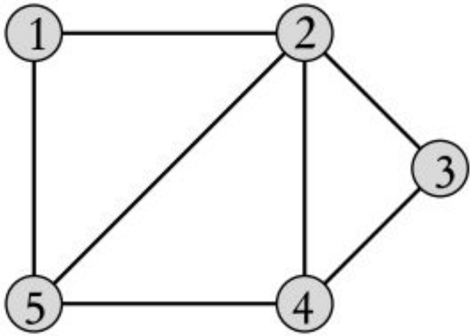
Parcurgeri în Grafuri

Reprezentari

Ce reprezentări cunoasteti ?

Reprezentari



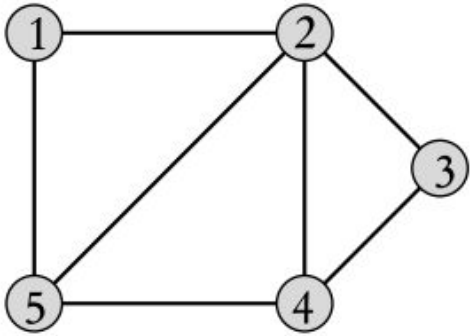
Ce reprezentări cunoasteti ?

- Reprezentare cu matrice de adiacență
- Liste de adiacenta
- Lista de muchii

Reprezentari

Ce reprezentări cunoasteti ?

- Reprezentare cu matrice de adiacență
    - În graf neorientat matricea nu mai este simetrica

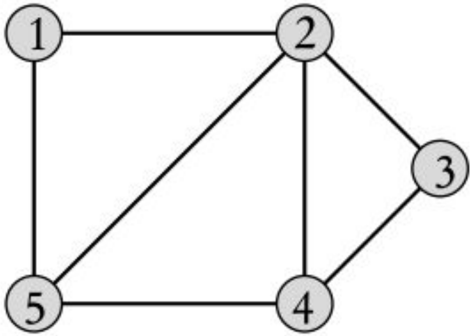


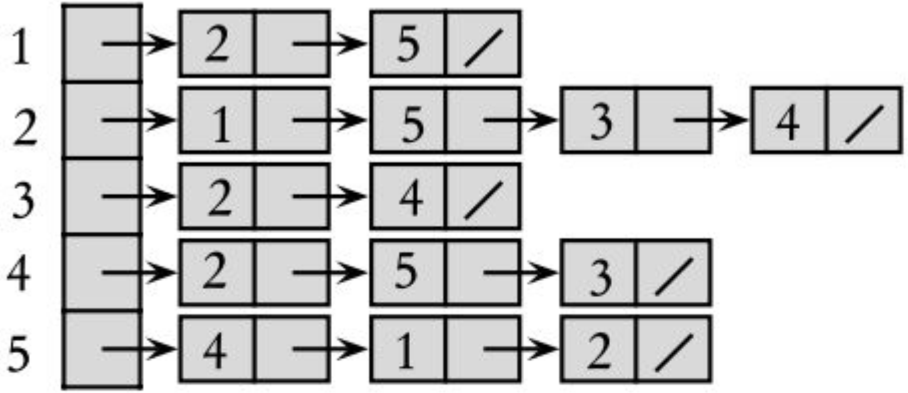
$$
\begin{array}{c|ccccc|}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 0 & 1 & 0 & 0 & 1 \\
2 & 1 & 0 & 1 & 1 & 1 \\
3 & 0 & 1 & 0 & 1 & 0 \\
4 & 0 & 1 & 1 & 0 & 1 \\
5 & 1 & 1 & 0 & 1 & 0 \\
\hline
\end{array}
$$

Reprezentari

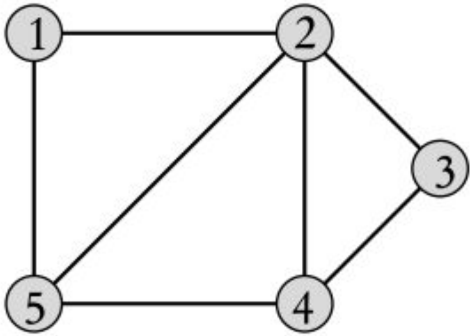
Ce reprezentări cunoasteti ?

- Liste de adiacenta





Reprezentari

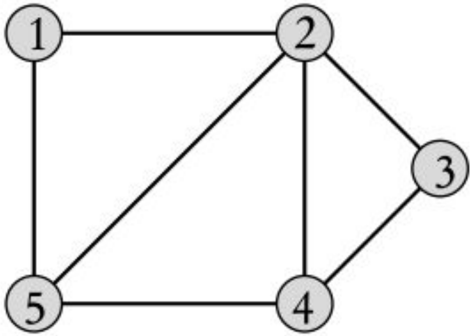


Ce reprezentări cunoasteti ?

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

Reprezentari

Ce reprezentări cunoasteti ?



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

Reprezentari

- De ce avem mai multe reprezentări ?

