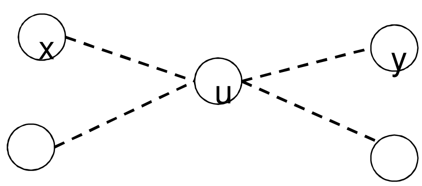
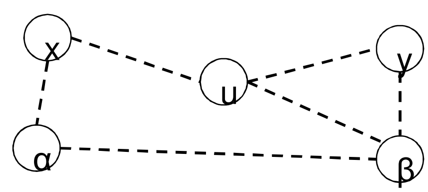
Algoritmi pe grafuri - 4

Puncte de articulație

- G=(N,A) graf neorientat, $u \in N$
- u este punct de articulație dacă $\exists$ $x,y \in N$, $x \neq y$, $x \neq u$, $y \neq u$, a.i. $\forall$ x- - ->y in G trece prin u



Orice drum x..y trece prin u=>u este punct de articulatie



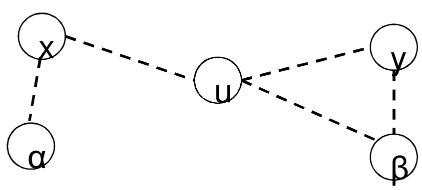
Exista x..α..y care nu trece prin u; u nu mai este punct de articulatie

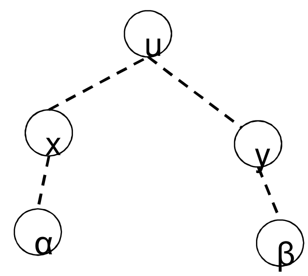
Teoremă

- G=(N,A), graf neorientat, $u \in N$; u este punct de articulație în G **<=>** în urma DFS în G una din proprietățile de mai jos este satisfacută
  – u rădăcină și u domină cel puțin 2 subarbori
  – u nu este rădăcină și $\exists$v descendent al lui u în Arb(u) a.i. $\forall x \in$ Arb(v) si $\forall$(x,z) parcurs de DFS(G) d(z)>=d(u)

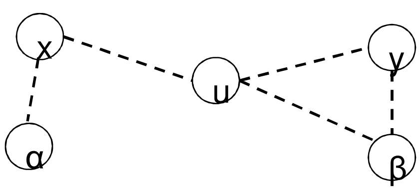
Exemplu

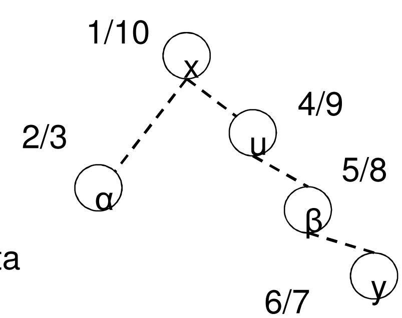
- p(u)=null si u domina cel putin 2 subarbori





- p(u)≠null si ∃v descendent al lui u in Arb(u) a.i. ∀x∈ Arb(v) si ∀(x,z) parcurs de DFS(G) d(z)>=d(u)

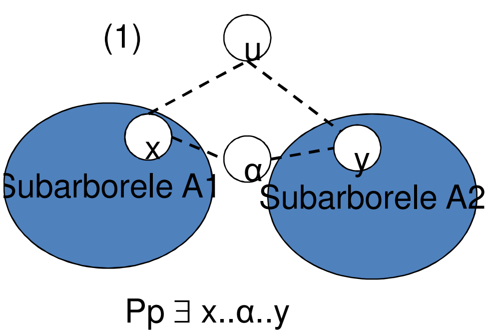




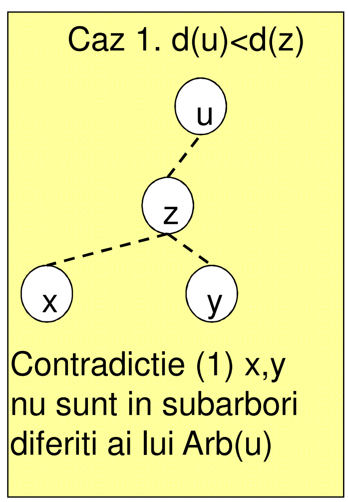
Pentru orice arc din subarborele lui β nu exista nici un arc inapoi spre un nod descoperit inaintea lui u

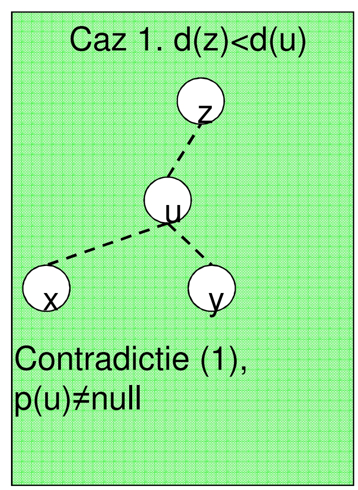
Puncte de articulatie. Demonstratie teorema

- p(u)=null si u domina cel putin 2 subarbori => u este punct de articulatie



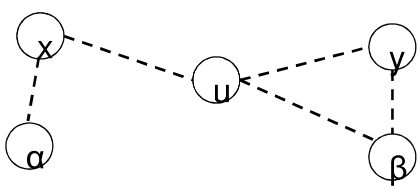
z= primul nod din x..α..y descoperit la DFS
Cf. T drumurilor albe x,y∈ Arb(z)



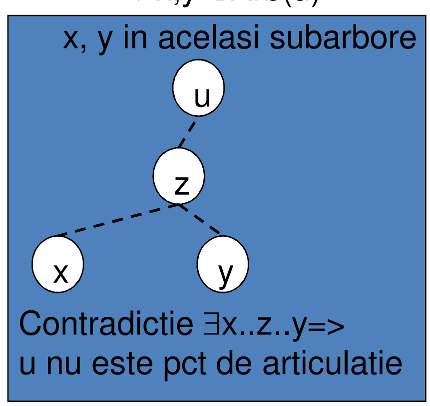


Puncte de articulatie. Demonstratie teorema

- u este punct de articulatie si este descoperit in ciclul principal al DFS =>p(u)=null si u domina cel putin 2 subarbori

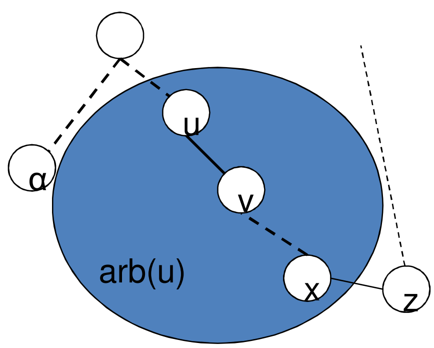


u – primul descoperit din x..y
=>x,y ∈ Arb(u)



Puncte de articulatie. Demonstratie teorema

- p(u)≠null si ∃v descendent al lui u in Arb(u) a.i. ∀x∈ Arb(v) si ∀(x,z) parcurs de DFS(G) d(z)>=d(u) => u este punct de articulatie



d(z)>d(u)=>z∈Arb(u)=> contradictie (z∉Arb(u))

d(z)<d(u)=>contradictie (ipoteza)

Algoritm (I)

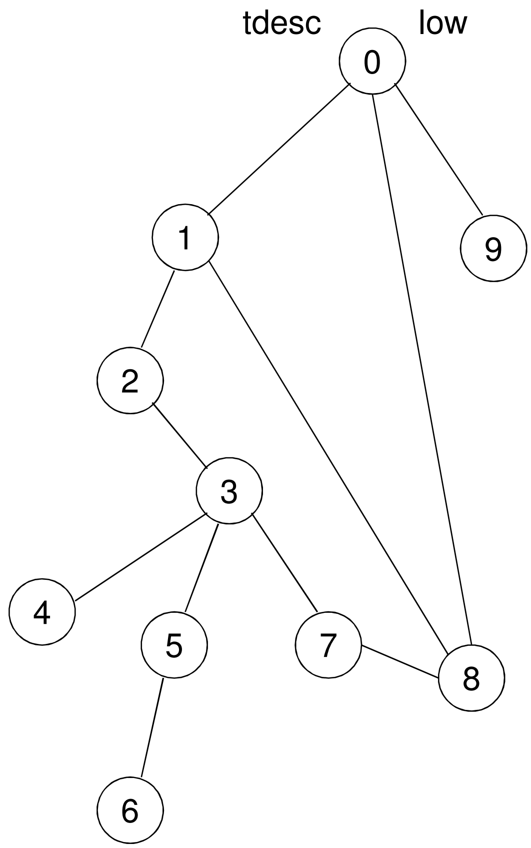
- Articulatii (G)
  - V=noduri(G)
  - Timp=0;
  - Foreach (u∈V)
    - {culoare[u]=alb; d[u]=0;low[u]=0; p[u]=null; subarb[u]=0; art[u]=0;
  - Foreach (u∈V)
    - If(culoare(u)==alb)
      - Exploreaza(u);
      - If(subarb[u]>1) //cazul in care u este radacina in arborele //DFS si are mai multi subarbori
        - » art[u]=1

