

Proiectarea Algoritmilor

Curs 8 – Puncte de articulație,
Punți, Drumuri minime

Bibliografie

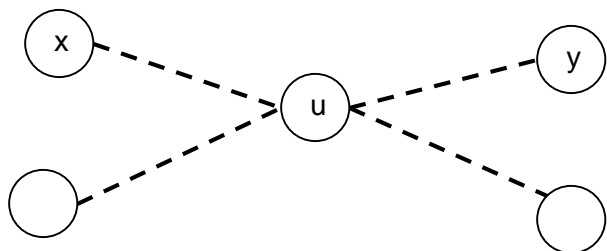
- [1] Giumale – Introducere în Analiza Algoritmilor cap. 5.3, 5.4, 5.4.1
- [2] Cormen – Introducere în Algoritmi cap. Heap-uri binomiale (20), Heap-uri Fibonacci (21), Drumuri minime de sursă unică - primele 2 subcapitole (25.1 și 25.2)
- [3] R. Sedgwick, K. Wayne - Algorithms and Data Structures Fall 2007 – Curs Princeton -
<http://www.cs.princeton.edu/~rs/AlgsDS07/06PriorityQueues.pdf>
- [4] Heap Fibonacci:
<http://www.cse.yorku.ca/~aaw/Jason/FibonacciHeapAnimation.html>
- [5] Fibonacci heap:
<http://www.growingwiththeweb.com/data-structures/fibonacci-heap/overview/>

Objective

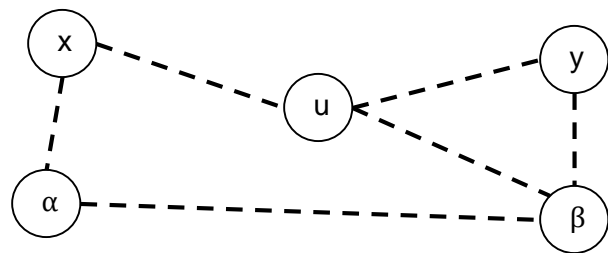
- “Descoperirea” algoritmilor de:
 - Identificare a punctelor de articulație;
 - Identificare a punților;
 - Identificare a drumurilor de cost minim.
- Identificarea structurilor de date necesare pentru reducerea complexității acestor algoritmi.

Puncte de articulație. Def. Exemple

- Definiție: $G = (V, E)$ graf neorientat, $u \in V$.
 u este punct de articulație dacă $\exists x, y \in V$,
 $x \neq y$, $x \neq u$, $y \neq u$, a.î. $\forall x..y$ în G trece prin u .



Orice drum $x..y$ trece prin $u \rightarrow$
 u este punct de articulație.



Exista $x..\alpha..y$ care nu trece prin
 $u \rightarrow u$ nu mai este punct de
articulație!

Algoritm naiv de detectare a punctelor de articulație

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

Complexitate?

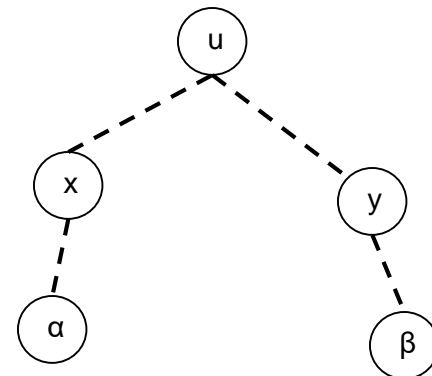
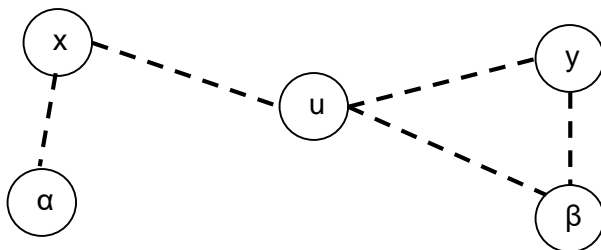
$O(V(V+E))$

Puncte de articulație. Teoremă

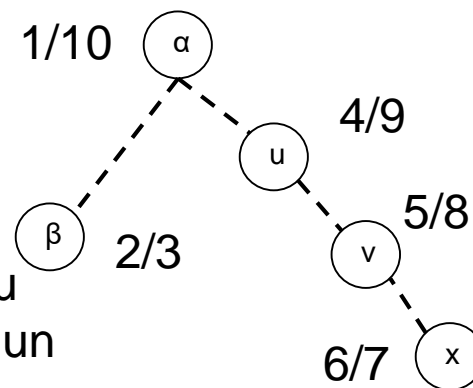
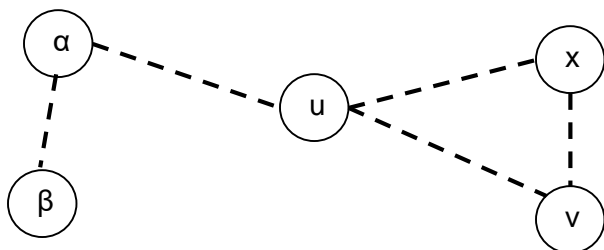
- **Teorema 5.15:** $G = (V, E)$, graf **neorientat** și $u \in V$. u este **punct de articulație** în G \Leftrightarrow în urma DFS în G una din proprietățile de mai jos este satisfăcută:
 - $p(u) = \text{null}$ și u domină cel puțin 2 subarbori;
 - $p(u) \neq \text{null}$ și $\exists v$ descendent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

Situații posibile

- 1) $p(u) = \text{null}$ și u domina cel puțin 2 subarbori:



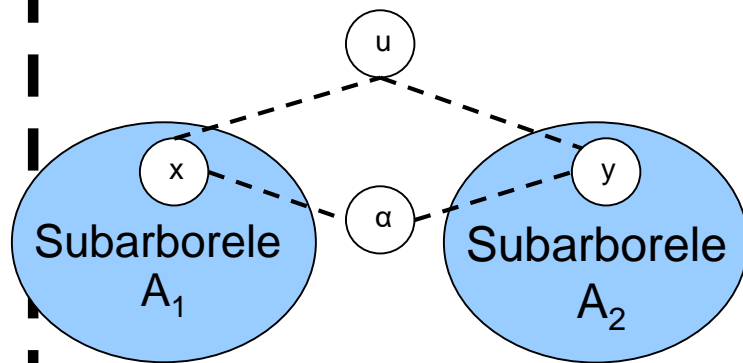
- 2) $p(u) \neq \text{null}$ și $\exists v$ descendent al lui u în $\text{Arb}(u)$ a.î.
 $\forall x \in \text{Arb}(v)$ și $\forall (x, z)$ parcursă de $\text{DFS}(G)$ $d(z) \geq d(u)$:



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

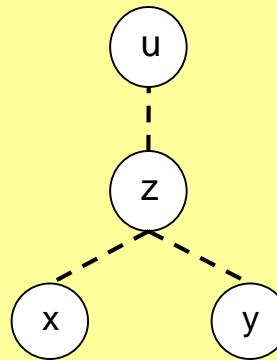
Puncte de articulație. Demonstrație teoremă (la)

- $p(u) = \text{null}$ și u domină cel puțin 2 subarbori $\Rightarrow u$ 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 $\exists x.. \alpha.. y$ și $u \notin x.. \alpha.. y$.
- $z =$ primul nod descoperit de DFS din care se poate ajunge la x și la y . Cf. **T drumurilor albe** $x, y \in \text{Arb}(z)$.
- Dar $x, y \in \text{Arb}(u) \rightarrow$ 2 cazuri:



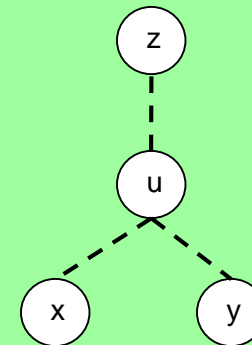
Pp $\exists x.. \alpha.. y$ și $u \notin x.. \alpha.. y$.

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



Contradicție (1) x, y nu sunt în subarbori diferiți ai lui $\text{Arb}(u)$.

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



Contradicție (1), $p(u) \neq \text{null}$.

Puncte de articulație. Demonstrație teoremă (Ib)

- u este punct de articulație și este descoperit în ciclul principal al DFS $\Rightarrow p(u) = \text{null}$ și u domină cel puțin 2 subarbori.
- **Dem (Reducere la absurd):** Fie nodurile x și y a.î. $u \in \forall x..y$. $u =$ primul nod descoperit din cale (altfel u nu mai e descoperit în ciclul principal al DFS) $\Rightarrow p(u) = \text{null}$ și $x, y \in \text{Arb}(u)$.

- pp că x, y aparțin la același subarbor
fie z rădăcina
 $x..z..y \rightarrow$
se contradicție

DFS(G)

$V = \text{noduri}(G)$

Pentru fiecare nod u ($u \in V$)

$c(u) = \text{alb}; p(u) = \text{null};$ // inițializare structură date

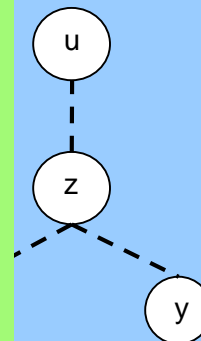
$\text{timp} = 0;$ // reține distanța de la rădăcina arborelui DFS până la nodul curent

Pentru fiecare nod u ($u \in V$)

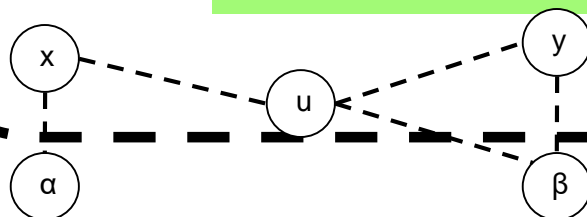
Dacă $c(u)$ este alb

Atunci $\text{explorare}(u);$ // explorez nodul

celăși subarbori

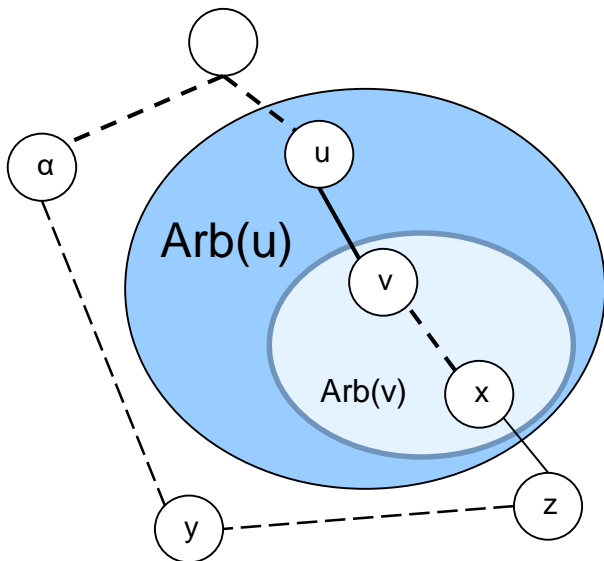


Contradicție $\exists x..z..y \Rightarrow$
 u nu este punct de articulație



Puncte de articulație. Demonstrație teoremă (IIa)

- $p(u) \neq \text{null}$ și $\exists v$ descendent al lui u în $\text{Arb}(u)$ a.î. $\forall x \in \text{Arb}(v)$ și $\forall (x,z)$ parcursă de $\text{DFS}(G)$ are $d(z) \geq d(u) \Rightarrow u$ este punct de articulație.



Dem (Reducere la absurd): Pp. u nu e punct de articulație $\rightarrow \exists w \in \text{Arb}(v)$, $y \notin \text{Arb}(u)$ a.î. $y..w$. Fie z primul nod din $y..w$ a.î. $z \notin \text{Arb}(u)$ și x ultimul nod din $w..y$ a.î. $x \in \text{Arb}(u) \rightarrow (x,z)$ taie frontiera $\text{Arb}(u)$.

Dacă $d(z) > d(u) \rightarrow u..x, z$ alb la $d(u) \rightarrow z \in \text{Arb}(u) \rightarrow$ contradicție ($z \notin \text{Arb}(u)$)

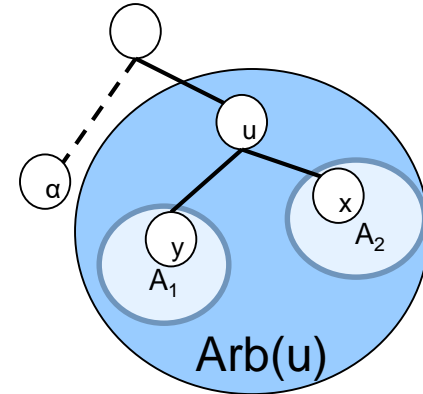
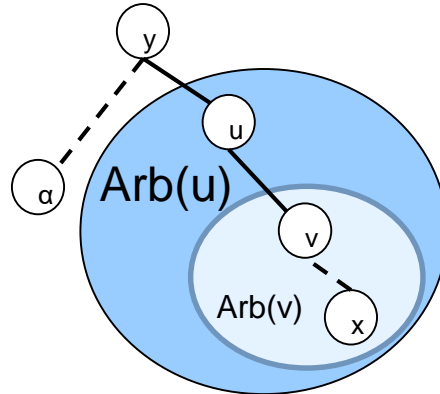
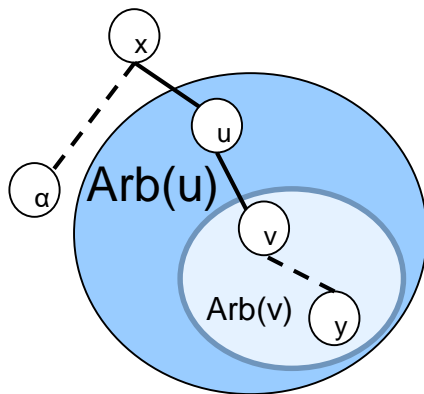
Dacă $d(z) < d(u) \rightarrow$ contradicție (ipoteza)

$\rightarrow \nexists y..w \rightarrow u$ punct de articulație

Puncte de articulație. Demonstrație teoremă (IIb)

- u este punct de articulație și nu este descoperit în ciclul principal al DFS $\Rightarrow p(u) \neq \text{null}$ și $\exists v$ descendent al lui u în $\text{Arb}(u)$ a.î. $\forall x \in \text{Arb}(v)$ și $\forall (x,z)$ parcursă de DFS(G) având $d(z) \geq d(u)$.

- **Dem:** Fie nodurile x și y a.î. $u \in \forall x..y$ și $p(u) \neq \text{null}$. 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 $\text{Arb}(v)$ către vreun predecesor al lui u . Altfel $\exists x..y$ a.î. $u \notin x..y$.
- 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 subarbor A_1 sau A_2 .

Puncte de articulație. Structuri de date.

- Structura de date de la DFS + pentru fiecare nod $u \in V$ se rețin:
 - $\text{Low}(u) = \min\{d(v) \mid v \text{ descoperit pornind din } u \text{ în cursul DFS și } c(v) \neq \text{alb}\}$
 - $\text{Subarb}(u) = \text{numărul subarborilor dominați de } u \text{ (dacă } e \geq 2, \text{ atunci avem un punct de articulație).}$

Idee algoritm

- Se aplică DFS și se salvează pentru fiecare nod până unde merge înapoi (low):
 $\text{low}[u] = \min \{d(u), d(v) \text{ pentru toate muchiile } (u,v), \text{low}(w) \text{ pentru toți fiii } w \text{ ai lui } u\}.$
- Pentru **eficiență**, trebuie ca fiii să se parcurgă înaintea părinților → ordinea inversă a $d(u)$.

Algoritm Tarjan (I)

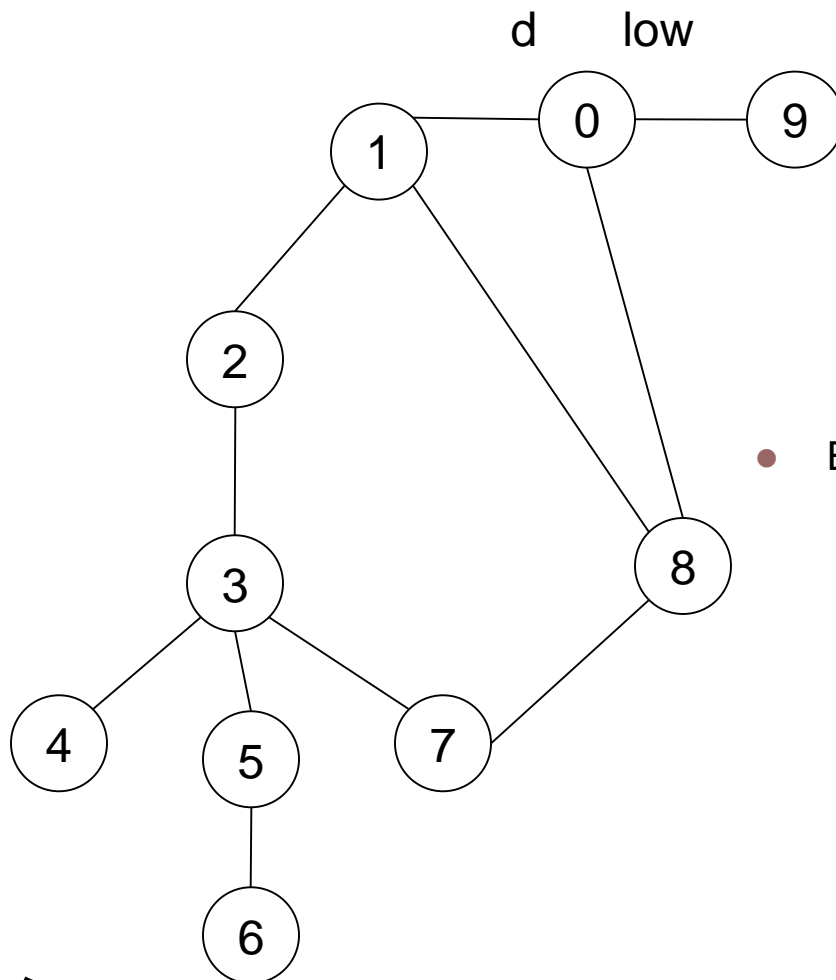
- Articulații (G)

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

Algoritm Tarjan (II)

- Explorează(u)
 - $d(u) = \text{low}(u) = \text{timp}++;$ // inițializări
 - $c(u) = \text{gri};$
 - **Pentru fiecare** nod $v \in \text{succs}(u)$
 - **Dacă** ($c(v)$ e alb)
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr subarbori
// dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - **Dacă** ($p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u)$) $\text{art}(u) = 1;$
// cazul 2 al teoremei
 - **Altfel** $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

