UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Proiectarea Algoritmilor

12. Puncte de articulație

Bibliografie

[1] Giumale – Introducere in Analiza Algoritmilor cap. 5.3, 5.4, 5.4.1

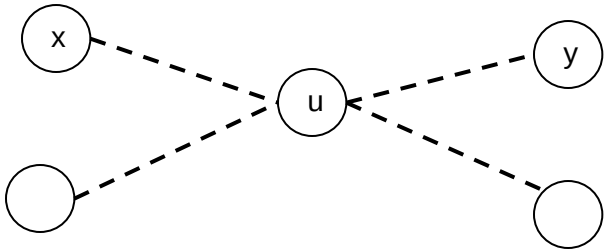
[2] Cormen – Introducere în Algoritmi cap. 20, 21, 25.1 si 25.2

[3] R. Sedgewick, K. Wayne - Algorithms and Data Structures Fall 2007 – Curs Princeton - http://www.cs.princeton.edu/~rs/AlgsDS07/

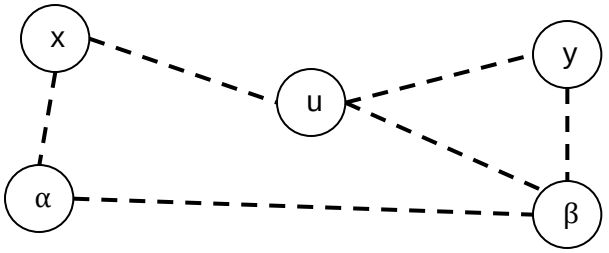
[4] Heap Fibonacci: http://www.cse.yorku.ca/~aaw/Jason/FibonacciHeapAnimation.html

Puncte de articulație. Def. Exemple

- Definiție: G = (V,E) graf neorientat, u∈V. U este punct de articulație dacă ∃ x,y∈V, x ≠ y, x ≠ u, y ≠ u, a.i. ∀ x..y în G trece prin u.



Orice drum x..y trece prin u → u este punct de articulație.



Exista x..α..y care nu trece prin u → u nu mai este punct de articulație!

Algoritm naiv de detectare a punctelor de articulație

- Elimină fiecare nod și verifică conectivitatea grafului rezultat:
  - Graf conex → nodul nu e punct de articulație.
  - Altfel → punct de articulație.

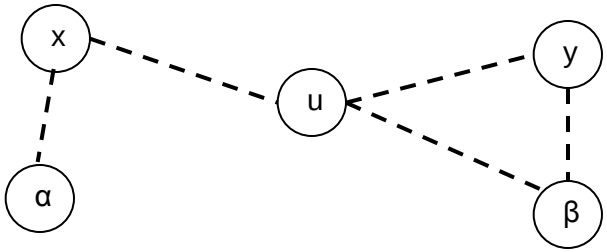
- Complexitate?
  - $O(V^2+E)$

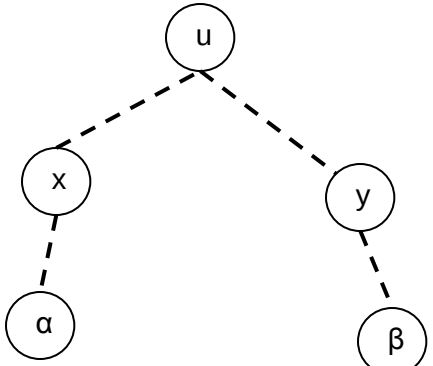
Puncte de articulație. Teoremă

- Teorema 5.15: G = (V,E), graf neorientat și $u \in V$. U este punct de articulație în G $\Leftrightarrow$ în urma DFS în G una din proprietățile de mai jos este satisfăcută:
  - p(u) = null și u domină cel puțin 2 subarbori;
  - p(u) ≠ null și $\exists v$ descendent al lui u în Arb(u) a.i. $\forall x \in$ Arb(v) și $\forall$(x,z) parcurs de DFS(G) avem d(z) ≥ d(u).

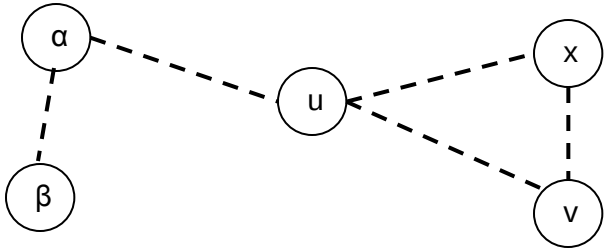
Situații posibile

- 1) p(u) = null și u domina cel puțin 2 subarbori:

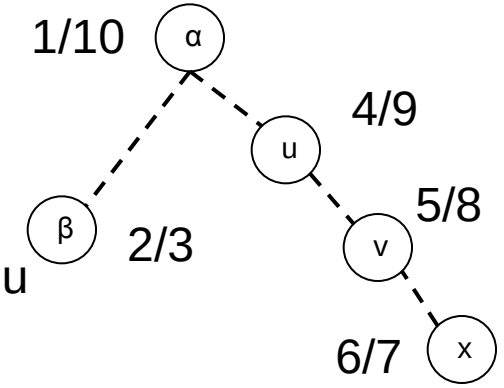




- 2) p(u) ≠ null și ∃v descendent al lui u în Arb(u) a.i. ∀x∈Arb(v) și ∀(x,z) parcurs de DFS(G) d(z) ≥ d(u):

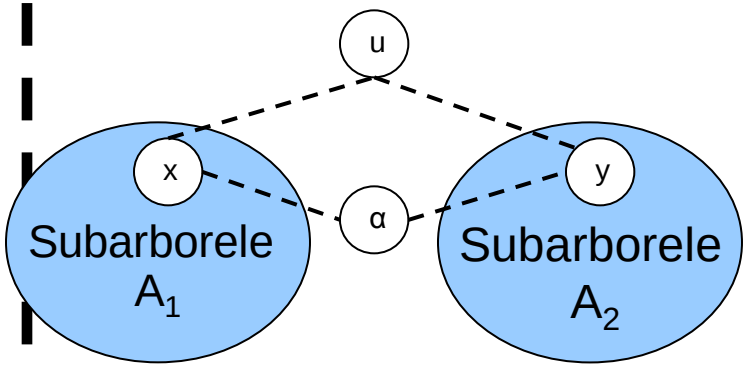


Pentru orice muchie din subarborele lui v nu există nici o muchie înapoi spre un nod descoperit înaintea lui u.



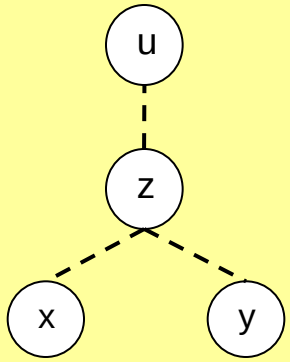
Puncte de articulație. Demonstrație teoremă (Ia)

- p(u) = null și u domină cel puțin 2 subarbori ⟹ u este punct de articulație.
- Dem (Reducere la absurd): Fie A_1 și A_2 cei 2 subarbori, $x \in A_1$, $y \in A_2$. Pp $\exists$ x..α..y și u $\notin$ x..α..y.
- z = primul nod descoperit la DFS din care se poate ajunge la x si la y. Cf. T drumurilor albe x,y ∈ Arb(z).
- Dar x,y∈Arb(u) ➔ 2 cazuri:



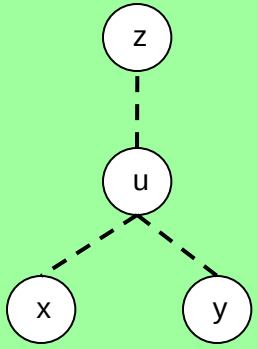
Pp $\exists$ x..α..y si u$\notin$ x..α..y.

Caz 1: d(u)<d(z):



Contradictie (1) x,y nu sunt in subarbori diferiti ai lui Arb(u).

Caz 2: d(z)<d(u):



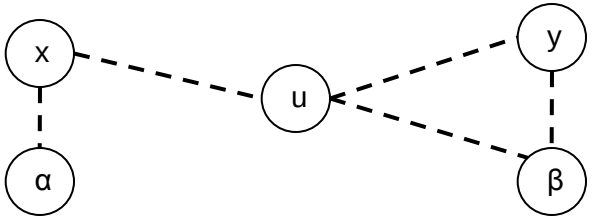
Contradictie (1), p(u) ≠ null.

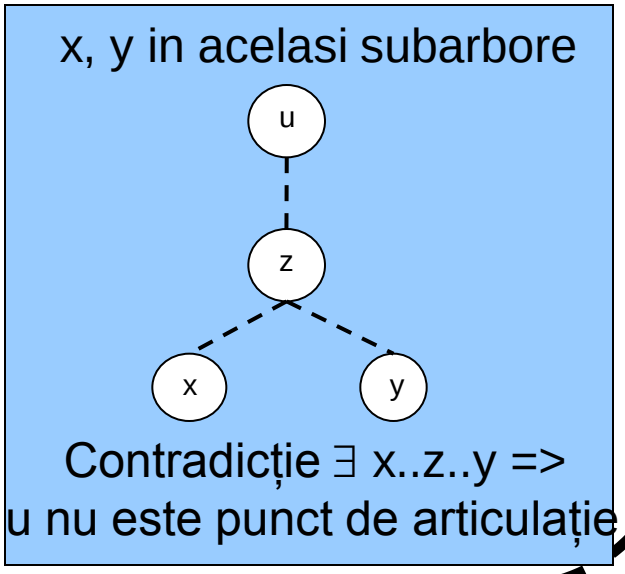
Puncte de articulație. Demonstrație teoremă (Ib)

- u este punct de articulație si este descoperit in ciclul principal al DFS ⟹ p(u) = null și u domina cel puțin 2 subarbori.

- Dem (Reducere la absurd): Fie nodurile x și y a.i. u ∈ ∀ x..y. u = primul nod descoperit din cale (altfel u nu mai e descoperit în ciclul principal al DFS) => p(u) = null și x, y ∈ Arb(u).

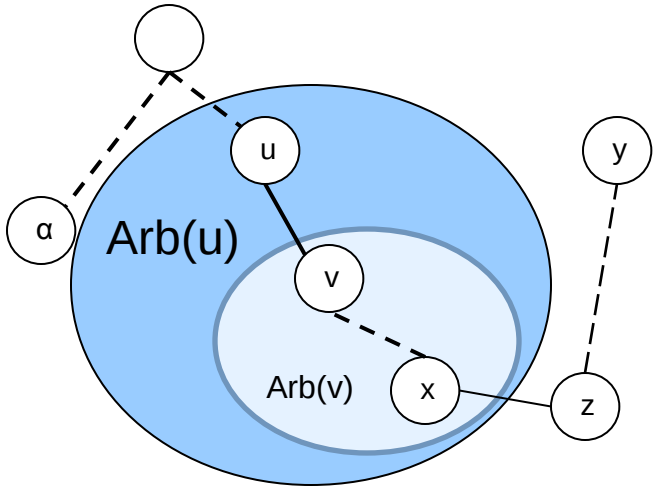
- pp că x, y sunt în același subarbore → fie z rădăcina celor 2 subarbori → ∃ x..z..y → u nu e punct de articulație → se contrazice ipoteza.





Puncte de articulație. Demonstrație teoremă (IIa)

- p(u) ≠ null și ∃ v descendent al lui u în Arb(u) a.i. ∀ x ∈ Arb(v) și ∀ (x,z) parcurs de DFS(G) având d(z) ≥ d(u) ⟹ u este punct de articulație.



Dem (Reducere la absurd): Pp. u nu e punct de articulație → ∃ w ∈ Arb(v), y ∉ Arb(u) a.i. y..w. Fie z primul nod din y..w a.i. z ∉ Arb(u) si x ultimul nod din w..y a.i. x ∈ Arb(u) → (x,z) taie frontiera Arb(u).

Dacă d(z) > d(u) → u..x,z alb la d(u) → z ∈ Arb(u) → contradicție (z ∉ Arb(u))

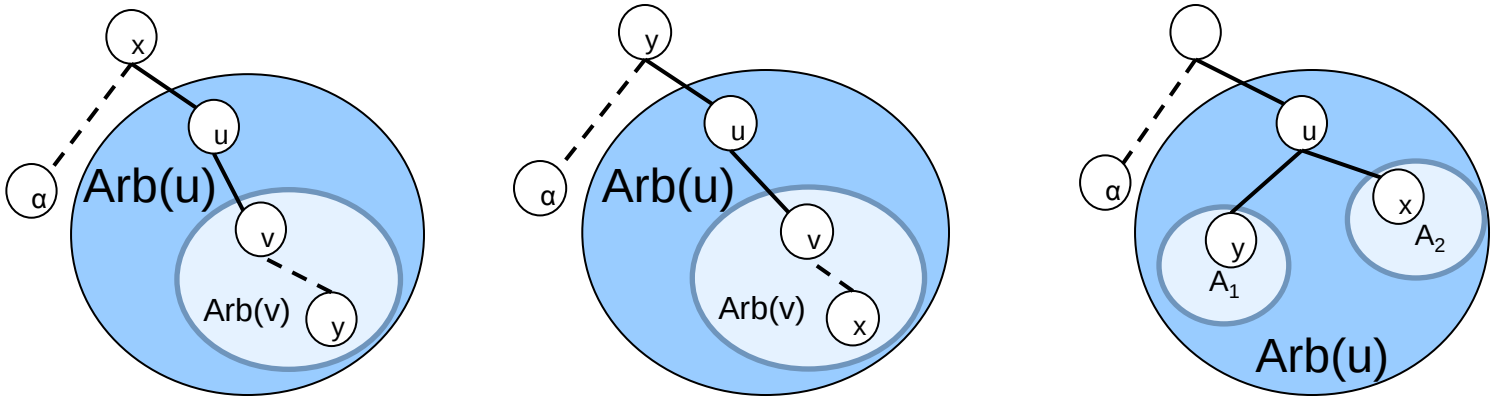
Dacă d(z) < d(u) → contradicție (ipoteza)

→ ∄ y..w → u punct de articulație

Puncte de articulație. Demonstrație teoremă (IIb)

- u este punct de articulație și nu este descoperit în ciclul principal al DFS $\Longrightarrow$ p(u) ≠ null 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) având d(z) ≥ d(u).

