Componente Tare Conexe

Ștefan Trăușan-Matu

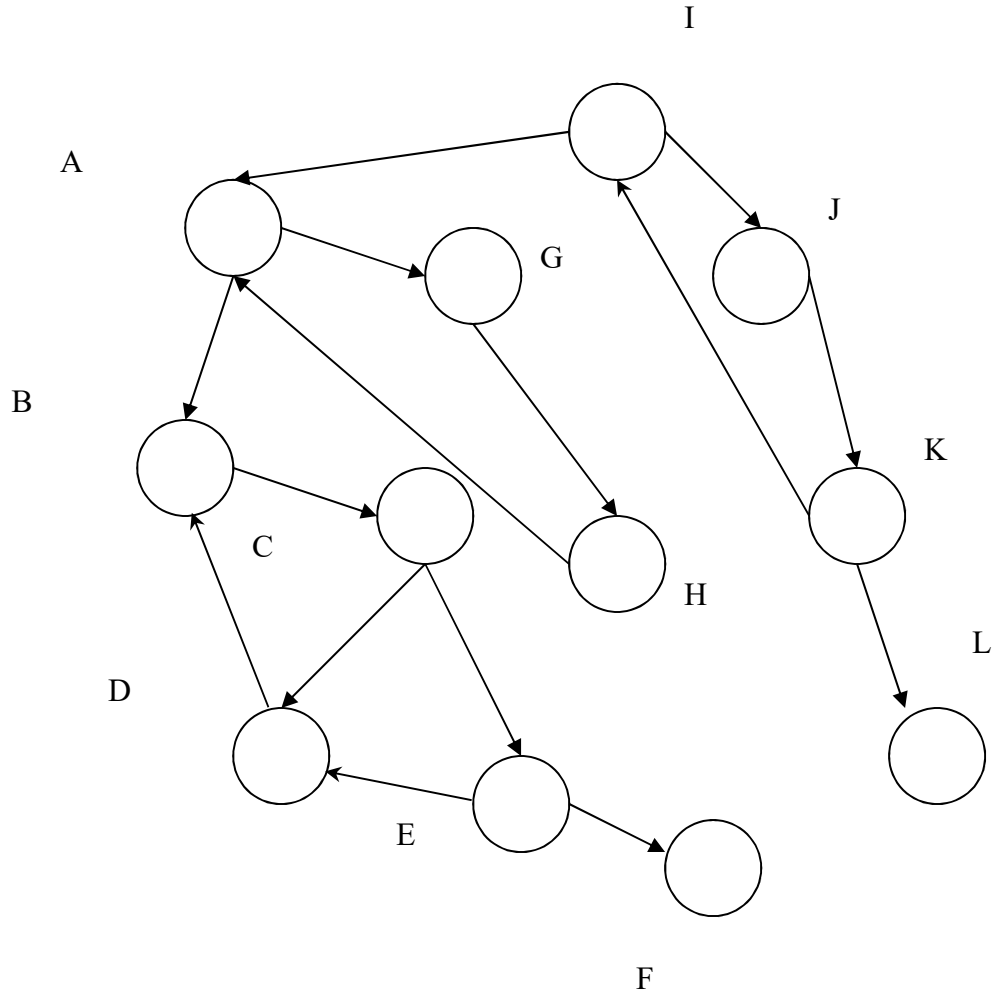
Componente ale unui graf

- G'=(N',A') este **subgraf** al grafului G=(N,A), ddacă $N' \subset N$ și $A' \subset A$
- G'=(N',A') este **indus** de G=(N,A) ddacă $\forall u,v \in N', (u,v) \in A \Rightarrow (u,v) \in A'$
- O **componentă** cu o proprietate P a unui graf G este un subgraf maximal cu proprietatea P, indus de acel graf G