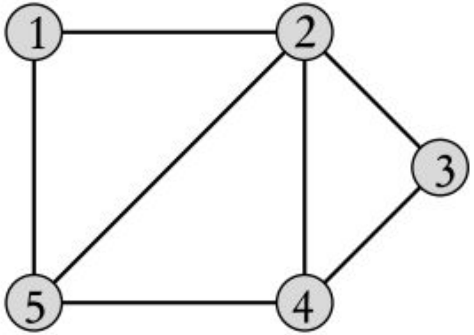
Reprezentari

Graful Transpus

-

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

0	0	1	1	0
0	0	0	0	0
1	0	0	0	1
1	0	0	0	0
0	0	1	0	0

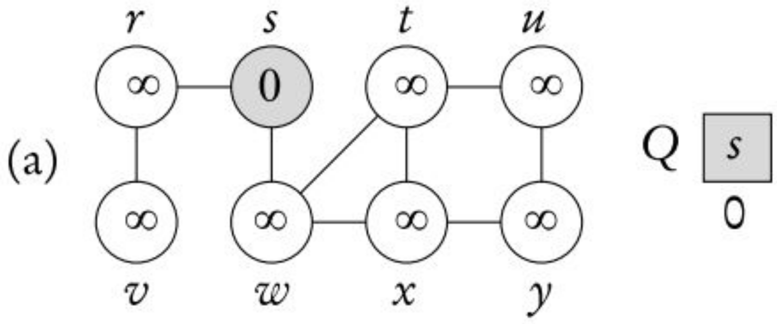


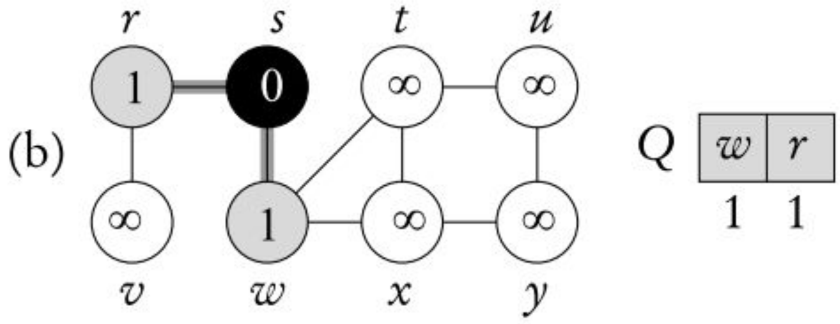
Parcurgerea în lățime

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.

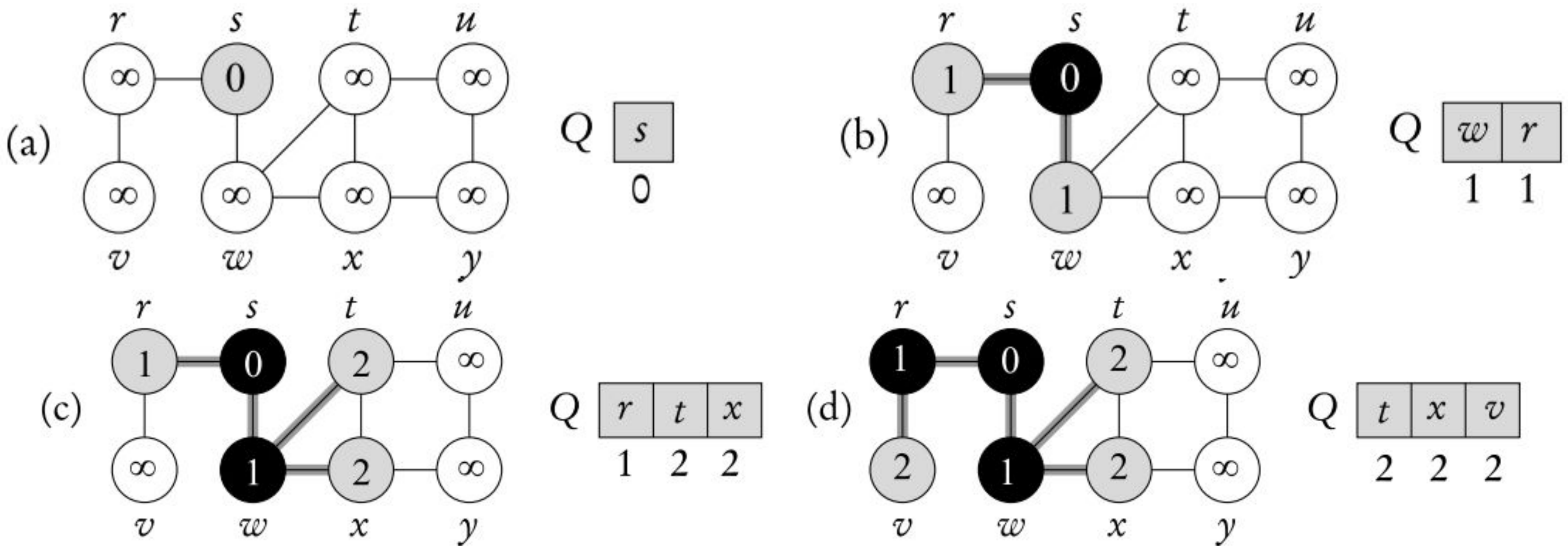
De asemenea algoritmul găsește distanța minima de la sursa la toate nodurile din graf.

