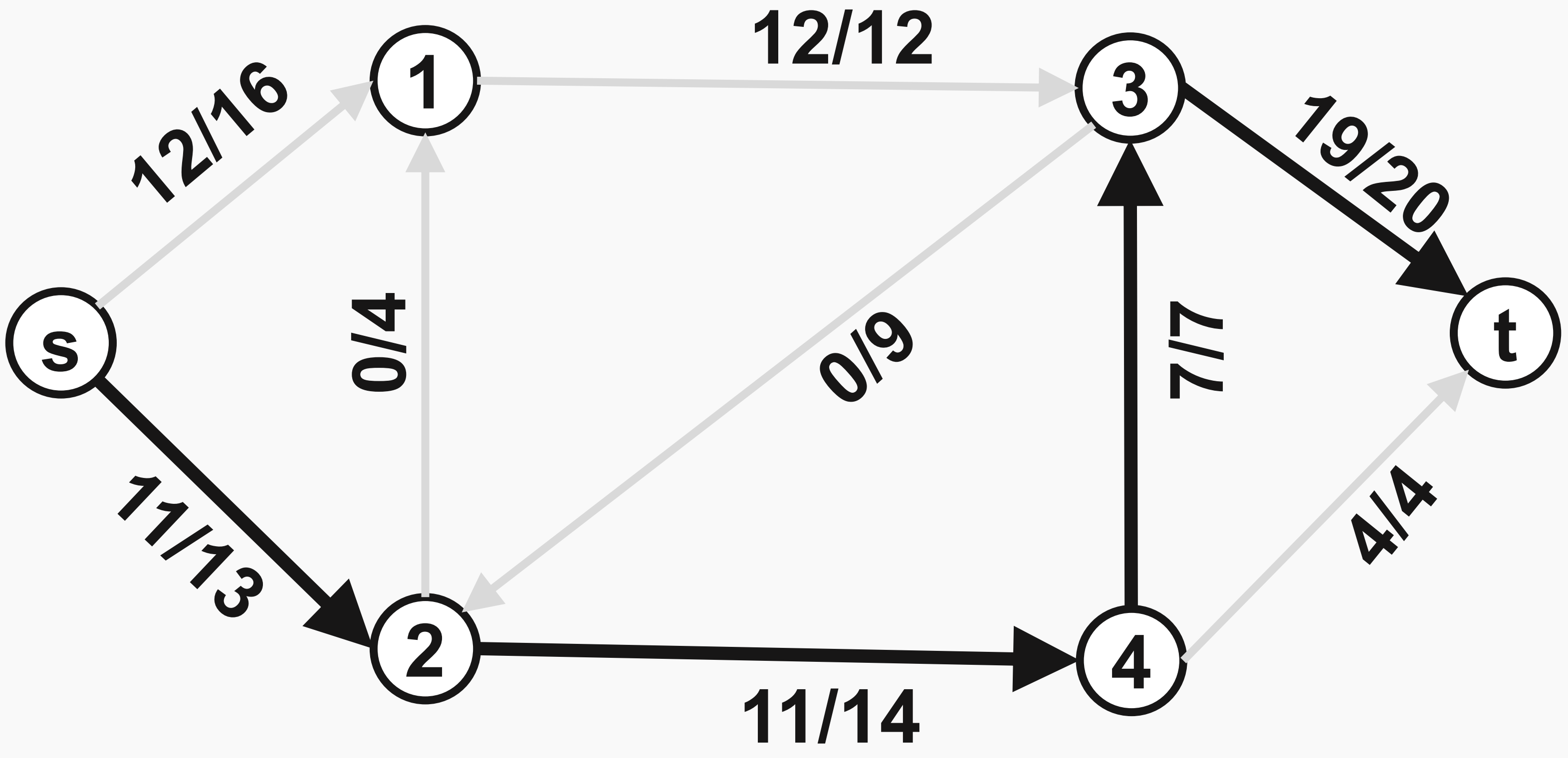
Algoritmul Ford-Fulkerson



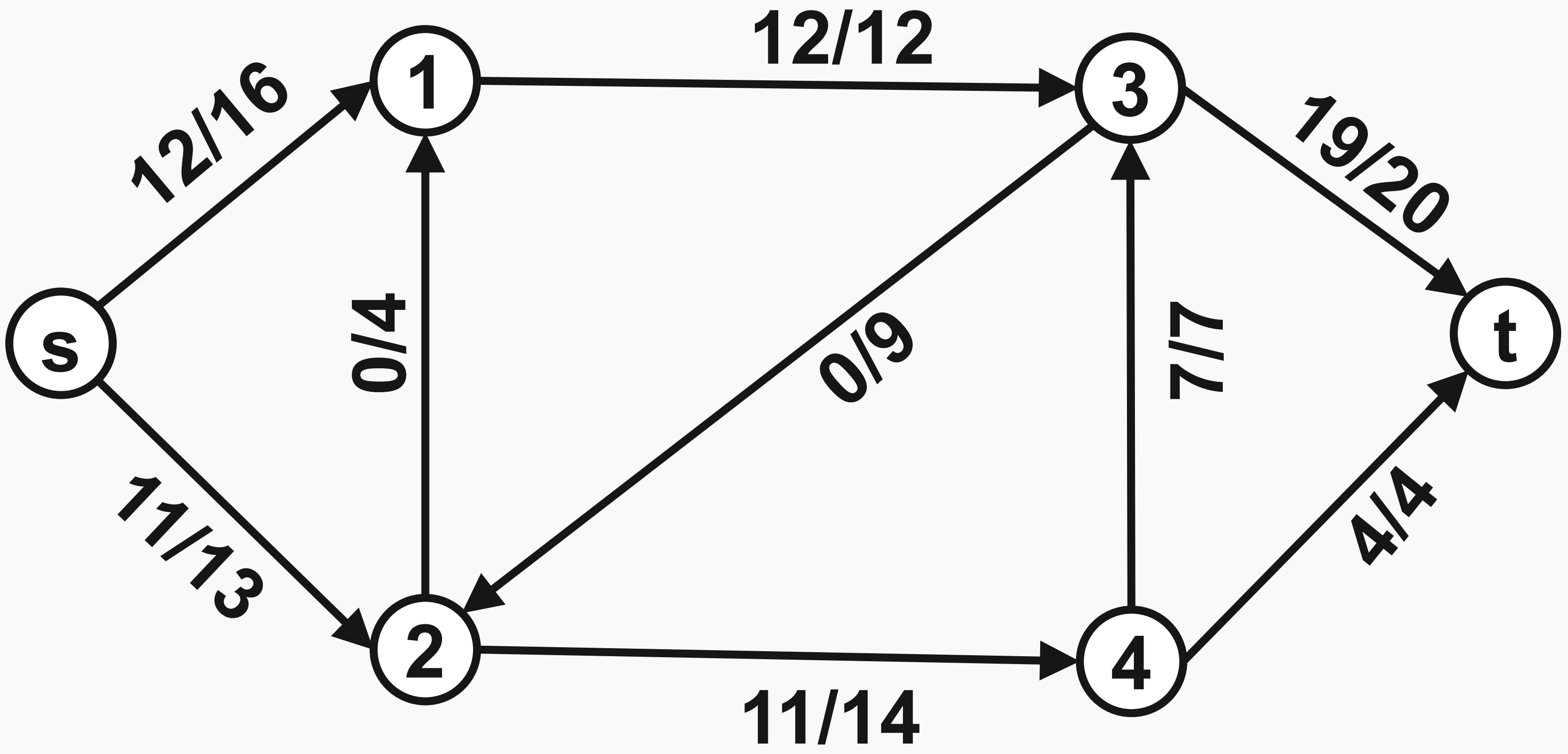
Algoritmul Ford-Fulkerson



Algoritmul Ford-Fulkerson

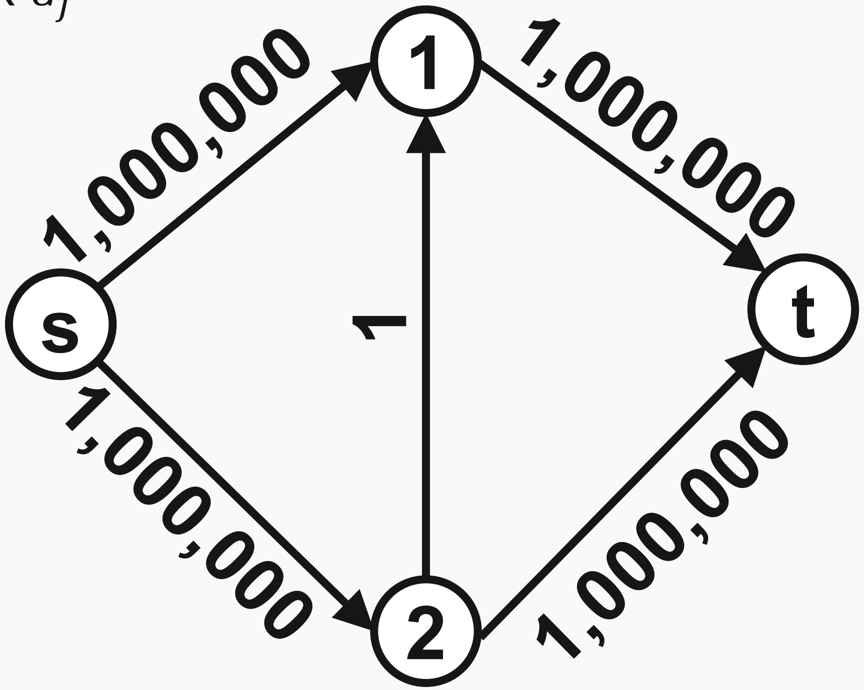


Algoritmul Ford-Fulkerson



Complexitate?

FORD-FULKERSON (G, s, t)
for each edge $(u, v) \in G.E$