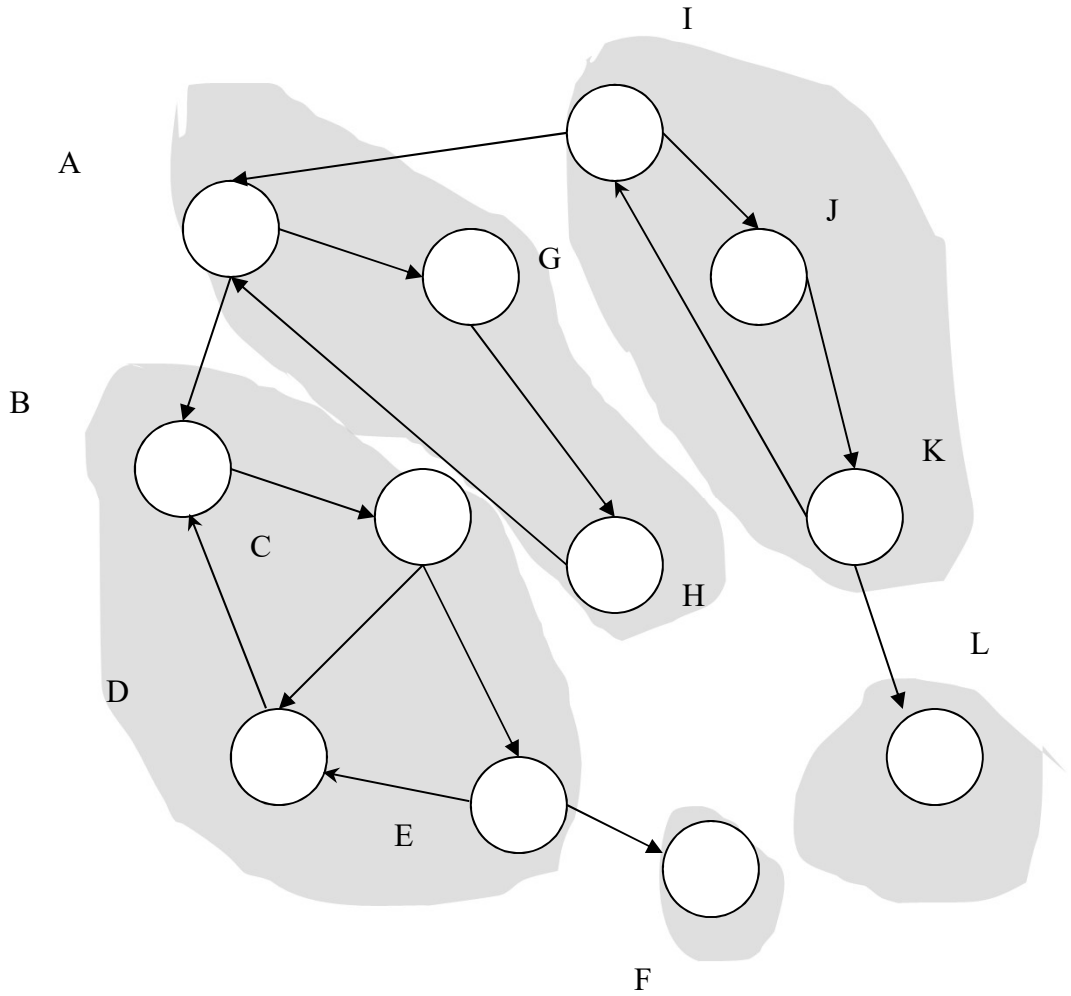
Componente

- **conexe** – grafuri *neorientate* – între oricare două noduri este o cale
- **tare conexe** – grafuri *orientate* – între oricare două noduri este o cale, și într-un sens și în celălat
- **biconectate** – grafuri *neorientate* – între oricare două noduri este două căi

