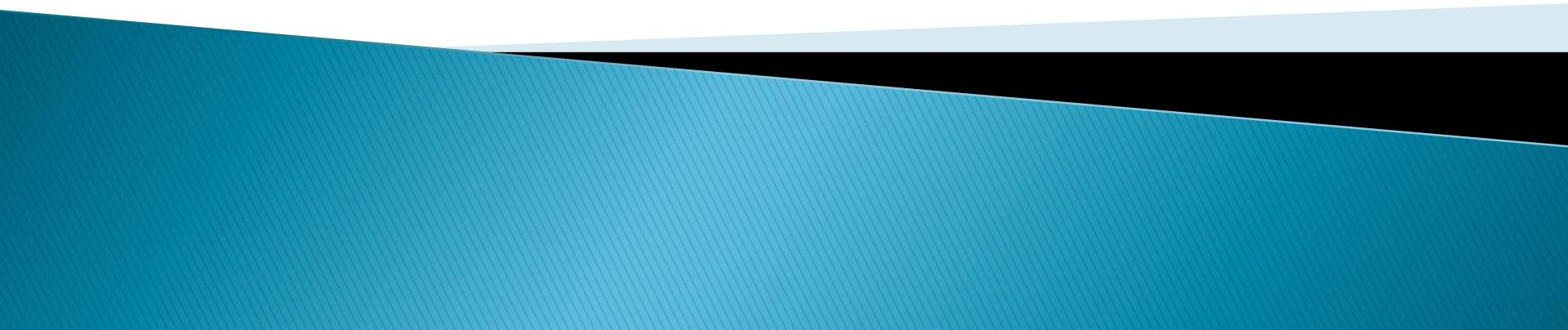


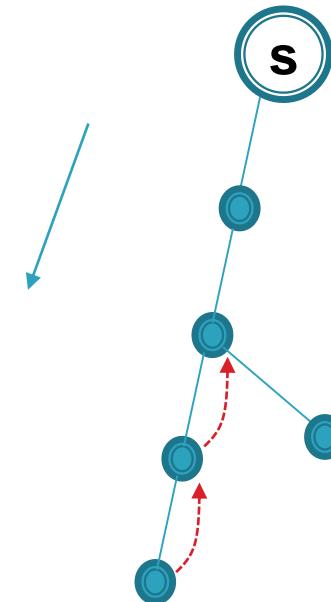
Parcurgerea în adâncime



Parcurgerea în adâncime

La un pas:

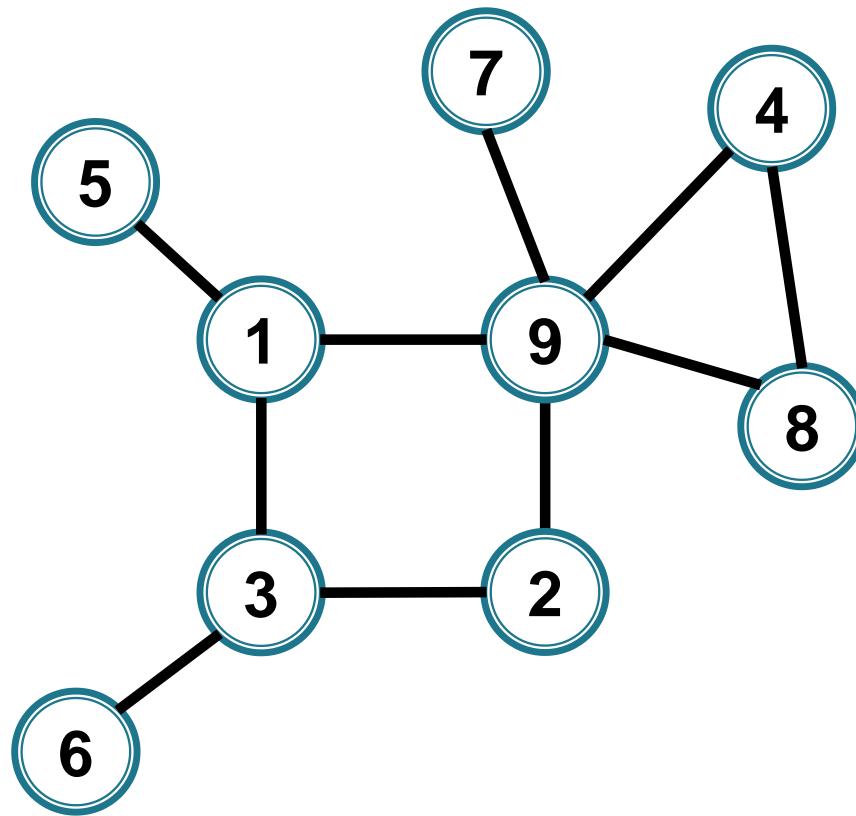
Coborâm adânc în graf,
mereu în primul vecin nevizitat
(ca o încercare de a ieși din labirint)

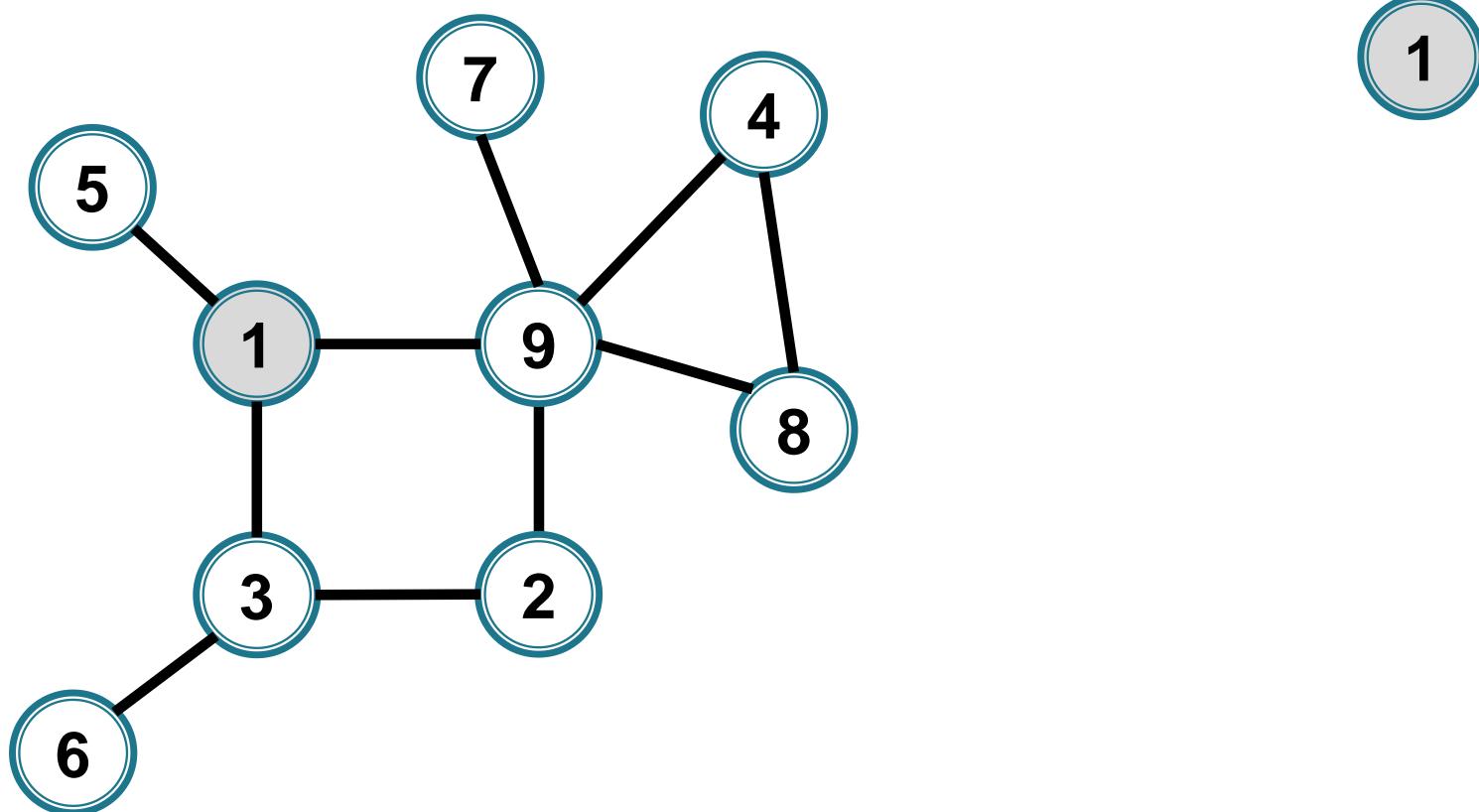


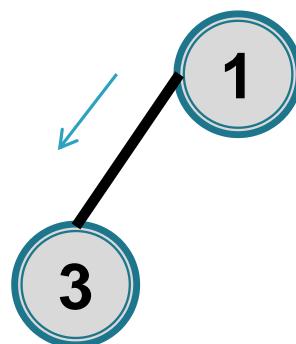
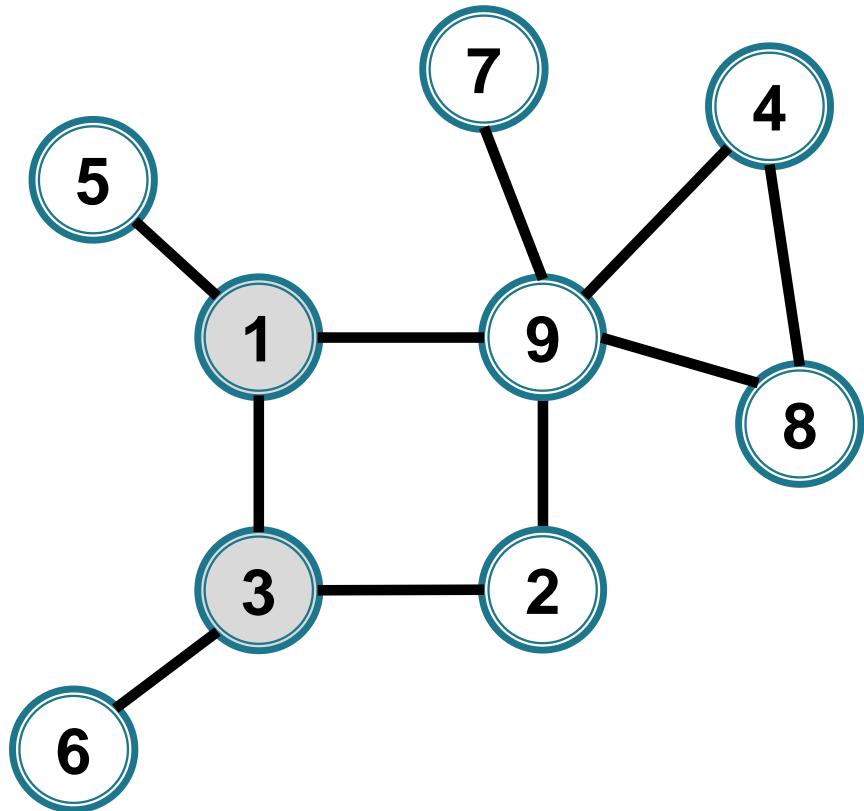
Când nu mai putem înainta
(vârful curent nu mai are vecini nevizitați)
ne întoarcem pe drumul pe care am venit (tată, bunic etc)
până găsim un vârf care mai are vecini nevizitați și reluăm
cotorârea

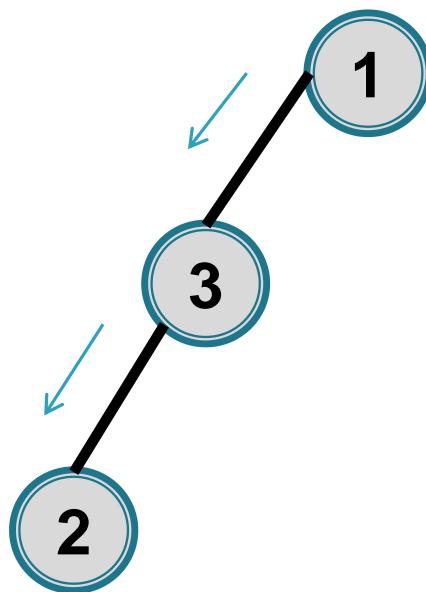
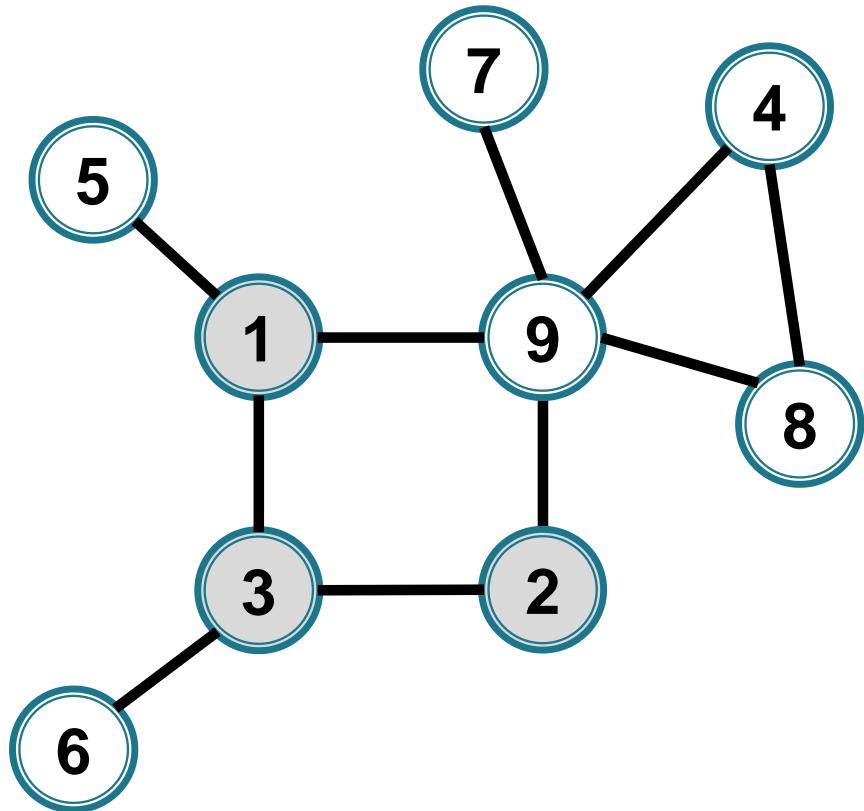
Parcurgerea în adâncime

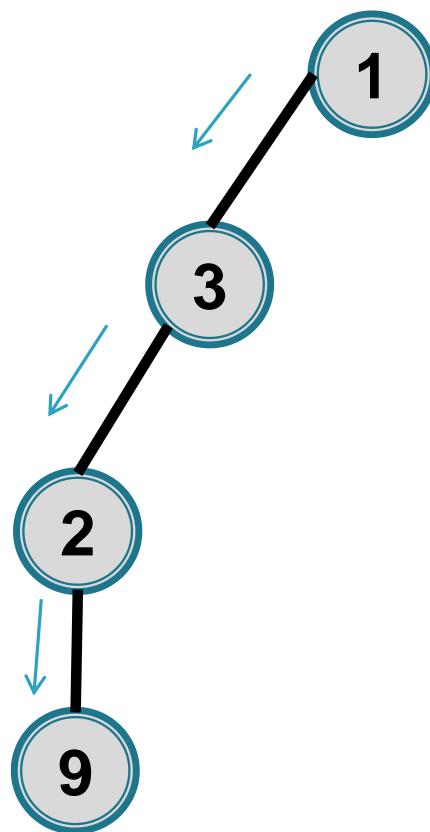
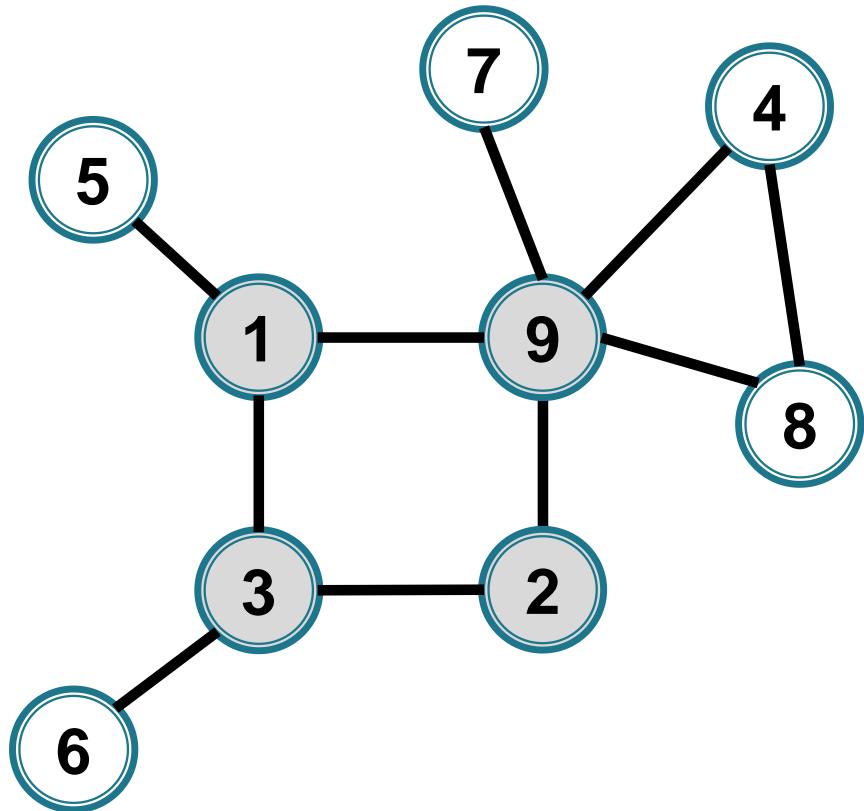
- Inițial: vârful de start s – devine vârf curent
- La un pas:
 - se trece la primul vecin nevizitat al vârfului curent, dacă există
 - altfel
 - se merge **înapoi** pe drumul de la s la vârful curent, până se ajunge la un vârf cu vecini nevizitați (dacă nu mai există un astfel de vârf parcurgerea de oprește)
 - se trece la **primul** dintre aceștia și se reia procesul

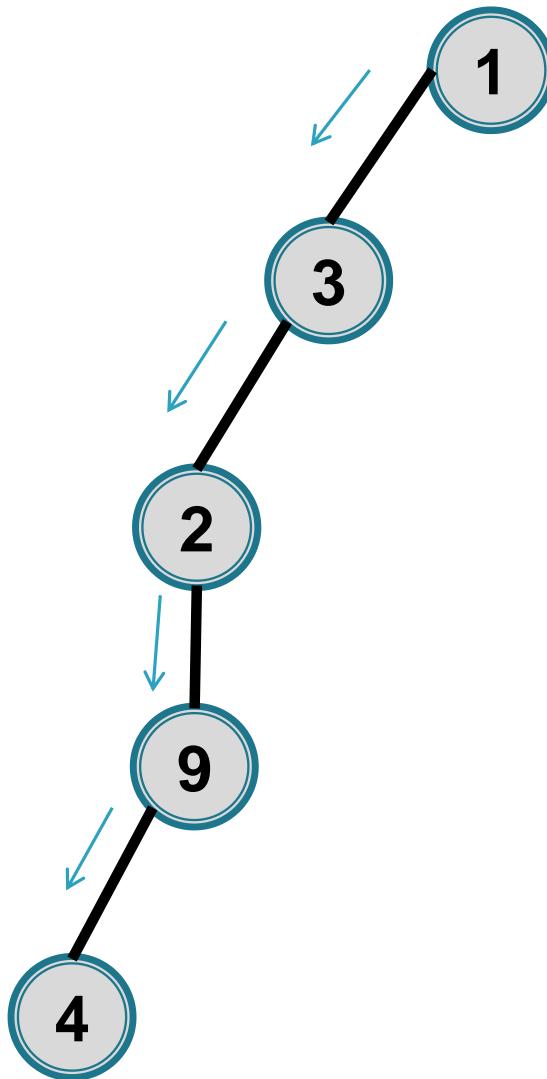
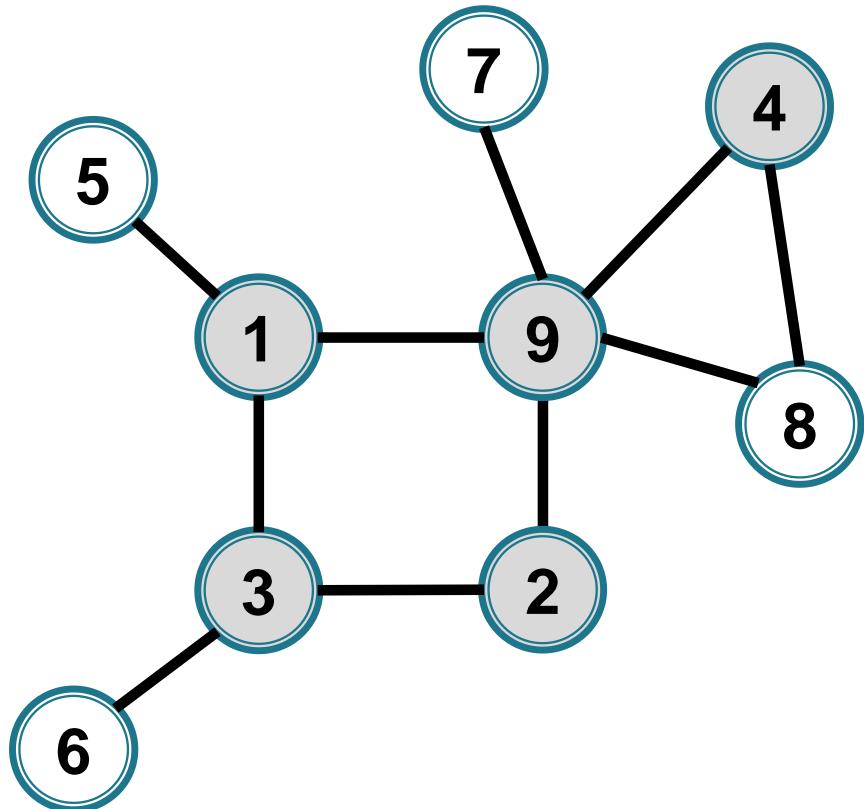


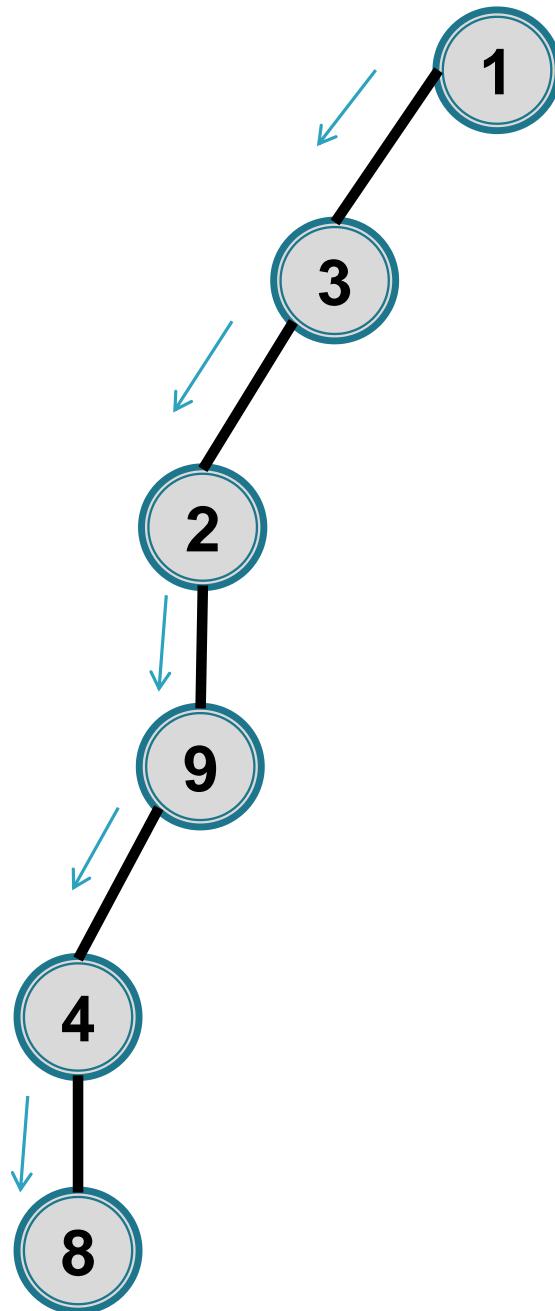
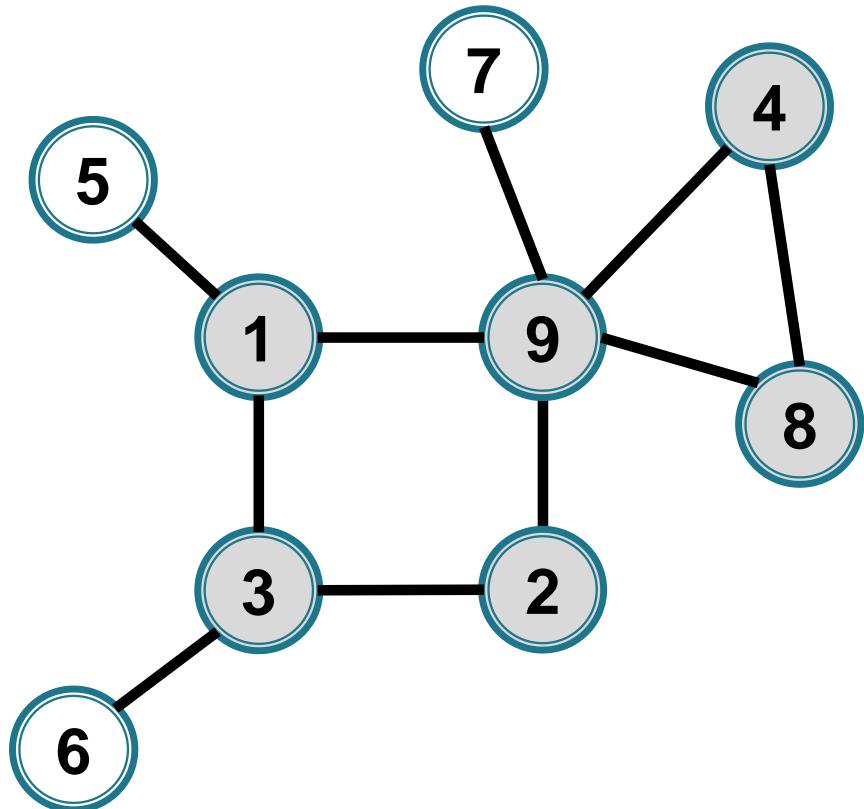


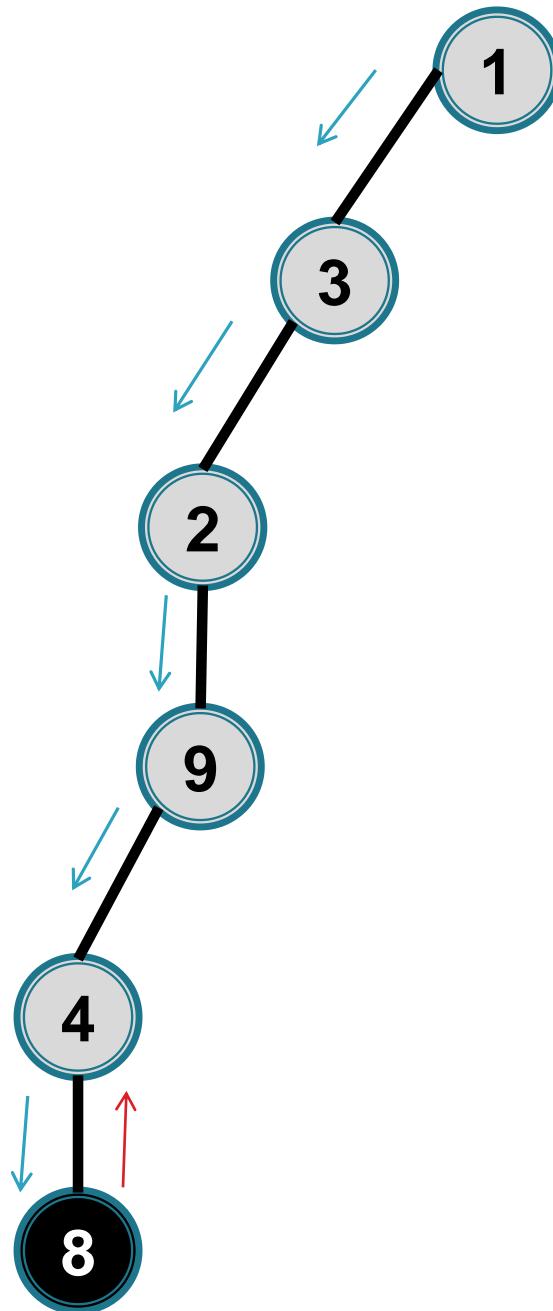
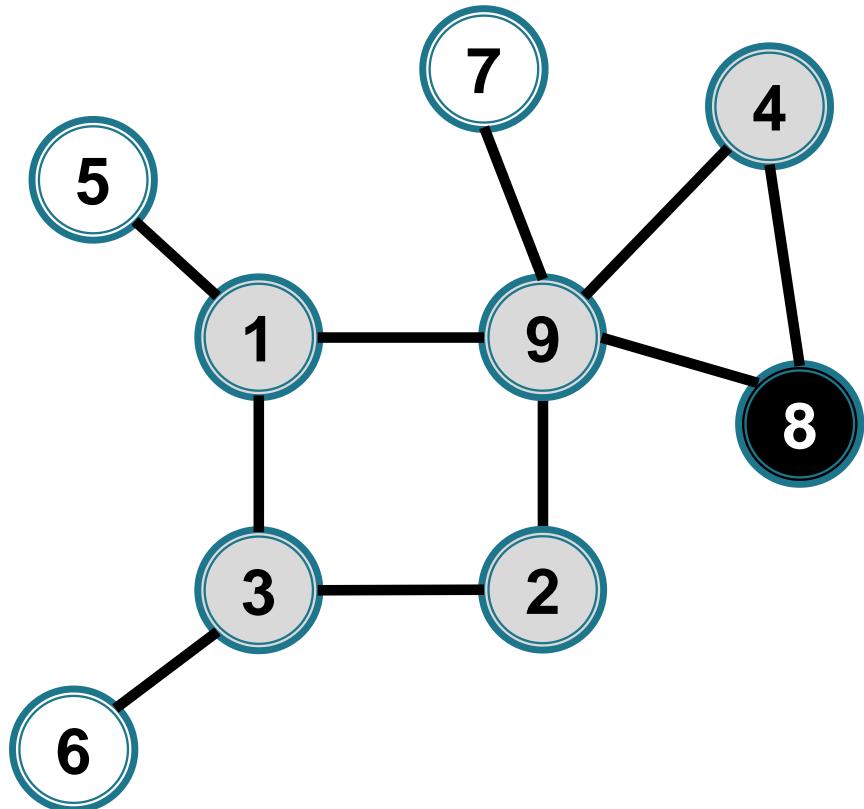


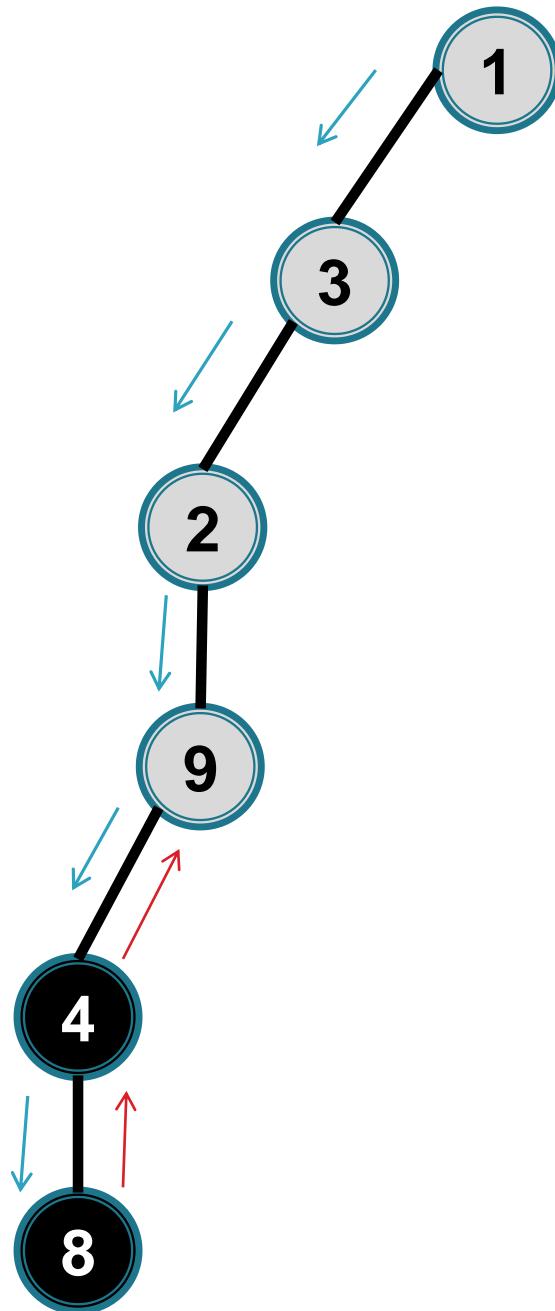
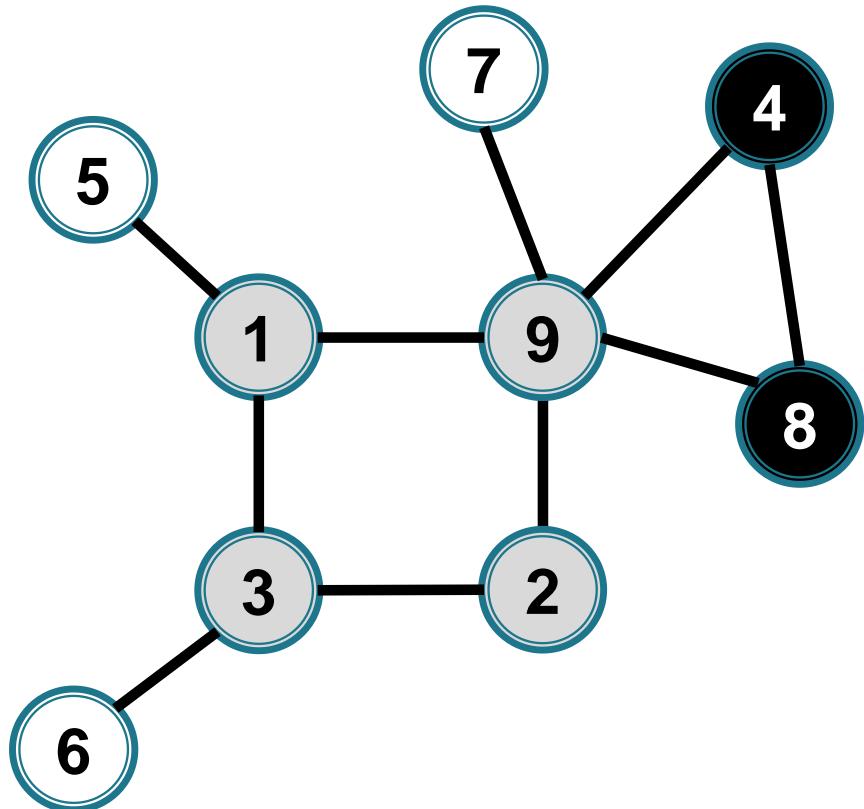