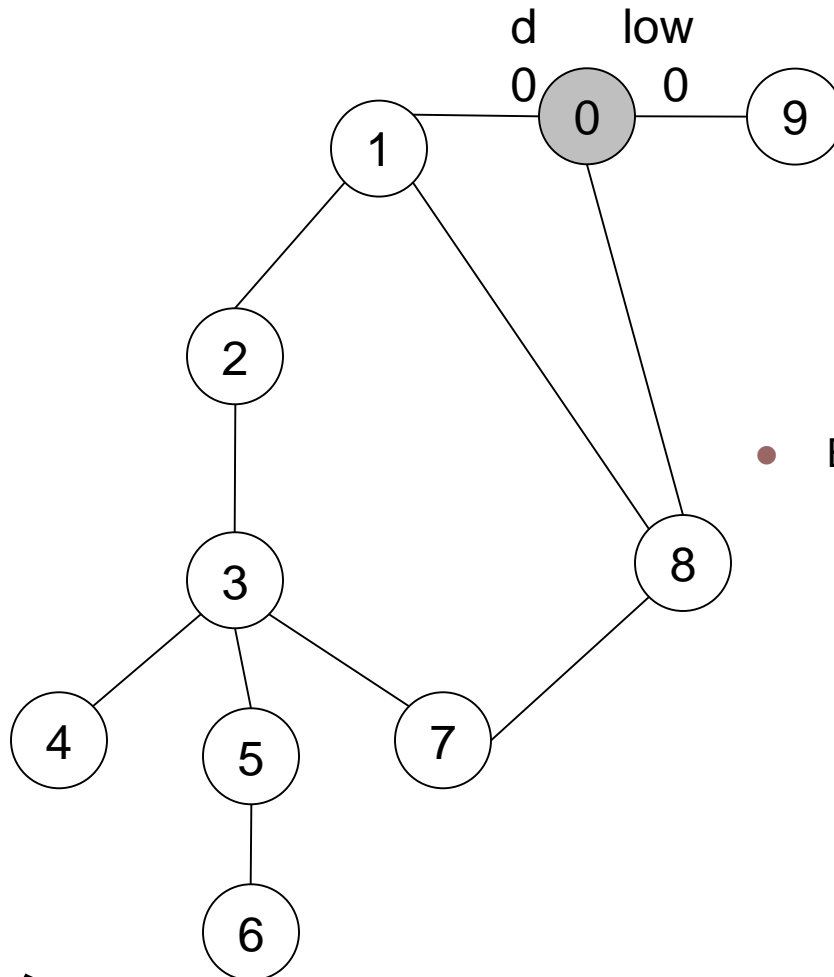
Exemplu rulare (1)



Timp = 0
C(i) = alb
D(i) = 0
Low(i) = 0
P(i) = null
Subarb(i) = 0
Art(i) = 0
Exploreaza (0)

- Explorează(u)
 - $d(u) = \text{low}(u) = \text{timp}++$; // inițializări
 - $c(u) = \text{gri}$;
 - Pentru fiecare nod $v \in \text{succs}(u)$
 - Dacă $(c(v) \text{ e alb})$
 - $p(v) = u$; $\text{subarb}(u)++$; // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - Dacă $(p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u))$ $\text{art}(u) = 1$;
// cazul 2 al teoremei
 - Altfel $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (2)



$\text{Low}(0) = d(0) = 0$

$\text{Timp} = 1$

$C(0) = \text{gri}$

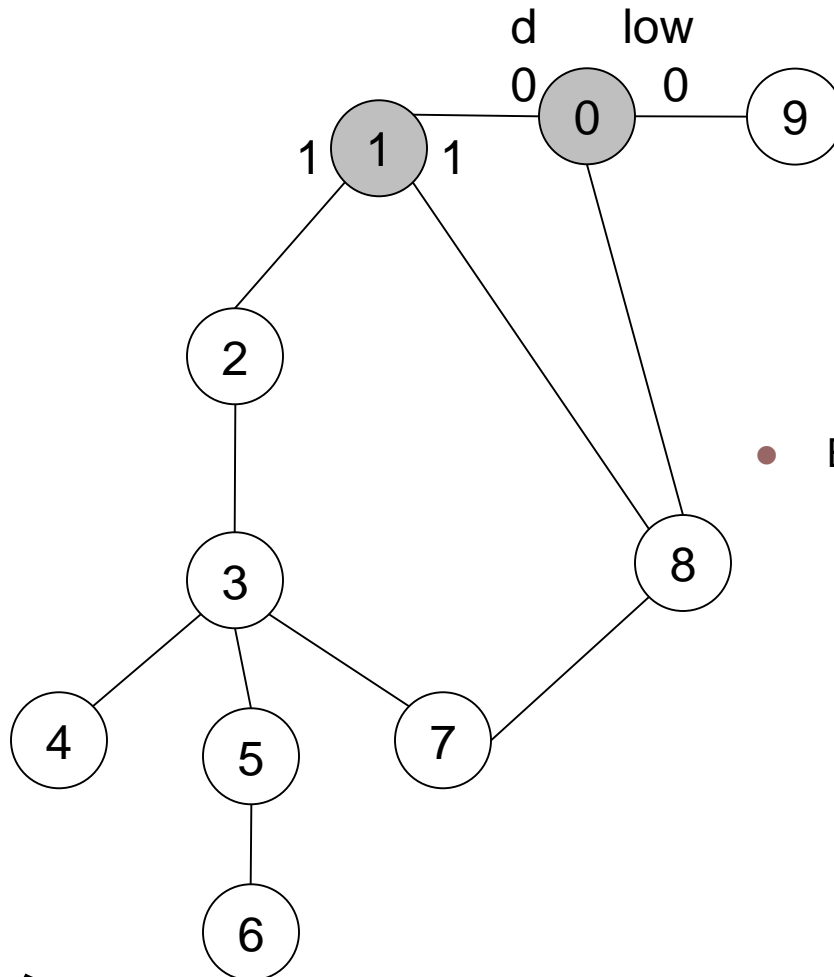
$P(1) = 0$

$\text{Subarb}(0) = 1$

Exploreaza (1)

- Explorează(u)
 - $d(u) = \text{low}(u) = \text{timp}++;$ // inițializări
 - $c(u) = \text{gri};$
 - Pentru fiecare nod $v \in \text{succs}(u)$
 - Dacă $(c(v) \text{ e alb})$
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - Dacă $(p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u))$ $\text{art}(u) = 1;$
// cazul 2 al teoremei
 - Altfel $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (3)



$\text{Low}(1) = d(1) = 1$

$\text{Timp} = 2$

$C(1) = \text{gri}$

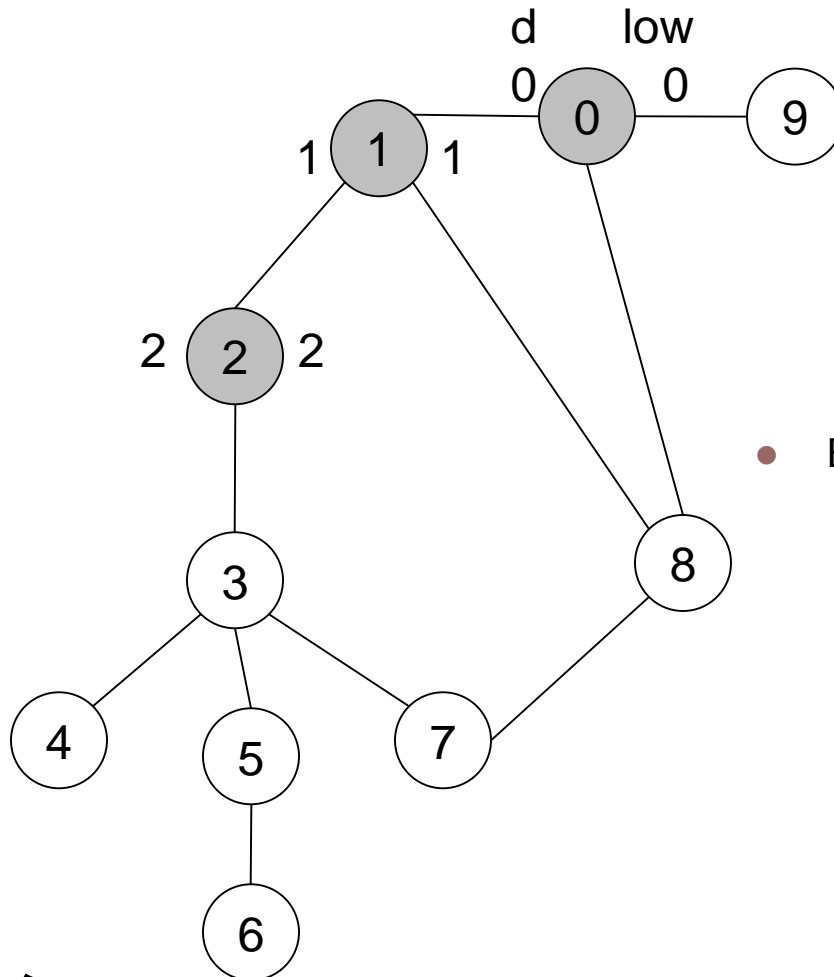
$P(2) = 1$

$\text{Subarb}(1) = 1$

Exploreaza (2)

- Explorează(u)
 - $d(u) = \text{low}(u) = \text{timp}++;$ // inițializări
 - $c(u) = \text{gri};$
 - **Pentru fiecare** nod $v \in \text{succs}(u)$
 - **Dacă** ($c(v)$ e alb)
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - **Dacă** ($p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u)$) $\text{art}(u) = 1;$
// cazul 2 al teoremei
 - **Altfel** $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (4)



$\text{Low}(2) = d(2) = 2$

$\text{Timp} = 3$

$C(2) = \text{gri}$

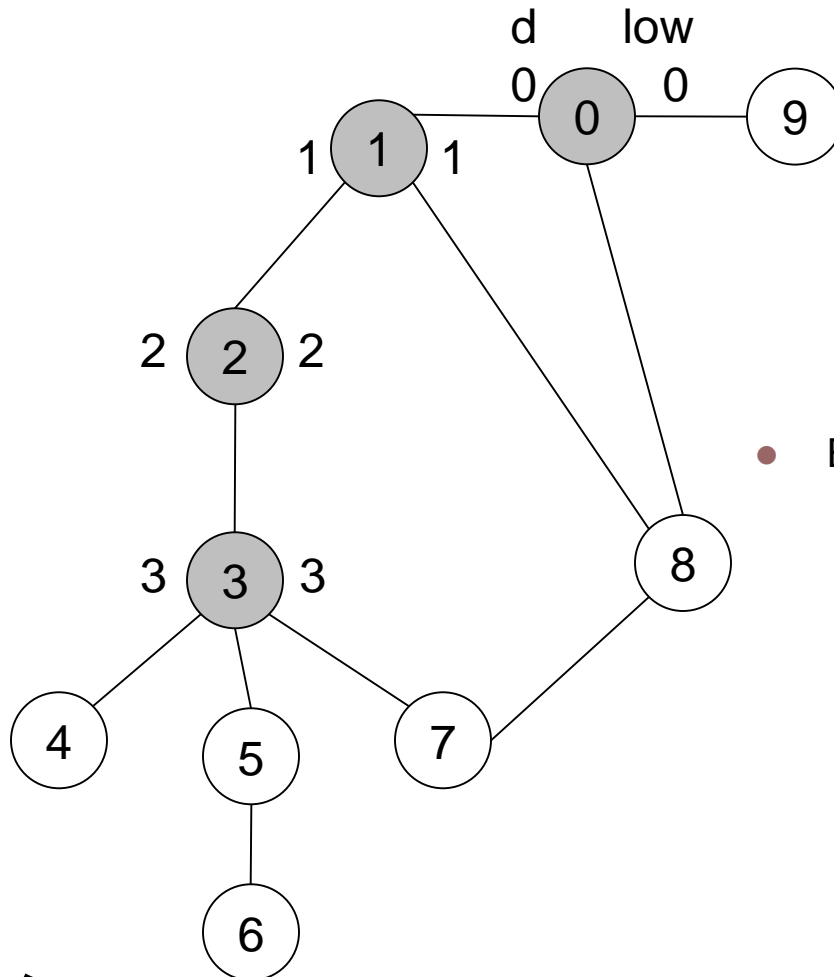
$P(3) = 2$

$\text{Subarb}(2) = 1$

Exploreaza (3)

- Explorează(u)
 - $d(u) = \text{low}(u) = \text{timp}++;$ // inițializări
 - $c(u) = \text{gri};$
 - Pentru fiecare nod $v \in \text{succs}(u)$
 - Dacă $(c(v) \text{ e alb})$
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - Dacă $(p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u))$ $\text{art}(u) = 1;$
// cazul 2 al teoremei
 - Altfel $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (5)



$\text{Low}(3) = d(3) = 3$

$\text{Timp} = 4$

$C(3) = \text{gri}$

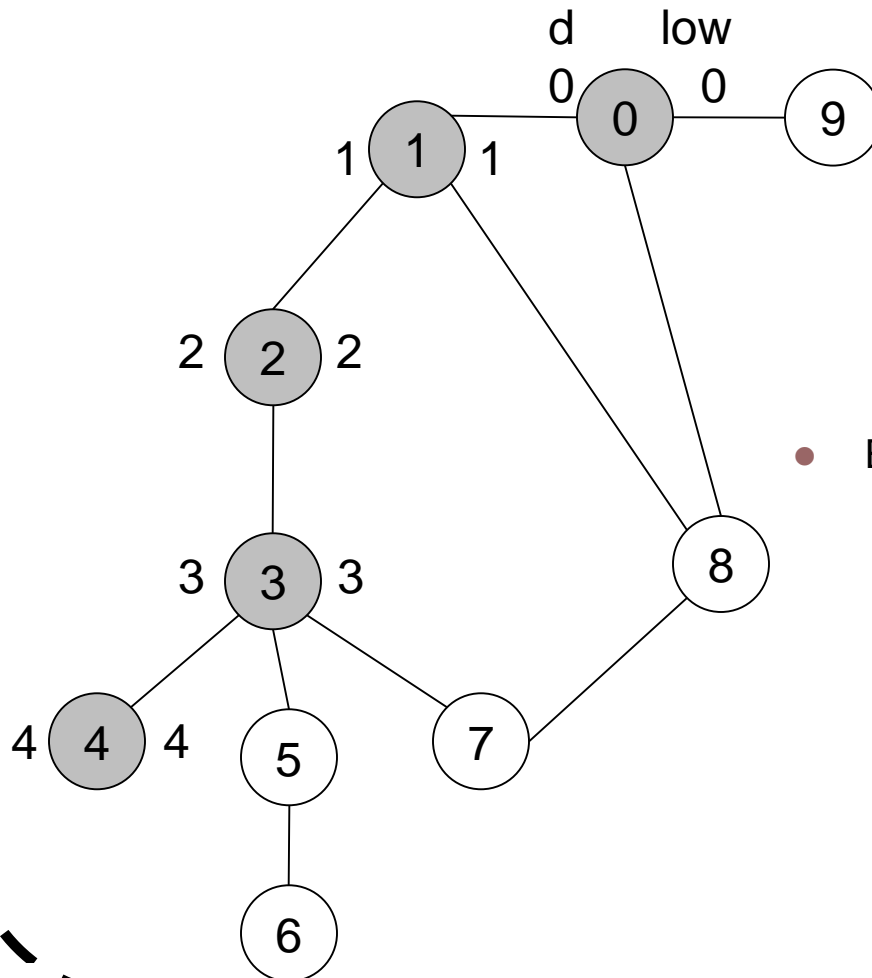
$P(4) = 3$

$\text{Subarb}(3) = 1$

Exploreaza (4)

- Explorează(u)
 - $d(u) = \text{low}(u) = \text{timp}++$; // inițializări
 - $c(u) = \text{gri}$;
 - **Pentru fiecare** nod $v \in \text{succs}(u)$
 - **Dacă** ($c(v)$ e alb)
 - $p(v) = u$; $\text{subarb}(u)++$; // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - **Dacă** ($p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u)$) $\text{art}(u) = 1$;
// cazul 2 al teoremei
 - **Altfel** $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

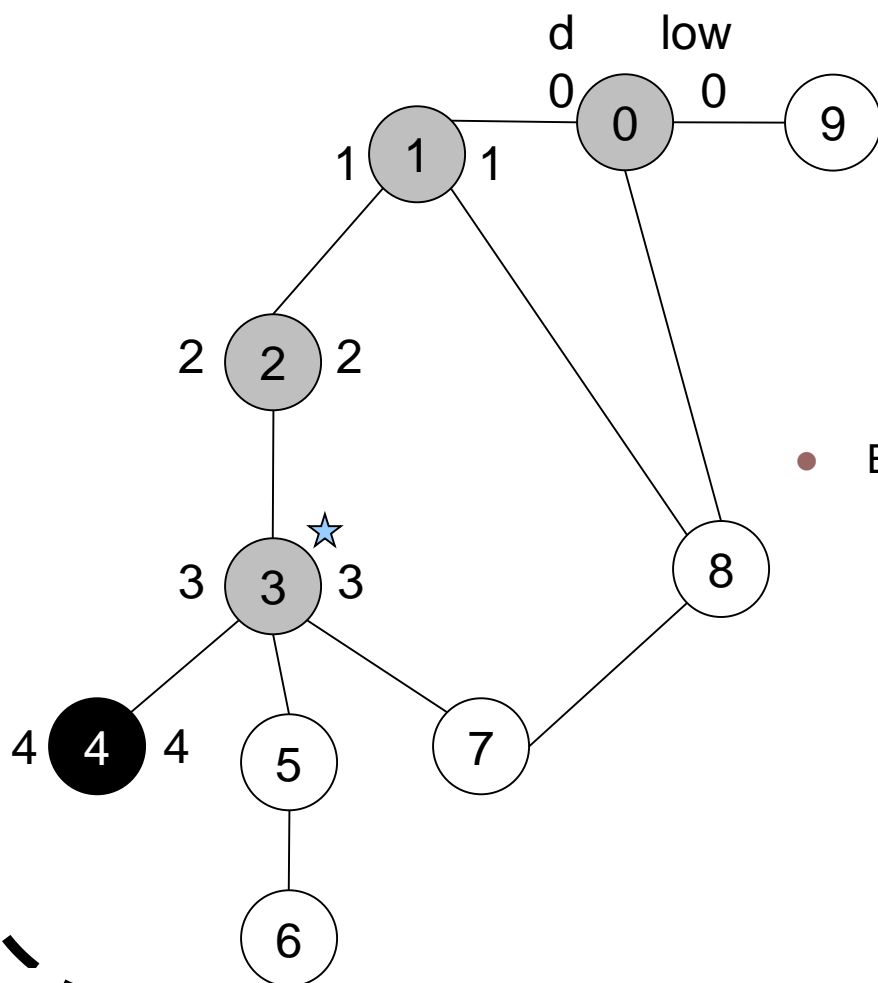
Exemplu rulare (6)