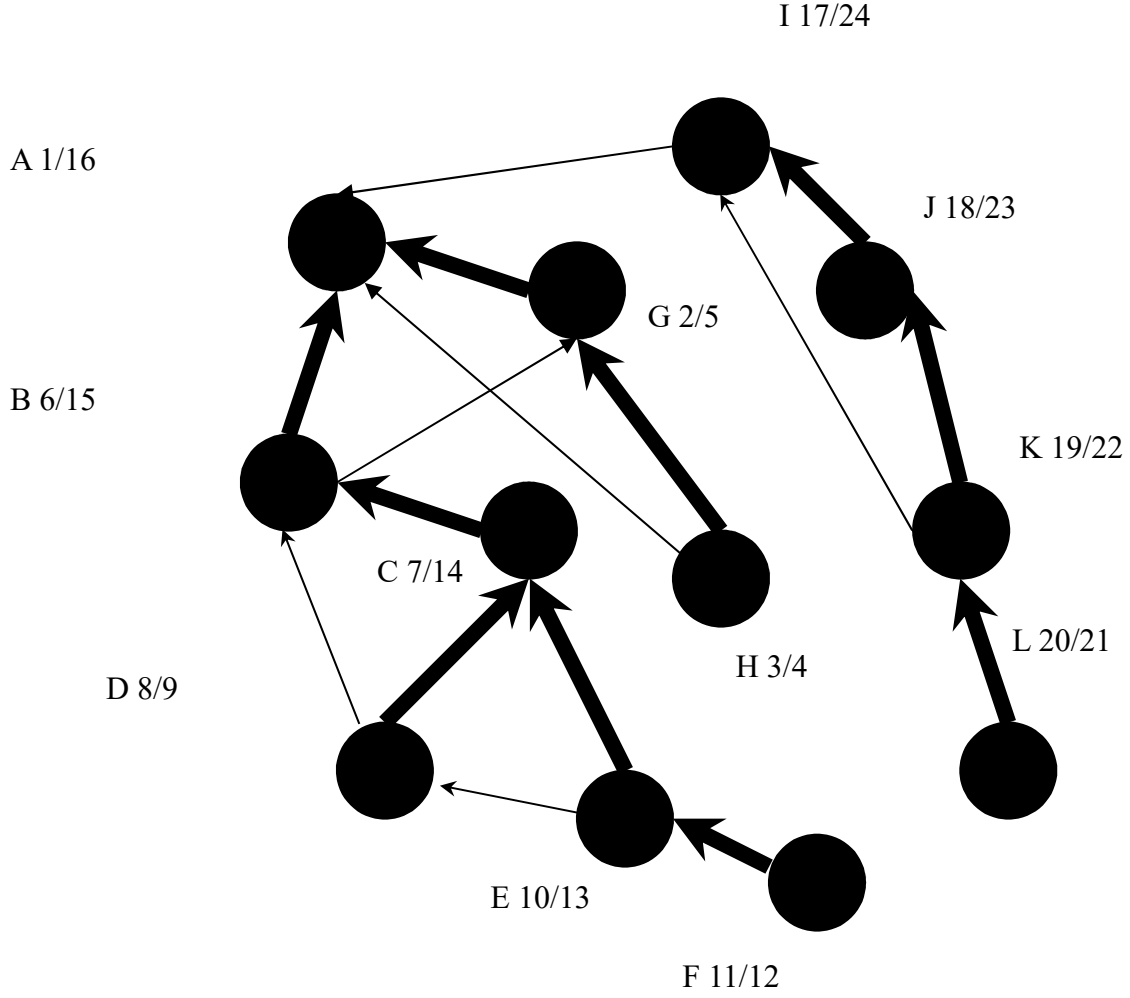
Componente tare conexe – graful inițial



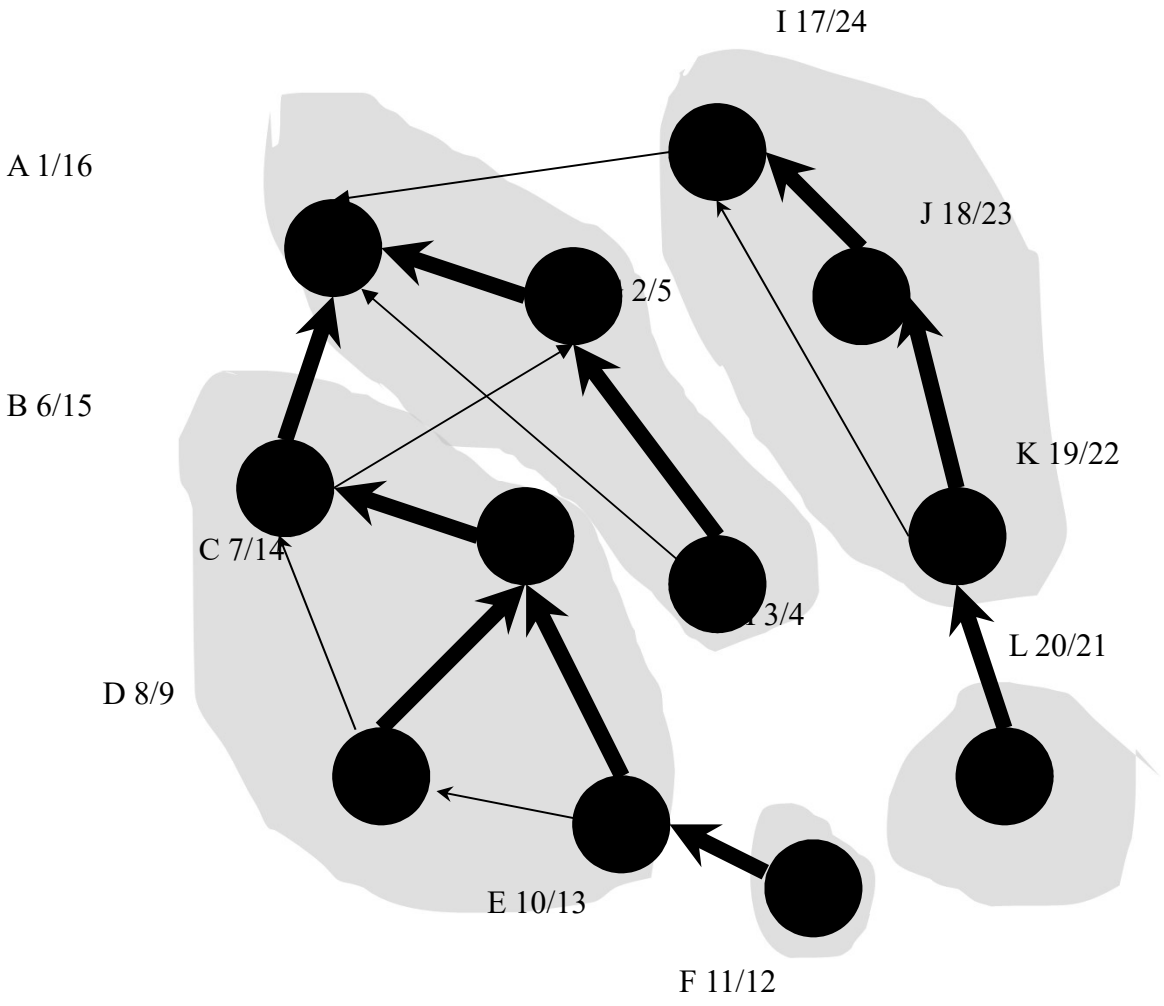
Componentele tare conexe



Parcurgerea în adâncime a grafului inițial



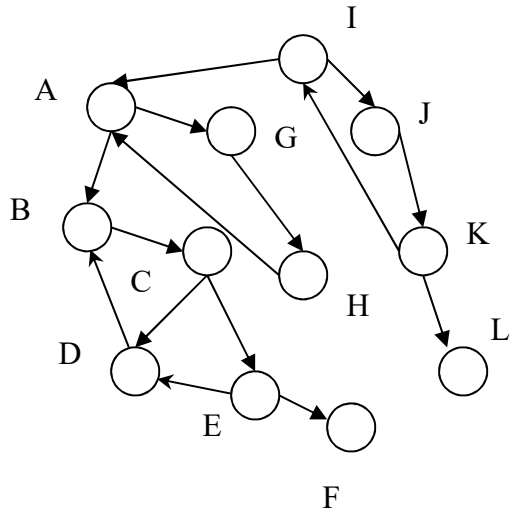
Proprietăți.....?



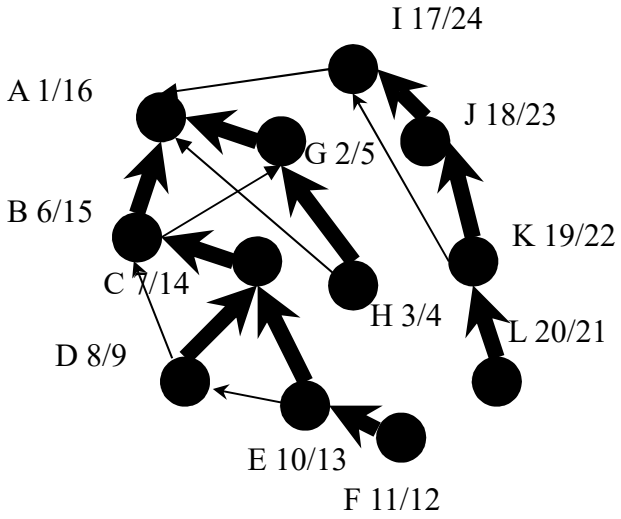
Lema: Oricare cale între 2 noduri ale unei CTC rămâne în acea CTC

Teorema 1: Nodurile din aceeași CTC sunt grupate în același arbore de parcurgere în adâncime

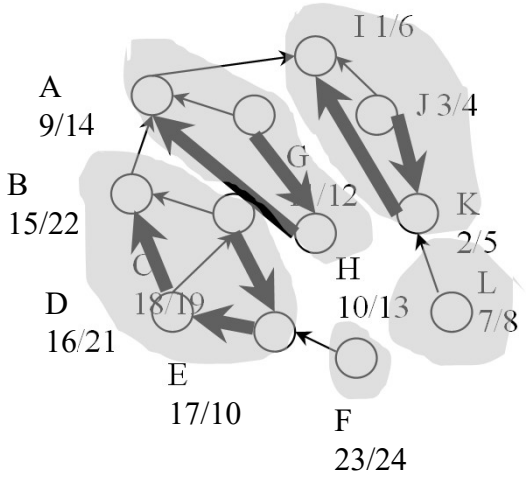
Graful componentelor (GCTC)



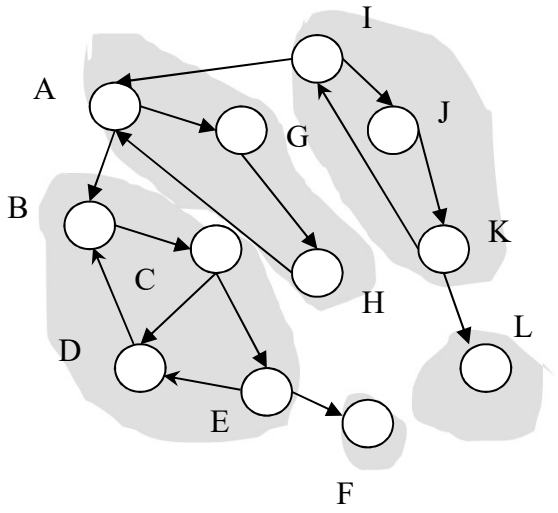
Graful componentelor (GCTC)