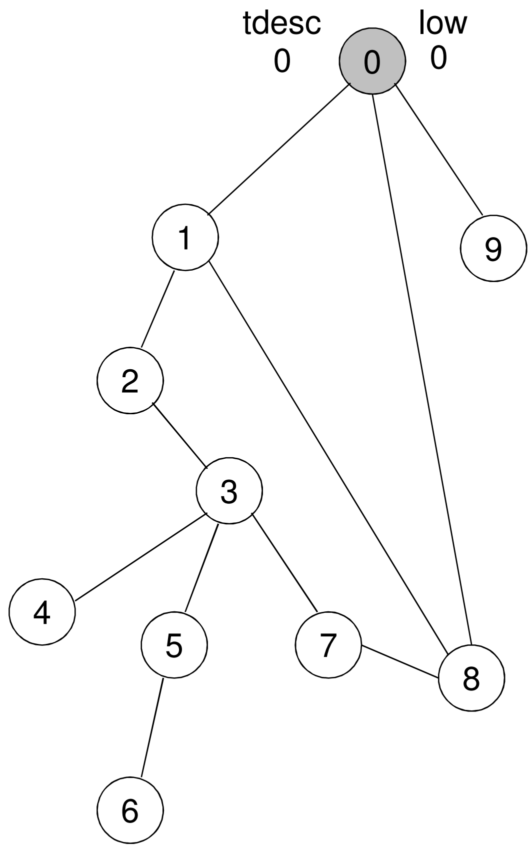
Algoritm (II)

- Exploreaza(u)
  - d[u]=low[u]=timp++;
  - culoare[u]=gri;
  - foreach v succesor al lui u
    - If (culoare[v]==alb)
      - P[v]=u; subarb[u]++;
      - Exploreaza(v);
      - low[u]=min{low[u],low[v]}
      - If(p[u]!=null && low[v]>=d[u]) art[u]=1; //cazul 2 al teoremei
    - Else low[u]=min{low[u], d[v]}

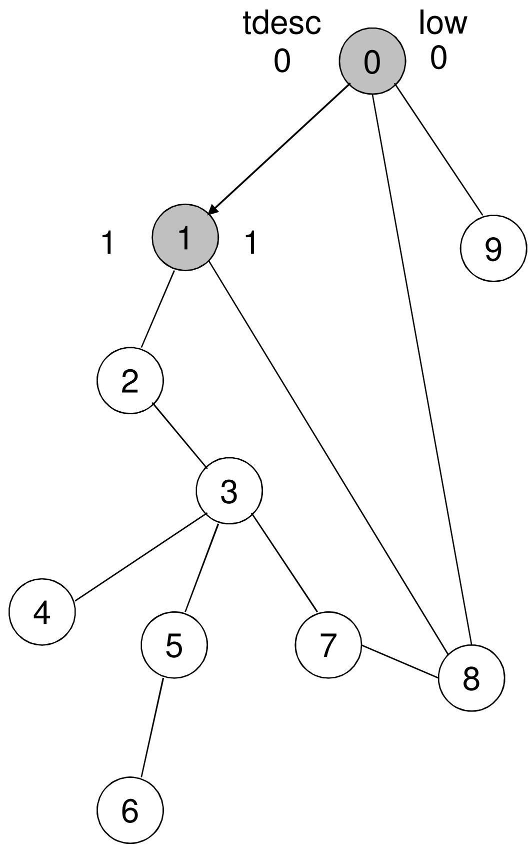
Algoritmul lui Tarjan

- Pentru fiecare v din V
  - Index[v]=-1; //index nedefinit
- timp=0;S=∅; //S- stiva
- DFS-Tarjan(v) //corespondentul lui viziteaza din DFS
  - Low[v]=timp;
  - Index[v]=timp;
  - timp++;
  - Push(S,v)
  - Pentru fiecare v’ succesor al lui v
    - if index[v’]=-1 //nod alb
      - DFS-Tarjan(v’)
      - low[v]=min(low[v], low[v’]);
    - Else if v’∈S
      - low[v]=min(low[v], low[v’]);
  - If low[v]=index[v]
    - Print “SCC:”
    - Repeta
      - V’=pop(S)
      - Print v’
    - Pana cand v’=v