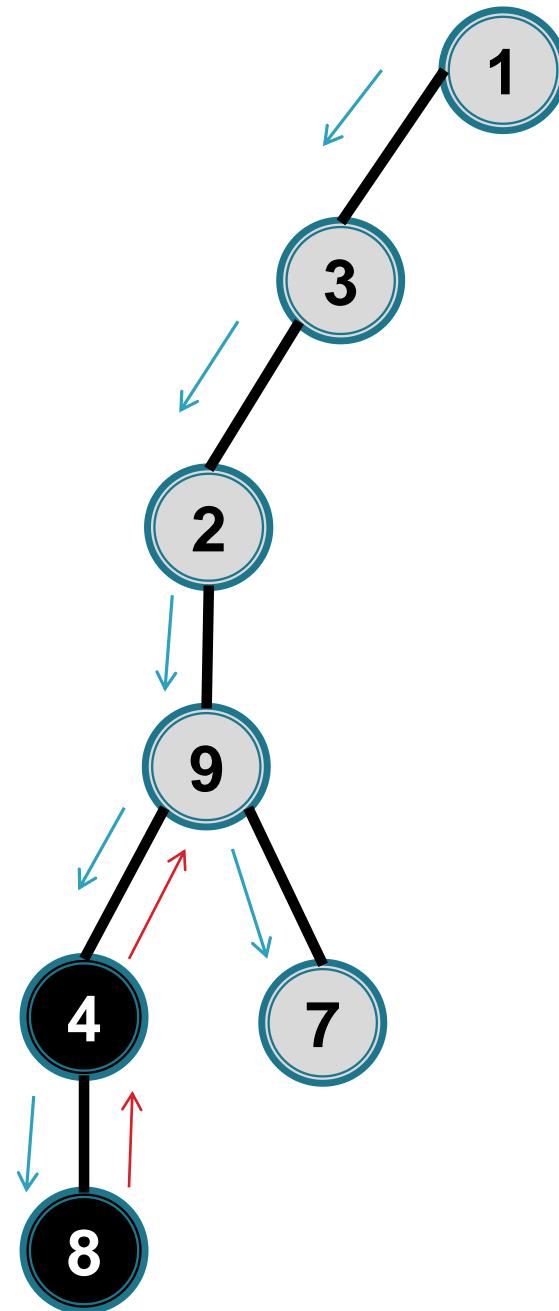
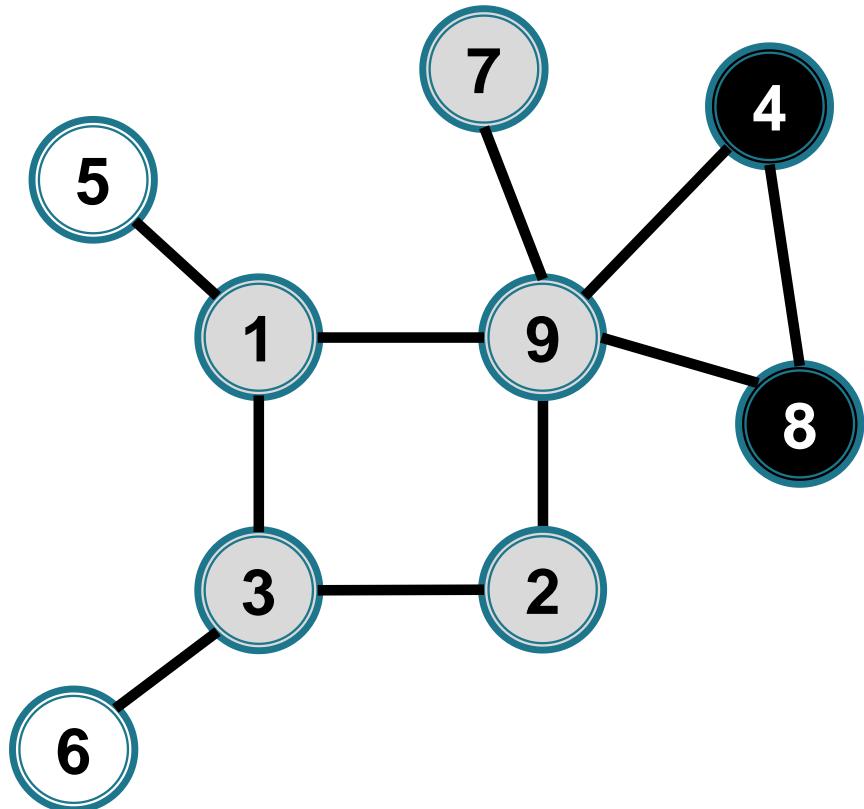


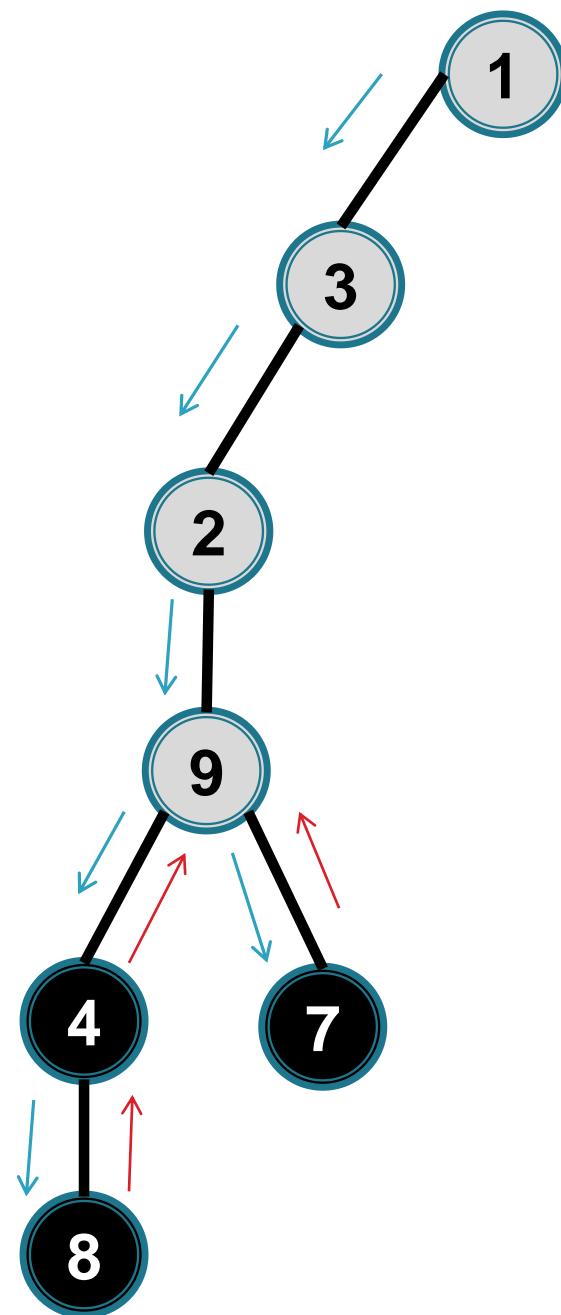
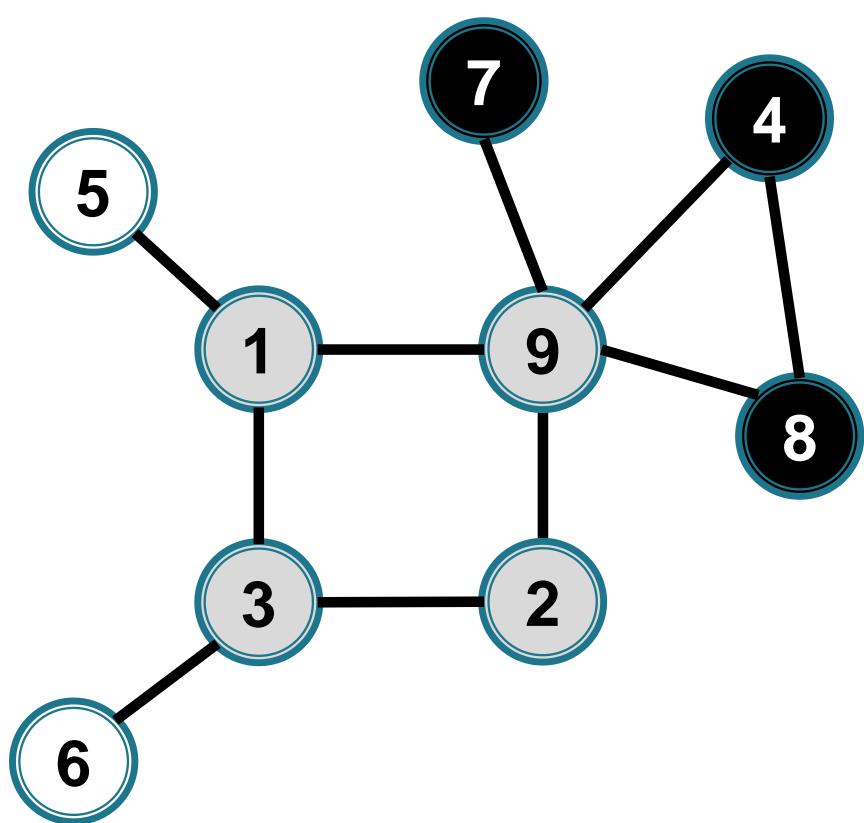


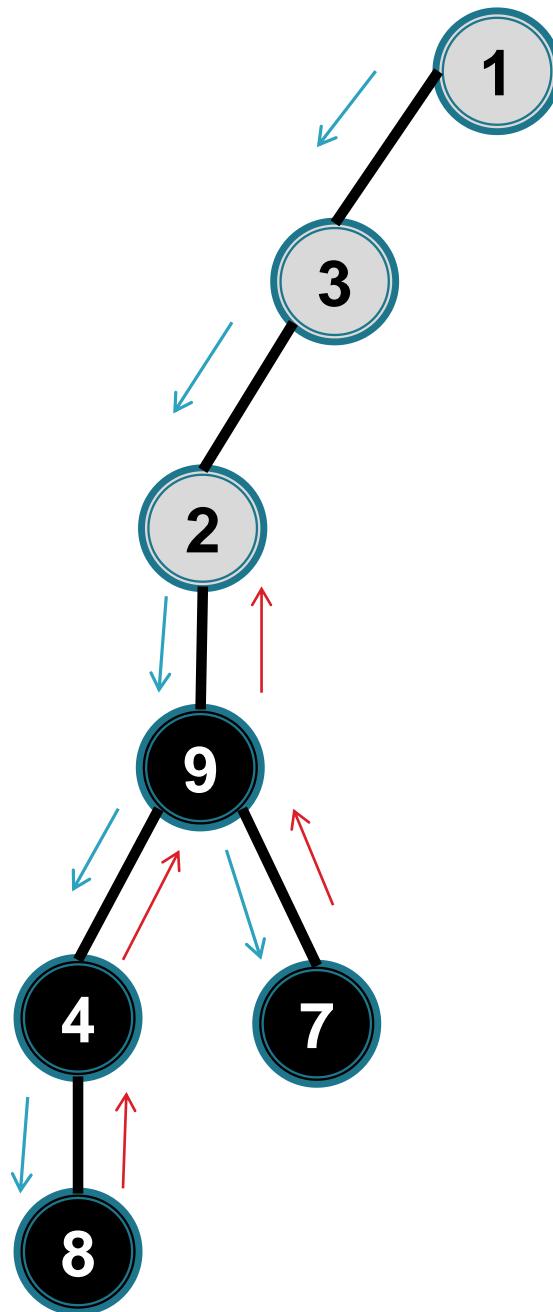
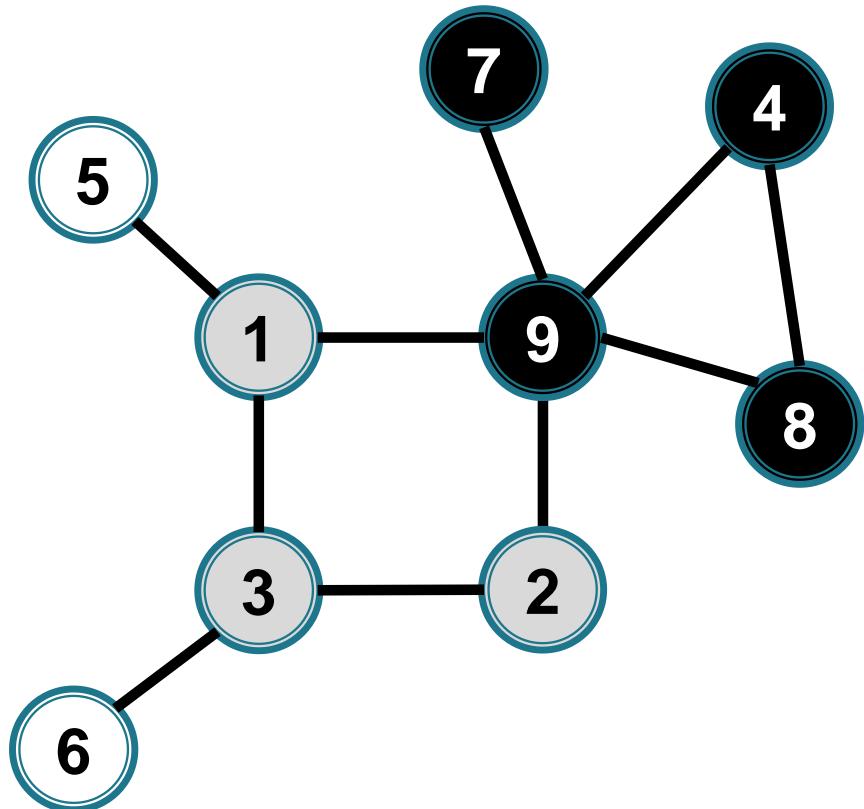


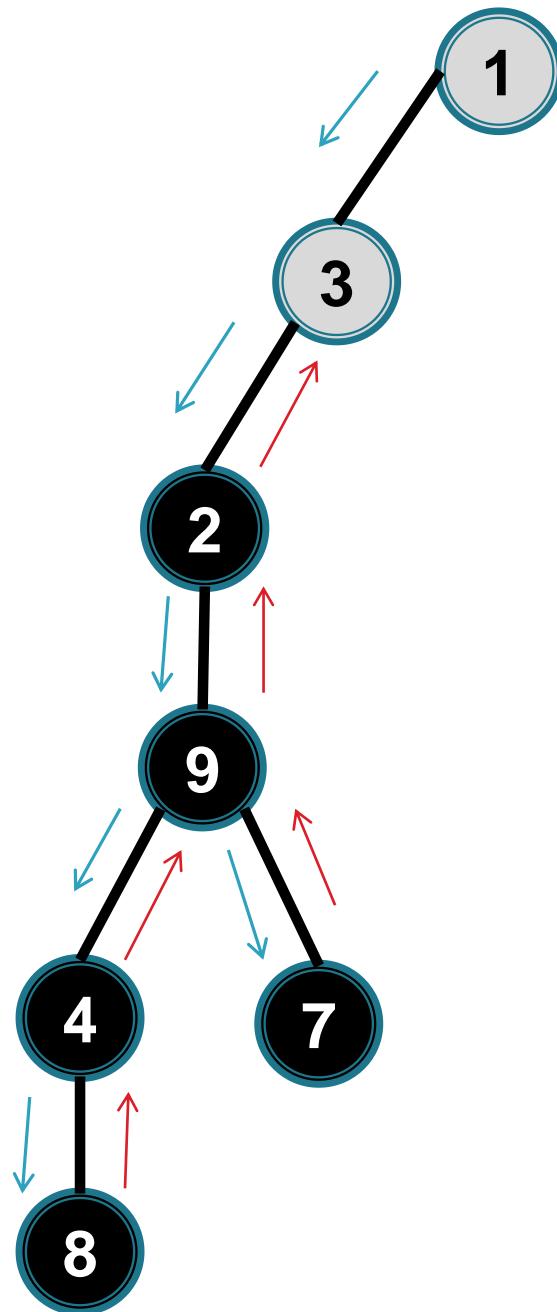
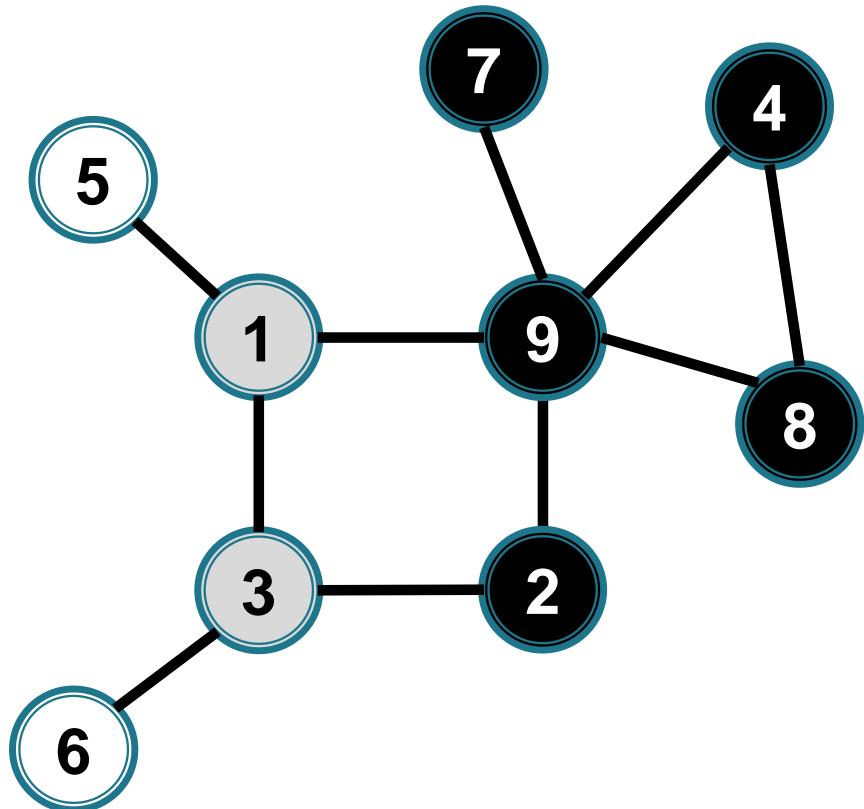


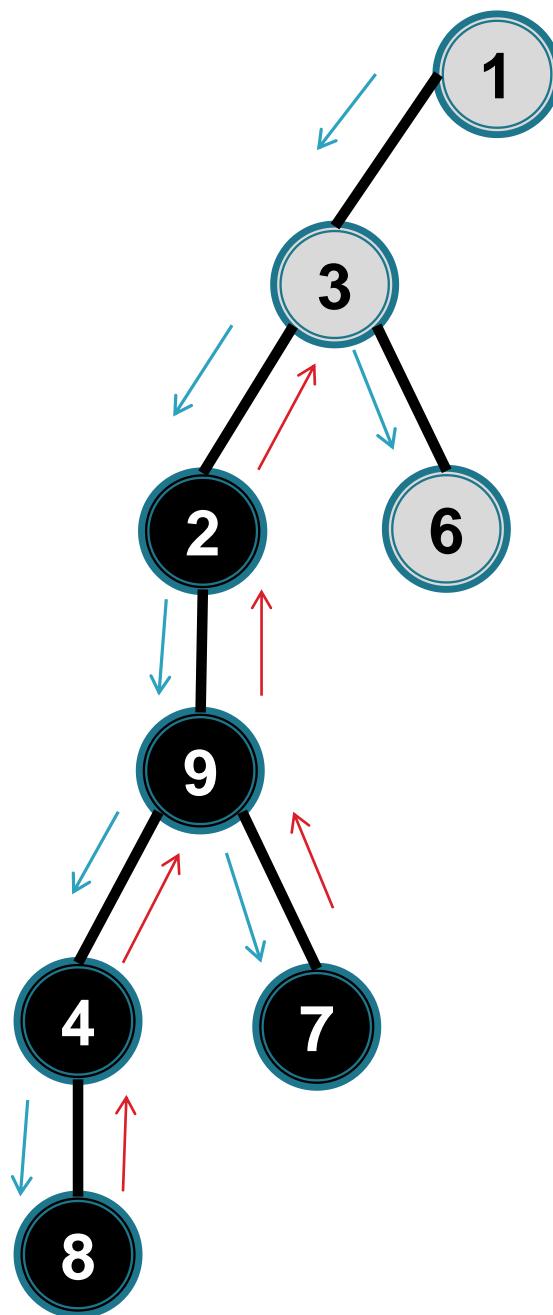
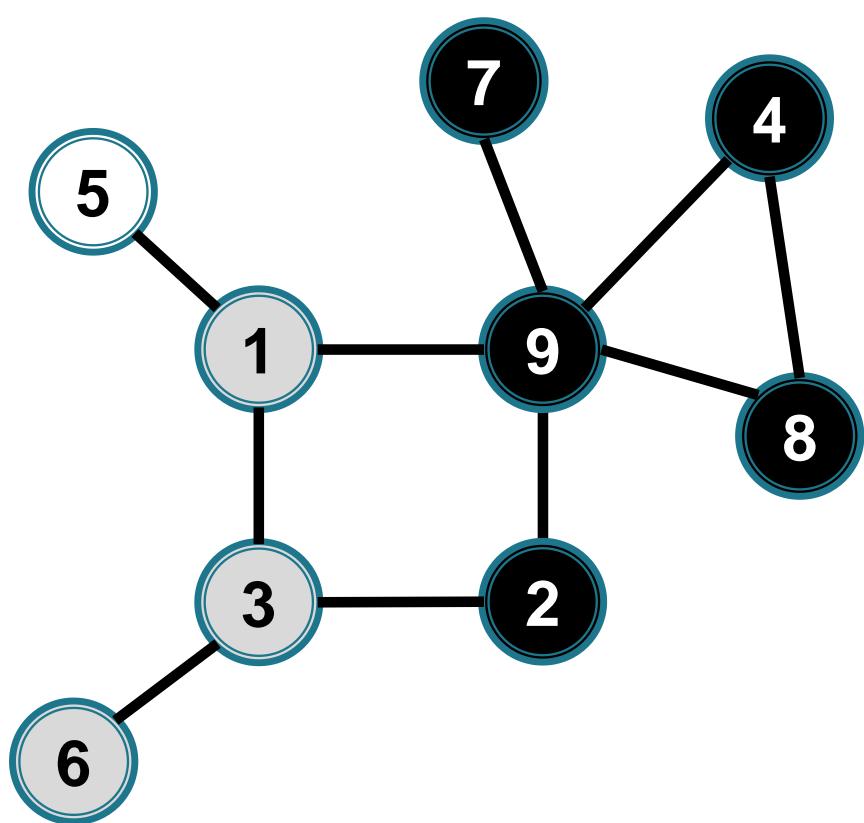


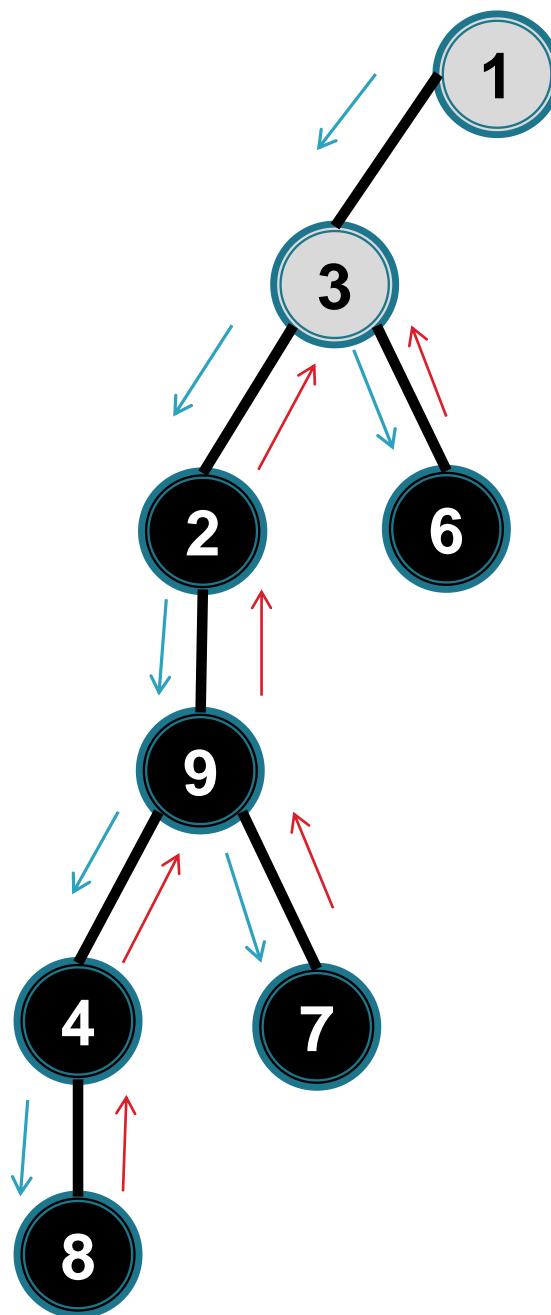
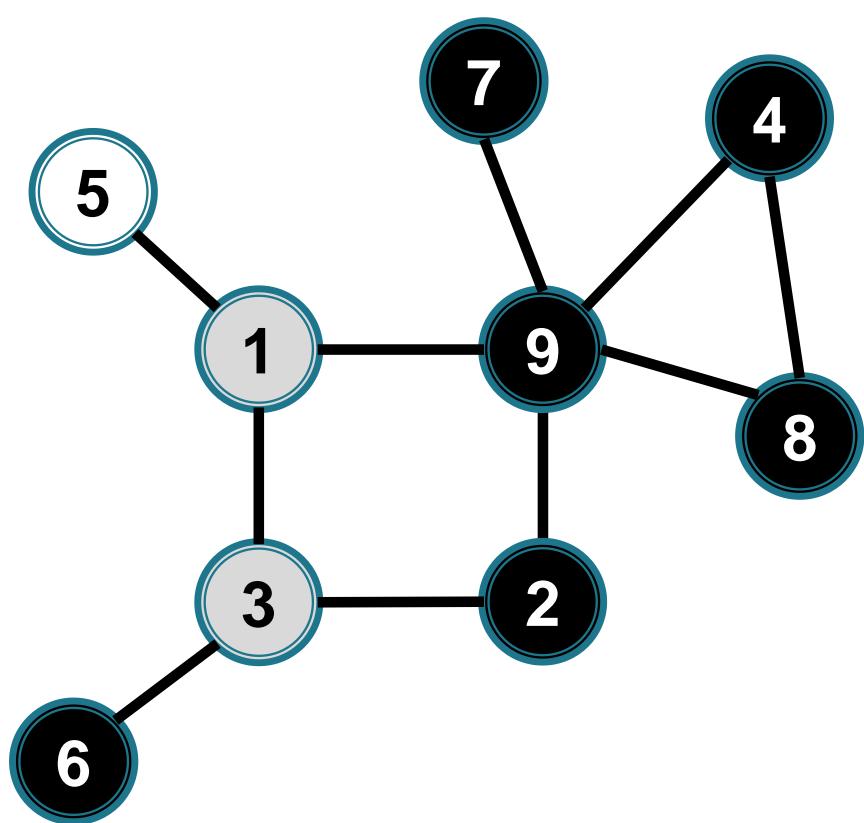


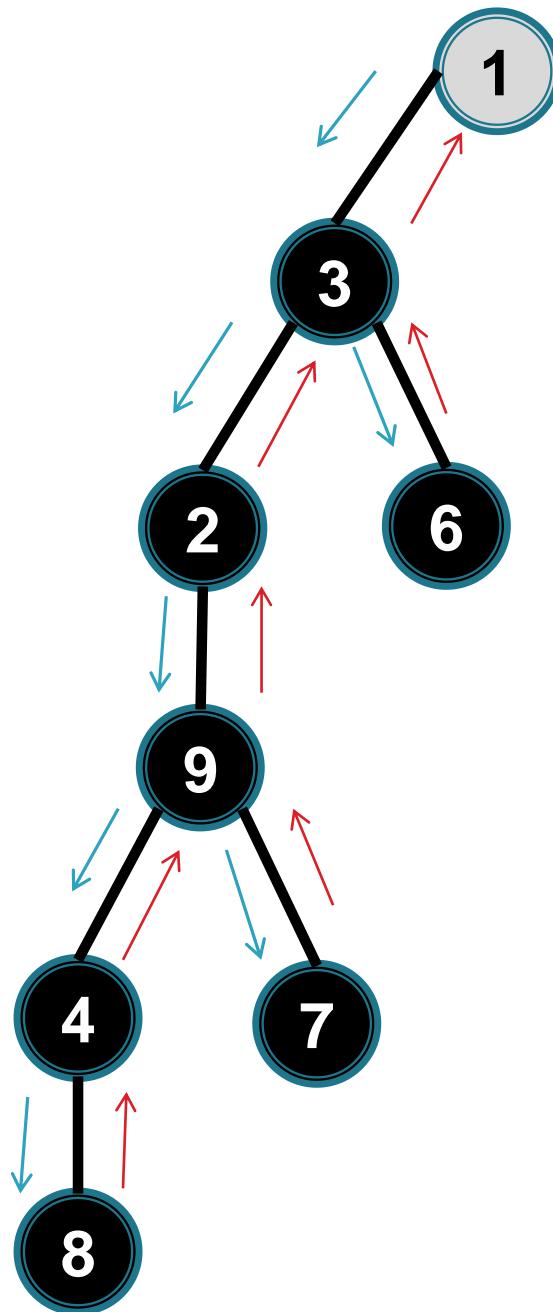
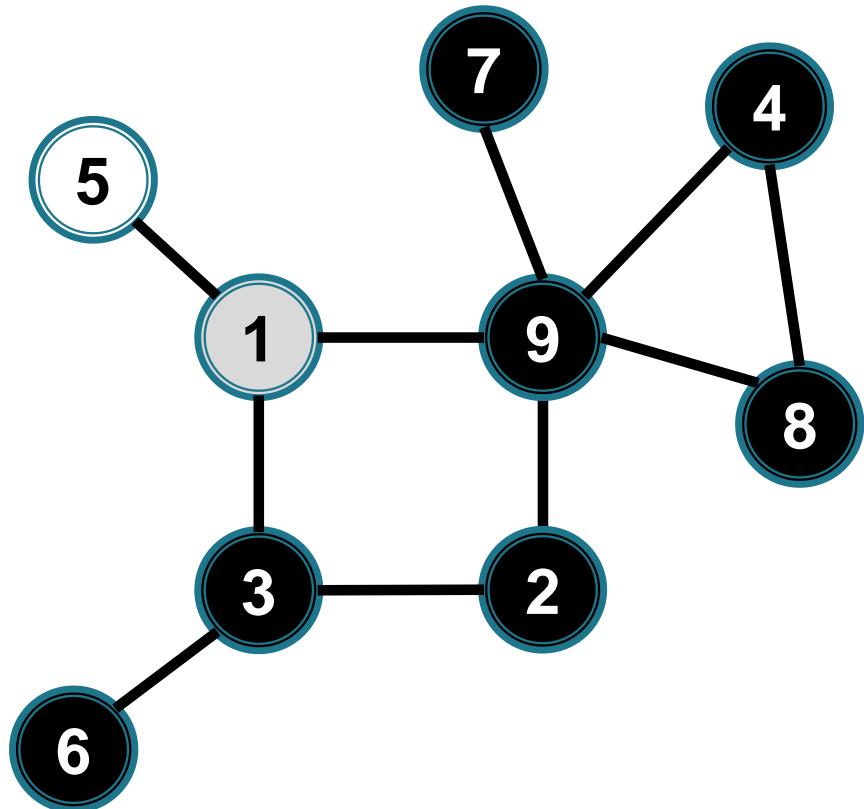


