Reprezentări ale grafurilor.
Parcurgeri în grafuri

Reprezentări

Reprezentări

Ce reprezentări ale grafurilor cunoașteți?

Reprezentări

Ce reprezentări ale grafurilor cunoașteți?

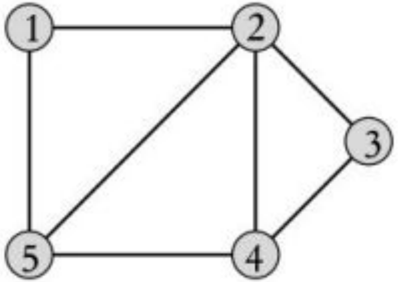
- matrice de adiacență
- liste de adiacență
- lista de muchii

Reprezentări

Ce reprezentări ale grafurilor cunoașteți?

- matrice de adiacență
  - într-un graf **orientat**, matricea în general nu este simetrică

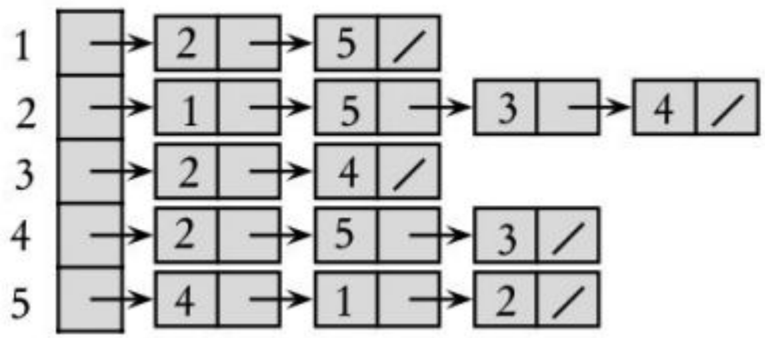
$$\begin{array}{c|ccccc|} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 1 & 1 & 1 \\ 3 & 0 & 1 & 0 & 1 & 0 \\ 4 & 0 & 1 & 1 & 0 & 1 \\ 5 & 1 & 1 & 0 & 1 & 0 \\ \hline \end{array}$$

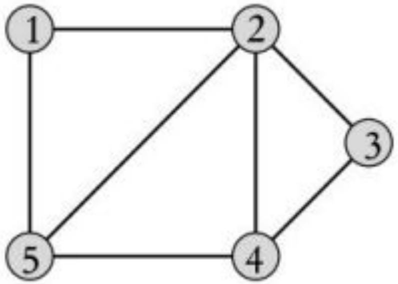


Reprezentări

Ce reprezentări ale grafurilor cunoașteți?

- liste de adiacență

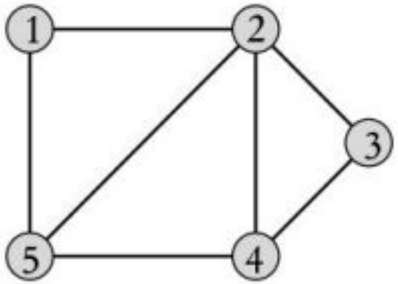




Reprezentări

Ce reprezentări ale grafurilor cunoașteți?

- liste de muchii
  - [(1,2), (1,5), (2,5), (2,4), (2,3), (3,4), (4,5)]



Reprezentări

De ce avem mai multe reprezentări?

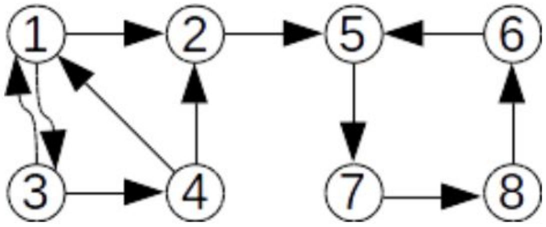
- Modul cum reprezentăm influențează complexitatea timp a algoritmilor implementați
  - Exemplu DF:
    - $O(n^2)$ matrice de adiacență
    - $O(n+m)$ lista de vecini
    - $O(n*m)$ lista de muchii
- Student:
  - În funcție de graful reprezentat, unele reprezentări consumă mai multă, respectiv mai puțină memorie
  - Memorie consumată:
    - Matrice de adiacență $O(n^2)$
    - Lista de vecini: $O(n+m)$
    - Lista de muchii: $O(m)$
  - Dacă avem graf dens (aproape complet), matricea de adiacență nu va mai fi rea … pentru că $m \sim n^2$

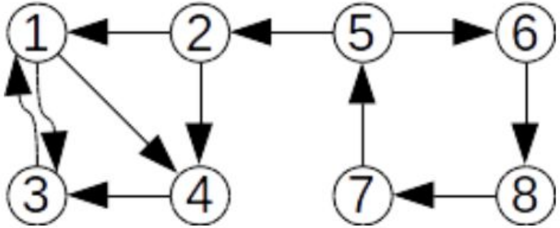
Reprezentări

Graful Transpus

Fie G = (V, E) un graf **orientat**. Se numește **graf transpus** al lui G graful orientat $G^T = (V, E^T)$ având aceeași mulțime de noduri, iar $E^T = \{(y, x) \mid (x, y) \in E\}$.

Practic, sensul arcelor se schimbă, pentru toate arcele din graful G.





Parcurgeri

Parcurgerea în lățime (BFS)

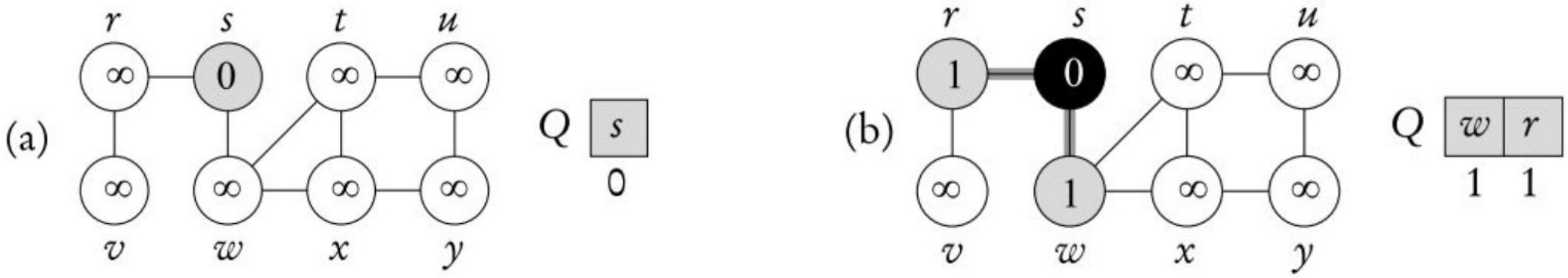
Dat fiind un graf G = (V, E) şi un nod sursă s, **căutarea în lăţime** explorează sistematic muchiile lui G pentru a “descoperi” fiecare nod care este accesibil din s.

De asemenea, algoritmul găsește distanța minimă de la sursa s la toate nodurile din graf.