- Dem: Fie nodurile x și y a.i. u $\in \forall$ x..y și p(u) ≠ null. Se pot forma 3 tipuri de structuri:



- Pentru primele 2 structuri, nu trebuie sa existe muchie care sa formeze ciclu de la nici un nod din Arb(v) către vreun predecesor al lui u. Altfel $\exists$ x..y a.i. u $\notin$ x..y.

- Pentru a 3-a structura, trebuie să $\nexists$ muchie care să formeze ciclu către un predecesor al lui u de la niciun nod din cel puțin un subarbore A_1 sau A_2.

Puncte de articulație. Structuri de date.

- Structura de date de la DFS + pentru fiecare nod u ∈ V se reține:
  - Low(u) = min{d(v) | v descoperit pornind din u in cursul DFS și c(v) ≠ alb}
  - Subarb(u) = numărul subarborilor dominați de u (dacă e ≥ 2, atunci avem un punct de articulație).

Idee algoritm

- Se aplică DFS și se salvează pentru fiecare nod până unde merge înapoi (low): low[u] = min {d(u), d(v) pt. toate muchiile înapoi (u,v), low(w) pentru toți fii w ai lui u}.

- Pentru eficiență, trebuie ca fii să se parcurgă înaintea părinților → ordinea inversă a d(u).

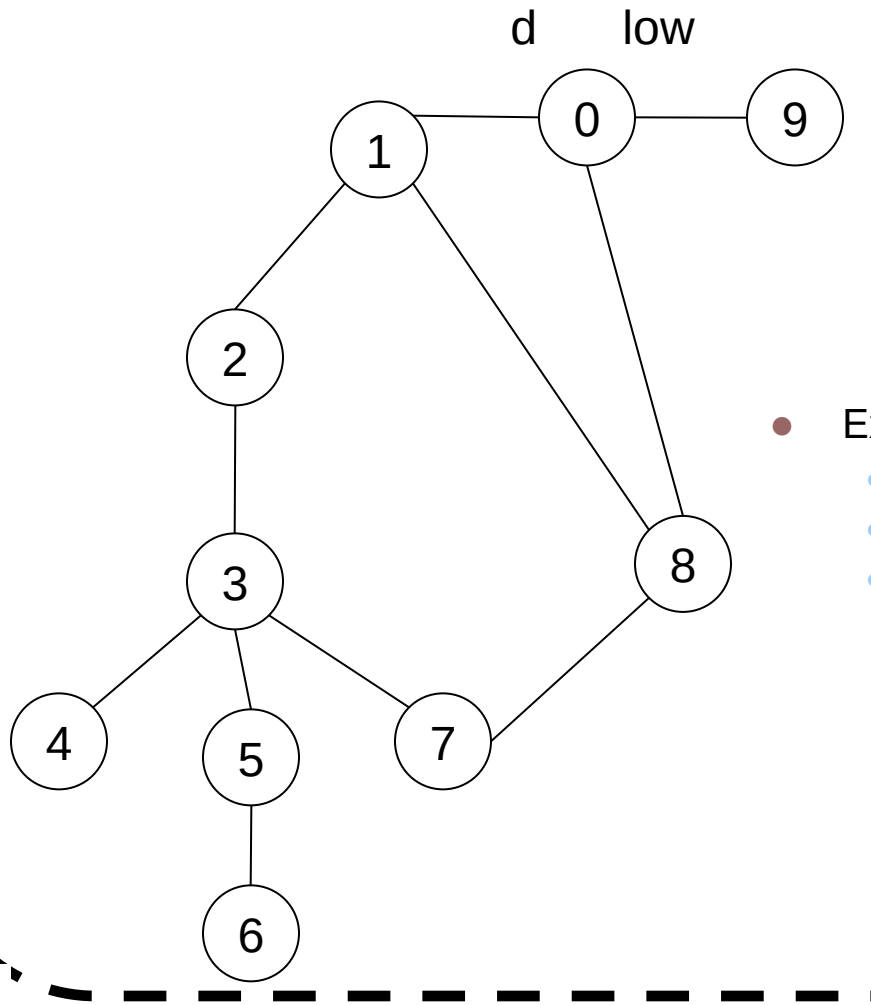
Algoritm Tarjan (I)

- Articulații (G)
  - V = noduri(G) // inițializări
  - Timp = 0;
  - **Pentru fiecare** (u ∈ V)
    - culoare[u] = alb;
    - d[u] = 0;
    - p[u] = null;
    - low[u] = 0;
    - subarb[u] = 0; // reține numărul de subarbori dominați de u
    - art[u] = 0; // reține punctele de articulație
  - **Pentru fiecare** (u∈V)
    - **Dacă** (culoare(u) e alb)
      - Exploreaza(u);
      - **Dacă** (subarb[u] > 1) // cazul în care u este rădăcina în arborele
        - art[u] = 1 // DFS și are mai mulți subarbori → cazul // 1 al teoremei

Algoritm Tarjan (II)

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr subarbori
        // dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1;
        // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

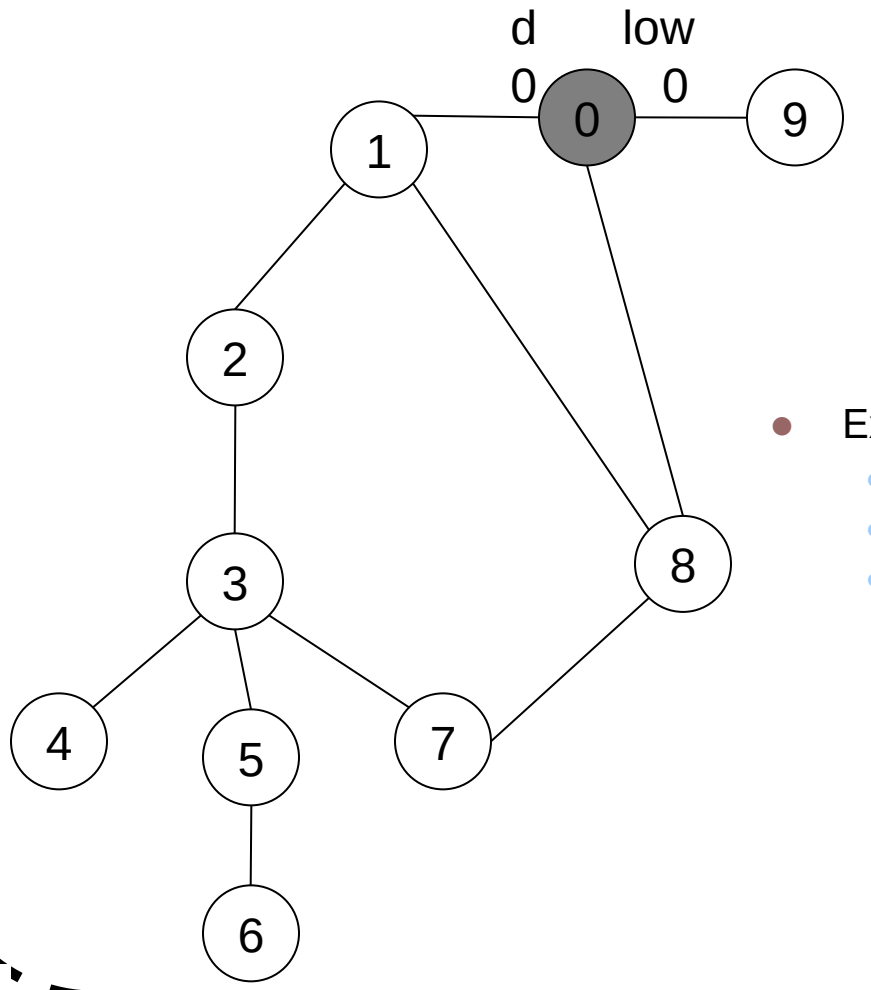
Exemplu rulare (1)



Timp = 0
Cul[i] = alb
d[i] = 0
Low[i] = 0
P[i] = null
Subarb[i] =0
Art[i] = 0
Exploreaza (0)

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

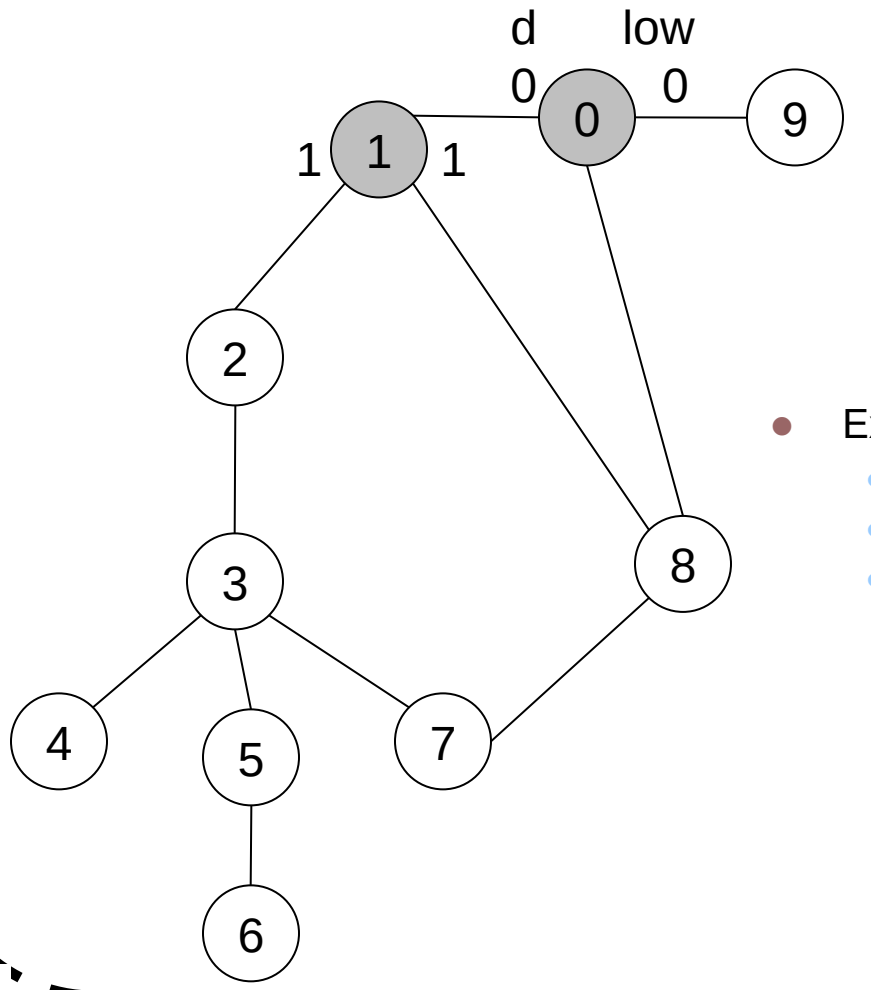
Exemplu rulare (2)



Low[0] = d[0] = 0
Timp = 1
Cul[0] = gri
P[1] = 0
Subarb[0] = 1
Exploreaza (1)

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

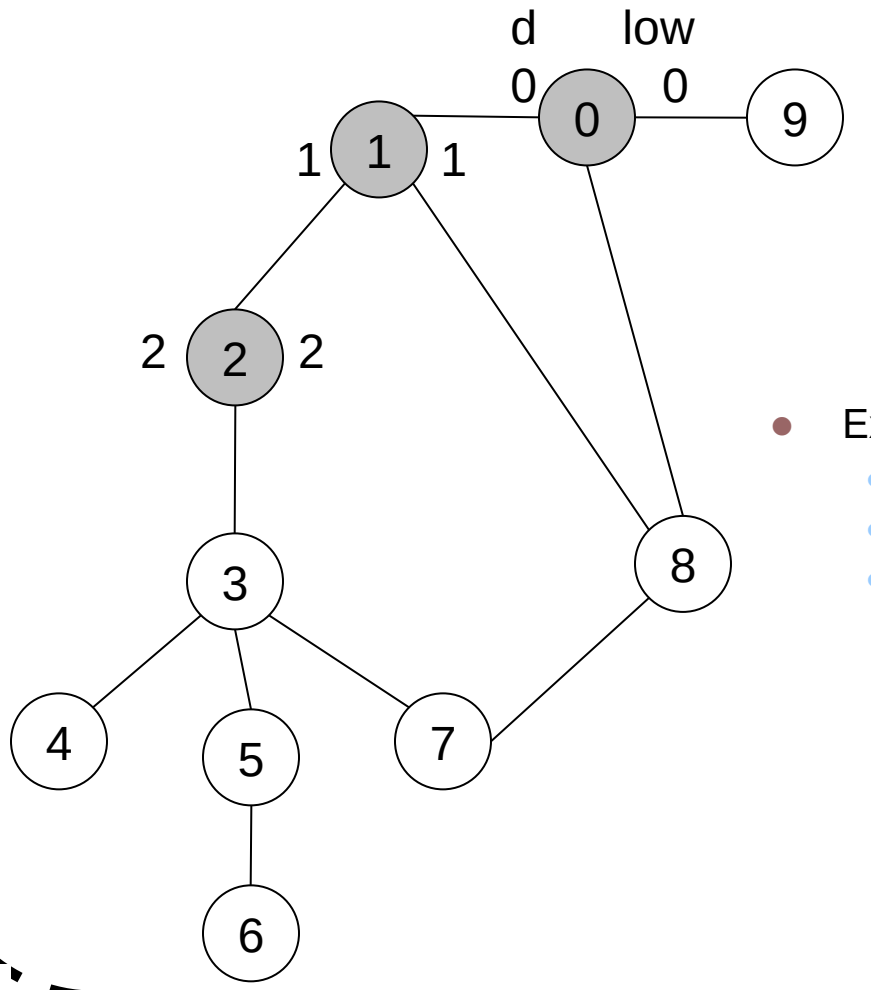
Exemplu rulare (3)