$\text{Low}(4) = d(4) = 4$
 $\text{Timp} = 5$
 $C(4) = \text{gri}$
 revenire

- Explorează(u)
 - $d(u) = \text{low}(u) = \text{timp}++$; // inițializări
 - $c(u) = \text{gri}$;
 - Pentru fiecare nod $v \in \text{succs}(u)$
 - Dacă ($c(v)$ e alb)
 - $p(v) = u$; $\text{subarb}(u)++$; // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - Dacă ($p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u)$) $\text{art}(u) = 1$;
// cazul 2 al teoremei
 - Altfel $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (7)



$\text{Low}(4) = d(4) = 4$

$\text{Timp} = 5$

$C(4) = \text{gri}$

revenire

$\text{Low}(3) = \min\{\text{low}(3), \text{low}(4)\} = 3$

$\text{Low}(4) > d(3) \rightarrow \text{art}(3) = 1$

$P(5) = 3$

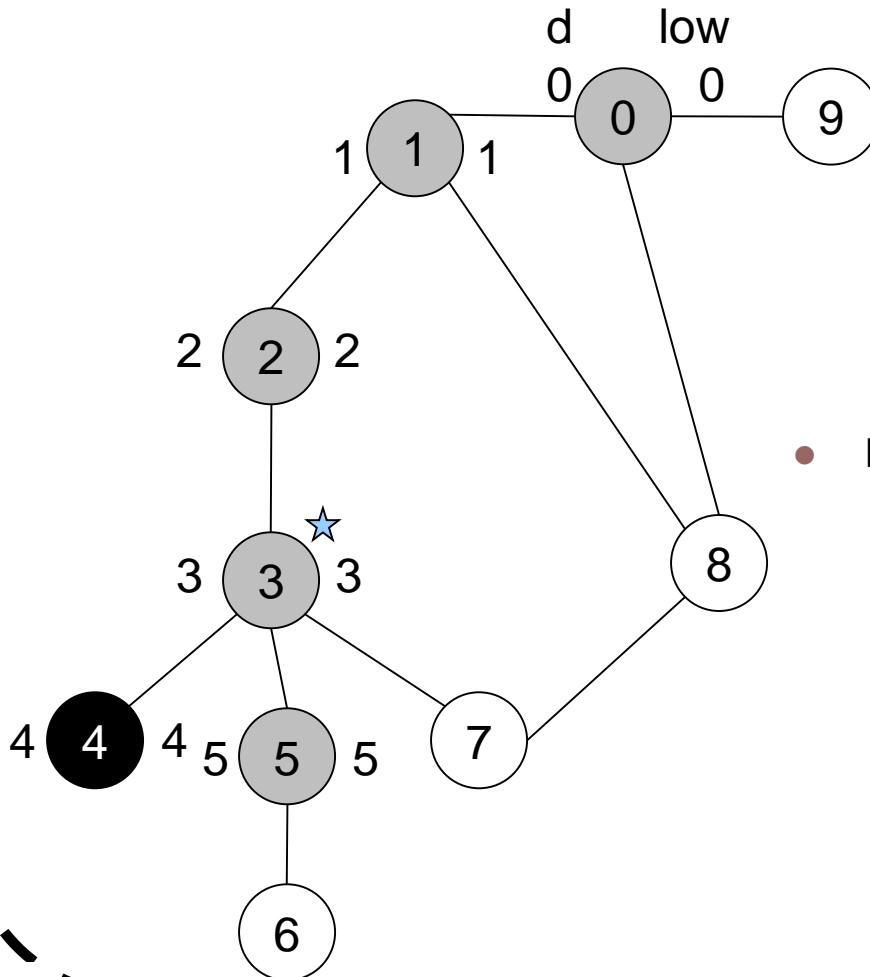
$\text{Subarb}(3) = 2$

Exploreaza (5)

● Explorează(u)

- $d(u) = \text{low}(u) = \text{timp}++;$ // inițializări
- $c(u) = \text{gri};$
- **Pentru fiecare** nod $v \in \text{succs}(u)$
 - **Dacă** ($c(v)$ e alb)
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - **Dacă** ($p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u)$) $\text{art}(u) = 1;$
// cazul 2 al teoremei
 - **Altfel** $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (8)


$$\text{Low}(5) = d(5) = 5$$

$T_{imp} = 6$

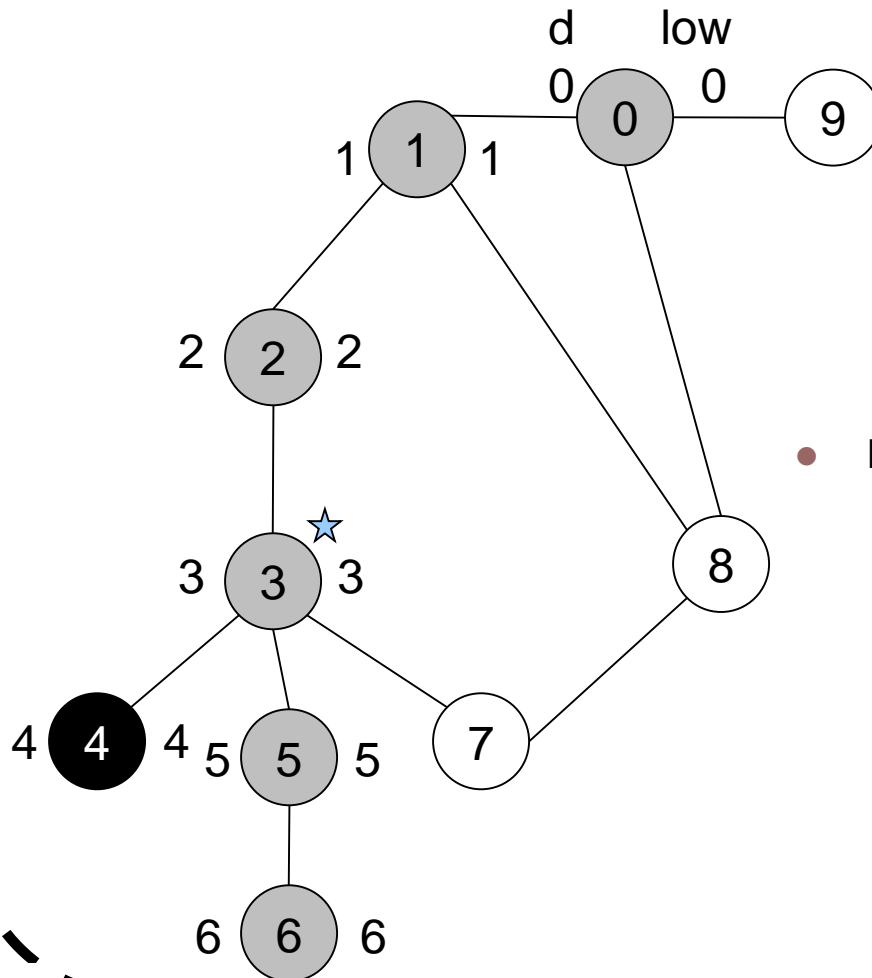
C(5) = gri

$$P(6) = 5$$
$$\text{Subarb}(5) = 1$$

Exploreaza (6)

- Explorează(u)
 - $d(u) = \text{low}(u) = \text{time}++$; // inițializări
 - $c(u) = \text{gri}$;
 - Pentru fiecare nod $v \in \text{succs}(u)$
 - Dacă ($c(v)$ e alb)
 - $p(v) = u$; $\text{subarb}(u)++$; // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - Dacă ($p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u)$) $\text{art}(u) = 1$;
// cazul 2 al teoremei
 - Altfel $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

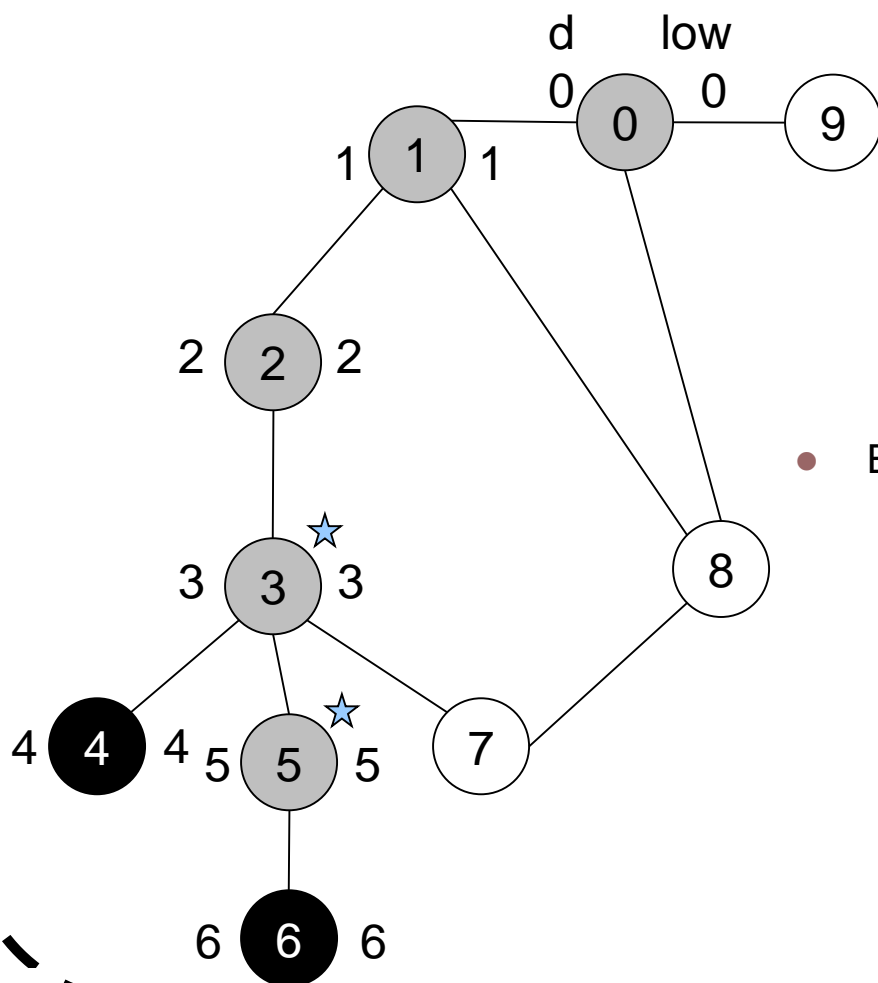
Exemplu rulare (9)



$\text{Low}(6) = d(6) = 6$
 $\text{Timp} = 7$
 $C(6) = \text{gri}$
 revenire

- Explorează(u)
 - $d(u) = \text{low}(u) = \text{timp}++;$ // inițializări
 - $c(u) = \text{gri};$
 - Pentru fiecare nod $v \in \text{succs}(u)$
 - Dacă $(c(v)$ e alb)
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - Dacă $(p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u))$ $\text{art}(u) = 1;$
// cazul 2 al teoremei
 - Altfel $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (10)



$\text{Low}(6) = d(6) = 6$

$\text{Timp} = 7$

$C(6) = \text{gri}$

revenire

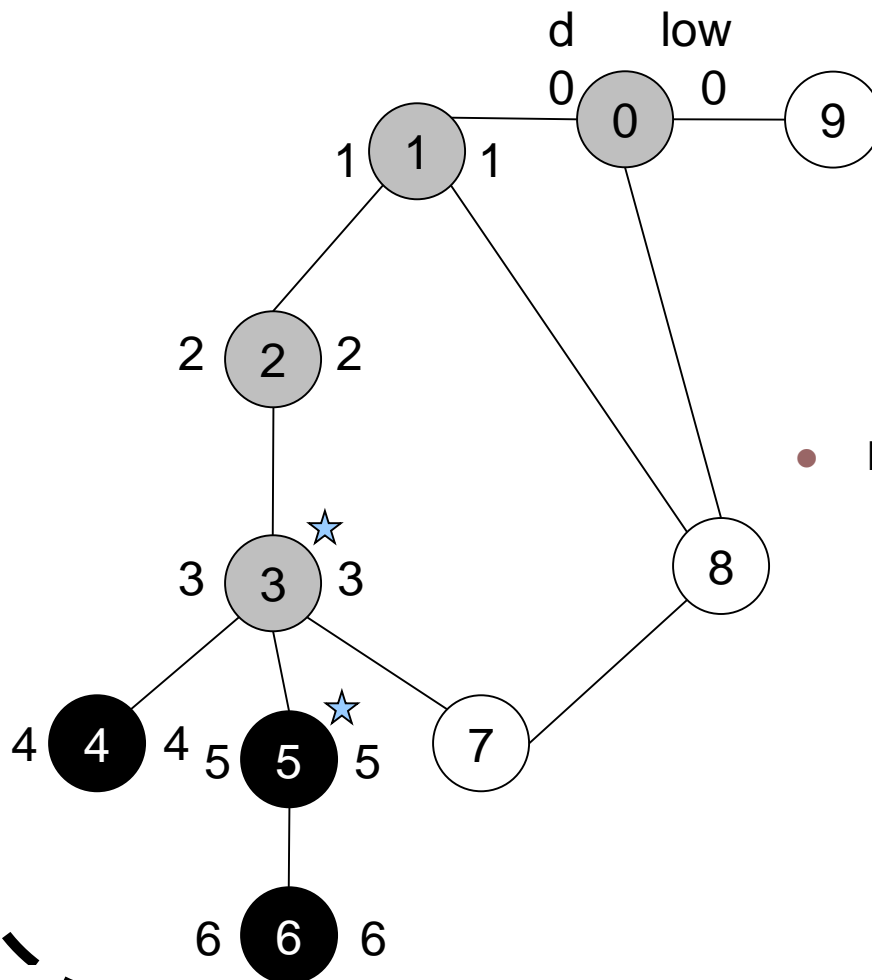
$\text{Low}(5) = \min \{ \text{low}(5), \text{low}(6) \} = 5$

$\text{Low}(6) > d(5) \rightarrow \text{art}(5) = 1$

revenire

- Explorează(u)
 - $d(u) = \text{low}(u) = \text{timp}++;$ // inițializări
 - $c(u) = \text{gri};$
 - Pentru fiecare nod $v \in \text{succs}(u)$
 - Dacă $(c(v) \text{ e alb})$
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - Dacă $(p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u))$ $\text{art}(u) = 1;$
// cazul 2 al teoremei
 - Altfel $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (11)



Low(5) = d(5) = 5

Timp = 7

C(5) = gri

revenire

Low(3) = min {low(3), low(5)} = 3

Low(5) > d(3) → art(3) = 1

P(7) = 3

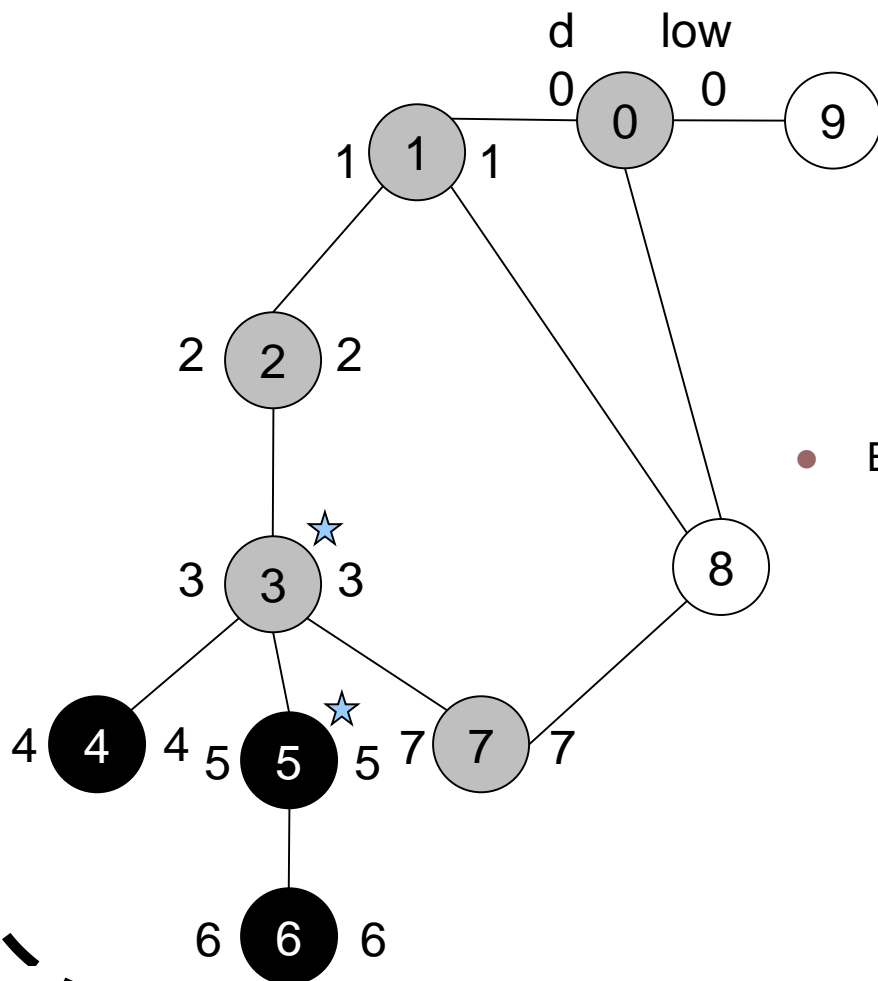
Subarb(3) = 3

Exploreaza (7)

● Explorează(u)

- d(u) = low(u) = timp++; // inițializări
- c(u) = gri;
- Pentru fiecare nod v ∈ succs(u)
 - Dacă (c(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)



