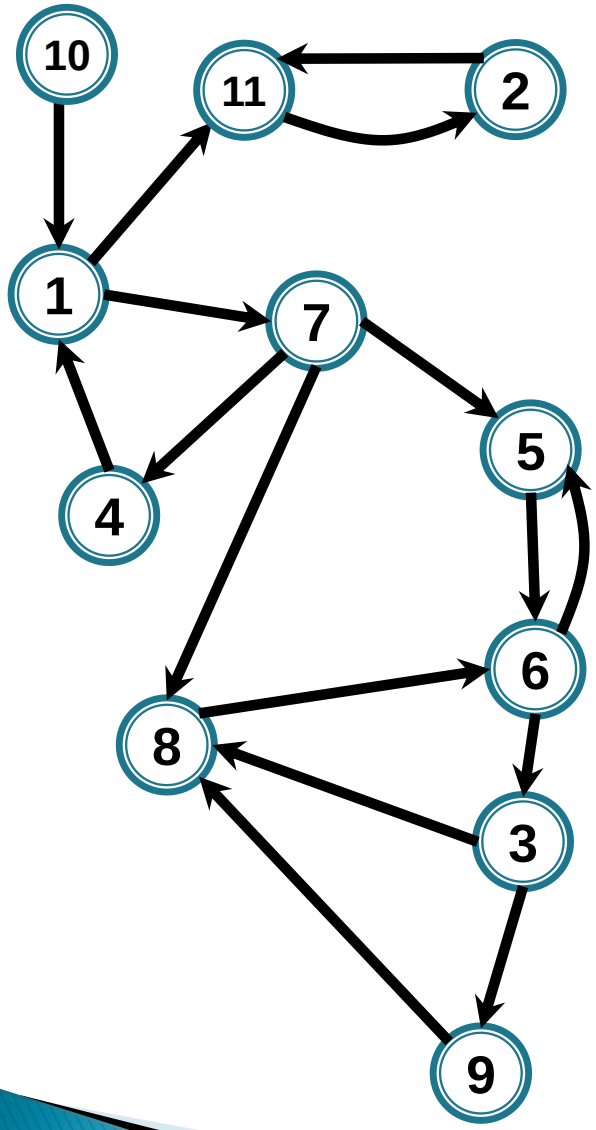
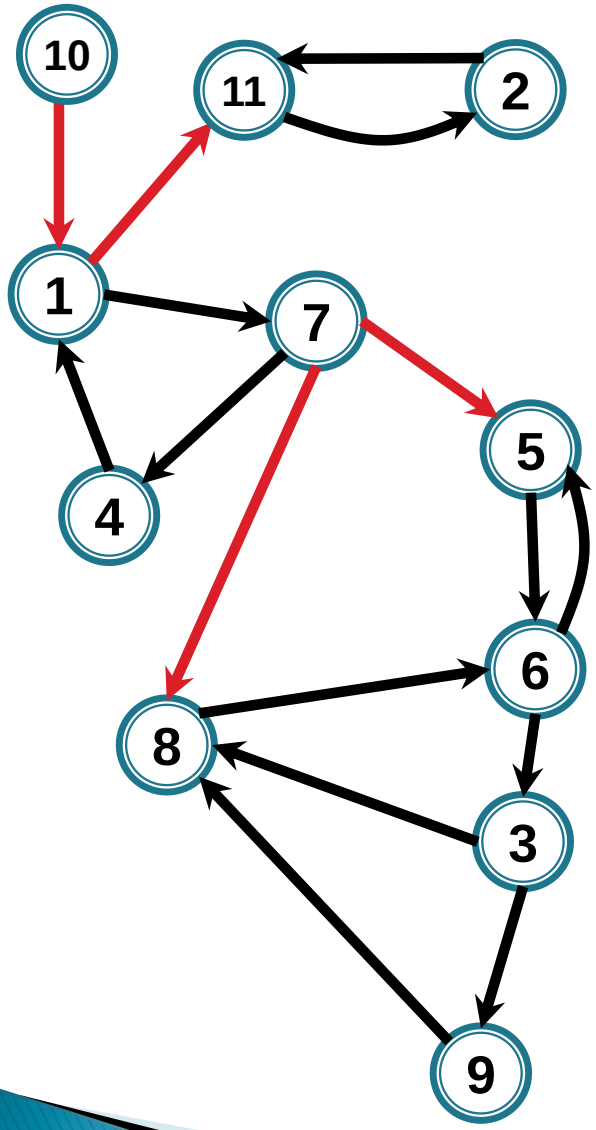
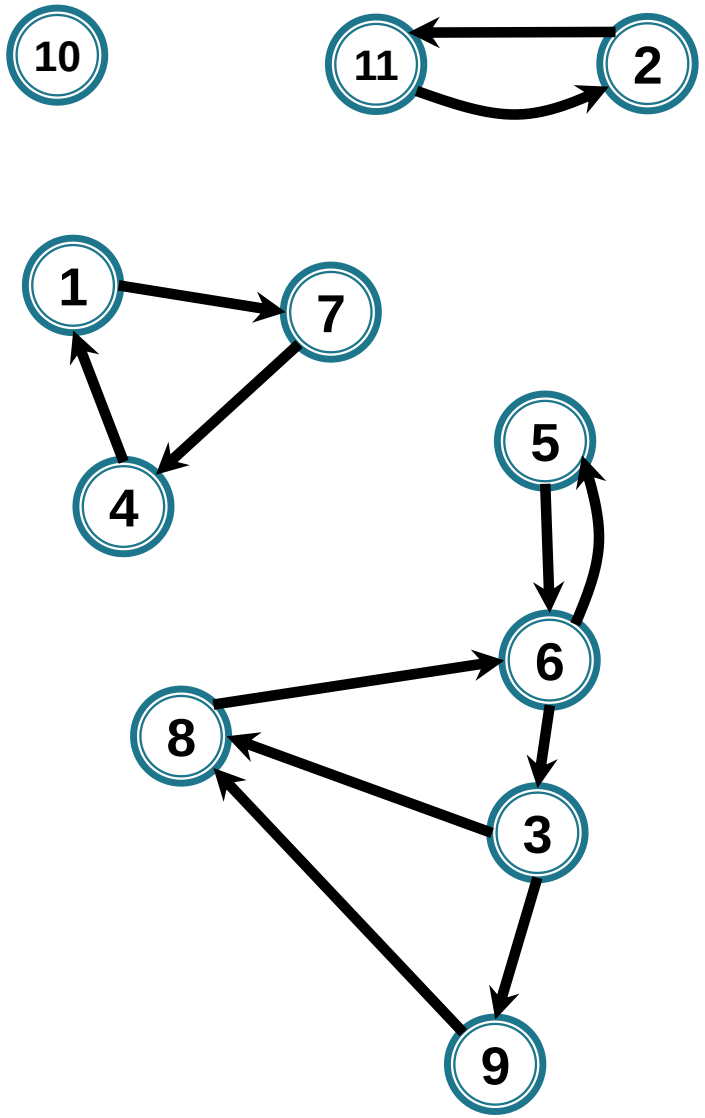
Componente tare conexe

Componente tare conexe



Componente tare conexe





Componentele tare conexe

Algoritmi componente tare conexe

- **Folosind mai multe parcurgeri?**

Posibilă idee:

componenta(x) =

multimea vârfurilor accesibile din x în G ∩

multimea vârfurilor accesibile din x în G^T

unde $G^T = (V, E^T)$, $E^T = \{yx \mid yx \in E\}$

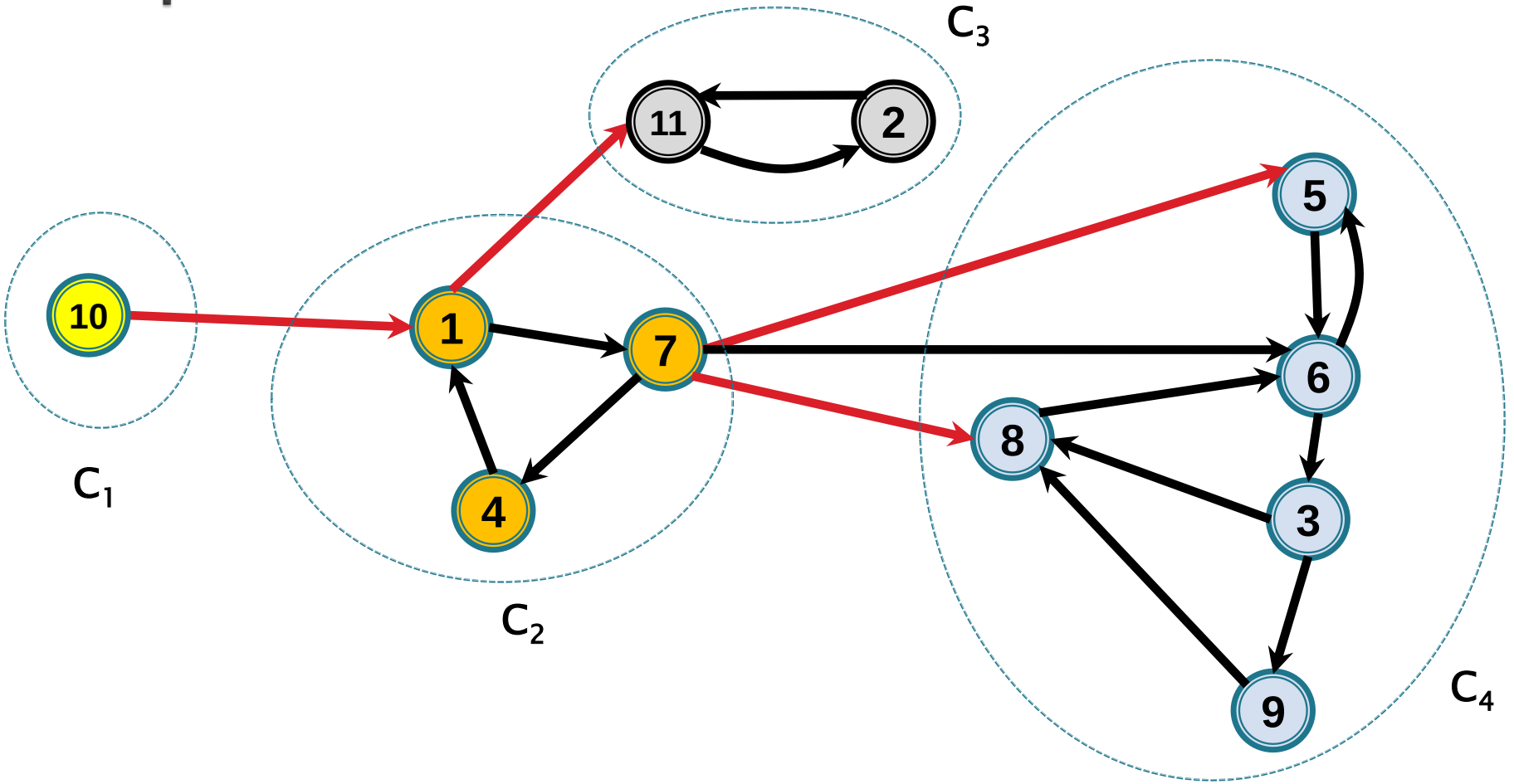
Algoritmi componente tare conexe

- **Observație**

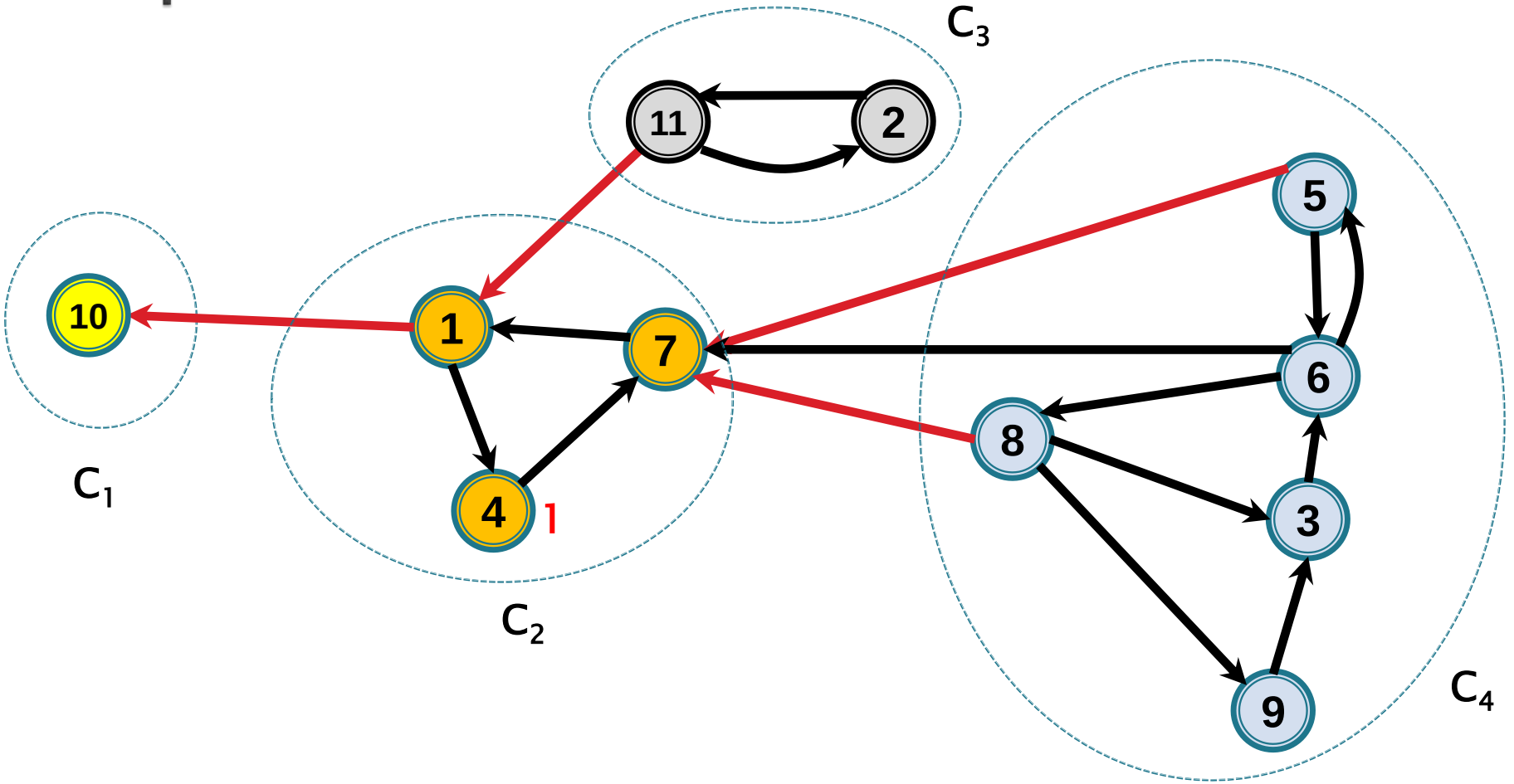
componente tare conexe din G =

componente tare conexe din G^T

Componente tare conexe



Componente tare conexe



G^T – ar fi bine sa determinăm întâi componenta lui 10: DF(G^T,10)
(sa nu se "amestece" componentele)

Algoritmi componente tare conexe

- Folosind doar două parcurgeri, una în G și una în G^T?

  DA, dar a doua într-o ordine particulară a vârfurilor (în funcție de ordinea în care au fost finalizate în DF)

  $\Rightarrow$ Algoritmul lui Kosaraju

Algoritmi componente tare conexe

- **Folosind o singură parcurgere?**

  DA, folosind o idee similară cu cea de la componente biconexe

  $\Rightarrow$ **Algoritmul lui Tarjan**

Algoritmul lui Kosaraju

Algoritmul lui Kosaraju

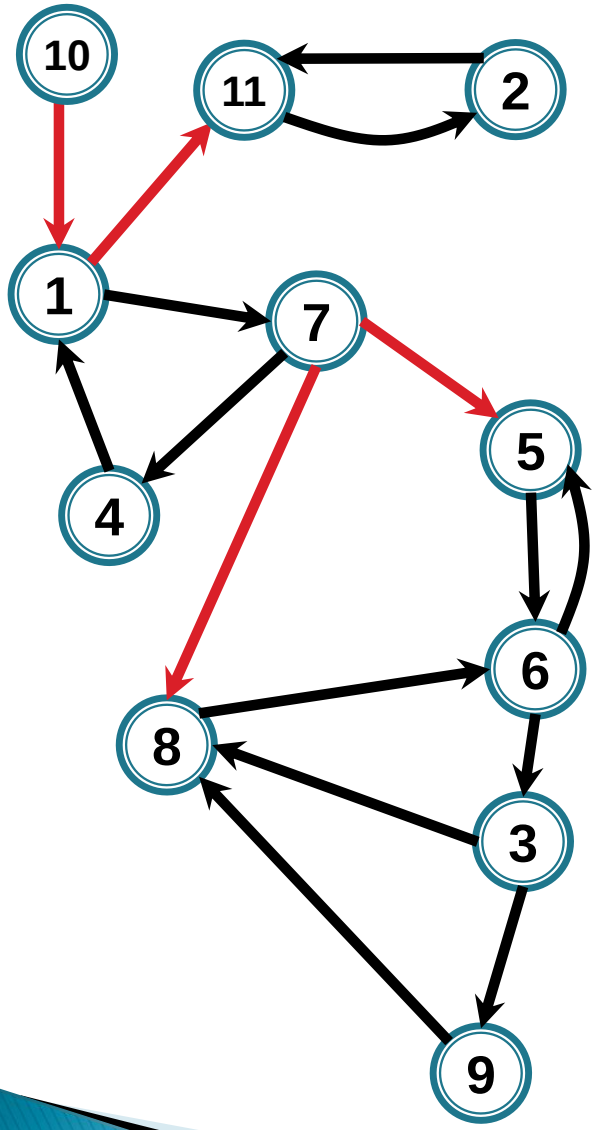
Pasul 1. DF(G,x) pentru fiecare vârf x nevizitat + o stivă S în care se introduc vârfurile când sunt finalizate