Low[1] = d[1] = 1
Timp = 2
Cul[1] = gri
P[2] = 1
Subarb[1] = 1
Exploreaza (2)

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

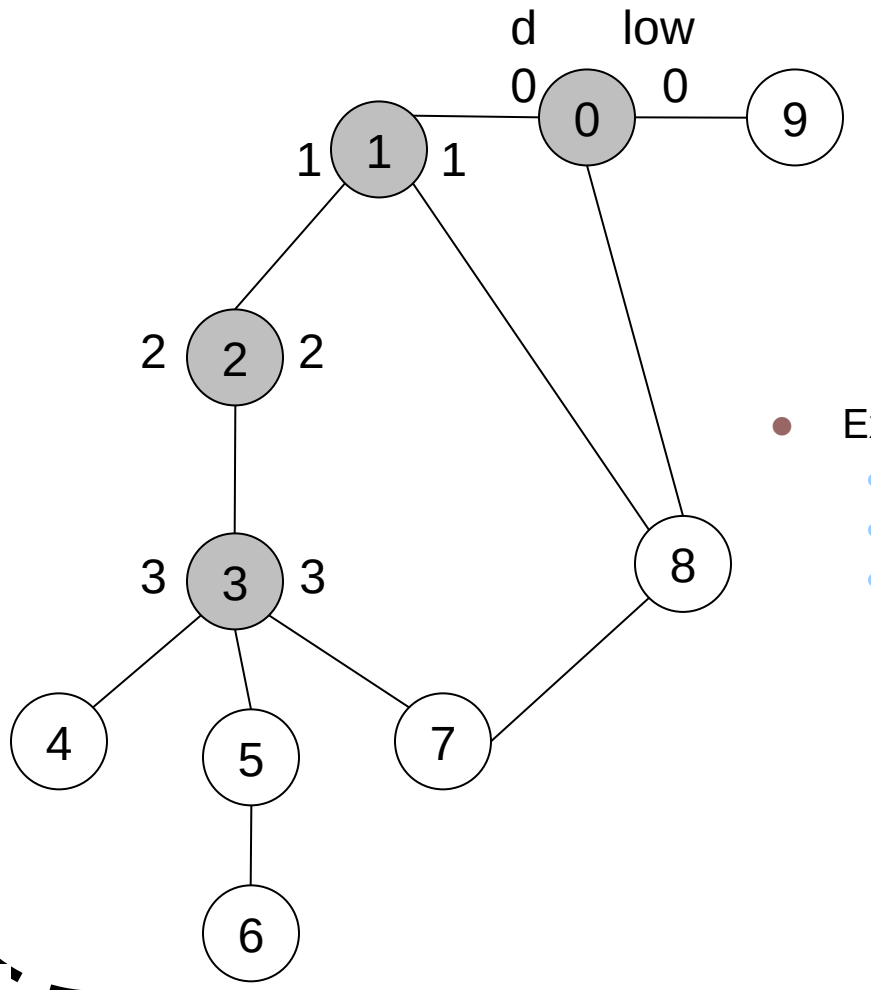
Exemplu rulare (4)



Low[2] = d[2] = 2
Timp = 3
Cul[2] = gri
P[3] = 2
Subarb[2] = 1
Exploreaza (3)

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

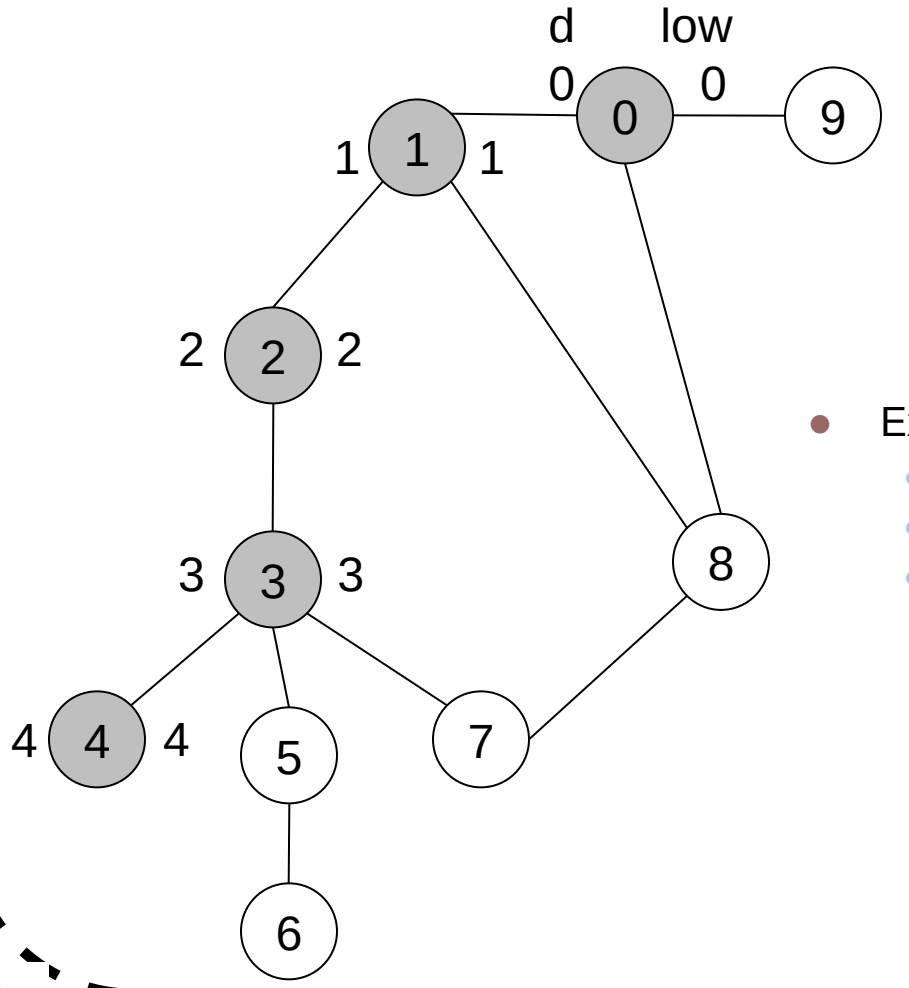
Exemplu rulare (5)



Low[3] = d[3] = 3
Timp = 4
Cul[3] = gri
P[4] = 3
Subarb[3] = 1
Exploreaza (4)

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

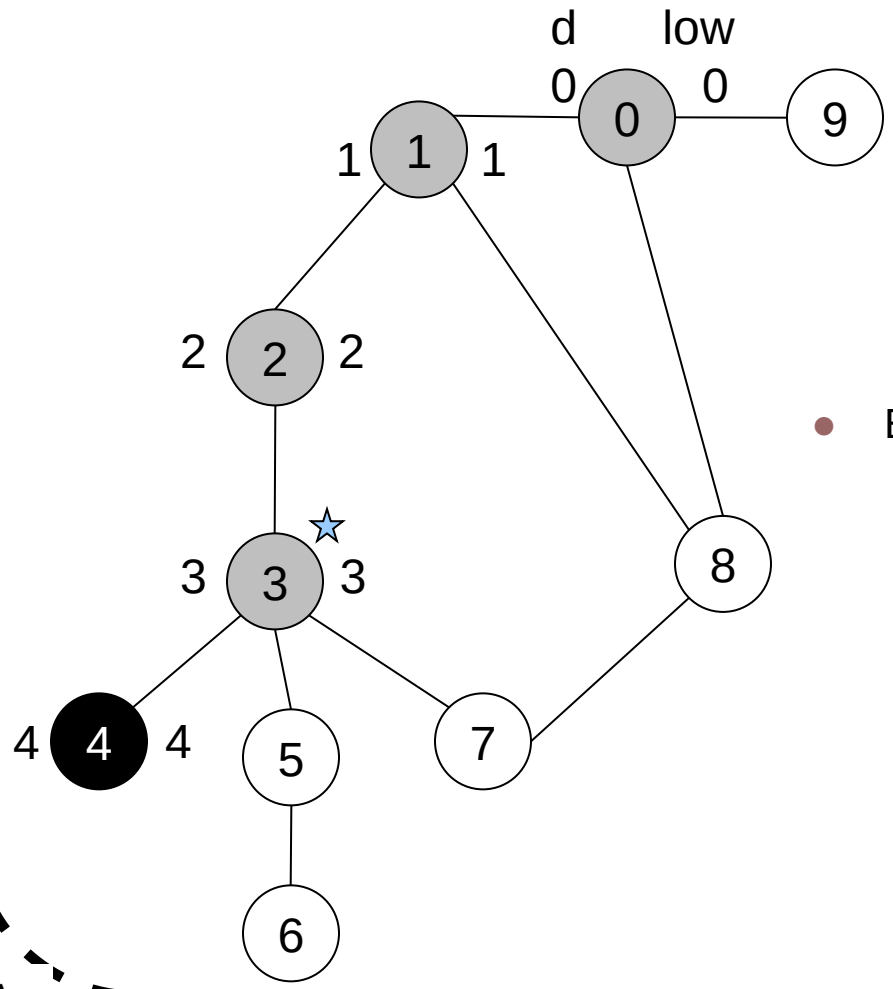
Exemplu rulare (6)



Low[4] = d[4]=4
Timp =5
Cul[4] =gri
revenire

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

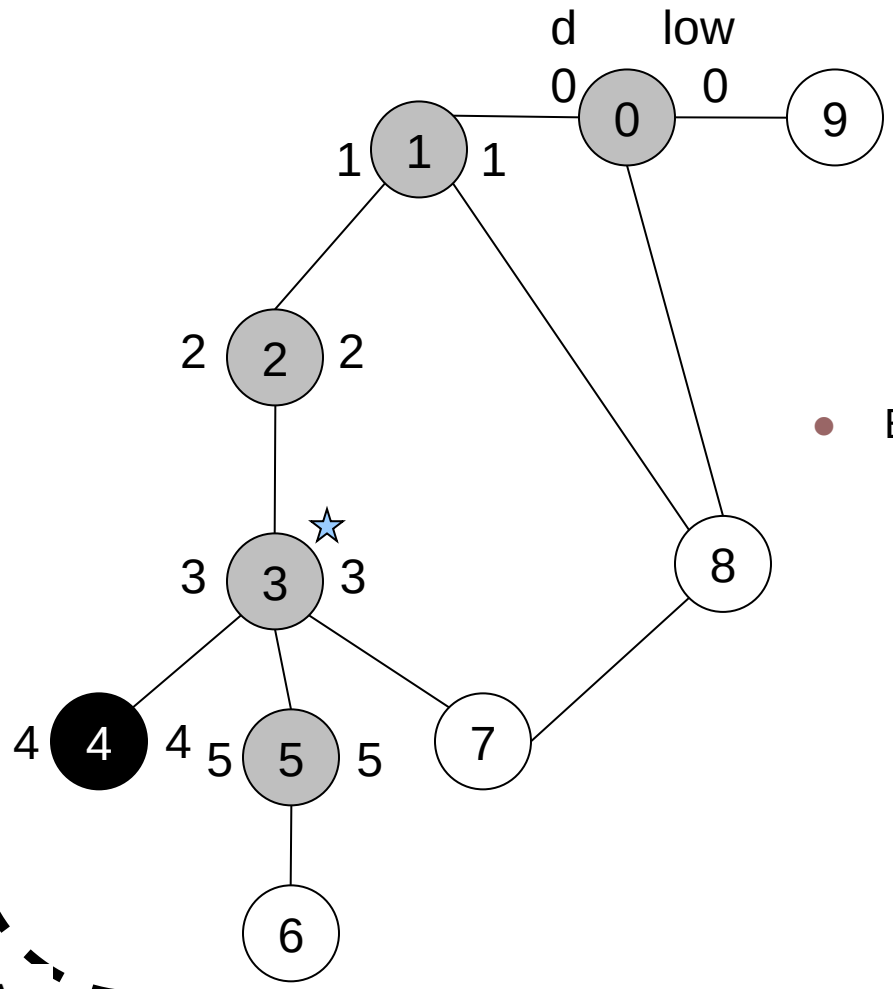
Exemplu rulare (7)