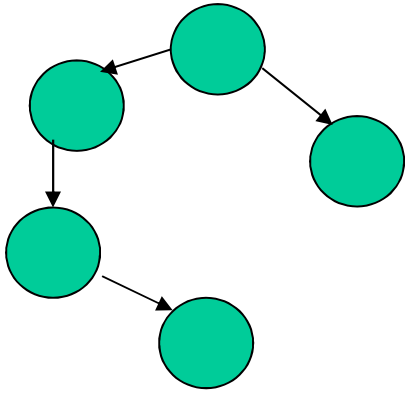
Graful componentelor (GCTC)



Graful componentelor (GCTC)



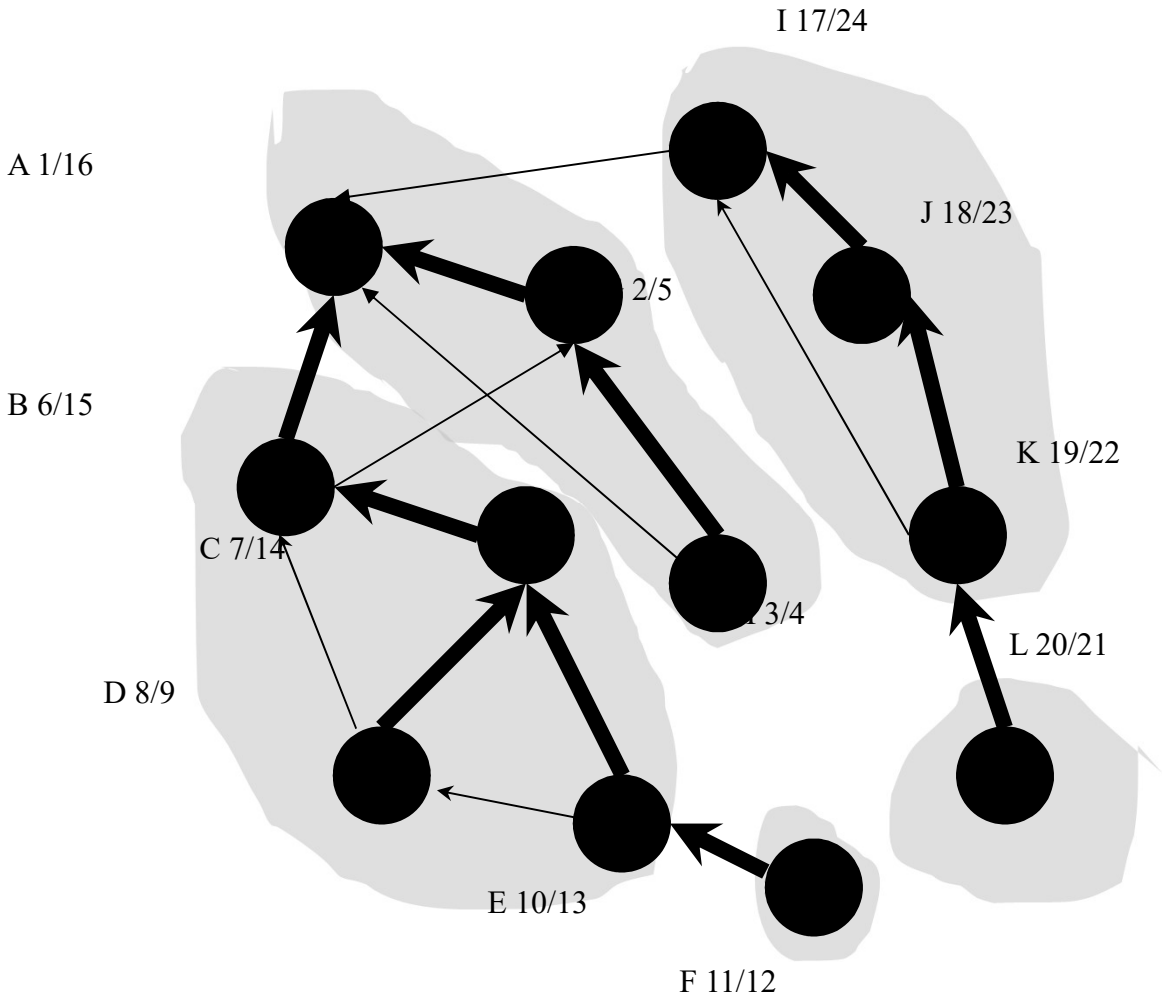
Graful componentelor (GCTC)