Parcurgerea în lățime : Exemplu

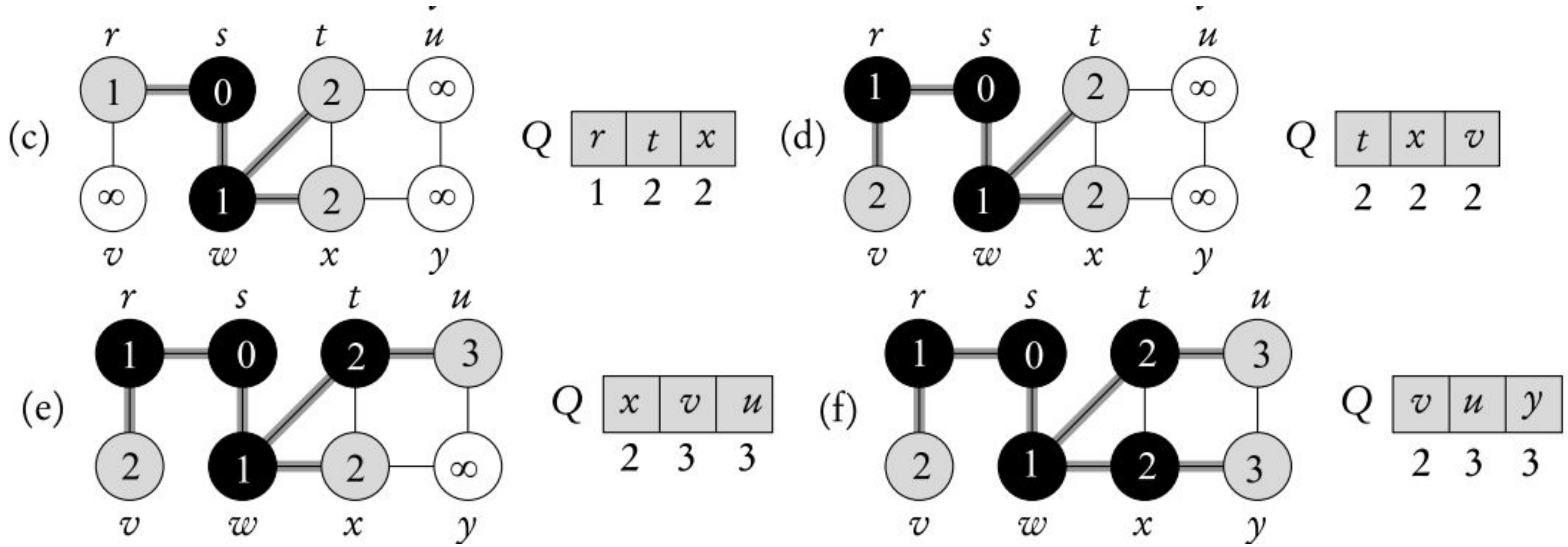




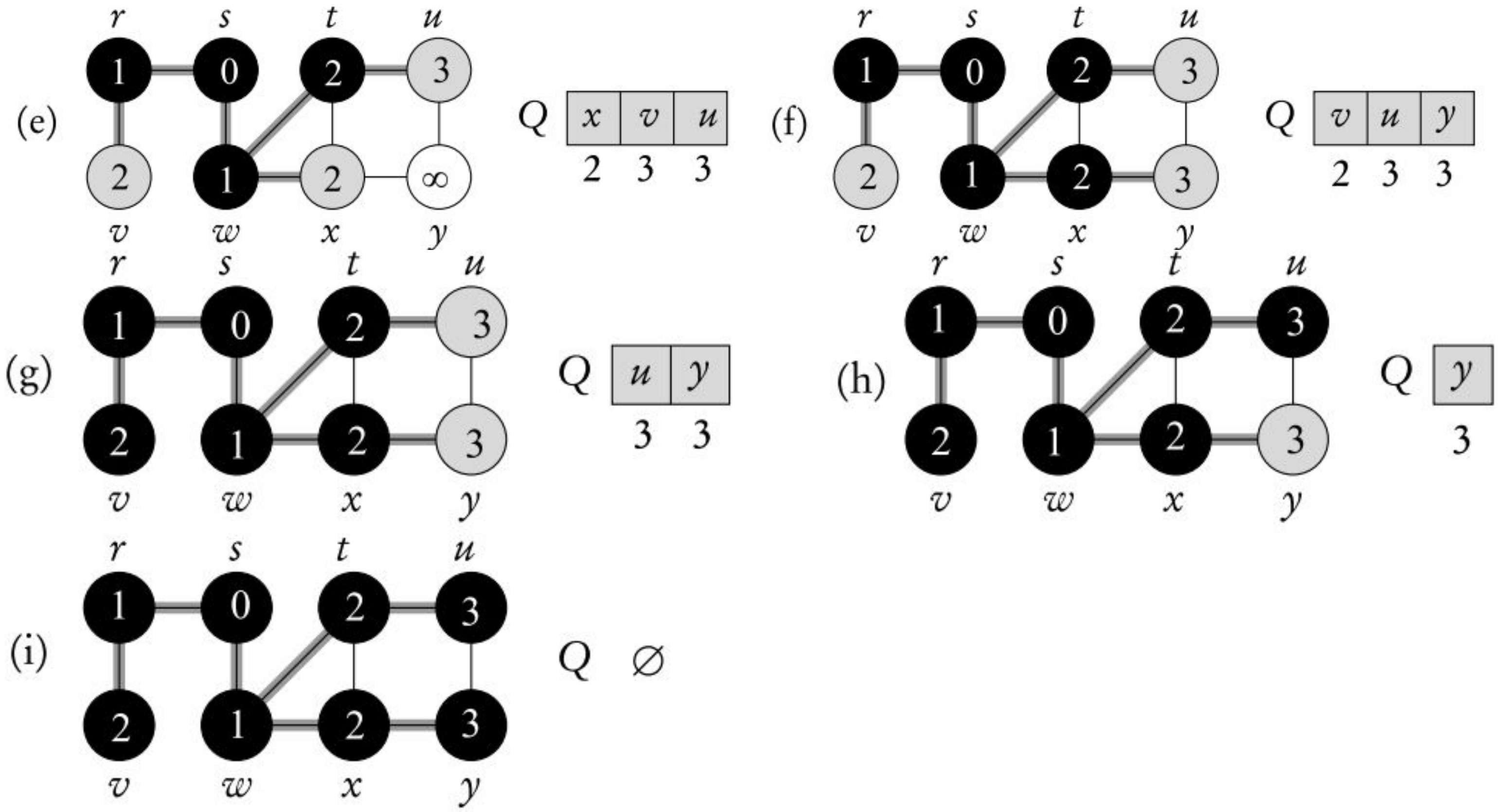
Parcurgerea în lățime : Exemplu



Parcurgerea în lățime : Exemplu



Parcurgerea în lățime : Exemplu



Parcurgerea în lățime: algoritm

Sa scriem impreuna pseudocodul:

Parcurgerea în lățime

Algoritm Cormen:

```
CL(G, s)
pentru fiecare vârf u ∈ V[G] − {s} execută
    color[u] ← ALB
    d[u] ← ∞
    π[u] ← NIL
color[s] ← GRI
d[s] ← 0
π[s] ← NIL
Q ← {s}
cât timp Q ≠ ∅ execută
    u ← cap[Q]
    pentru fiecare vârf v ∈ Adj[u] execută
        dacă color[v] = ALB atunci
            color[v] ← GRI
            d[v] ← d[u] + 1
            π[v] ← u
            PUNE-ÎN-COADĂ(Q, v)
    SCOATE-DIN-COADĂ(Q)
    color[u] ← NEGRU
```

Parcurgerea în lățime: complexitate

??