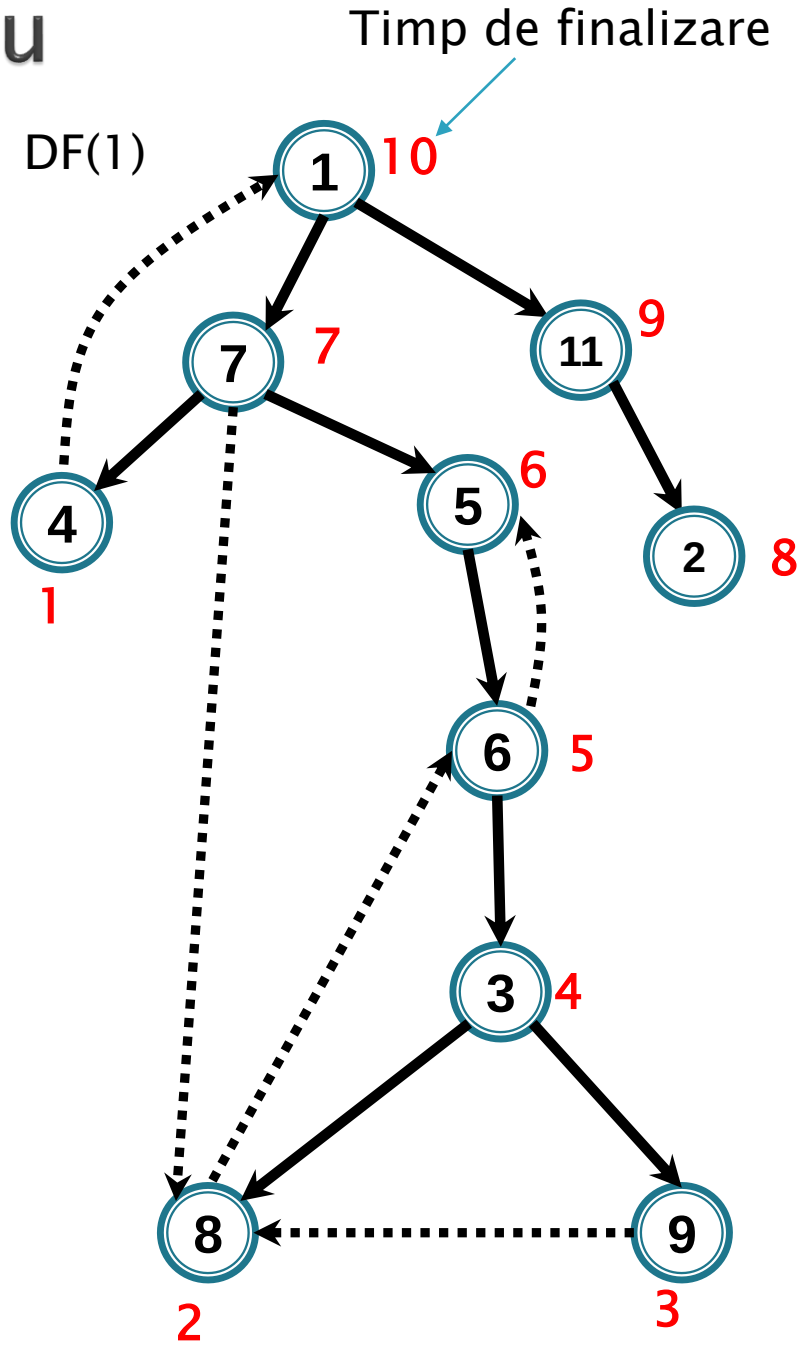
Pasul 2. DF(G^T,x) pentru x (nevizitat) în ordinea în care sunt scoase din S (=**descrescător în raport timpul de finalizare** – similar cu ?)

Avem

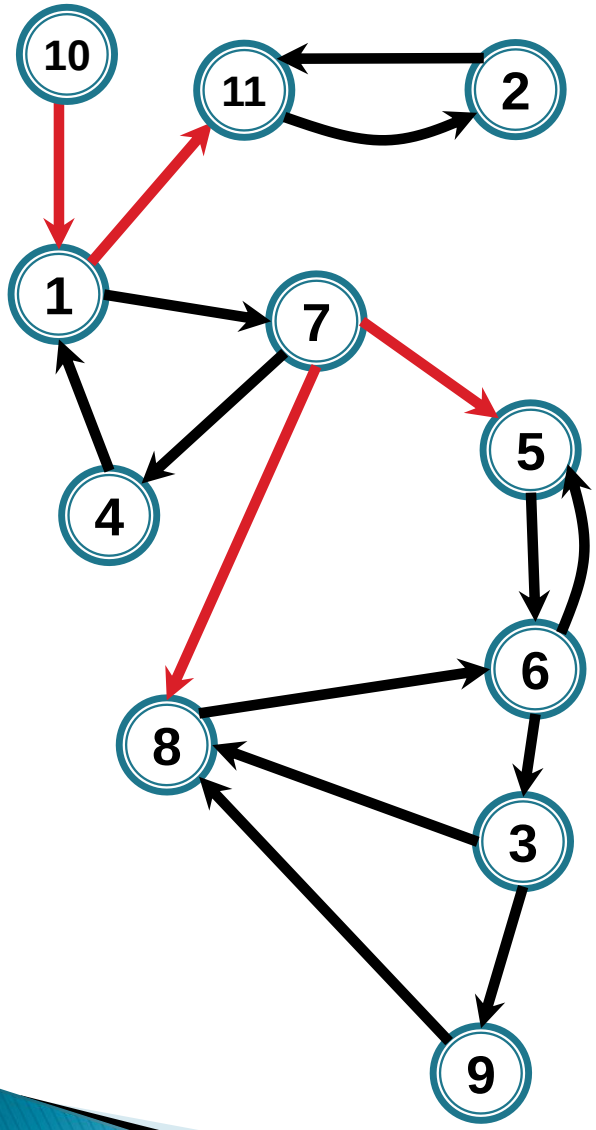
vârfuri vizitate în DF(G^T,x) = componenta tare conexă a lui x