$\text{Low}(7) = d(7) = 7$

$\text{Timp} = 8$

$C(7) = \text{gri}$

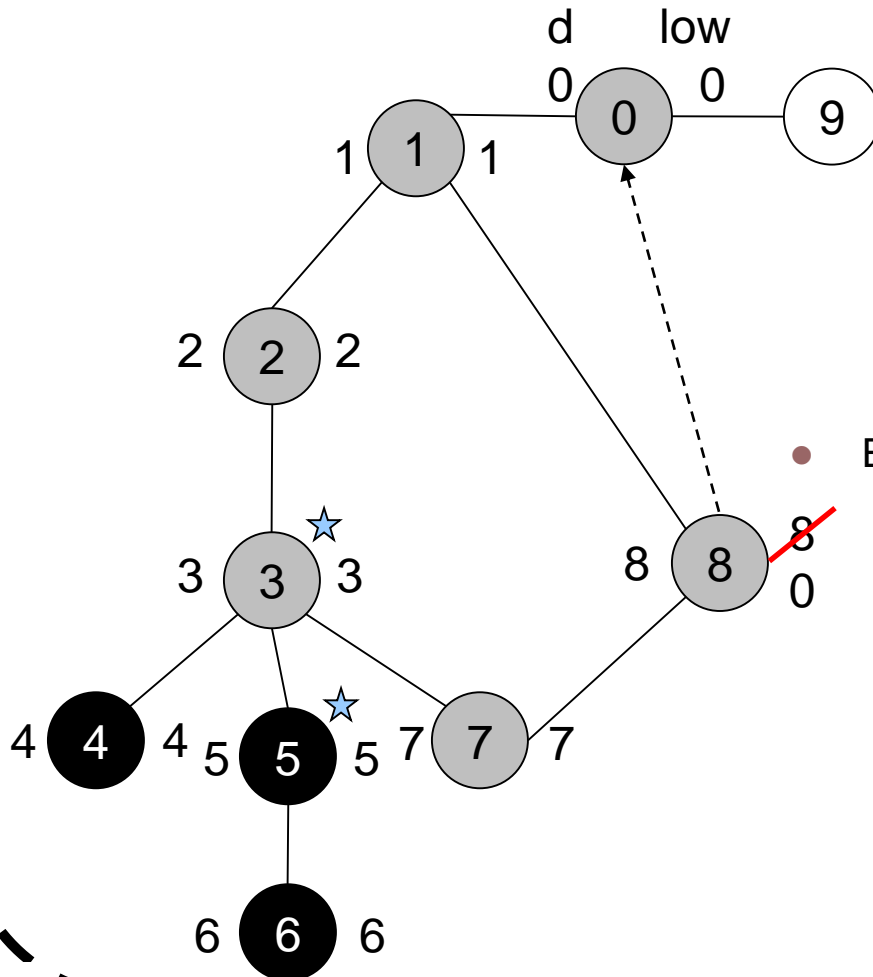
$P(8) = 7$

$\text{Subarb}(7) = 1$

Exploreaza (8)

- Explorează(u)
 - $d(u) = \text{low}(u) = \text{timp}++;$ // inițializări
 - $c(u) = \text{gri};$
 - **Pentru fiecare** nod $v \in \text{succs}(u)$
 - **Dacă** ($c(v)$ e alb)
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - **Dacă** ($p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u)$) $\text{art}(u) = 1;$
// cazul 2 al teoremei
 - **Altfel** $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (13)



$\text{Low}(8) = d(8) = 8$

$\text{Timp} = 9$

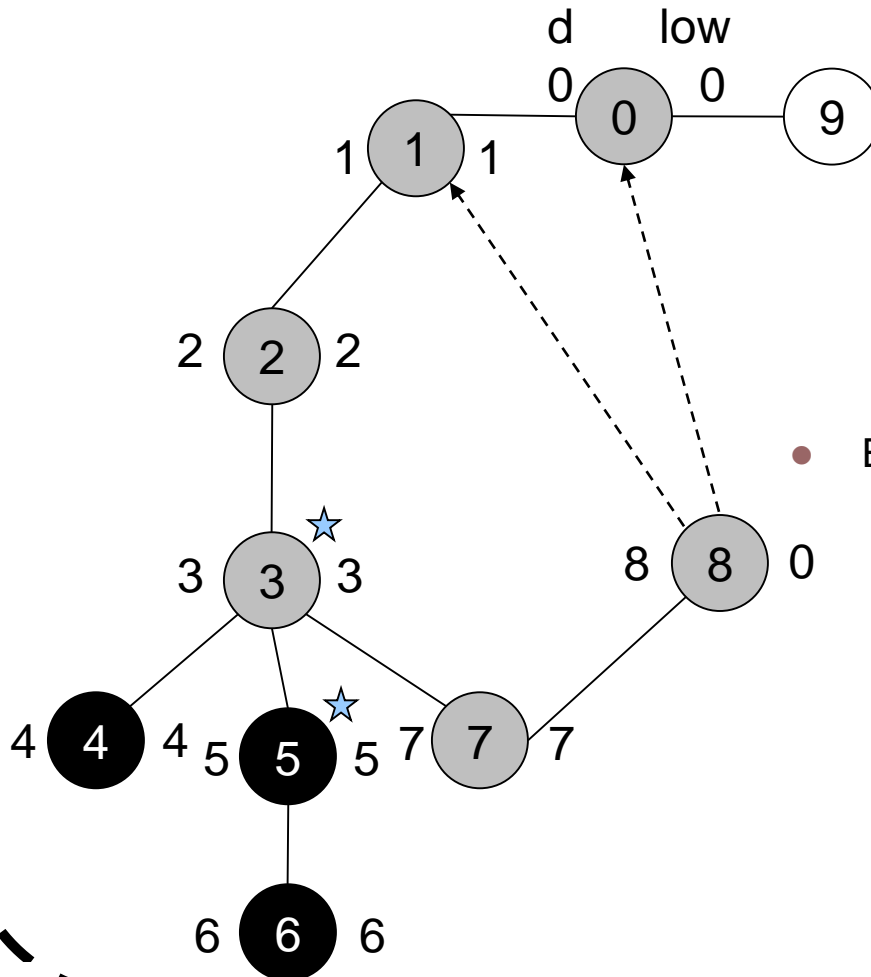
$C(8) = \text{gri}$

$\text{Low}(8) = \min\{d(0), \text{low}(8)\} = 0$

Explorează(u)

- $d(u) = \text{low}(u) = \text{timp}++$; // inițializări
- $c(u) = \text{gri}$;
- **Pentru fiecare** nod $v \in \text{succs}(u)$
 - **Dacă** ($c(v)$ e alb)
 - $p(v) = u$; $\text{subarb}(u)++$; // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - **Dacă** ($p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u)$) $\text{art}(u) = 1$;
// cazul 2 al teoremei
 - **Altfel** $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

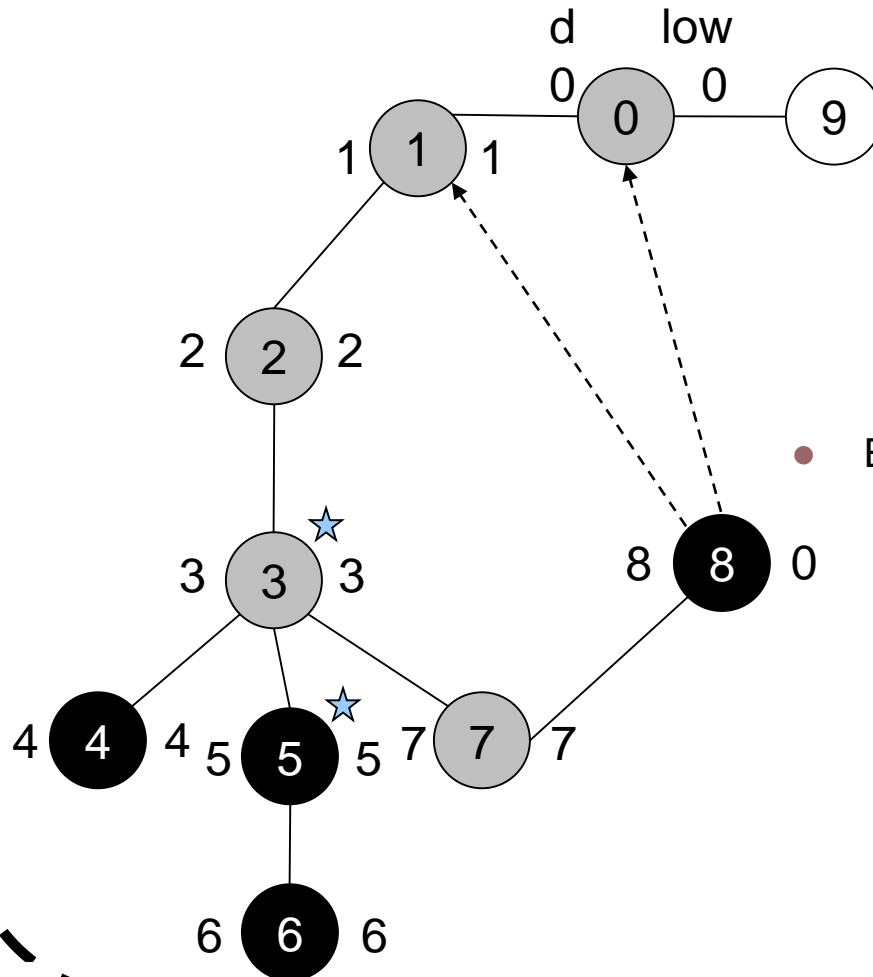
Exemplu rulare (14)



$d(8) = 8$
 $Low(8) = 0$
 $Time = 9$
 $C(8) = gri$
 $Low(8) = \min\{d(1), low(8)\} = 0$

- Explorează(u)
 - $d(u) = low(u) = timp++$; // inițializări
 - $c(u) = gri$;
 - **Pentru fiecare** nod $v \in succs(u)$
 - **Dacă** ($c(v)$ 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** ($p(u) \neq null \ \&\& \ low(v) \geq d(u)$) $art(u) = 1$;
// cazul 2 al teoremei
 - **Altfel** $low(u) = \min\{low(u), d(v)\}$ // actualizare low

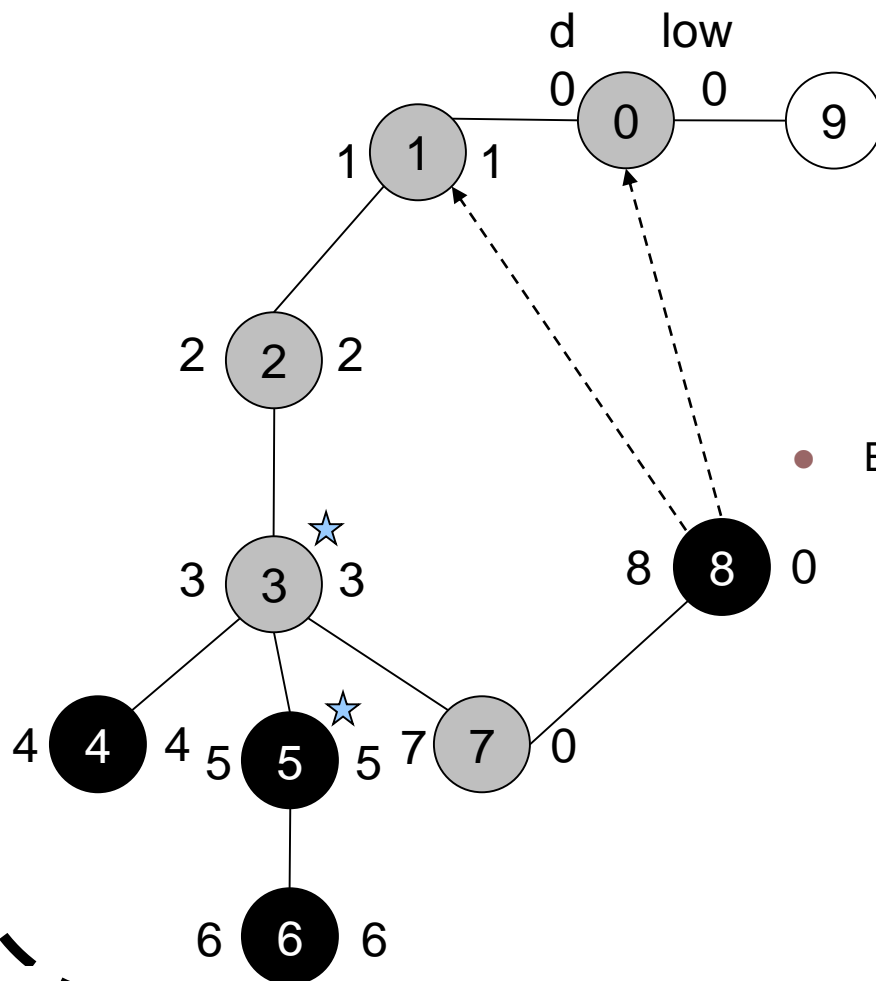
Exemplu rulare (15)



$d(8) = 8$
 $Low(8) = 0$
 $Timp = 9$
 $C(8) = gri$
 revenire

- Explorează(u)
 - $d(u) = low(u) = timp++$; // inițializări
 - $c(u) = gri$;
 - Pentru fiecare nod $v \in succs(u)$
 - Dacă ($c(v)$ 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 ($p(u) \neq null \ \&\& \ low(v) \geq d(u)$) $art(u) = 1$;
// cazul 2 al teoremei
 - Altfel $low(u) = \min\{low(u), d(v)\}$ // actualizare low

Exemplu rulare (16)

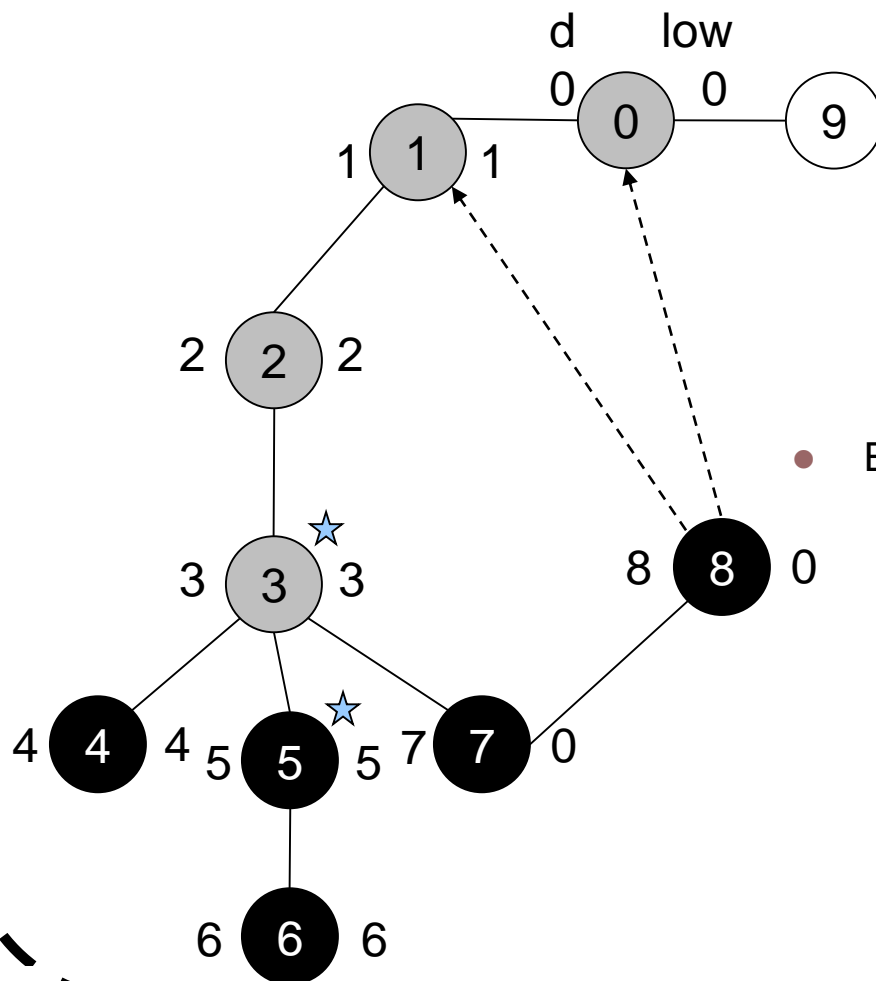


$\text{low}(7) = \min(\text{low}(7), \text{low}(8)) = 0$
 $\text{low}(8) < d(7) \rightarrow$ nu se modifică $\text{art}(7)$

Explorează(u)

- $d(u) = \text{low}(u) = \text{timp}++;$ // inițializări
- $c(u) = \text{gri};$
- Pentru fiecare nod $v \in \text{succs}(u)$
 - Dacă $(c(v)$ e alb)
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - Dacă $(p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u))$ $\text{art}(u) = 1;$ // cazul 2 al teoremei
 - Altfel $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (17)

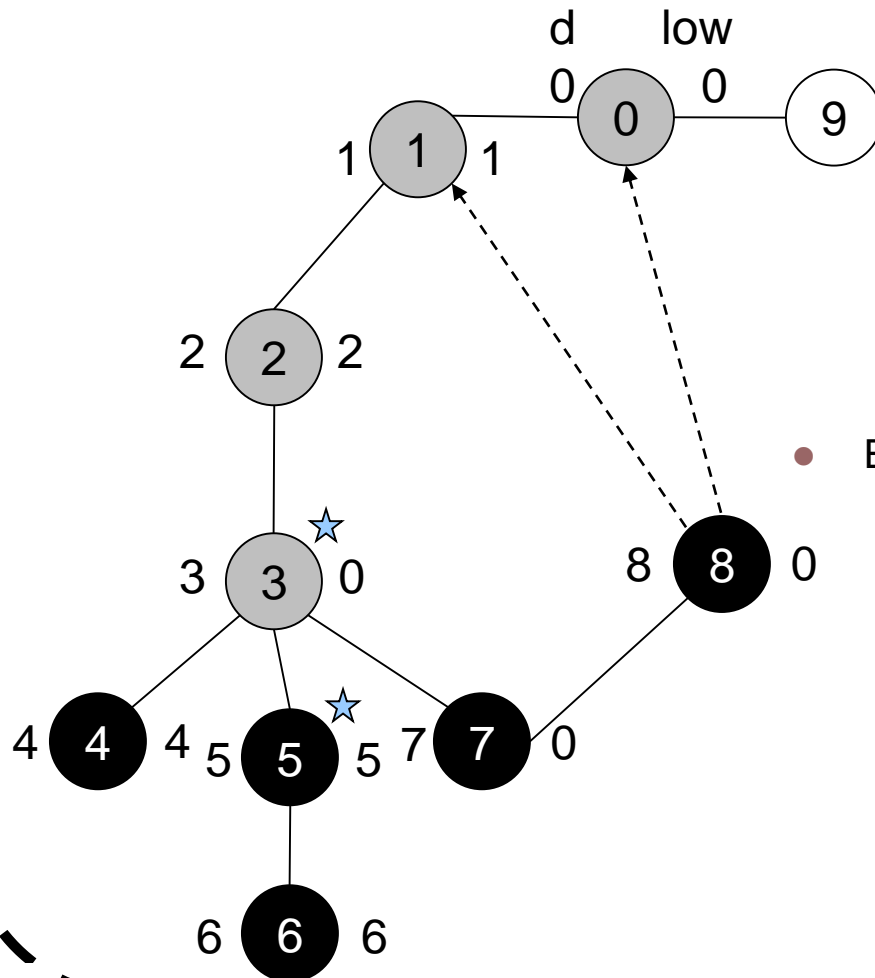


$\text{low}(7) = \min(\text{low}(7), \text{low}(8)) = 0$
revenire

Explorează(u)

- $d(u) = \text{low}(u) = \text{timp}++$; // inițializări
- $c(u) = \text{gri}$;
- **Pentru fiecare** nod $v \in \text{succs}(u)$
 - **Dacă** ($c(v)$ e alb)
 - $p(v) = u$; $\text{subarb}(u)++$; // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - **Dacă** ($p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u)$) $\text{art}(u) = 1$;
// cazul 2 al teoremei
 - **Altfel** $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (18)