Parcurgerea în lățime: complexitate

O(V+E) sau O(n+m)

Parcurgerea în lățime: complexitate

O(V+E) sau O(n+m)

Parcurgerea în lățime: aplicații

Ieșirea din labirint în număr minim de pași:

0 0 -> punct de pornire

0 7   -> punct de iesire

```
 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 în număr minim de pași:

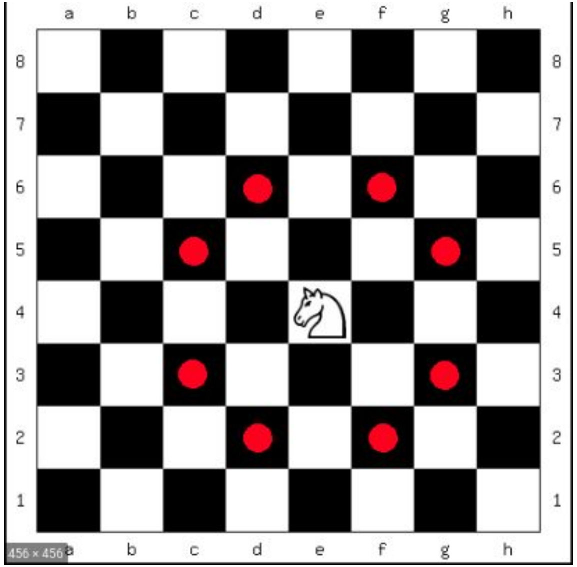
0 0 -> punct de pornire

0 7   -> punct de iesire

```
 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 minima a calului pe tabla de sah.



Parcurgerea în adâncime : algoritm

1. Se începe explorarea dintr-un nod nevizitat
2. Se cauta un vecin nevizitat care devine noul nod curent . Cat timp noul curent are un vecin nevizitat repetam pasul 2 -> intrăm în adâncime.
3. Cand nodul curent nu are nici un vecin nevizitat, ne întoarcem la strămoșii lui (tatăl, bunicul…) pana găsim un nod care are vecini vizitati și reluăm pasul 2.
4. Dacă exista noduri nevizitate reluăm pasul 1…

   Cand se intampla cazul 4?

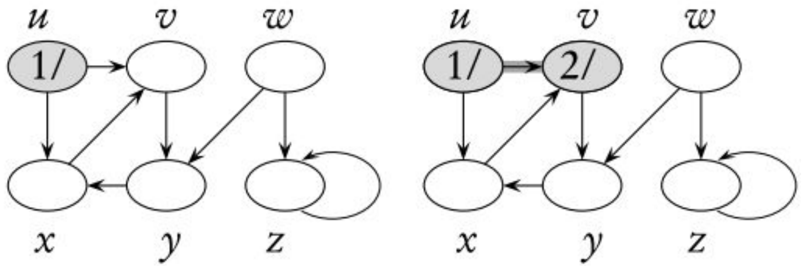
Parcurgerea în adâncime: cu timpi de intrare și ieșire

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

1. De fiecare data cand ajungem într-un nod sau cand terminam de vizitat toți vecinii incrementam un contor și ținem minte aceste informații …

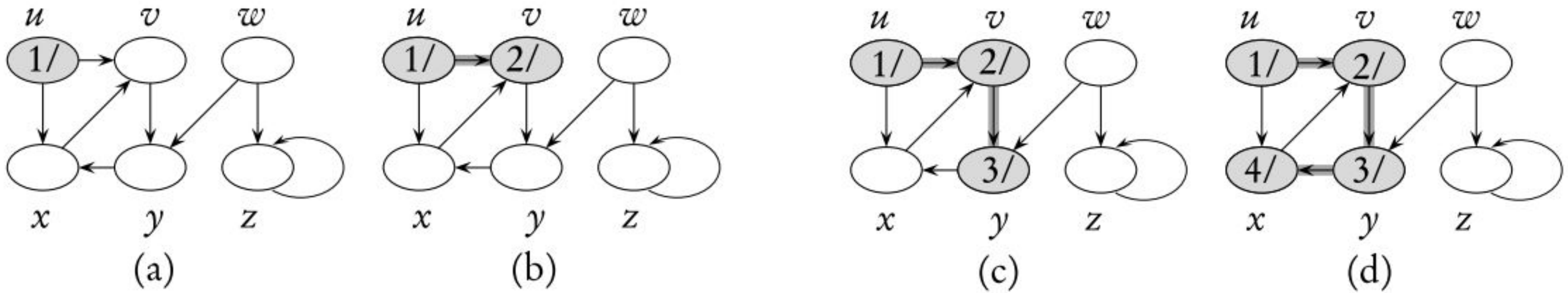
Observație: (O sa existe situații în care cronologia va fi un pic diferită, cum s-a intamplat la RMQ -> LCA)