Low[4] = d[4] = 4
Timp = 5
Cul[4] = gri
revenire
Low [3] = min {low[3], low[4]} = 3
Low[4] > d[3] → art[3] = 1
P[5] = 3
Subarb[3] = 2
Exploreaza (5)

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

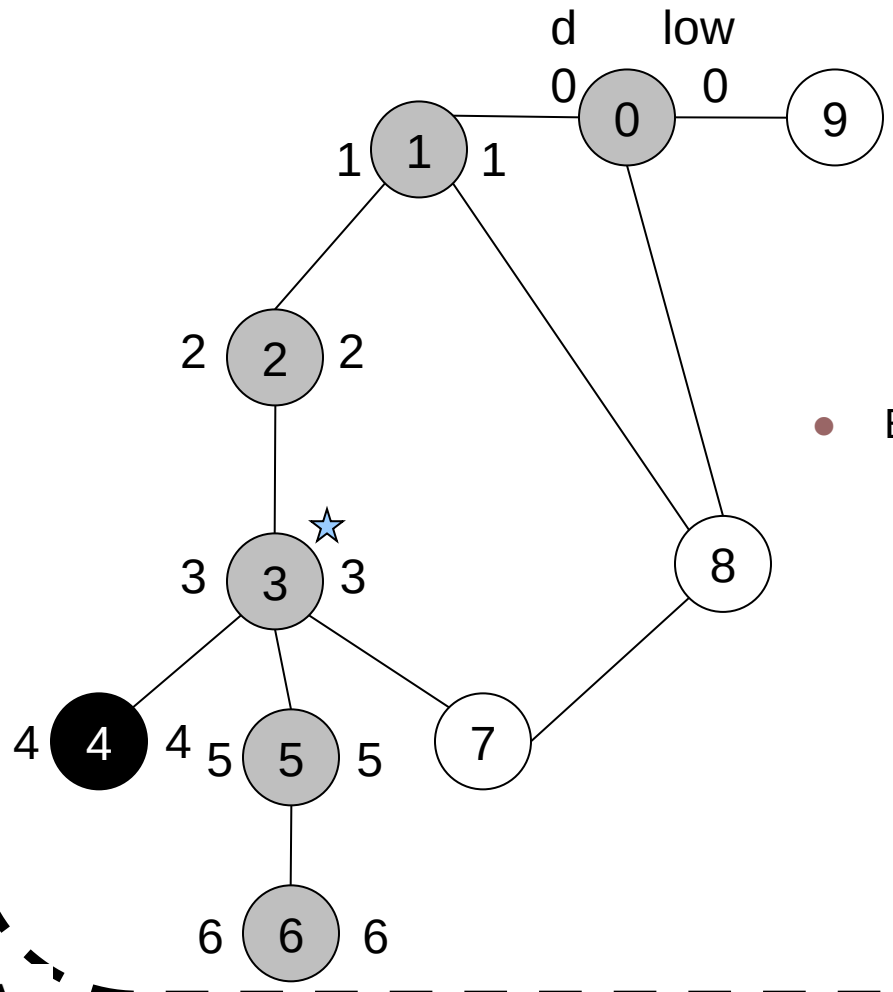
Exemplu rulare (8)



Low[5] = d[5] = 5
Timp = 6
Cul[5] = gri
P[6] = 5
Subarb[5] = 1
Exploreaza (6)

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

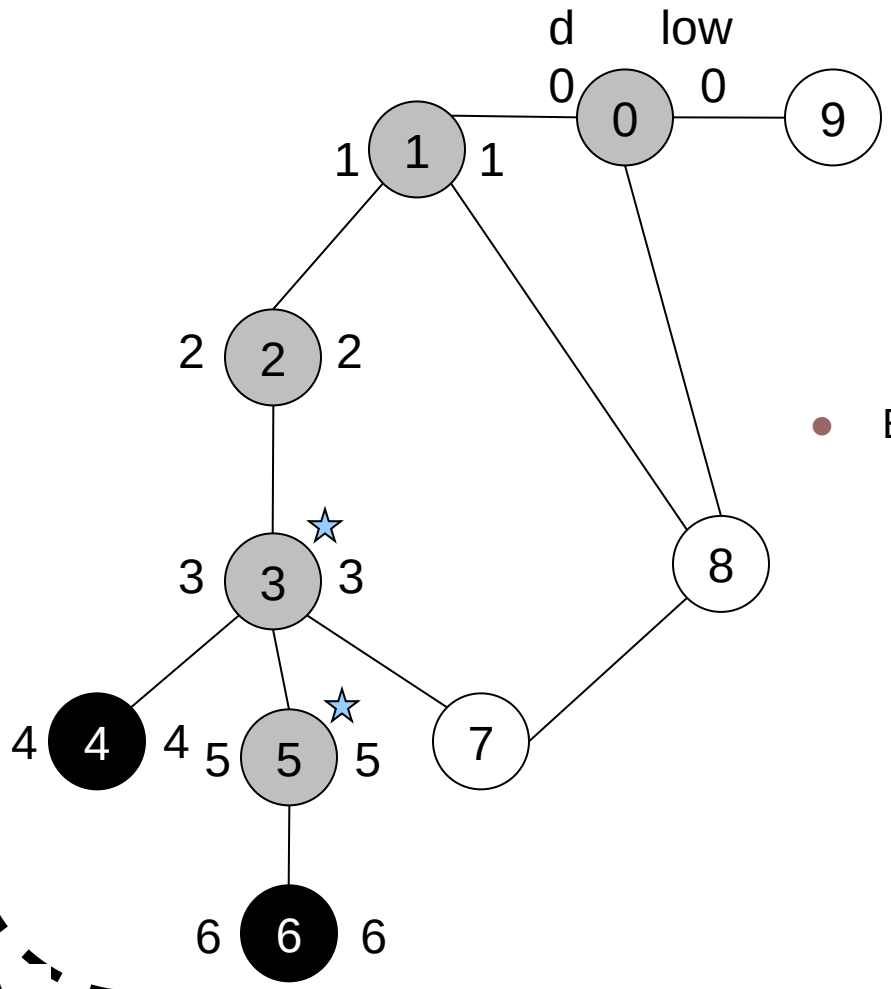
Exemplu rulare (9)



Low[6] = d[6] = 6
Timp = 7
Cul[6] = gri
revenire

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

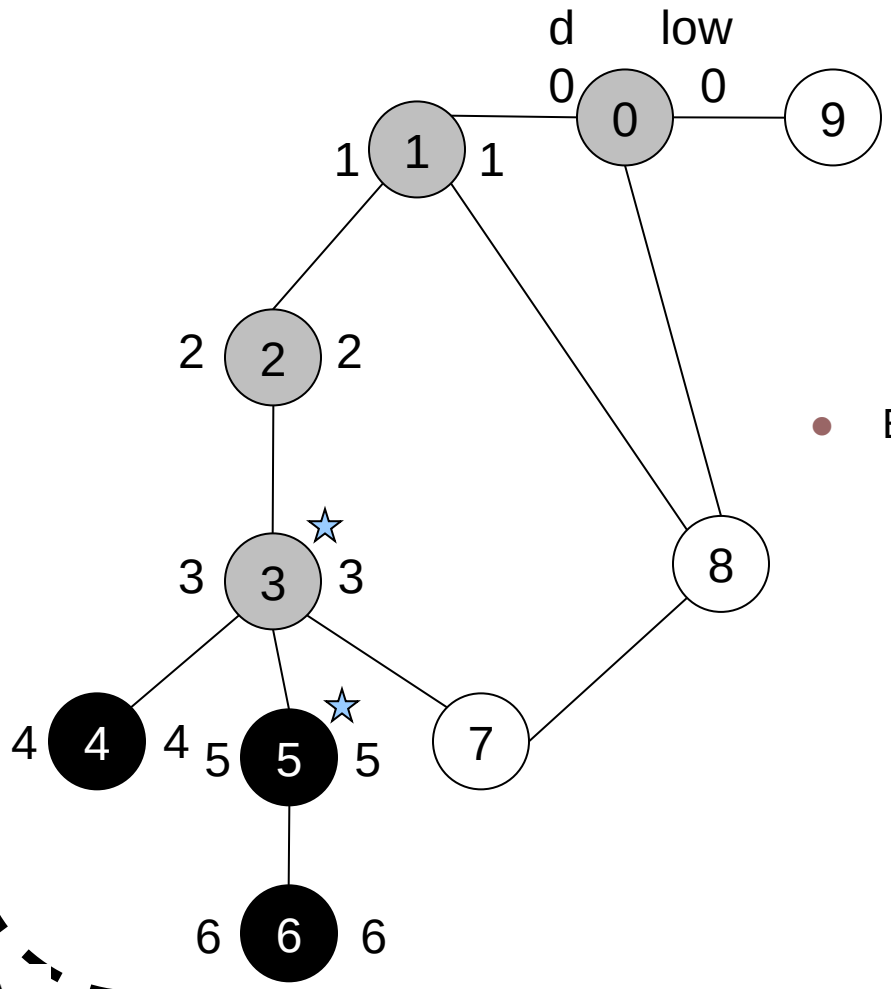
Exemplu rulare (10)



Low[6] = d[6] = 6
Timp = 7
Cul[6] = gri
revenire
Low [5] = min {low[5], low[6]} = 5
Low[6] > d[5] → art[5] = 1
revenire

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

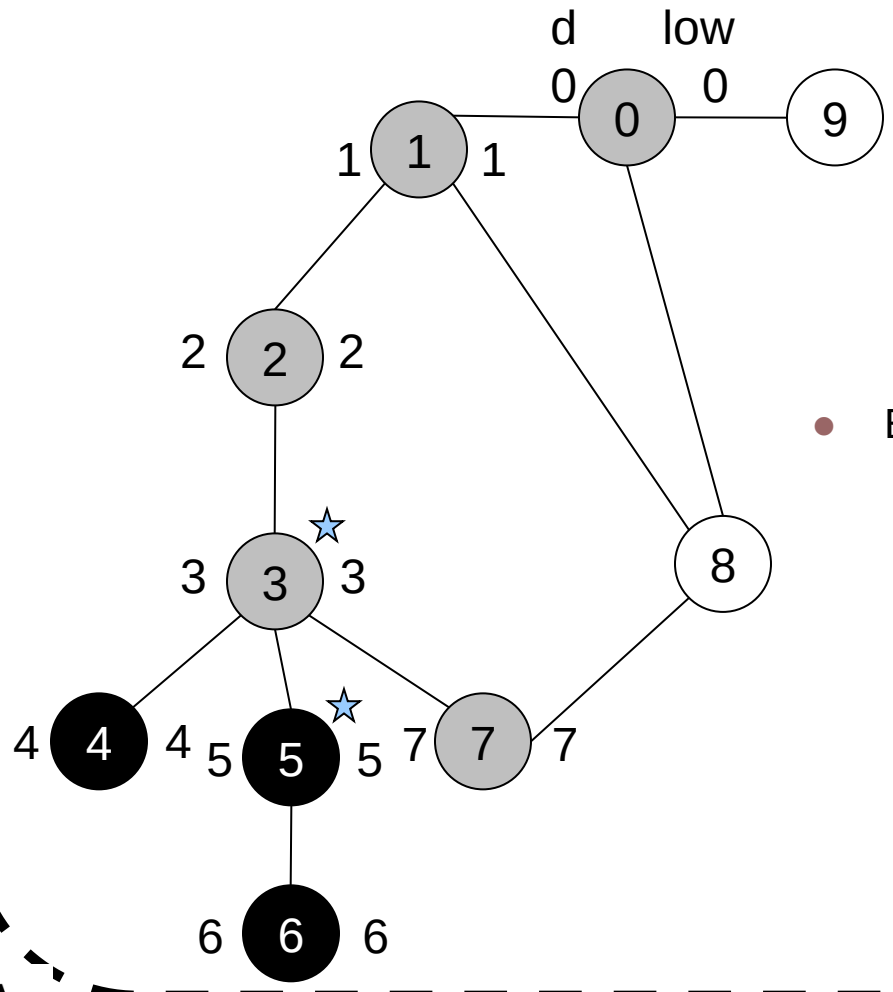
Exemplu rulare (11)



Low[5] = d[5] = 5
Timp = 7
Cul[5] = gri
revenire
Low [3] = min {low[3], low[5]} = 3
Low[5] > d[3] → art[3] = 1
P[7] = 3
Subarb[3] = 3
Exploreaza (7)