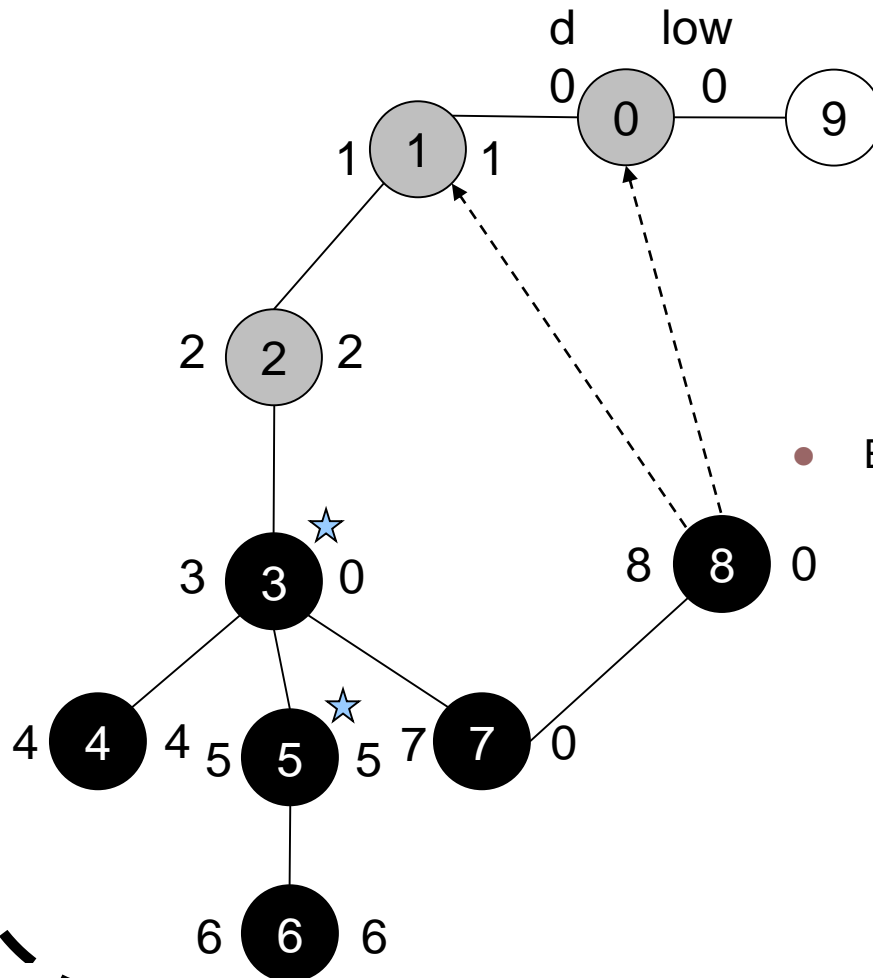


$\text{low}(3) = \min(\text{low}(3), \text{low}(7)) = 0$
 $\text{low}(7) < d(3) \rightarrow$ nu se modifică $\text{art}(3)$

Explorează(u)

- $d(u) = \text{low}(u) = \text{timp}++;$ // inițializări
- $c(u) = \text{gri};$
- **Pentru fiecare** nod $v \in \text{succs}(u)$
 - **Dacă** $(c(v)$ e alb)
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - **Dacă** $(p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u))$ $\text{art}(u) = 1;$
// cazul 2 al teoremei
 - **Altfel** $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (19)

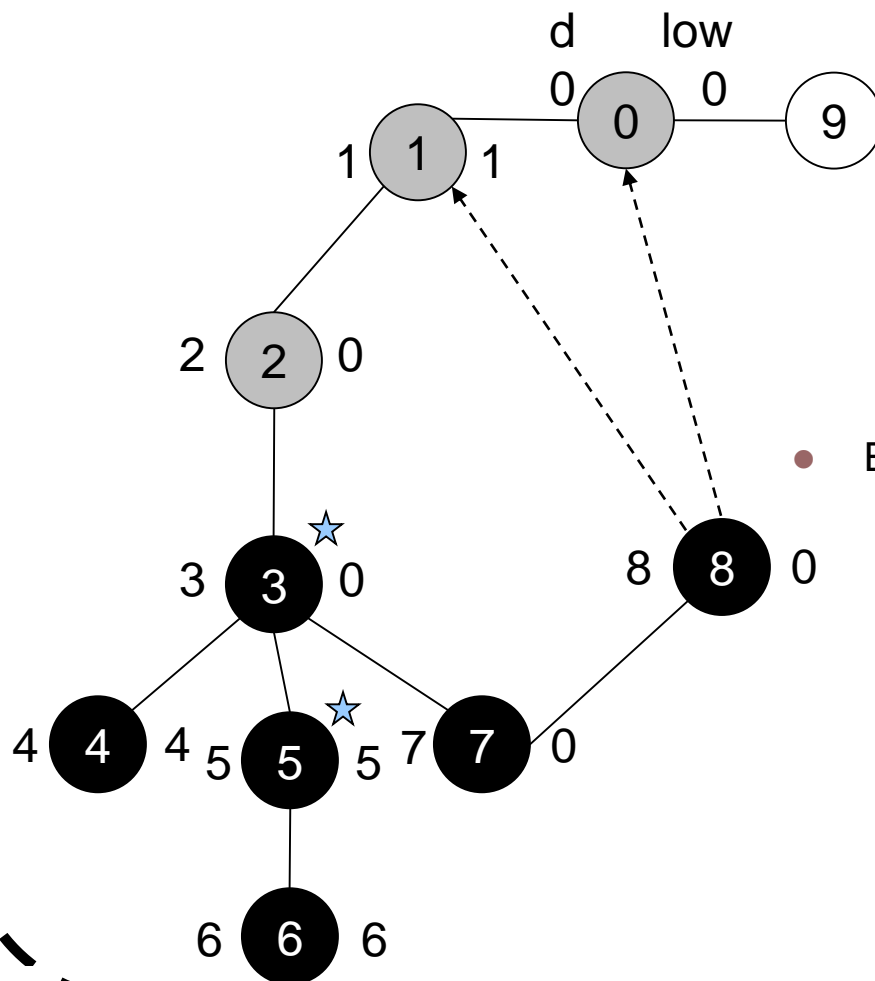


$\text{low}(3) = \min(\text{low}(3), \text{low}(7)) = 0$
 $\text{low}(7) < d(3) \rightarrow$ nu se modifică $\text{art}(3)$
 revenire

Explorează(u)

- $d(u) = \text{low}(u) = \text{timp}++;$ // inițializări
- $c(u) = \text{gri};$
- Pentru fiecare nod $v \in \text{succs}(u)$
 - Dacă $(c(v)$ e alb)
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - Dacă $(p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u))$ $\text{art}(u) = 1;$
// cazul 2 al teoremei
 - Altfel $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (20)

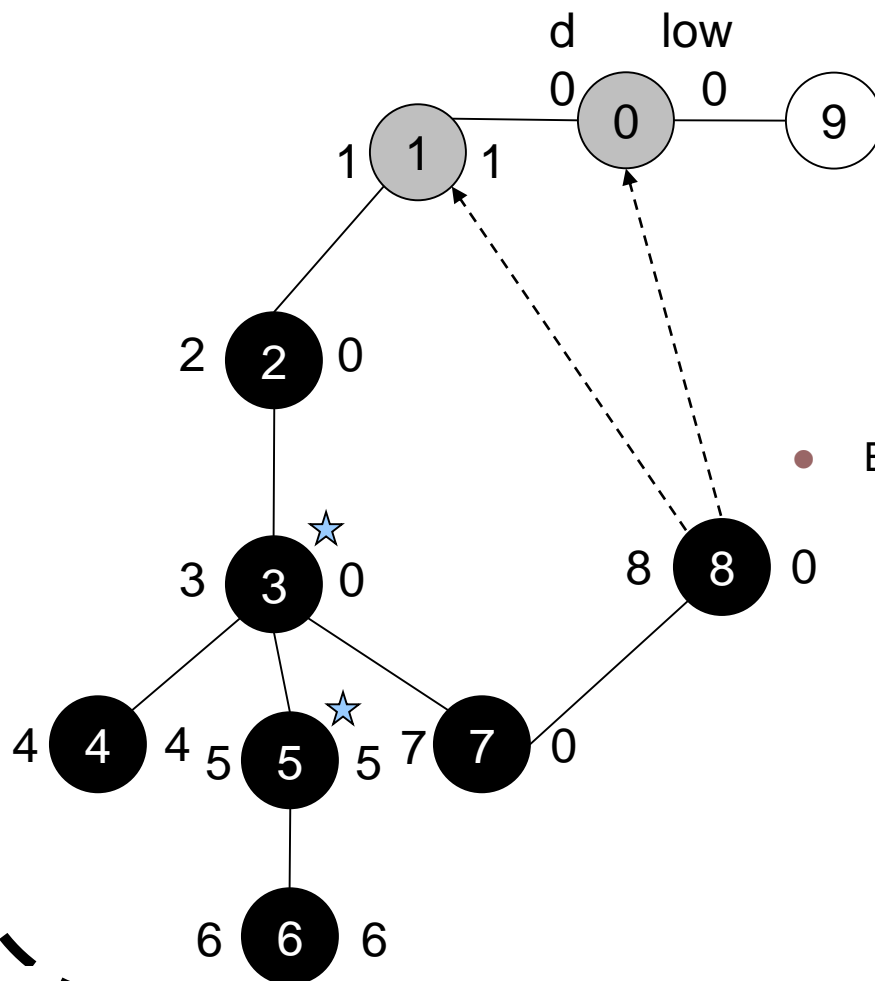


$\text{low}(2) = \min(\text{low}(2), \text{low}(3)) = 0$
 $\text{low}(3) < d(2) \rightarrow$ nu se modifică $\text{art}(2)$

Explorează(u)

- $d(u) = \text{low}(u) = \text{timp}++;$ // inițializări
- $c(u) = \text{gri};$
- **Pentru fiecare** nod $v \in \text{succs}(u)$
 - **Dacă** $(c(v)$ e alb)
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - **Dacă** $(p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u))$ $\text{art}(u) = 1;$
// cazul 2 al teoremei
 - **Altfel** $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (21)

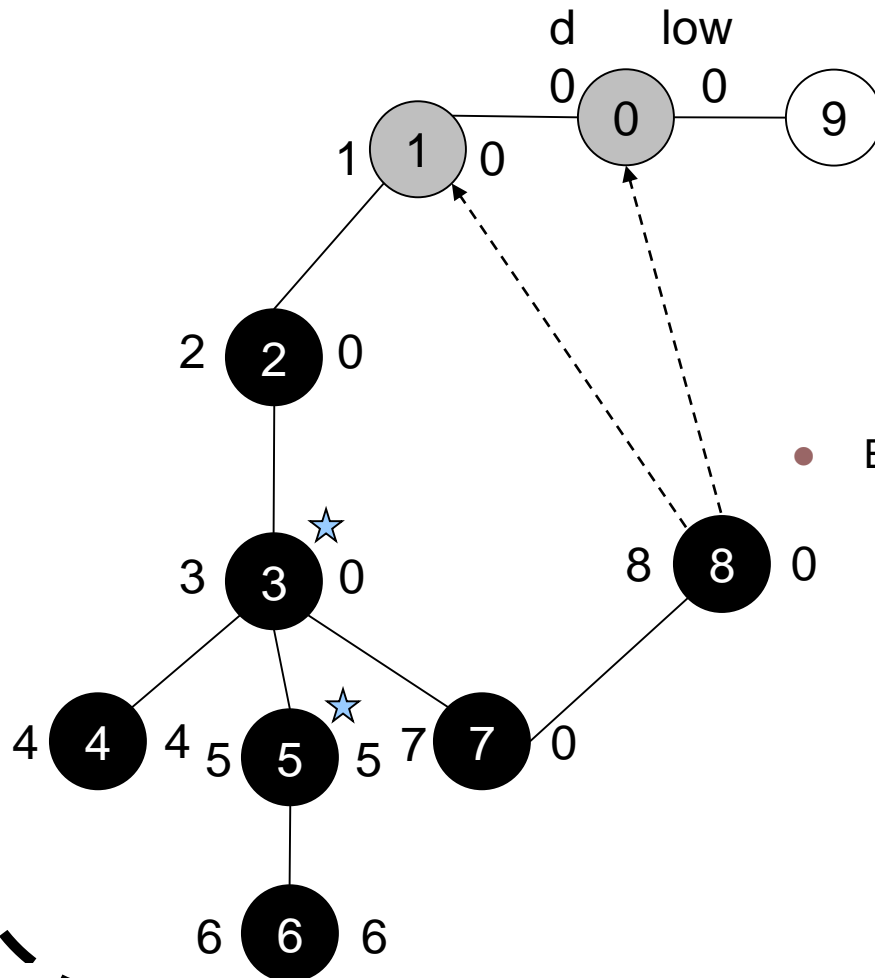


$\text{low}(2) = \min(\text{low}(2), \text{low}(3)) = 0$
 $\text{low}(3) < d(2) \rightarrow$ nu se modifică $\text{art}(2)$
 revenire

Explorează(u)

- $d(u) = \text{low}(u) = \text{ timp}++;$ // inițializări
- $c(u) = \text{gri};$
- **Pentru fiecare** nod $v \in \text{succs}(u)$
 - **Dacă** $(c(v)$ e alb)
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - **Dacă** $(p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u))$ $\text{art}(u) = 1;$
// cazul 2 al teoremei
 - **Altfel** $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (22)

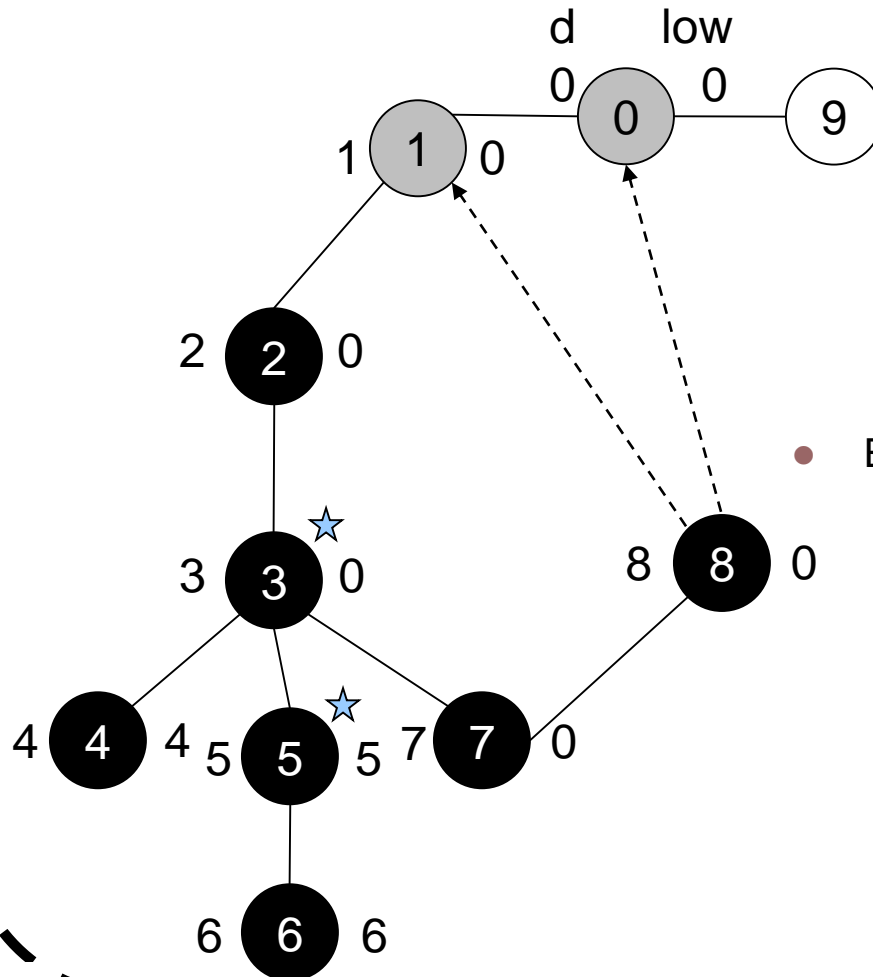


$\text{low}(1) = \min(\text{low}(1), \text{low}(2)) = 0$
 $\text{low}(2) < d(1) \rightarrow \text{nu se modifică art}(1)$

Explorează(u)

- $d(u) = \text{low}(u) = \text{timp}++$; // inițializări
- $c(u) = \text{gri}$;
- **Pentru fiecare** nod $v \in \text{succs}(u)$
 - **Dacă** ($c(v)$ e alb)
 - $p(v) = u$; $\text{subarb}(u)++$; // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - **Dacă** ($p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u)$) $\text{art}(u) = 1$;
// cazul 2 al teoremei
 - **Altfel** $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (23)

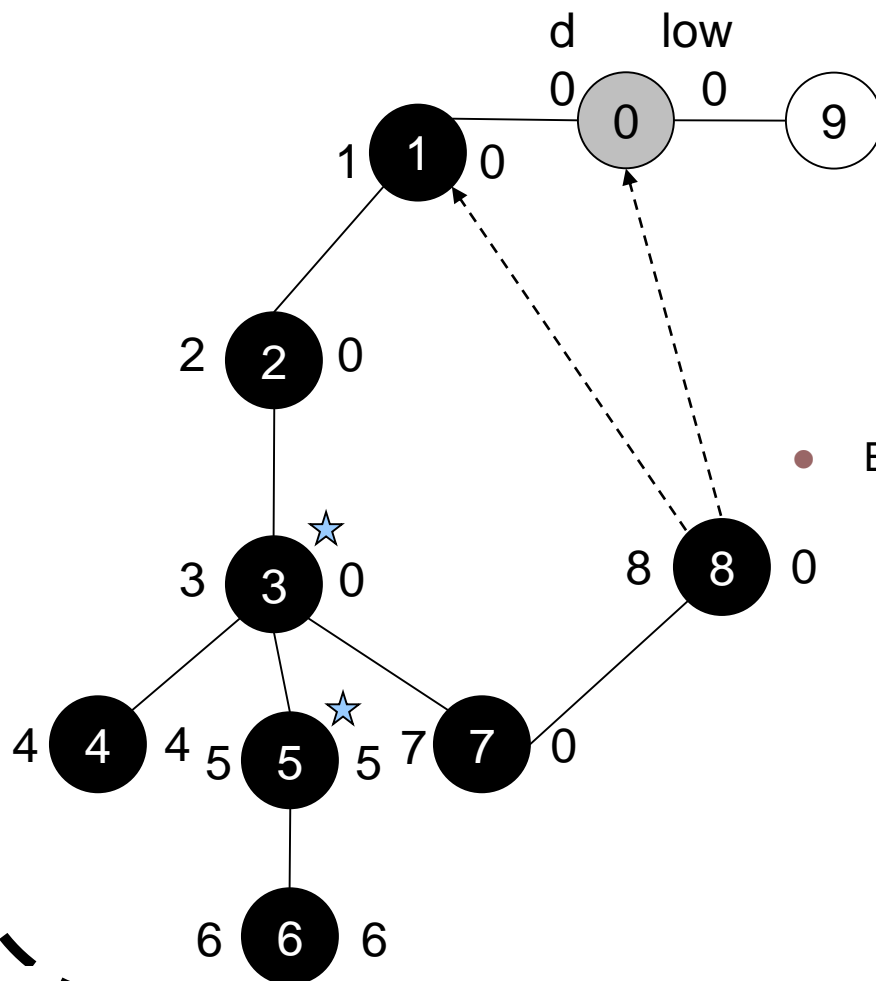


$\text{low}(1) = 0$

$\text{low}(1) = \min(\text{low}(1), d(8)) = 0$

- Explorează(u)
 - $d(u) = \text{low}(u) = \text{timp}++;$ // inițializări
 - $c(u) = \text{gri};$
 - Pentru fiecare nod $v \in \text{succs}(u)$
 - Dacă $(c(v)$ e alb)
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - Dacă $(p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u))$ $\text{art}(u) = 1;$
// cazul 2 al teoremei
 - Altfel $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (24)