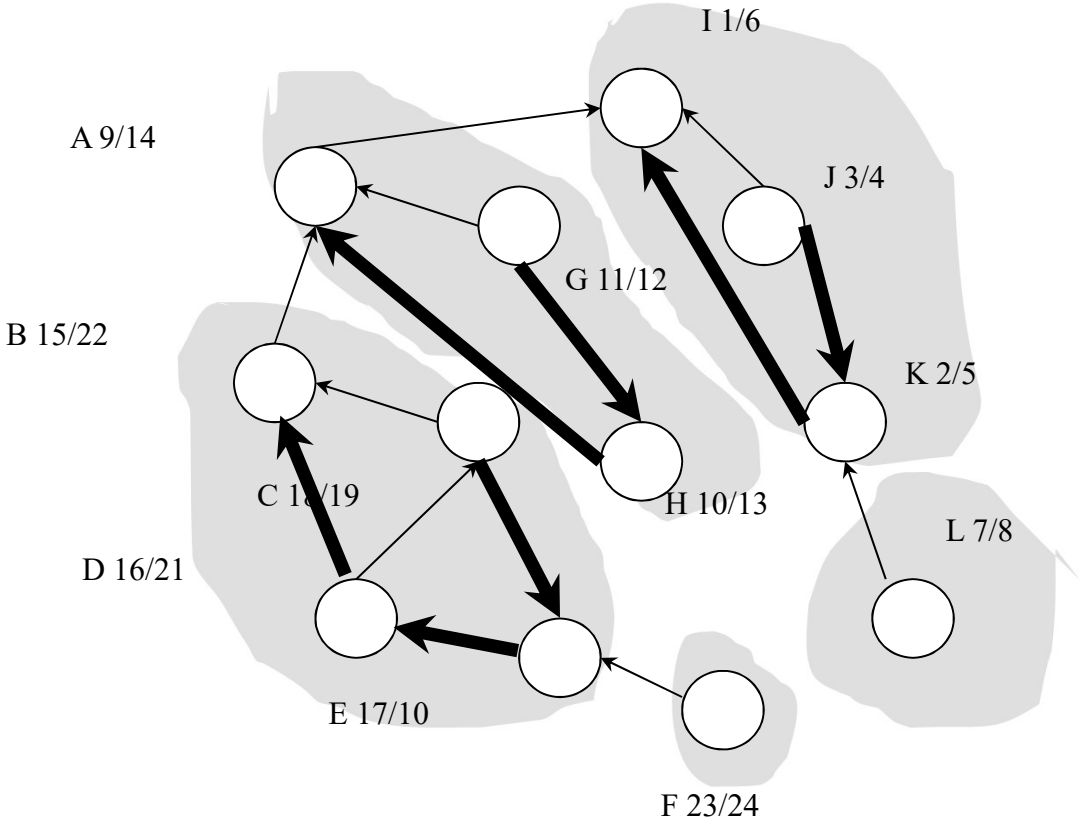
Strămoșul unui nod

- G=(N,A), $u \in N$, $\Phi(u)$= strămoșul lui u este accesibil din u și este terminat ultimul într-o parcurgere în adâncime a lui G
  - $\Phi(u) \in R(u)$
  - $\Phi(u).finis = \max\{v.finis | v \in R(u)\}$
- **Teoremă proprietăți strămoș:** $\Phi(u)$ satisface urmatoarele proprietati
  - $u.finis \leq \Phi(u).finis$
  - $\forall v \in R(u)\ \Phi(v).finis \leq \Phi(u).finis$
  - $\Phi(\Phi(u)) = \Phi(u)$
- $\Phi(u)$ este primul nod din CTC descoperit de DFS(G)

A, C, I, F, L sunt strămoși ai nodurilor din componenta conexa din care fac parte



Parcurgerea în adâncime a grafului transpus în ordinea ....



Componente Tare Conexe (CTC)

Teorema 2: G=(N,A), $\forall u \in N$, u este descendent al lui $\Phi(u)$ in Arb($\Phi(u)$)

Corolar: u și $\Phi(u)$ sunt în aceeași CTC

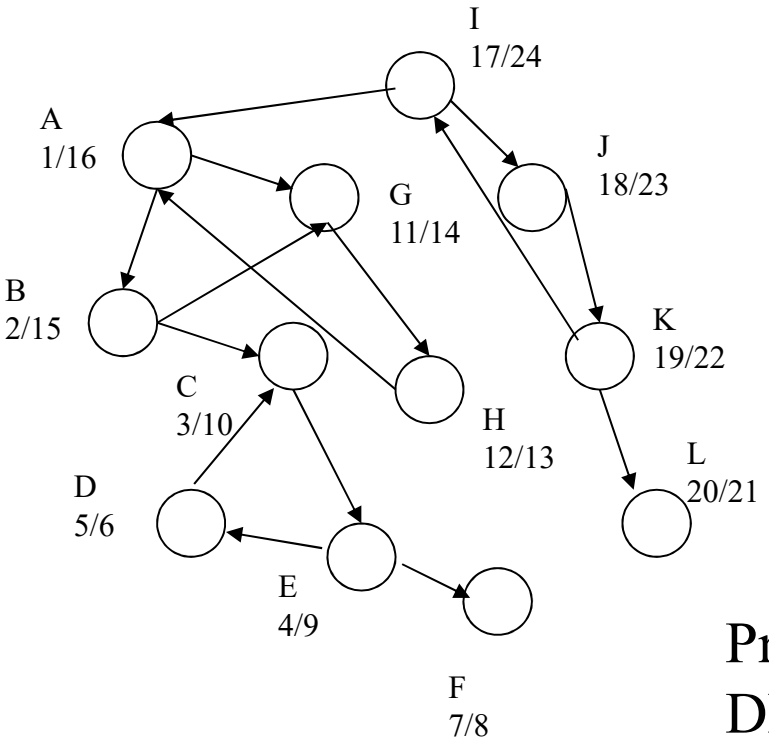
Teorema 3: G=(N,A), $u,v \in N$;

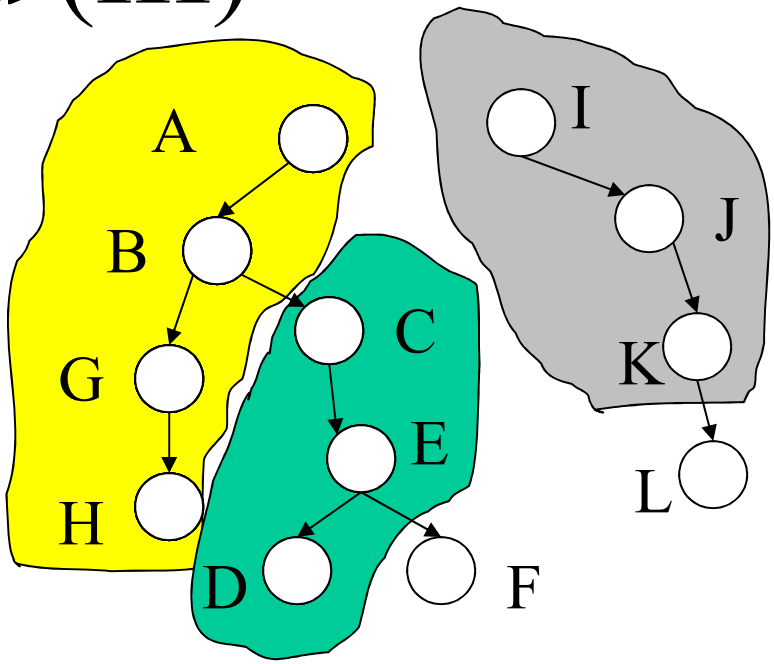
u și v aparțin aceleiași CTC

<=>

$\Phi(u)=\Phi(v)$

Exemplu (III)