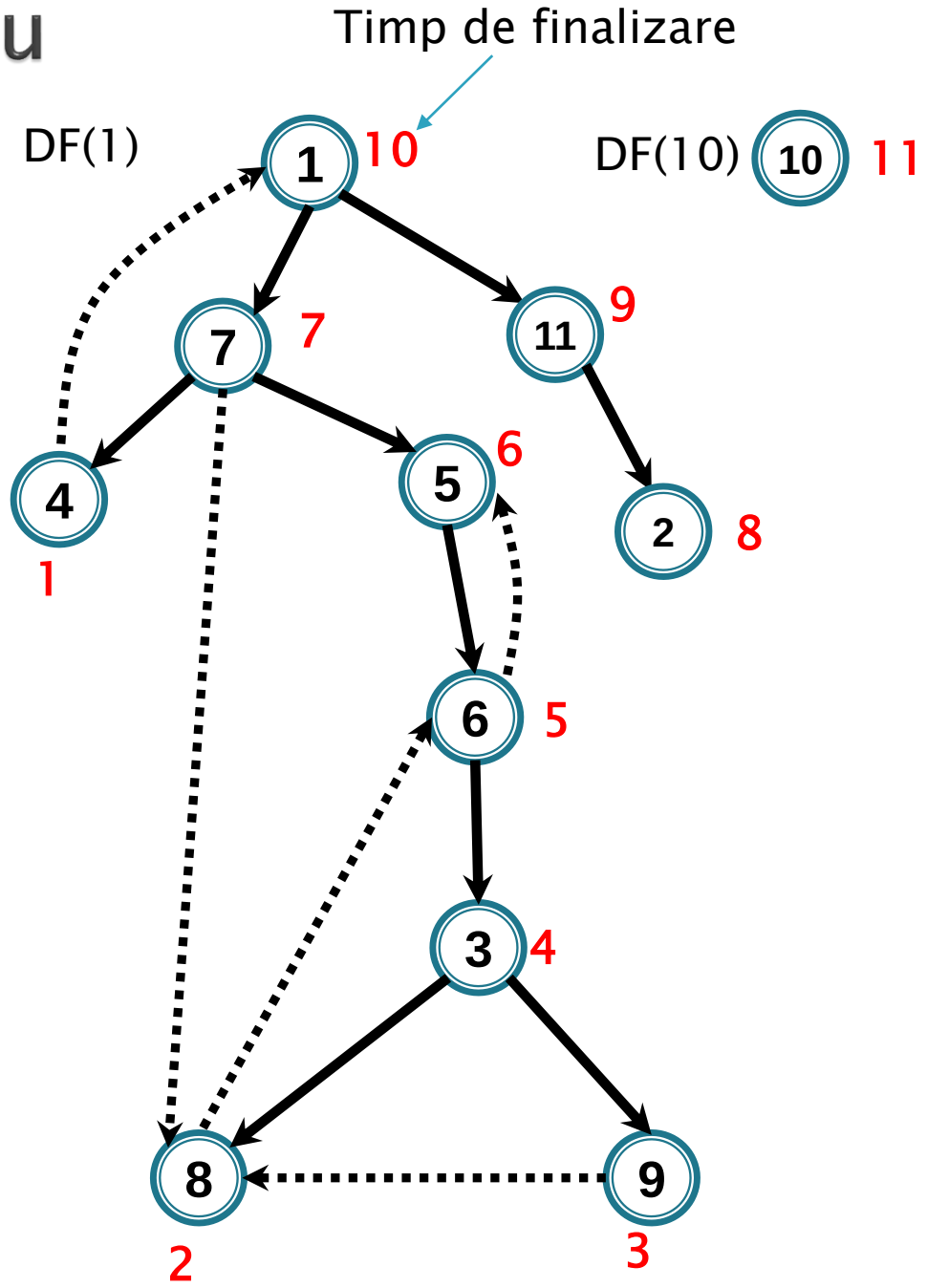
Algoritmul lui Kosaraju



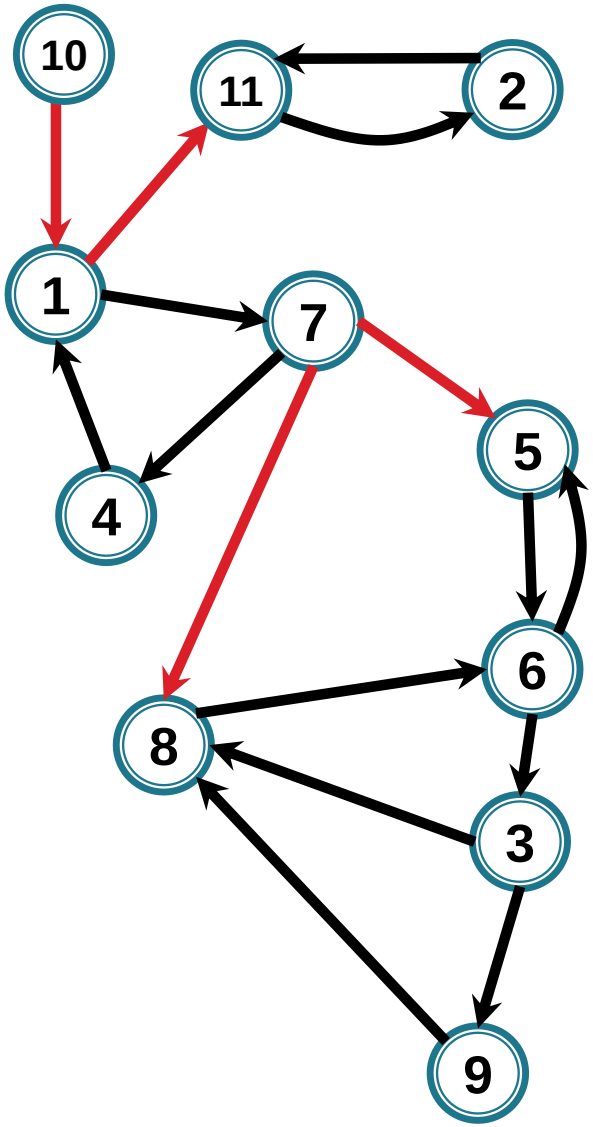


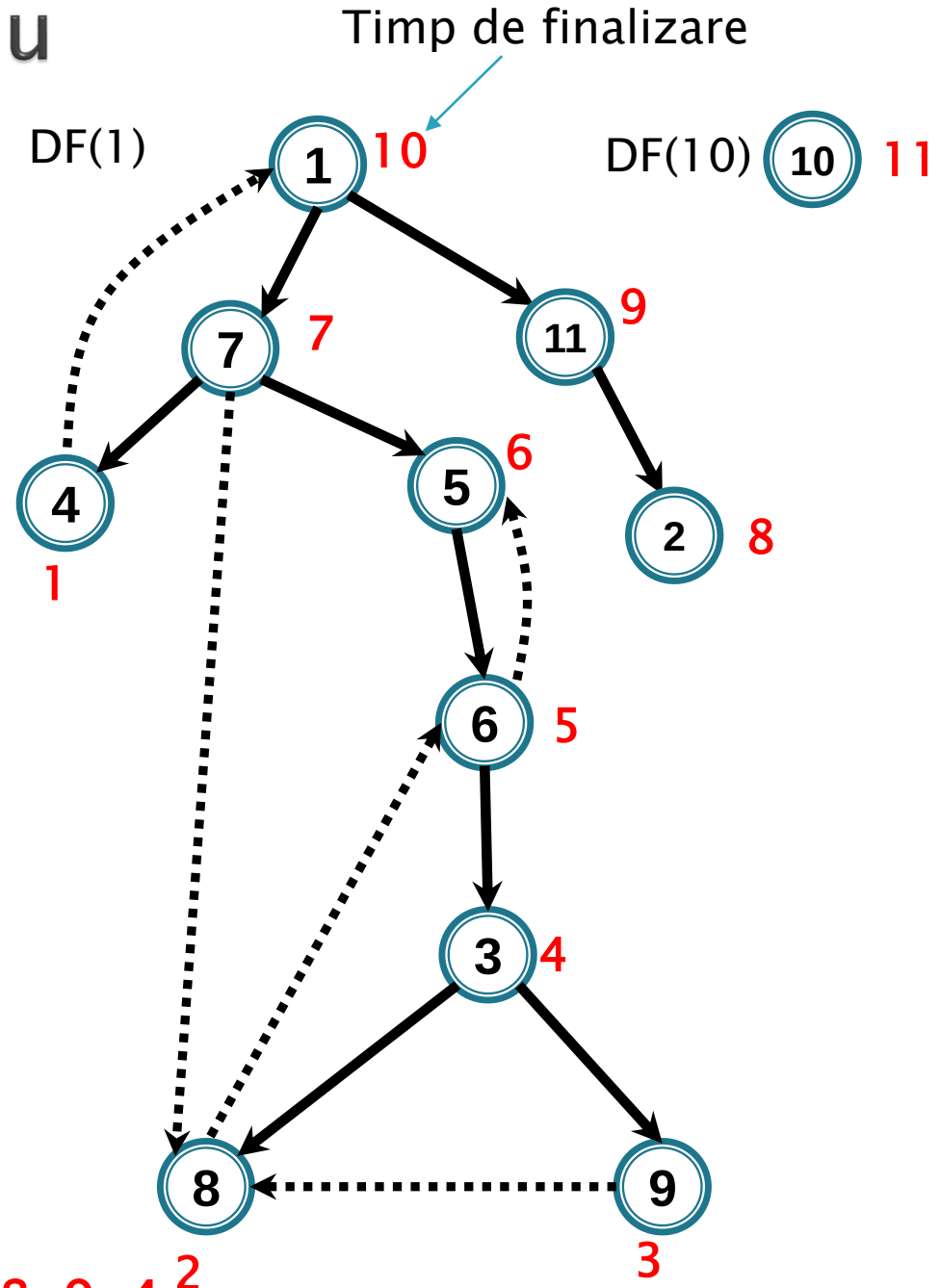
Algoritmul lui Kosaraju





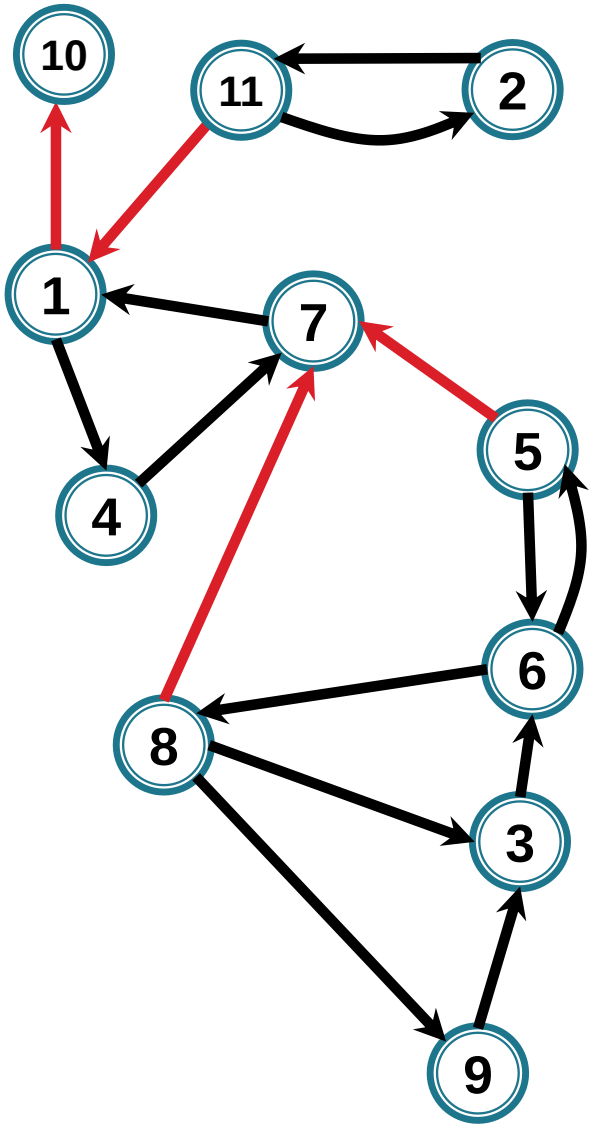
Algoritmul lui Kosaraju





Ordinea: 10, 1, 11, 2, 7, 5, 6, 3, 8, 9, 4