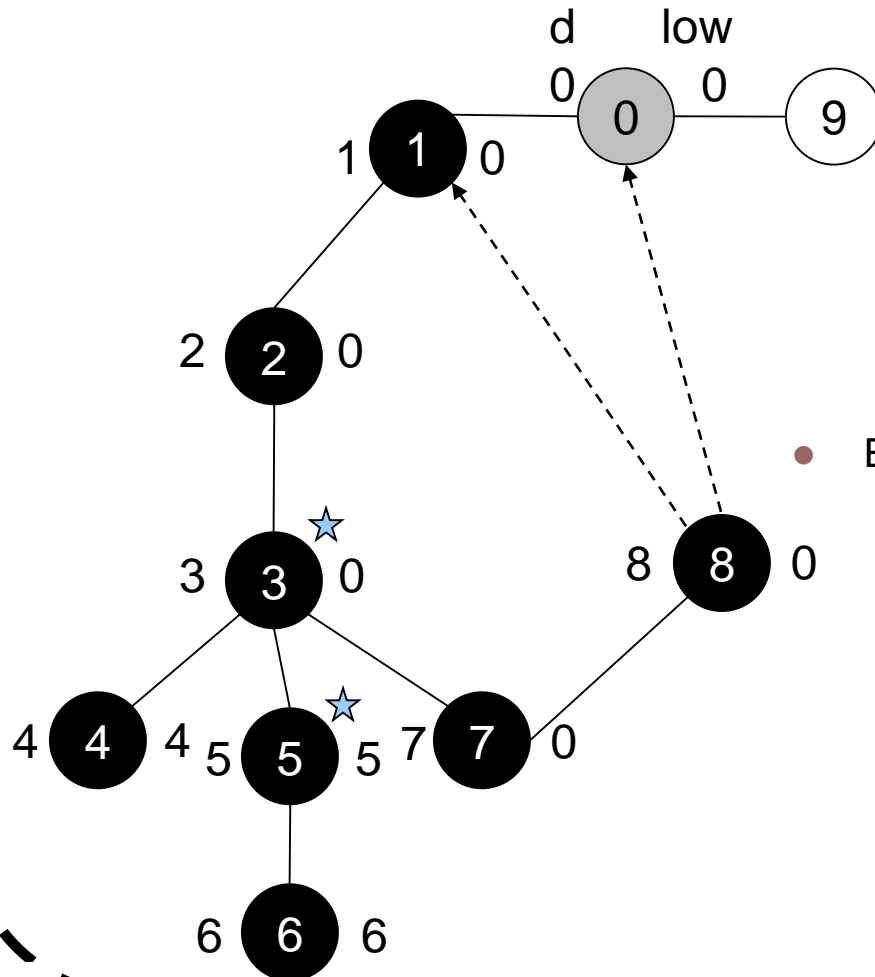


$\text{low}(1) = 0$
revenire

Explorează(u)

- $d(u) = \text{low}(u) = \text{ timp}++$; // inițializări
- $c(u) = \text{gri}$;
- **Pentru fiecare** nod $v \in \text{succs}(u)$
 - **Dacă** ($c(v)$ e alb)
 - $p(v) = u$; $\text{subarb}(u)++$; // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - **Dacă** ($p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u)$) $\text{art}(u) = 1$;
// cazul 2 al teoremei
 - **Altfel** $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

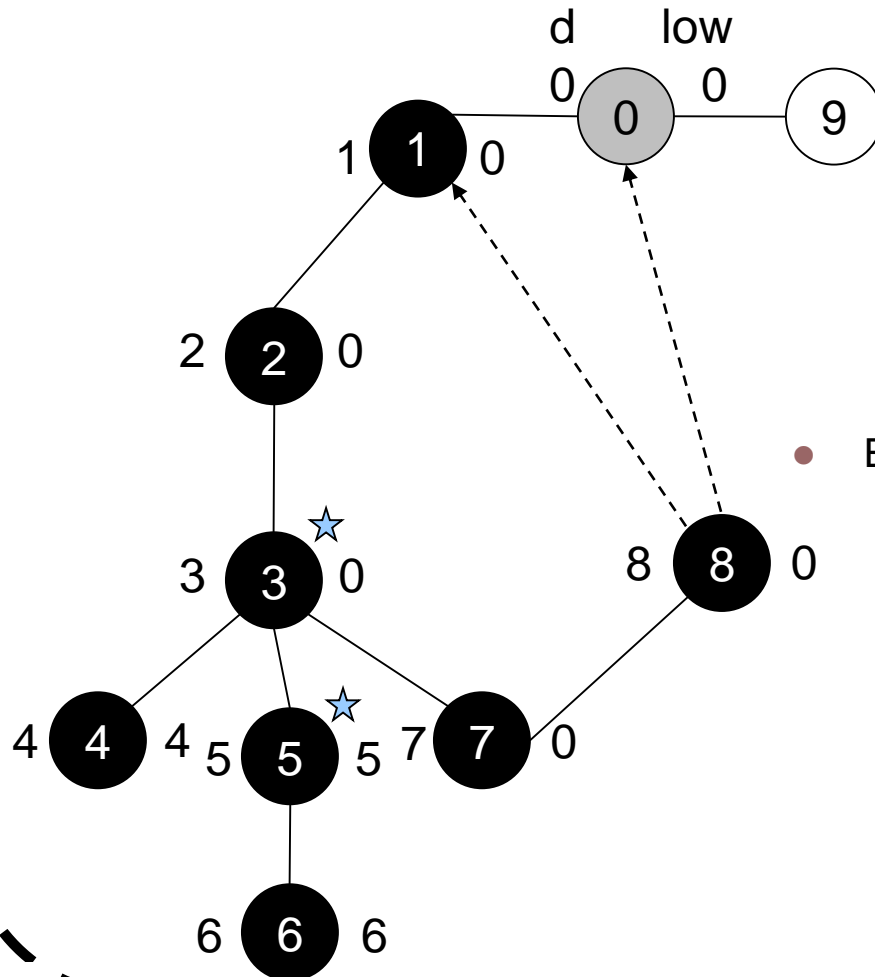
Exemplu rulare (25)



$\text{low}(0) = \min\{\text{low}(1), \text{low}(0)\} = 0$
 $P(0) = \text{null} \rightarrow \text{continuă cu următorul copil}$

- Explorează(u)
 - $d(u) = \text{low}(u) = \text{timp}++;$ // inițializări
 - $c(u) = \text{gri};$
 - Pentru fiecare nod $v \in \text{succs}(u)$
 - Dacă $(c(v)$ e alb)
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - Dacă $(p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u))$ $\text{art}(u) = 1;$
// cazul 2 al teoremei
 - Altfel $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (26)



$\text{low}(0) = \min(\text{low}(0), d(8)) = 0$

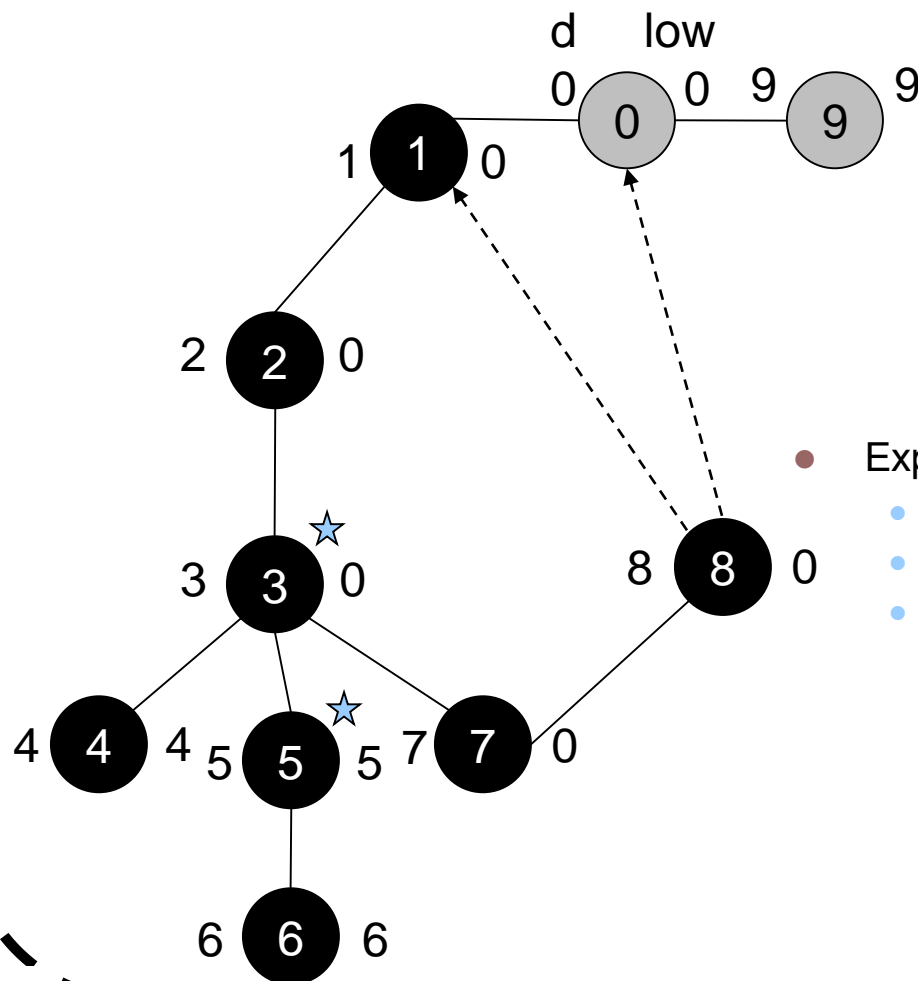
$P(9) = 0$

$\text{Subarb}(0) = 2$

Exploreaza (9)

- Explorează(u)
 - $d(u) = \text{low}(u) = \text{timp}++;$ // inițializări
 - $c(u) = \text{gri};$
 - Pentru fiecare nod $v \in \text{succs}(u)$
 - Dacă $(c(v)$ e alb)
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - Dacă $(p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u))$ $\text{art}(u) = 1;$
// cazul 2 al teoremei
 - Altfel $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (27)

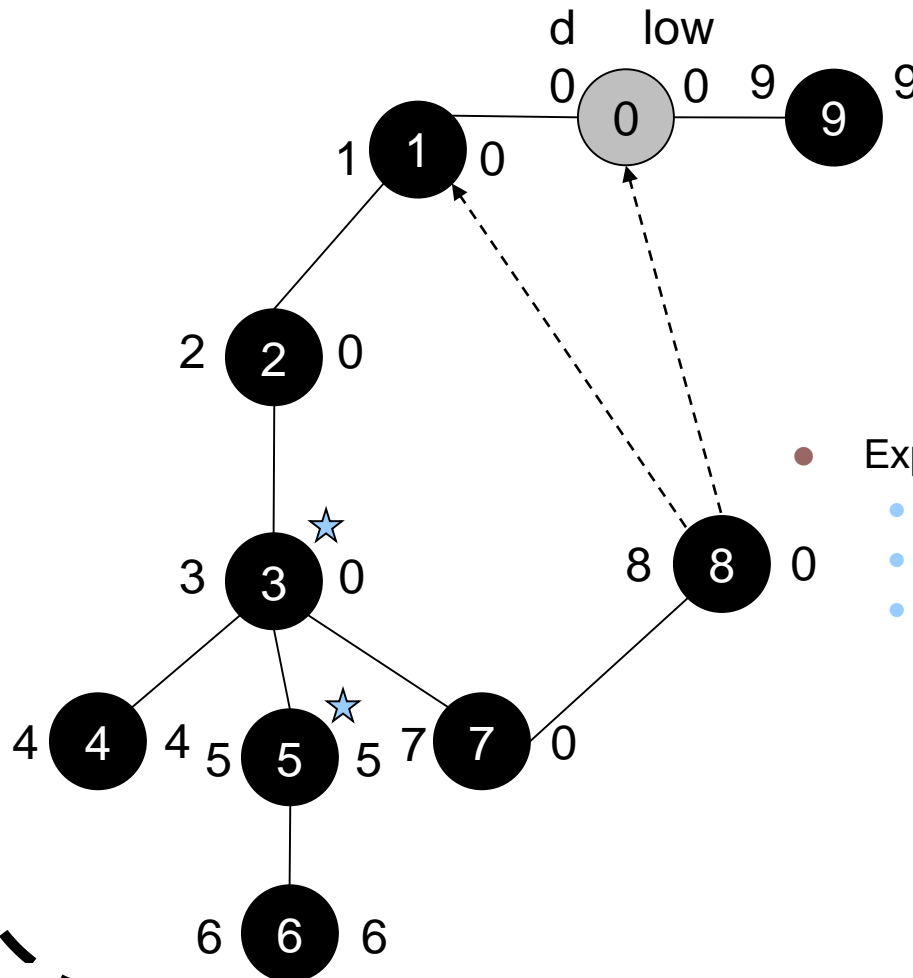


$\text{Low}(9) = d(9) = 9$
 $\text{Timp} = 10$
 $C(9) = \text{gri}$
 revenire

Explorează(u)

- $d(u) = \text{low}(u) = \text{timp}++$; // inițializări
- $c(u) = \text{gri}$;
- **Pentru fiecare** nod $v \in \text{succs}(u)$
 - **Dacă** ($c(v)$ e alb)
 - $p(v) = u$; $\text{subarb}(u)++$; // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - **Dacă** ($p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u)$) $\text{art}(u) = 1$;
// cazul 2 al teoremei
 - **Altfel** $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (28)

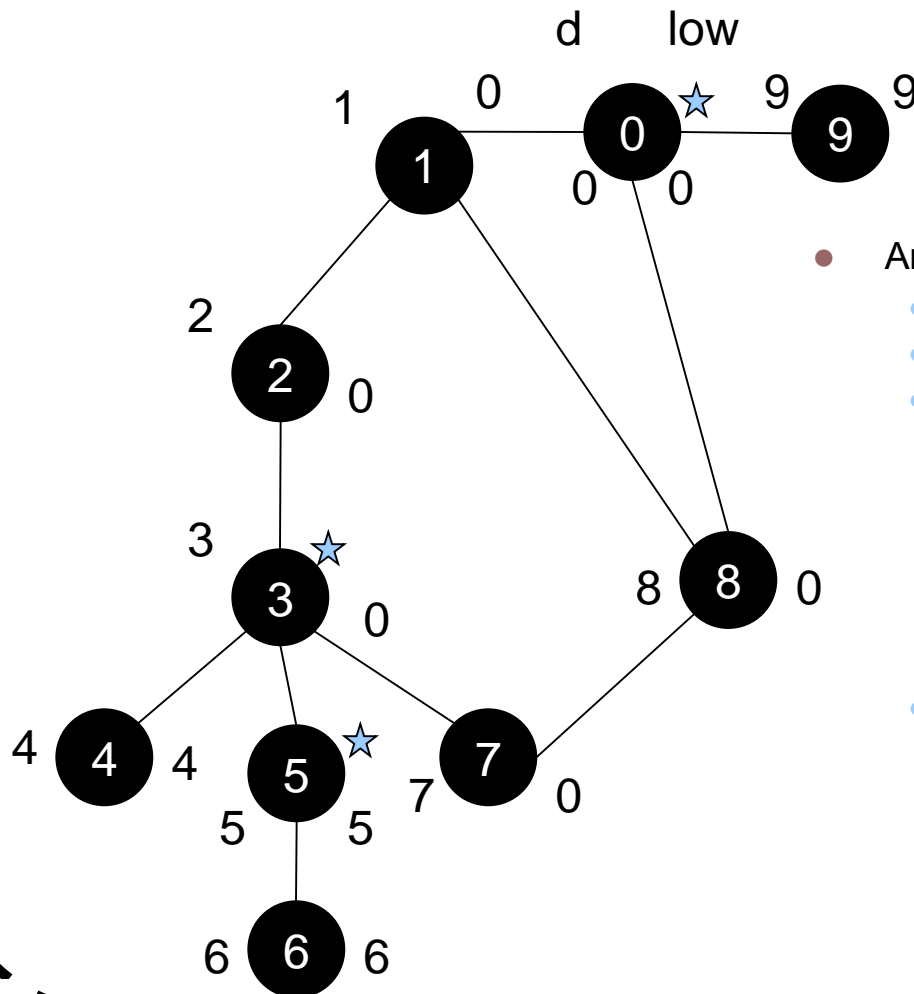


$\text{low}(0) = \min(\text{low}(0), \text{low}(9)) = 0$
 $P(0) = \text{null} \rightarrow \text{revenire}$

● Explorează(u)

- $d(u) = \text{low}(u) = \text{timp}++;$ // inițializări
- $c(u) = \text{gri};$
- Pentru fiecare nod $v \in \text{succs}(u)$
 - Dacă $(c(v)$ e alb)
 - $p(v) = u;$ $\text{subarb}(u)++;$ // actualizare nr
// subarbori dominați de u
 - Explorează(v);
 - $\text{low}(u) = \min\{\text{low}(u), \text{low}(v)\}$ // actualizare low
 - Dacă $(p(u) \neq \text{null} \ \&\& \ \text{low}(v) \geq d(u))$ $\text{art}(u) = 1;$
// cazul 2 al teoremei
 - Altfel $\text{low}(u) = \min\{\text{low}(u), d(v)\}$ // actualizare low

Exemplu rulare (29)