$(u, v).f = 0$
while there exists a path p from s to t in the residual network G_f
$c_f(p) = \min\{c_f(u, v): (u, v)\ is\ in\ p\}$
for each edge (u, v) in p
if $(u, v) \in G.E$
$(u, v).f = (u, v).f + c_f(p)$
else
$(v, u).f = (v, u).f - c_f(p)$

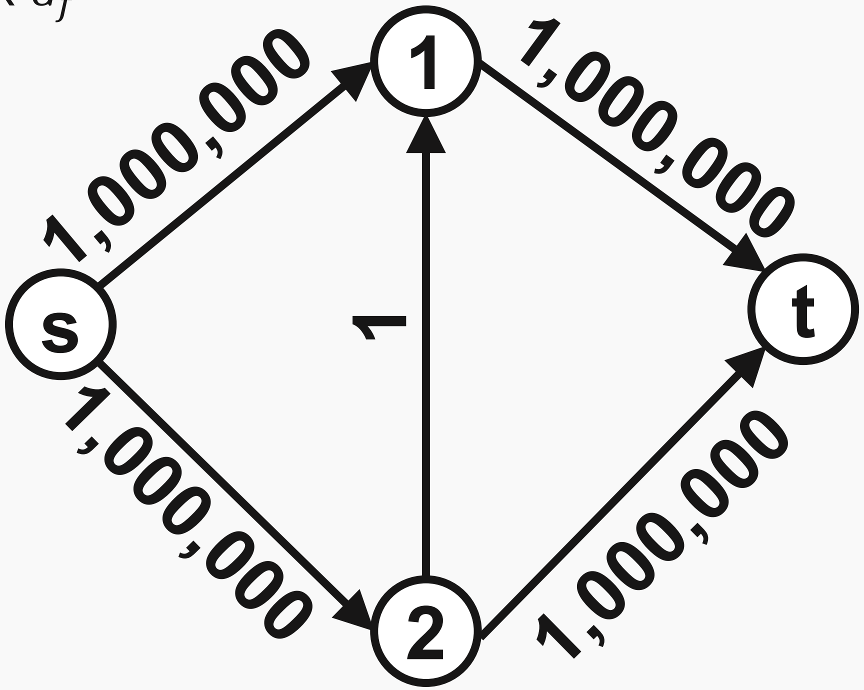


Complexitate?