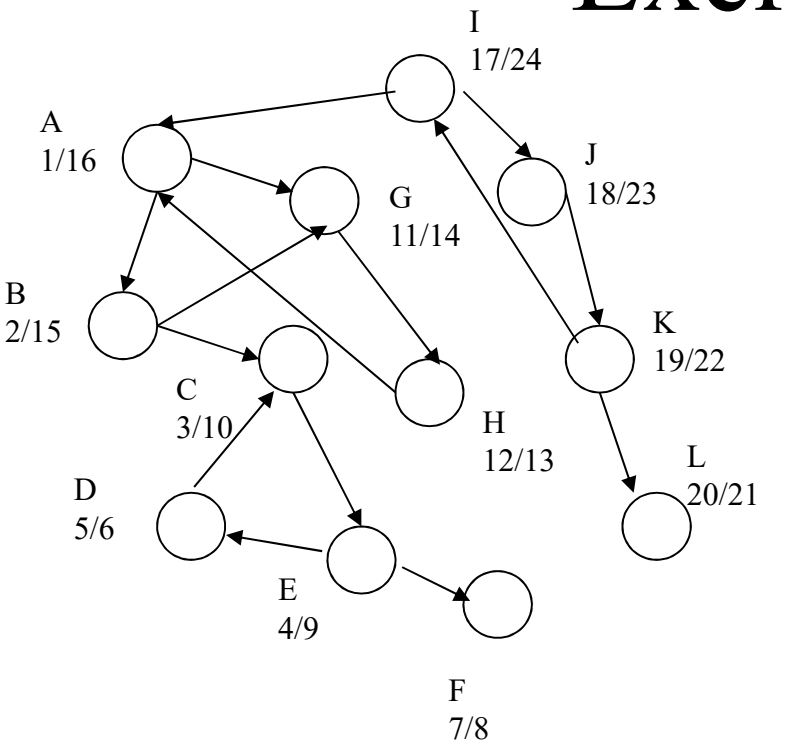
Primul nod dintr-o CTC descoperit la DFS va avea copii in arborele generat de DFS toate elementele componentei conexe!

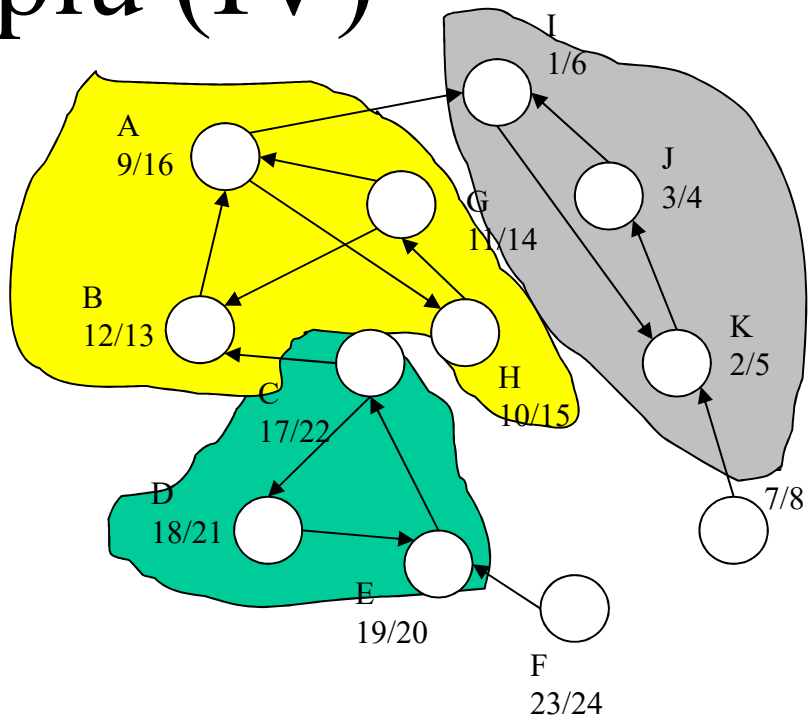
Componente Tare Conexe (CTC)

- Problemă: trebuie să eliminăm nodurile care nu sunt în componenta conexă
- Vrem ca fiecare arbore construit să conțină o CTC

  => idee – eliminăm nodurile ce nu aparțin CTC
  - Daca aparțin Arb(u) și nu CTC => ∃ u..v și nu v..u
  - =>DFS pe graful transpus

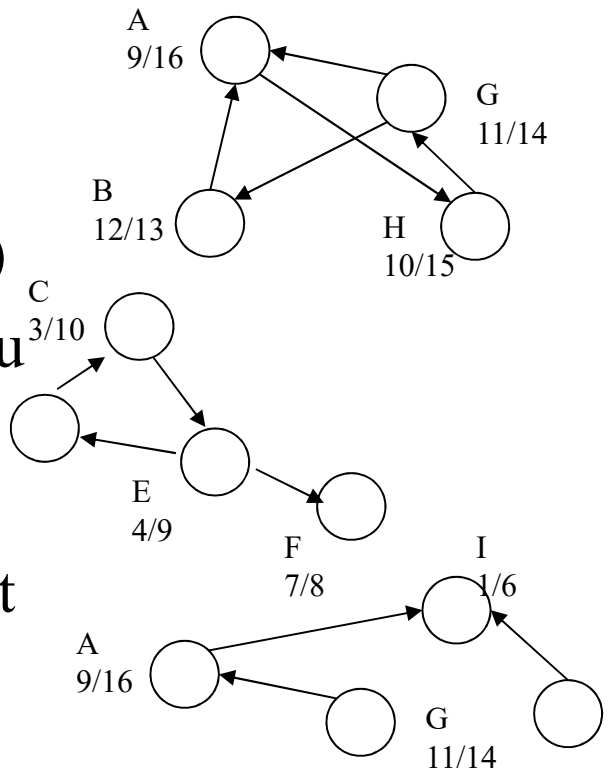
Exemplu (IV)