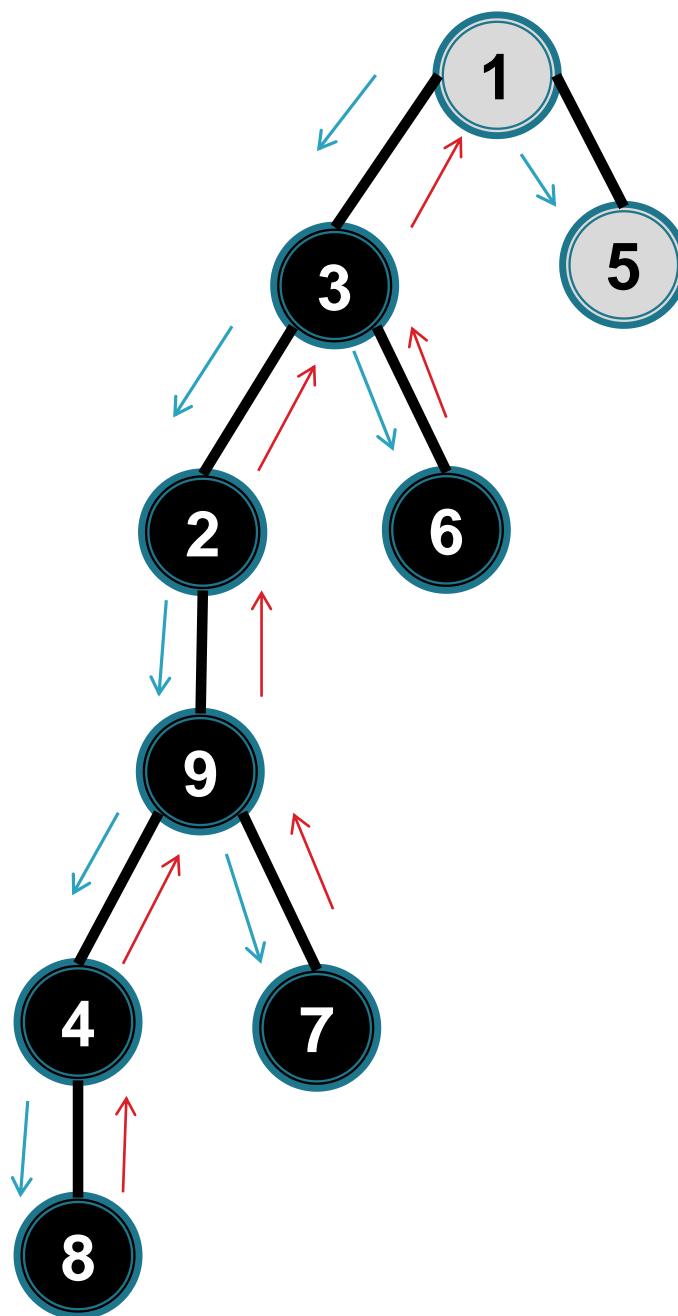
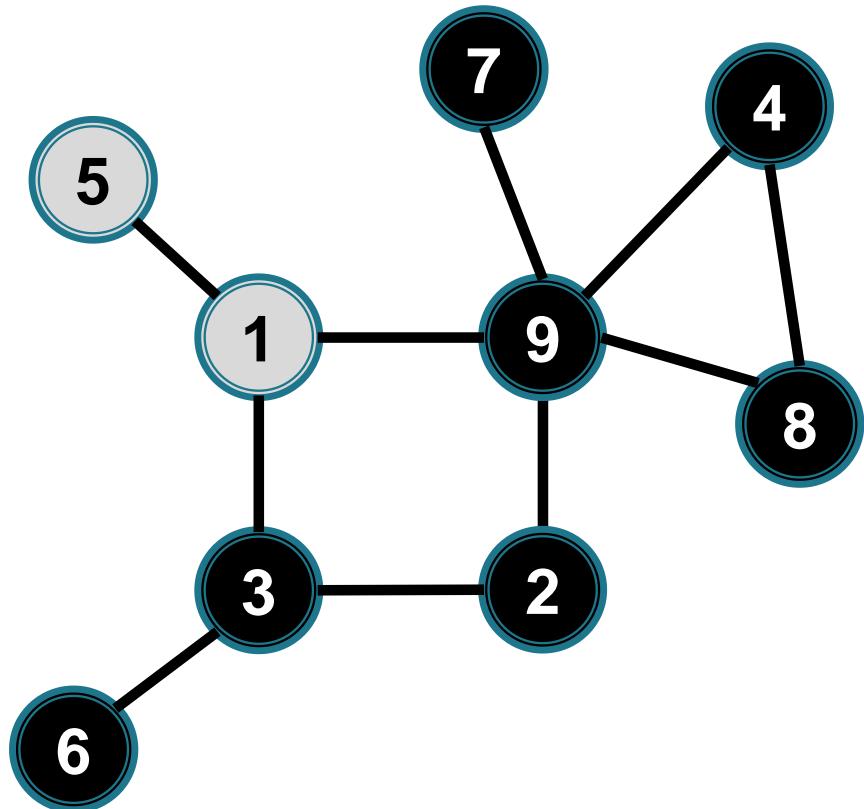


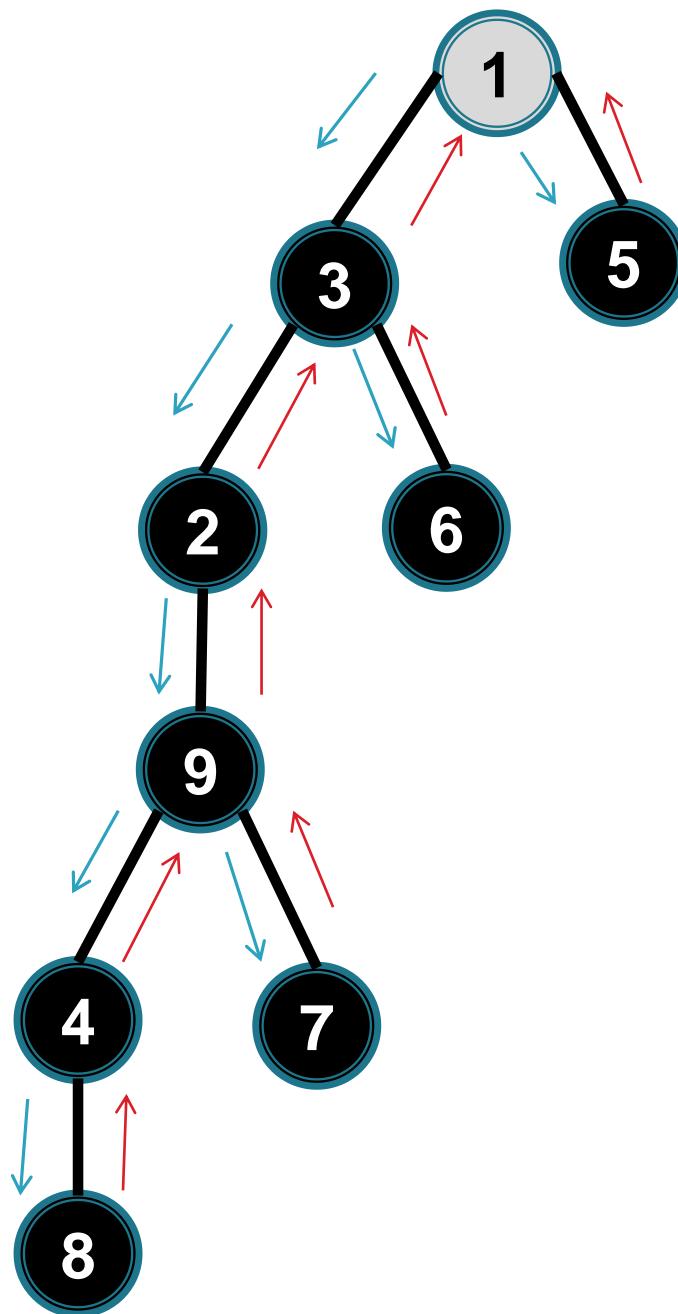
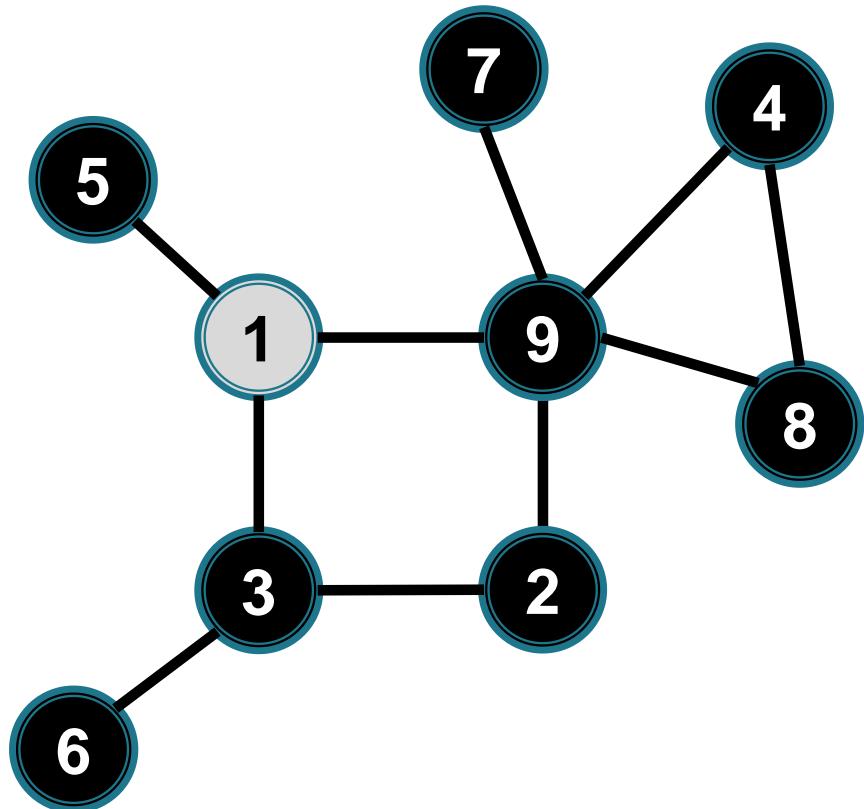


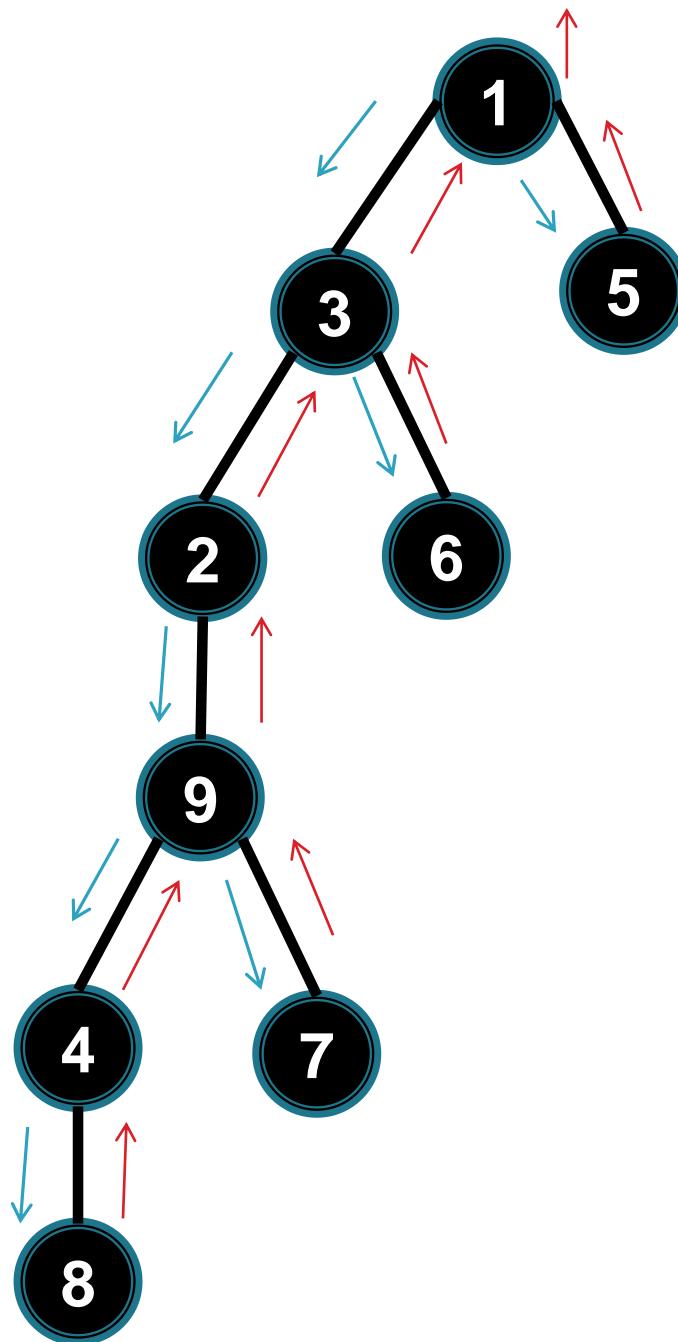
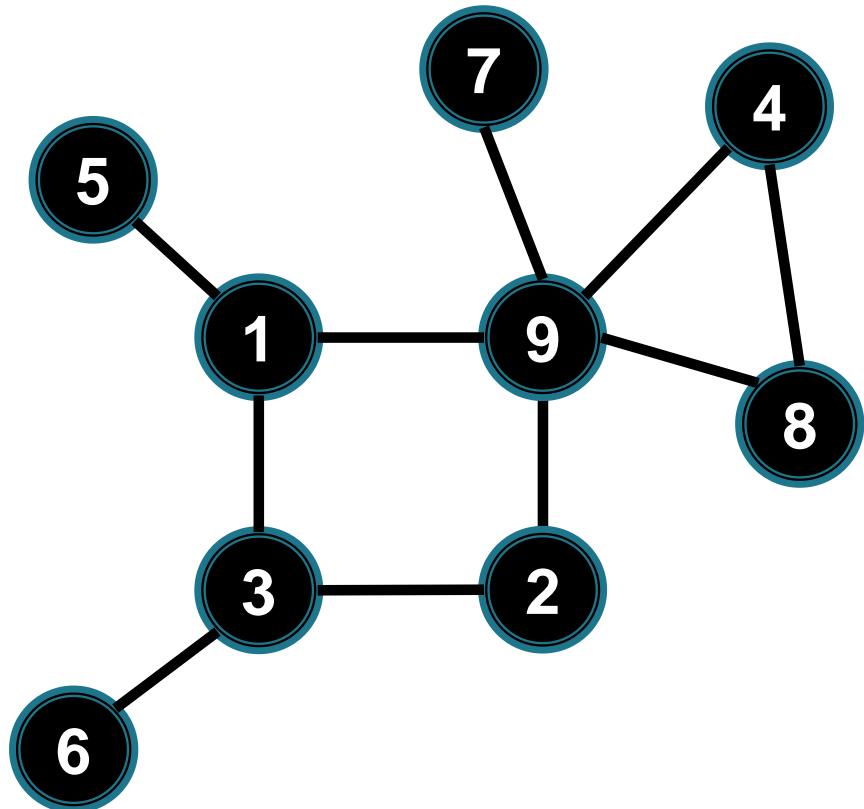


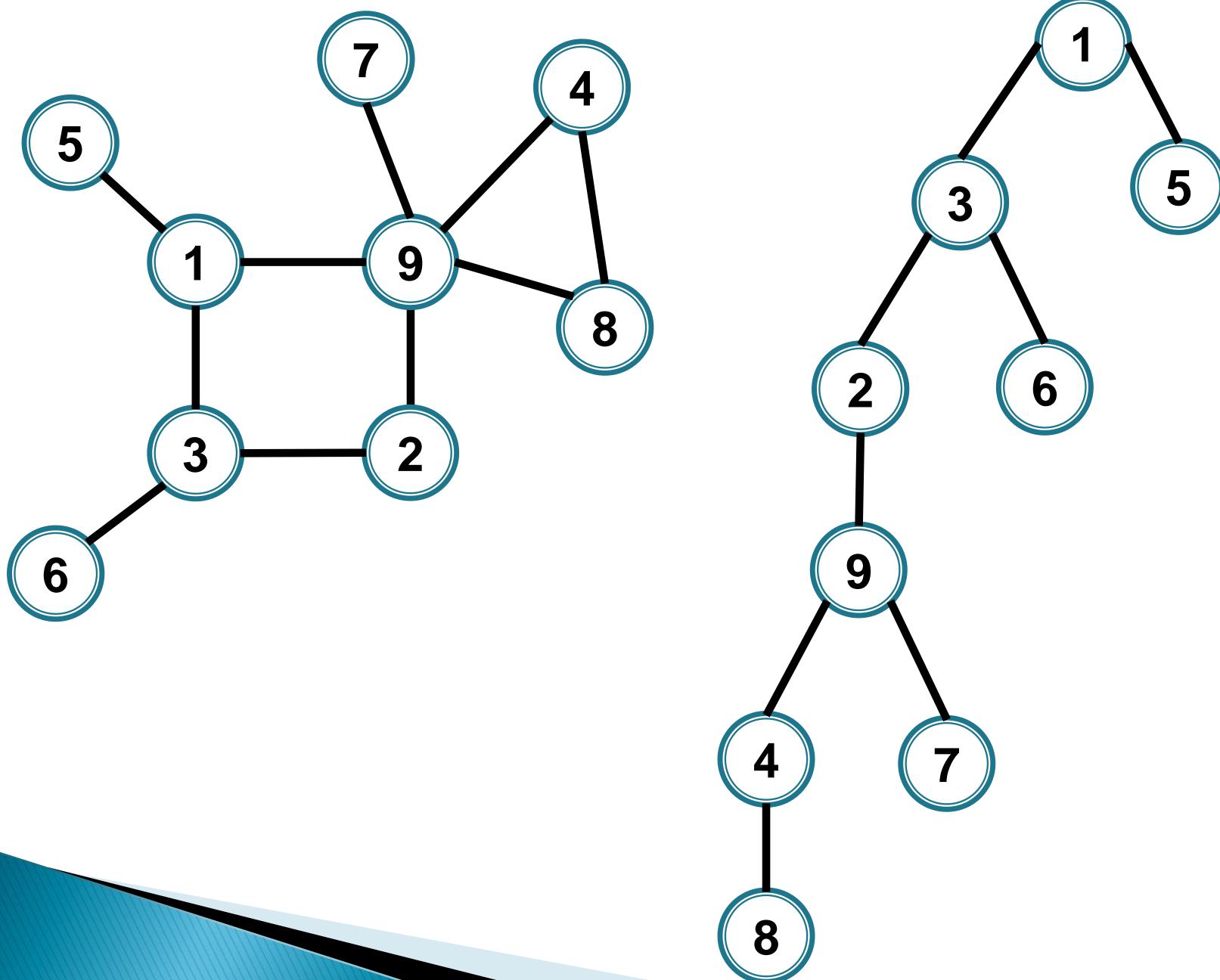






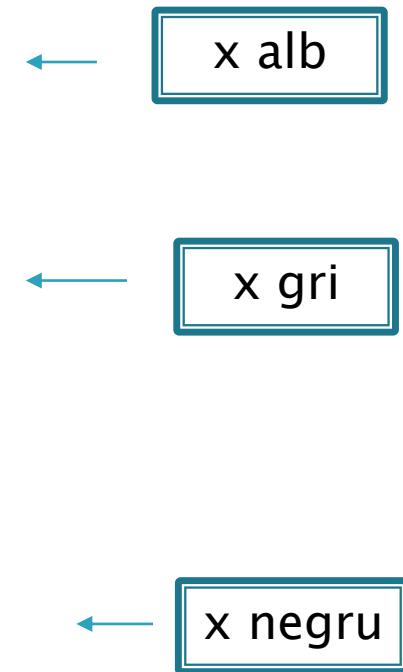






DFS (x)

```
//incepe explorarea varfului x  
viz[x] = 1  
pentru fiecare xy ∈ E //y vecin al lui x  
daca viz[y]==0 atunci  
    tata[y] = x  
    d[y] = d[x]+1 //nivel, nu distanta  
    DFS(y)  
//s-a finalizat explorarea varfului x
```



Apel:

DFS (s)

Proprietăți



Proprietăți DF

Starea unui vârf:

- **nevizitat** – culoarea albă
- **în explorare** – culoare gri
- **finalizat** – culoare neagră

Dacă memorăm culoarea fiecărui vârf, nu mai este necesar și vectorul vizitat, deoarece

$$\text{viz}[x] = 0 \Leftrightarrow \text{culoare}[x] = \text{alb}$$

Proprietăți DF

Dacă menținem pe parcursul algoritmului o variabilă de timp care se modifică atunci când se schimbă starea unui vârf (îl descoperim sau îl finalizăm), putem asocia fiecărui vârf două valori:

- ▶ **desc[v]** – momentul la care vârful a fost **descoperit**
(a devenit **gri**)
- ▶ **fin[v]** – momentul la care vârful a fost **finalizat**
(a devenit **negru**)

Proprietăți DF

Dacă dorim determinarea unor elemente de structură în graf precum circuite, componente tare conexe etc poate fi util să memorăm în parcurgerea DFS culoarea fiecărui vârf și/sau momentul în care a fost descoperit sau finalizat.

Un pseudocod complet în acest caz este următorul:

```

DFS(x)
    culoare[x] = gri
    timp = timp + 1
    desc[x]= timp; //incepe explorarea varfului x
    pentru fiecare xy ∈ E //y vecin al lui x
        daca culoare[y]==alb atunci
            tata[y] = x
            d[y] = d[x]+1 //nivel, nu distanta
            DFS(y)
    culoare[x] = negru
    timp = timp + 1
    fin[x] = timp //s-a finalizat explorarea varfului x

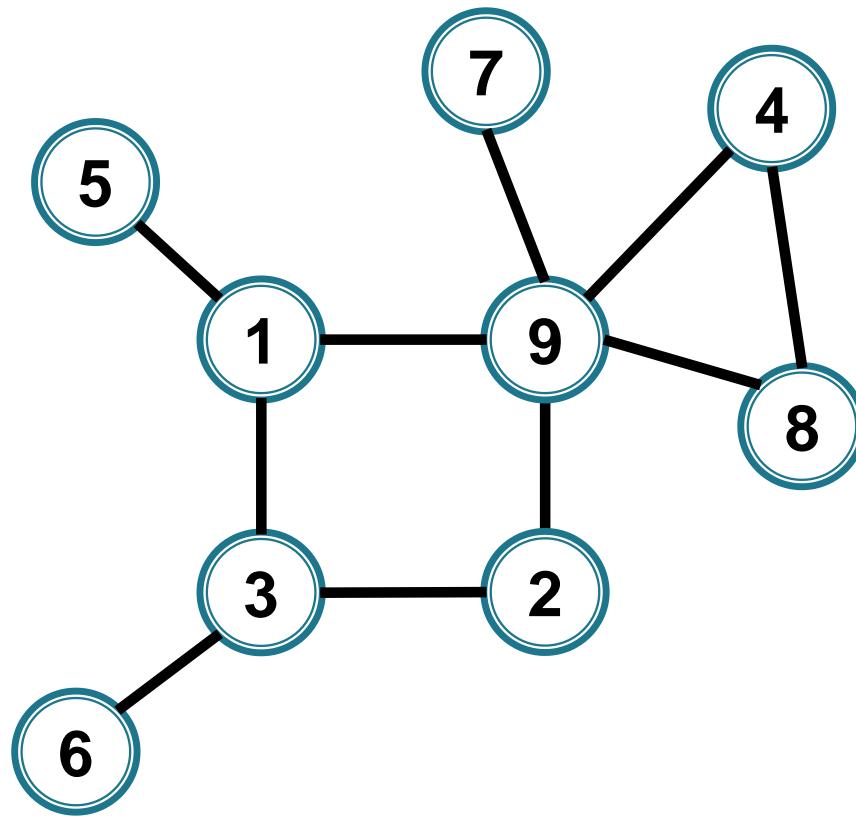
```

Apel:

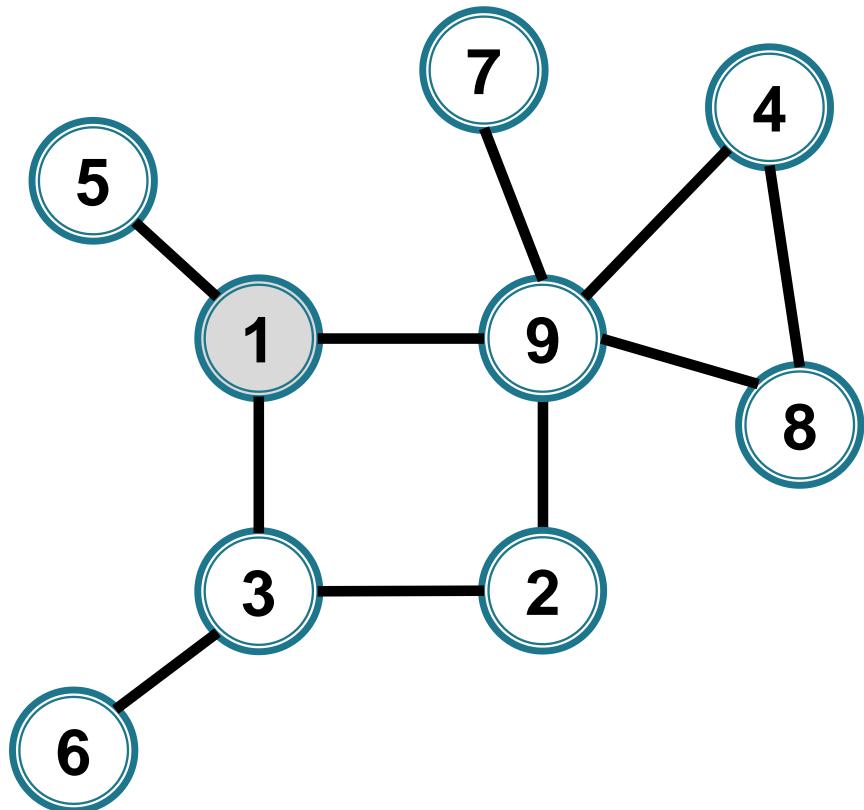
```

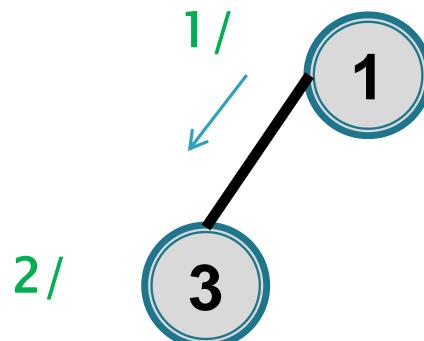
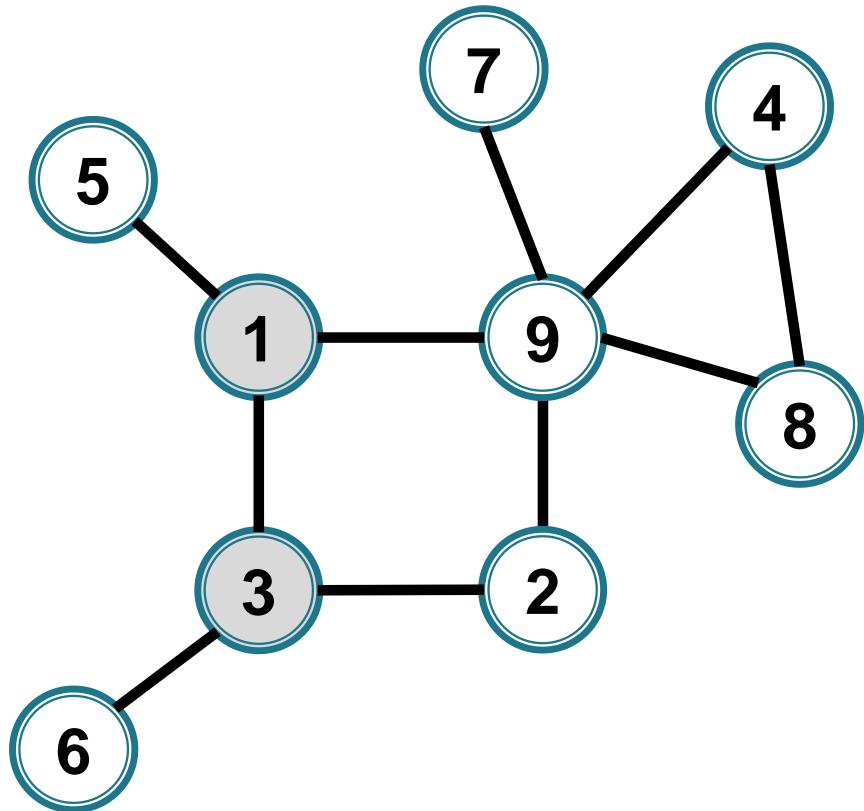
pentru fiecare x in V execută
    culoare[x] = alb
timp = 0
pentru fiecare x in V execută
    daca culoare[x] == alb atunci
        DFS(x)

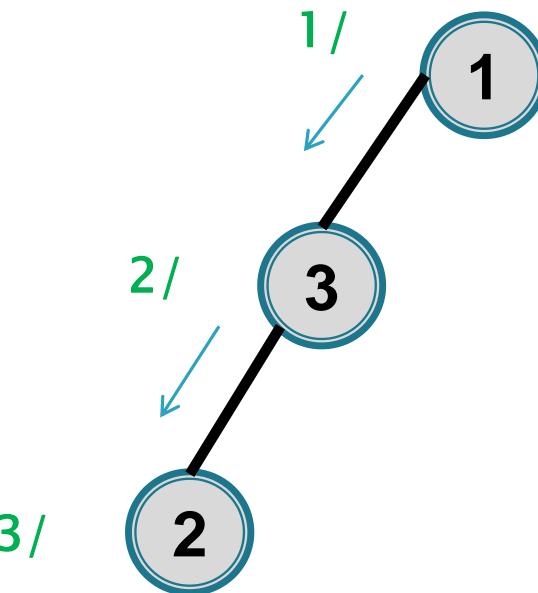
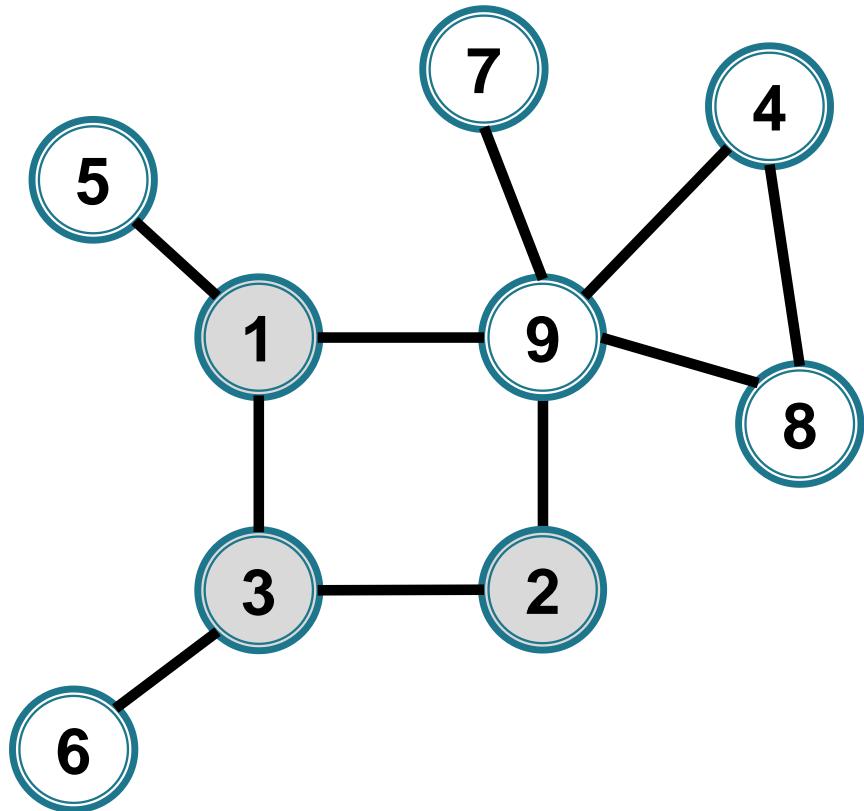
```

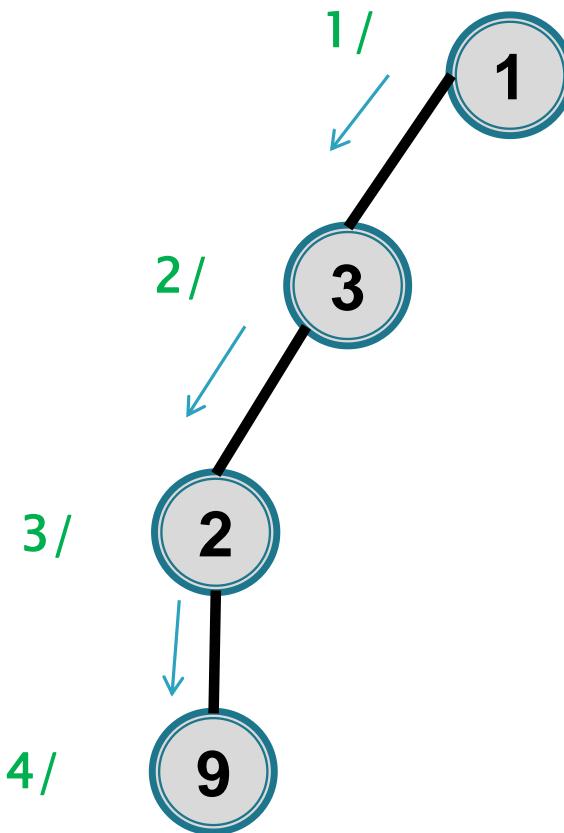
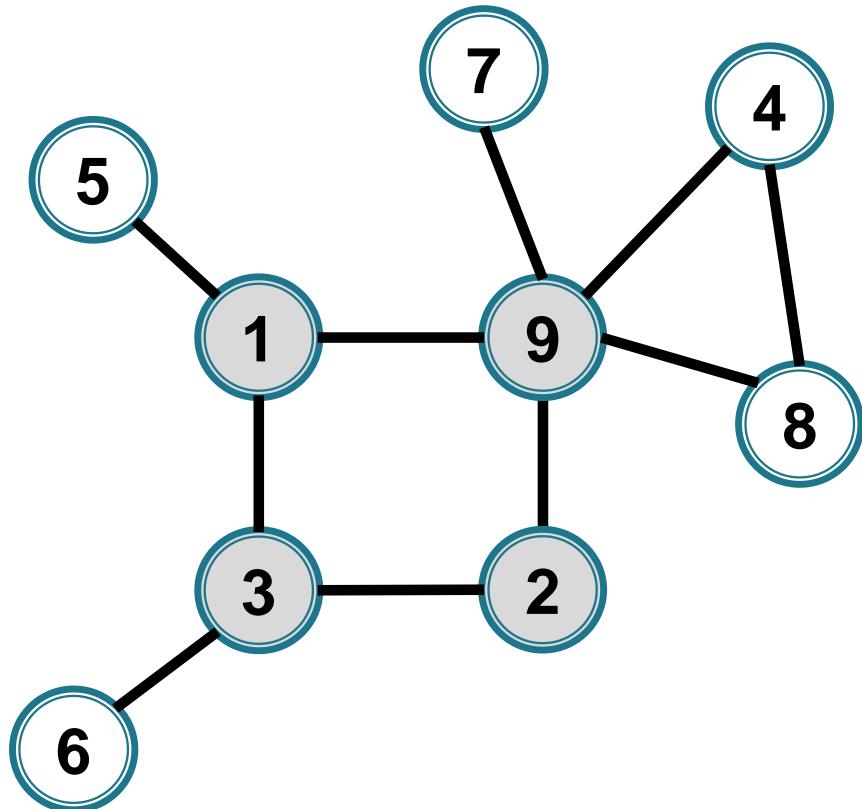