Dacă alegem p random

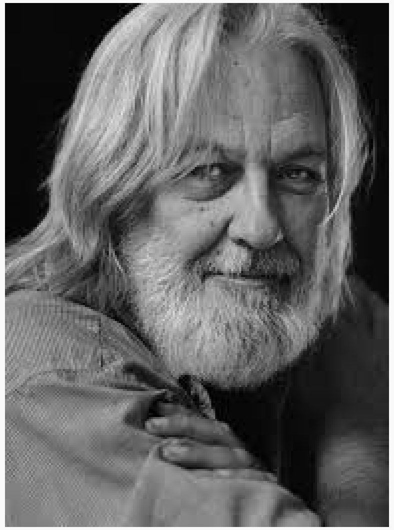
$$O(E|f^*|)$$

FORD-FULKERSON (G, s, t)
for each edge $(u, v) \in G.E$
$\quad (u, v).f = 0$
while there exists a path p from s to t in the residual network G_f
$\quad c_f(p) = \min\{c_f(u, v) : (u, v)\ is\ in\ p\}$
$\quad$ **for** each edge (u, v) in p
$\quad\quad$ **if** $(u, v) \in G.E$
$\quad\quad\quad (u, v).f = (u, v).f + c_f(p)$
$\quad\quad$ **else**
$\quad\quad\quad (v, u).f = (v, u).f - c_f(p)$



Algoritmul Edmonds–Karp (1972)

- Dacă alegem p folosind BFS



$$O(VE^2)$$

Există și algoritmi cu complexitate mai bună: $O(V^3)$ și $O(VE)$

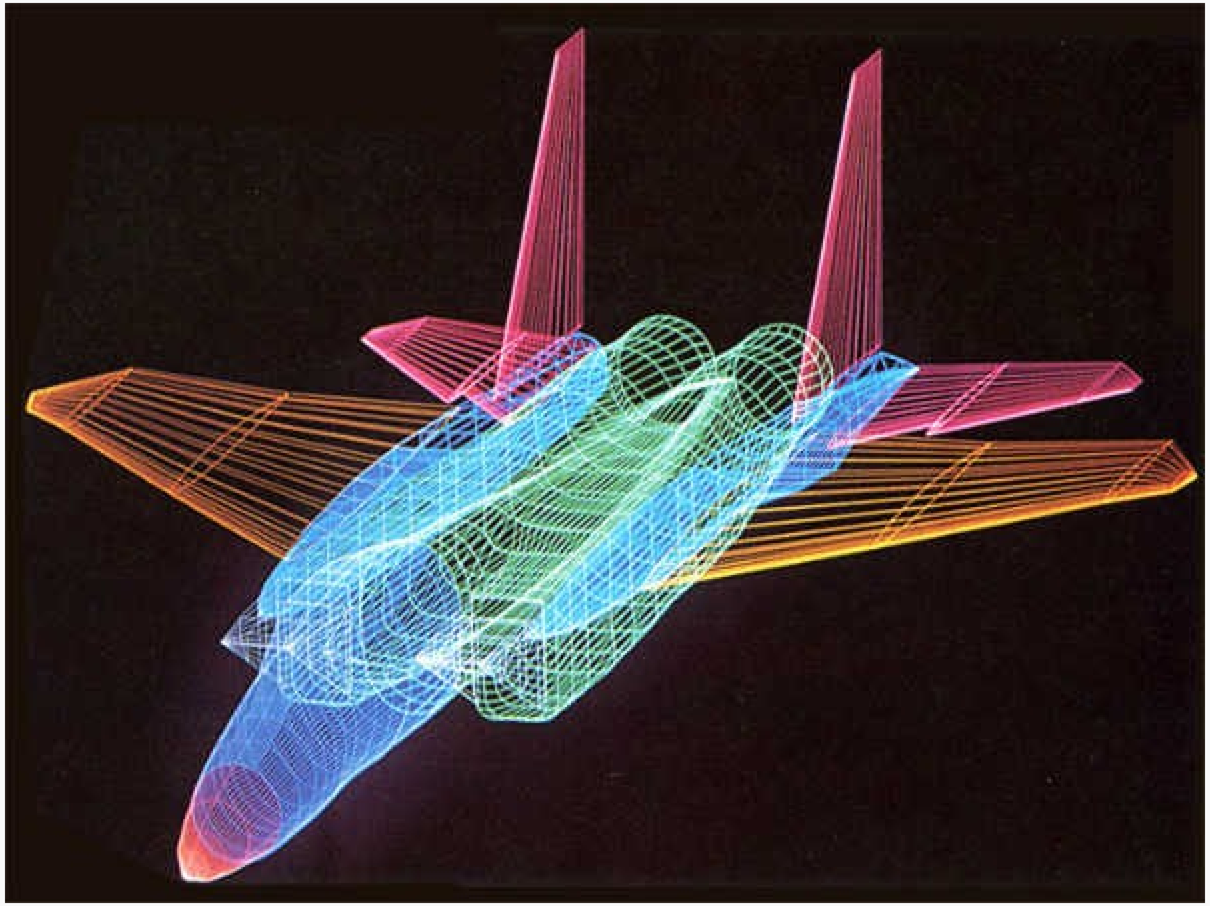
Alte considerente grafuri

- Se pot schimba în timp.