I
17/24
A
1/16
J
18/23
G
11/14
B
2/15
K
19/22
C
3/10
H
12/13
L
20/21
D
5/6
E
4/9
F
7/8

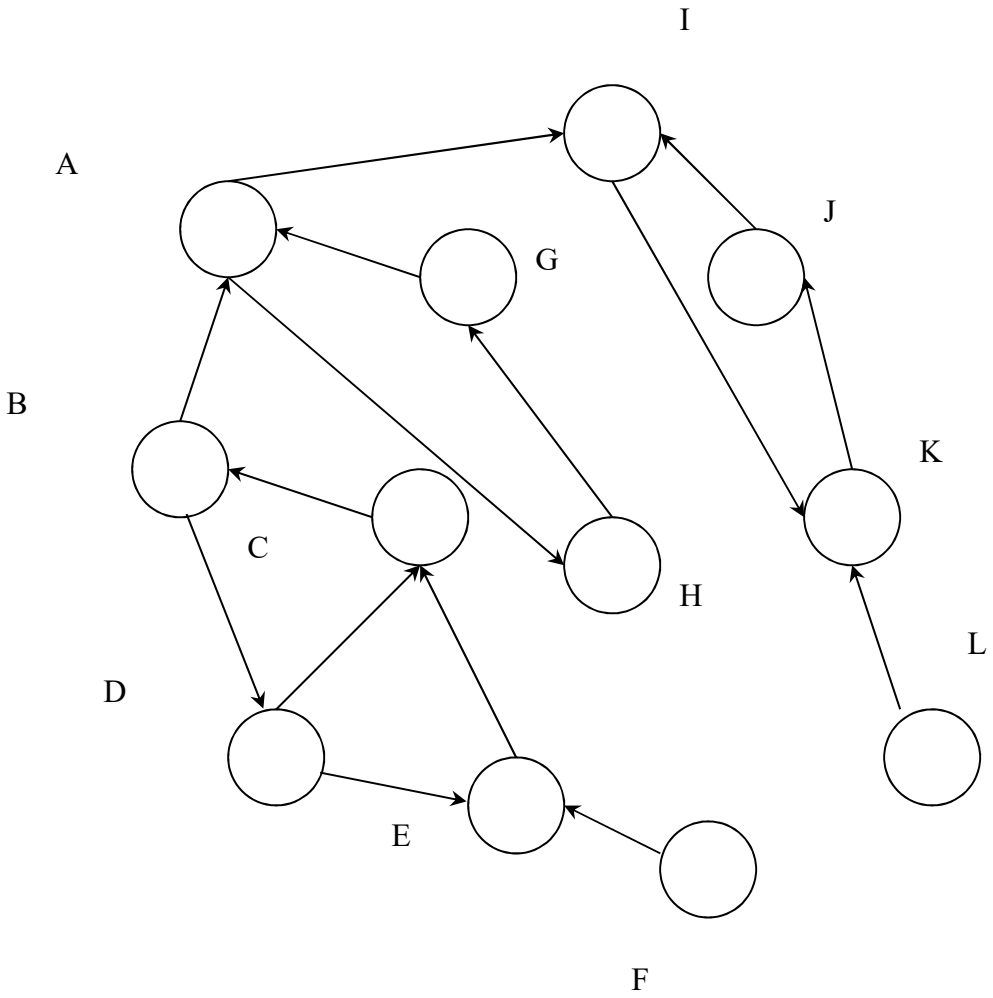

I
1/6
A
9/16
J
3/4
G
11/14
B
12/13
K
2/5
H
10/15
C
17/22
7/8
D
18/21
E
19/20
F
23/24

Componente Tare Conexe (CTC)

- Cazuri in DFS(G^T)
  - v este in CTC descoperita din u=> v poate fi descoperit din u si in DFS(G^T)
  - v$\notin$CTC dar v$\in$Arb(u) in DFS(G)=> nu va fi atins in DFS(G^T) din u
  - v$\notin$CTC dar $\exists$ v..u in G =>v.finis>u.finis=> v va fi deja colorat in negru cand se exploreaza u

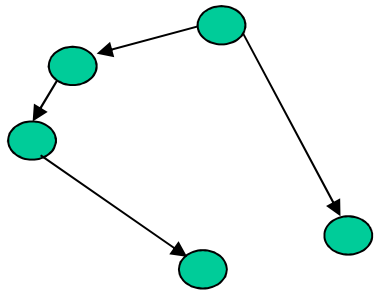


Graful transpus



Observatii

- Inlocuind componentele tare conexe cu noduri obtinem un graf aciclic
- Prima parcurgere DFS este o sortare topologica (de ce?)



Algoritmul lui Kosaraju

- CTC(G)

  Parc-ad(G)

  G^T=Transpune(G)

  Parc-ad(G^T) **cu modificarea că în bucla principală se tratează nodurile în ordinea descrescătoare a timpilor de finiș de la primul Parc-ad**

- Componentele conexe sunt reprezentate de padurea de arbori generati de Parc-ad(G^T)

Algoritmul lui Tarjan (I)

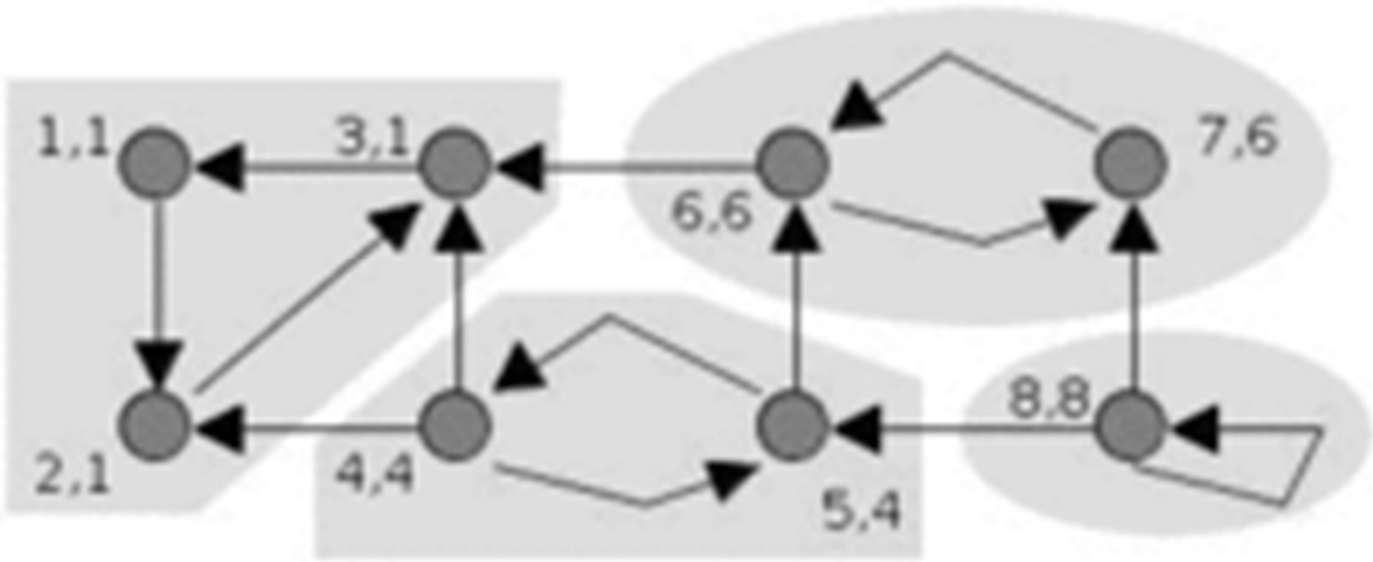
- Bazat tot pe DFS
- Folosește o singură parcurgere în adâncime
- Determină din parcurgere care sunt strămoșii ("rădăcinile") CTC
- O stivă suplimentară în care:
  - nodurile nu sunt automat eliminate la întoarcerea din recursivitate a DFS-rec
  - un nod rămâne ddacă există o cale la un nod anterior din stivă
  - u.min_inapoi este timpul minim de debut al unui nod accesibil din u