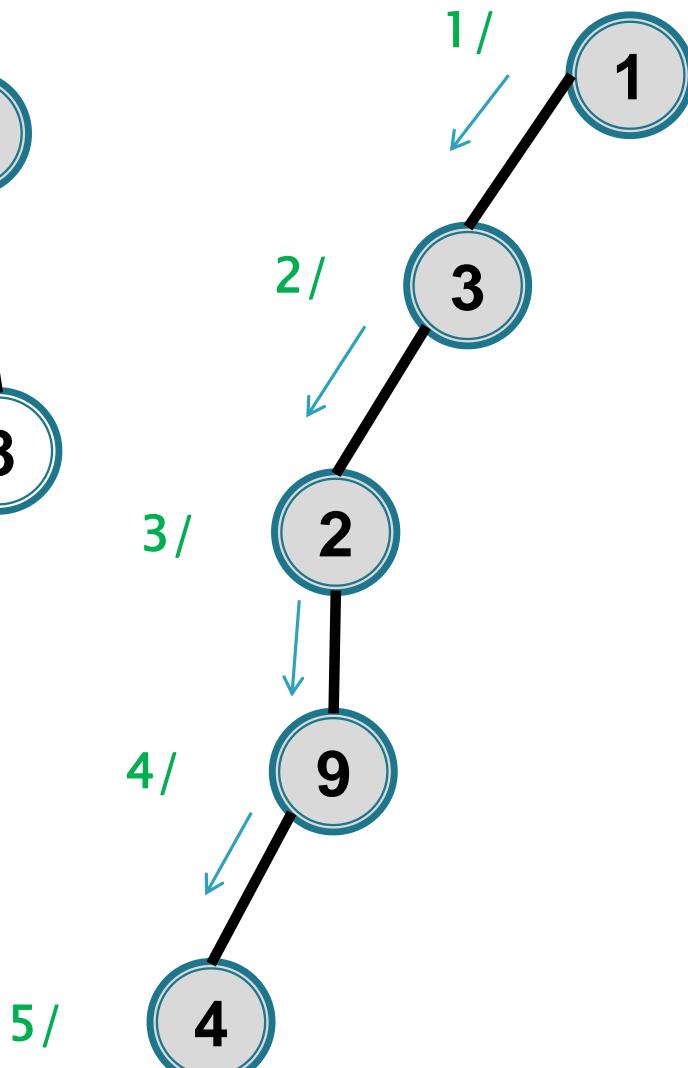
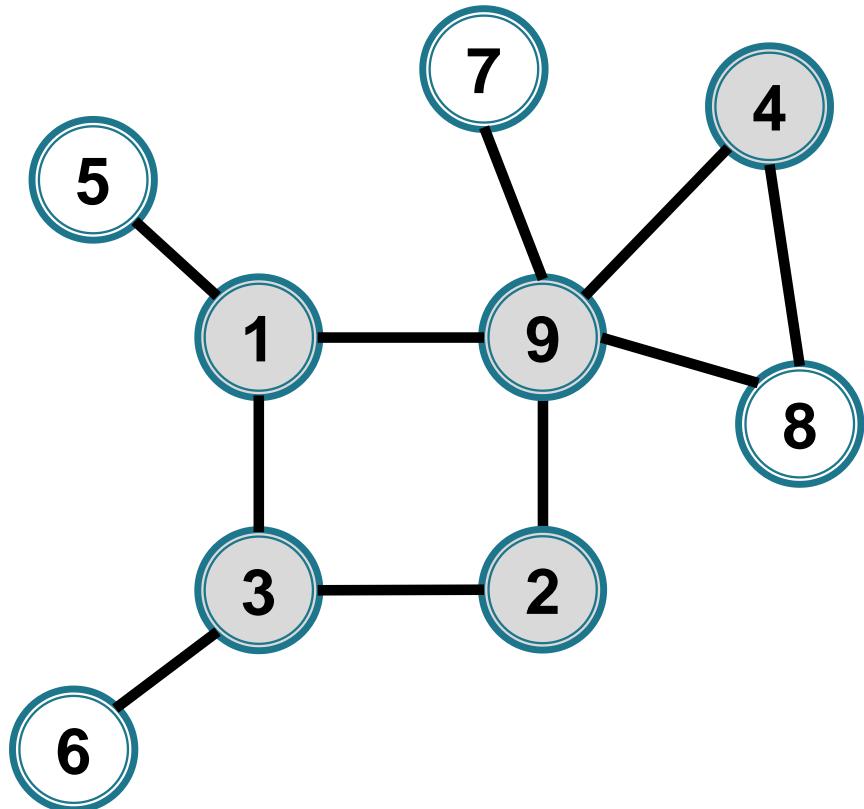
1 /

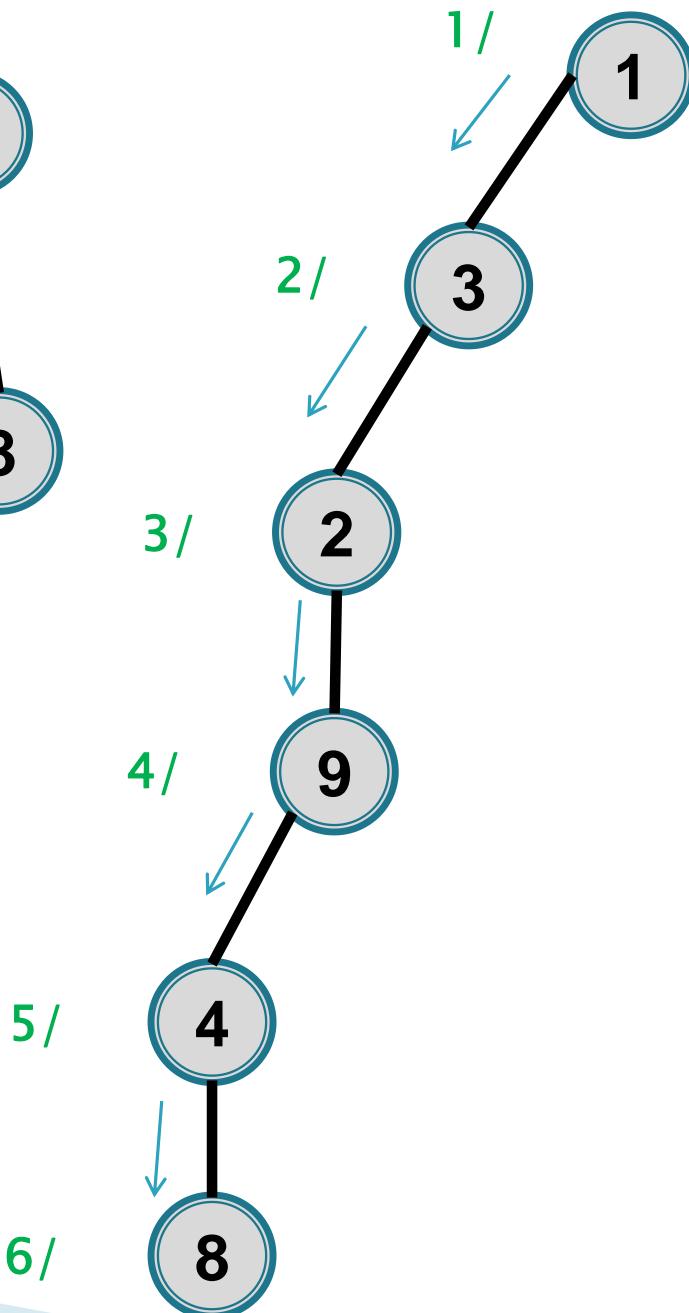
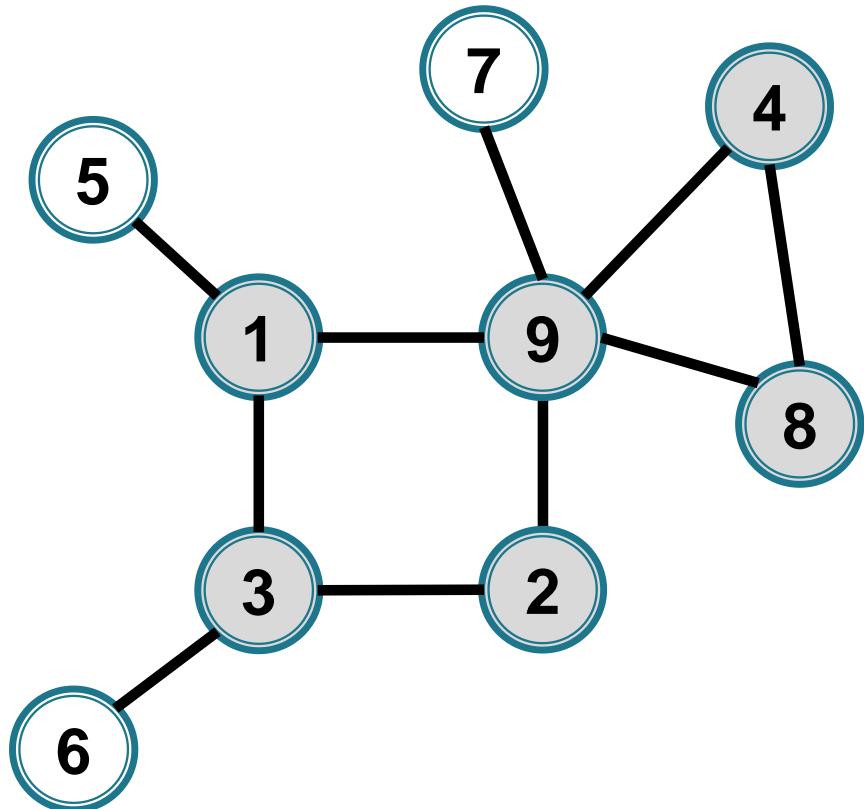


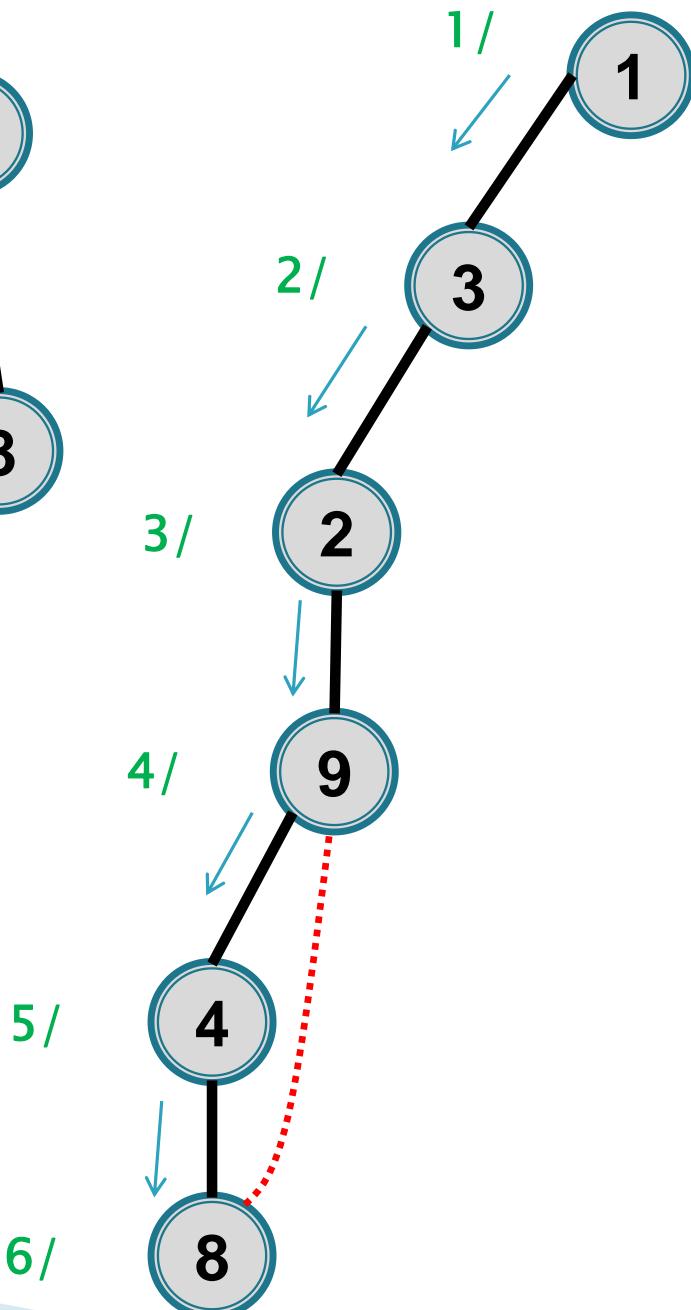
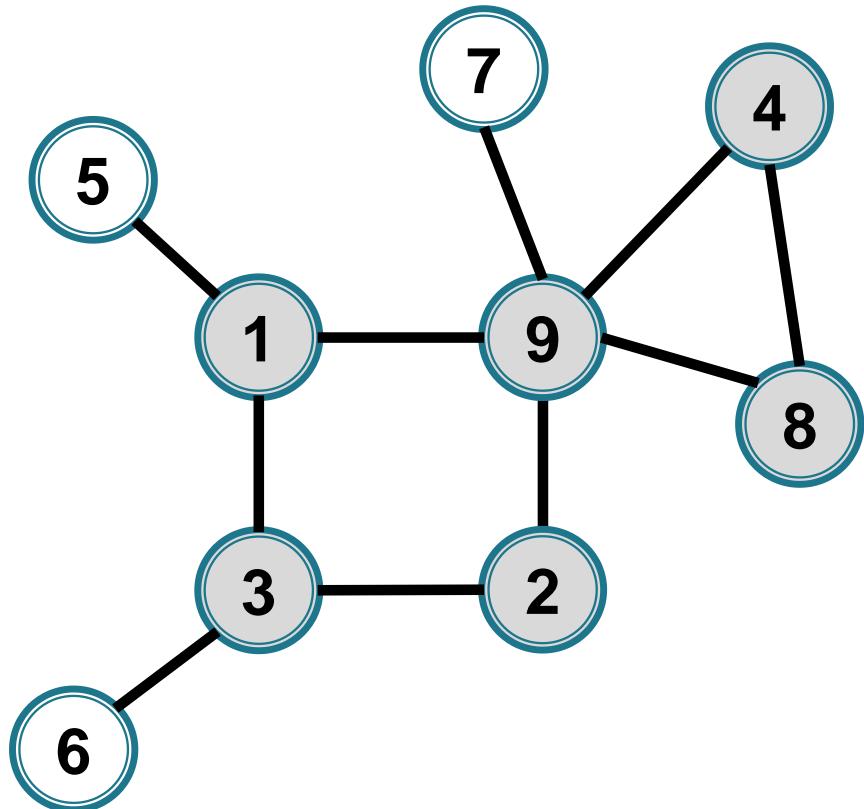


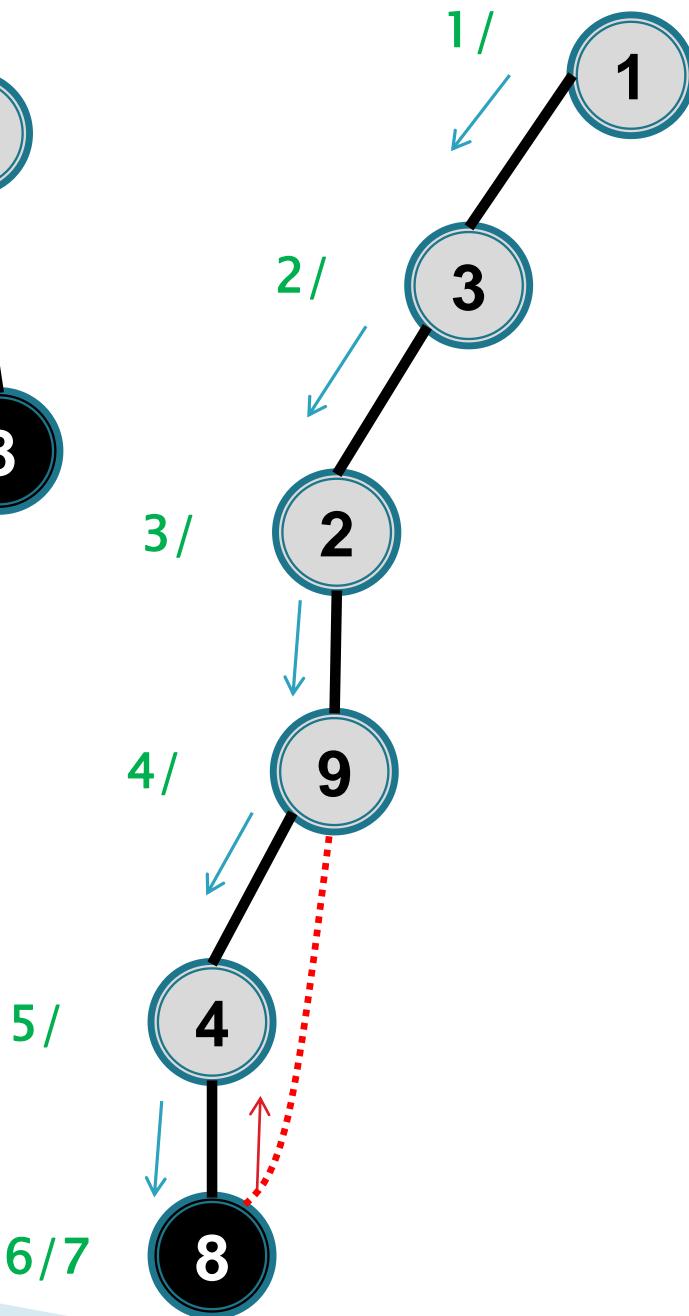
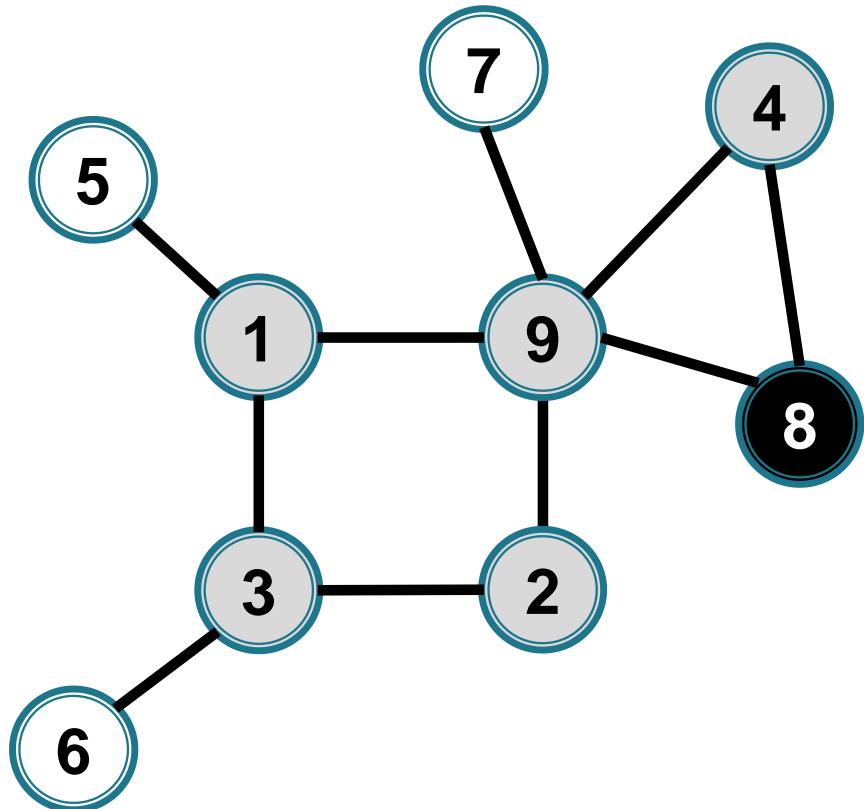


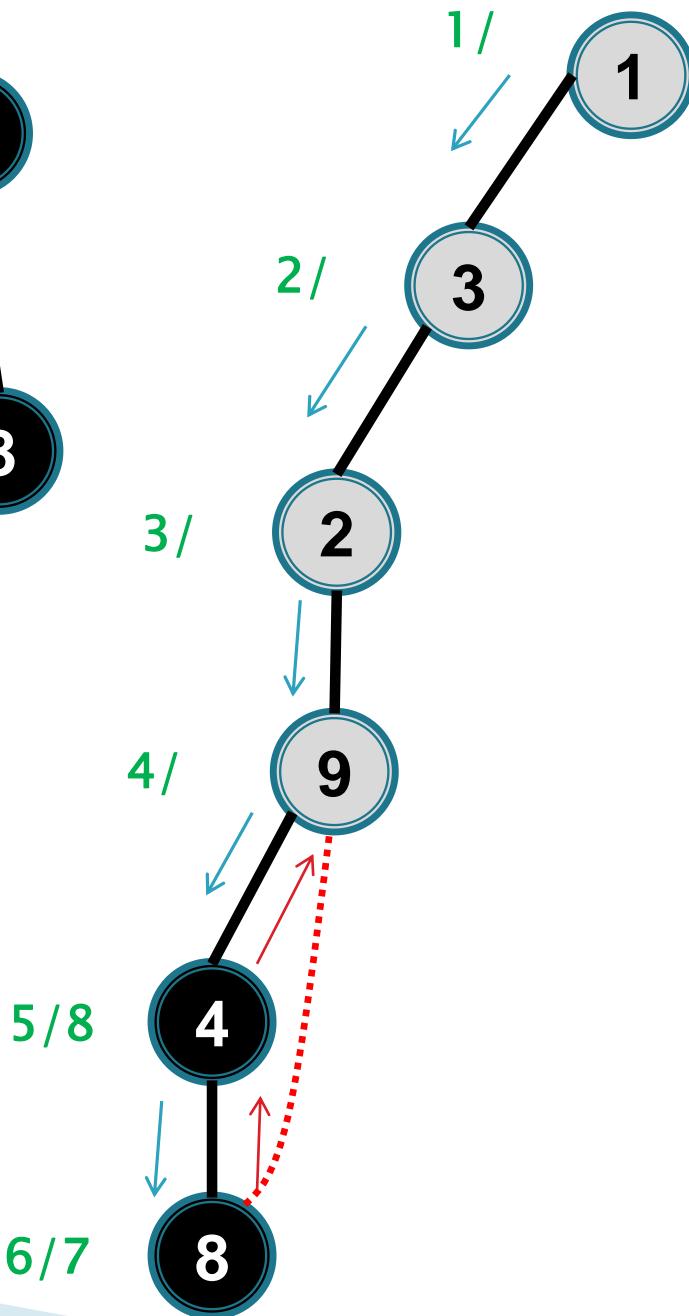
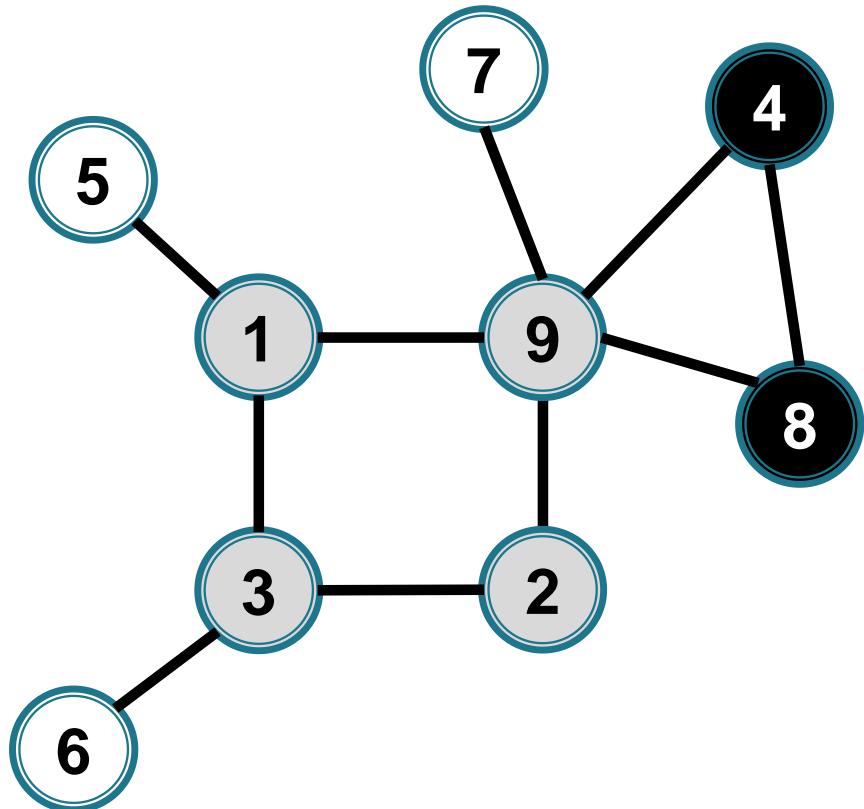


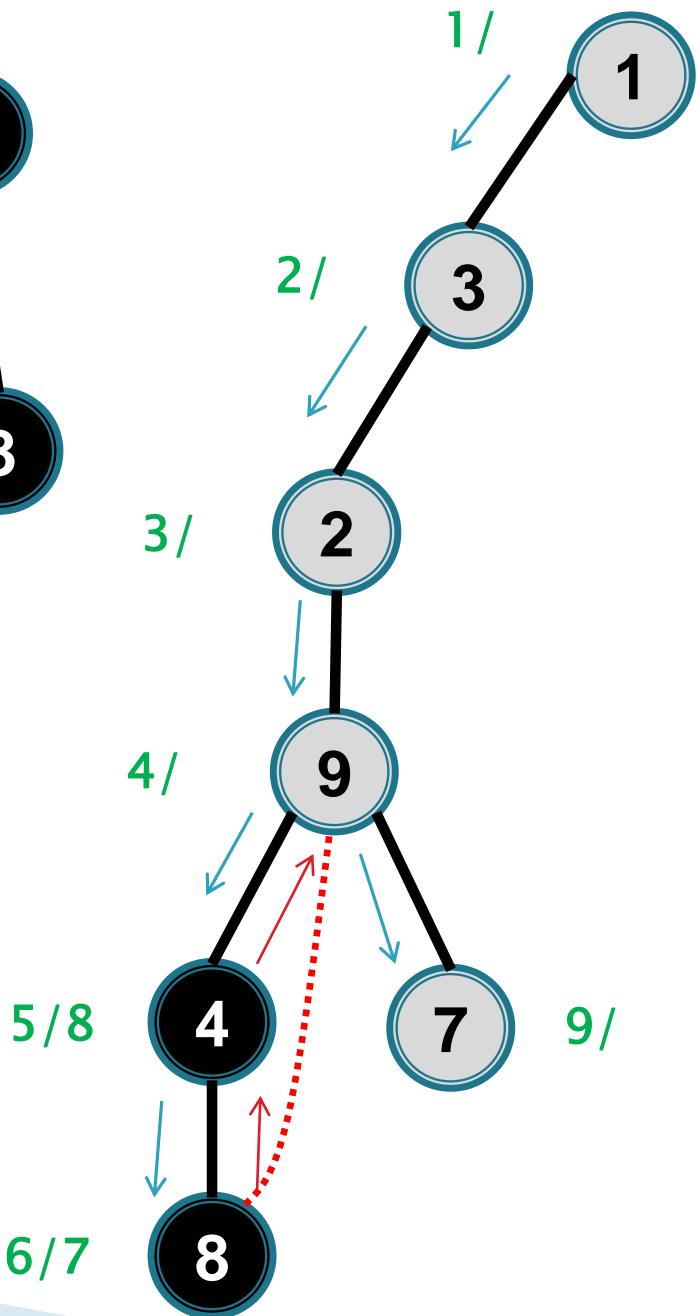
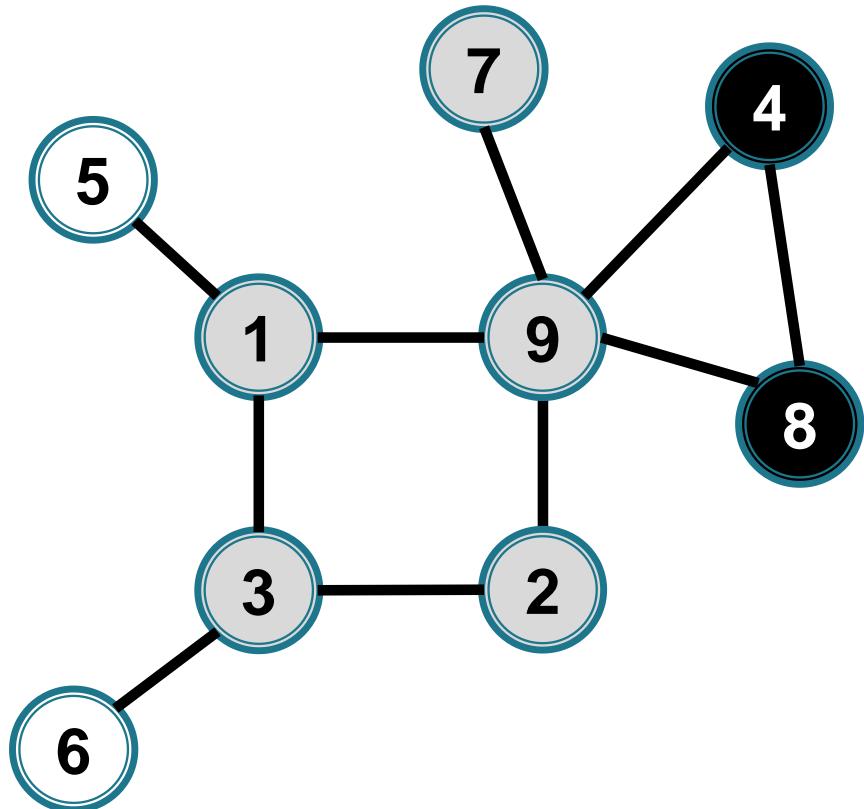