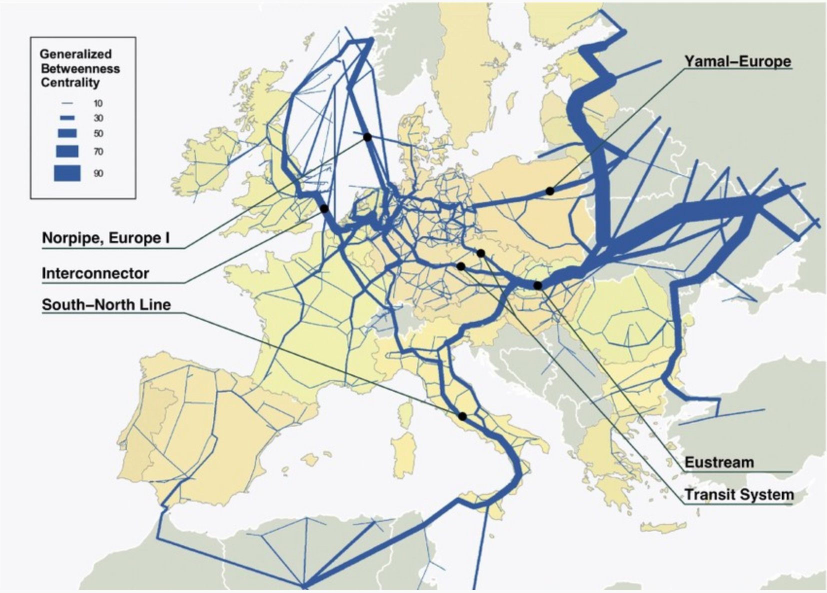
- LineGraph – Pentru un graf non-direcțional muchiile devin noduri și nodurile muchii.