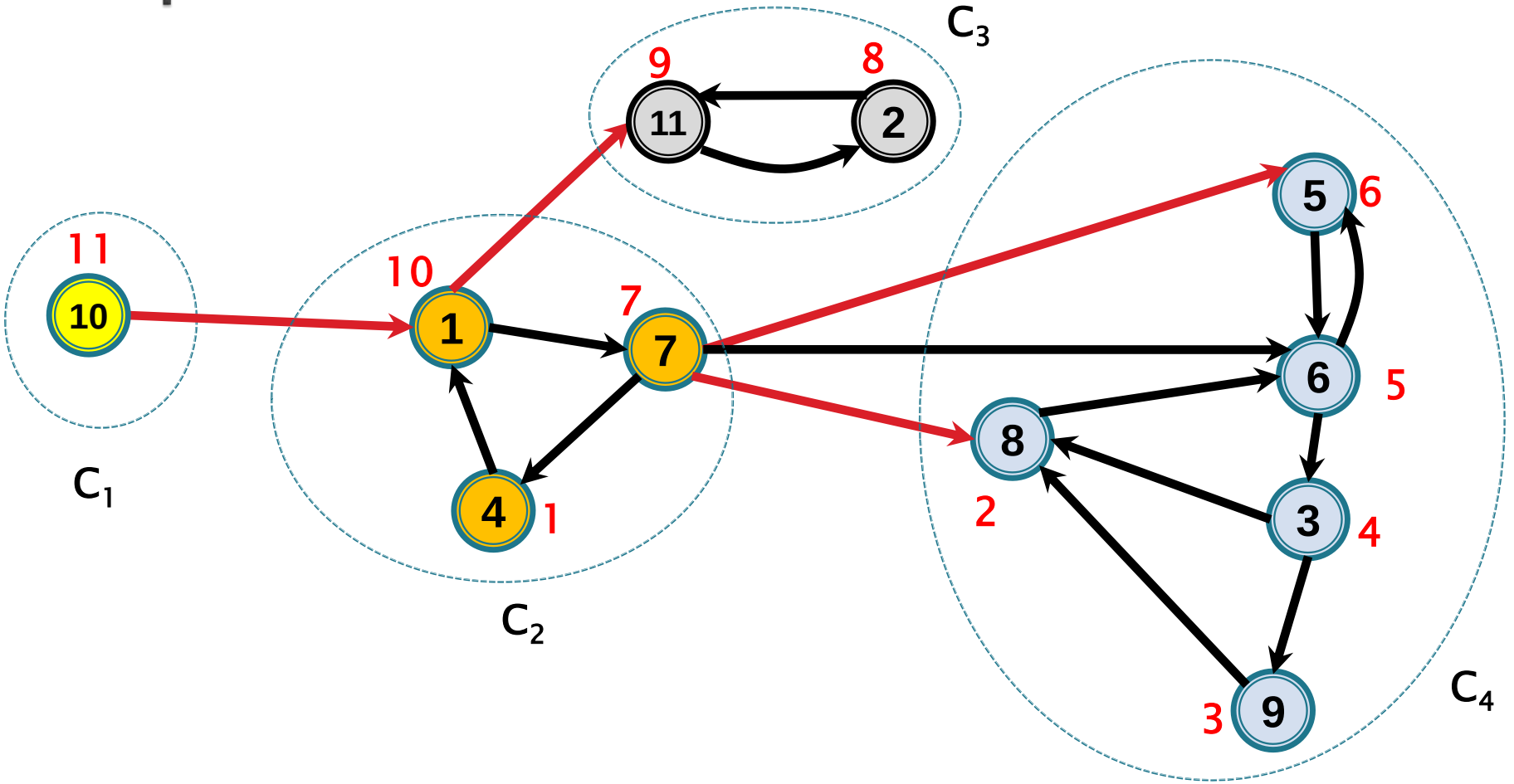
Algoritmul lui Kosaraju



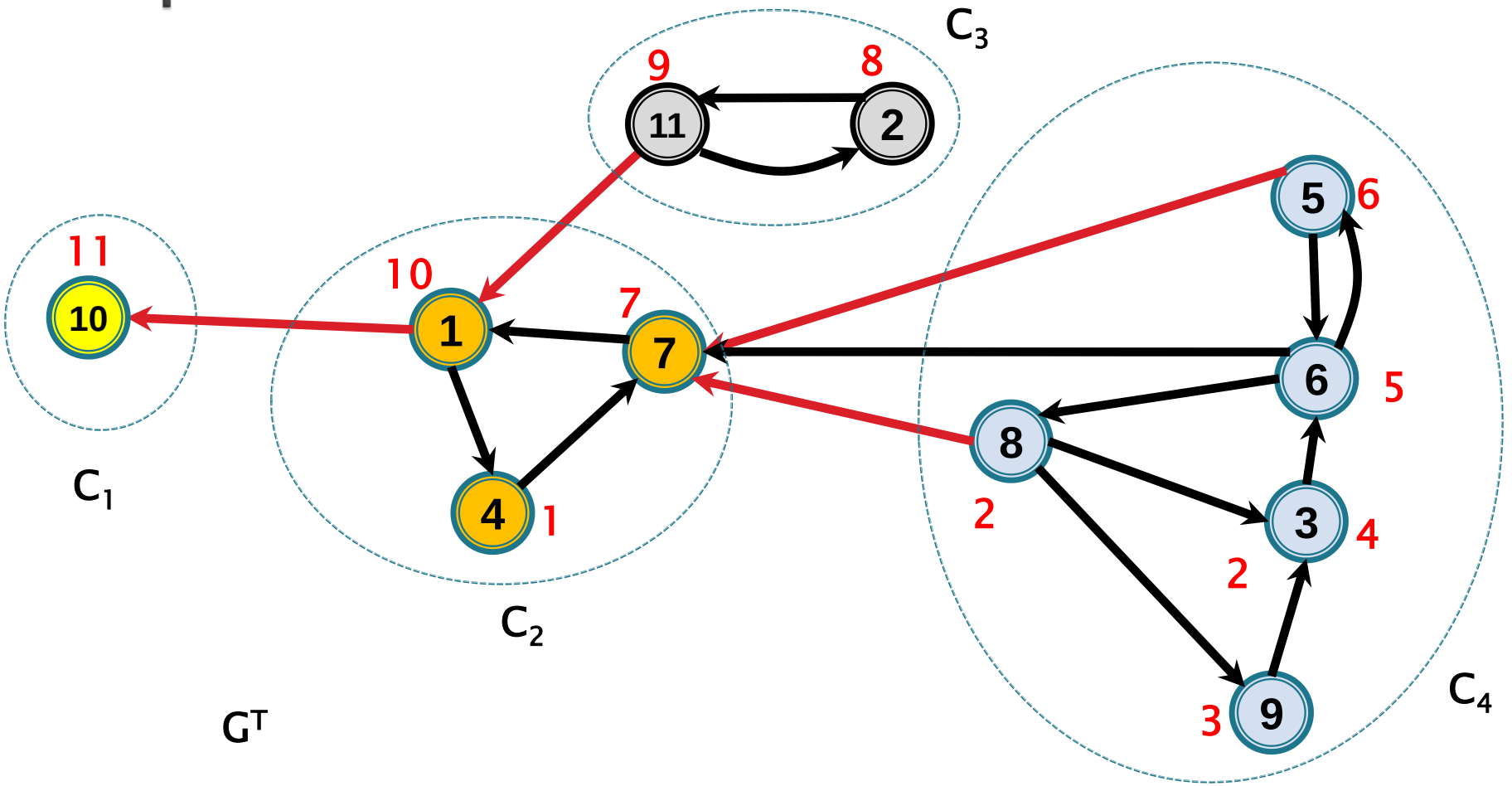
G^T

Ordinea: 10, 1, 11, 2, 7, 5, 6, 3, 8, 9, 4

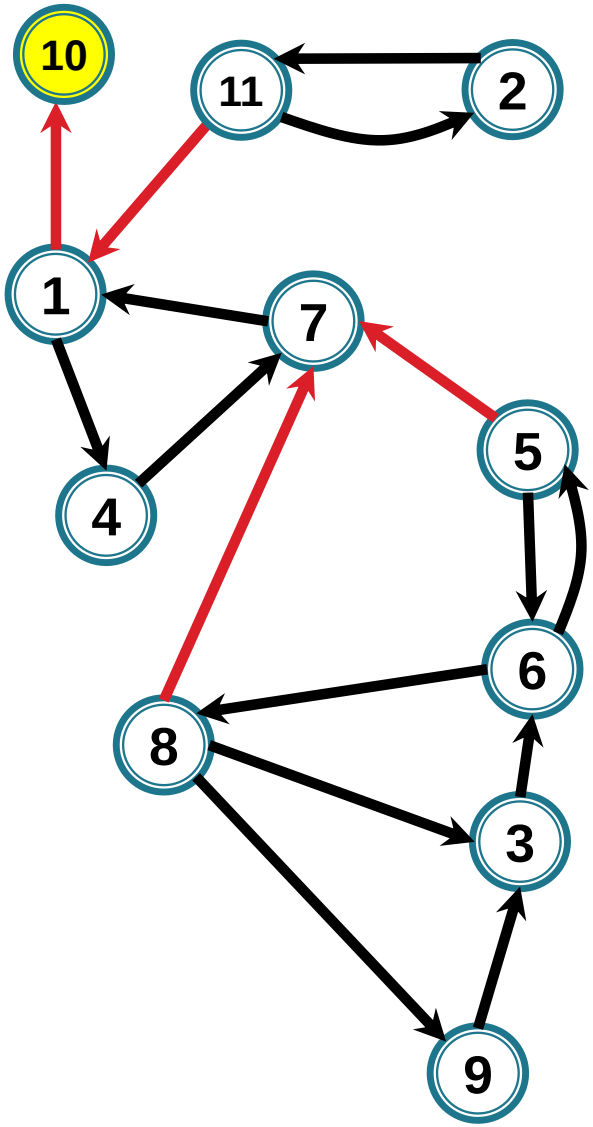
Componente tare conexe



Componente tare conexe



Algoritmul lui Kosaraju



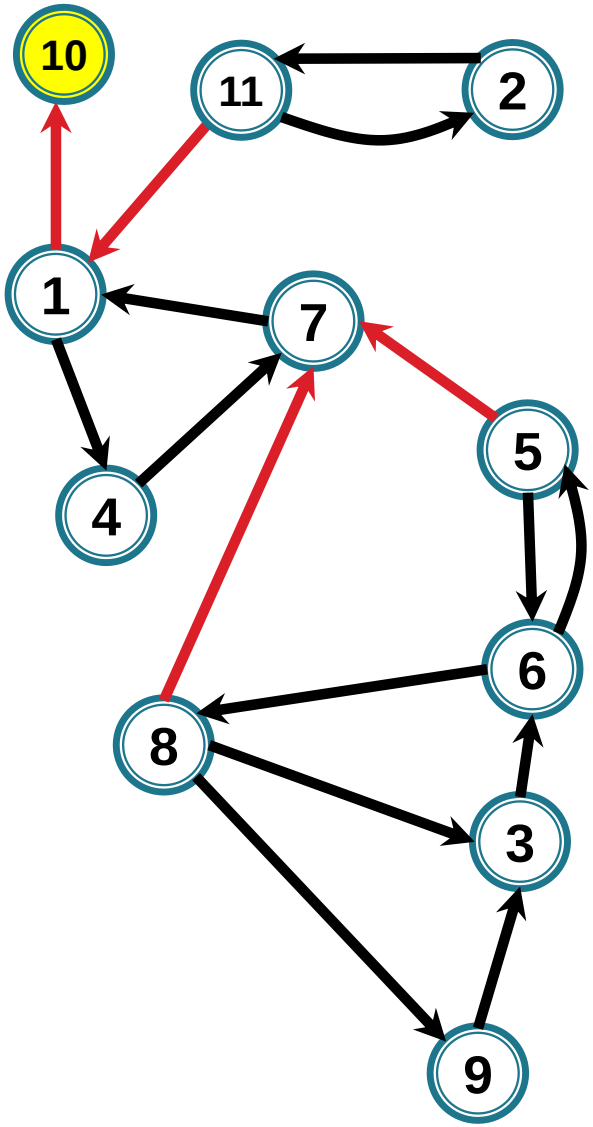
DF(10)