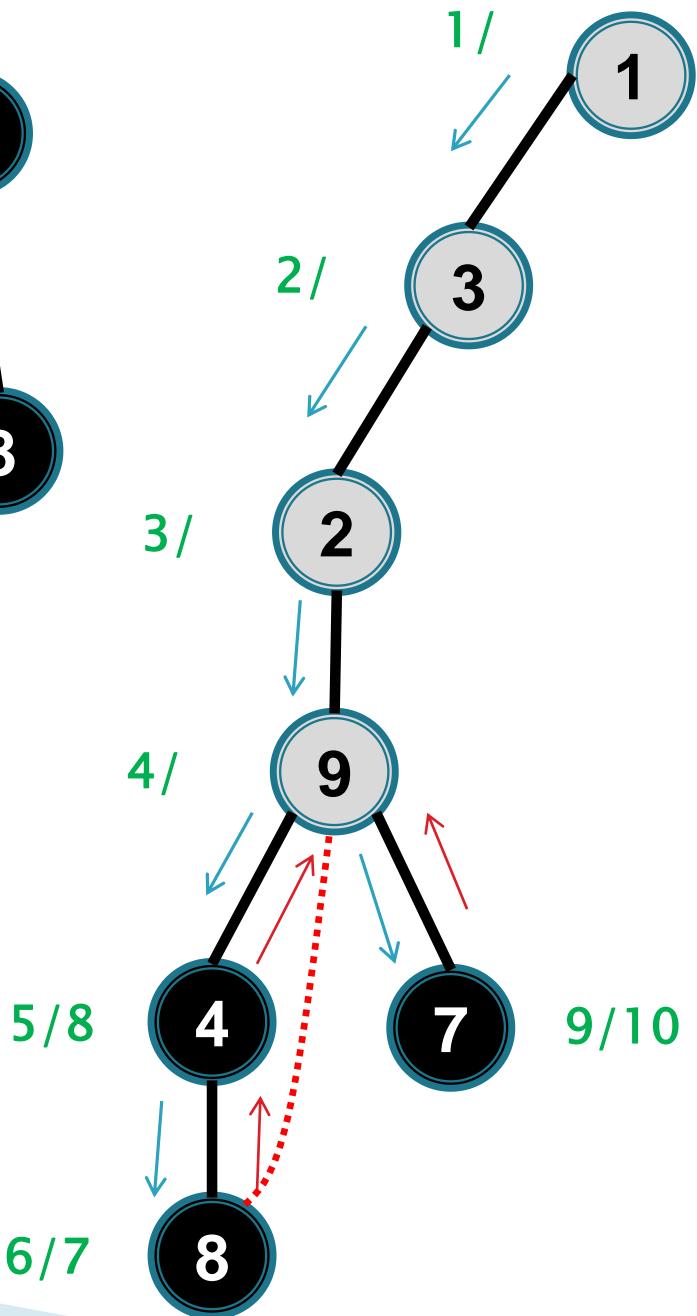
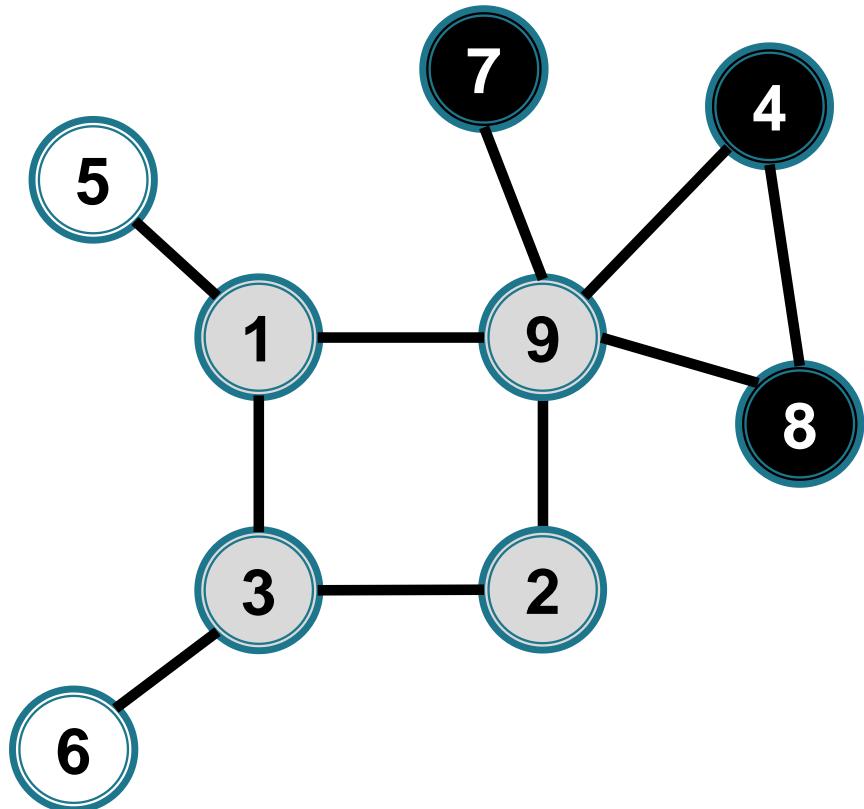


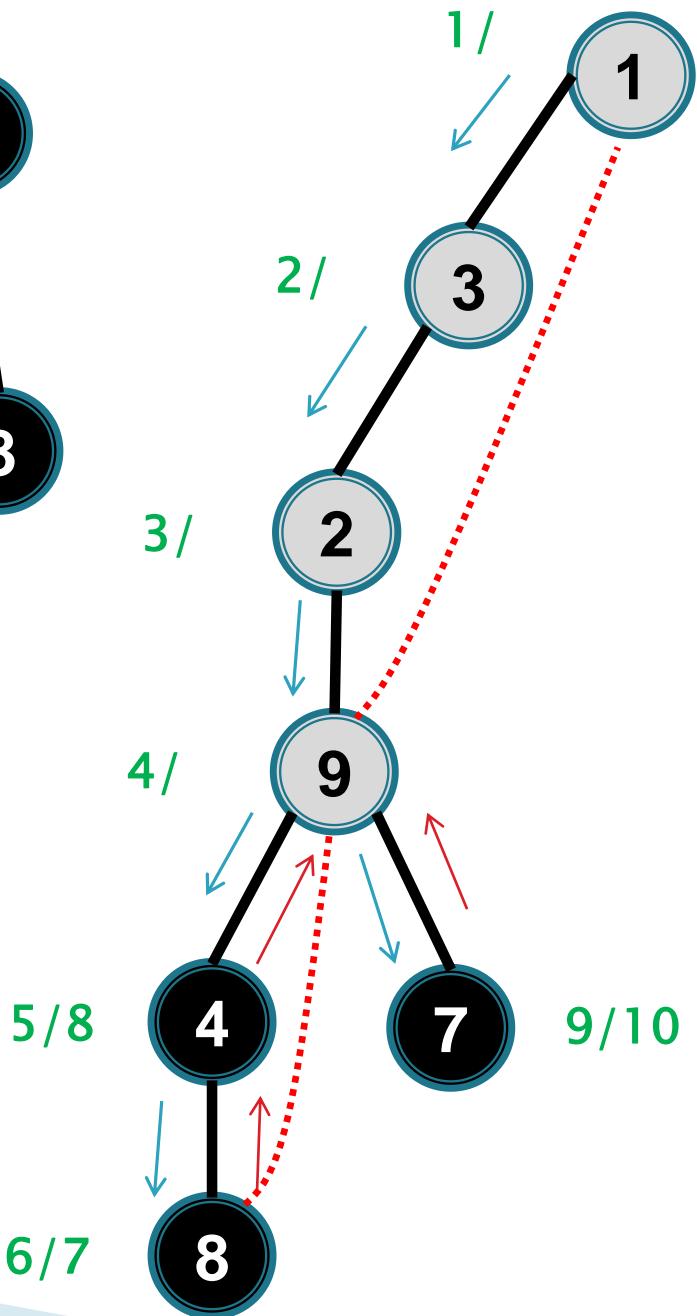
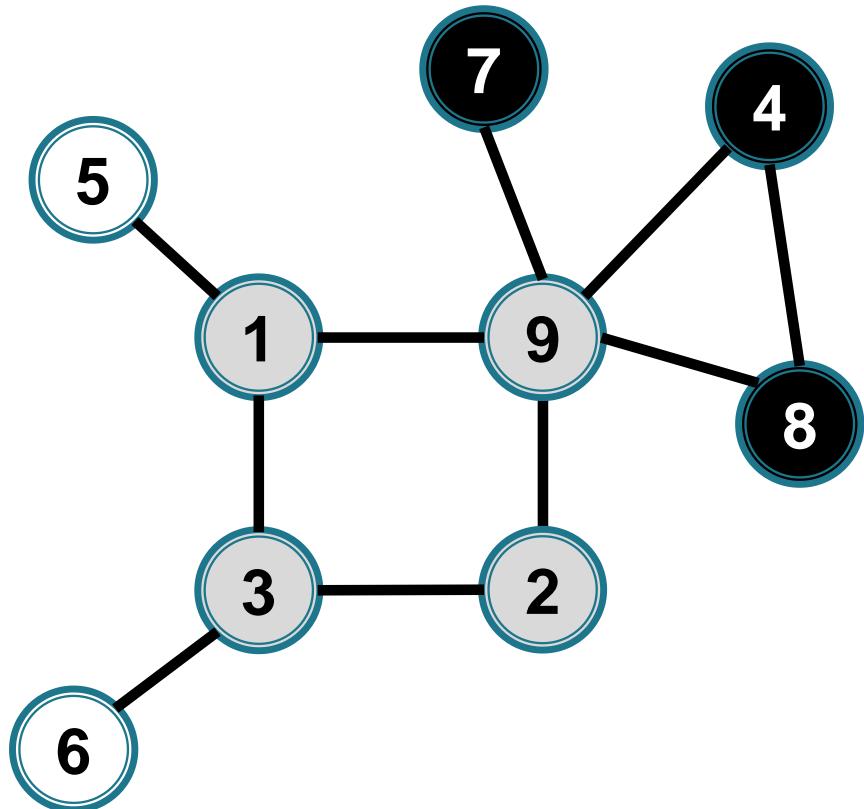


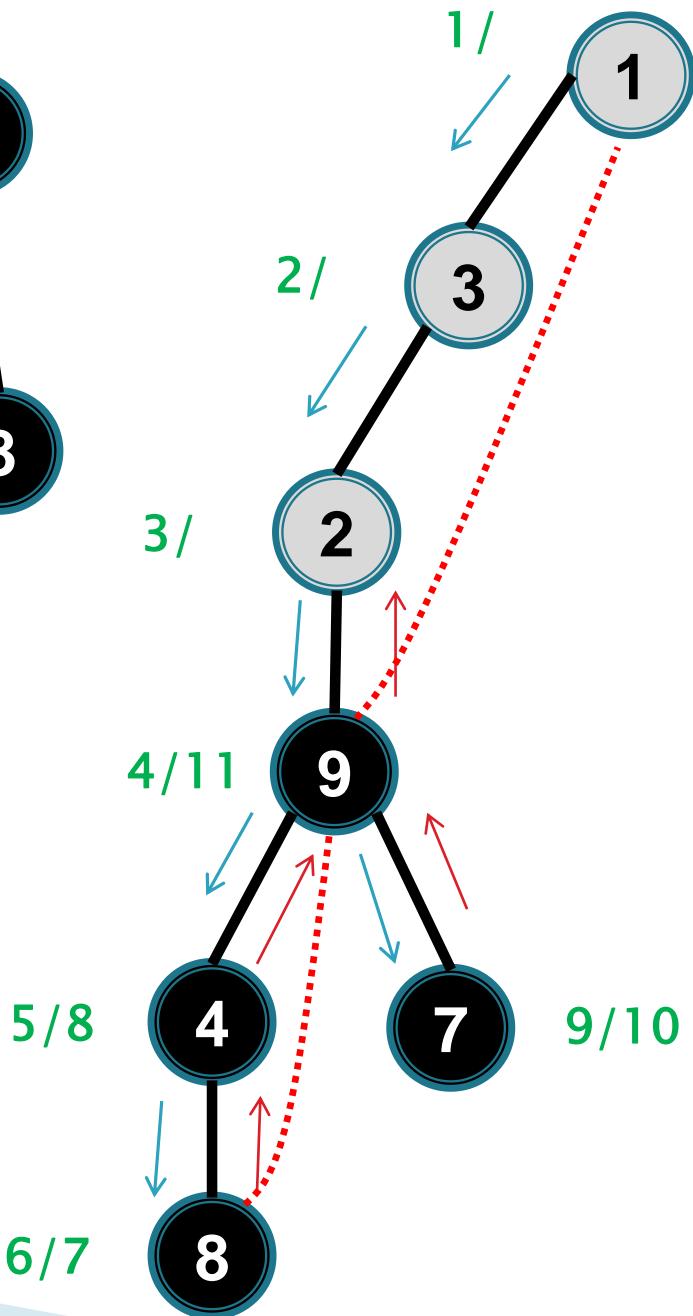
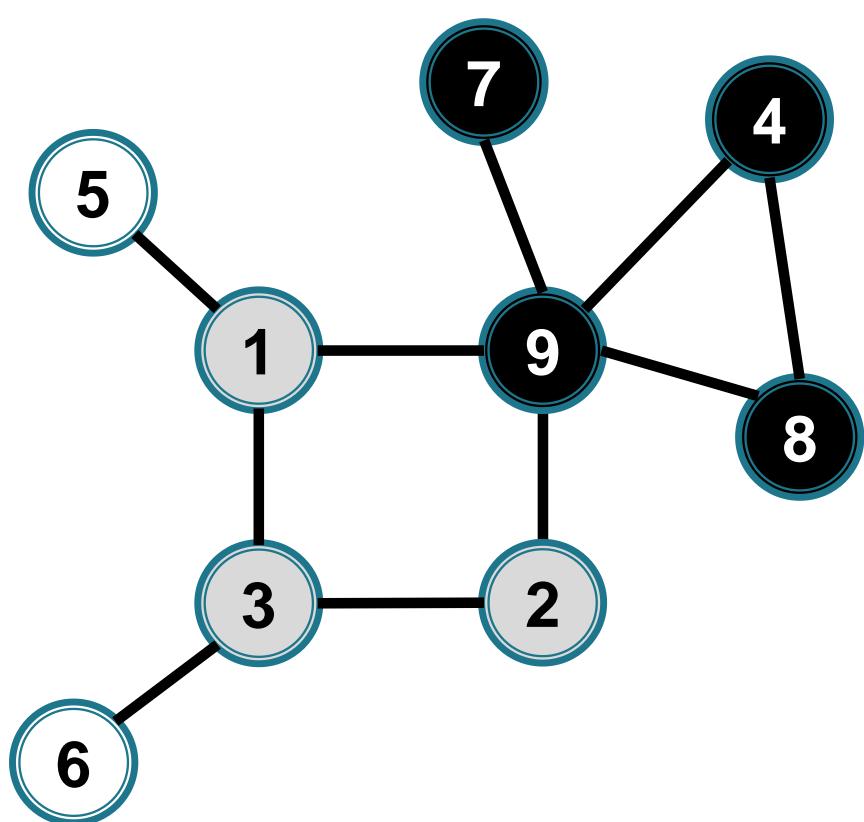


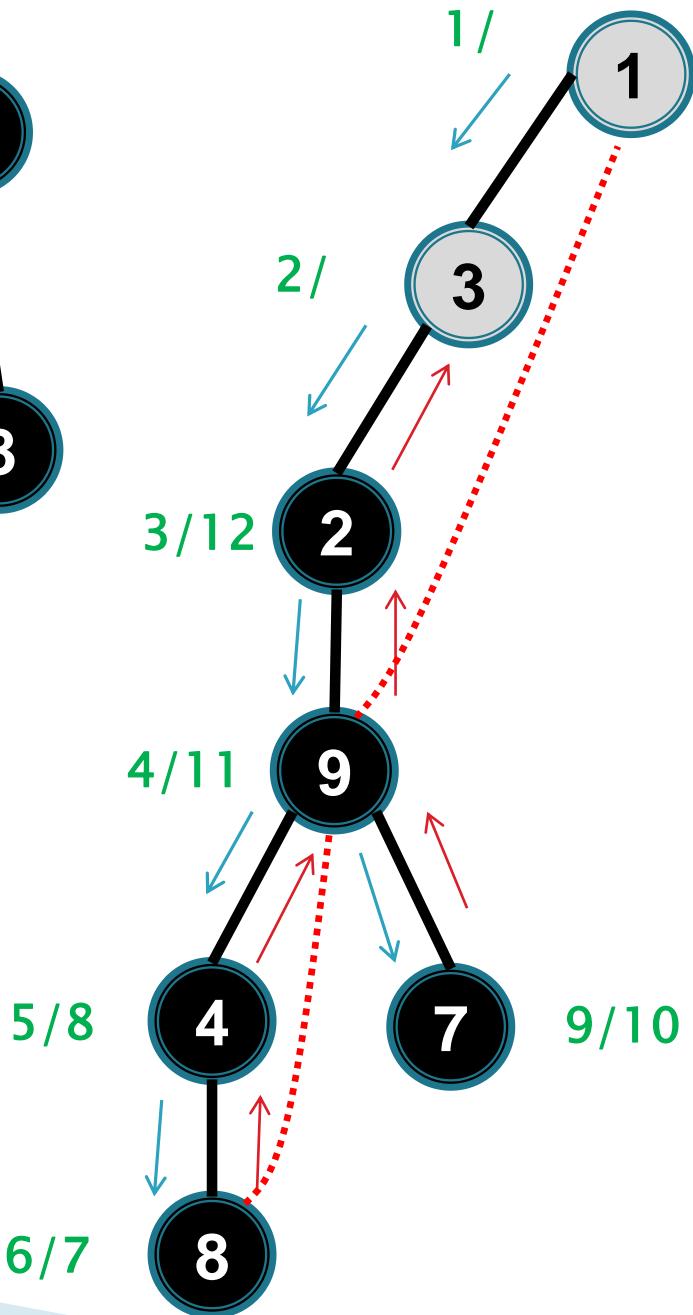
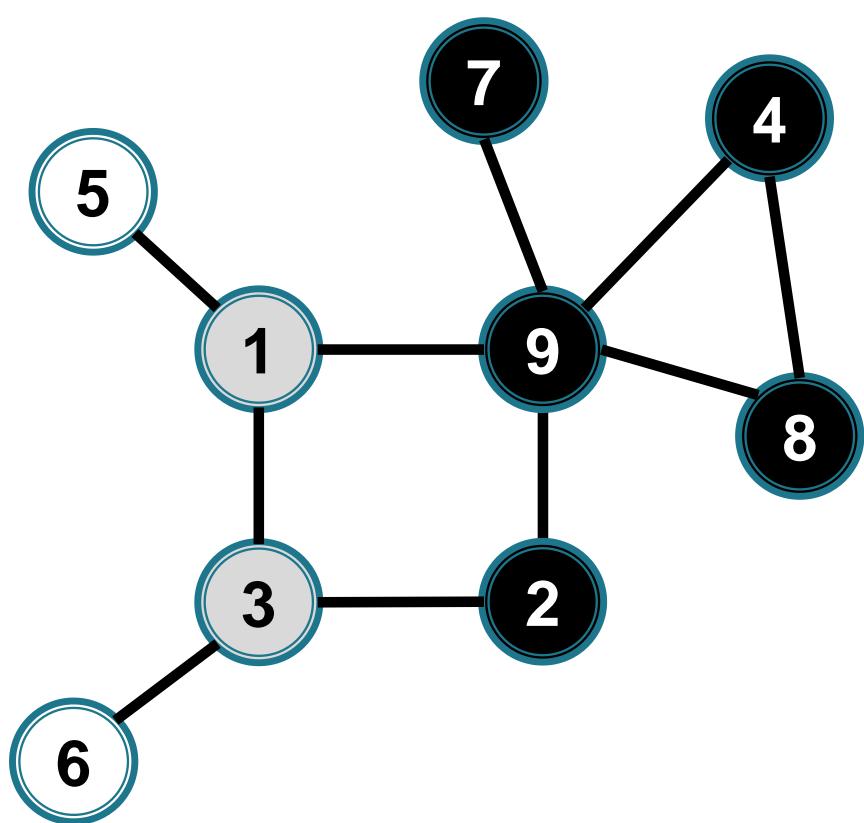


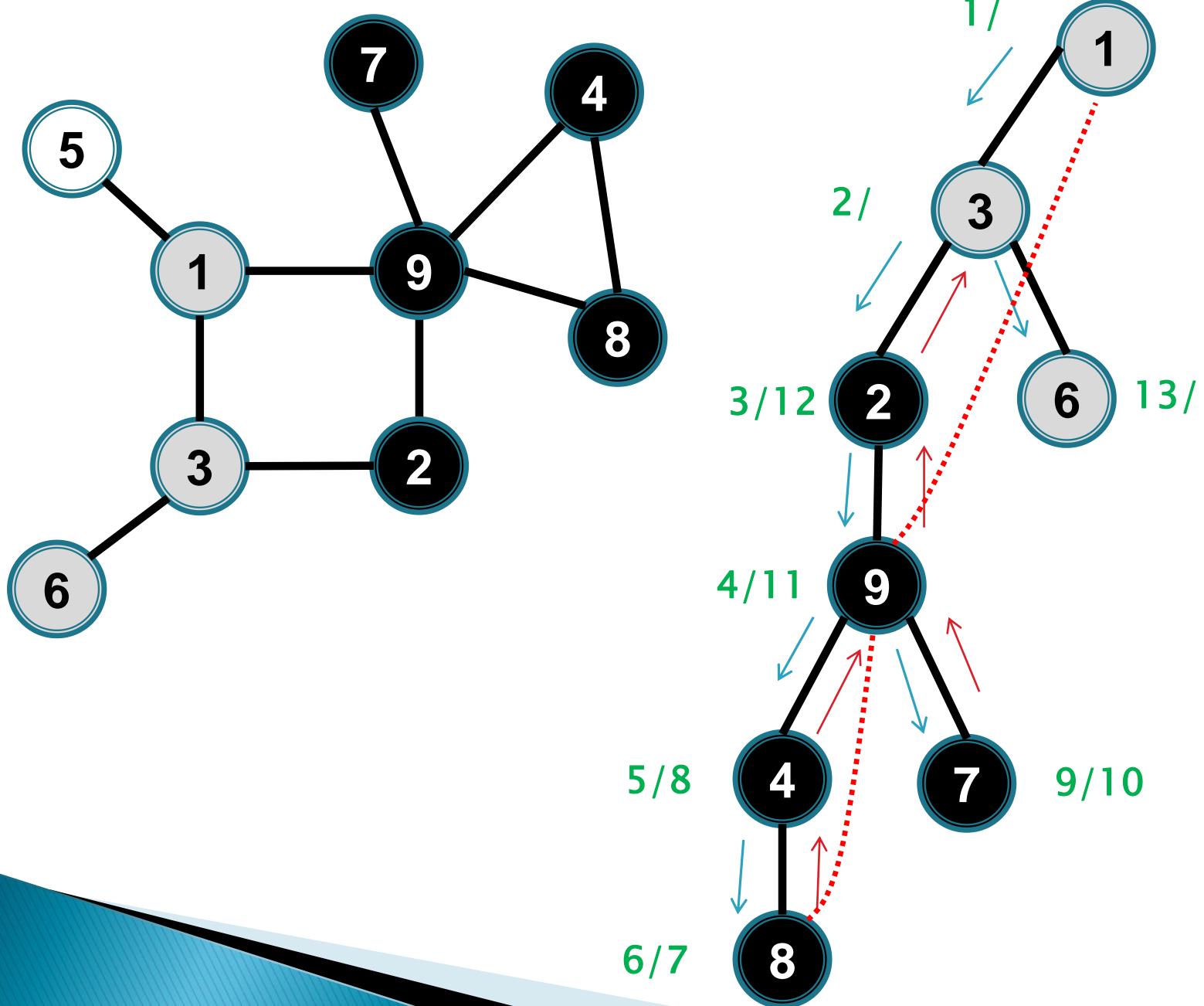


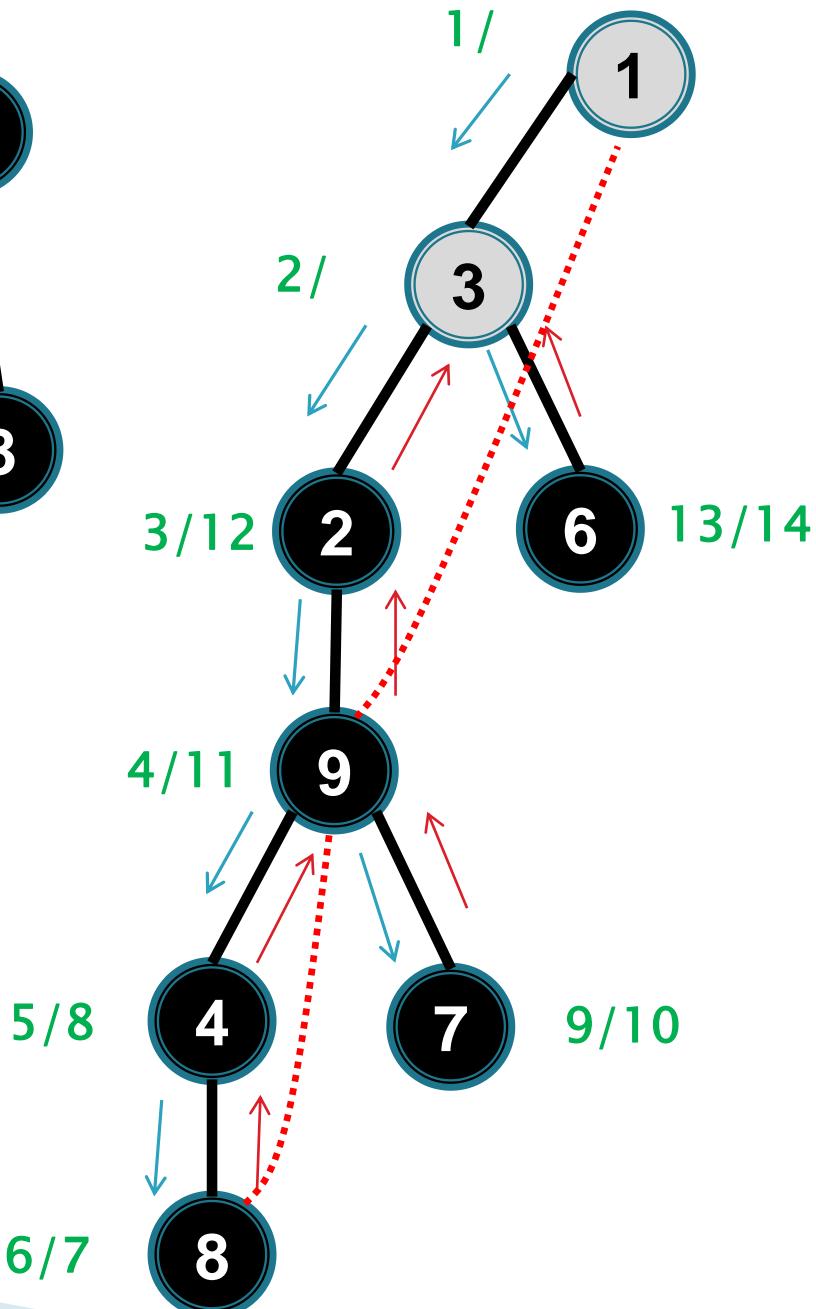
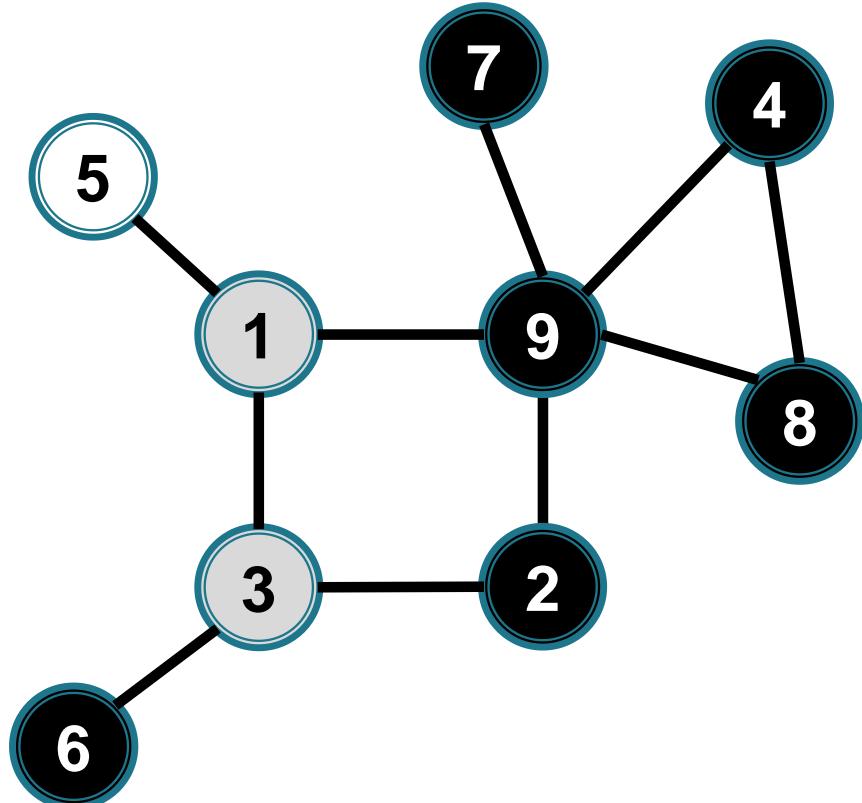


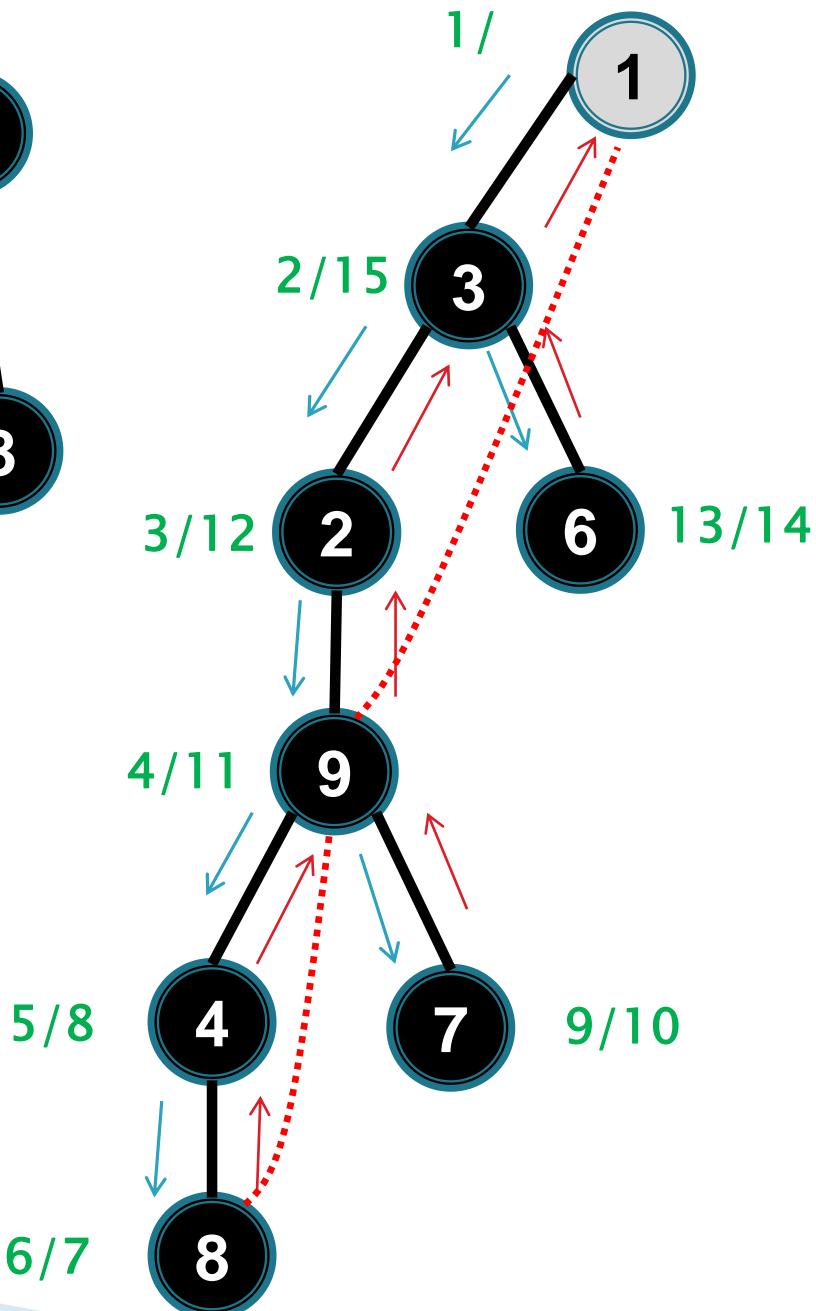
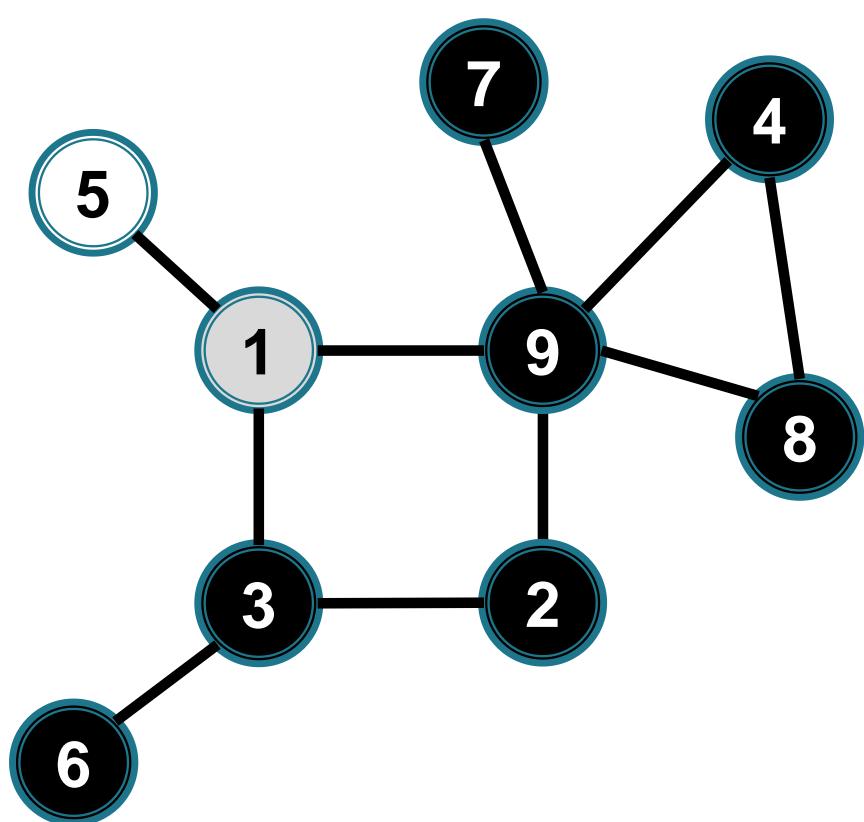