Use case grafuri – Granițe

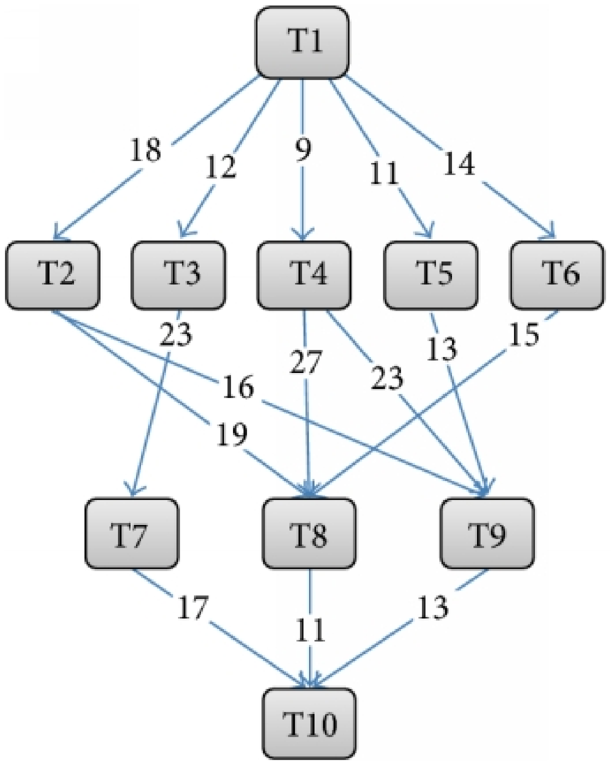
Use case grafuri – Grafică calculator



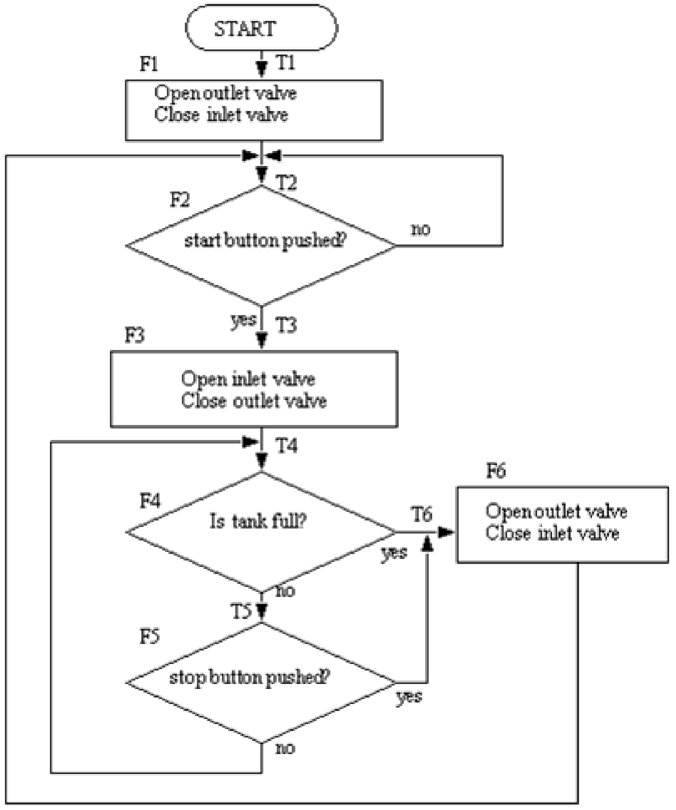
Use caser grafuri – utilități



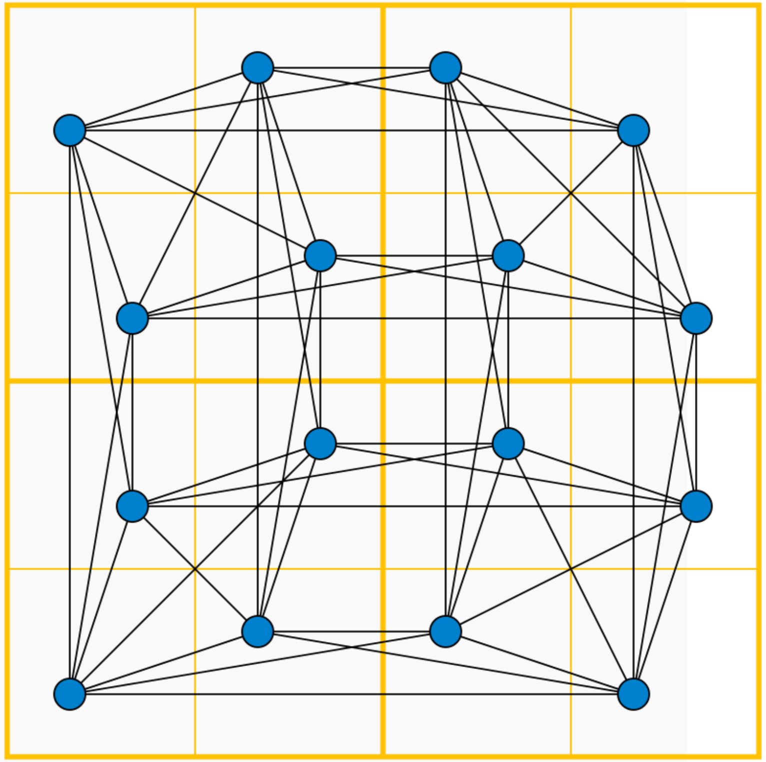
Use case grafuri – Code

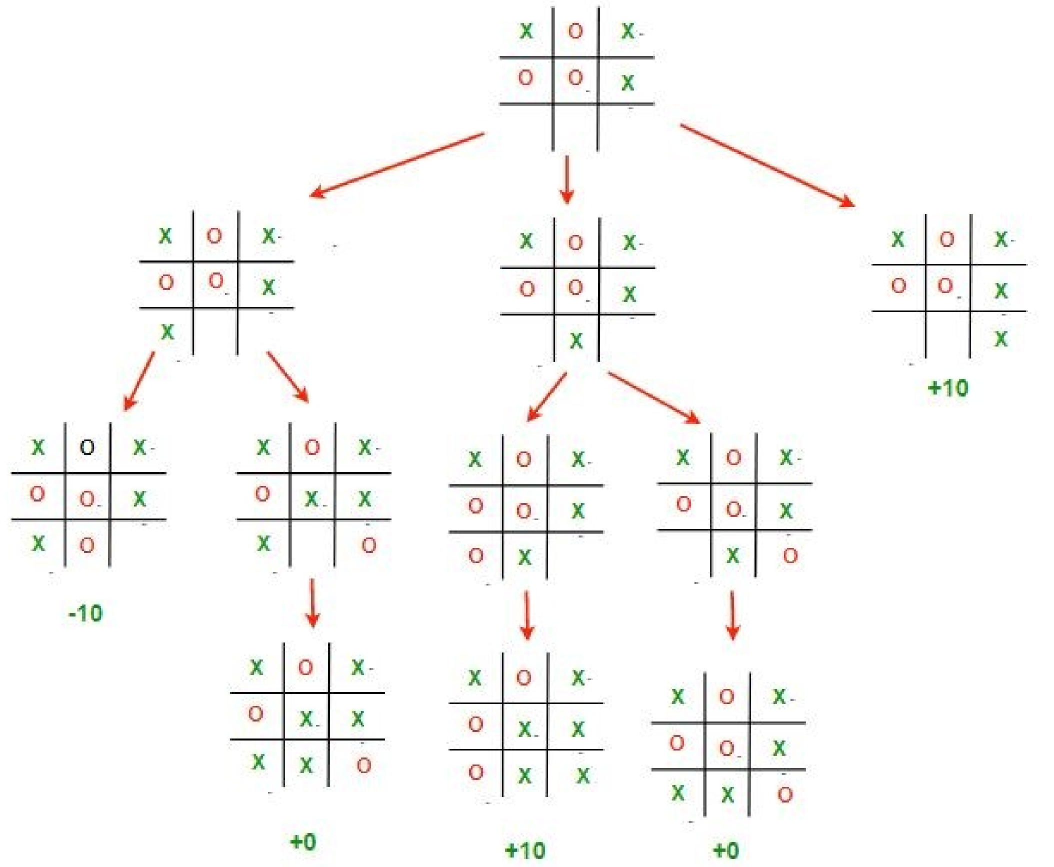


Task	P_1	P_2	P_3
T1	14	16	9
T2	13	19	18
T3	11	13	19
T4	13	8	7
T5	12	13	10
T6	13	16	9
T7	7	15	11
T8	5	11	14
T9	18	12	20
T10	21	7	16