$\text{Subarb}(0) = 2 > 1 \rightarrow \text{art}(0) = 1$

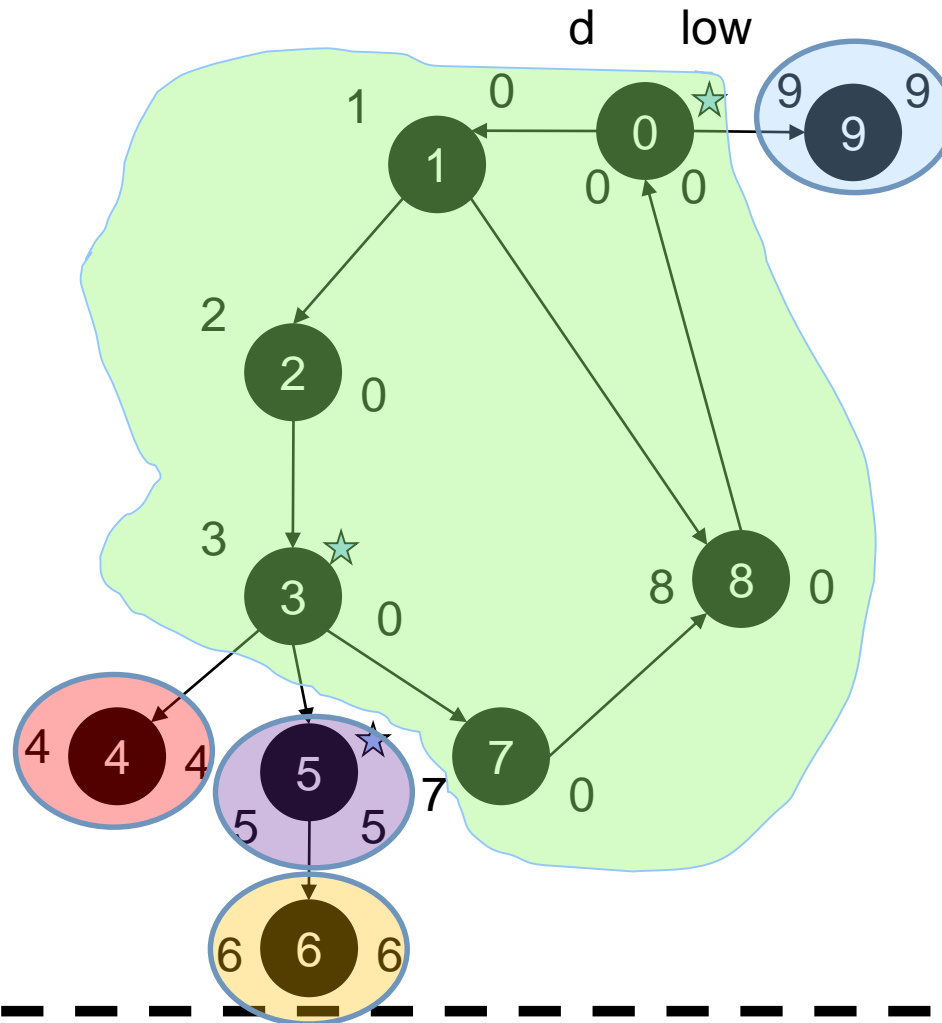
Articulații (G)

- $V = \text{noduri}(G)$ // inițializări
- $\text{Timp} = 0$;
- **Pentru fiecare** $(u \in V)$
 - $c(u) = \text{alb}$;
 - $d(u) = 0$;
 - $p(u) = \text{null}$;
 - $\text{low}(u) = 0$;
 - $\text{subarb}(u) = 0$; // reține numărul de subarbori // dominați de u
 - $\text{art}(u) = 0$; // reține punctele de articulație
- **Pentru fiecare** $(u \in V)$
 - Dacă $c(u)$ e alb
 - Explorează(u);
 - Dacă $(\text{subarb}(u) > 1)$ // cazul în care u este // rădăcina în arborele
 - $\text{art}(u) = 1$ // DFS și are mai mulți subarbori
 - // \rightarrow cazul 1 al teoremei

Algoritmul lui Tarjan adaptat pentru determinarea CTC

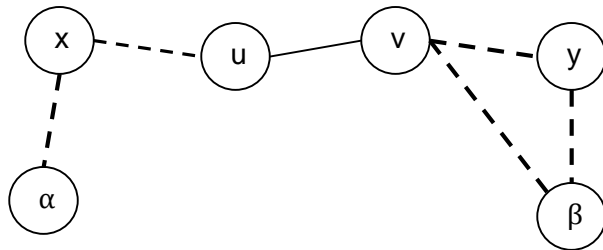
- index = 0 // nivelul pe care este nodul în arborele DFS
- S = empty // se folosește o stivă care se inițializează cu \emptyset
- **Pentru fiecare** v din V
 - **Dacă** (v.index e nedefinit) **atunci** // se pornește DFS din fiecare nod pe care // nu l-am vizitat încă
 - Tarjan(v)
- Tarjan(v)
 - v.index = index // se setează nivelul nodului v
 - v.lowlink = index // reține strămoșul nodului v
 - index = index + 1 // incrementez nivelul
 - S.push(v) // introduc v în stivă
 - **Pentru fiecare** (v, v') din E // se prelucrează succesorii lui v
 - **Dacă** (v'.index e nedefinit **sau** v' e în S) **atunci** // CTC deja identificate sunt ignorate
 - **Dacă** (v'.index e nedefinit) **atunci** Tarjan(v') // dacă nu a fost vizitat v' într-o 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)

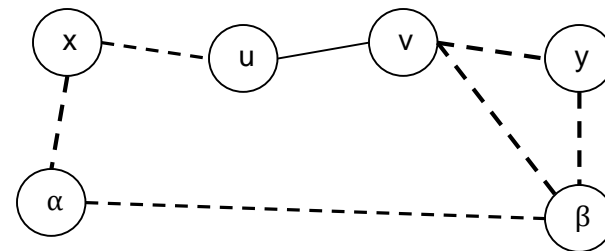


Punți

- Definiție: $G = (V, E)$, graf neorientat și $(u, v) \in E$. (u, v) este **punte** în $G \Leftrightarrow \exists x, y \in V, x \neq y$, a.î. $\forall x..y$ conține muchia (u, v) .



Orice drum $x..y$ trece prin (u, v)
 $\Rightarrow (u, v)$ este **punte**



(u, v) **nu este punte**

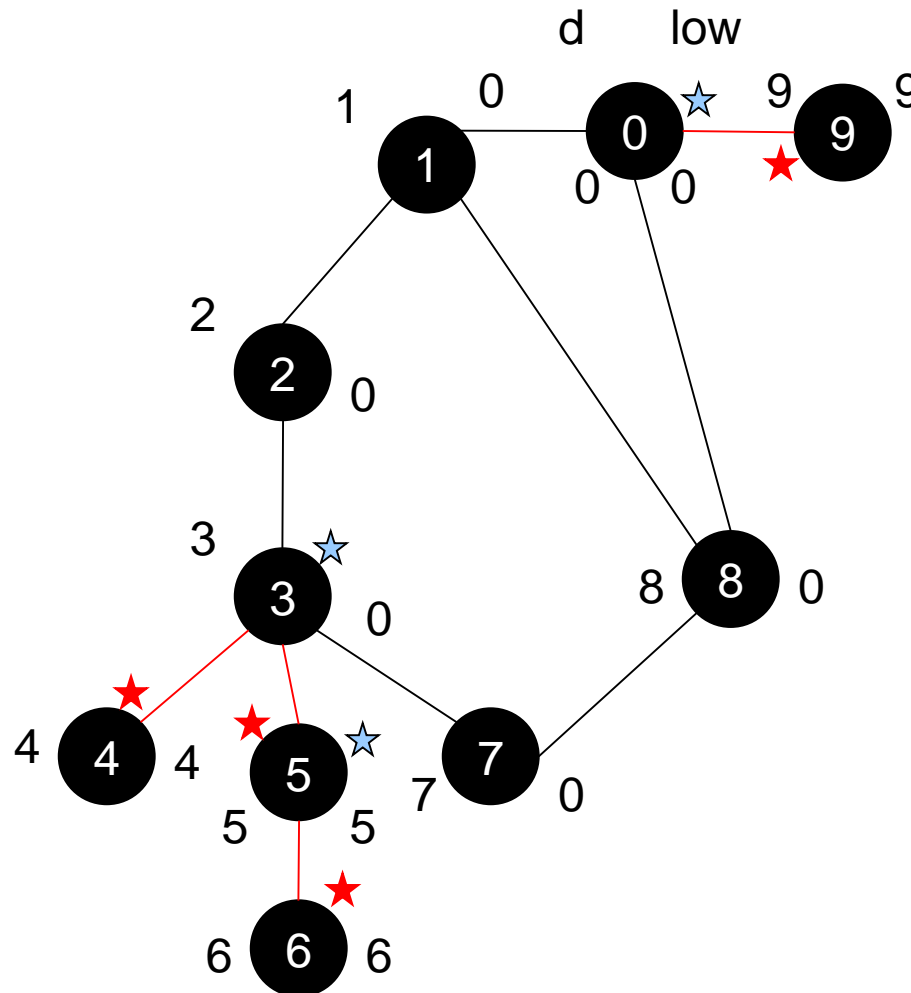
Algoritm punți (I)

- **Punți(G)**
 - $V = \text{noduri}(G)$ // inițializări
 - $\text{Timp} = 0;$
 - **Pentru fiecare** nod u ($u \in V$)
 - $c(u) = \text{alb};$
 - $d(u) = 0;$
 - $p(u) = \text{null};$
 - $\text{low}(u) = 0;$
 - $\text{punte}(u) = 0;$ // înlocuiește: $\text{subarb}(u) = 0;$ $\text{art}(u) = 0;$
 - **Pentru fiecare** nod u ($u \in V$)
 - **Dacă** $c(u)$ e alb
 - Explorează(u)

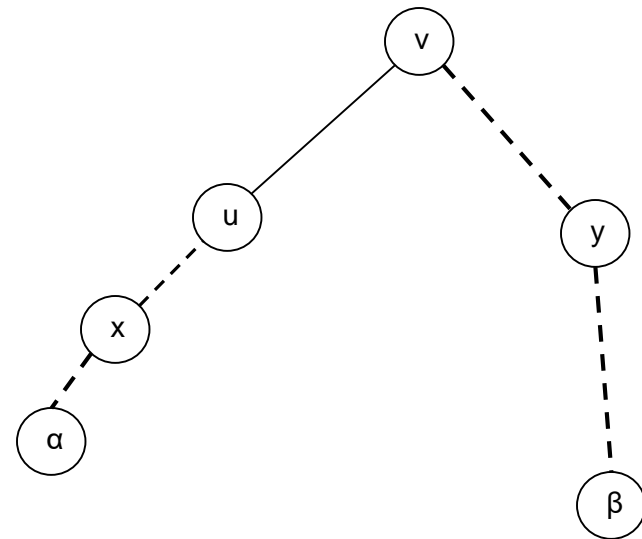
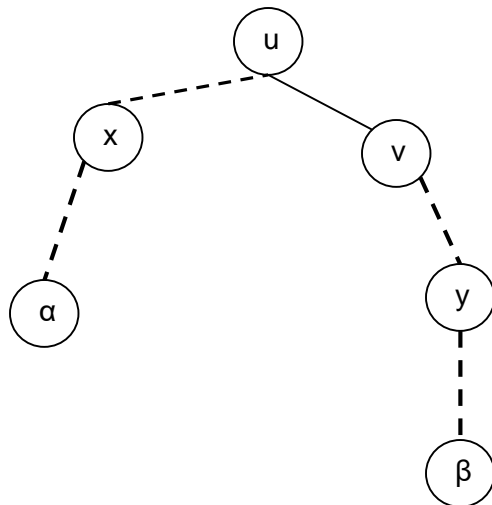
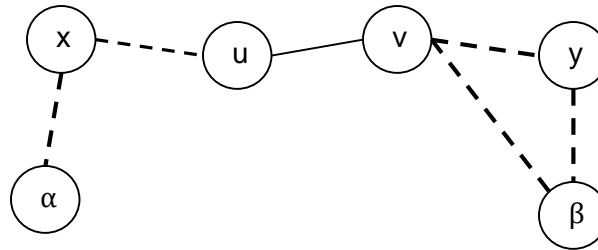
Algoritm punți (II)

- Explorează(u)
 - $d(u) = low(u) = timp++$; // inițializări
 - $c(u) = gri$;
 - **Pentru fiecare** nod v ($v \in succs(u)$)
 - **Dacă** $c(v)$ e alb
 - $p(v) = u$; // se elimină: $subarb(u)++$;
 - Explorează(v);
 - $low(u) = \min\{low(u), low(v)\}$ // actualizare low
 - **Dacă** ($low(v) > d(u)$) $punte(v) = 1$;
 - // în loc de: **Dacă**($p(u) \neq null \ \&\& \ low(v) \geq d(u)$)
 - Altfel
 - **Dacă** ($p(u) \neq v$) $low(u) = \min\{low(u), d(v)\}$ // actualizare low

Exemplu rulare (Punți)



Exemplu



DFS din u ; puntea este detectată în v

DFS din v ; puntea este detectată în u

Drumuri de cost minim

- $G = (V, E)$ un graf, iar $w: E \rightarrow \mathbb{R}$ o funcție de cost asociată arcelor grafului ($w(u, v) = \text{costul arcului } (u, v)$).
- $\text{Cost}(u..v) = \text{costul drumului } u..v$ (este aditiv – costul drumului = suma costurilor arcelor).
- Variante:
 1. Drumuri punct – multipunct: pentru un nod dat $s \in V$, să se găsească un drum de cost minim de la s la $\forall u \in V$; **Dijkstra, Bellman-Ford**
 2. Drumuri multipunct – punct: pentru un nod dat $e \in V$, să se găsească un drum de cost minim de la $\forall u \in V$ la e ; **G^T și apoi 1**
 3. Drumuri punct – punct: pentru două noduri date u și $v \in V$, să se găsească un drum $u..v$ de cost minim; **Folosind 1**
 4. Drumuri multipunct – multipunct: $\forall u, v \in V$, să se găsească un drum $u..v$ de cost minim. **Floyd-Warshall**
 5. Drumuri de cost maxim!

Temă de gândire pentru acasă – posibil subiect de examen!

Optimalitatea drumurilor minime (I)

- **Lemă 25.1 (Subdrumurile unui drum minim sunt drumuri optimale):** $G = (V, E)$, $w : E \rightarrow \mathbb{R}$ funcție de cost asociată. Fie $p = v_1 v_2 \dots v_k$ un drum optim de la v_1 la v_k . Atunci pentru orice i și j cu $1 \leq i \leq j \leq k$, subdrumul lui p de la v_i la v_j este un drum minim.
- **Dem:** Fie $p_{ij} = v_i \dots v_j$ subdrumul din p dintre v_i și v_j . $\rightarrow p = v_1 \dots v_i \dots v_j \dots v_k \Rightarrow \text{cost}(p) = \text{cost}(v_1 \dots v_i) + \text{cost}(v_i \dots v_j) + \text{cost}(v_j \dots v_k)$.
- Pp. prin absurd că $v_i \dots v_j$ nu e optim $\Rightarrow \exists p'$ a.î. $\text{cost}(p') < \text{cost}(v_i \dots v_j) \Rightarrow p$ nu e drum minim \rightarrow Contrazice ipoteza $\rightarrow p_{ij}$ este drum minim.

Optimalitatea drumurilor minime (II)

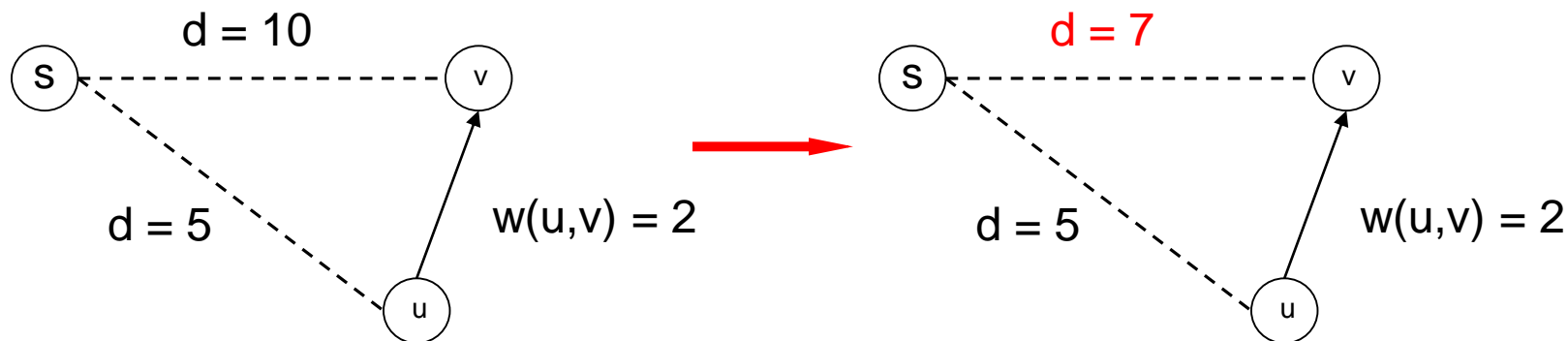
- **Corolar 25.2:** $G = (V, E)$, $w : E \rightarrow \mathbb{R}$ funcție de cost asociată. Fie $p = s..uv$ un drum optim de la s la v . Atunci costul optim al acestui drum poate fi scris ca $\delta(s, v) = \delta(s, u) + w(u, v)$.
- **Dem:** Conform teoremei anterioare, $s..u$ e un drum optim $\Rightarrow \text{cost}(s..u) = \delta(s, u)$.
- **Lemă 25.3:** $G = (V, E)$, $w : E \rightarrow \mathbb{R}$ funcție de cost asociată. $\forall (u, v) \in E$ avem $\delta(s, v) \leq \delta(s, u) + w(u, v)$.
- **Dem:** Orice drum optim are costul mai mic ca al oricărui alt drum.

Drumuri minime de sursă unică

- Sunt concepuți pentru **grafuri orientate**.
- Bazați pe **algoritmi Greedy**.
- Se pornește de la nodul de start și pe baza unui optim local, drumurile sunt extinse și optimizate până la soluția finală.
- Notatii:
 - $d[v]$ = costul drumului descoperit s..v;
 - $\delta(u,v)$ = costul drumului optim u..v; $\delta(u,v)=\infty$ dacă $v \notin R(u)$;
 - $p(v)$ = predecesorul lui v pe drumul s..v.

Drumuri minime de sursă unică

- **Relaxarea arcelor** \rightarrow dacă $d[v] > d[u] + w(u,v)$, atunci actualizează $d[v]$.



- Example: Dijkstra și Bellman–Ford.