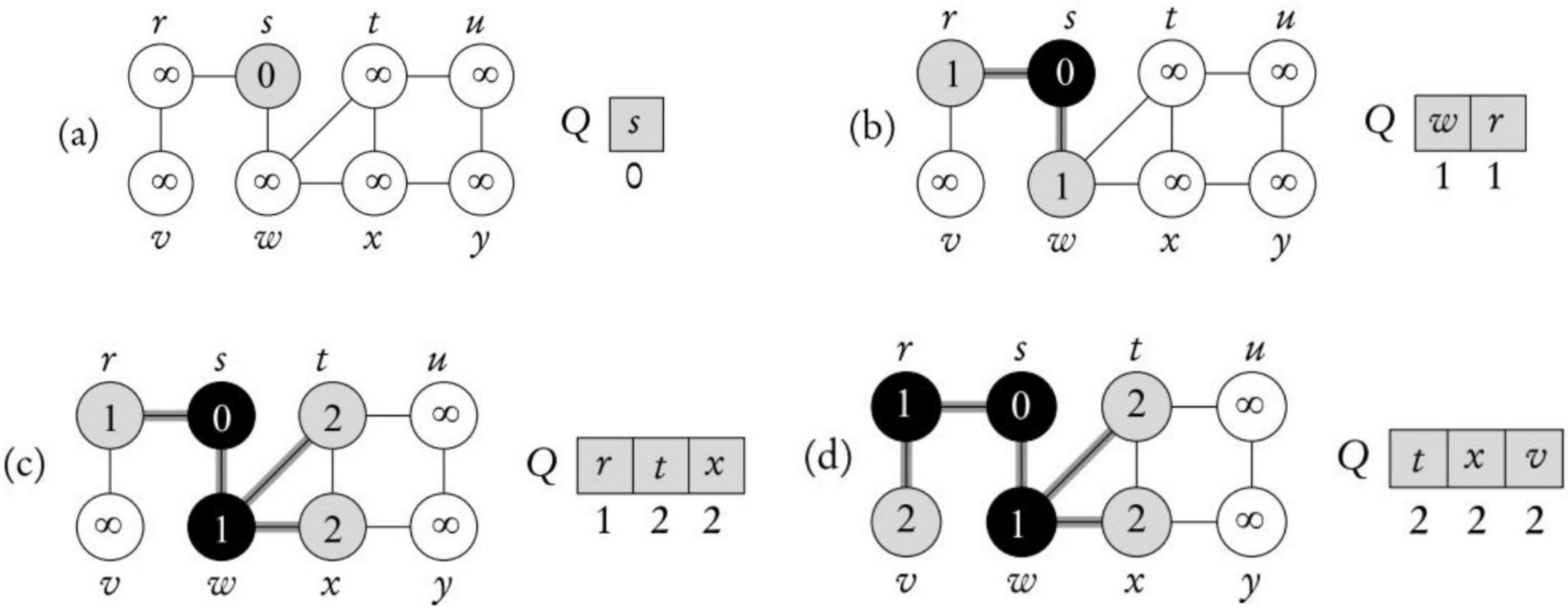
Parcurgerea în lățime (BFS)

1. Se începe explorarea dintr-un nod nevizitat, care se adaugă într-o coadă
2. Cât timp există elemente în coadă
    a. Se scoate din coadă
    b. Se pun în coadă toți vecinii nevizitați