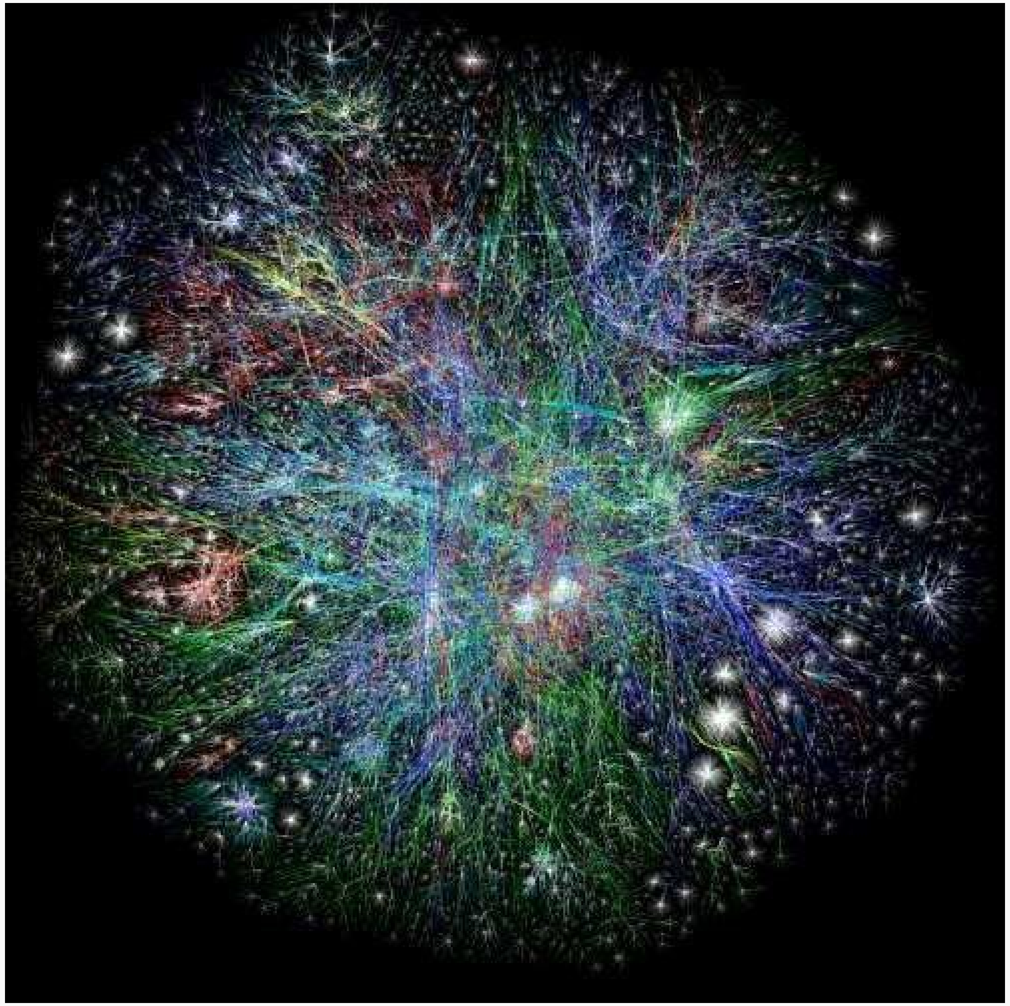


Use case grafuri – Reprezentare Jocuri

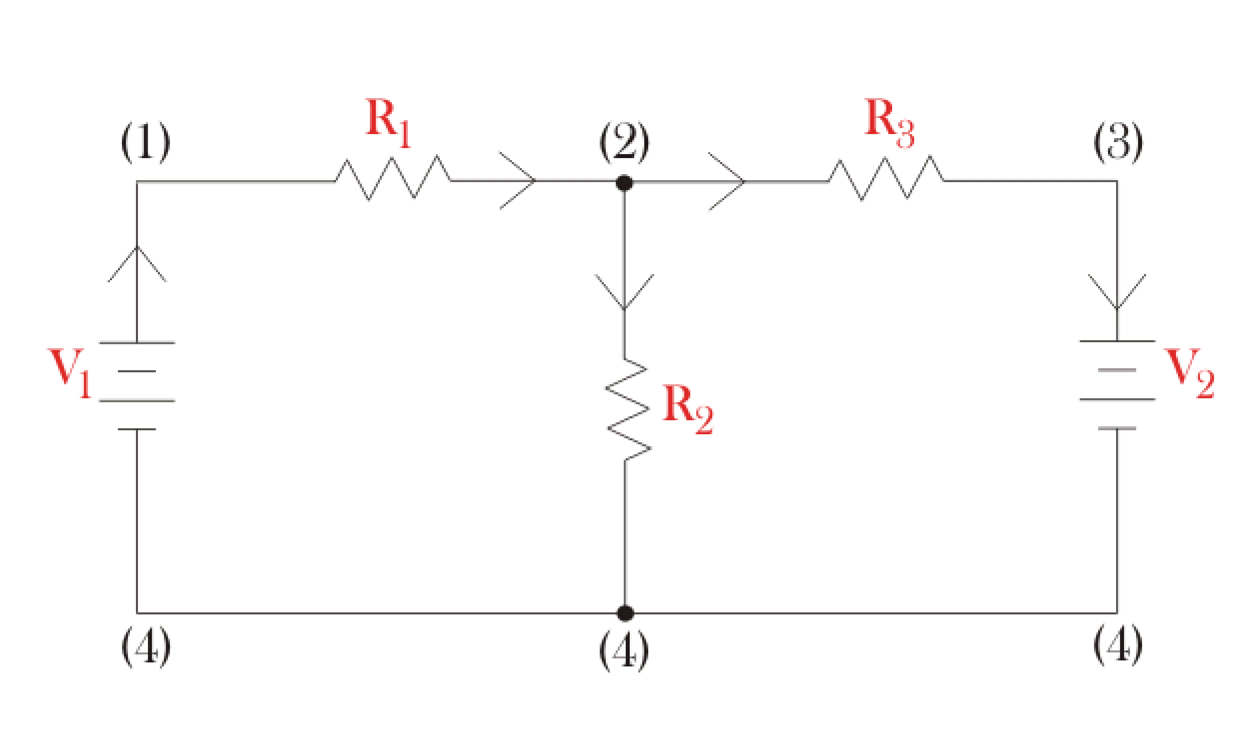




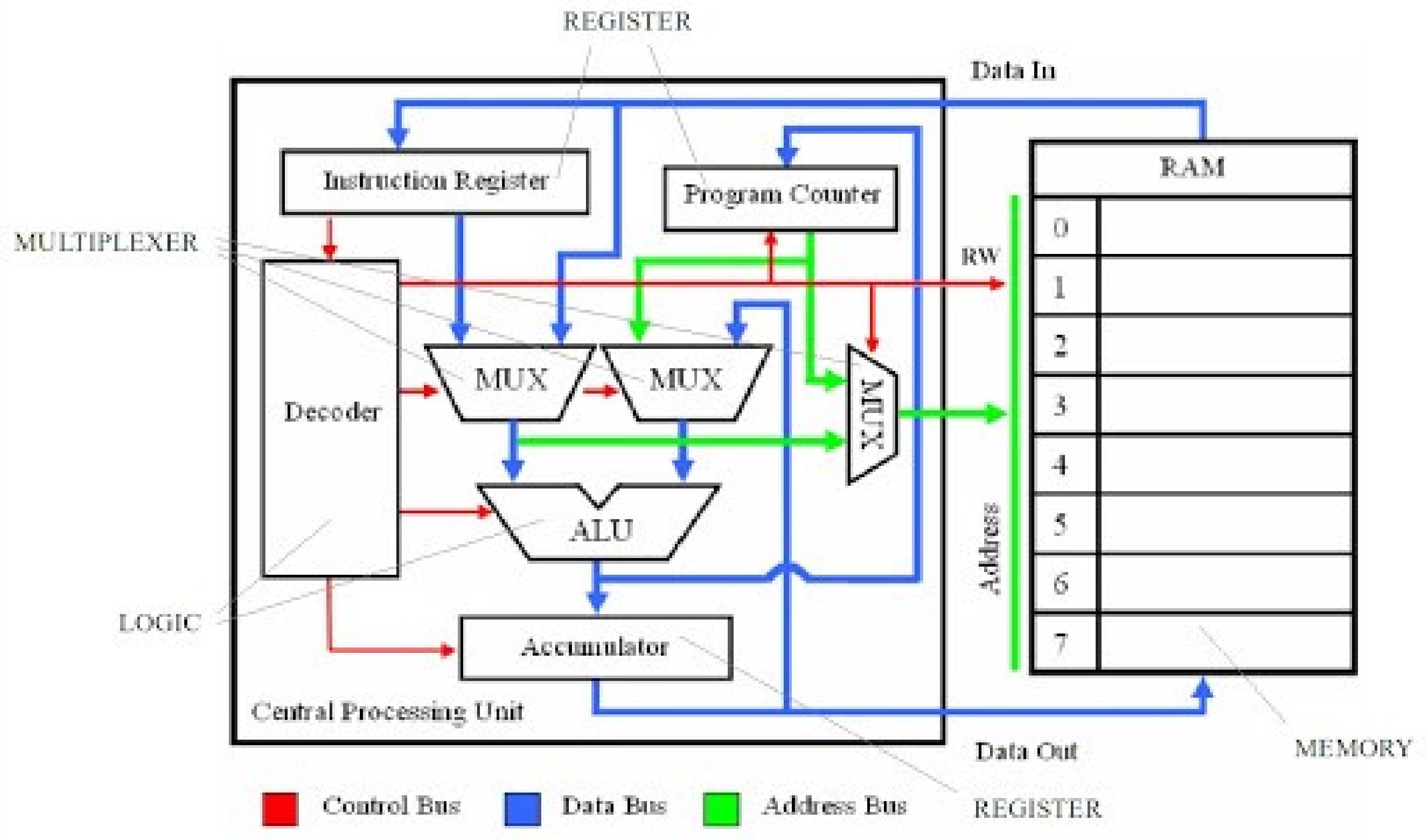