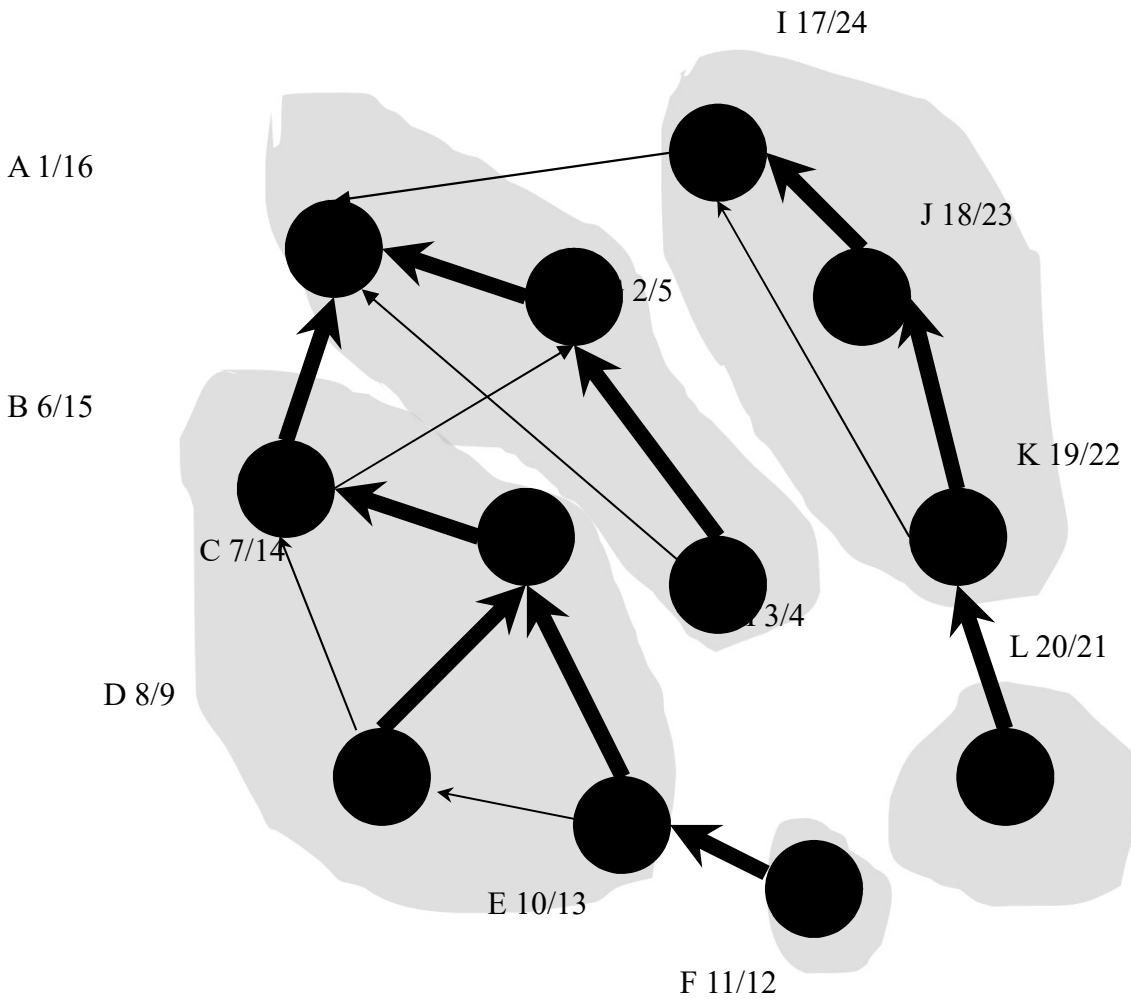
Algoritmul lui Tarjan (II)

pentru fiecare n din N **repetă**
    n.debut← -1 // *echivalent cu n.culoare←alb*
    S←∅; // *S este stiva*
    timp← 0
pentru fiecare n din N **repetă**
    dacă n.debut=-1 **atunci** Parc-ad-rec_Tarjan(n)

Algoritmul lui Tarjan (III)

```
Parc-ad-rec_Tarjan(n)
      n.debut←timp                          // si devine nod gri
      n.min_inapoi ←timp
      timp← timp+1
      Push(S,n)
      Pentru fiecare v succesor al lui n repetă
         dacă v.debut=-1                    // nod alb
           atunci
              Parc-ad-rec_Tarjan(v)
              n.min_inapoi ←min(n.min_inapoi,v.min_inapoi)
           altfel dacă v∈S
                    atunci n.min_inapoi←min(n.min_inapoi,v.debut)
    dacă n.min_inapoi=n.debut
     tipărește "CTC"
     repetă
           u=pop(S)
            tipărește u
     până când u=n
```


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


I 17/24
A 1/16
J 18/23
2/5
B 6/15
K 19/22
C 7/14
3/4
L 20/21
D 8/9
E 10/13
F 11/12