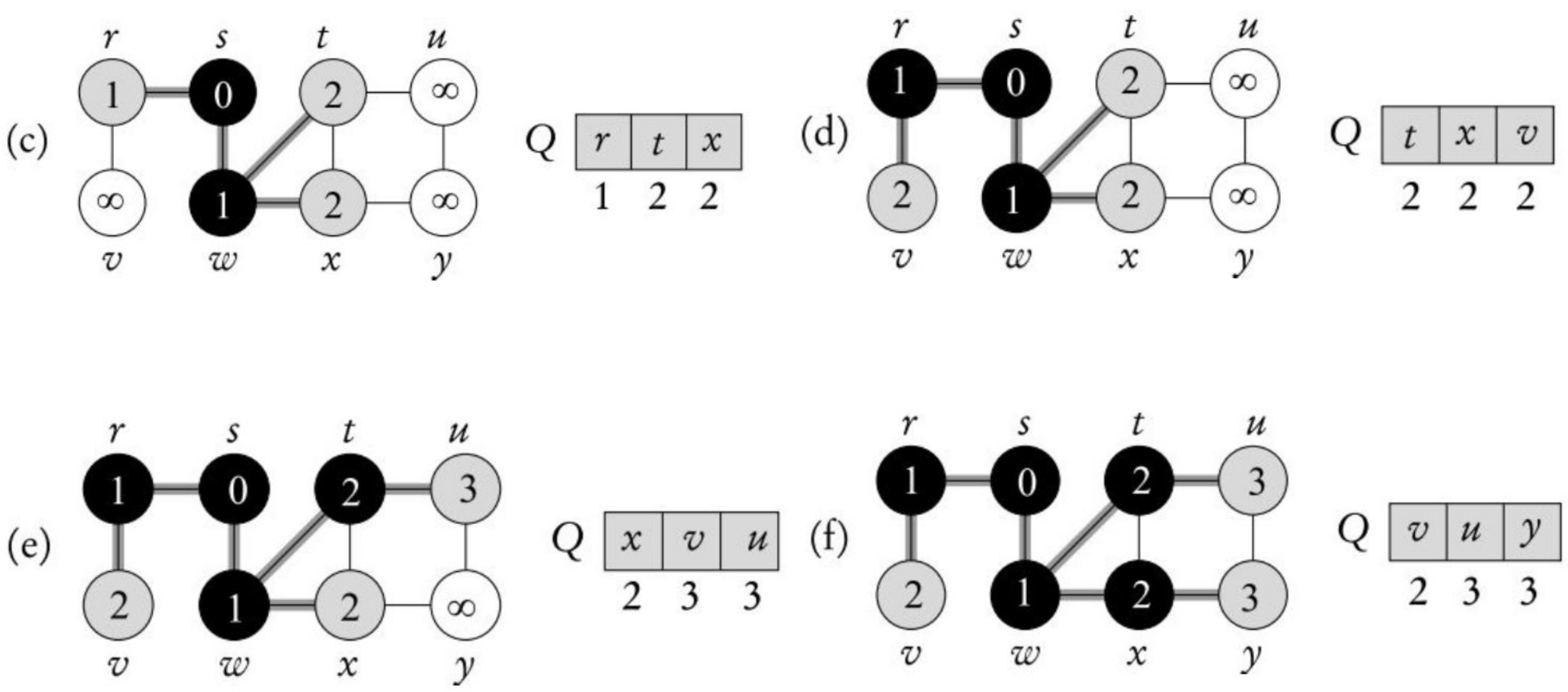
Parcurgerea în lățime – Exemplu



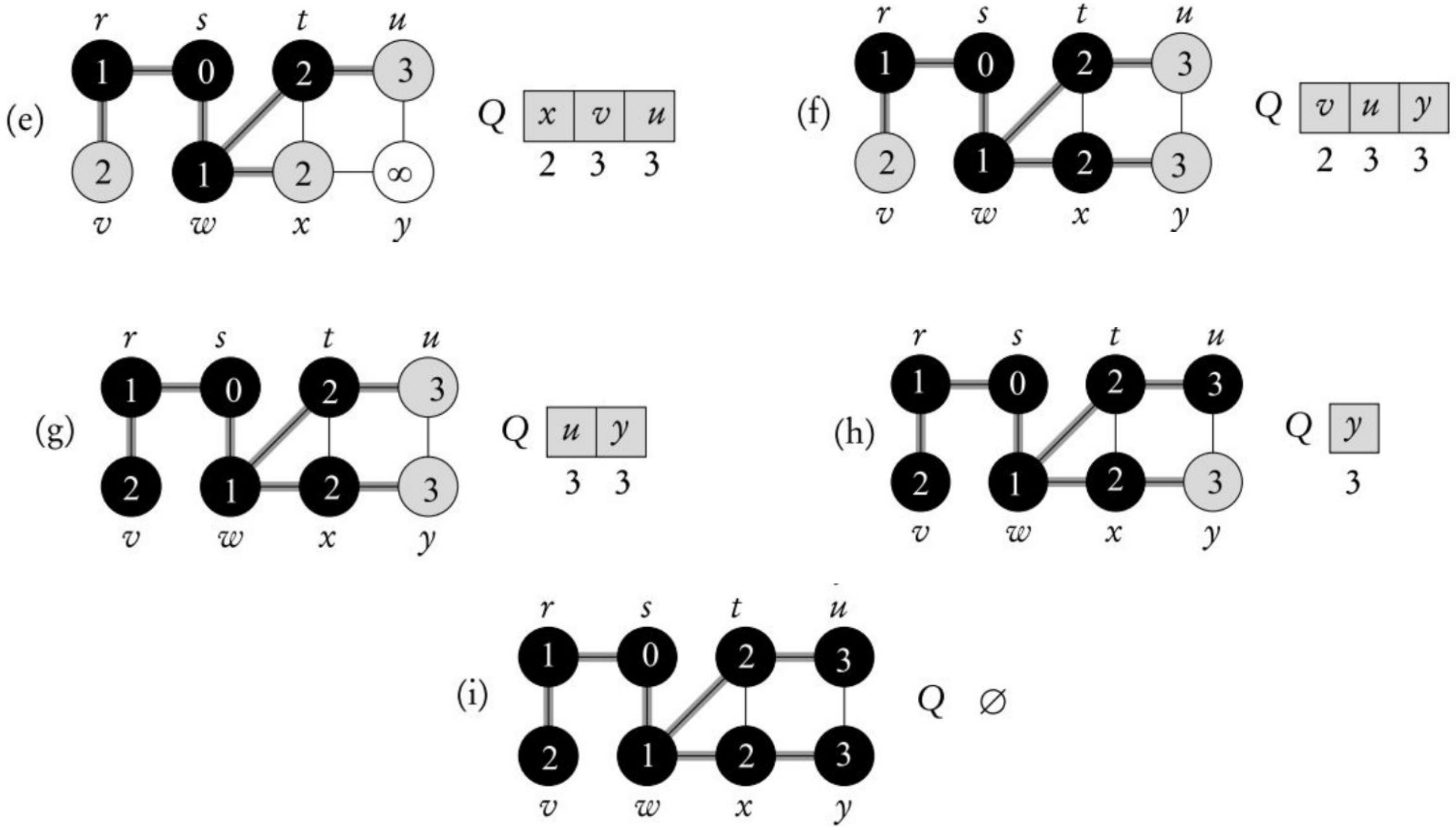
Parcurgerea în lățime – Exemplu



Parcurgerea în lățime – Exemplu



Parcurgerea în lățime - Exemplu



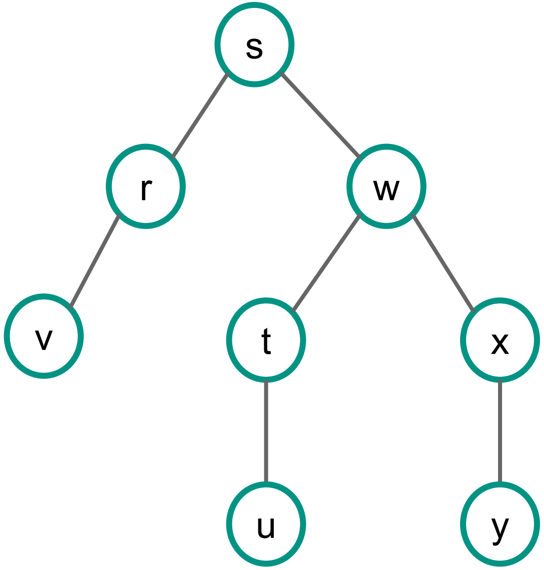
Parcurgerea în lățime - Algoritm

Să scriem împreună pseudocodul:

Parcurgerea în lățime – Algoritm

Algoritm Cormen:

Pi → tata



$\mathrm{CL}(G, s)$

pentru fiecare vârf $u \in V[G] - \{s\}$ **execută**
    $color[u] \leftarrow$ ALB
    $d[u] \leftarrow \infty$
    $\pi[u] \leftarrow$ NIL
$color[s] \leftarrow$ GRI
$d[s] \leftarrow 0$
$\pi[s] \leftarrow$ NIL
$Q \leftarrow \{s\}$
cât timp $Q \neq \emptyset$ **execută**
    $u \leftarrow cap[Q]$
    pentru fiecare vârf $v \in Adj[u]$ **execută**
        dacă $color[v] =$ ALB **atunci**
            $color[v] \leftarrow$ GRI
            $d[v] \leftarrow d[u] + 1$
            $\pi[v] \leftarrow u$
            PUNE-ÎN-COADĂ(Q, v)
    SCOATE-DIN-COADĂ(Q)
    $color[u] \leftarrow$ NEGRU

Parcurgerea în lățime - Complexitate

Care este complexitatea algoritmului?

Parcurgerea în lățime - Complexitate

Care este complexitatea algoritmului?

- O(V+E) sau O(n+m)
  - unde
    - V = n = numărul de noduri
    - E = m = numărul de muchii

Parcurgerea în lățime – Aplicații

Ieșirea din labirint într-un număr minim de pași

0 0 → punct de pornire

0 7 → punct de ieșire

```
0  0  0  0  0  0 -1 0
0 -1 -1 -1 -1 -1 -1 0
0  0  0  0  0  0 -1 0
-1 -1 -1 -1 -1 0 -1 0
0   0  0  0 -1 0  0 0
```