Algoritmul lui Dijkstra (I)

- Folosește o coadă de priorități în care se adaugă nodurile în funcție de distanța cunoscută în momentul respectiv de la s până la nod.
 - Se folosește **NUMAI** pentru costuri pozitive ($w(u,v) > 0, \forall u,v \in V$).
 - Dijkstra_generic (G,s)
 - V = nodurile lui G
 - **Cât timp** ($V \neq \emptyset$)
 - u = nod din V cu $d[u]$ min
 - $V = V - \{u\}$
 - **Pentru fiecare** ($v \in$ succesorii lui u) relaxare_arc(u,v)
- // optimizare drum $s..v$ pentru $v \in$ succesorilor lui u

Relaxarea arcelor (I)

- **Lemă 25.5:** $G = (V, E)$, $w : E \rightarrow \mathbb{R}$ funcție de cost asociată. $\forall v \in V$, $d[v]$ obținut de algoritmul lui Dijkstra respectă $d[v] \geq \delta(s, v)$. În plus, odată atinsă valoarea $\delta(s, v)$, ea nu se mai modifică.
- **Dem:**
 - $\forall v \in V, v \notin R(s) \rightarrow d[v] = \delta(s, v) = \infty$; $d[s] = \delta(s, s) = 0$ (inițializare)
 - Pt $v \in R(s)$, inițializare $\rightarrow d[v] = \infty \geq \delta(s, v)$. Dem. prin reducere la absurd că după oricâte relaxări, relația se menține. Fie v primul vârf pentru care relaxarea (u, v) determină $d[v] < \delta(s, v) \rightarrow$ după relaxarea (u, v) : $d[u] + w(u, v) = d[v] < \delta(s, v) \leq \delta(s, u) + w(u, v) \rightarrow d[u] < \delta(s, u)$. Dar relaxarea nu modifică $d[u]$, iar v e primul pentru care $d[v] < \delta(s, v)$. Contrazice presupunerea! $\Rightarrow d[v] \geq \delta(s, v), \forall v \in V$
 - Cum $d[v] \geq \delta(s, v) \Rightarrow$ odată ajuns la $d[v] = \delta(s, v)$, ea nu mai scade. Cum relaxarea nu crește valorile $\Rightarrow d[v]$ nu se mai modifică.

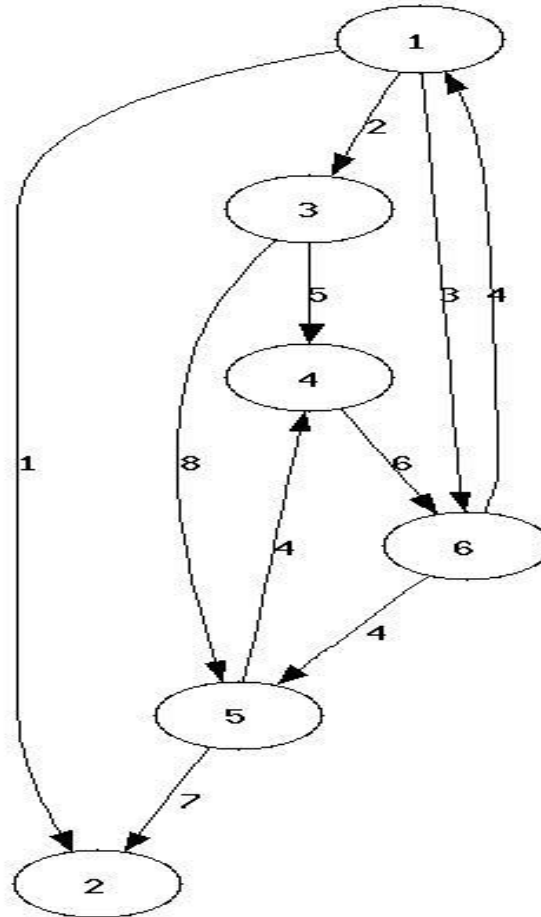
Relaxarea arcelor (II)

- **Lemă 25.7:** $G = (V, E)$, $w : E \rightarrow \mathbb{R}$ funcție de cost asociată. Fie $p = s..uv$ un drum optim de la s la v . Dacă $d[u] = \delta(s, u)$ la un moment dat, atunci începând cu momentul imediat următor relaxării arcului (u, v) avem $d[v] = \delta(s, v)$
- **Dem:**
 - Dacă înainte de relaxare $d[v] > d[u] + w(u, v)$, prin relaxare $\rightarrow d[v] = d[u] + w(u, v)$. Altfel, $d[v] \leq d[u] + w(u, v) \Rightarrow$ după relaxare avem $d[v] \leq d[u] + w(u, v)$.
 - Cum $d[u] = \delta(s, u)$ și relaxarea (u, v) nu modifică $d[u] \Rightarrow d[v] \leq d[u] + w(u, v) = \delta(s, u) + w(u, v) = \delta(s, v)$ (conf. **Corolar 25.2**) $\rightarrow d[v] = \delta(s, v)$

Algoritmul lui Dijkstra (II)

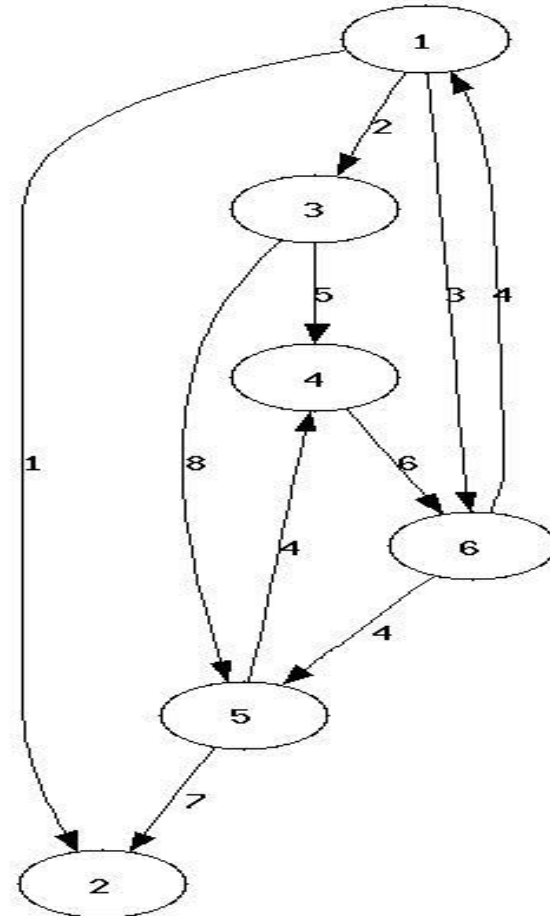
- Dijkstra(G,s)
 - Pentru fiecare ($u \in V$) // inițializări
 - $d[u] = \infty$; $p[u] = \text{null}$;
 - $d[s] = 0$;
 - $Q = \text{construiește_coada}(V)$ // coadă cu priorități
 - Cât timp ($Q \neq \emptyset$)
 - $u = \text{ExtrageMin}(Q)$; // extrage din V elementul cu $d[u]$ minim
 - // $Q = Q - \{u\}$ – se execută în cadrul lui ExtrageMin
 - Pentru fiecare ($v \in Q$ și v din succesorii lui u)
 - Dacă ($d[v] > d[u] + w(u,v)$)
 - $d[v] = d[u] + w(u,v)$ // actualizez distanța
 - $p[v] = u$ // și părintele

Exemplu (I)



Exemplu (II)

- $d[1] = 0$;
- (1): $d[2] = 1$; $d[3] = 2$; $d[6] = 3$;
- (2): $d[4] = 7$; $d[5] = 10$;
- (3): $d[5] = 7$;
- Dijkstra(G, s)
 - Pentru fiecare ($u \in V$)
 - $d[u] = \infty$; $p[u] = \text{null}$;
 - $d[s] = 0$;
 - $Q = \text{construiește_coada}(V)$ // coadă cu priorități
 - Cât timp ($Q \neq \emptyset$)
 - $u = \text{ExtrageMin}(Q)$; // extrage din V elementul cu $d[u]$ // minim
 - // $Q = Q - \{u\}$ – se execută în cadrul lui ExtrageMin
 - Pentru fiecare ($v \in Q$ și v din succesorii lui u)
 - Dacă ($d[v] > d[u] + w(u, v)$)
 - $d[v] = d[u] + w(u, v)$ // actualizez distanța
 - $p[v] = u$ // și părintele



Complexitate Dijkstra

- Depinde de ExtrageMin – coadă cu priorități.
- Operații ce trebuie realizate pe coadă + frecvența lor:
 - insert – V ;
 - delete – V ;
 - conține? – E ;
 - micșorează_val – E ;
 - este_vidă? – V .
- Dijkstra(G, s)
 - Pentru fiecare ($u \in V$)
 - $d[u] = \infty$; $p[u] = \text{null}$;
 - $d[s] = 0$;
 - $Q = \text{construieste_coada}(V)$ // coadă cu priorități
 - Cât timp ($Q \neq \emptyset$)
 - $u = \text{ExtrageMin}(Q)$; // extrage din V elementul cu $d[u]$ minim
 - // $Q = Q - \{u\}$ – se execută în cadrul lui ExtrageMin
 - Pentru fiecare ($v \in Q$ și v din succesorii lui u)
 - Dacă ($d[v] > d[u] + w(u, v)$)
 - $d[v] = d[u] + w(u, v)$ // actualizez distanța
 - $p[v] = u$ // și părintele

Implementare cu vectori

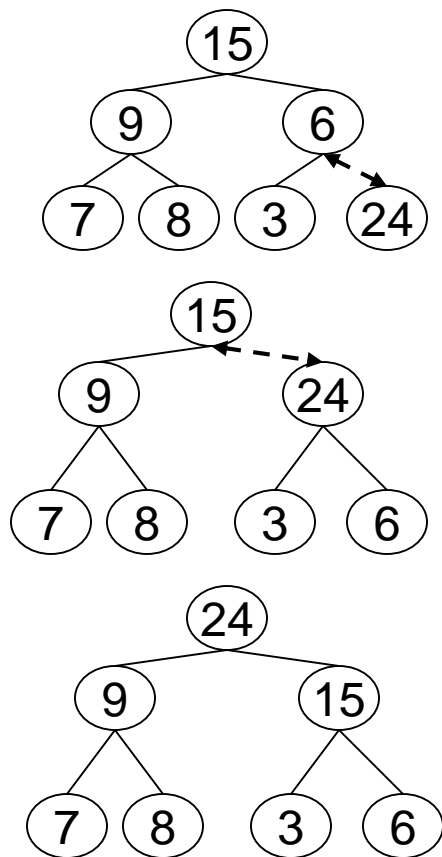
- Costuri:
 - insert – $1 * V = V$;
 - delete – $V * V = V^2$ (necesită căutarea minimului);
 - conține? – $1 * E = E$;
 - micșorează_val – $1 * E = E$;
 - este_vidă? – $1 * V = V$;
- Cea mai bună metodă pentru grafuri “dese” ($E \approx V^2$)!

Implementare cu heap binar

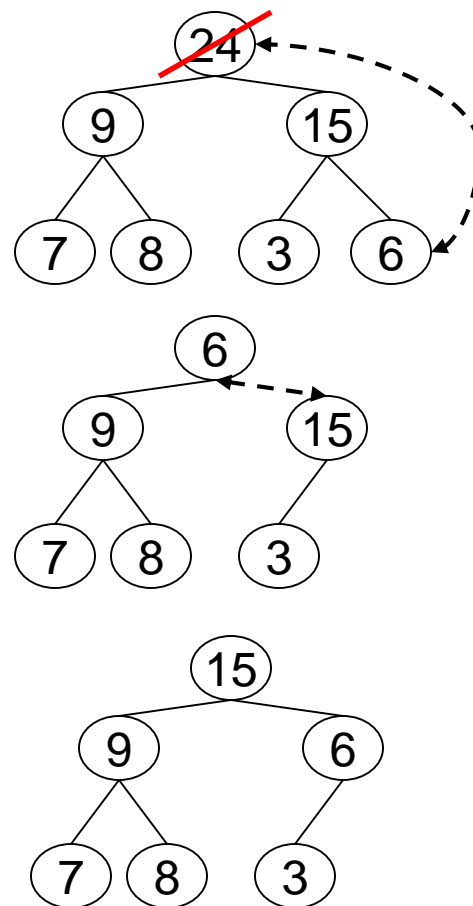
- Heap binar – structură de date de tip arbore binar + 2 constrângeri:
 - Fiecare nivel este complet; ultimul se umple de la stânga la dreapta;
 - $\forall u \in \text{Heap}; u \geq \text{răd}(\text{st}(u))$ și $u \geq \text{răd}(\text{dr}(u))$ (u este \geq decât ambii copii ai săi) unde \geq este o relație de ordine pe mulțimea pe care sunt definite elementele heapului.

Operatii pe Heap Binar

insert



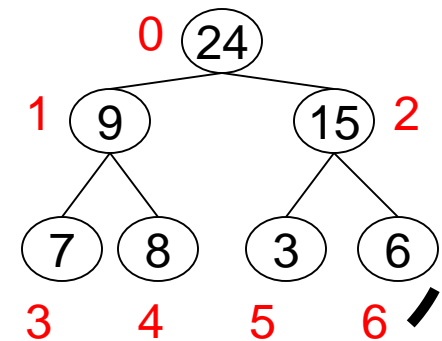
delete



Implementare Heap Binar

- Implementare folosind vectori.
- $Poziție[i]$ = unde se găsește în indexul de valori elementul de pe poziția i din heap.
- $Reverse[i]$ = unde se găsește în heap elementul de pe poziția i din valoare.
- Implementare disponibilă la [3].

Index	0	1	2	3	4	5	6
Valoare	7	6	15	8	24	9	3
Poziție							
Reverse							



Heap Binar

- Costuri:

- insert – $\log V * V = V \log V$;
- delete – $\log V * V = V \log V$;
- conține? – $1 * E = E$;
- micșorează_val – $\log V * E = E \log V$;
- este_vidă? – $1 * V = V$.

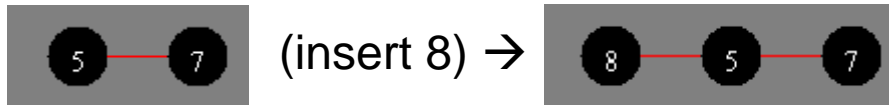
- Eficient dacă graful are arce puține comparativ cu numărul de noduri.

Heap Fibonacci

- Poate fi format din mai mulți arbori.
- Cheia unui părinte \leq cheia oricărui copil.
- Fiind dat un nod u și un heap H :
 - $p(u)$ – părintele lui u ;
 - $\text{copil}(u)$ – legătura către unul din copiii lui u ;
 - $\text{st}(u)$, $\text{dr}(u)$ – legătura la frații din stânga și din dreapta (cei de pe primul nivel sunt legați între ei astfel);
 - $\text{grad}(u)$ – numărul de copii ai lui u ;
 - $\text{min}(H)$ – cel mai mic nod din H ;
 - $n(H)$ – numărul de noduri din H .

Operatii Heap Fibonacci

- Inserare nod – $O(1)$
 - construiește un nou arbore cu un singur nod

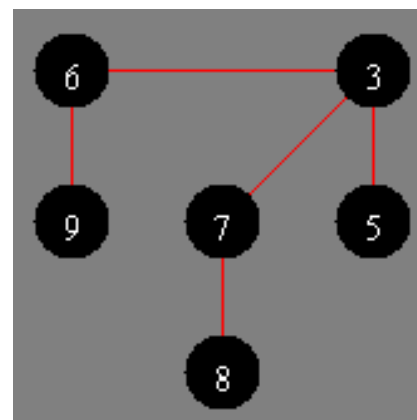
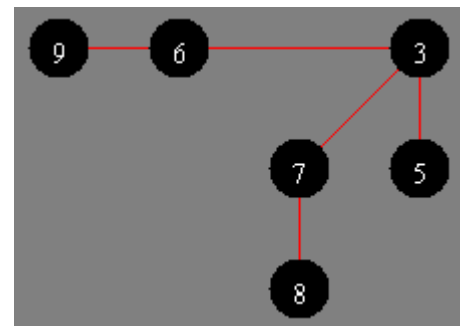
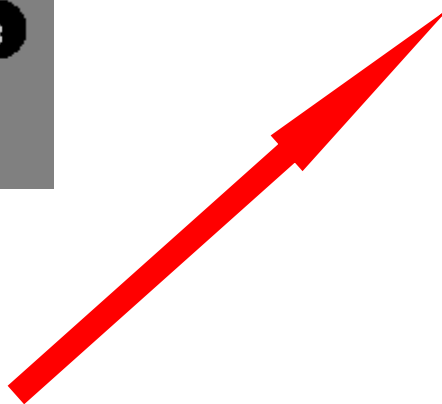
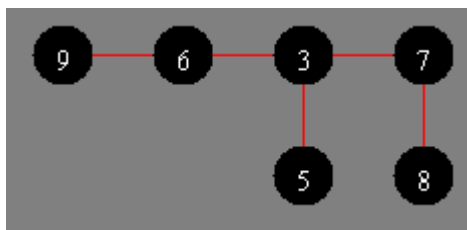
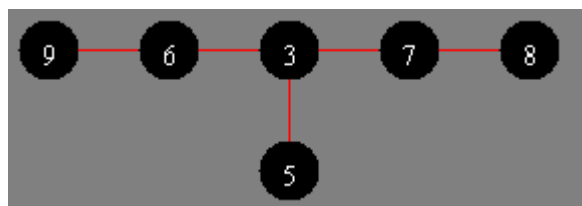


- Min – accesibil direct - $\min(H)$ – $O(1)$
- ExtrageMin $O(\log n)$ – **cost amortizat!**
 - Mută copiii minimului pe prima coloană;
 - **Consolidează** heap-ul.

Operatii Heap Fibonacci

- Consolidare Heap
 - Cât timp există 2 arbori cu grade egale $\text{Arb}(x)$ și $\text{Arb}(y)$, $x < y$:
 - $\text{Arb}(y)$ transformat în copil al lui x ;
 - $\text{grad}[x] ++$;
- Applet și implementare disponibile la [4].
- Gif + implementare puteți găsi la [5]

Consolidare Heap



Costuri Heap Fibonacci

- Costuri:
 - insert – $1 * V = V$;
 - delete – $\log V * V = V \log V$ (amortizat!);
 - micșorează_val – $1 * E = E$;
 - este_vidă? – $1 * V = V$.
- Cea mai rapidă structură dpdv teoretic.

Concluzii Dijkstra

- Implementarea trebuie realizată în funcție de tipul grafului pe care lucrăm:
 - vectori pentru grafuri “dese” - $O(V^2)$;
 - heap pentru grafuri “rare”: HB - $O(E \log V)$, HF - $O(V \log V + E)$
- Heapul Fibonacci este mai eficient decât heapul binar dar mai dificil de implementat.

ÎNTREBĂRI?