Parcurgerea în adâncime: exemplu

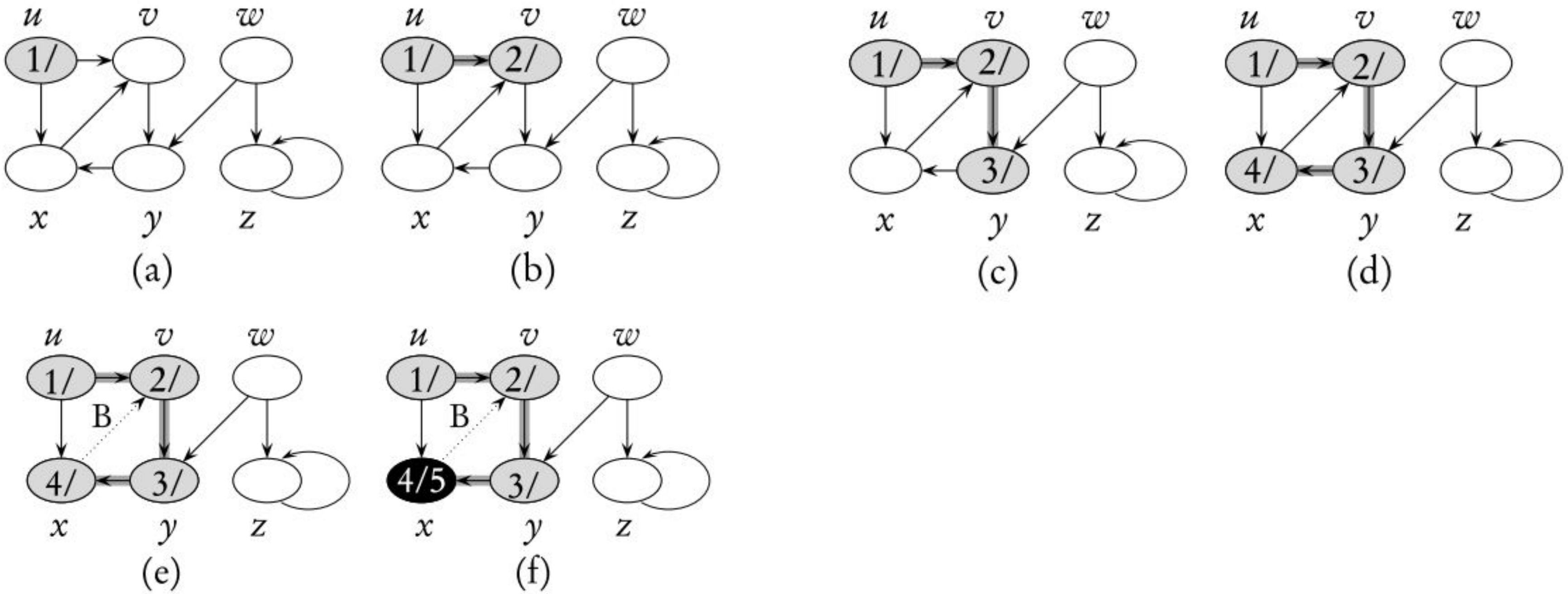


(a) (b)