Parcurgerea în lățime - Aplicații

Ieșirea din labirint într-un număr minim de pași

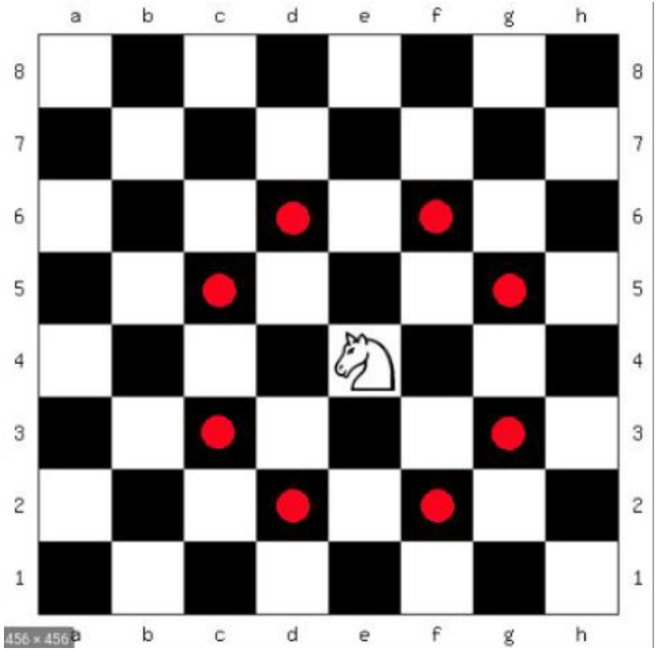
0 0 → punct de pornire

0 7 → punct de ieșire

```
0  1  2  3  4  5 -1 15

1 -1 -1 -1 -1 -1 -1 14

2  3  4  5  6  7 -1 13

-1 -1 -1 -1 -1 8 -1 12

0  0  0  0 -1 9  10 11
```

Parcurgerea în lățime – Aplicații

Drum de lungime minimă a calului pe tabla de șah



Parcurgerea în adâncime (DFS)

1. Se începe explorarea dintr-un nod nevizitat
2. Se caută un vecin nevizitat, care devine nodul curent.
   Cât timp nodul curent are un vecin nevizitat, repetăm pasul 2 (intrăm în adâncime)
3. Când nodul curent nu are niciun vecin nevizitat, ne întoarcem la strămoșii lui (tatăl, bunicul etc), până găsim un nod care are vecini nevizitați și reluăm pasul 2
4. Dacă există noduri nevizitate, reluăm pasul 1

Când se întâmplă pasul 4?

Parcurgerea în adâncime (DFS)

1. Se începe explorarea dintr-un nod nevizitat
2. Se caută un vecin nevizitat, care devine nodul curent.
   Cât timp nodul curent are un vecin nevizitat, repetăm pasul 2 (intrăm în adâncime)
3. Când nodul curent nu are niciun vecin nevizitat, ne întoarcem la strămoșii lui (tatăl, bunicul etc), până găsim un nod care are vecini nevizitați și reluăm pasul 2
4. Dacă există noduri nevizitate, reluăm pasul 1

Când se întâmplă pasul 4?

- Când graful are mai multe componente conexe

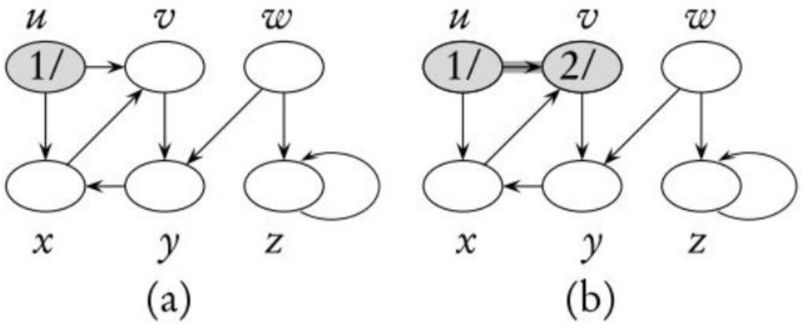
Parcurgerea în adâncime - Cu timpi de intrare și ieșire

Pentru o parcurgere în adâncime, este uneori util să ținem minte cronologia parcurgerii.

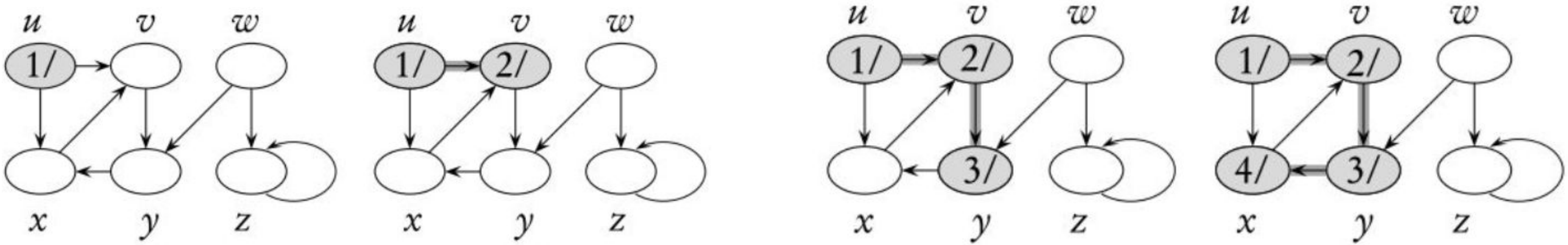
- De fiecare dată când ajungem într-un nod sau când terminăm de vizitat toți vecinii, incrementăm un contor și ținem minte aceste informații

Observație: O să existe situații în care cronologia va fi un pic diferită, cum s-a întâmplat la RMQ → LCA