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

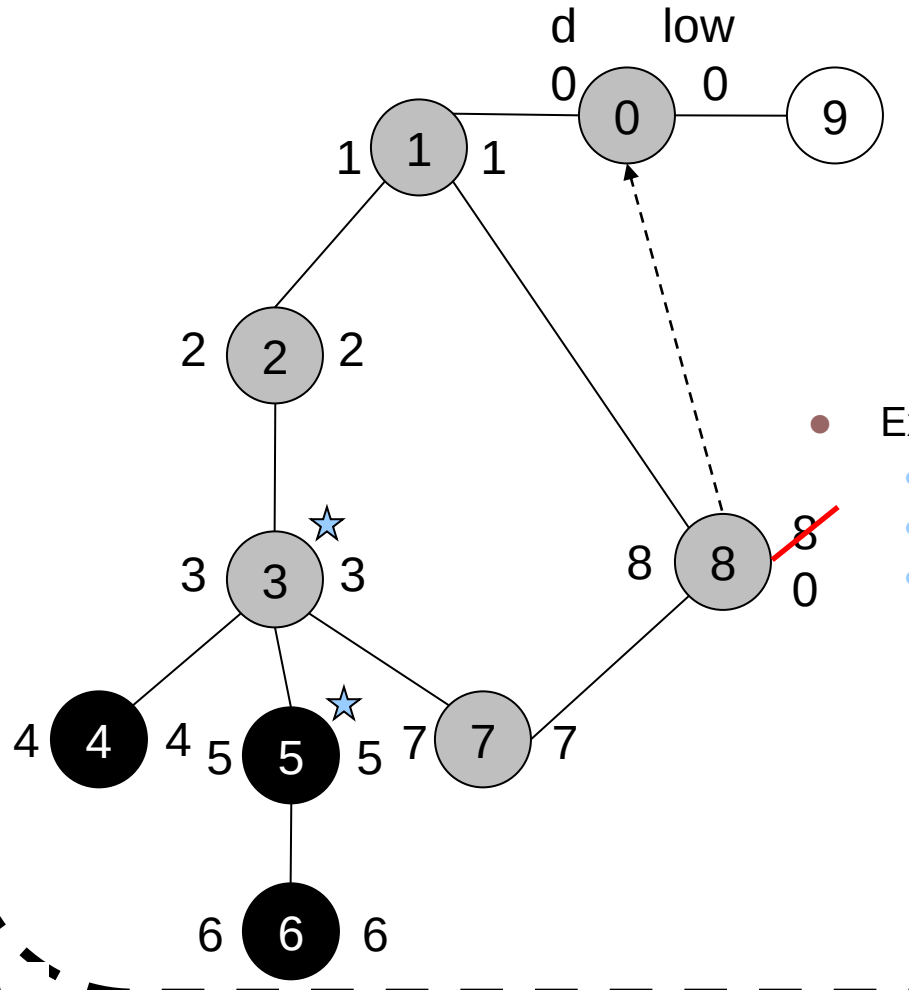
Exemplu rulare (12)



Low[7] = d[7] = 7
Timp = 8
Cul[7] = gri
P[8] = 7
Subarb[7] = 1
Exploreaza (8)

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

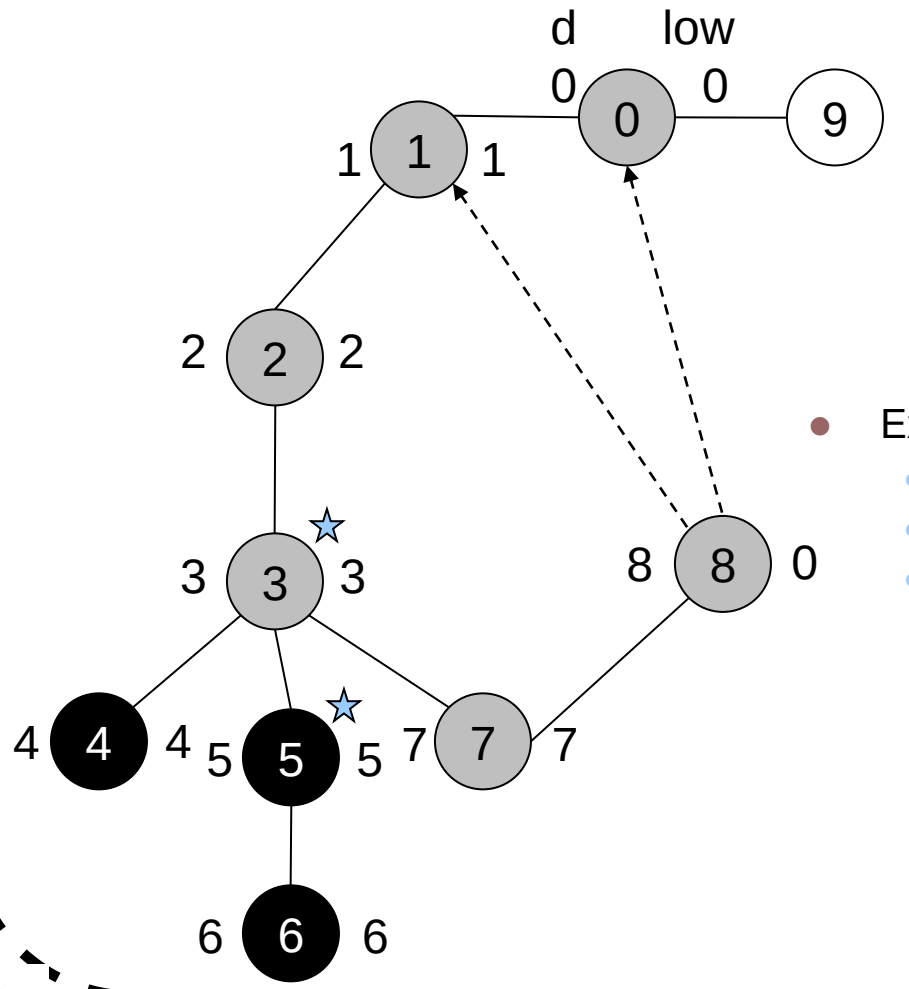
Exemplu rulare (13)



Low[8] = d[8] = 8
Timp = 9
Cul[8] = gri
Low[8] = min{d[0], low[8]} = 0

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

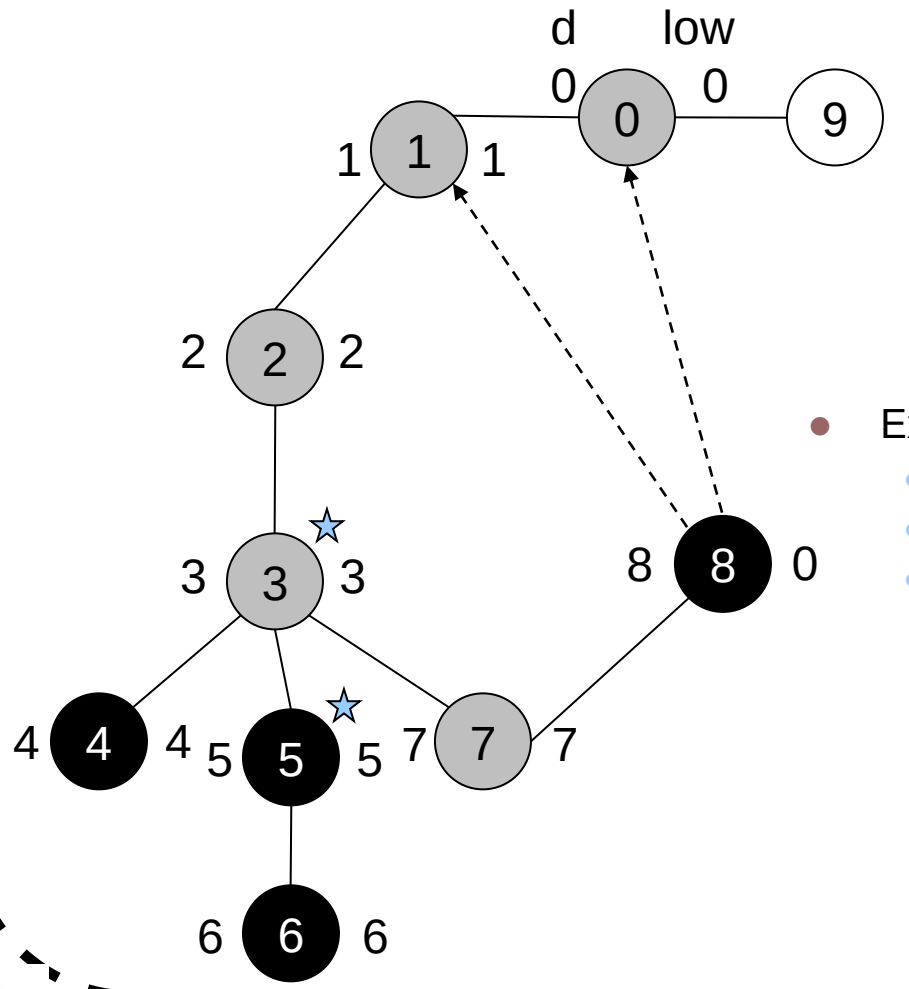
Exemplu rulare (14)



d[8] = 8
Low[8] = 0
Timp = 9
Cul[8] = gri
Low[8] = min{d[1], low[8]} = 0

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

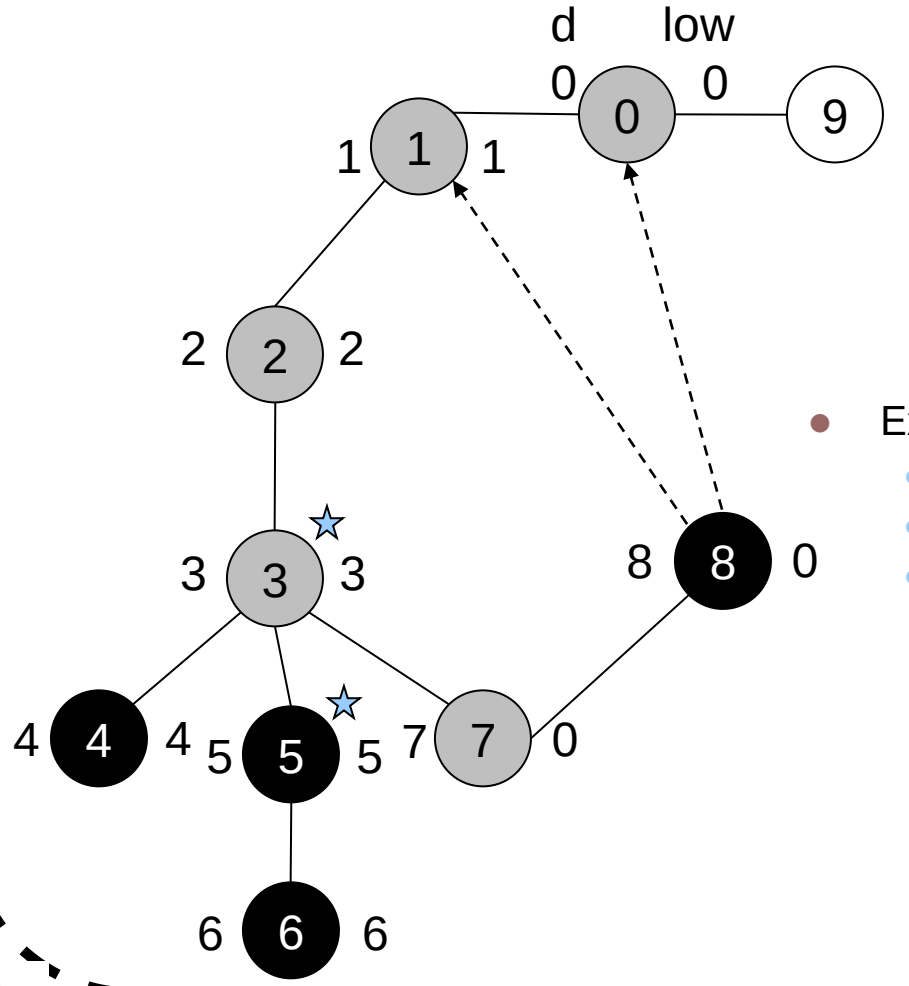
Exemplu rulare (15)



d[8] = 8
Low[8] = 0
Timp = 9
Cul[8] = gri
revenire

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

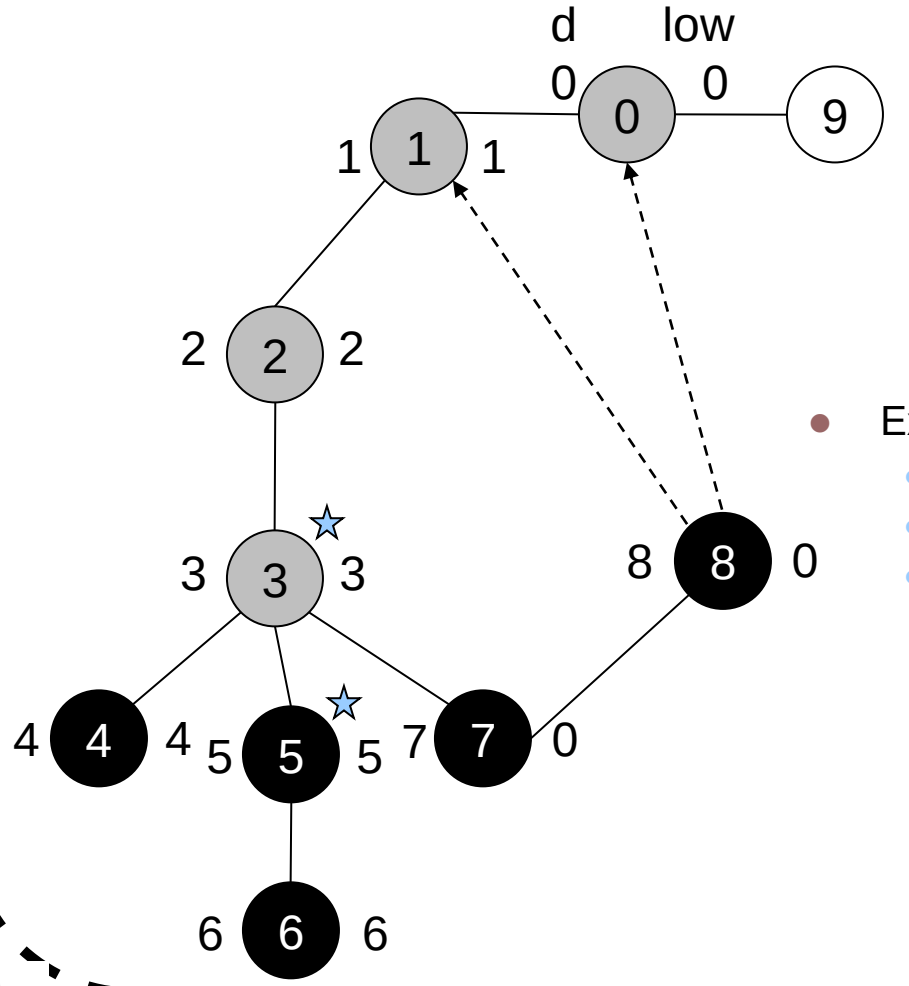
Exemplu rulare (16)



low[7] = min(low[7], low[8]) = 0
low[8] < d[7] → nu se modifică art[7]