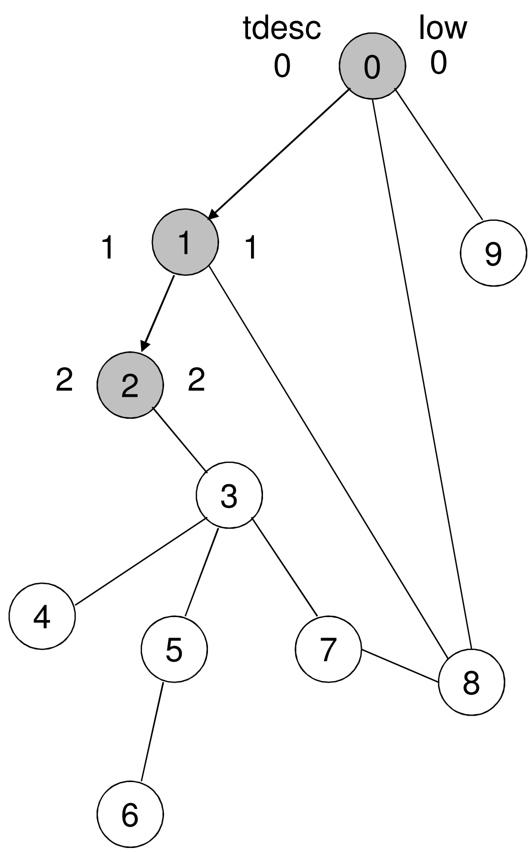

tdesc
low
0
1
9
2
3
4
5
7
8
6



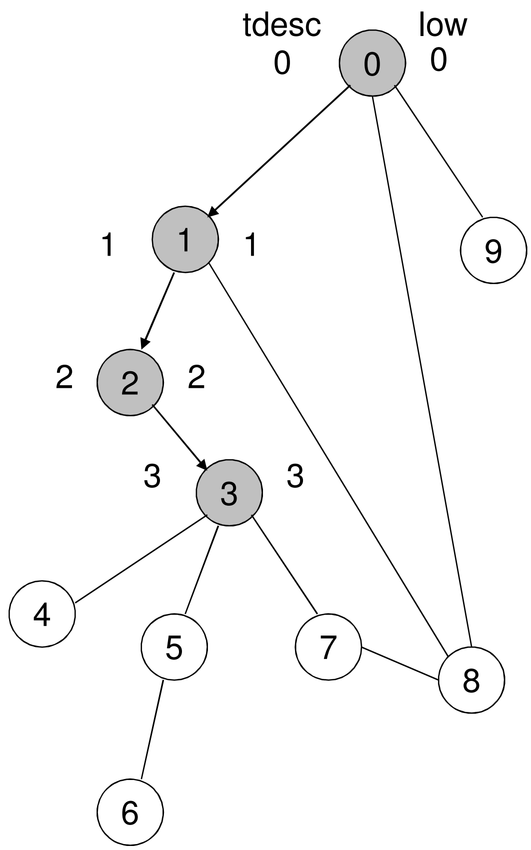
low = tdesc



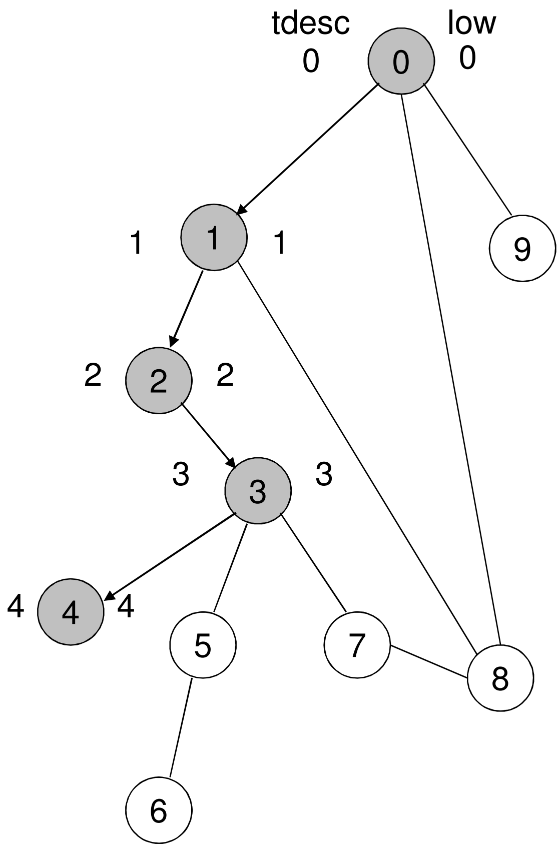
low = tdesc



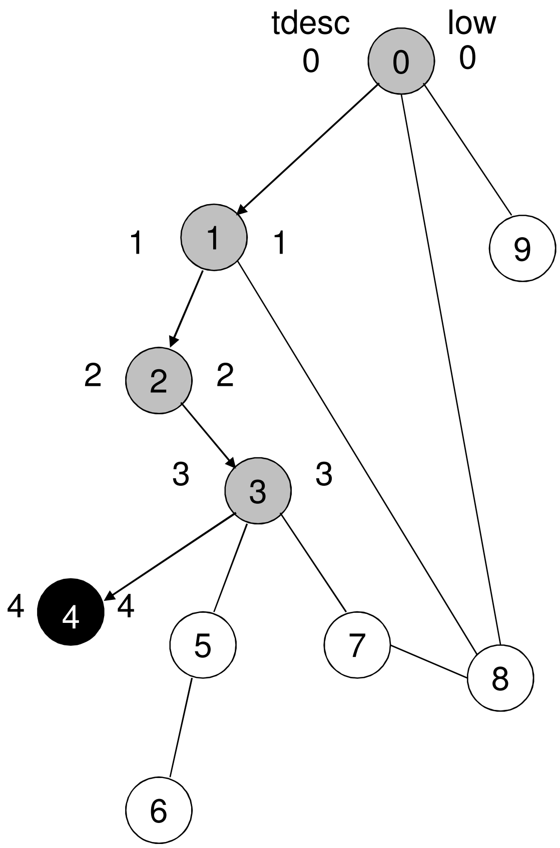
low = tdesc

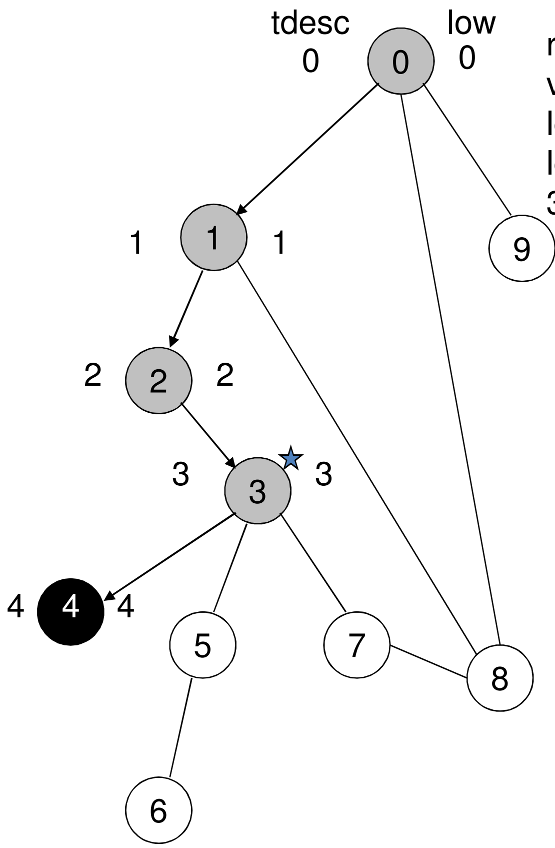


low = tdesc

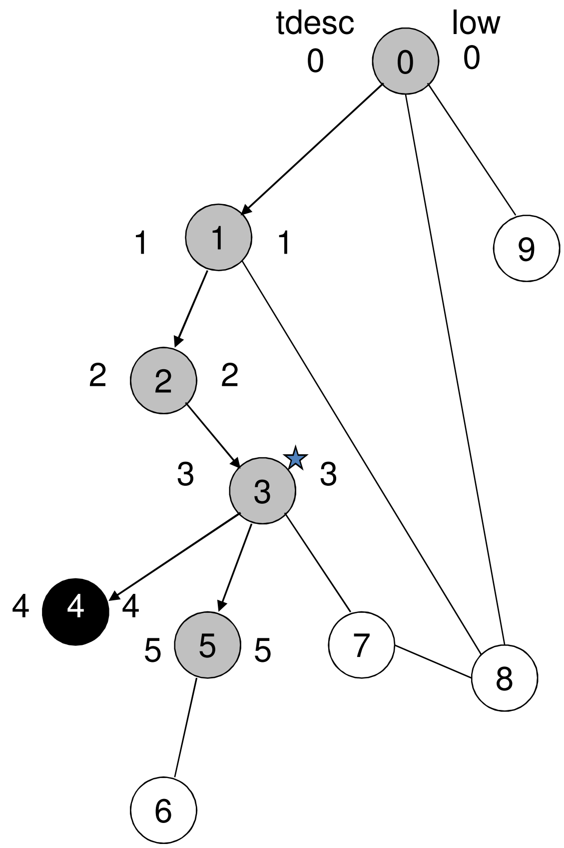


low = tdesc

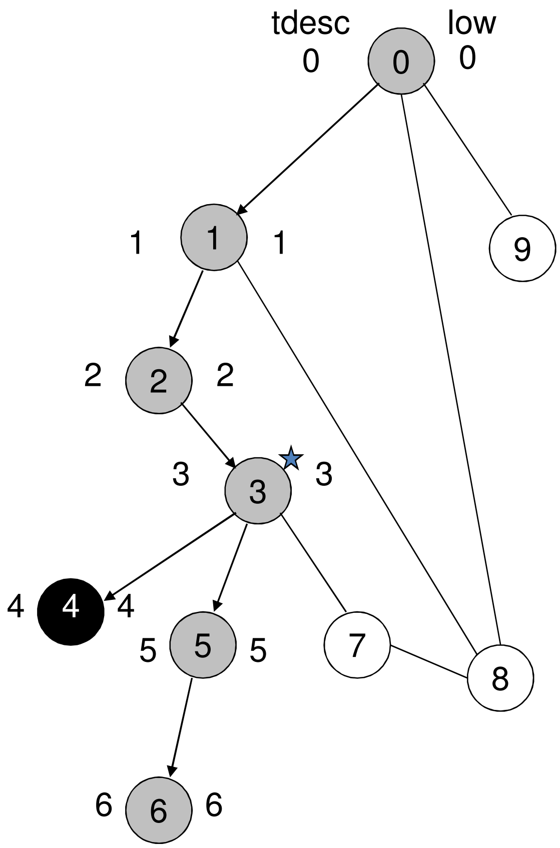

tdesc
0
0
low
0
1
1
1
9
2
2
2
3
3
3
4
4
4
5
7
8
6



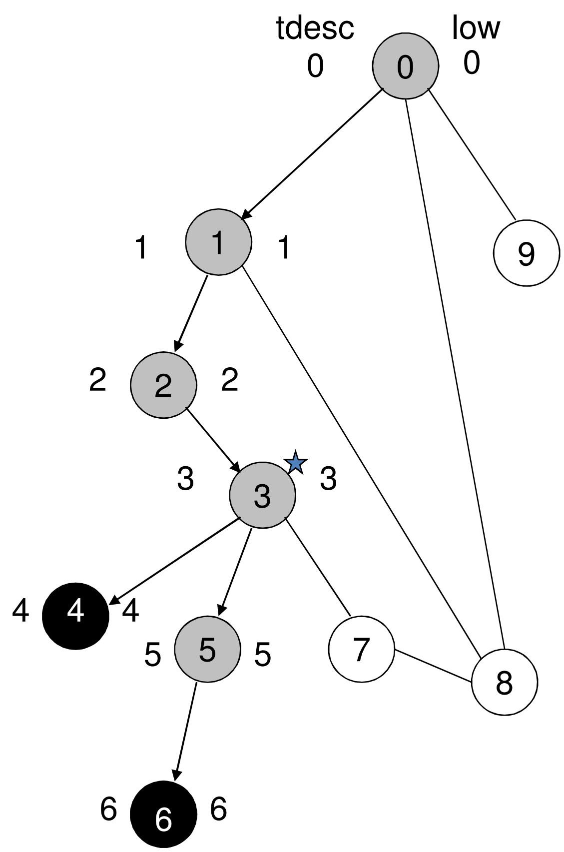
revenire la nodul 3;
verificare low copil:
low[3] = min(low[3], low[4]) = low[3]
low[4] >= tdesc[3] =>
3 este p. de art.