Parcurgerea în adâncime - Exemplu

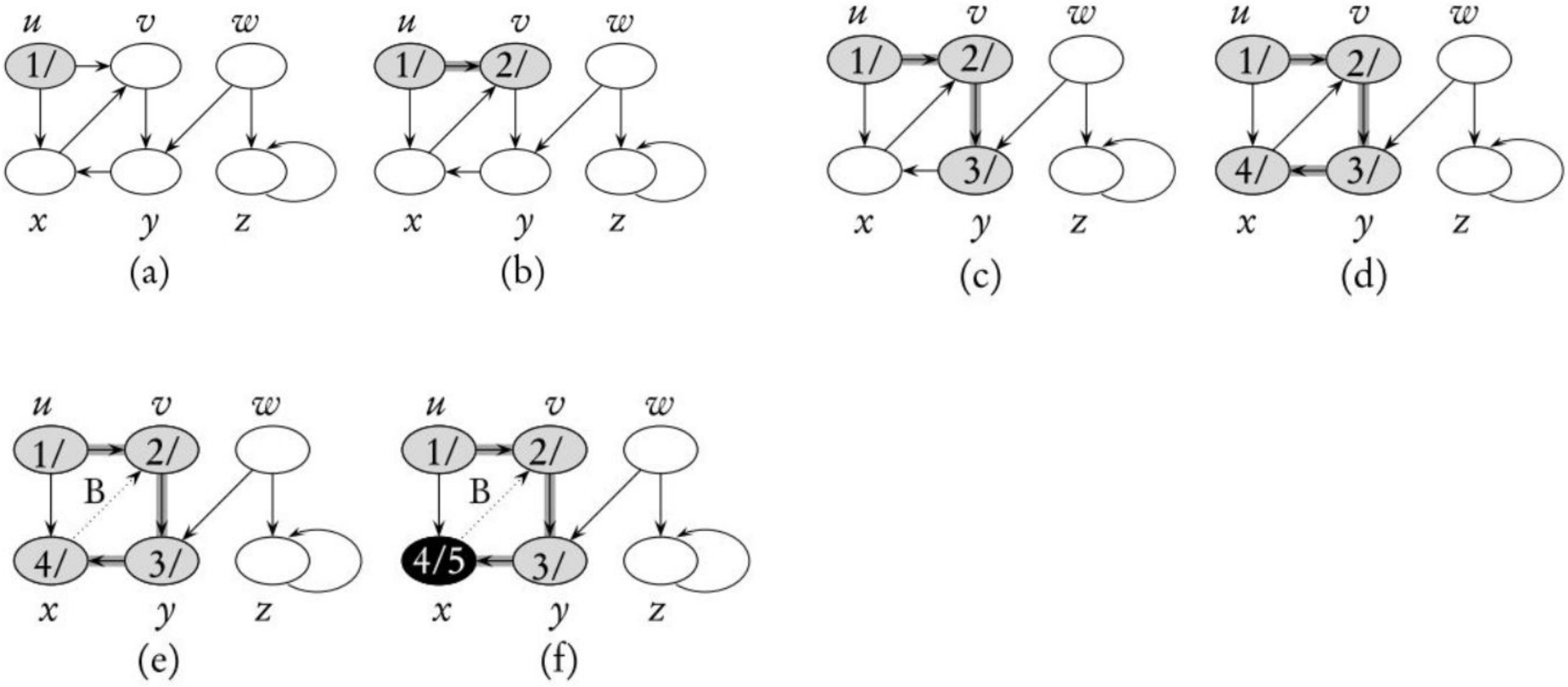


Parcurgerea în adâncime - Exemplu

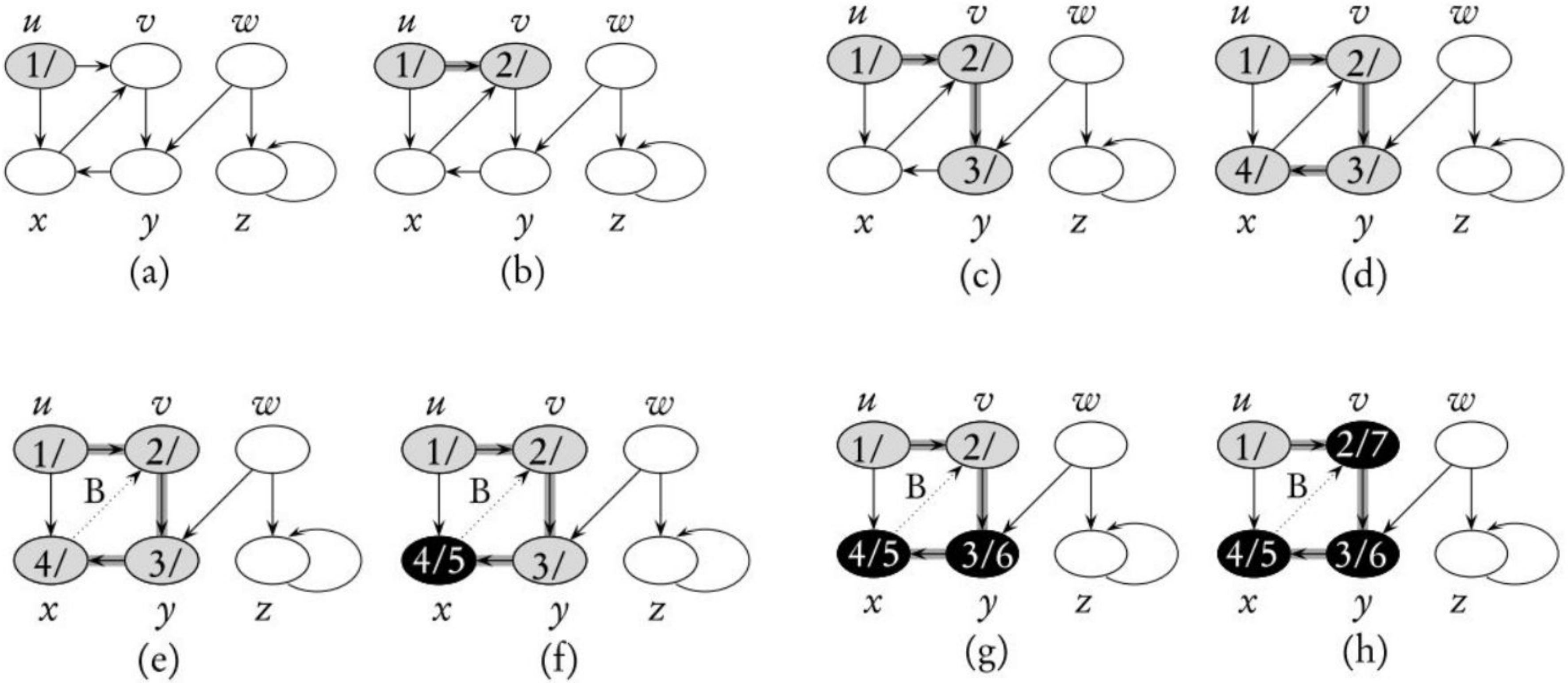


(a) (b) (c) (d)