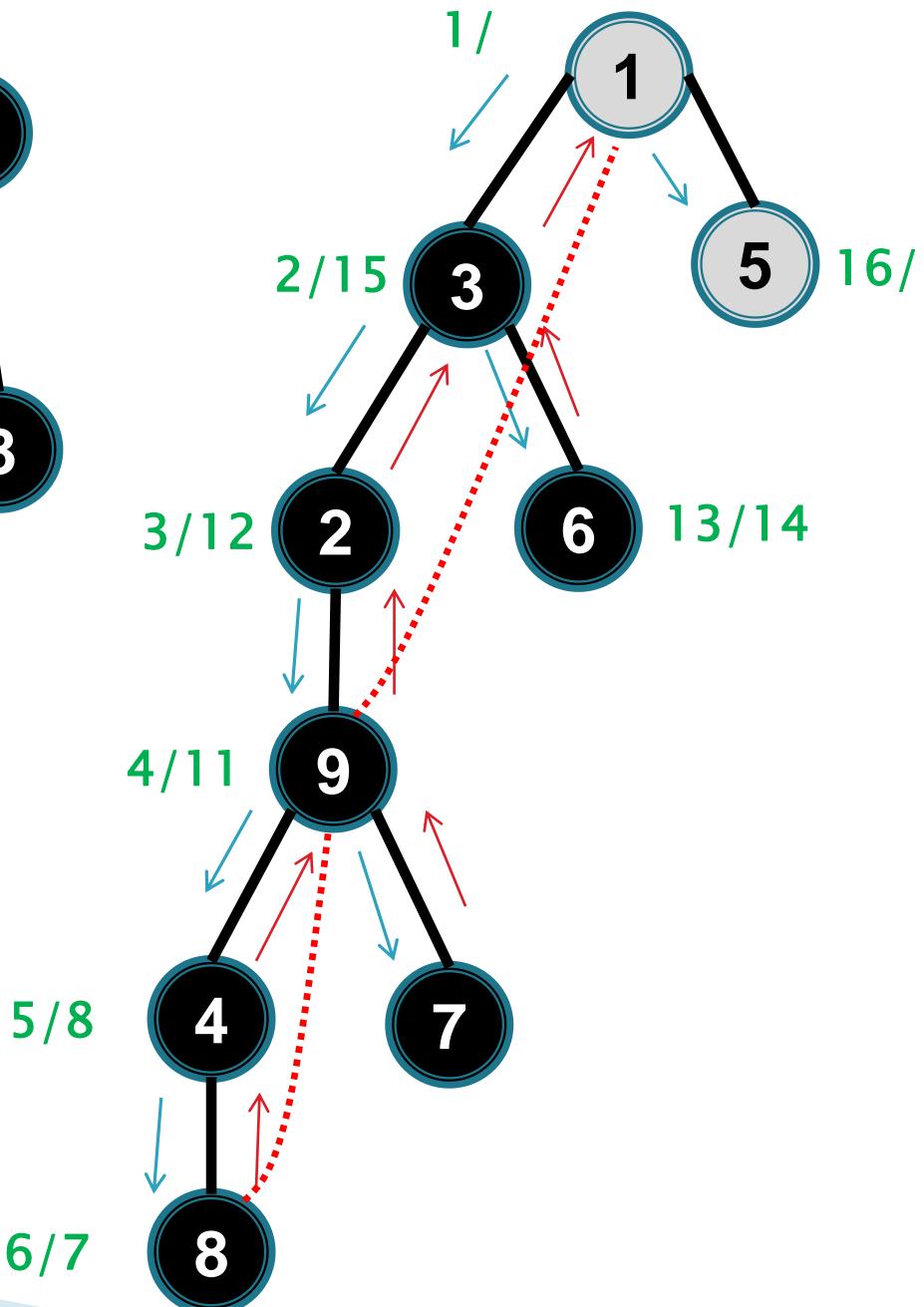
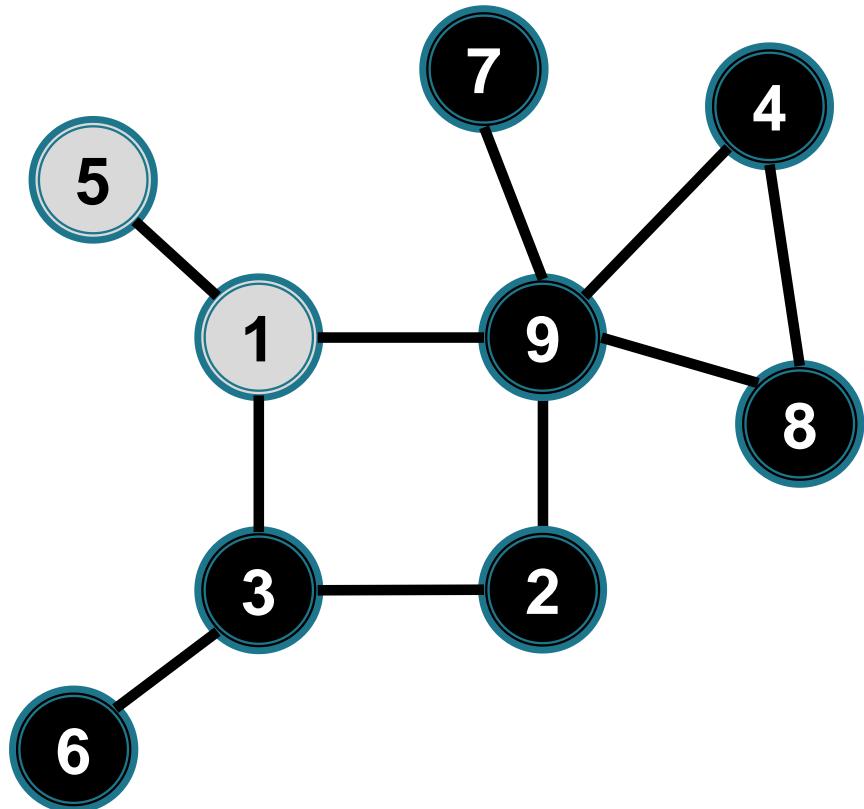


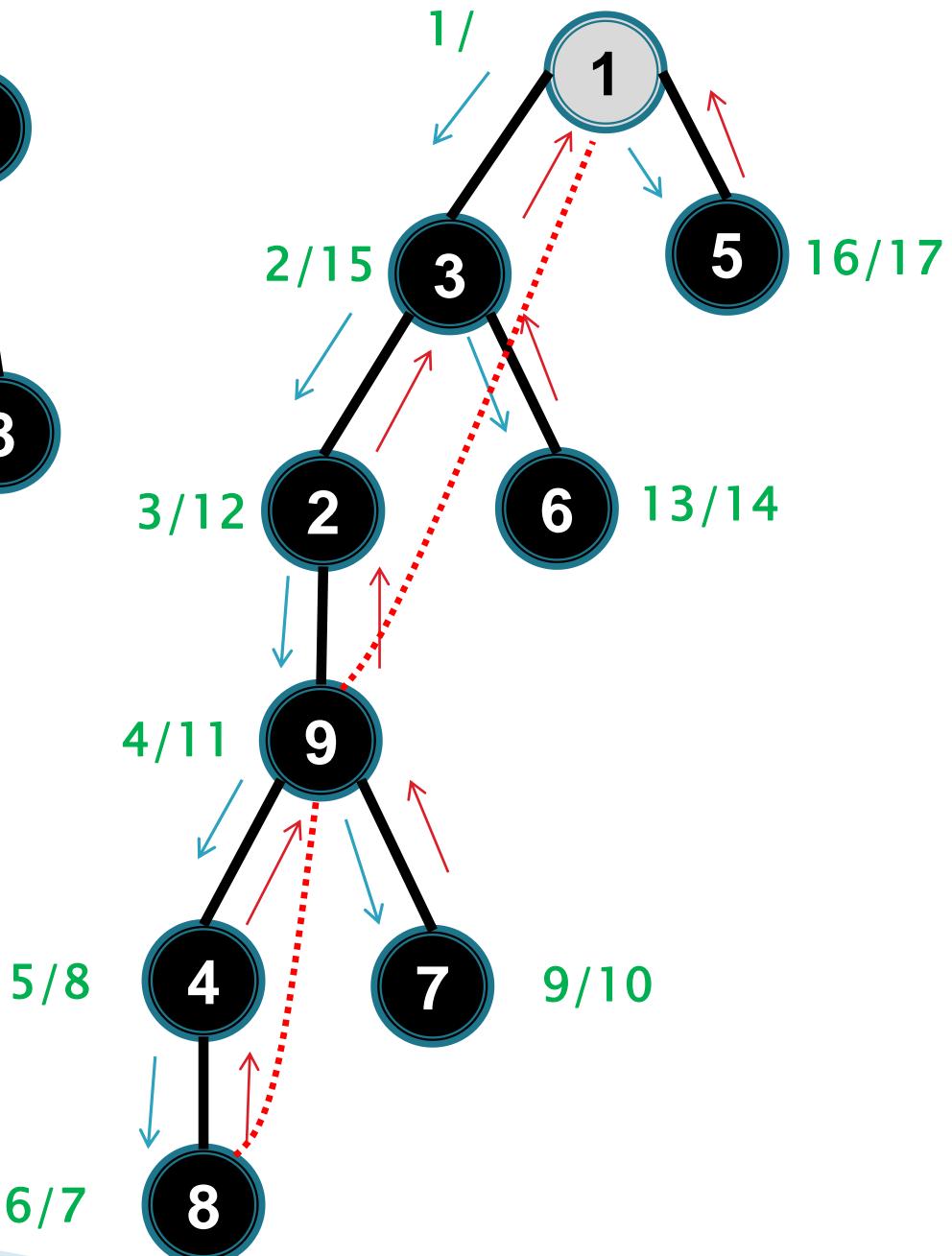
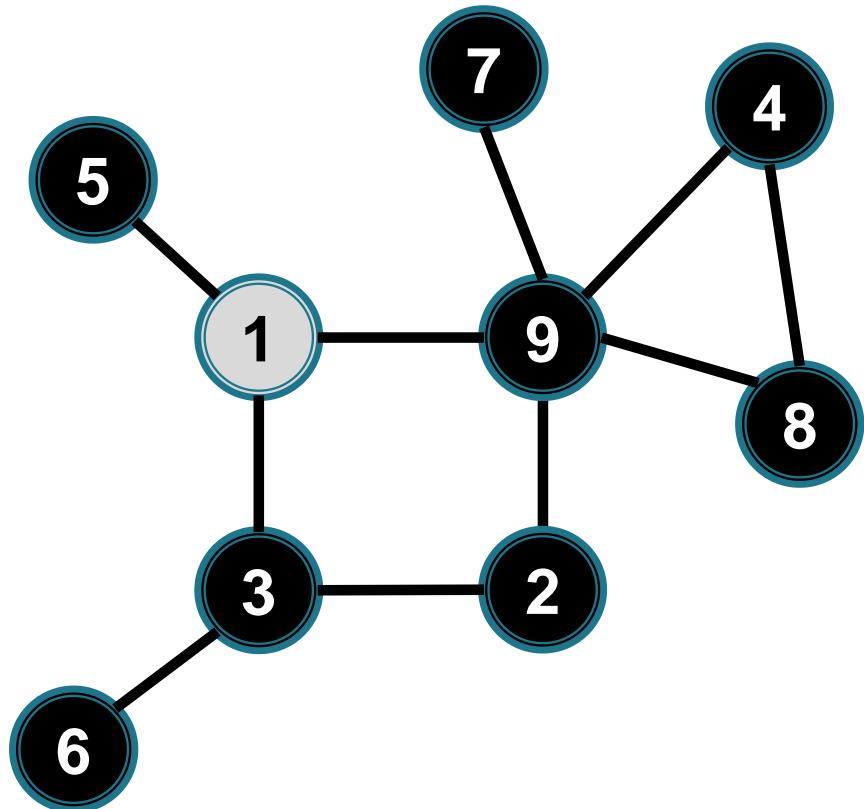


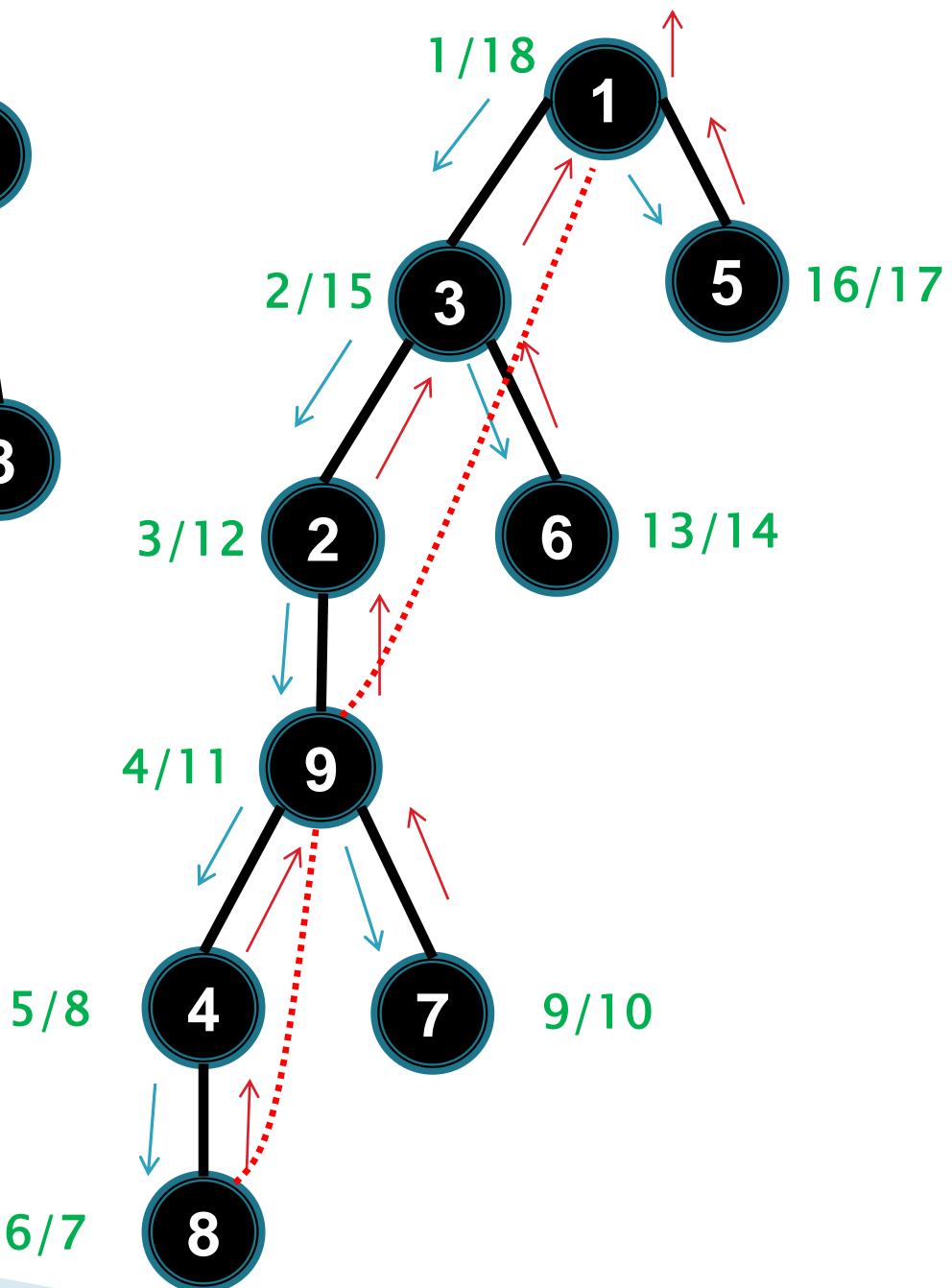
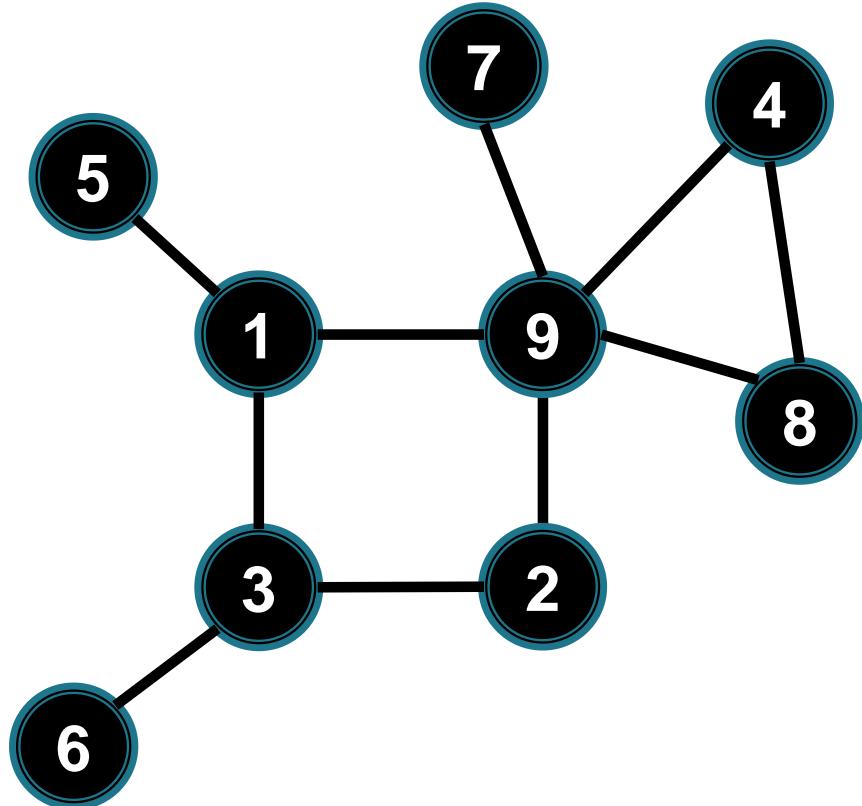




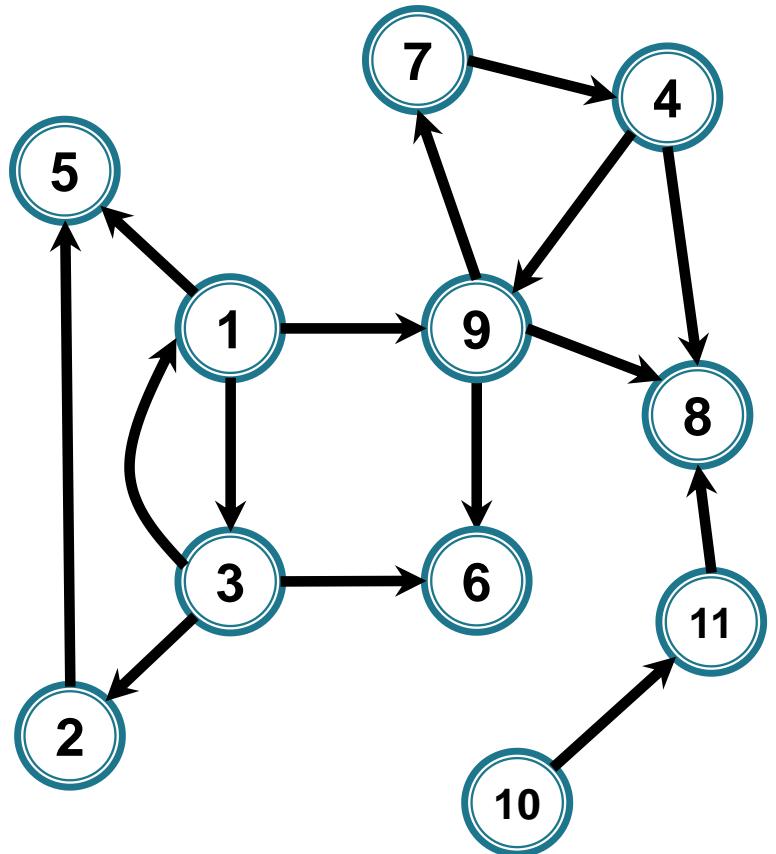




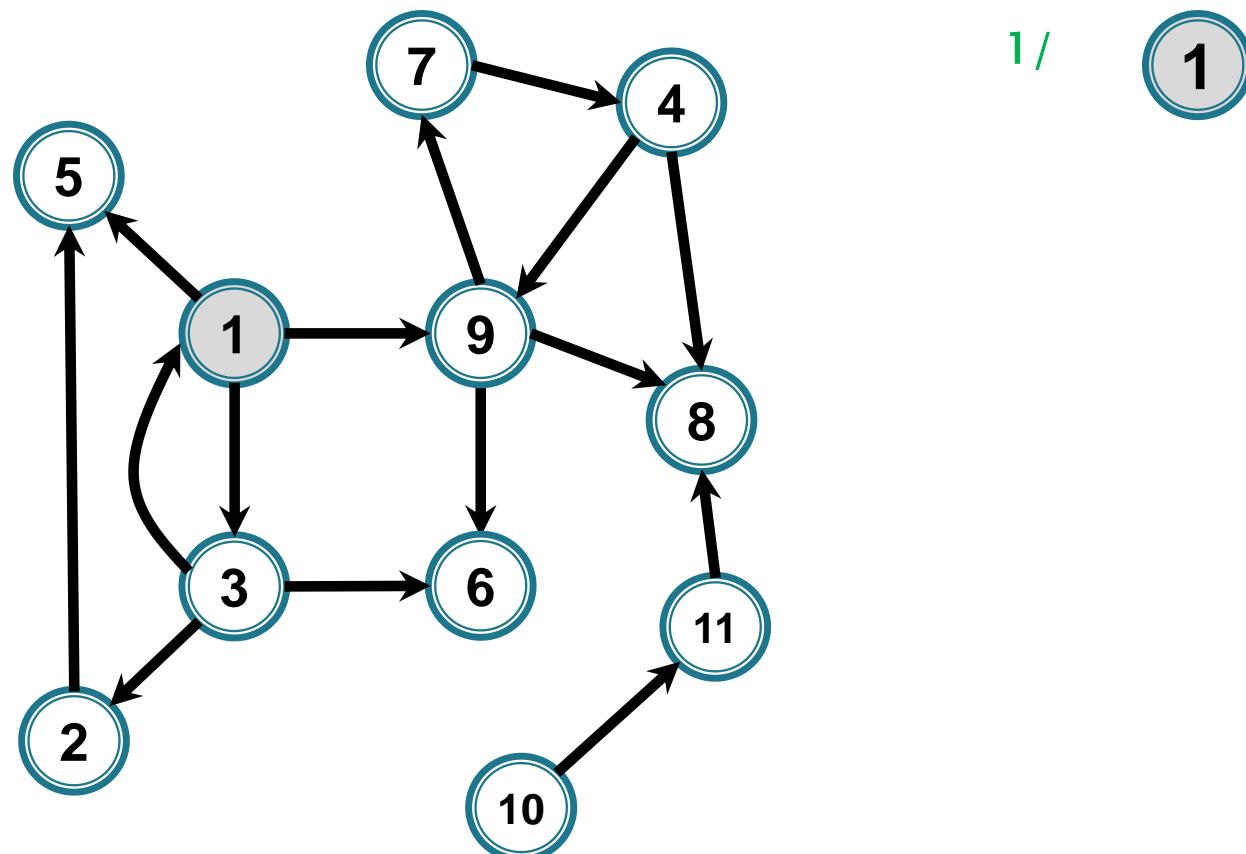




Exemplu DF graf orientat



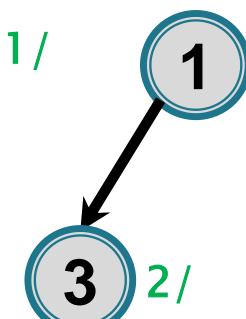
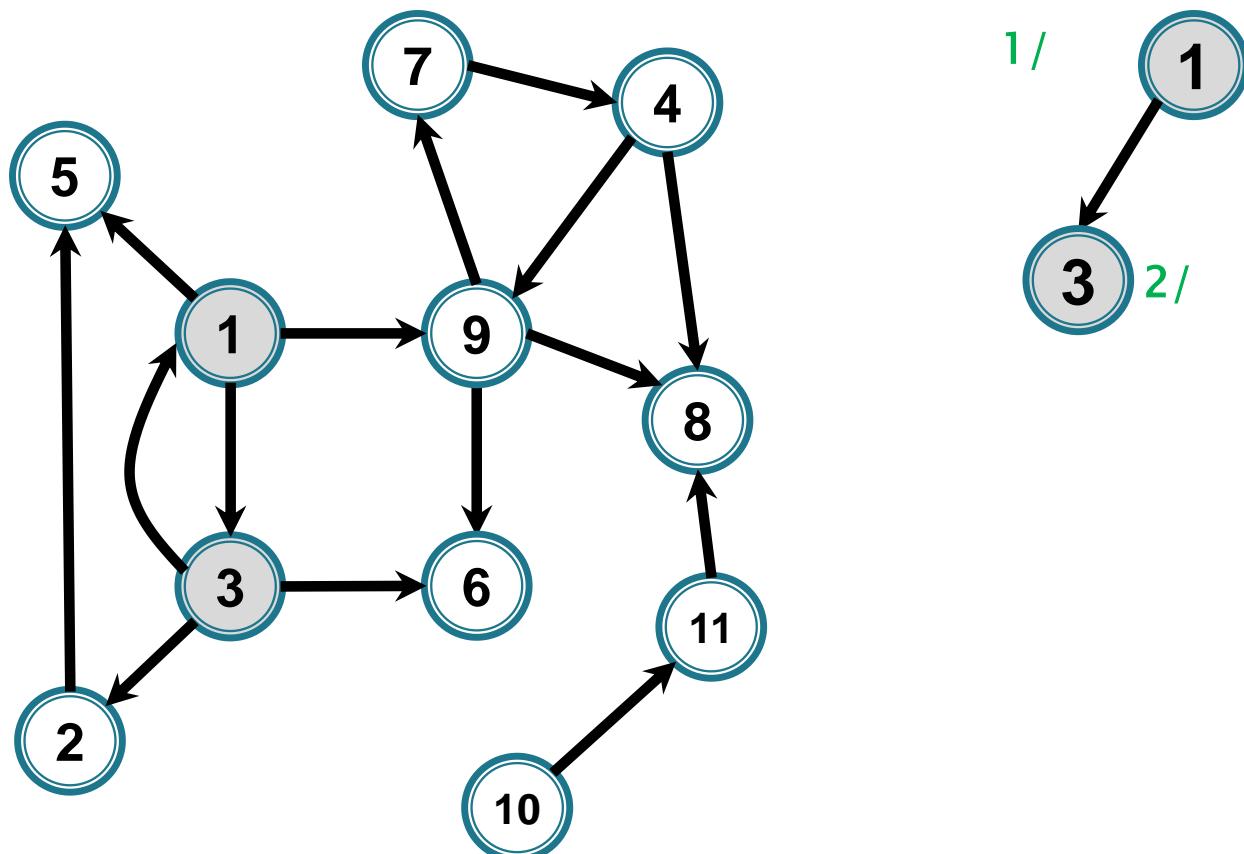
Exemplu DF graf orientat



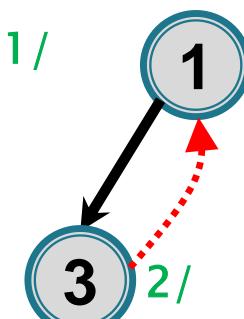
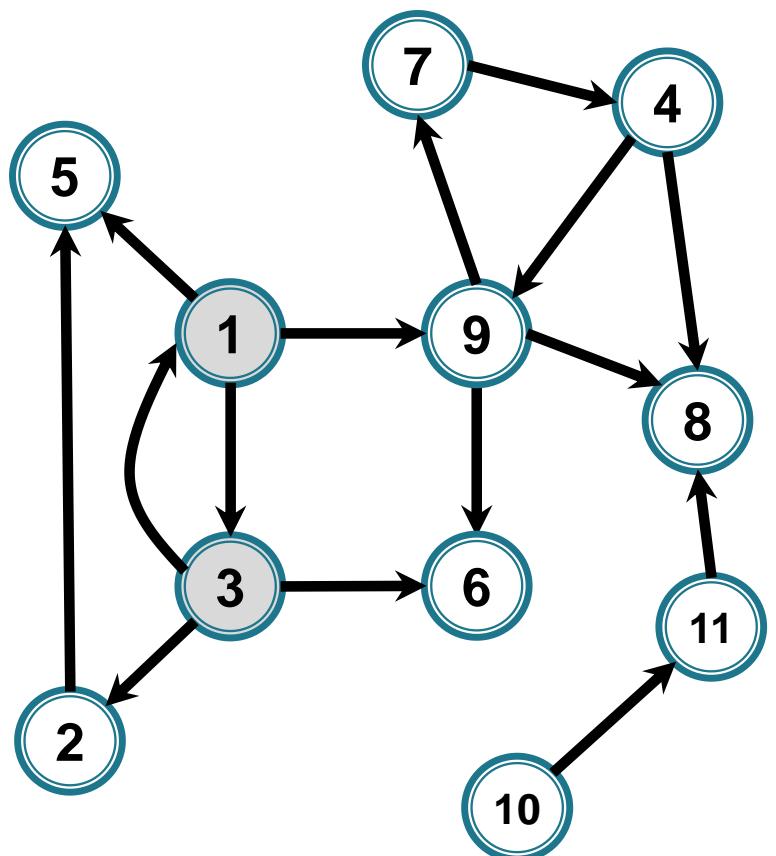
1 /



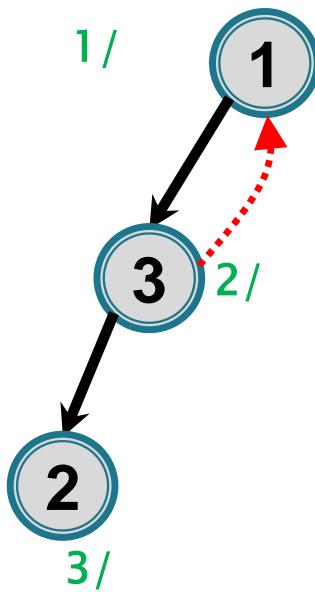
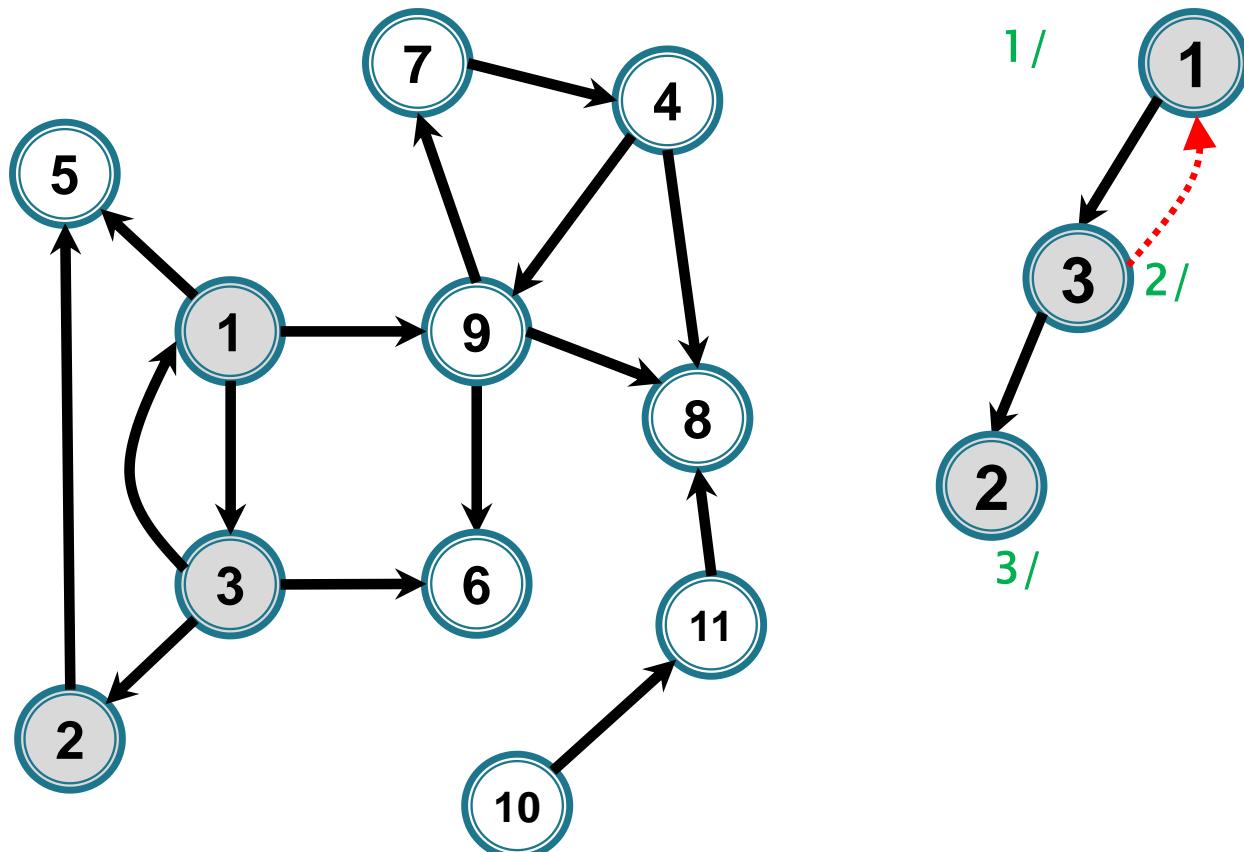
Exemplu DF graf orientat