Use case grafuri - Internet



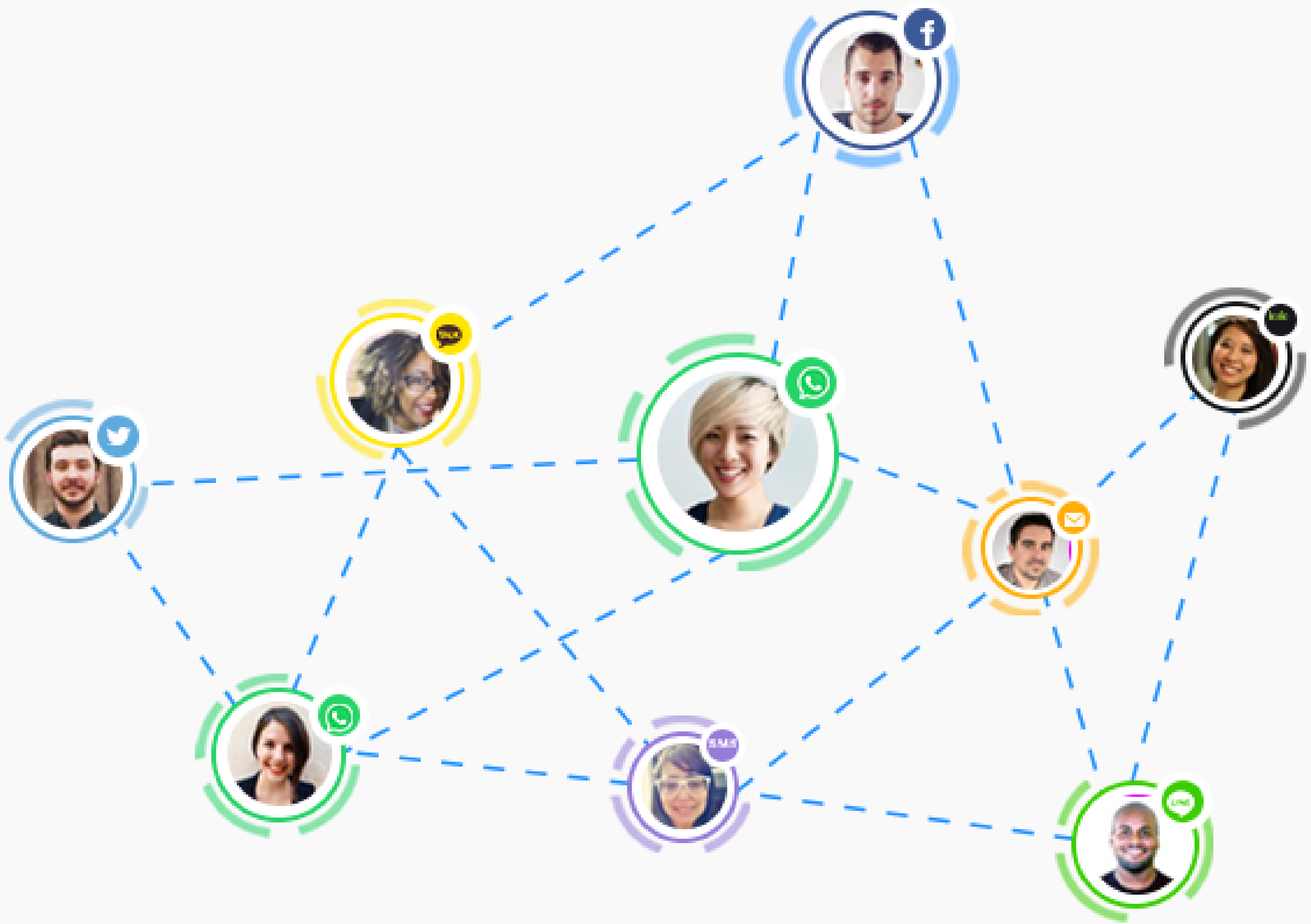
Use case grafuri – Circuite electrice



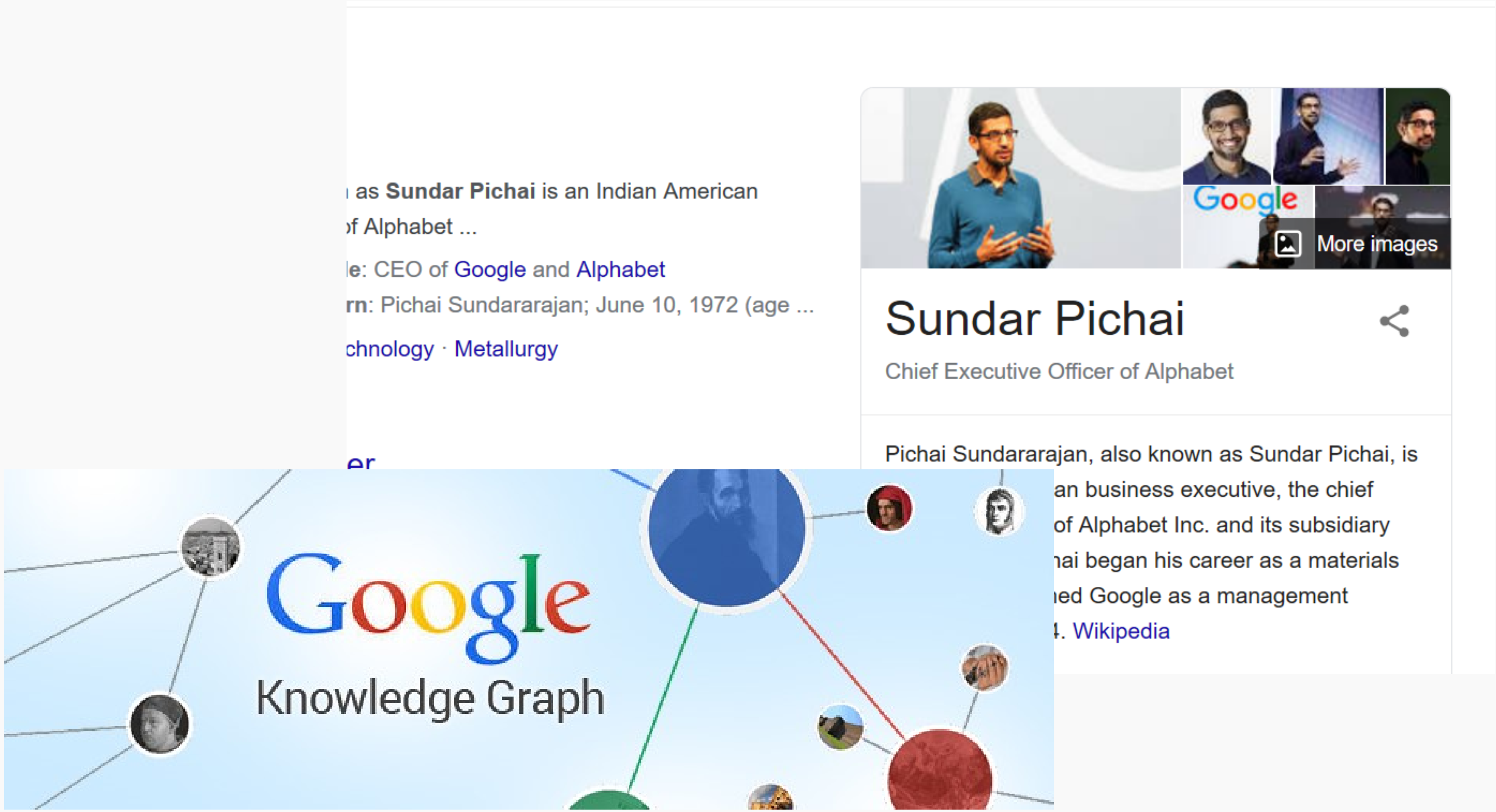
Use case grafuri – Circuite logice



Use case grafuri – Grafuri sociale



Use case-uri – Knowledge graph



Use case-uri – Organigrame