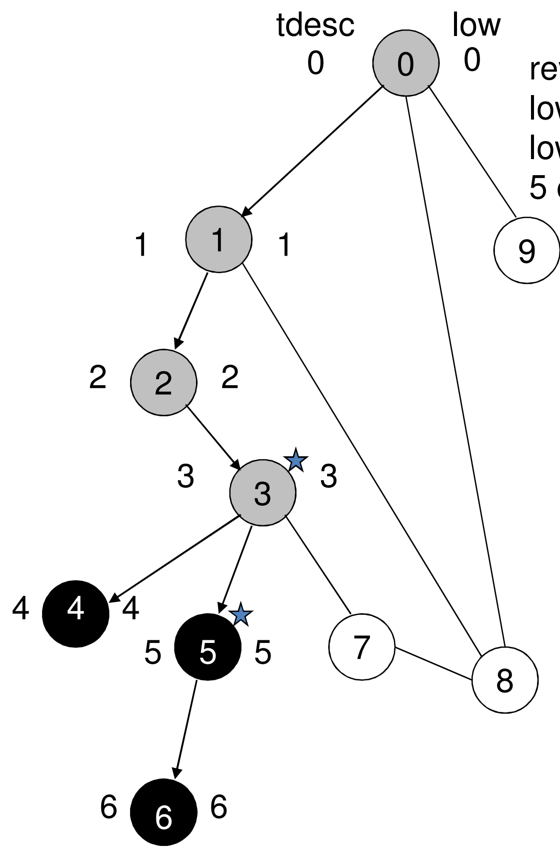


low = tdesc

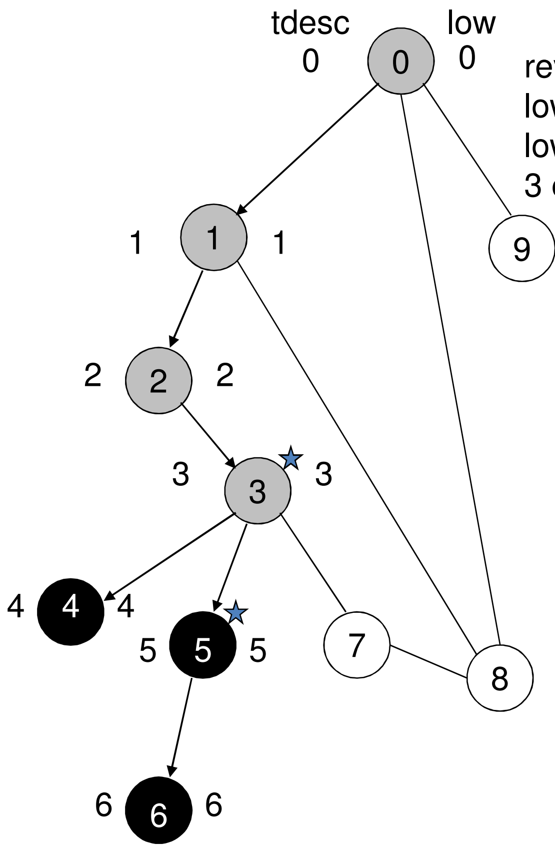


low = tdesc

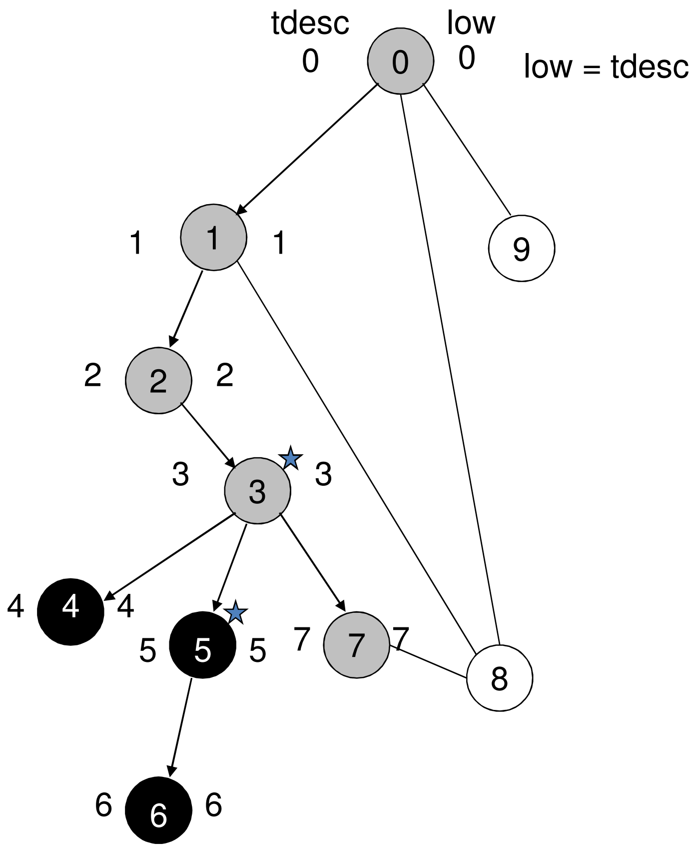

tdesc
low
0
0
0
1
1
1
9
2
2
2
3
3
3
4
4
4
5
5
5
7
8
6
6
6

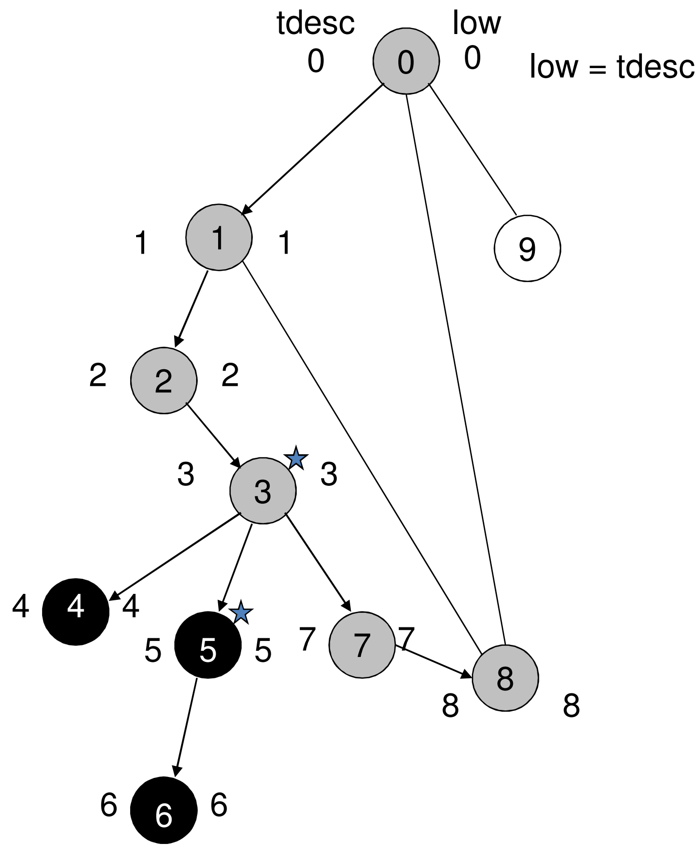


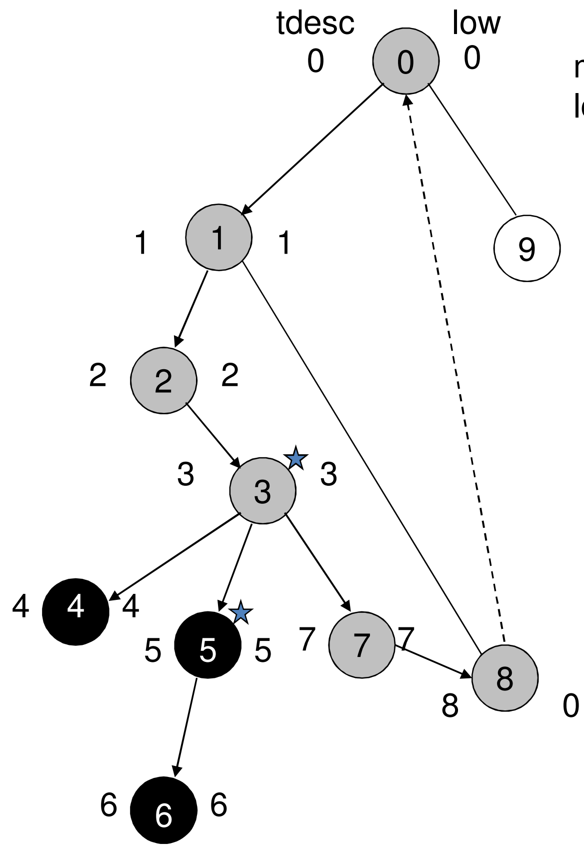
revenire la 5; verificare low copil:
low[5] = min(low[5], low[6]) = low[5]
low[6] >= tdesc[5] =>
5 este p. de art.