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

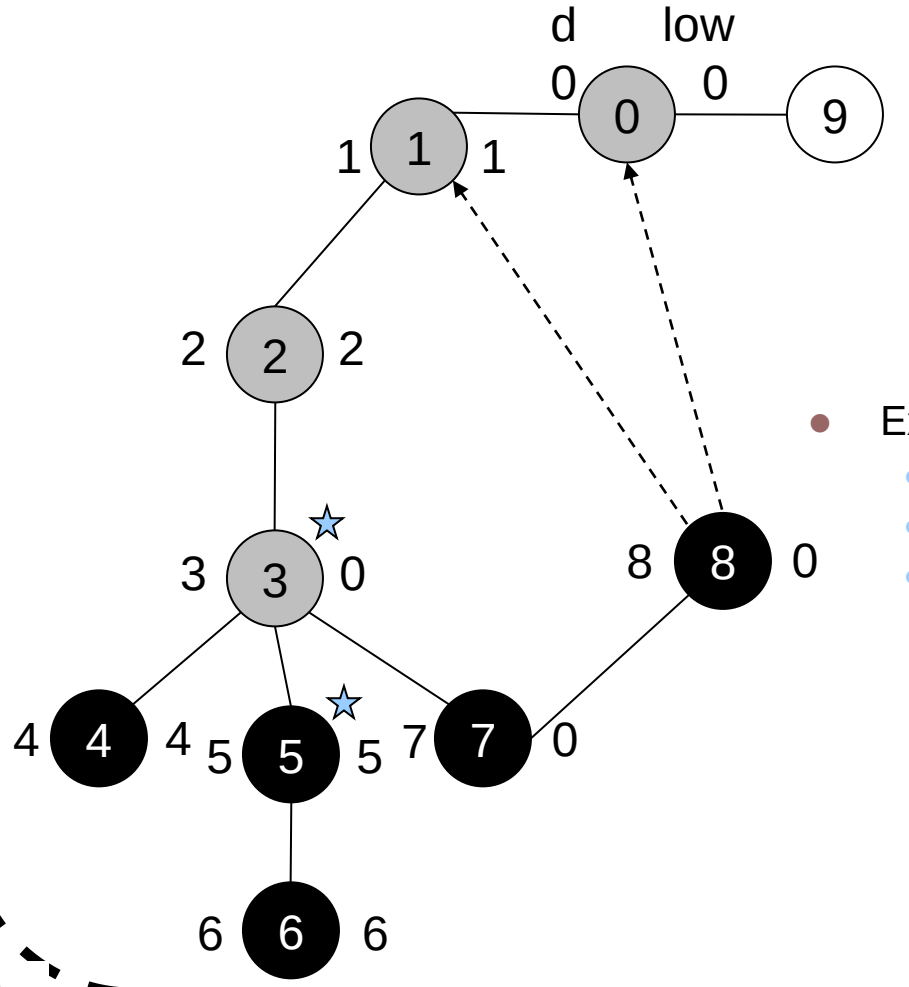
Exemplu rulare (17)



low[7] = min(low[7], low[8]) = 0
revenire

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

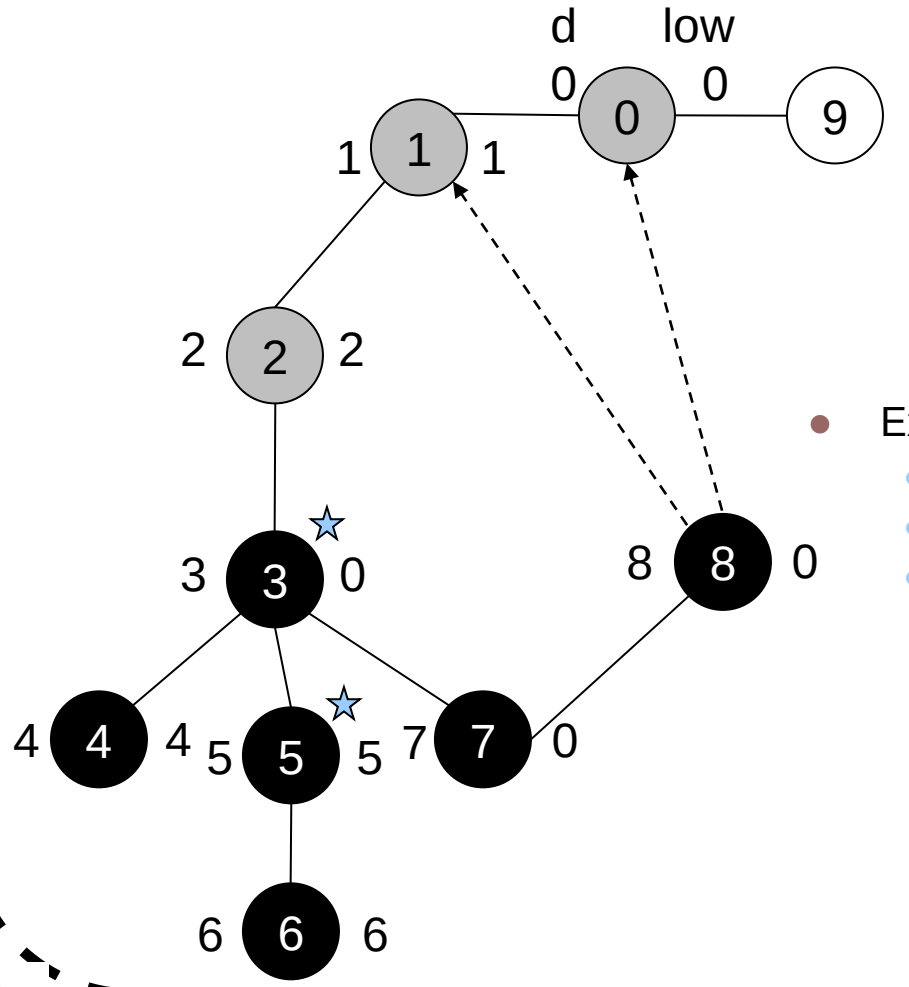
Exemplu rulare (18)



low[3] = min(low[3], low[7]) = 0
low[7] < d[3] → nu se modifică art[3]

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

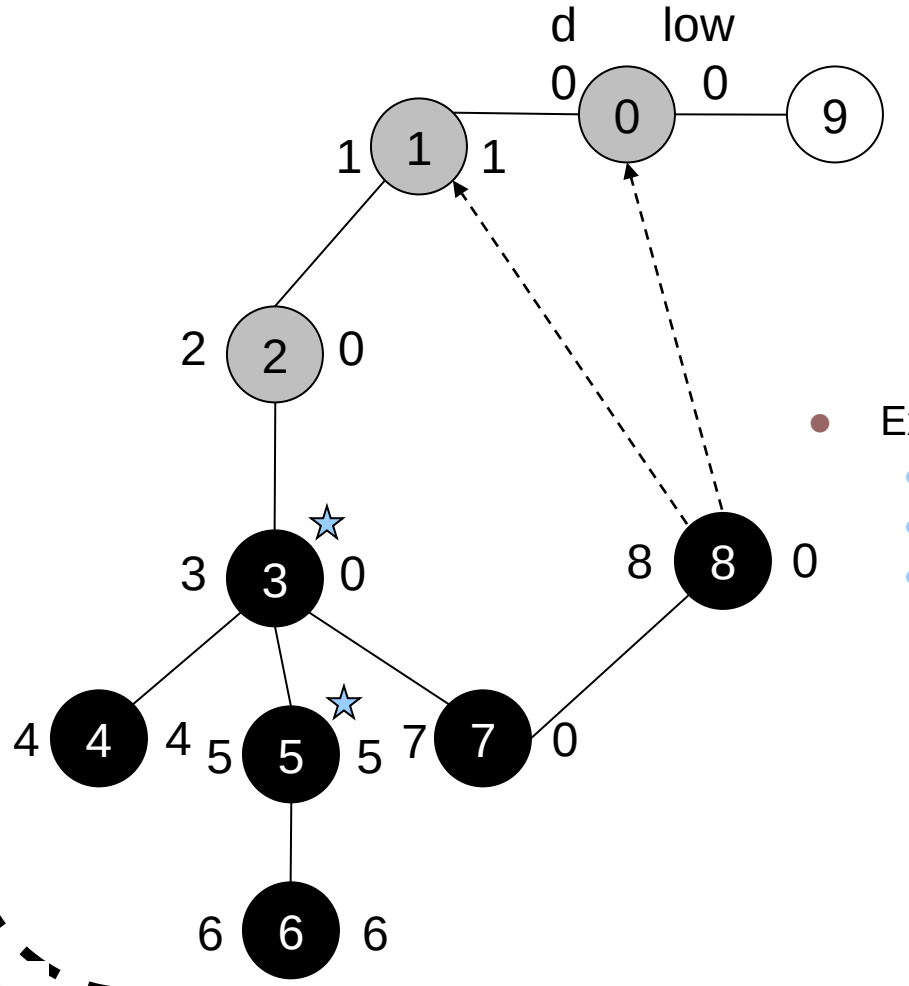
Exemplu rulare (19)



low[3] = min(low[3], low[7]) = 0
low[7] < d[3] → nu se modifică art[3]
revenire

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]}
        // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1;
        // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]}
      // actualizare low

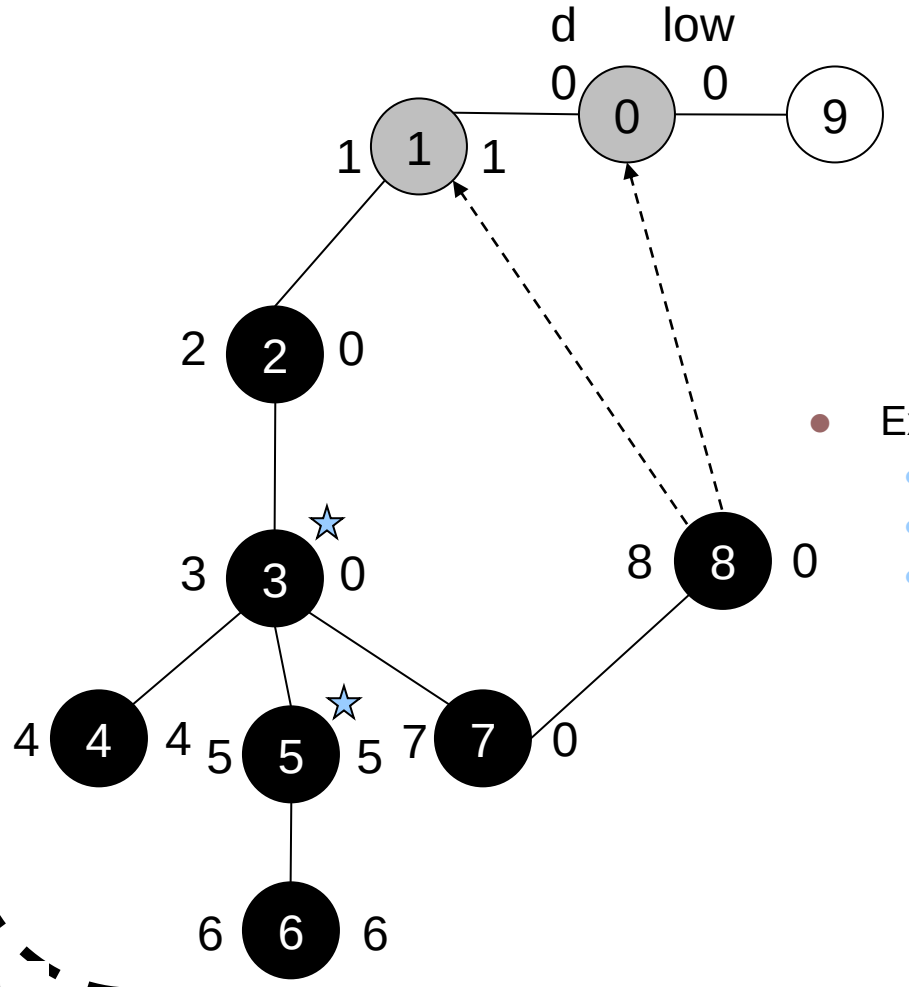
Exemplu rulare (20)



low[2] = min(low[2], low[3]) = 0
low[3] < d[2] → nu se modifică art[2]