Componenta tare conexă: 10

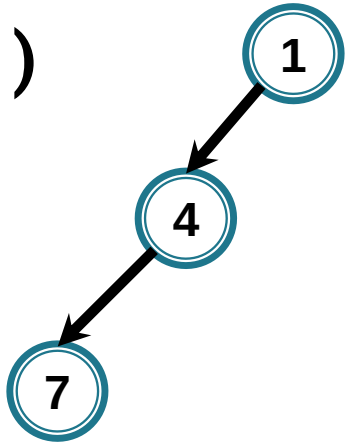
G^T

Ordinea: 10, 1, 11, 2, 7, 5, 6, 3, 8, 9, 4

Algoritmul lui Kosaraju



DF(1)

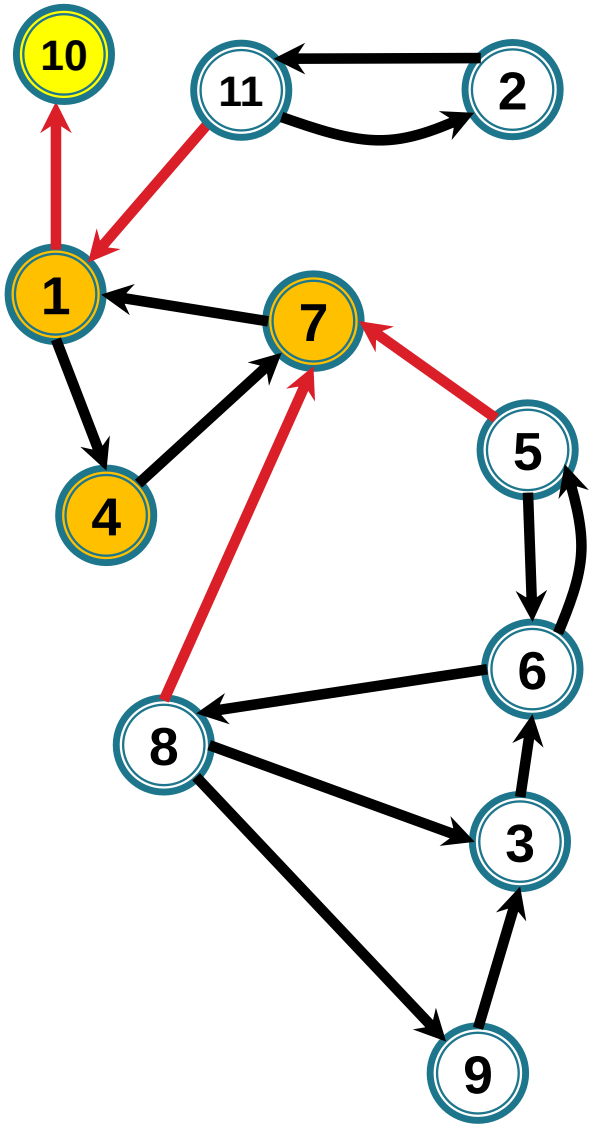


Componenta tare conexă: 1,4,7

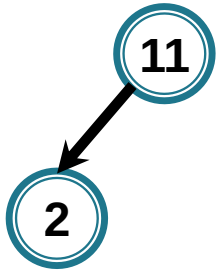
G^T

Ordinea: 10, 1, 11, 2, 7, 5, 6, 3, 8, 9, 4

Algoritmul lui Kosaraju



DF(11)

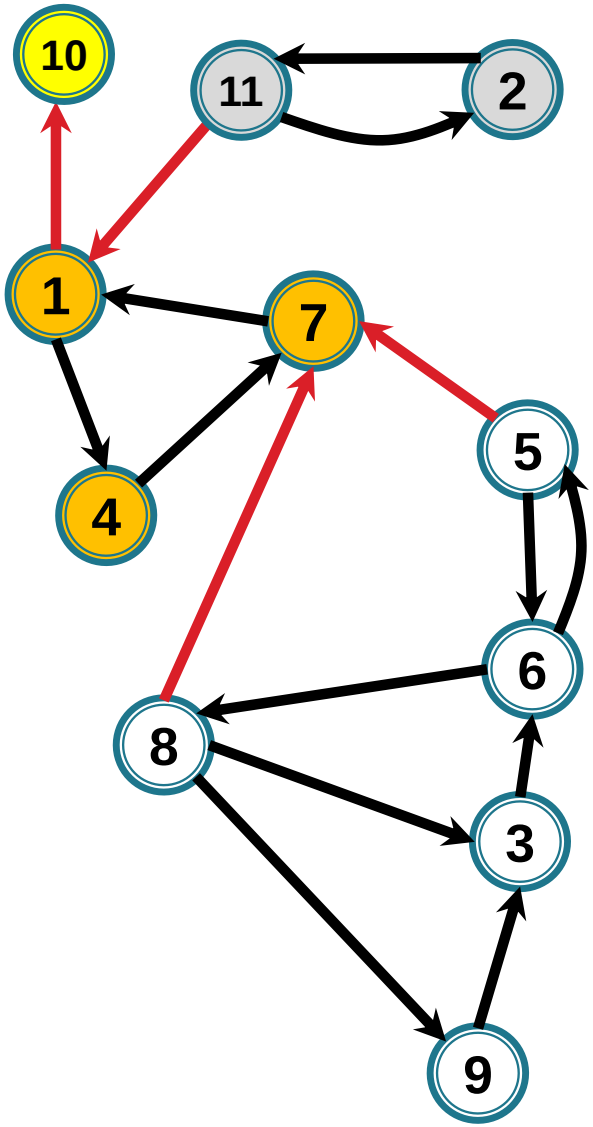


Componenta tare conexă: 11,2

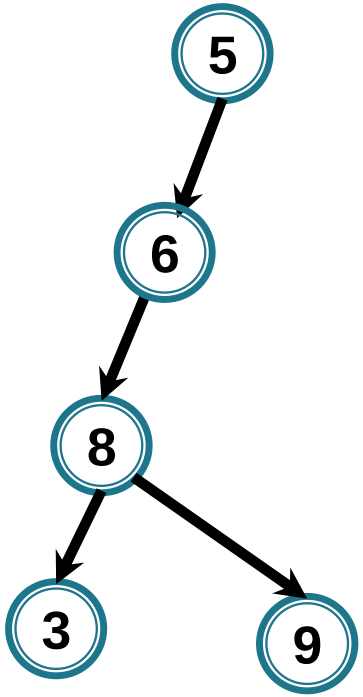
G^T

Ordinea: 10, 1, 11, 2, 7, 5, 6, 3, 8, 9, 4

Algoritmul lui Kosaraju



DF(5)