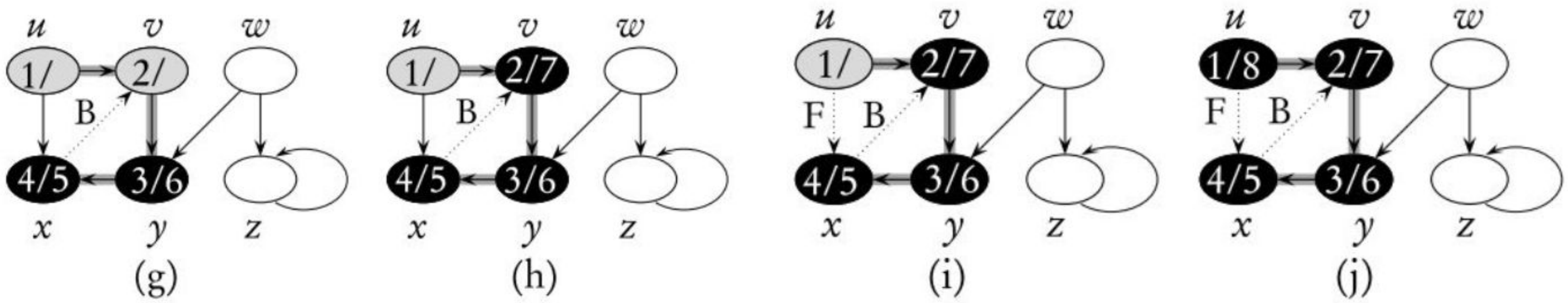
Parcurgerea în adâncime - Exemplu



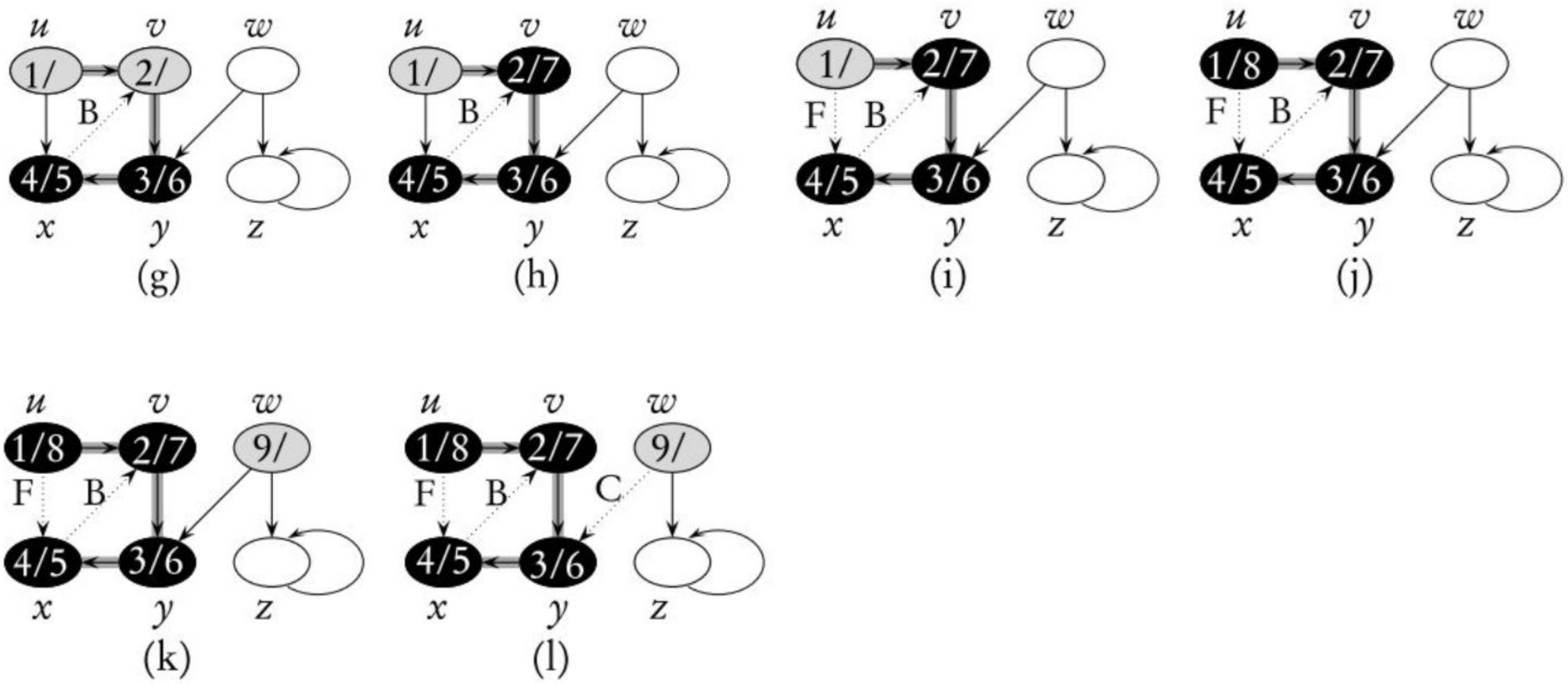
Parcurgerea în adâncime - Exemplu



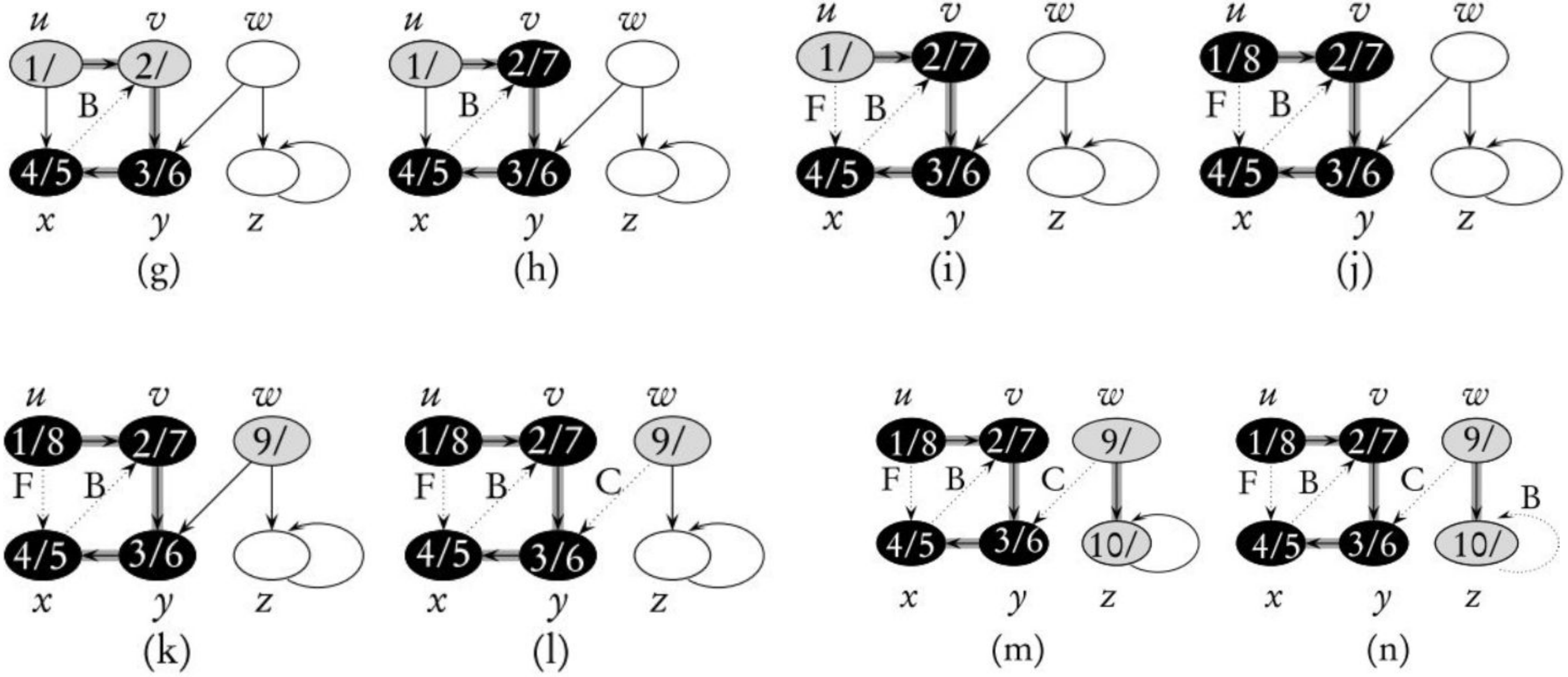
Parcurgerea în adâncime - Exemplu



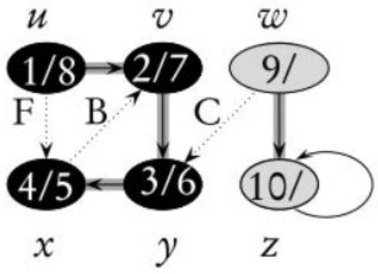
Parcurgerea în adâncime - Exemplu



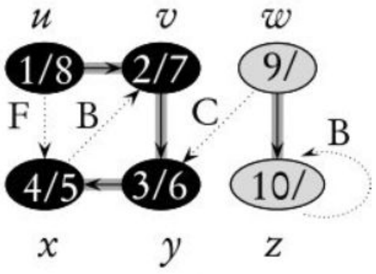
Parcurgerea în adâncime - Exemplu



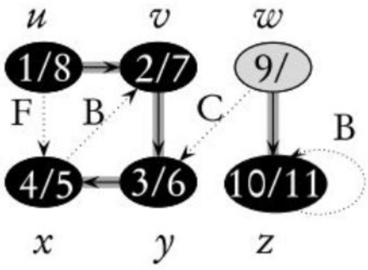
Parcurgerea în adâncime - Exemplu



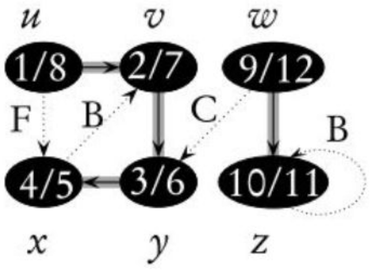
(m)