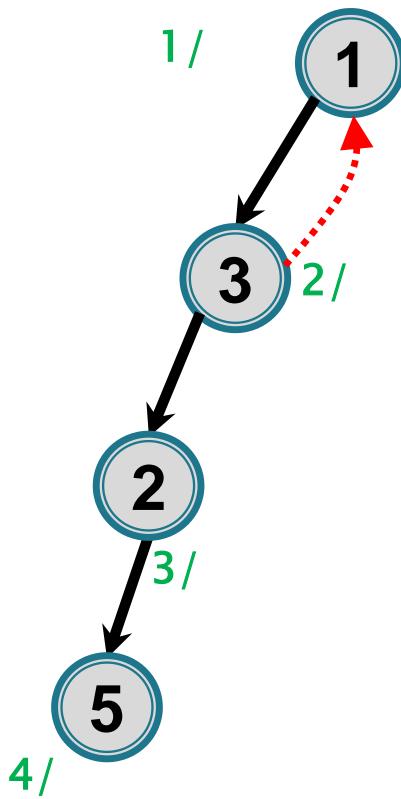
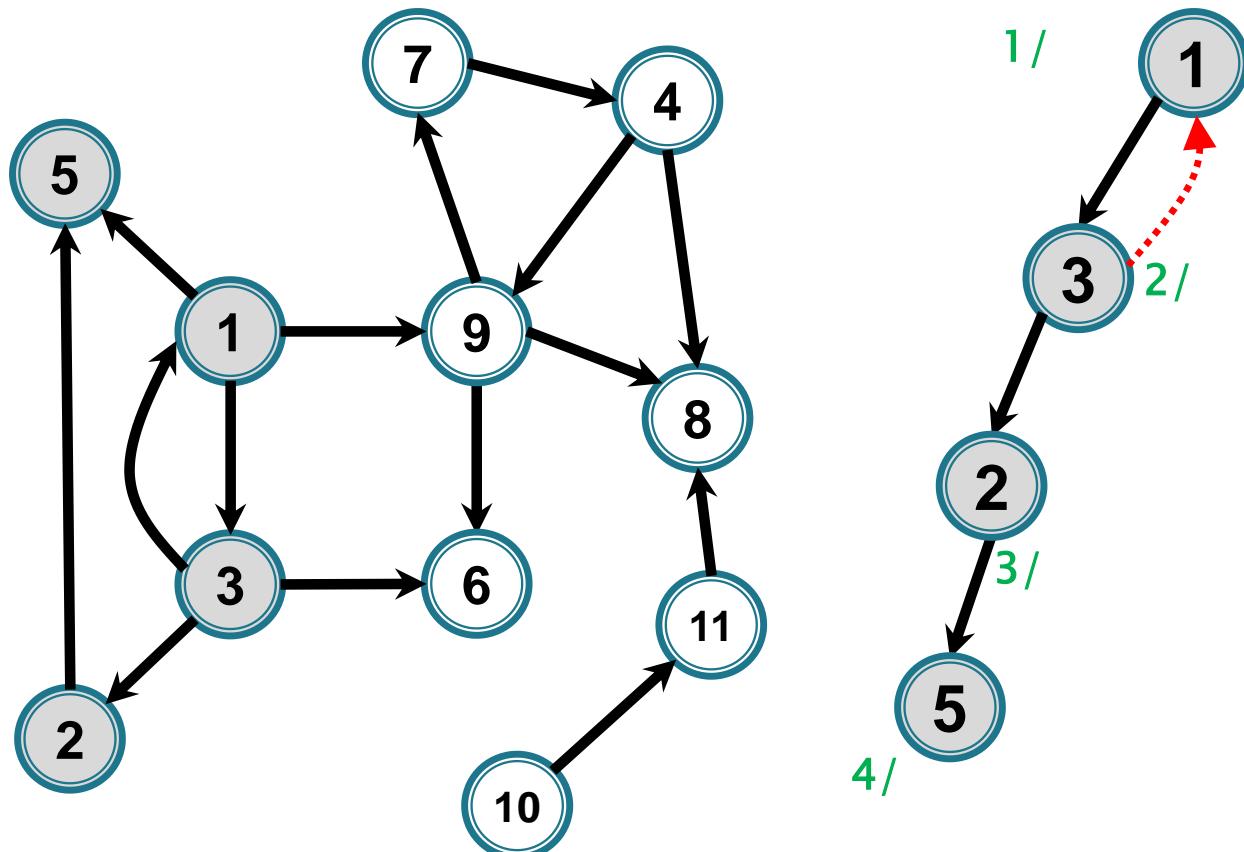
Exemplu DF graf orientat



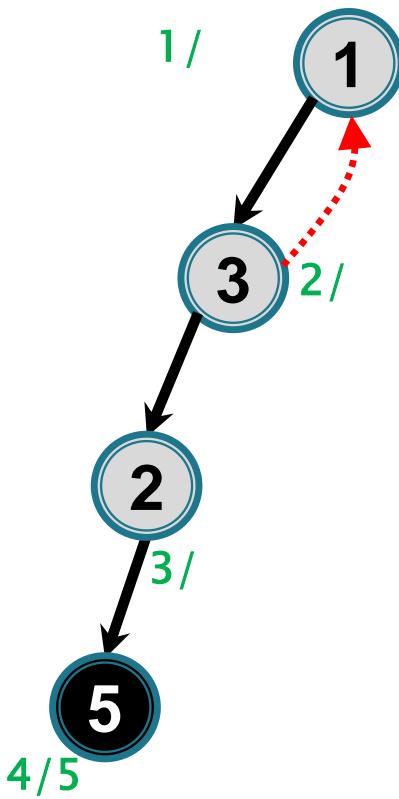
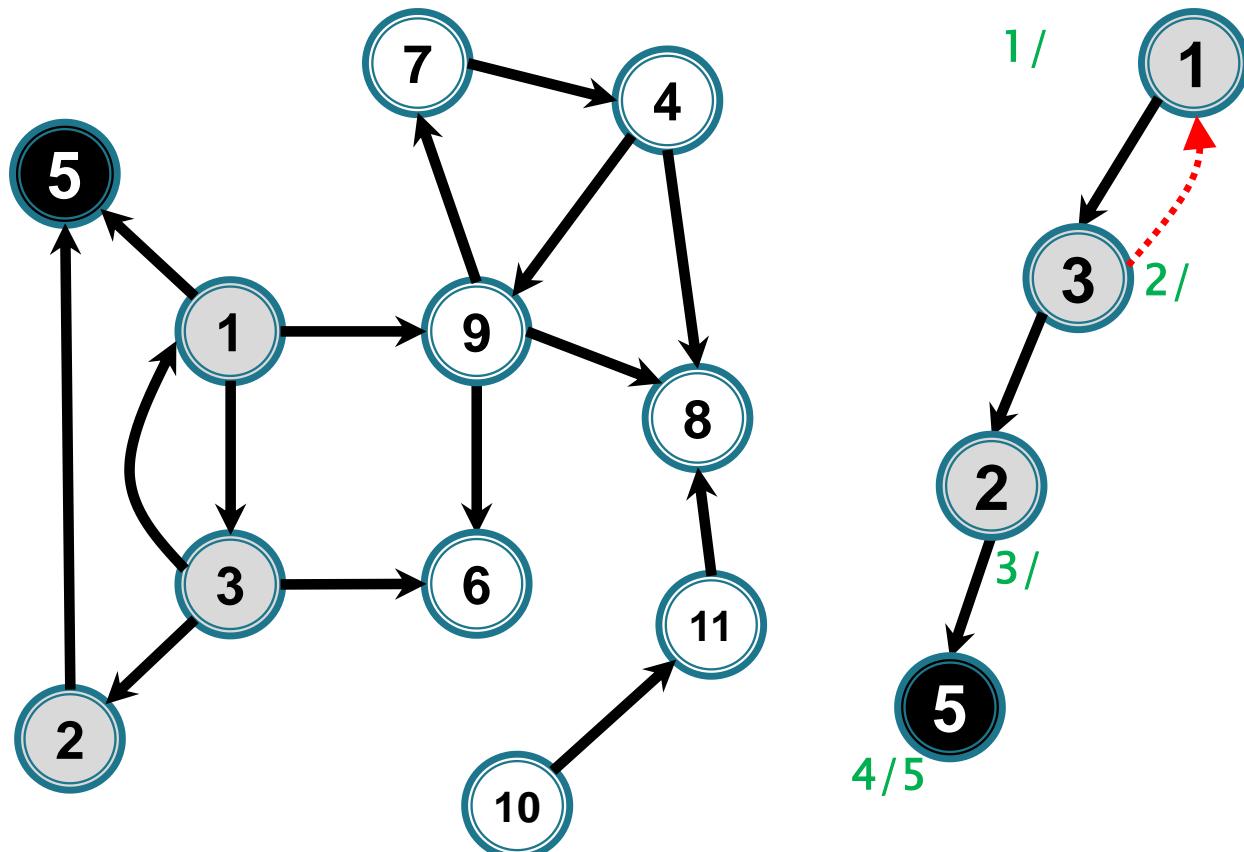
Exemplu DF graf orientat



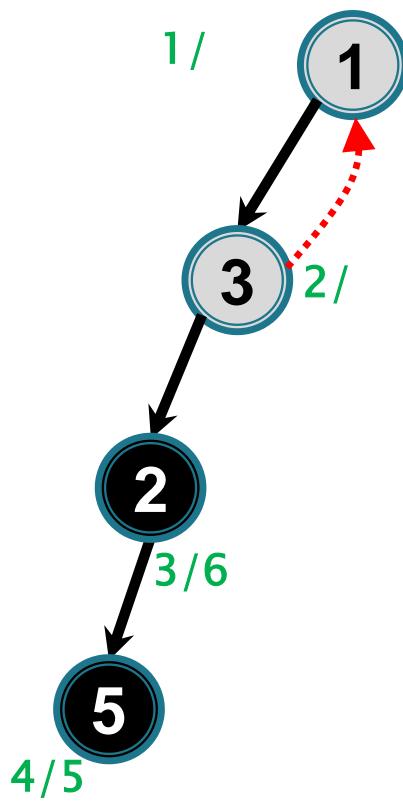
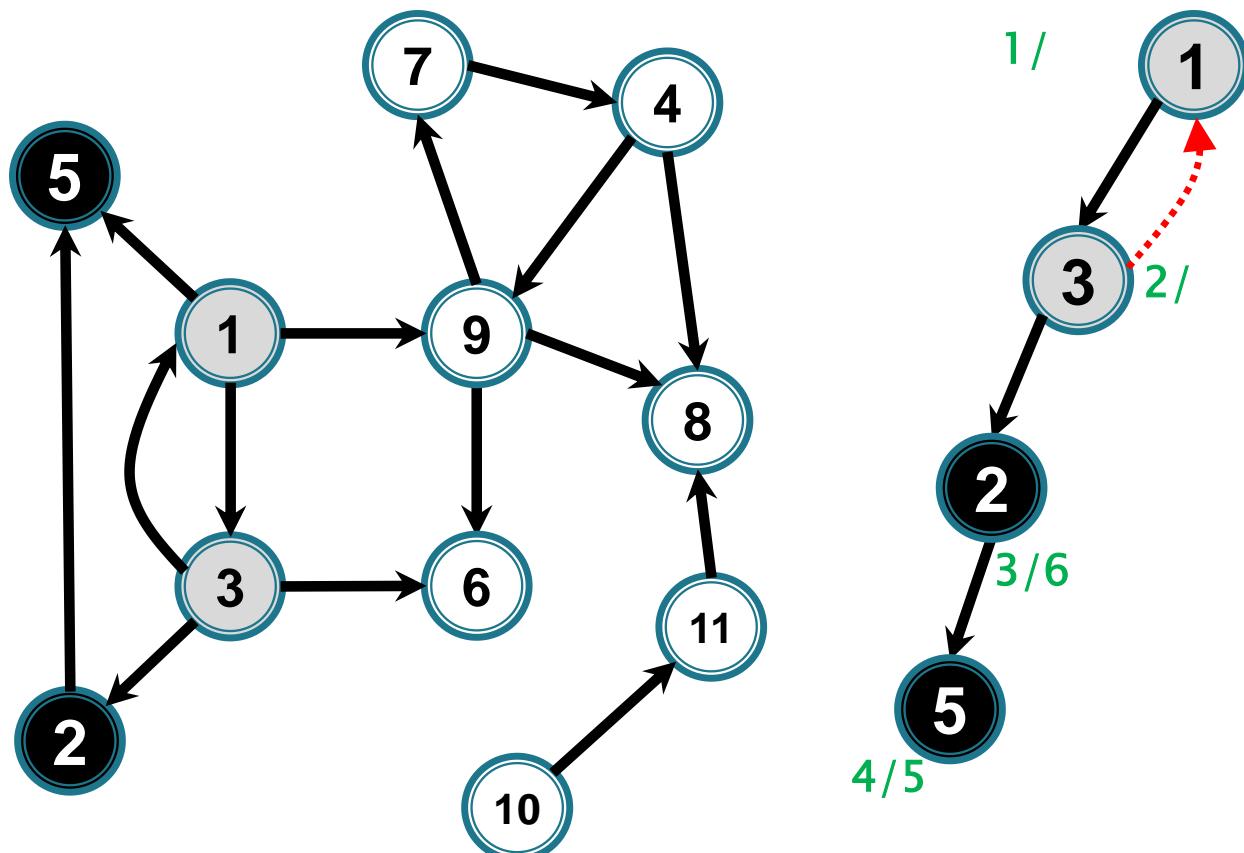
Exemplu DF graf orientat



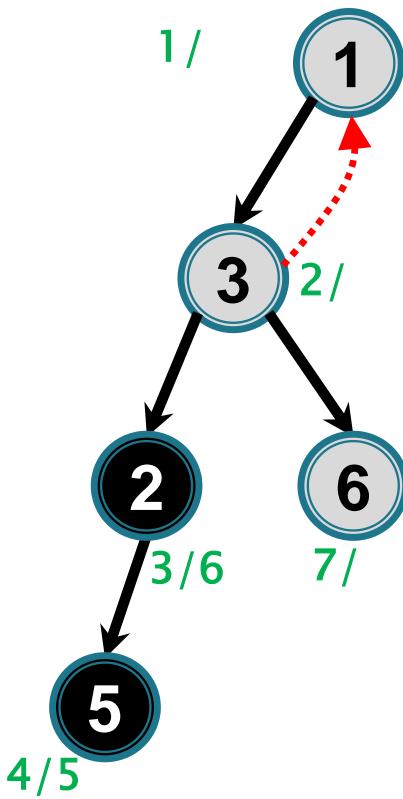
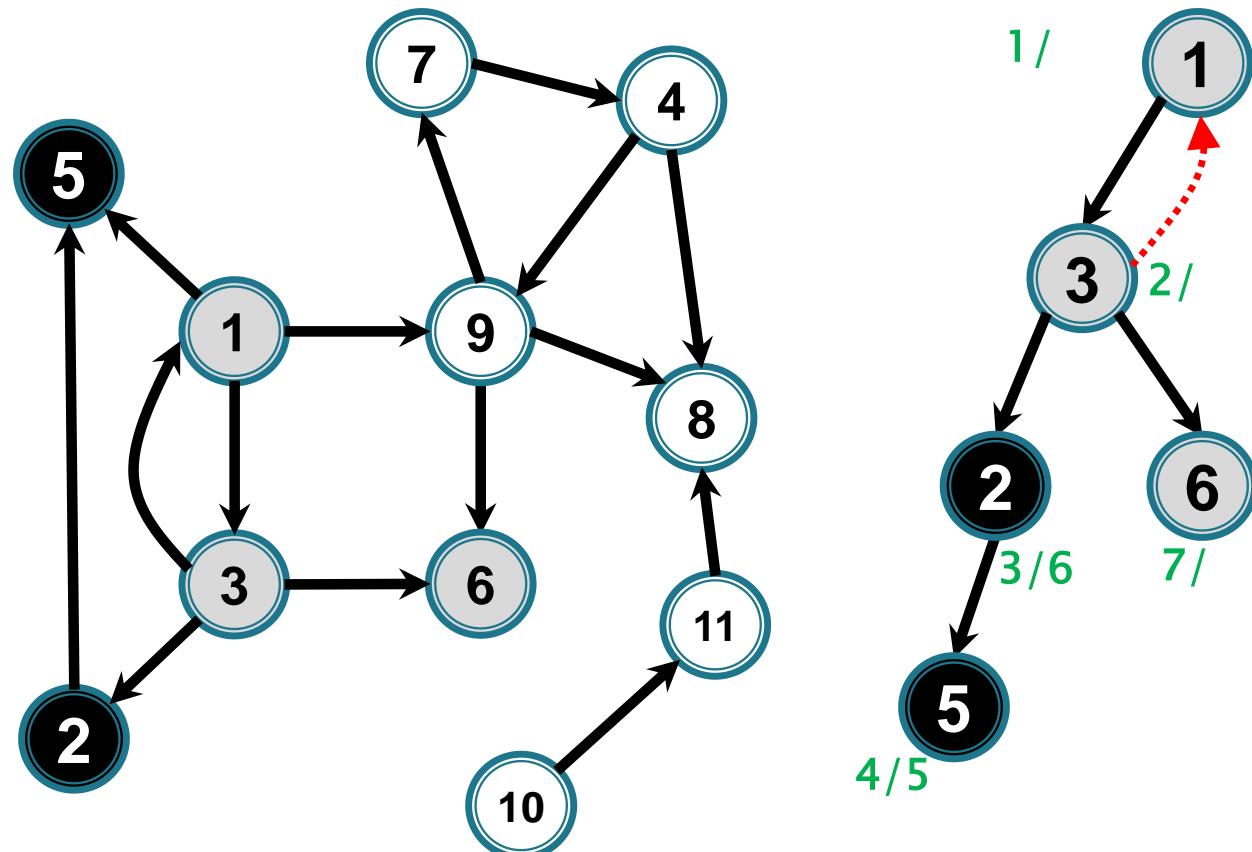
Exemplu DF graf orientat



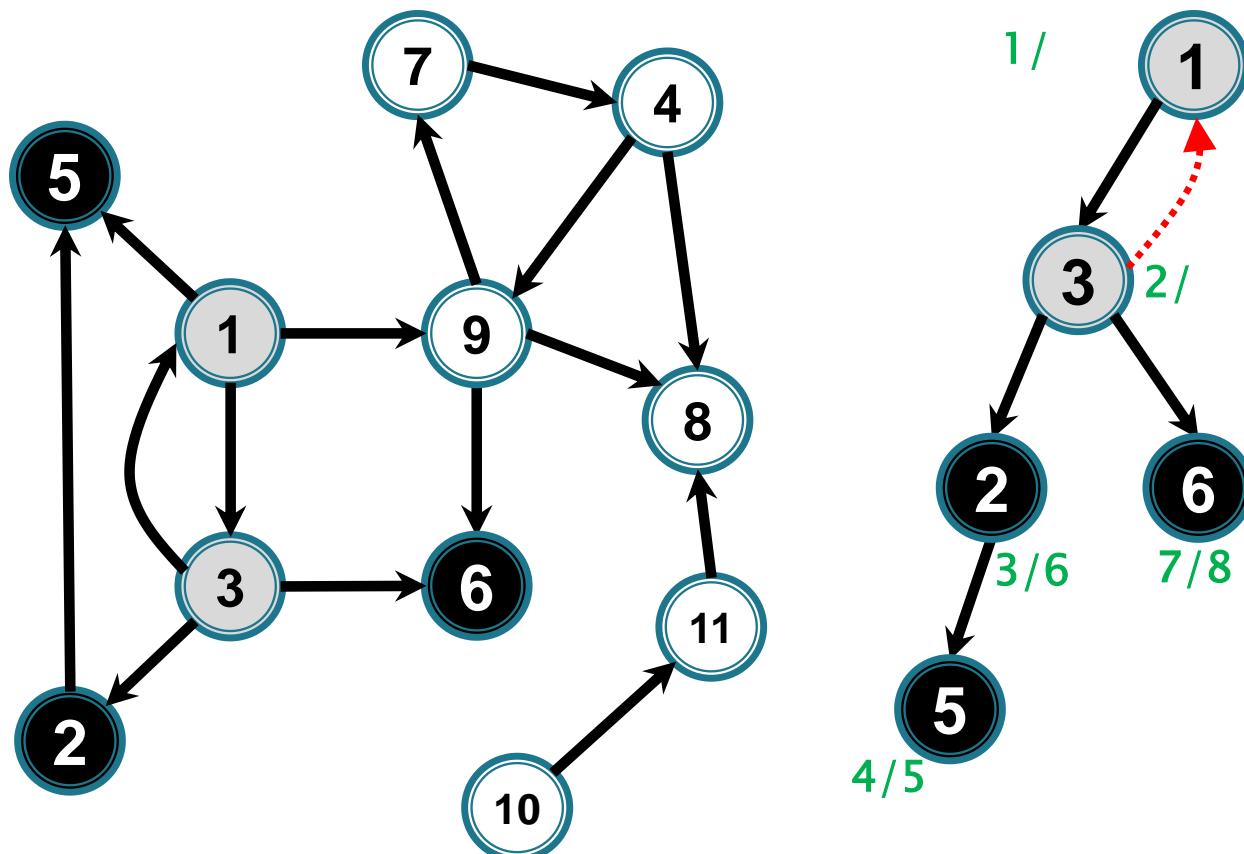
Exemplu DF graf orientat



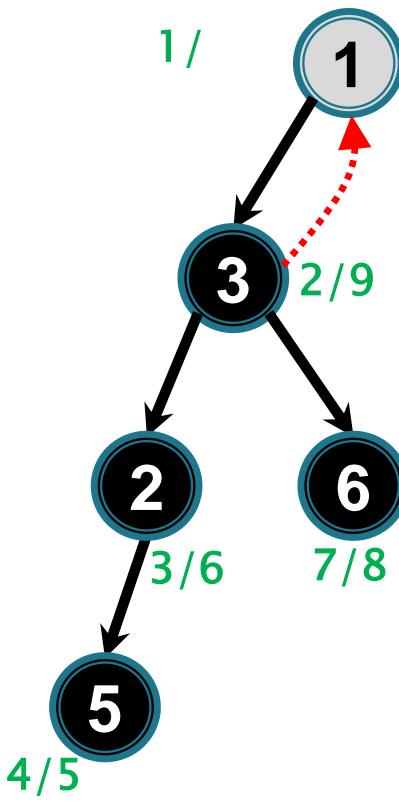
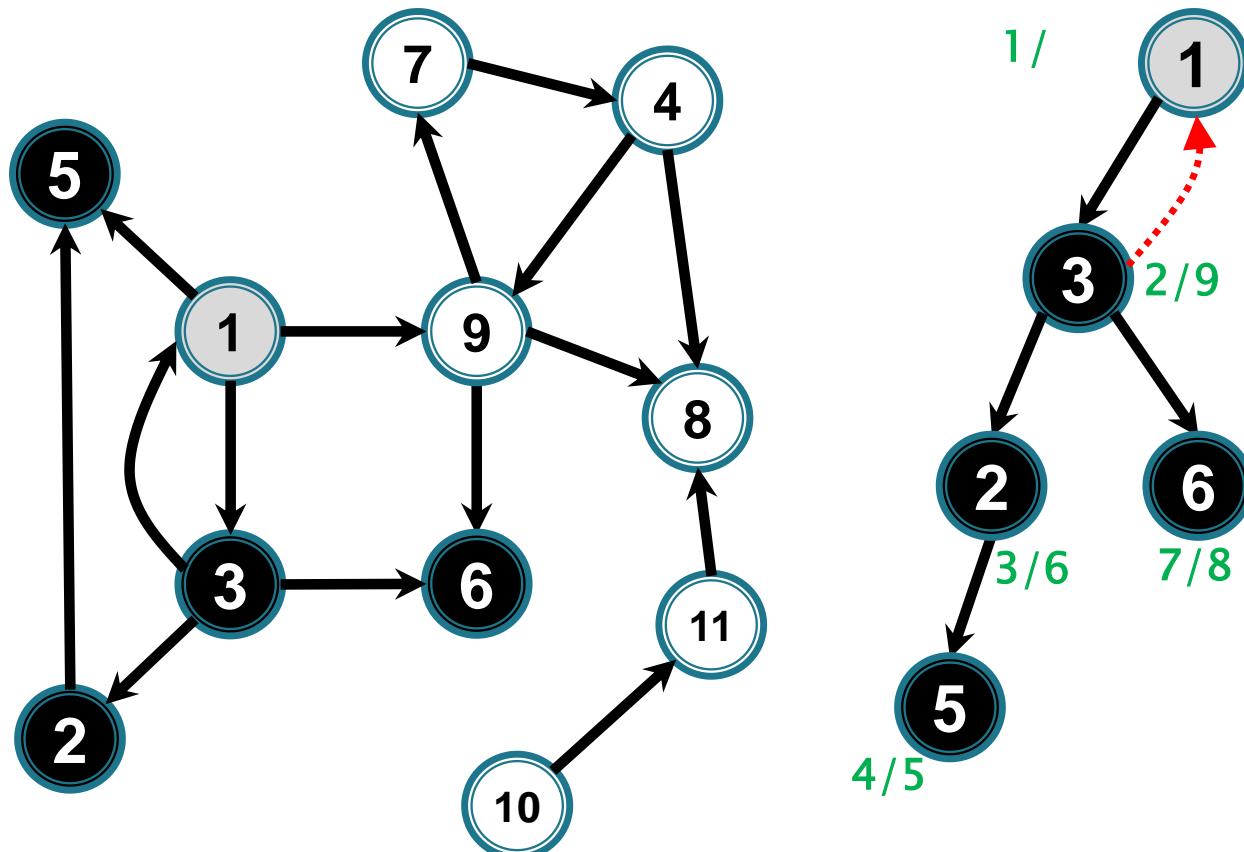
Exemplu DF graf orientat



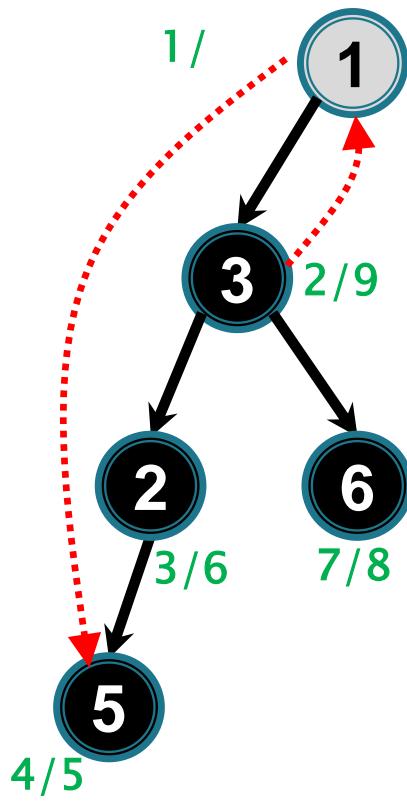
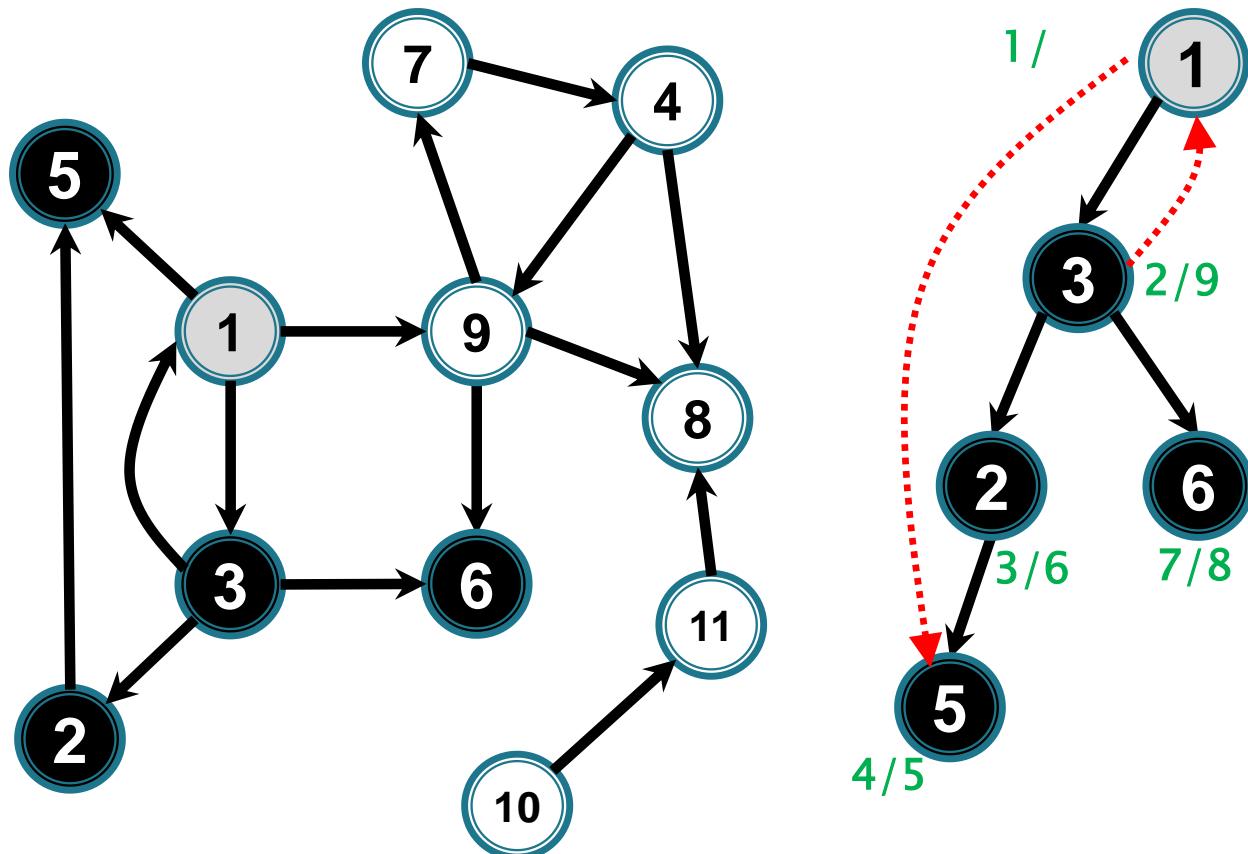
Exemplu DF graf orientat