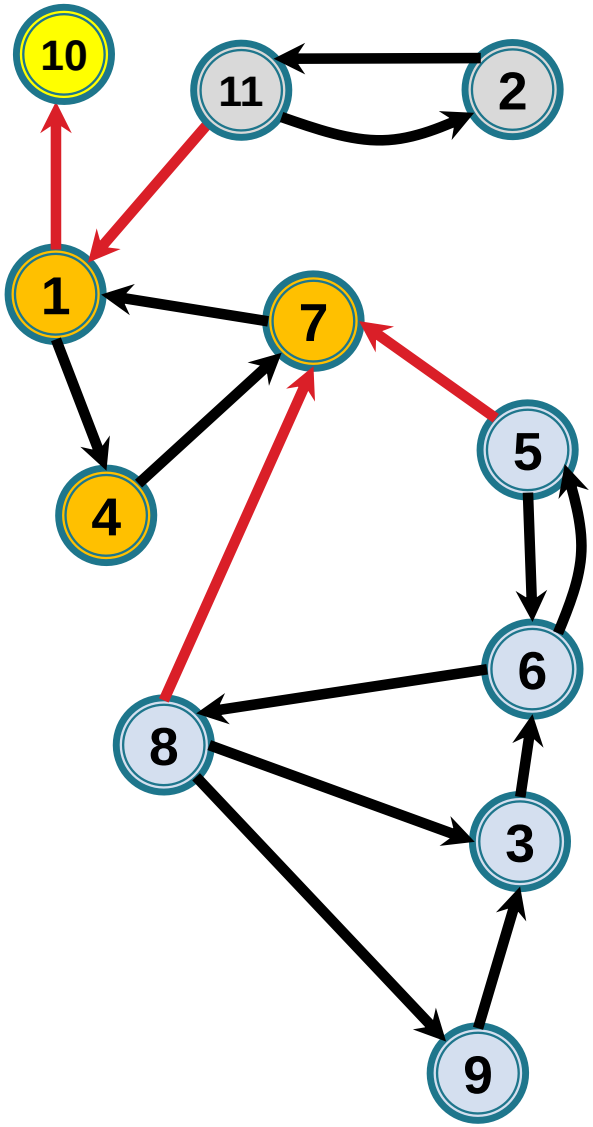


Componenta tare conexă: 5,6,8,3,9

G^T

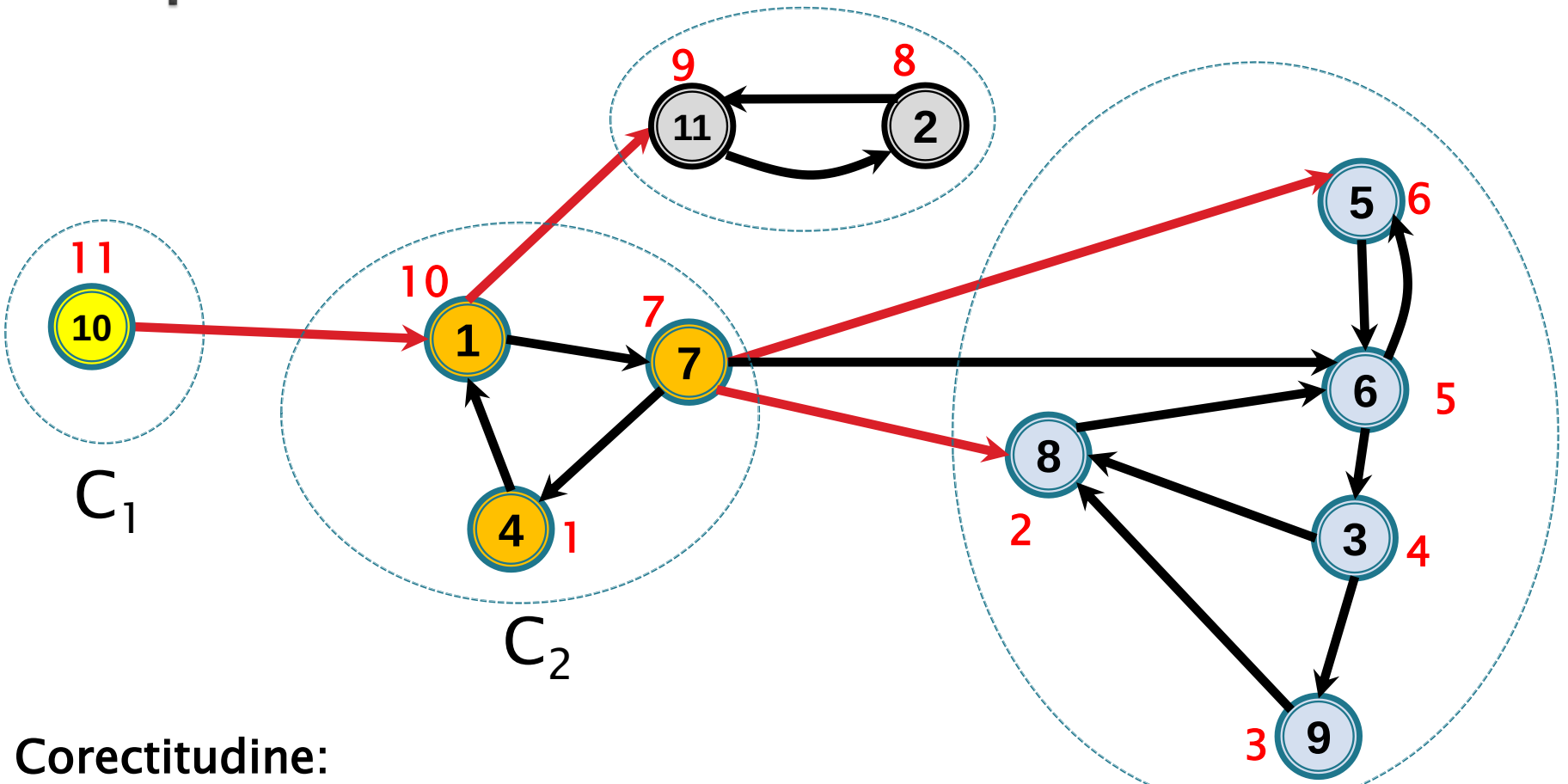
Ordinea: 10, 1, 11, 2, 7, 5, 6, 3, 8, 9, 4

Algoritmul lui Kosaraju



G^T

Componente tare conexe



Corectitudine:

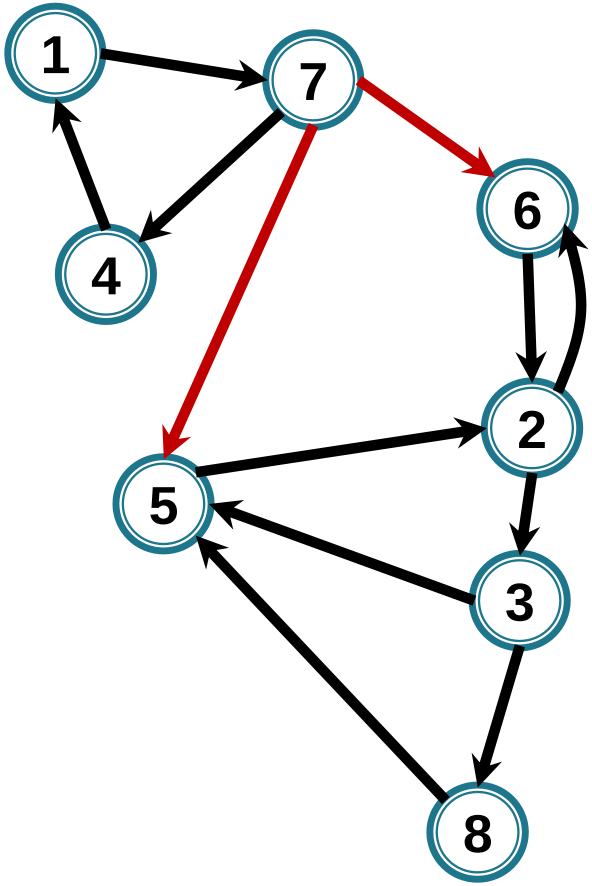
$$\max\{fin[u] \mid u \in C_1\} > \max\{fin[u] \mid u \in C_2\}$$

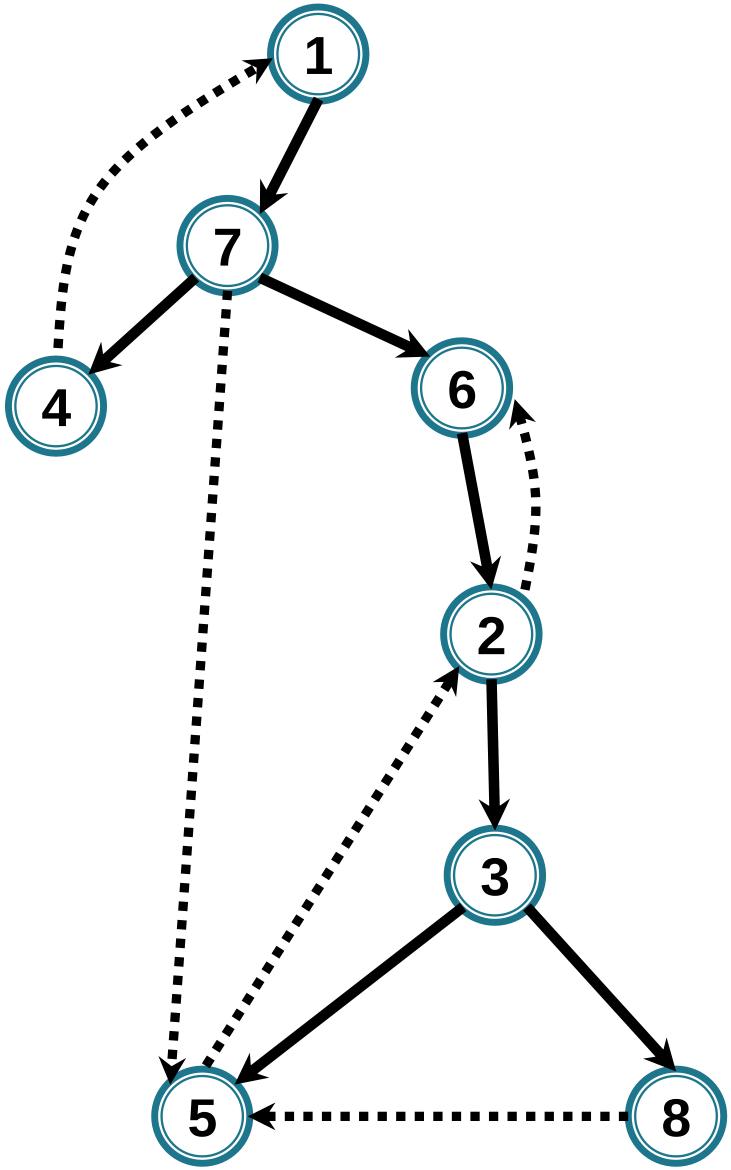
http://web.stanford.edu/class/archive/cs/cs161/cs161.1176/Slides/Lecture10.pdf