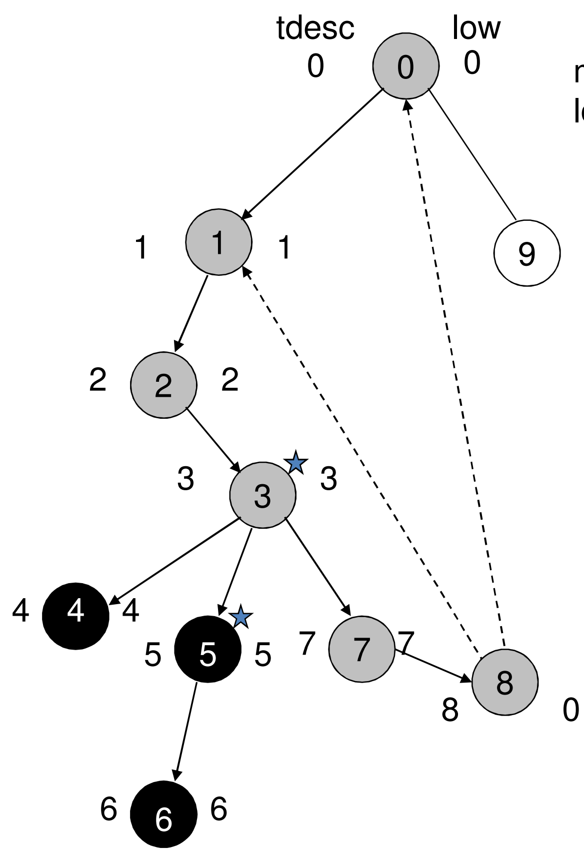
revenire la 3; verificare low copil:
low[3] = min(low[3], low[5]) = low[3]
low[5] >= tdesc[3] =>
3 este p. de art.


tdesc
0
0
low
0
low = tdesc
1
1
1
9
2
2
2
3
3
3
4
4
4
5
5
5
7
7
7
8
6
6
6

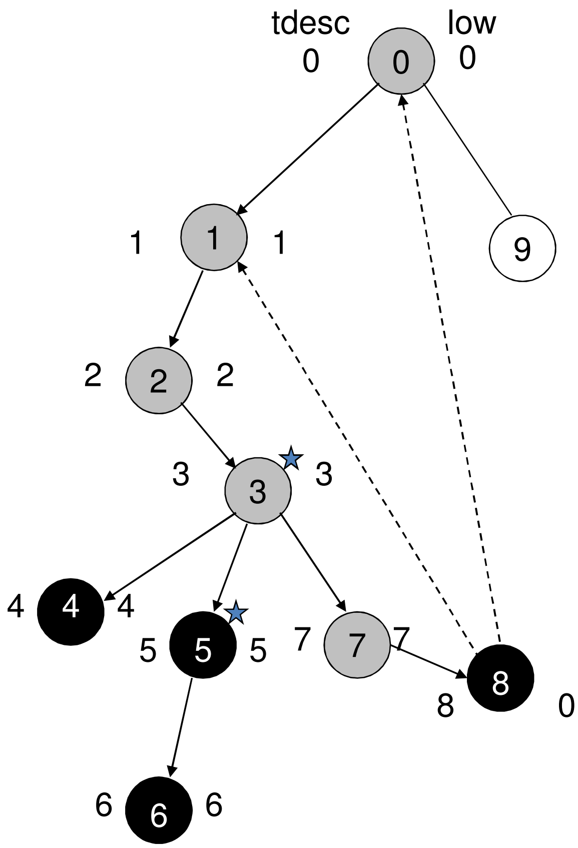

tdesc
0
0
low
0
low = tdesc
1
1
1
9
2
2
2
3
3
3
4
4
4
5
5
5
7
7
7
8
8
8
6
6
6

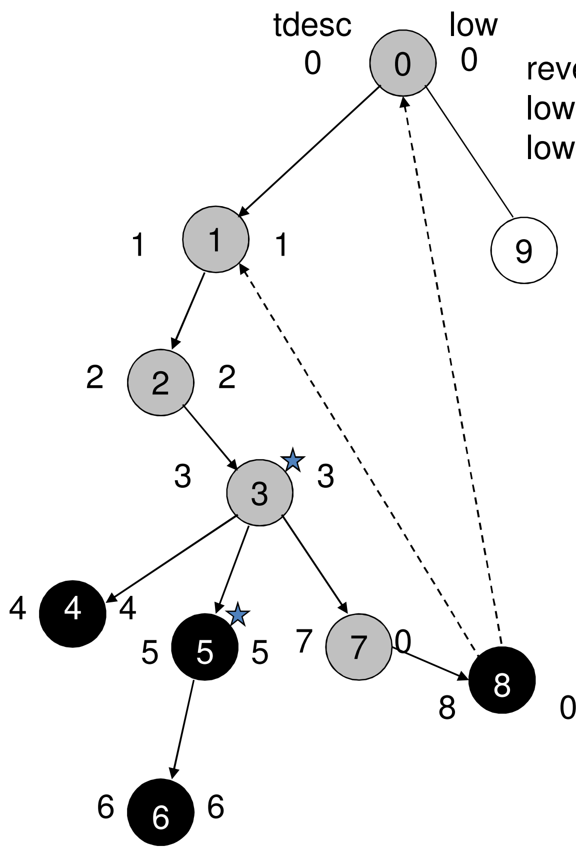


muchie inapoi 8-0
low[8] = min(low[8], tdesc[0]) = 0

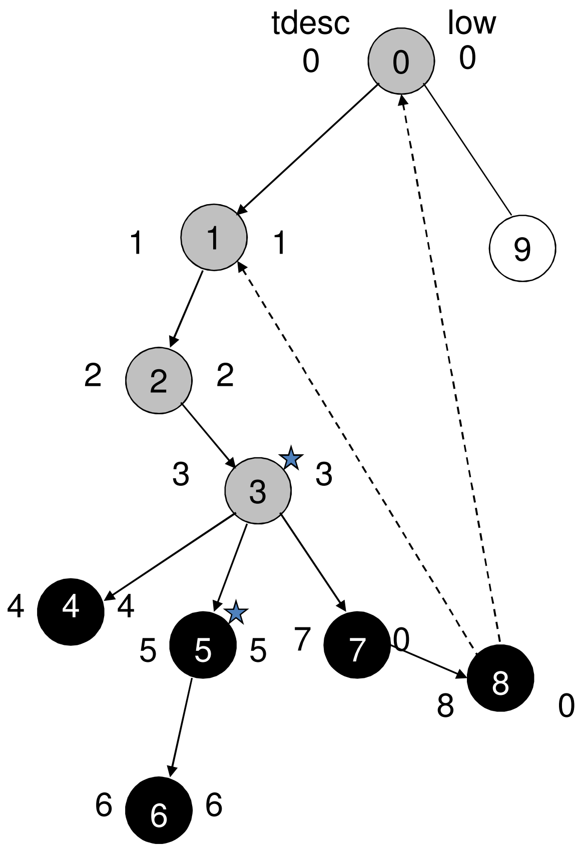


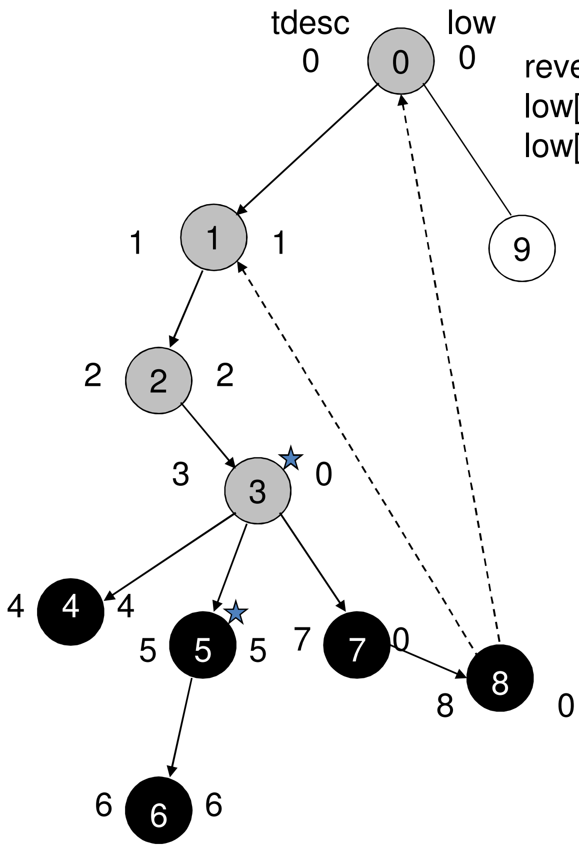
muchie inapoi 8-1
low[8] = min(low[8], tdesc[1]) = 0