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

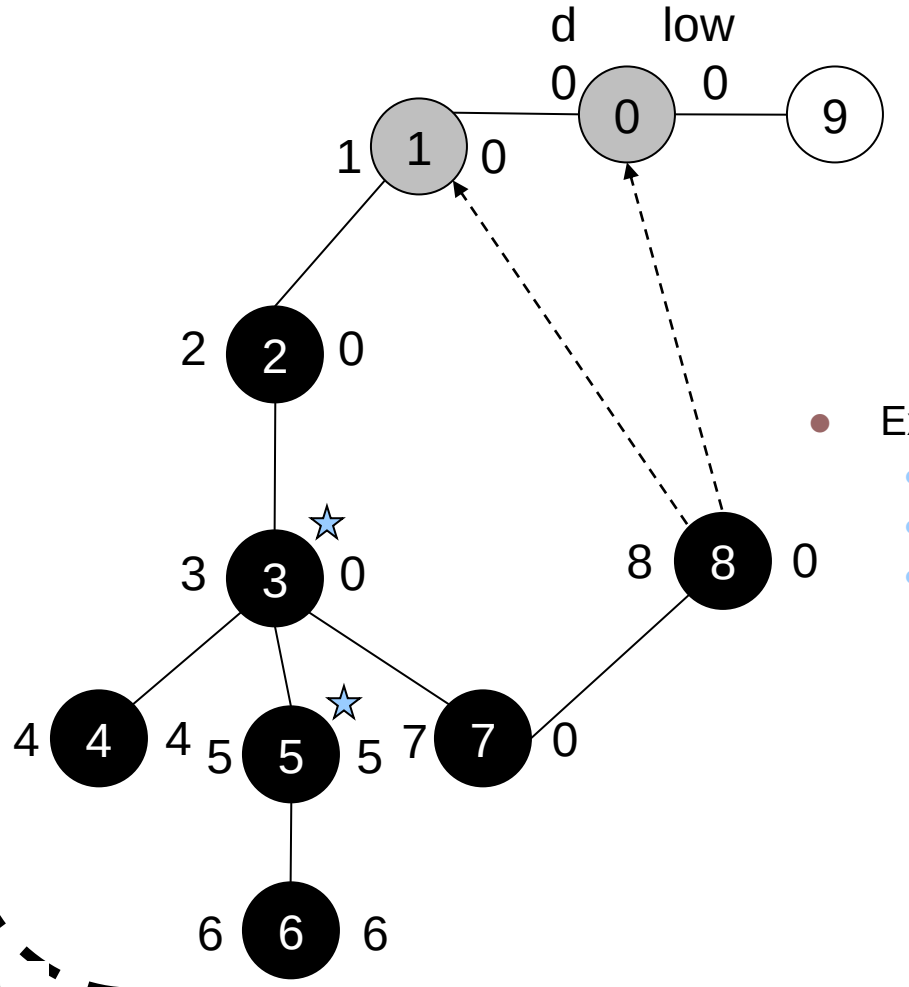
Exemplu rulare (21)



low[2] = min(low[2], low[3]) = 0
low[3] < d[2] → nu se modifică art[2]
revenire

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]}
        // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1;
        // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]}
      // actualizare low

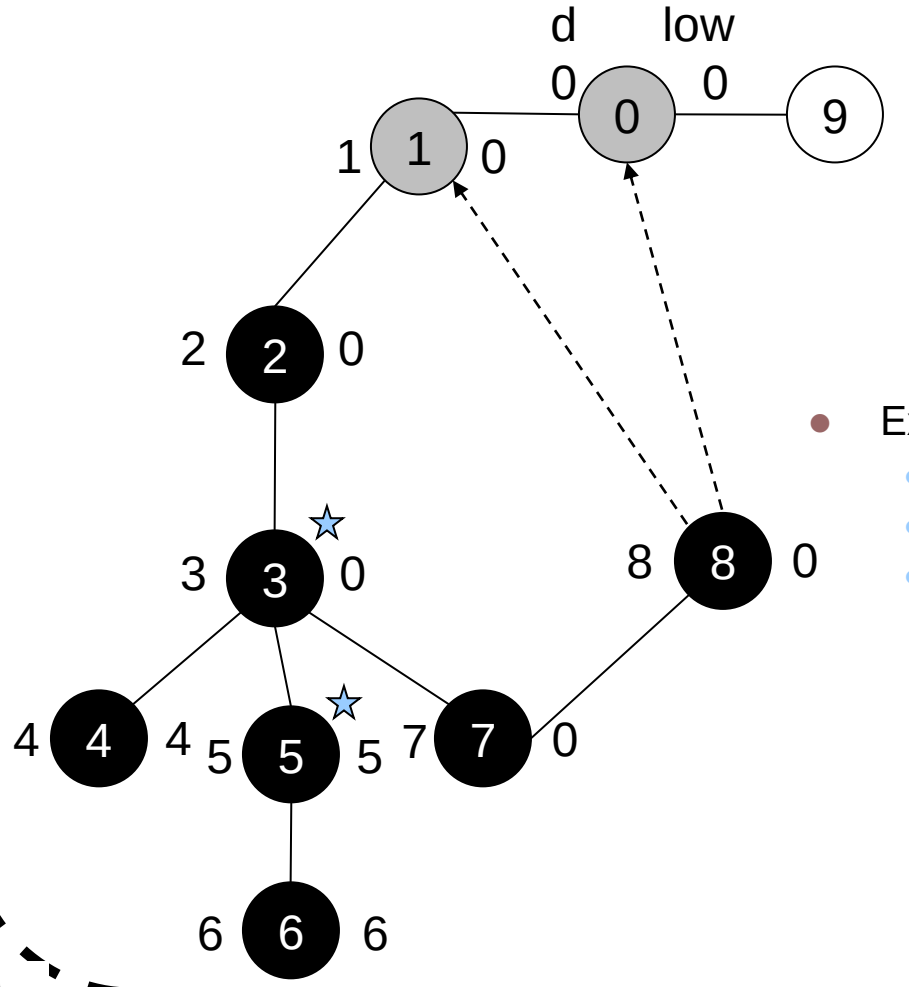
Exemplu rulare (22)



low[1] = min(low[1], low[2]) = 0
low[2] < d[1] → nu se modifică art[1]

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

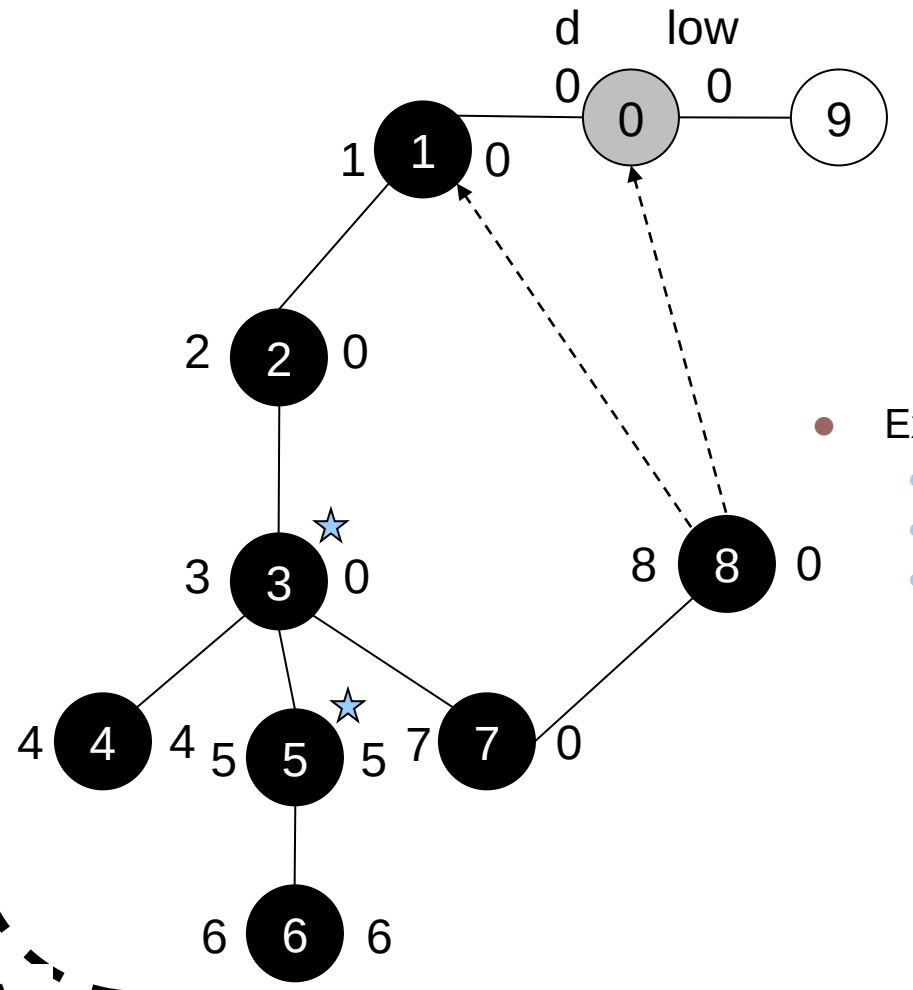
Exemplu rulare (23)



low[1] = 0
low[1] = min(low[1], d[8]) = 0

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

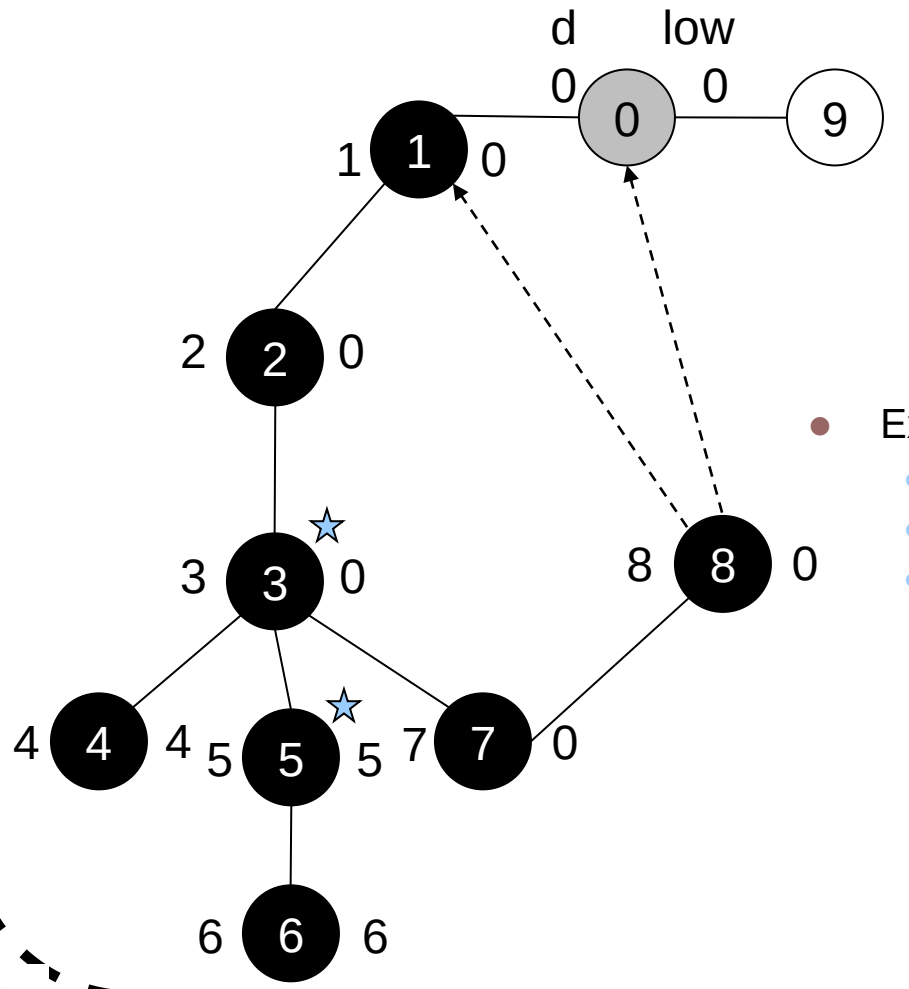
Exemplu rulare (24)