R. Sedgewick, K. Wayne – Algorithms, 4th Edition

Algoritmul lui Tarjan

Algoritmul lui Tarjan



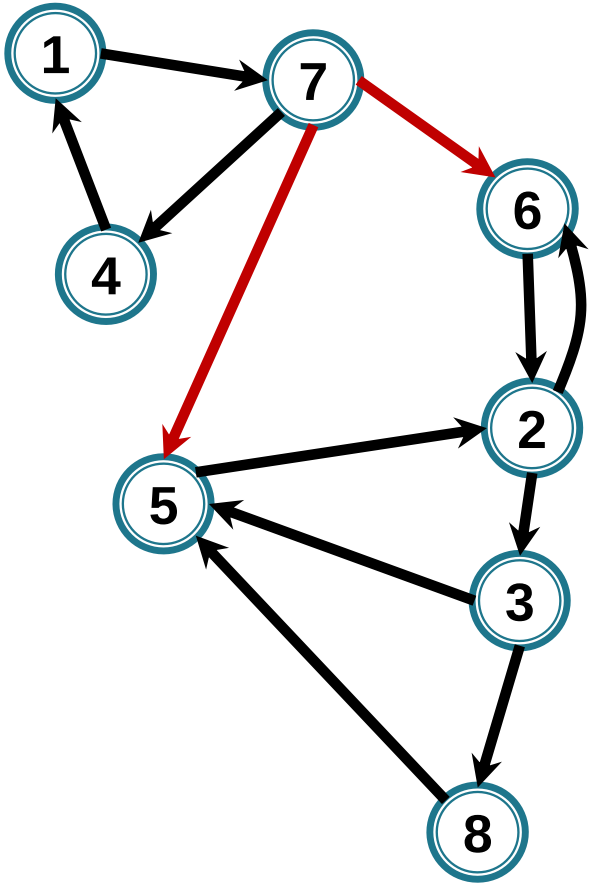


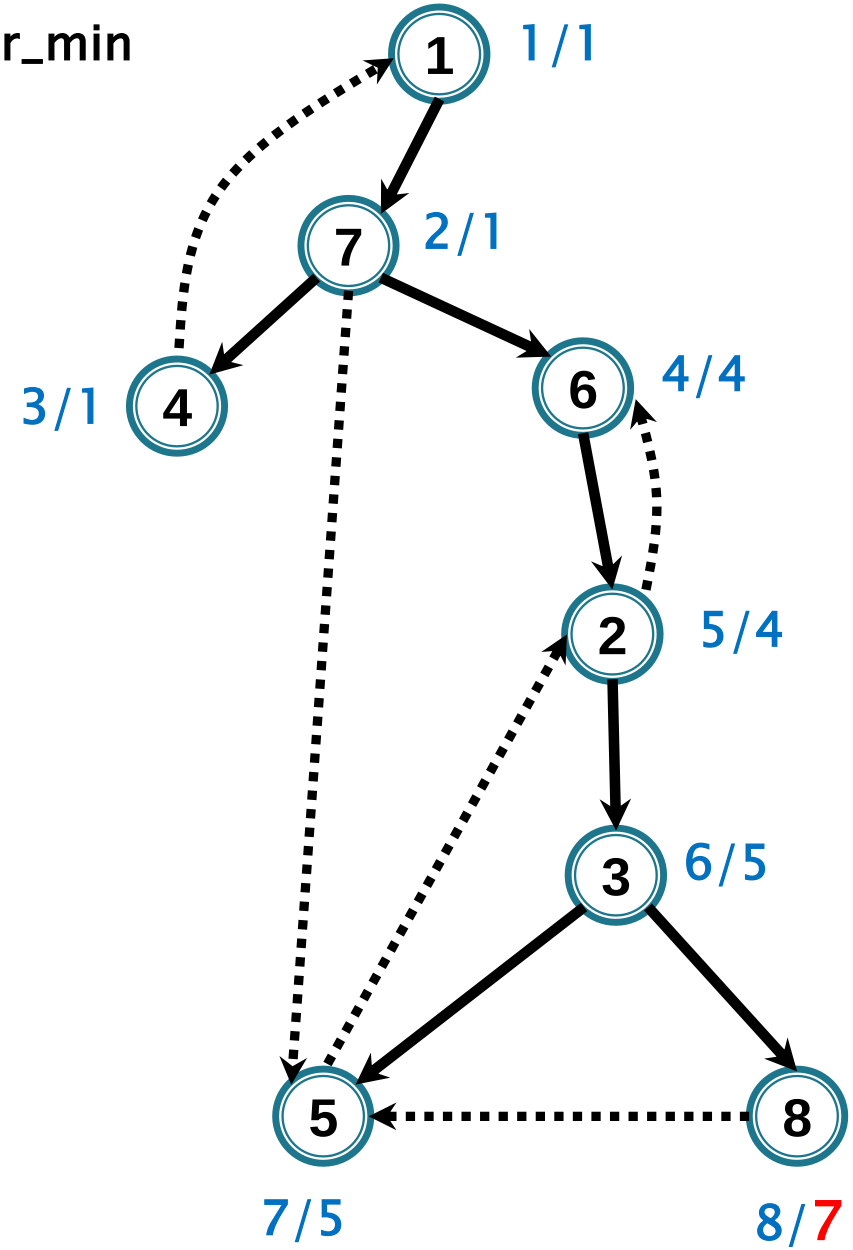
Algoritmul lui Tarjan

nr[v] = numărul vârfului în parcurgerea DF
(al câtelea a fost vizitat)

nr_min[v] = min { nr[u] | putem ajunge din v in u folosind din arboreldoar vârfurile din subarborele de rădăcina ve DF (inclusiv v) }

Algoritmul lui Tarjan



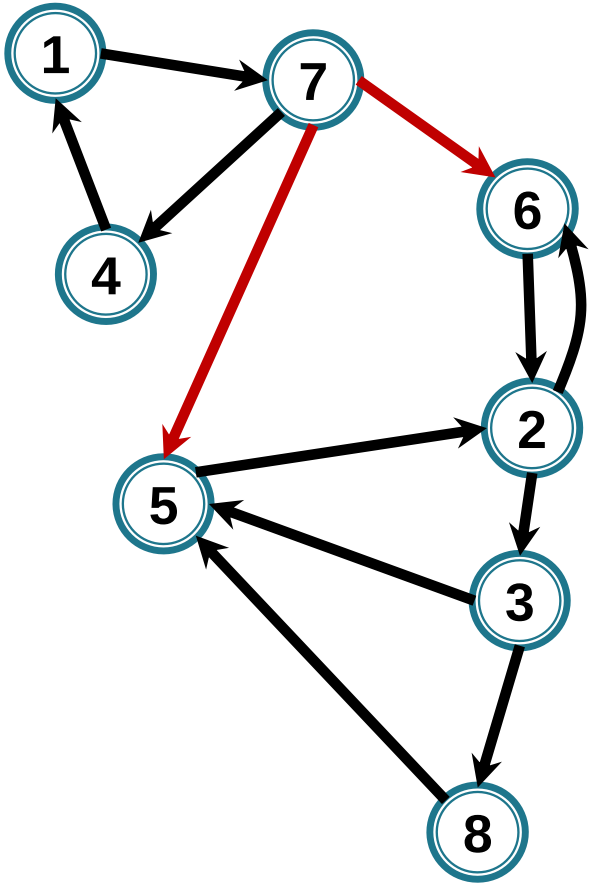


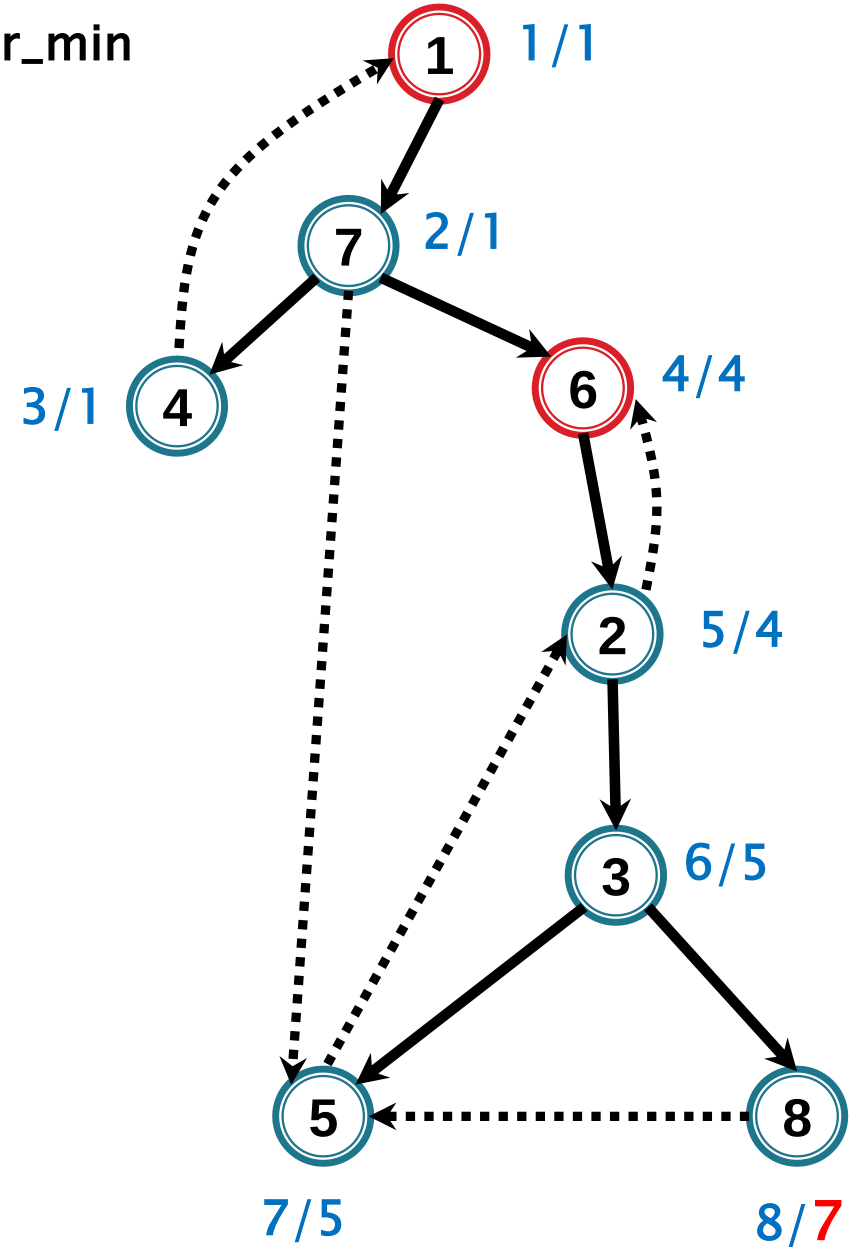
Algoritmul lui Tarjan

v este rădăcina unei componente tare conexe ⇔

nr[v] = nr_min[v]

Algoritmul lui Tarjan





Indicații implementare

```
void df(int i){
      viz[i] = 1;
      nr[i]=nrdf++;
      nr_min[i] = nr[i];
      S.push(i), in_stiva[i] = 1;
      for(j vecin al lui i)
            if(viz[j]==0){
                  df(j);
                  nr_min[i] = min{nr_min[i], nr_min[j] }
            }
            else
               if(in_stiva[j])
                     nr_min[i] = min{nr_min[i], nr[j] }
      if (nr_min[i] == nr[i]) {
            do { u = S.top(); afiseaza(u); s.pop();
                in_stiva[u]=0;
            } while (u != i);
}
```