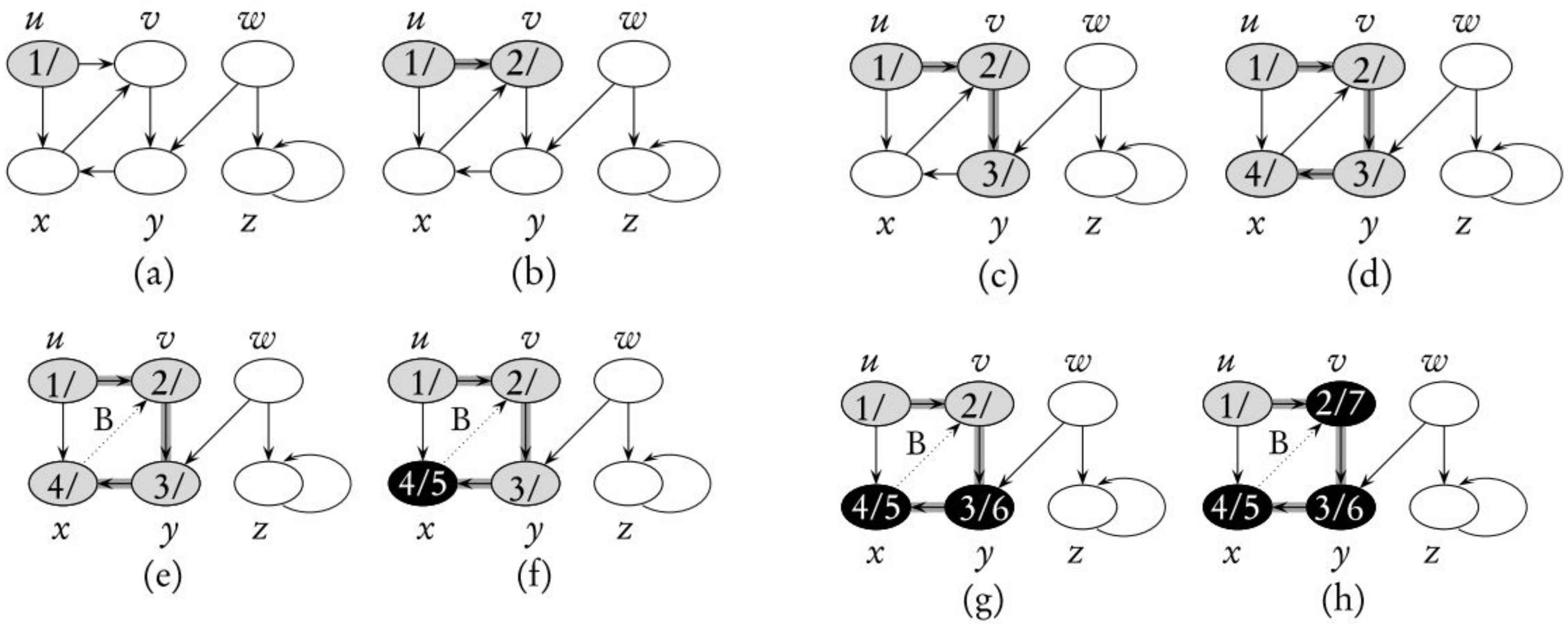
Parcurgerea în adâncime: exemplu



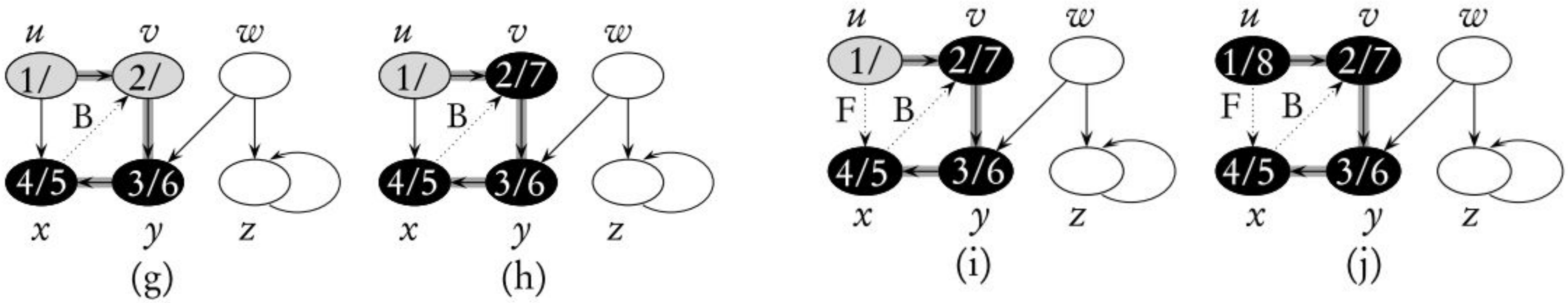
Parcurgerea în adâncime: exemplu



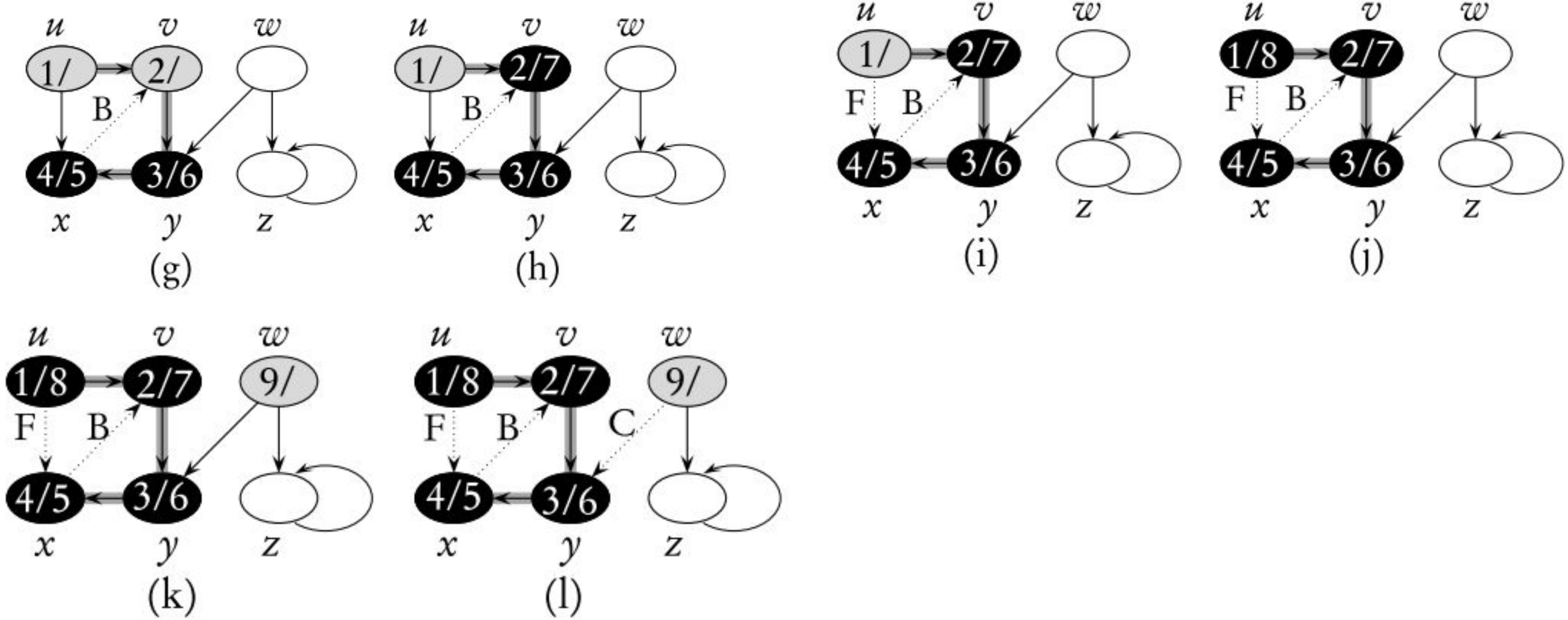
Parcurgerea în adâncime: exemplu



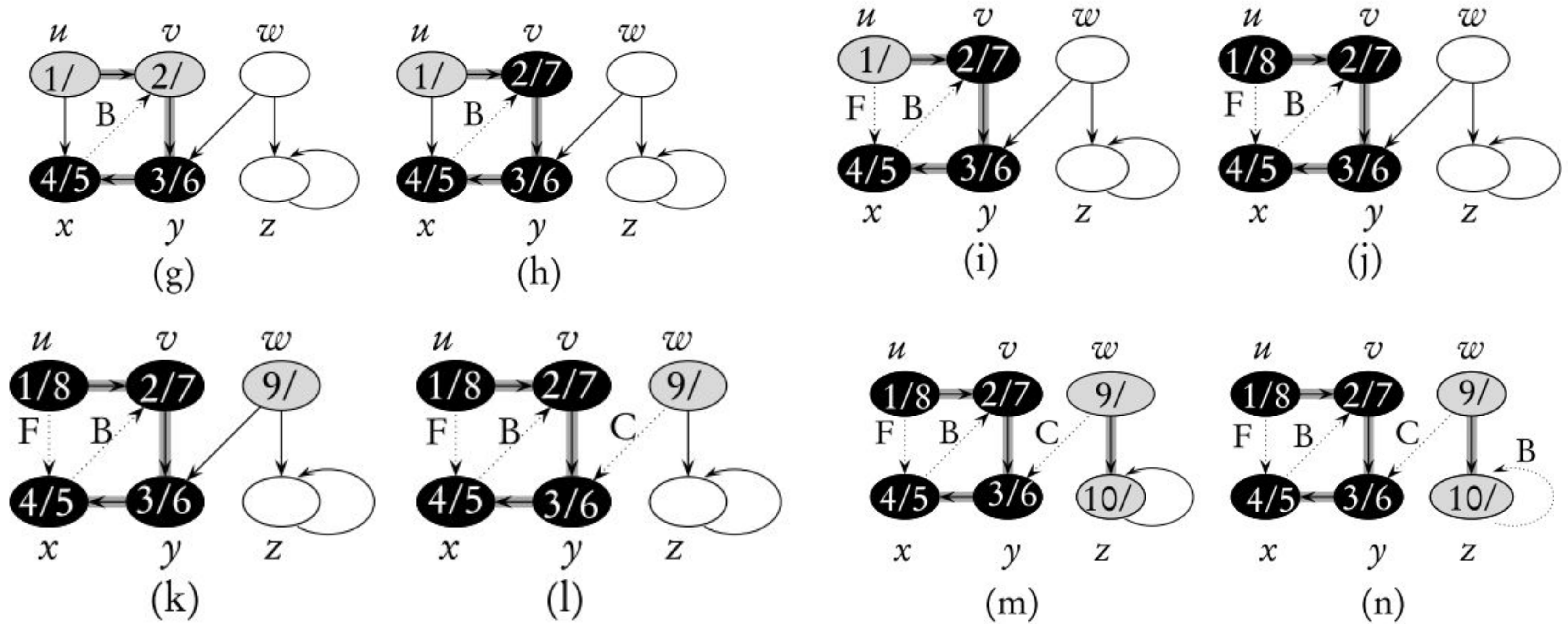
Parcurgerea în adâncime: exemplu



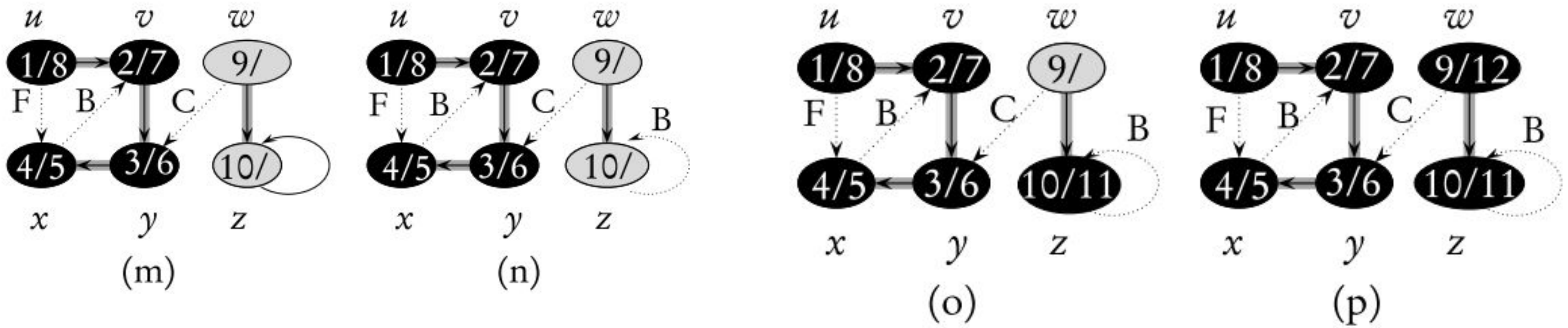
Parcurgerea în adâncime: exemplu



Parcurgerea în adâncime: exemplu



Parcurgerea în adâncime: exemplu



Parcurgerea în adâncime: algoritm

Sa scriem impreuna pseudocodul:

Parcurgerea în adâncime: 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: Teorema parantezelor

Teorema parantezelor: În orice căutare în adâncime a unui graf (orientat sau neorientat) G = (V, E), pentru orice 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, sau