(n)



(o)



(p)

Parcurgerea în adâncime - Algoritm

Să scriem împreună pseudocodul:

Parcurgerea în adâncime - Algoritm

Algoritm Cormen:

CA(G)

pentru fiecare vârf $u \in V[G]$ **execută**
    $culoare[u] \leftarrow$ ALB
    $\pi[u] \leftarrow$ NIL
$timp \leftarrow 0$
pentru fiecare vârf $u \in V[G]$ **execută**
    dacă $culoare[u] =$ ALB **atunci**
        CA-VIZITĂ(u)

CA-VIZITĂ(u)

$culoare[u] \leftarrow$ GRI ▷ Vârful alb u tocmai a fost descoperit.
$d[u] \leftarrow timp \leftarrow timp + 1$
pentru fiecare $v \in Adj[u]$ **execută** ▷ Explorează muchia (u, v).
    dacă $culoare[v] =$ALB **atunci**
        $\pi[v] \leftarrow u$
        CA-VIZITĂ(v)
$culoare[u] \leftarrow$ NEGRU ▷ Vârful u este colorat în negru. El este terminat.
$f[u] \leftarrow timp \leftarrow timp + 1$

Parcurgerea în adâncime - Complexitate

Care este complexitatea algoritmului?

Parcurgerea în adâncime - Complexitate

Care este complexitatea algoritmului?

- O(V+E) sau O(n+m)
  - unde
    - V = n = numărul de noduri
    - E = m = numărul de muchii

Parcurgerea în adâncime - Teorema parantezelor

Teorema parantezelor

În orice căutare în adâncime a unui graf (orientat sau neorientat) G = (V, E), pentru oricare două vârfuri u şi v, exact una din următoarele trei condiţii este adevărată:

- intervalele [d[u], f[u]] şi [d[v], f[v]] sunt total disjuncte
- intervalul [d[u], f[u]] este conţinut, în întregime, în intervalul [d[v], f[v]], iar u este un descendent al lui v în arborele de adâncime
- intervalul [d[v], f[v]] este conţinut, în întregime, în intervalul [d[u], f[u]], iar v este un descendent al lui u în arborele de adâncime

Parcurgerea în adâncime – Teorema parantezelor

Teorema parantezelor: Demonstrație

Începem cu cazul în care d[u] < d[v].
În funcţie de valoarea de adevăr a inegalităţii d[v] < f[u], există două subcazuri care trebuie considerate.

1. **În primul subcaz:** d[v] < f[u], deci v a fost descoperit, în timp ce u era încă gri. Aceasta implică faptul că v este un descendent al lui u. Mai mult, deoarece v a fost descoperit înaintea lui u, toate muchiile care pleacă din el sunt explorate, iar v este terminat, înainte ca algoritmul să revină pentru a-l termina pe u. De aceea, în acest caz, intervalul [d[v], f[v]] este conţinut în întregime în intervalul [d[u], f[u]].
2. **În celălalt subcaz:** f[u] < d[v] şi din inegalitatea d[u] < f[u] implică faptul că intervalele [d[u], f[u]] şi [d[v], f[v]] sunt disjuncte.

Cazul în care d[v] < d[u] este similar, inversând rolurile lui u şi v în argumentaţia de mai sus.

Parcurgerea în adâncime - Teorema parantezelor

Corolarul 23.7 (Interclasarea intervalelor descendenților)

Vârful v este un descendent al lui u în pădurea de adâncime pentru un graf G orientat sau neorientat **dacă şi numai dacă** $d[u] < d[v] < f[v] < f[u]$.

Parcurgerea în adâncime - Teorema parantezelor

(a)

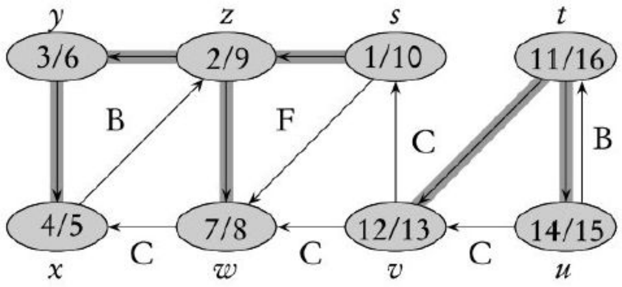
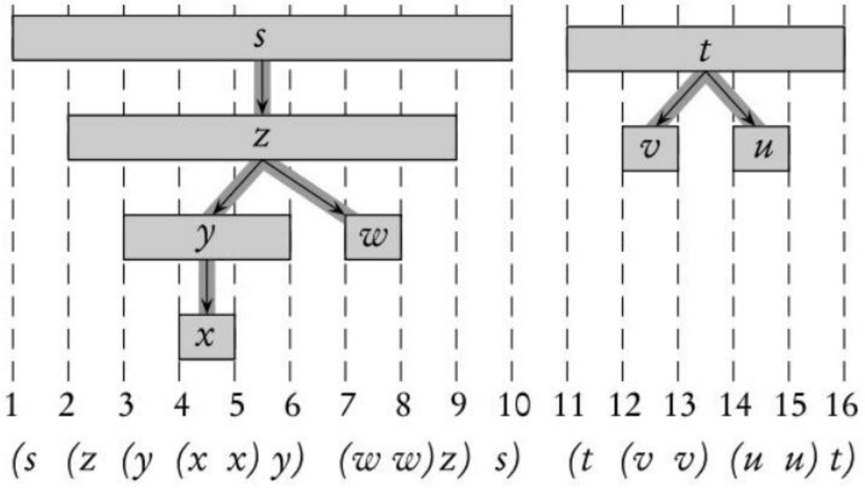


Diagrama b exemplifică foarte bine teorema anterioară.