tdesc
low
0
0
0
1
1
1
9
2
2
2
3
3
3
4
4
4
5
5
5
7
7
7
8
8
0
6
6
6

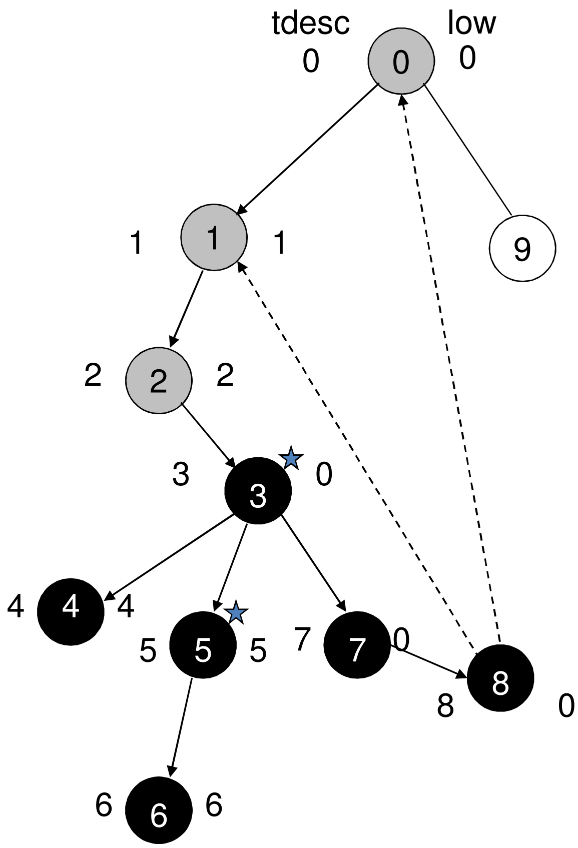


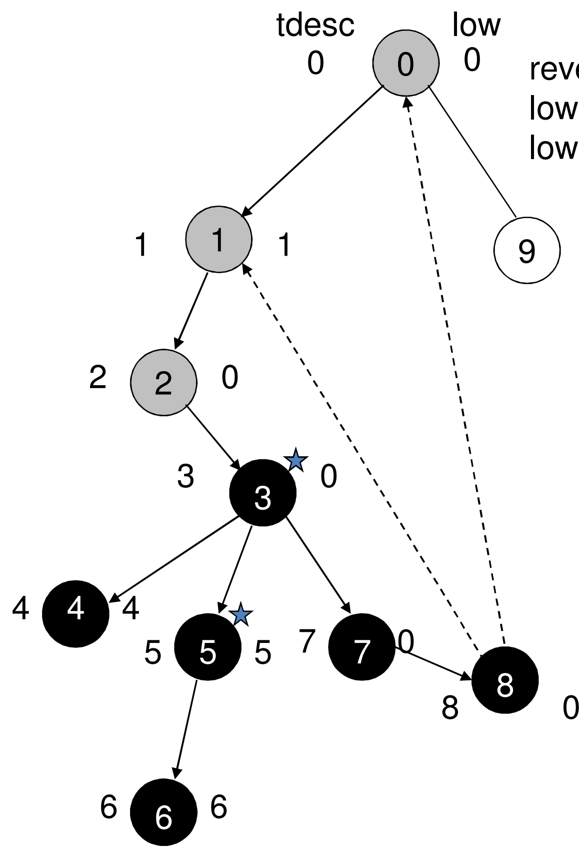
revenire la 7; verificare low copil:
low[7] = min(low[7], low[8]) = low[8]
low[8] >= tdesc[7] ? NU


tdesc
low
0
0
1
1
2
2
3
3
4
4
5
5
6
6
7
0
8
0
9

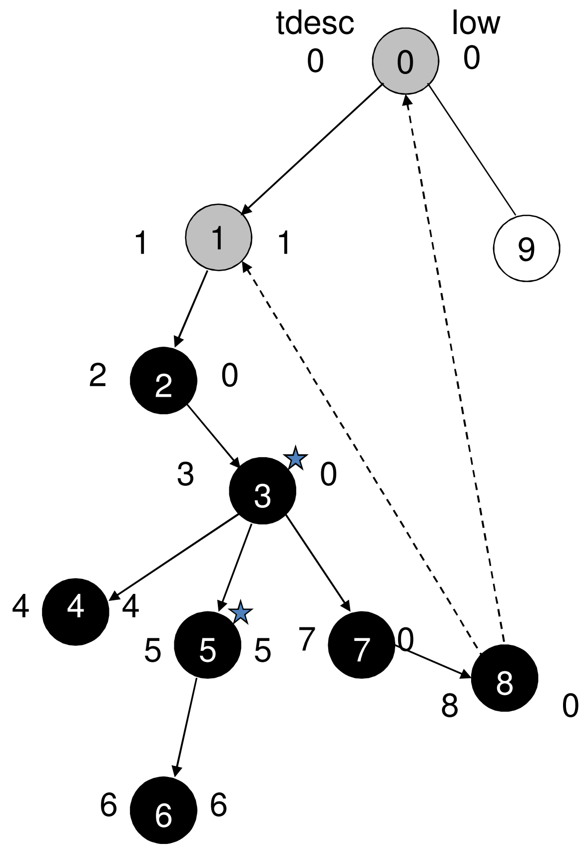


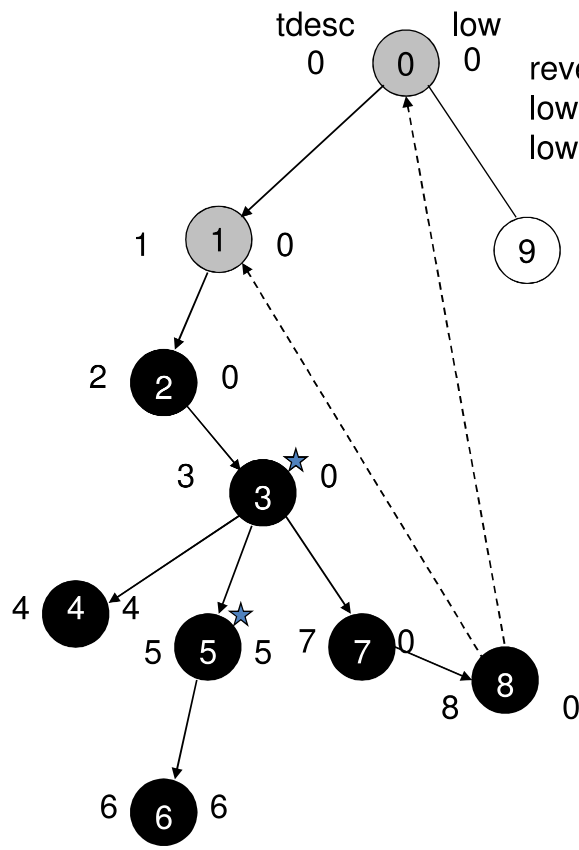
revenire la 3; verificare low copil:
low[3] = min(low[3], low[7]) = low[7]
low[7] >= tdesc[3] ? NU


tdesc
0
0
low
0
1
1
1
9
2
2
2
3
3
0
4
4
4
5
5
5
7
7
0
8
8
0
6
6
6

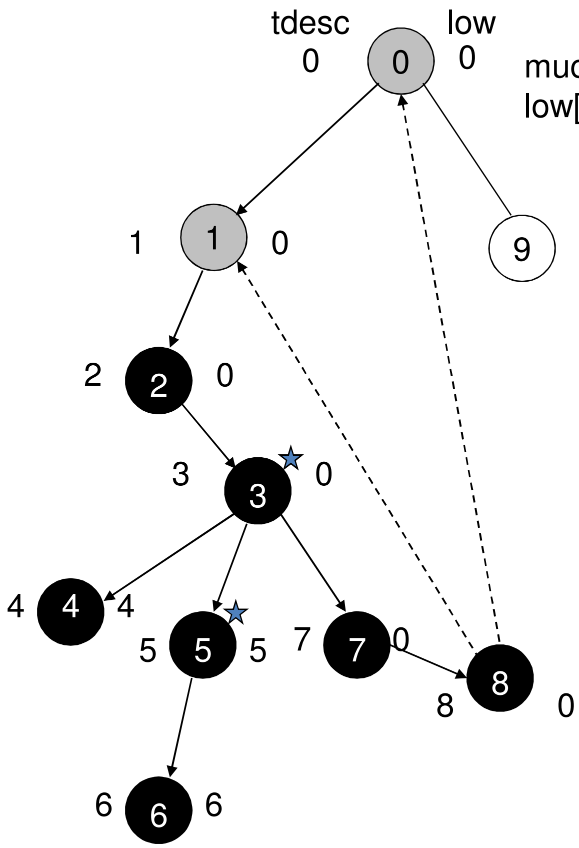


revenire la 2; verificare low copil:
low[2] = min(low[2], low[3]) = low[3]
low[3] >= tdesc[2] ? NU