- 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

Demonstraţie. 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. 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]]. În celălalt subcaz f[u] < d[v] şi inegalitatea (23.1) 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.

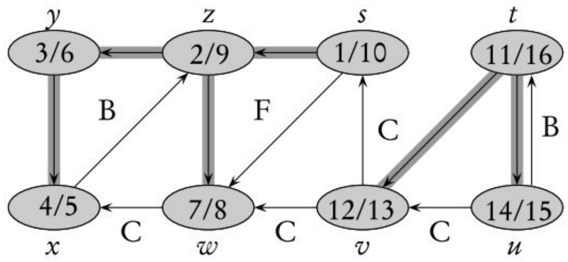
Parcurgerea în adâncime: Teorema parantezelor

Corolarul 23.7 (Interclasarea intervalelor descendenţilor)

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: proprietăți

(a)



(b)

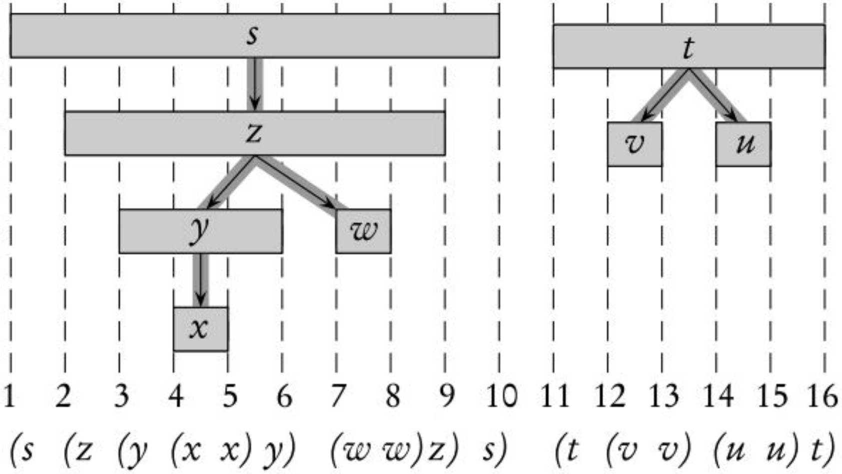
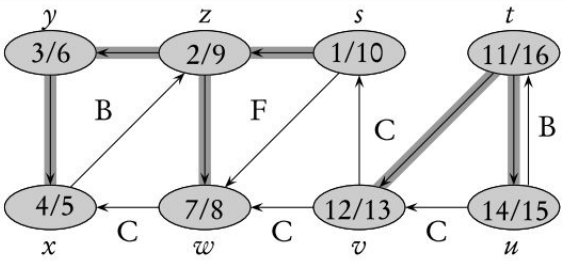


Diagrama b exemplifica foarte bine teorema anterioara.