(b)



Parcurgerea în adâncime - Teorema parantezelor

(a)

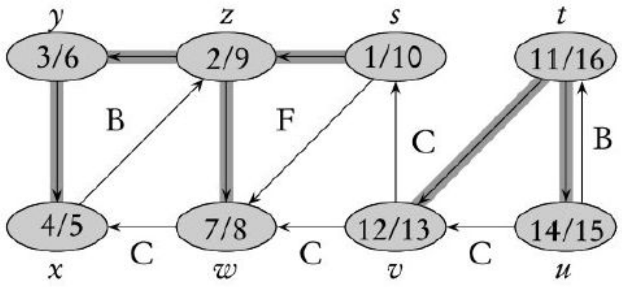
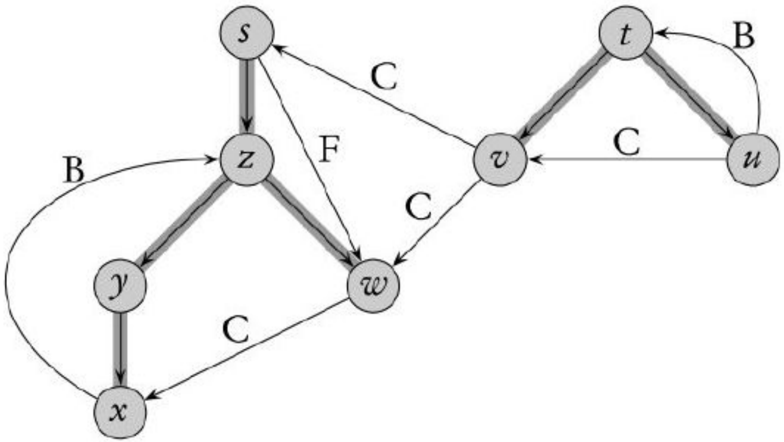


Diagrama c exemplifică muchiile de întoarcere despre care vom vorbi imediat.

(c)



Parcurgerea în adâncime - Clasificarea muchiilor

Clasificarea muchiilor

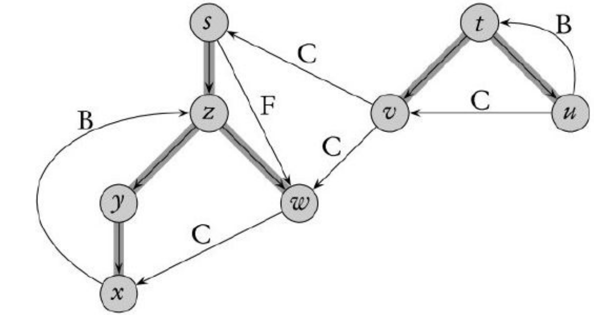
1. **Muchiile de arbore** sunt muchii din pădurea de adâncime G_π. Muchia (u, v) este o muchie de arbore dacă v a fost descoperit explorând muchia (u, v).
2. **Muchiile înapoi** sunt acele muchii (u, v) care unesc un vârf u cu un strămoş v într-un arbore de adâncime. Buclele (muchii de la un vârf la el însuşi) care pot apărea într-un graf orientat sunt considerate muchii înapoi.
3. **Muchiile înainte** sunt acele muchii (u, v) ce nu sunt muchii de arbore şi conectează un vârf u cu un descendent v într-un arbore de adâncime.
4. **Muchiile transversale** sunt toate celelalte muchii. Ele pot uni vârfuri din acelaşi arbore de adâncime, cu condiţia ca unul să nu fie strămoşul celuilalt, sau pot uni vârfuri din arbori de adâncime diferiţi.

Parcurgerea în adâncime - Clasificarea muchiilor

Clasificarea muchiilor

1. **Muchiile de arbore** sunt muchii din pădurea de adâncime Gπ. Muchia (u, v) este o muchie de arbore dacă v a fost descoperit explorând muchia (u, v).
2. **Muchiile înapoi** sunt acele muchii (u, v) care unesc un vârf u cu un strămoş v într-un arbore de adâncime. Buclele (muchii de la un vârf la el însuşi) care pot apărea într-un graf orientat sunt considerate muchii înapoi.
3. **Muchiile înainte** sunt acele muchii (u, v) ce nu sunt muchii de arbore şi conectează un vârf u cu un descendent v într-un arbore de adâncime.
4. **Muchiile transversale** sunt toate celelalte muchii. Ele pot uni vârfuri din acelaşi arbore de adâncime, cu condiţia ca unul să nu fie strămoşul celuilalt, sau pot uni vârfuri din arbori de adâncime diferiţi.

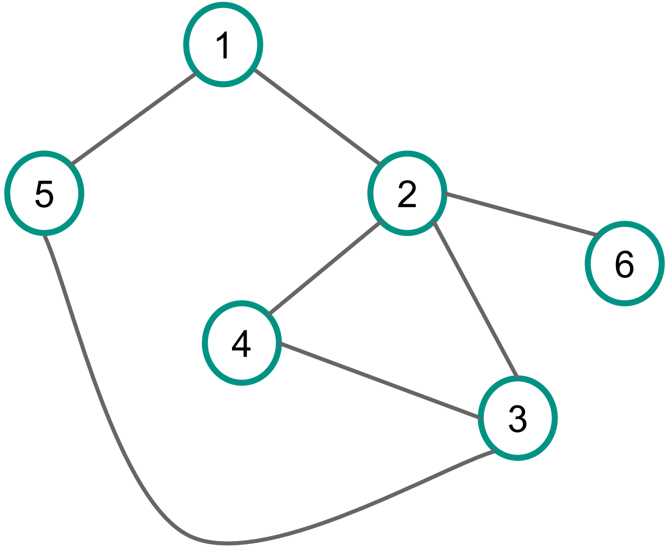
(c)



Parcurgerea în adâncime - Clasificarea muchiilor

Într-un graf neorientat nu vom avea toate cele 4 categorii de muchii.

Ce categorii vom avea?

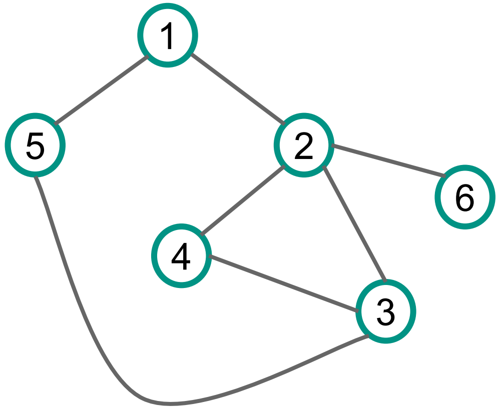


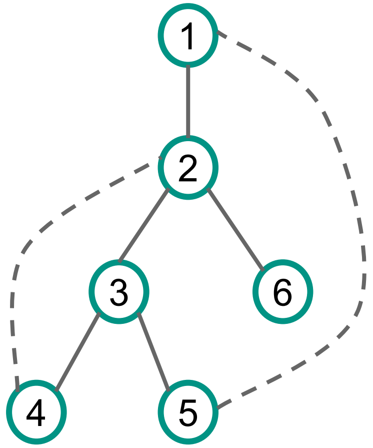
Parcurgerea în adâncime - Clasificarea muchiilor

Într-un graf neorientat nu vom avea toate cele 4 categorii de muchii.

Ce categorii vom avea?

- doar primele două categorii (muchii de arbore și muchii înapoi)





Parcurgerea în adâncime - Clasificarea muchiilor

Într-un graf neorientat vom avea un ciclu dacă găsim ce fel de muchie?

Parcurgerea în adâncime - Clasificarea muchiilor

Într-un graf neorientat vom avea un ciclu dacă găsim ce fel de muchie?

- muchie înapoi