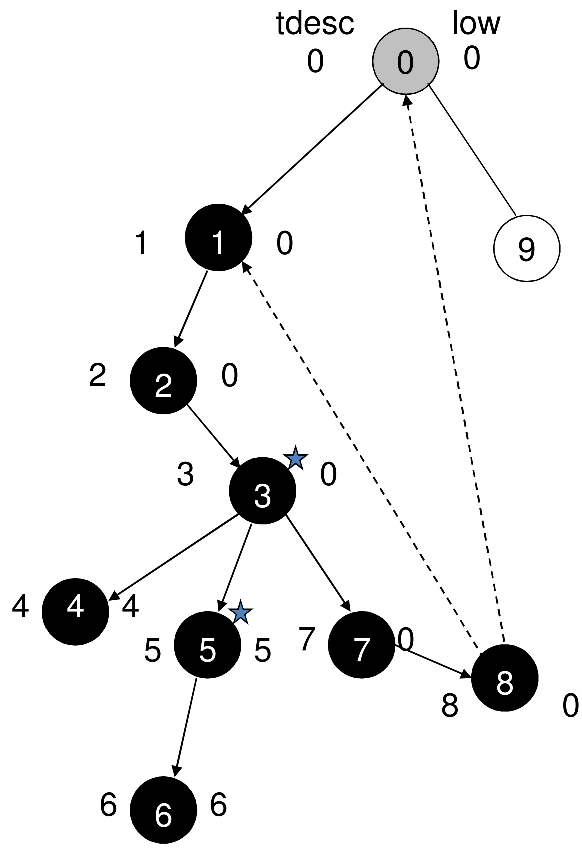

tdesc
low
0
0
0
1
1
1
9
2
2
0
3
3
0
4
4
4
5
5
5
7
7
0
8
8
0
6
6
6



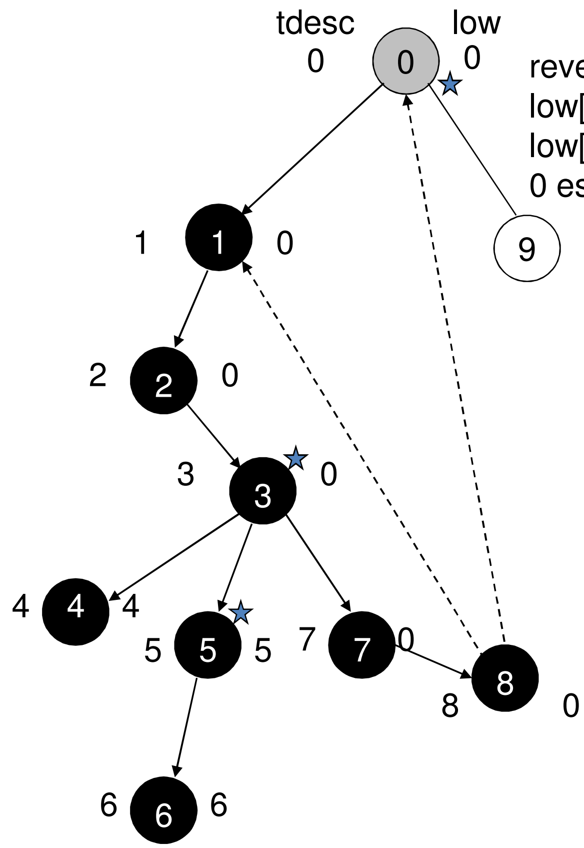
revenire la 1; verificare low copil:
low[1] = min(low[1], low[2]) = low[2]
low[2] >= tdesc[1] ? NU



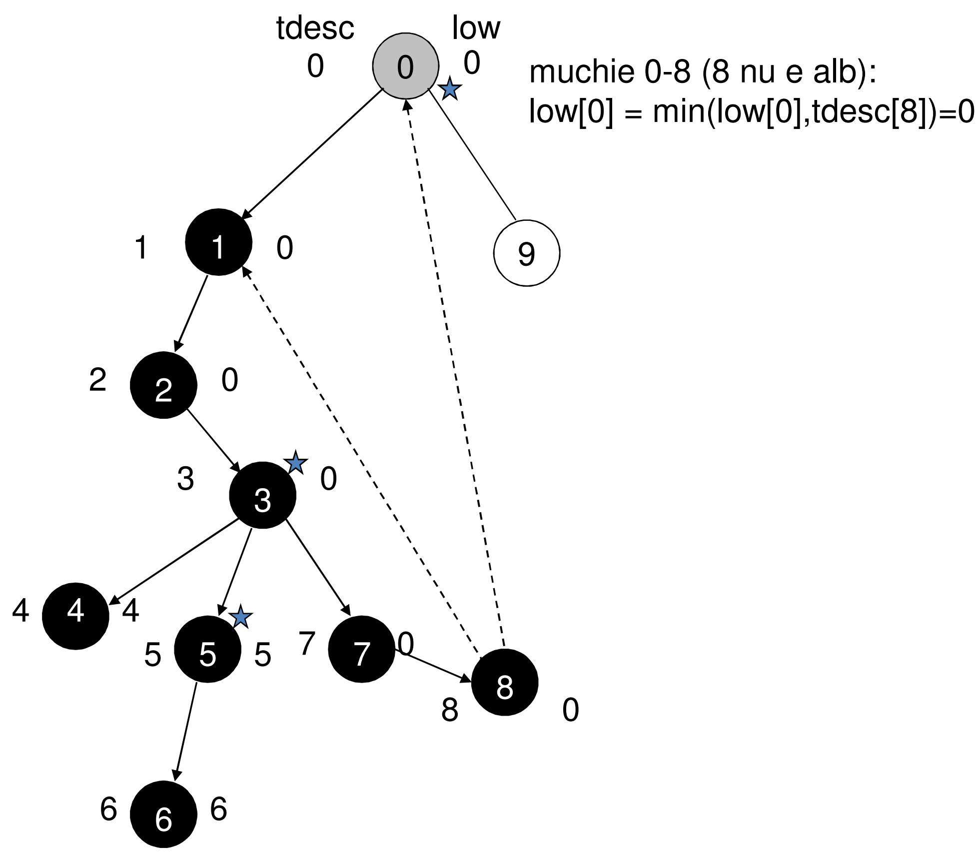
muchie 1-8 (8 nu e alb):
low[1] = min(low[1],tdesc[8])=0