Parcurgerea în adâncime: proprietăți

(a)



(c)

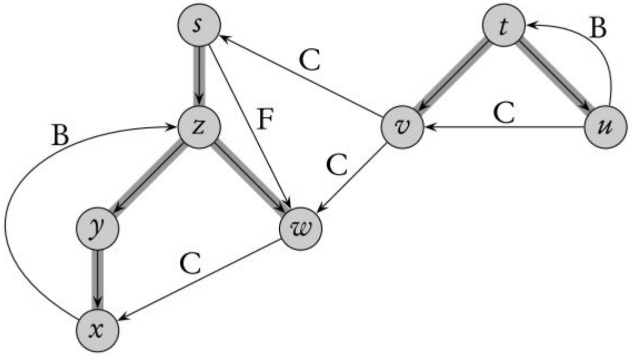


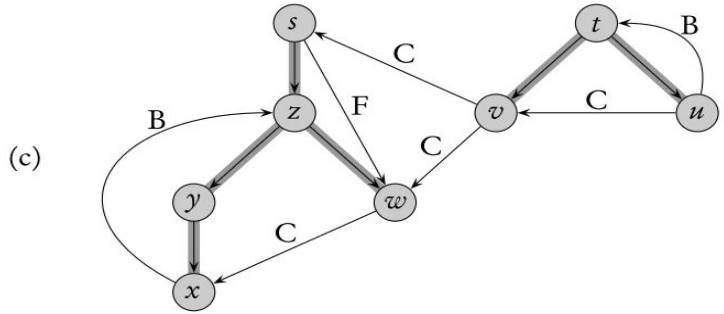
Diagrama ce exemplifica muchiile de întoarcere despre care vom vorbi imediat.

Parcurgerea în adâncime

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.

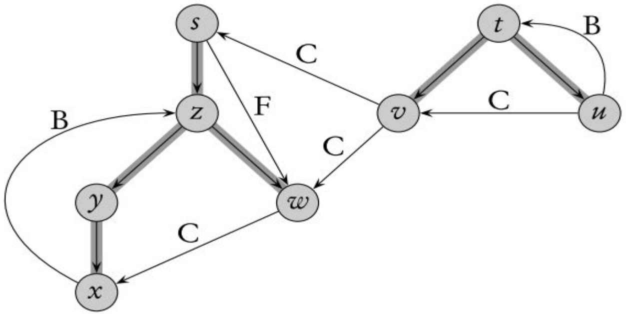
Parcurgerea în adâncime



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.

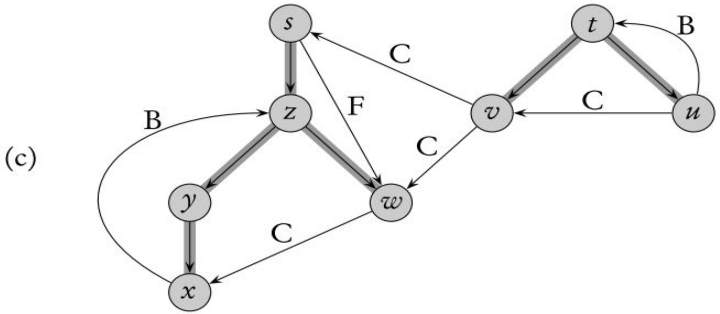
Parcurgerea în adâncime

(c)



Într-un graf orientat nu vom avea toate cele 4 categorii…
Ce categorii vom avea?

Parcurgerea în adâncime

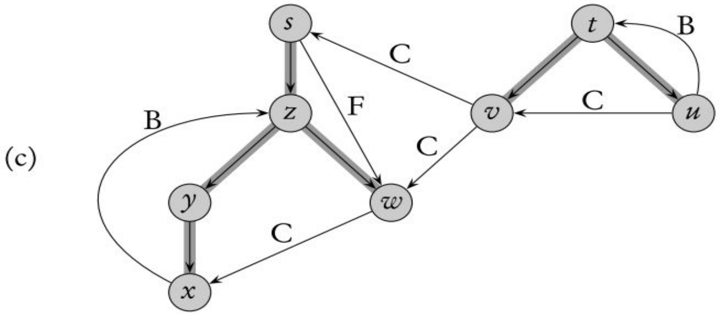


Într-un graf orientat nu vom avea toate cele 4 categorii…

Ce categorii vom avea?

- Doar primele 2 categorii.

Parcurgerea în adâncime

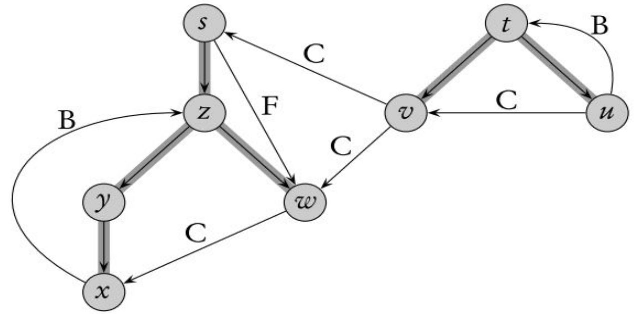


Într-un graf orientat vom avea un ciclu dacă găsim ce fel de muchie?

-

Parcurgerea în adâncime

(c)



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

- Muchie inapoi...