low[1] = 0
revenire

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

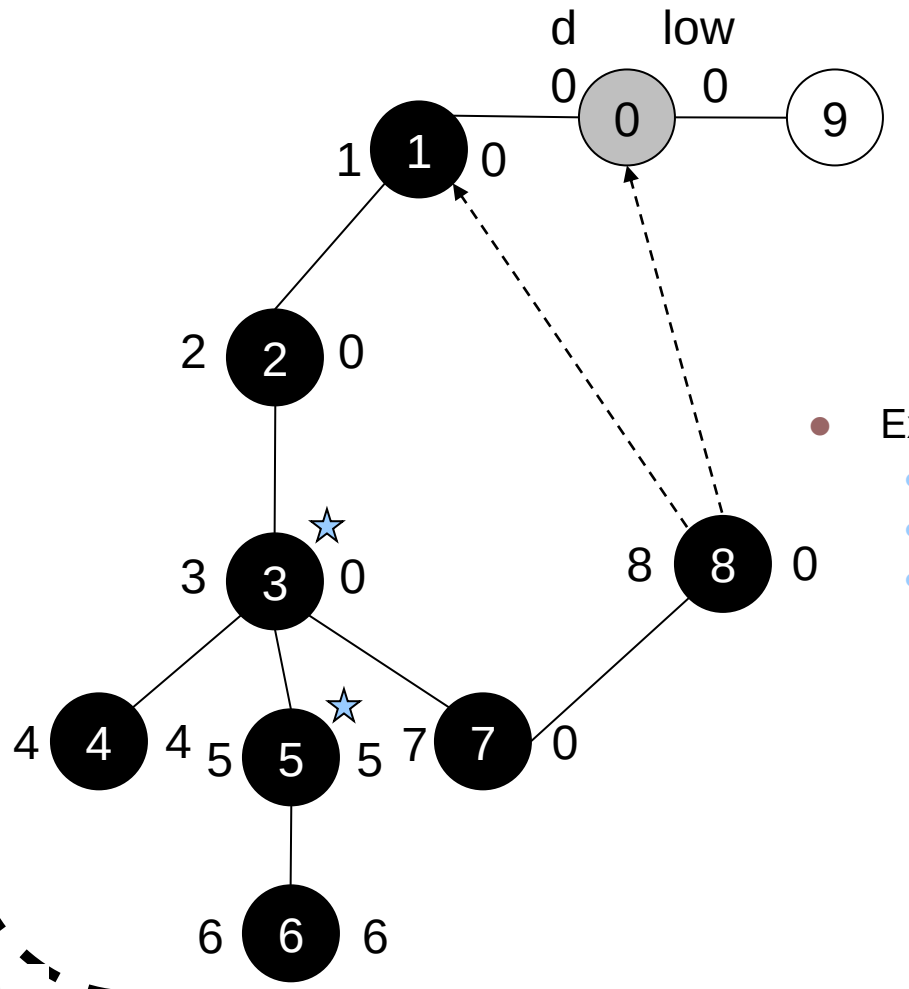
Exemplu rulare (25)



low[0] = min{low[1], low[0]} = 0
P[0] = null → continuă cu următorul copil

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]}
        // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]}
      // actualizare low

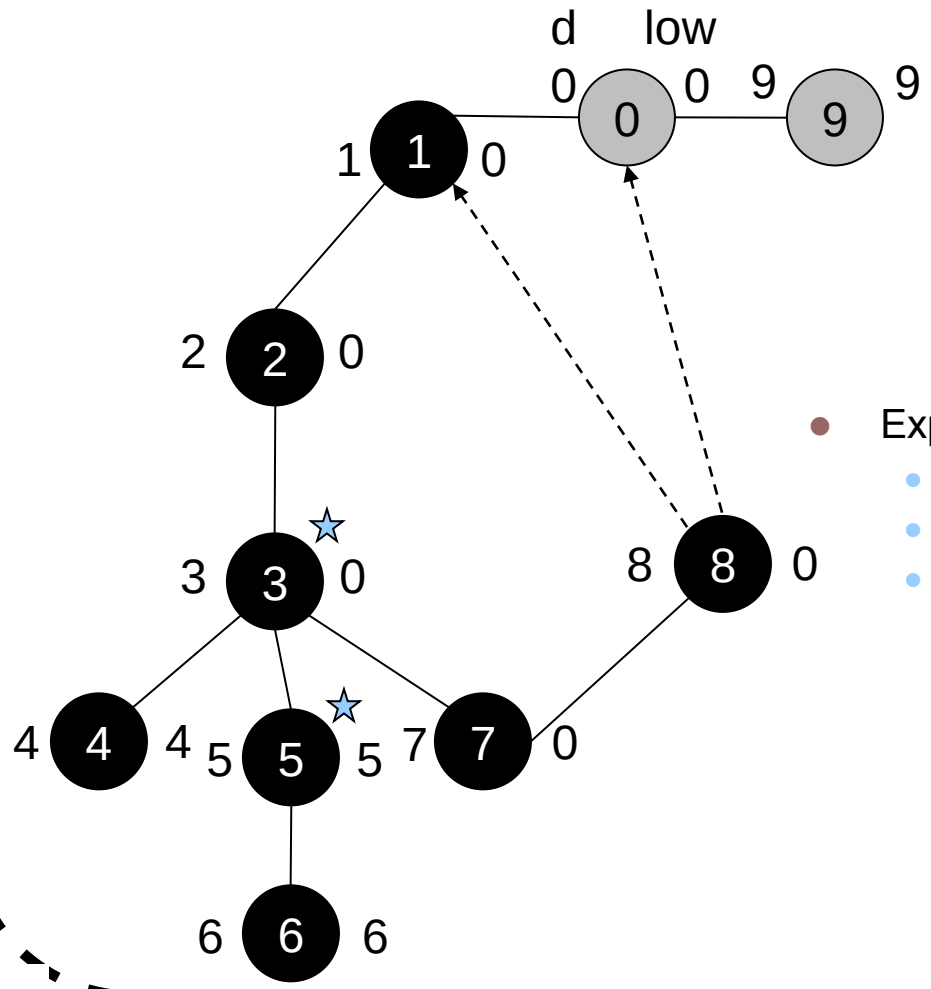
Exemplu rulare (26)



low[0] = min(low[0],d [8]) = 0
P[9] = 0
Subarb[0] = 2
Exploreaza (9)

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

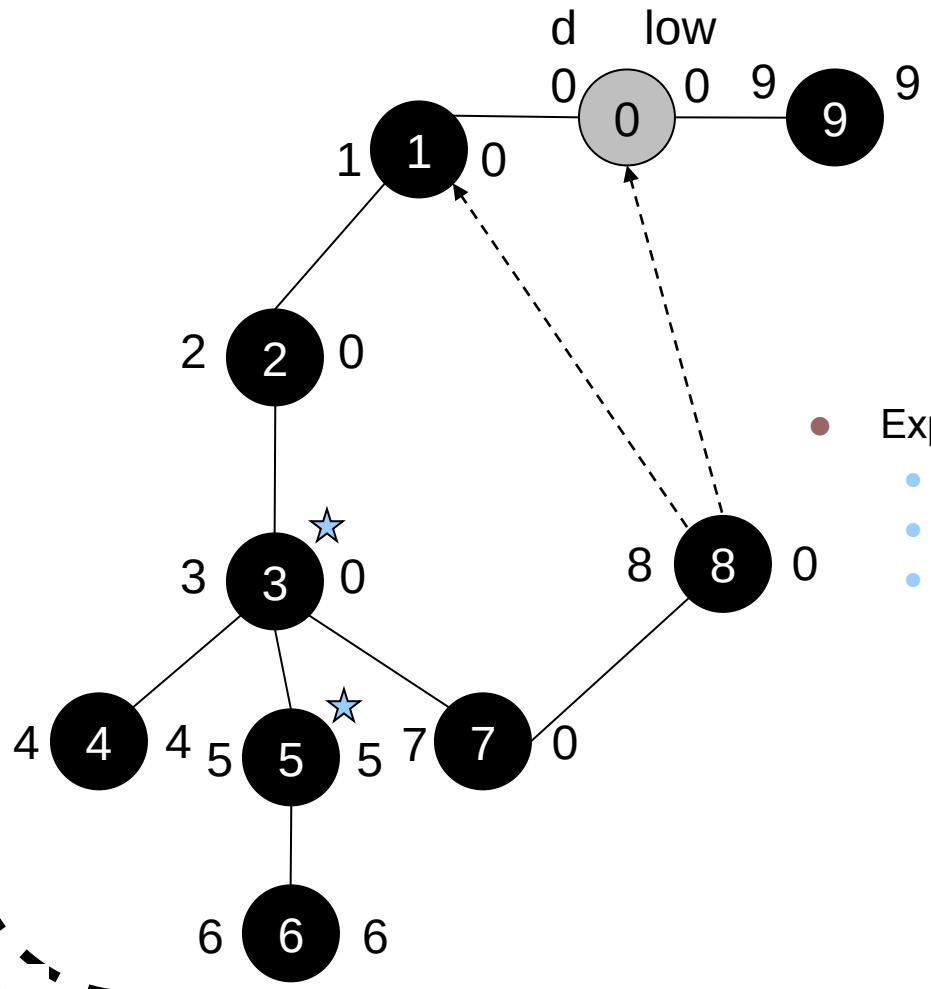
Exemplu rulare (27)



Low[9] = d[9] = 9
Timp = 10
Cul[9] = gri
revenire

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

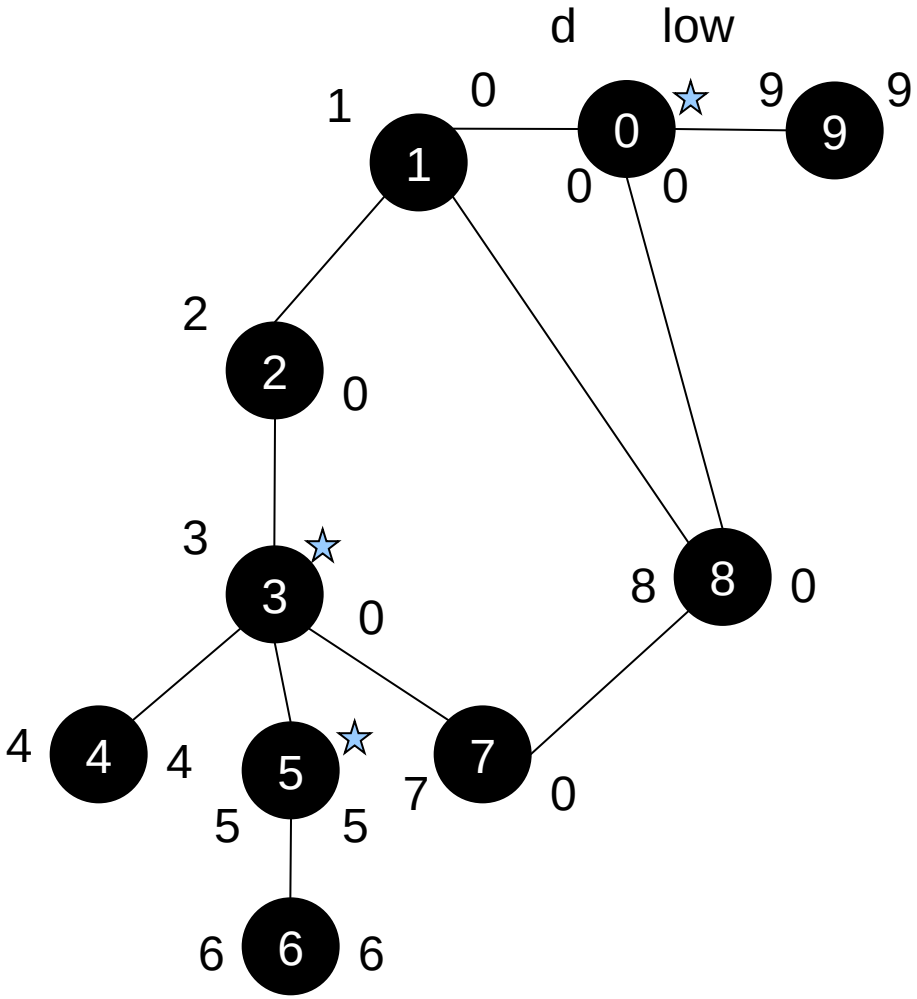
Exemplu rulare (28)



low[0] = min(low[0], low[9]) = 0
P[0] = null → revenire

- Explorează(u)
  - d[u] = low[u] = timp++; // inițializări
  - culoare[u] = gri;
  - **Pentru fiecare** (v succesor al lui u)
    - **Dacă** (culoare[v] e alb)
      - p[v] = u; subarb[u]++; // actualizare nr // subarbori dominați de u
      - Explorează(v);
      - low[u] = min{low[u], low[v]} // actualizare low
      - **Dacă** (p[u] != null && low[v] ≥ d[u]) art[u] = 1; // cazul 2 al teoremei
    - **Altfel** low[u] = min{low[u], d[v]} // actualizare low

Exemplu rulare (29)



Algoritmul lui Tarjan adaptat pentru determinarea CTC

- index = 0 // nivelul pe care este nodul în arborele DFS
- S = empty // se folosește o stivă care se inițializează cu Ø
- **Pentru fiecare** v din V
  - **Dacă** (v.index e nedefinit) **atunci** // se pornește DFS din fiecare nod pe care nu l-am vizitat încă
    - Tarjan(v)

- Tarjan(v)
  - v.index = index // se setează nivelul nodului v
  - v.lowlink = index // reține strămoșul nodului v
  - index = index + 1 // incrementez nivelul
  - S.push(v) // introduc v în stiva
  - **Pentru fiecare** (v, v') din E // se prelucrează succesorii lui v
    - **Dacă** (v'.index e nedefinit **sau** v' e in S) **atunci** // CTC deja identificate sunt ignorate
      - **Dacă** (v'.index e nedefinit) **atunci** tarjan(v') // dacă nu a fost vizitat v' intru în recursivitate
      - v.lowlink = min(v.lowlink, v'.lowlink) //actualizez strămoșul
  - **Dacă** (v.lowlink == v.index) **atunci** // printez CTC începând de la coadă spre rădăcină
    - print "CTC:"
    - **Repetă**
      - v' = S.pop // extrag nodul din stiva și îl printez
      - print v'
    - **Până când** (v' == v) // până când extrag rădăcina

Exemplu rulare (CTC)