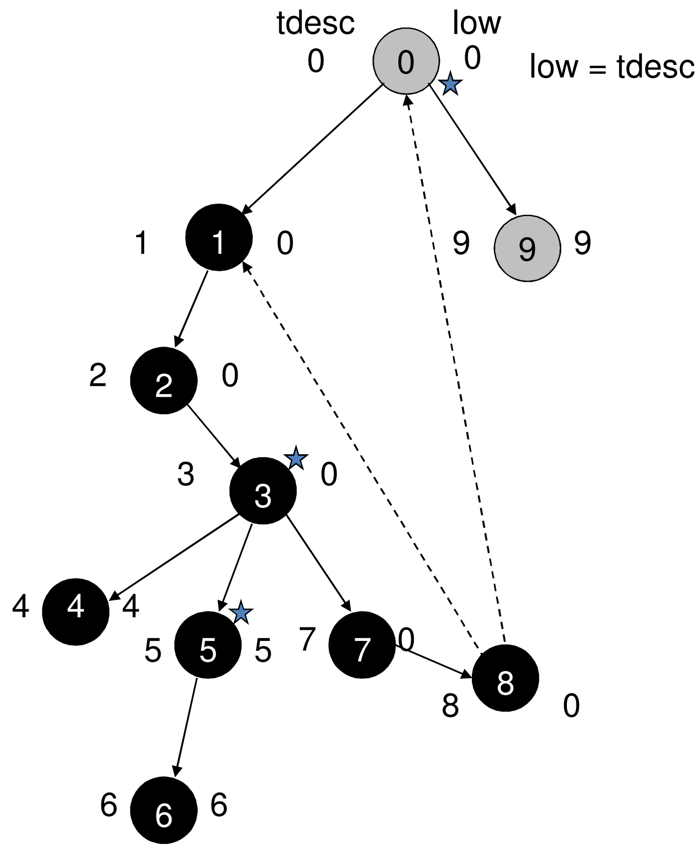


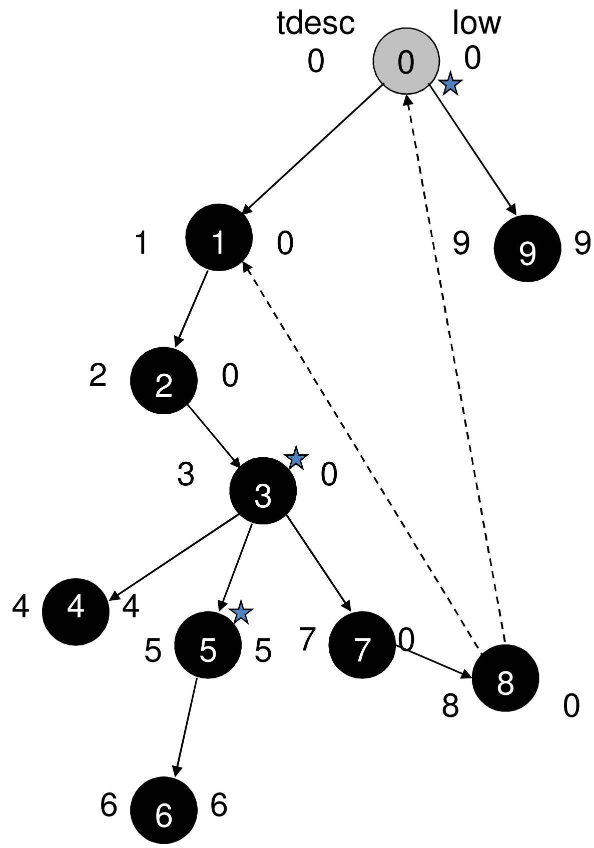
tdesc
low
0
0
0
1
1
0
9
2
2
0
3
3
0
4
4
4
5
5
5
7
7
0
8
8
0
6
6
6



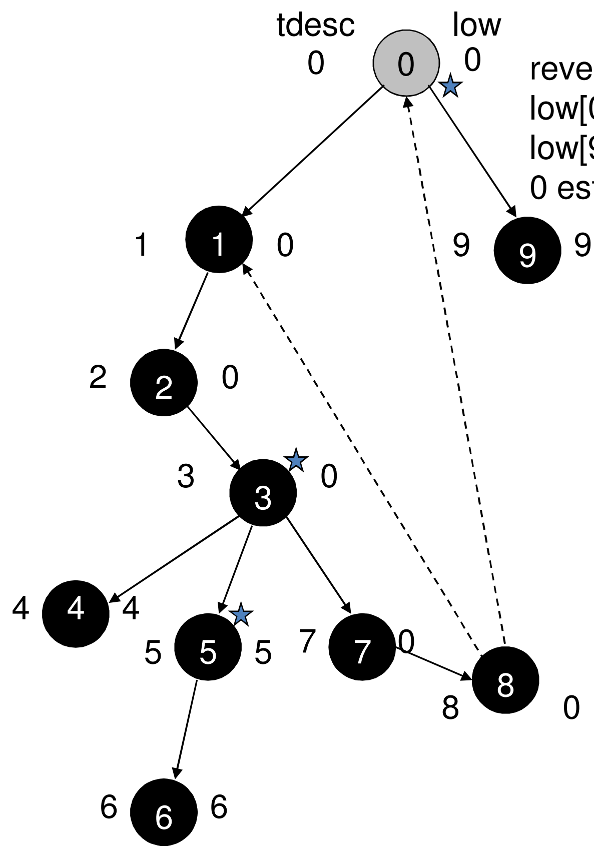
revenire la 0; verificare low copil:
low[0] = min(low[0], low[1]) = 0
low[1] >= tdesc[0] =>
0 este p. de art.


tdesc
low
0
0
0
muchie 0-8 (8 nu e alb):
low[0] = min(low[0],tdesc[8])=0
1
1
0
9
2
2
0
3
3
0
4
4
4
5
5
5
7
7
0
8
8
0
6
6
6

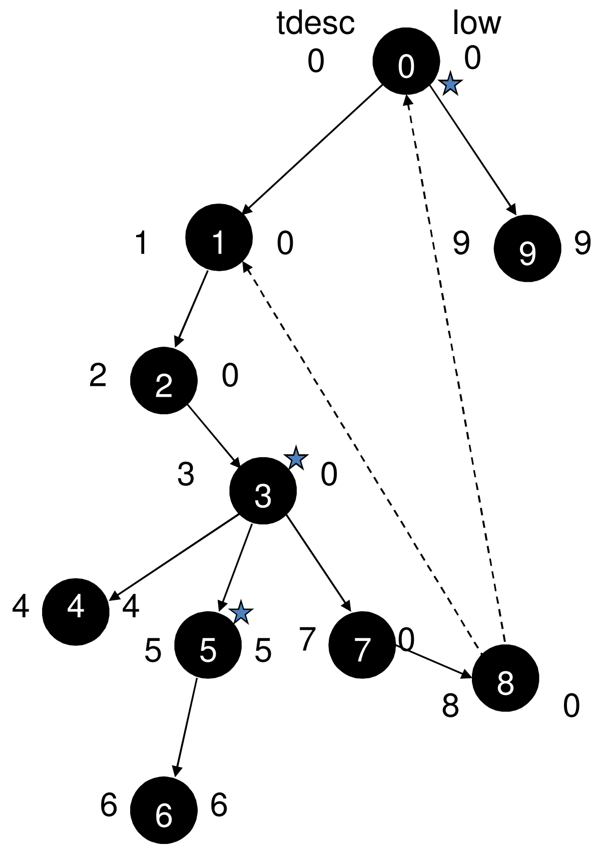

tdesc
low
0
0
0
low = tdesc
1
1
0
9
9
9
2
2
0
3
3
0
4
4
4
5
5
5
7
7
0
8
8
0
6
6
6



tdesc
low
0
0
0
1
1
0
9
9
9
2
2
0
3
3
0
4
4
4
5
5
5
7
7
0
8
8
0
6
6
6



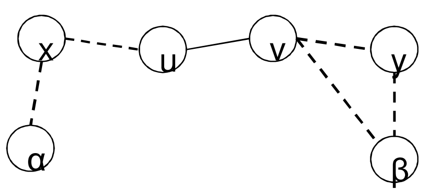
revenire la 0; verificare low copil:
low[0] = min(low[0], low[9]) = 0
low[9] >= tdesc[0] =>
0 este p. de art.



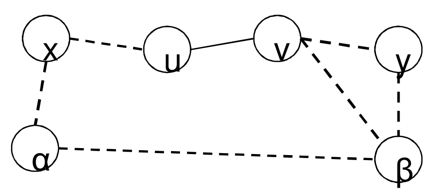
DFS finalizat;
rădăcina are 2 fii =>
0 punct de articulaţie
(altfel ar fi fost şters
marcajul)

Punti

- G=(V,E), graf neorientat si (u,v)∈ E
- (u,v) este punte in G **<=>** ∃x,y∈ V, x≠y, a.i. ∀ x..y contine muchia (u,v)



Orice drum x..y trece prin (u,v) =>(u,v) este punte

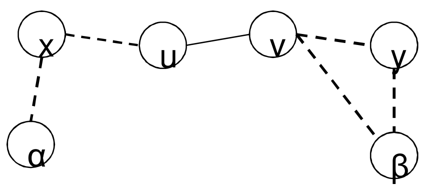


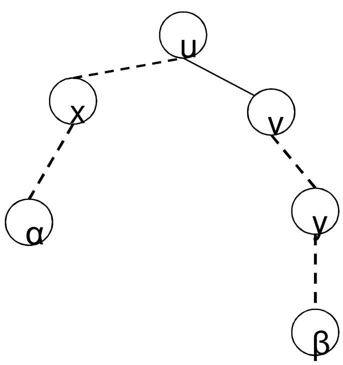
(u,v) nu este punte

Algoritm

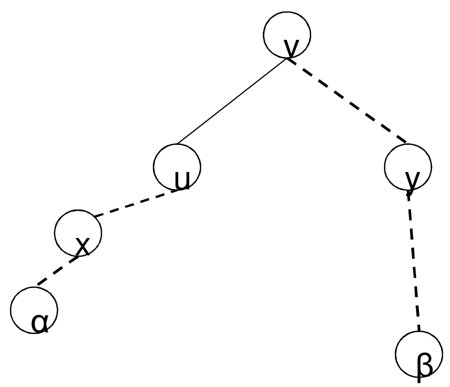
- Exploreaza(u)
  - d[u]=low[u]=timp++;
  - culoare[u]=gri;
  - foreach v succesor al lui u
    - If (culoare[v]==alb)
      - P[v]=u; subarb[u]++;
      - Exploreaza(v);
      - low[u]=min{low[u],low[v]}
      - If(low[v]>=d[u]) punte[v]=1; //If(p[u]!=null && low[v]>=d[u])
    - Else
      - low[u]=min{low[u], d[v]}

Exemplu





DFS din u; puntea este detectata in v



DFS din v; puntea este detectata in u