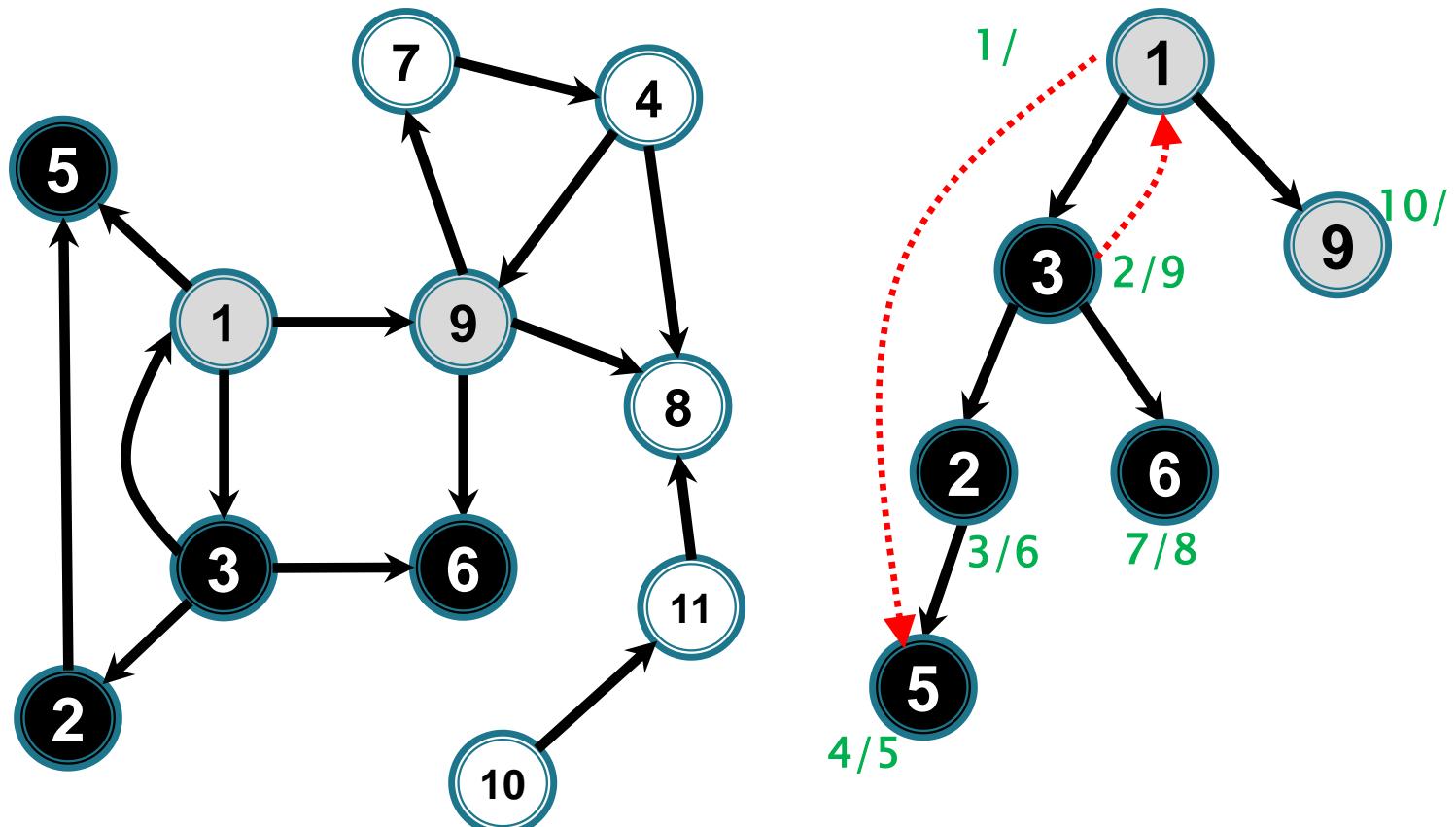
Exemplu DF graf orientat



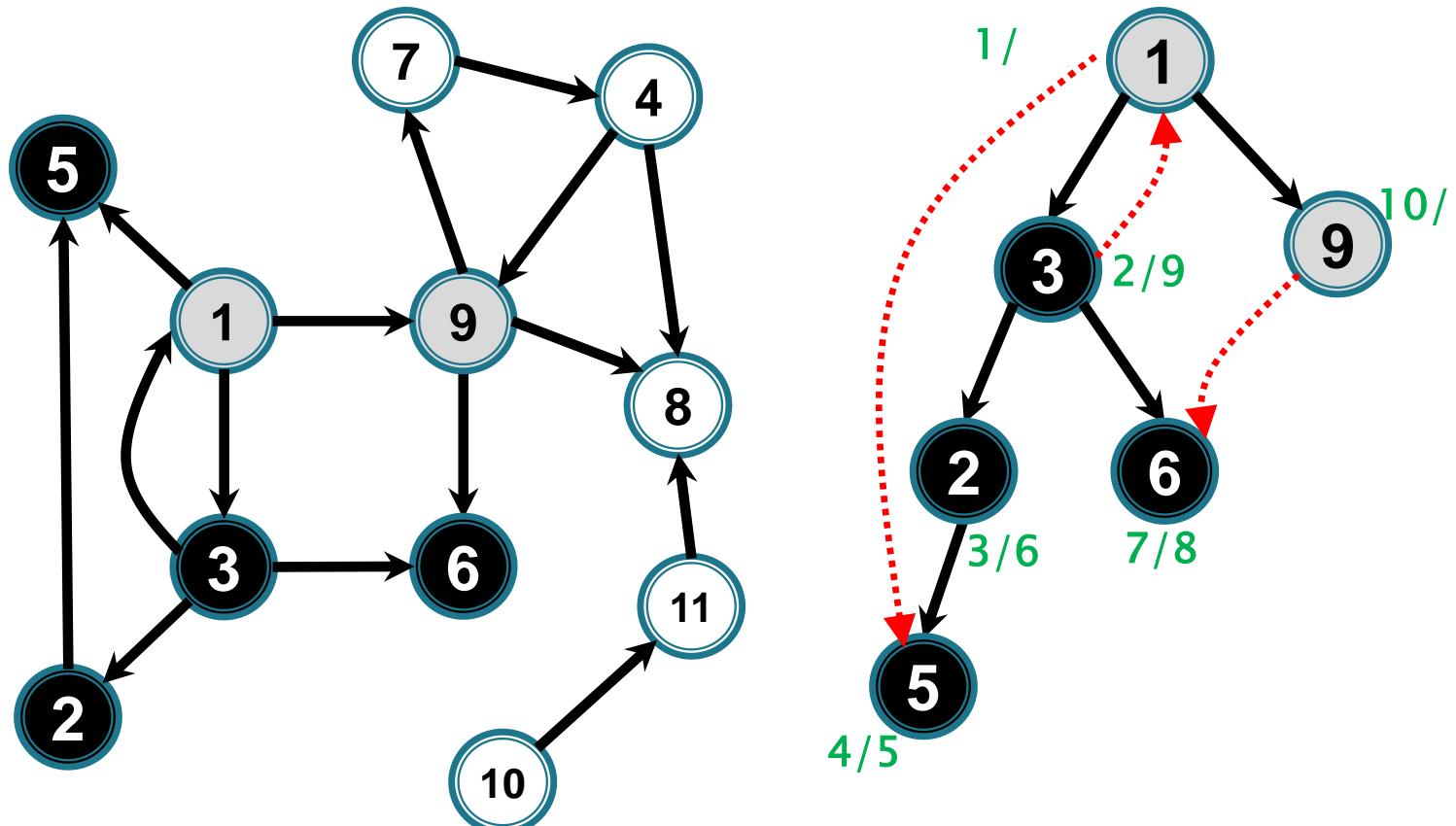
Exemplu DF graf orientat



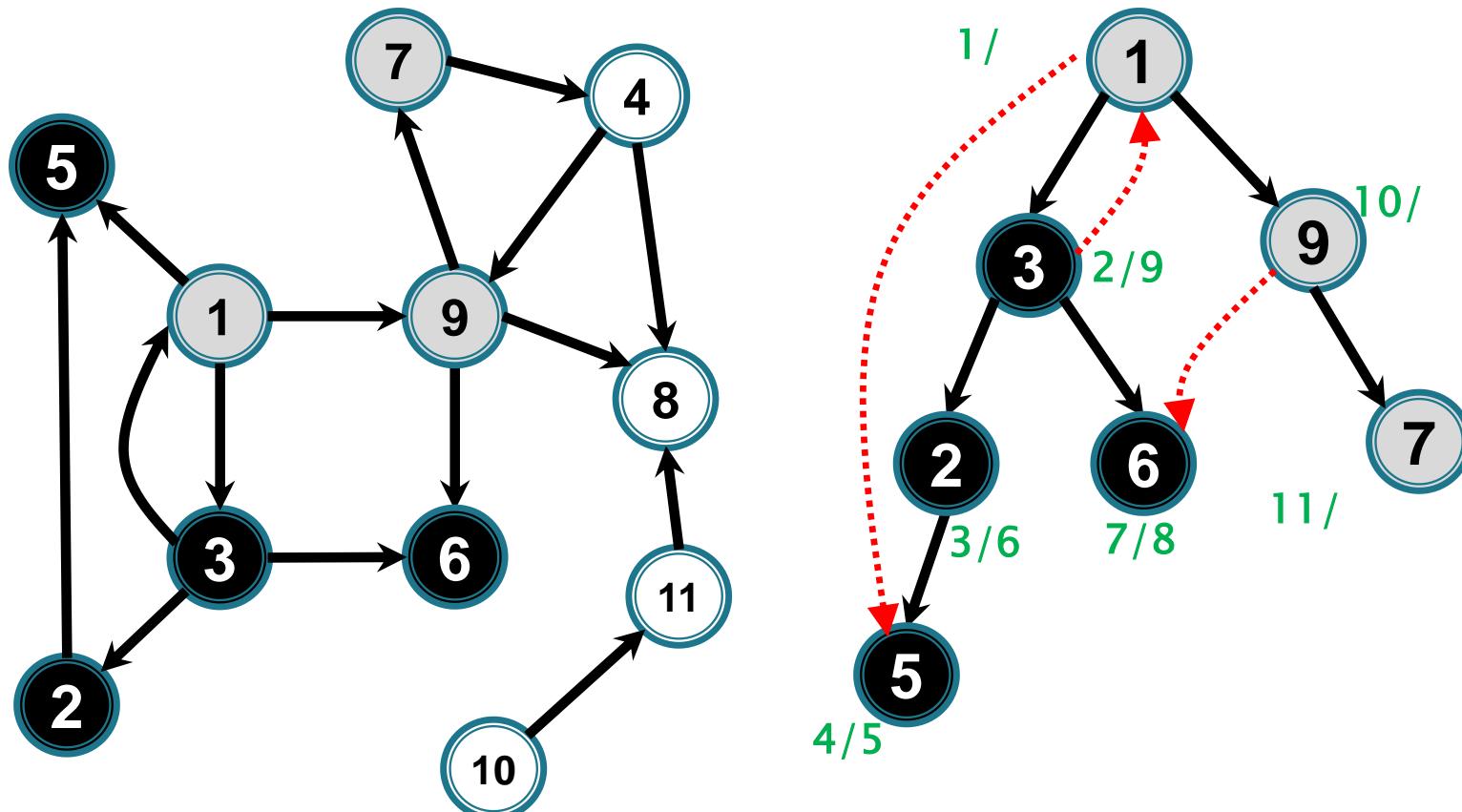
Exemplu DF graf orientat



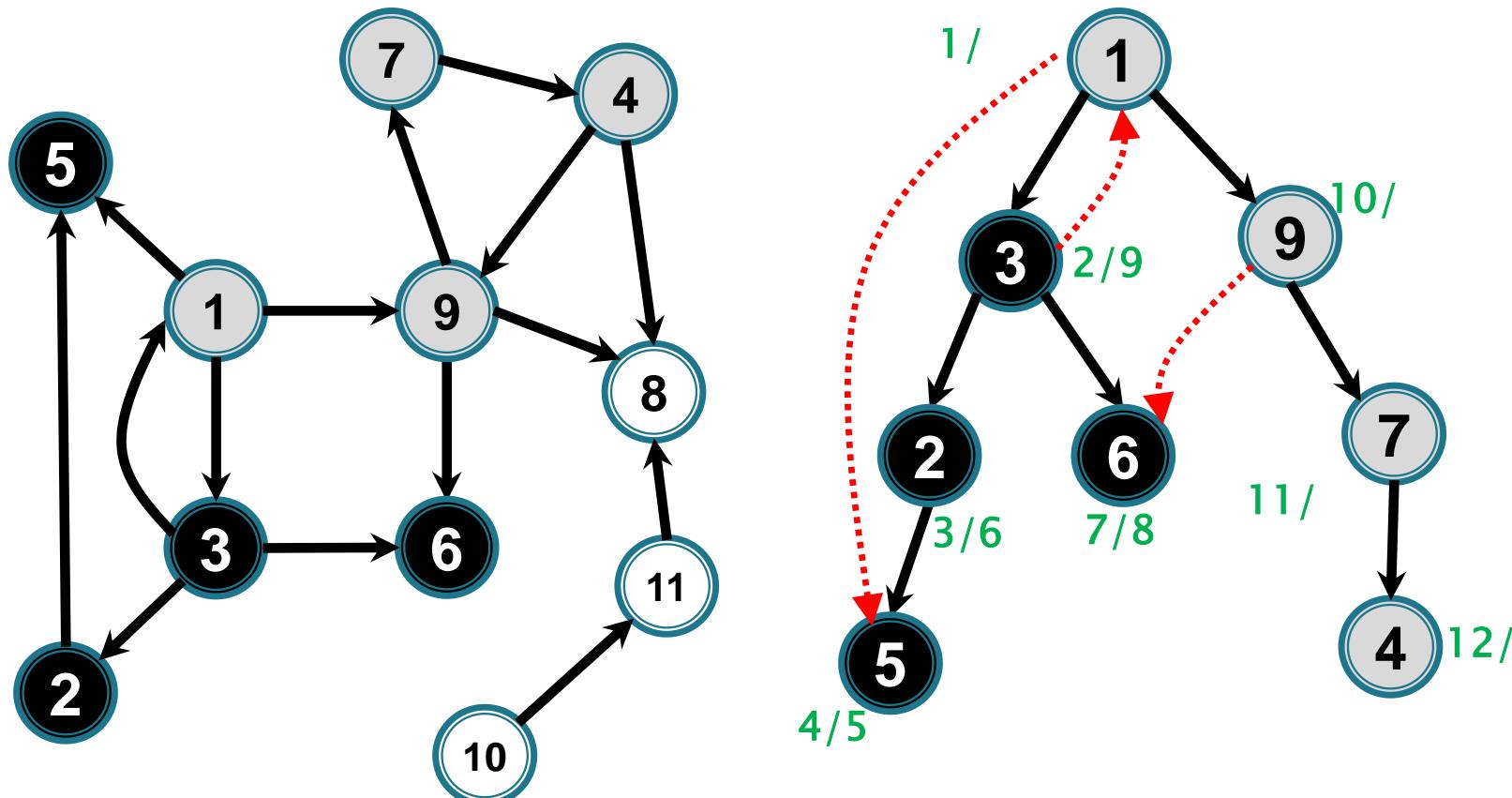
Exemplu DF graf orientat



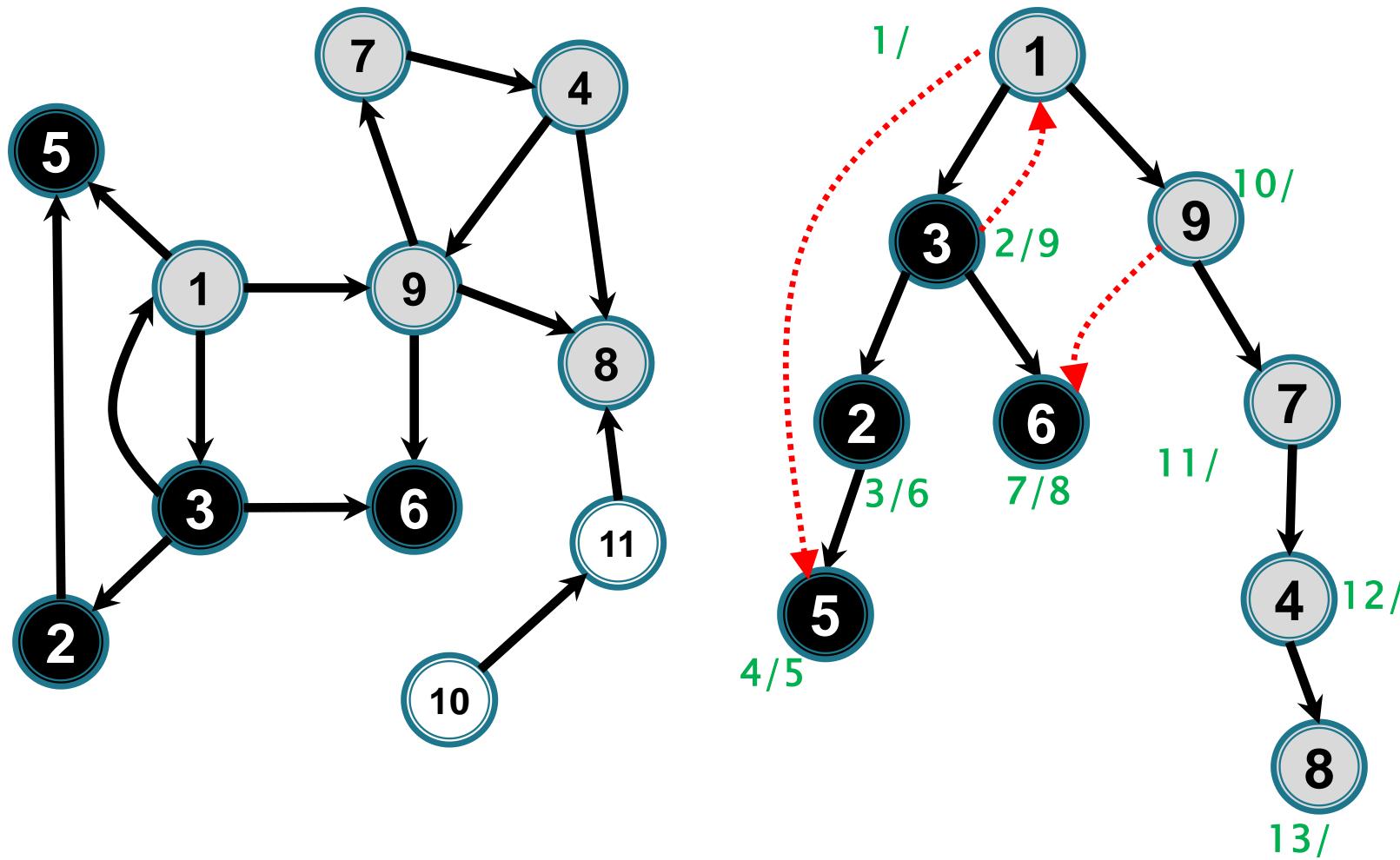
Exemplu DF graf orientat



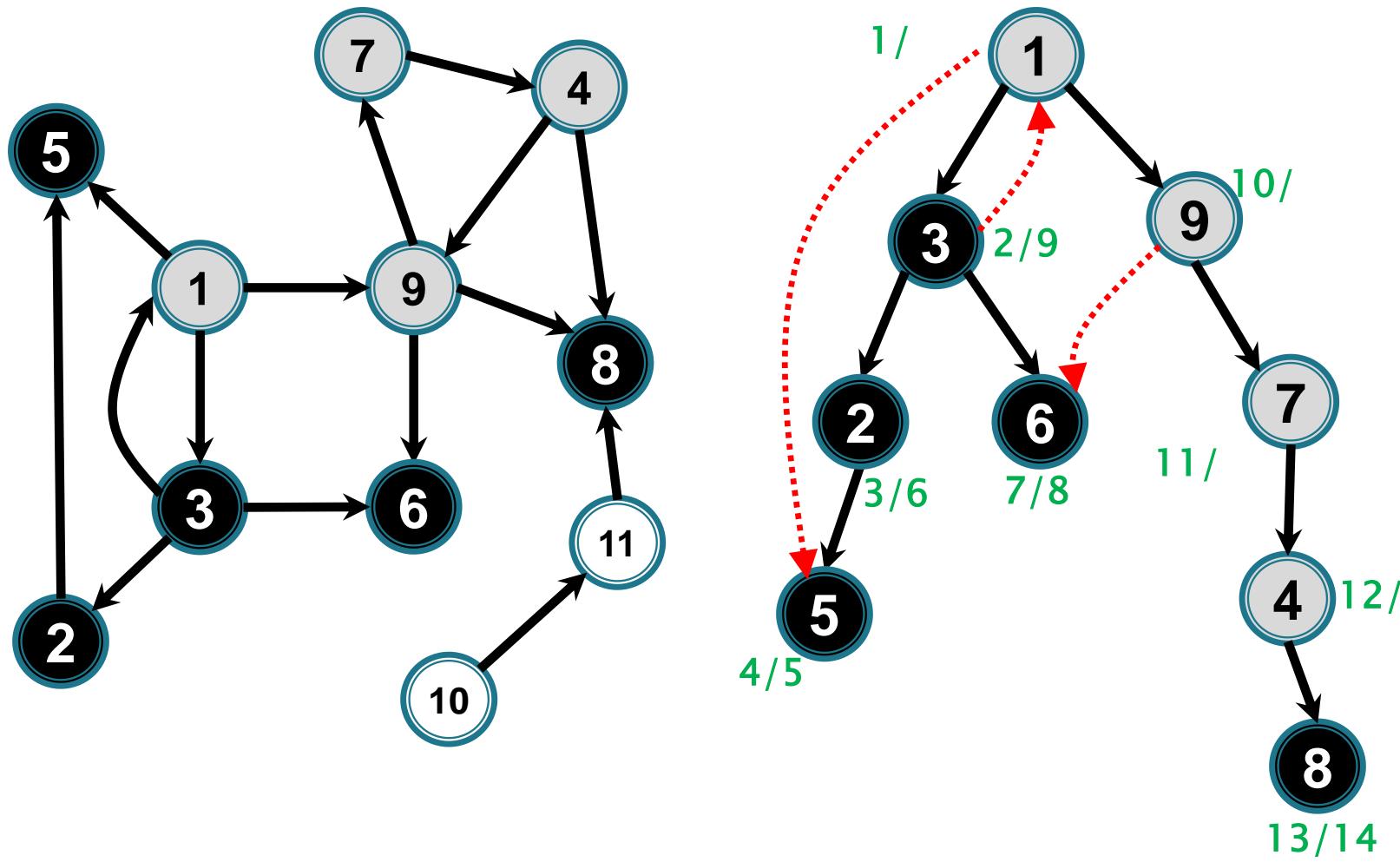
Exemplu DF graf orientat



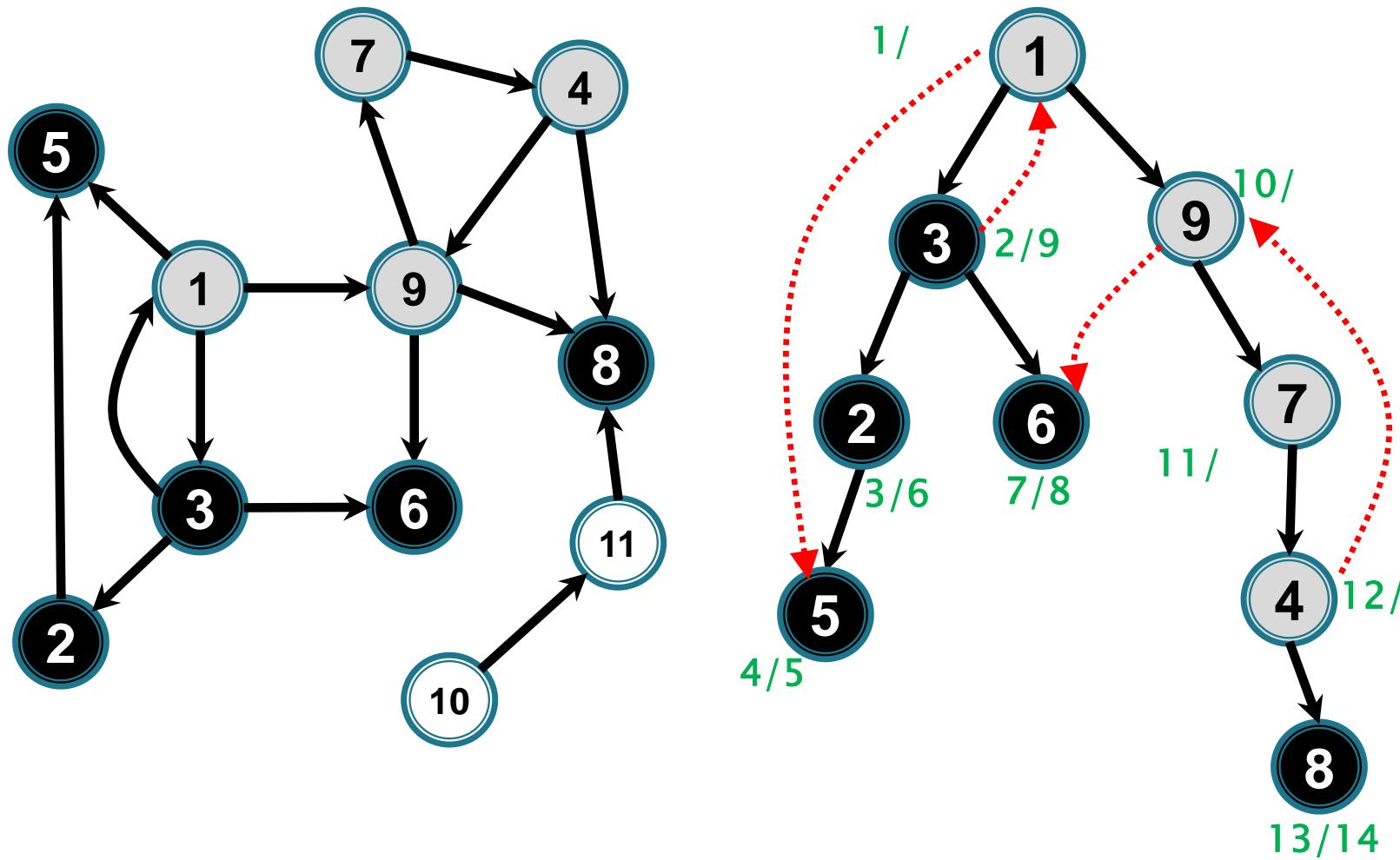
Exemplu DF graf orientat



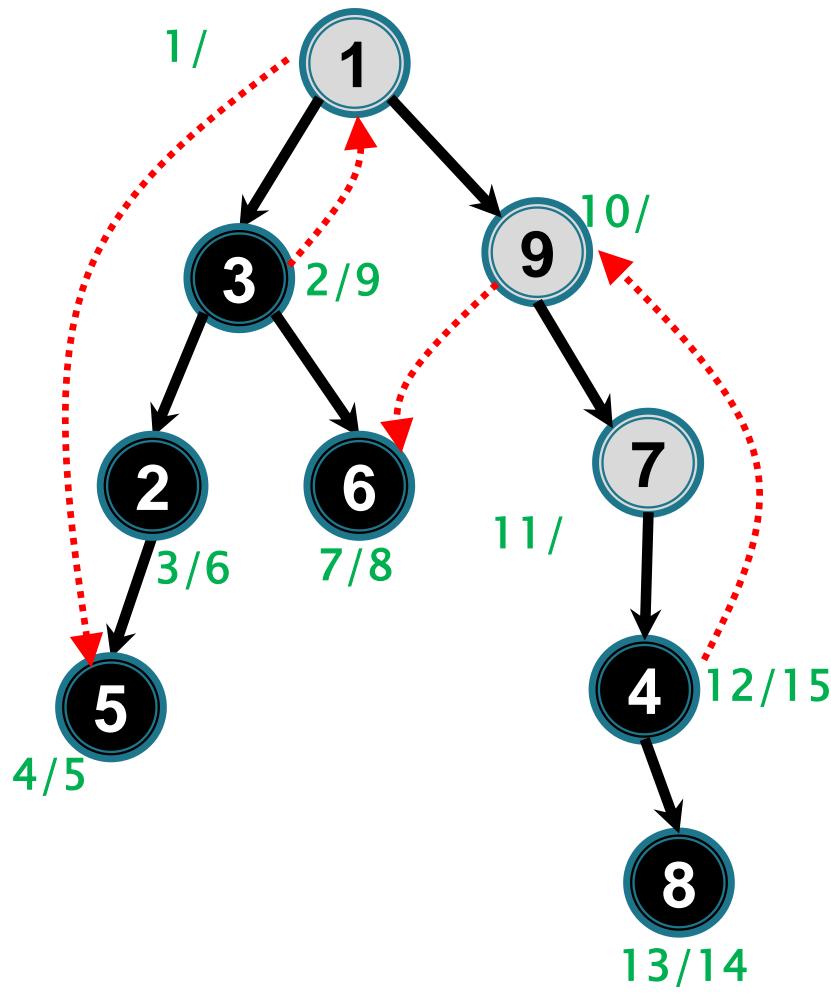
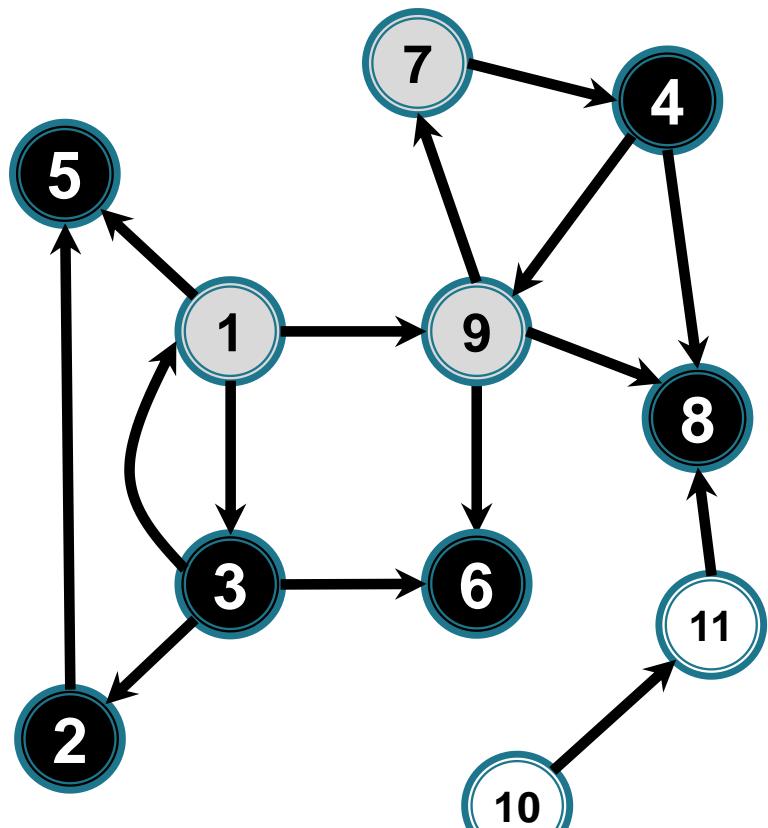
Exemplu DF graf orientat



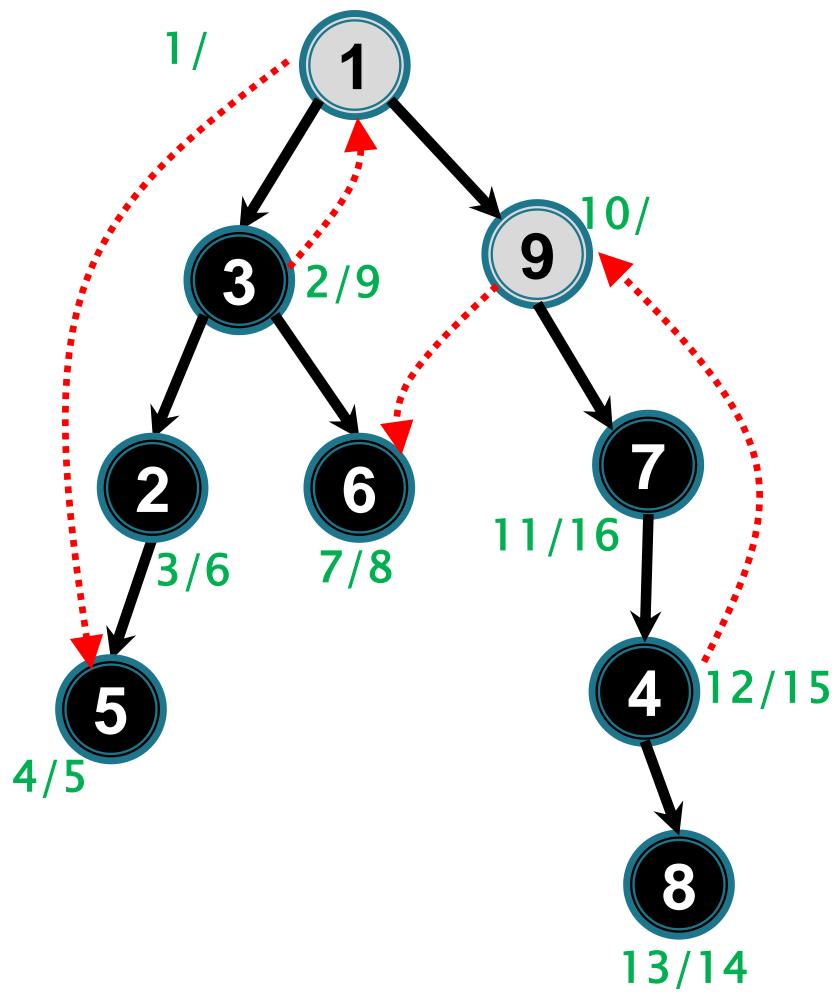
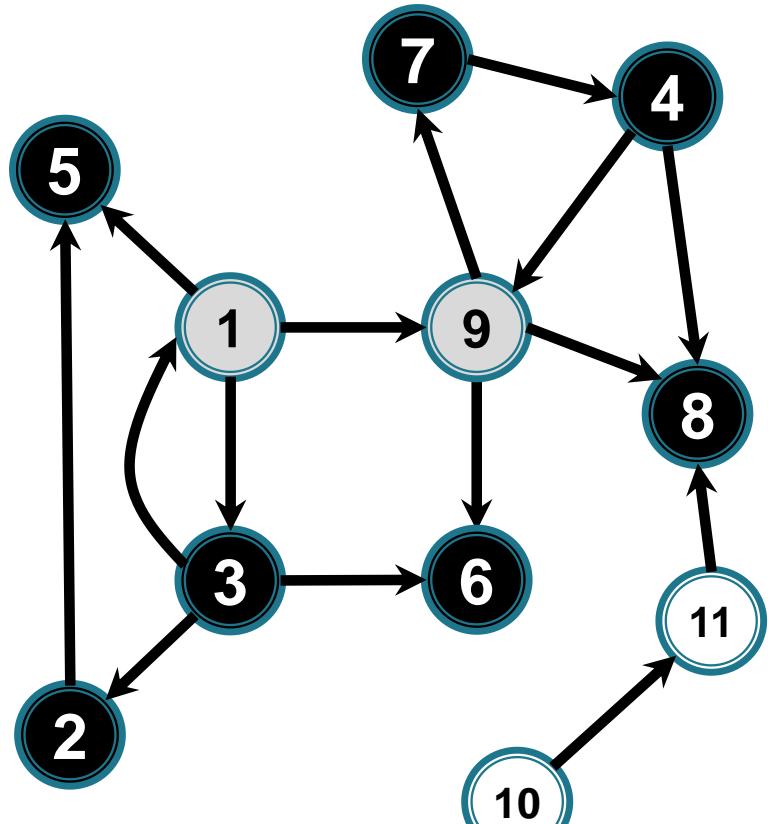
Exemplu DF graf orientat



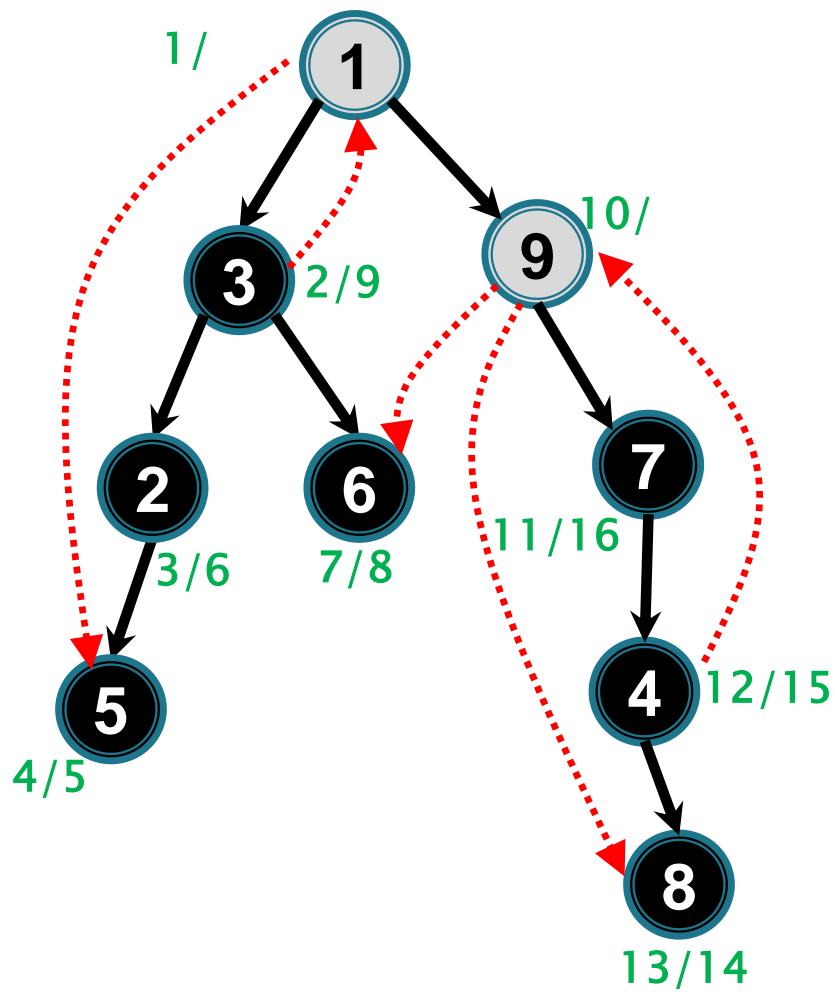
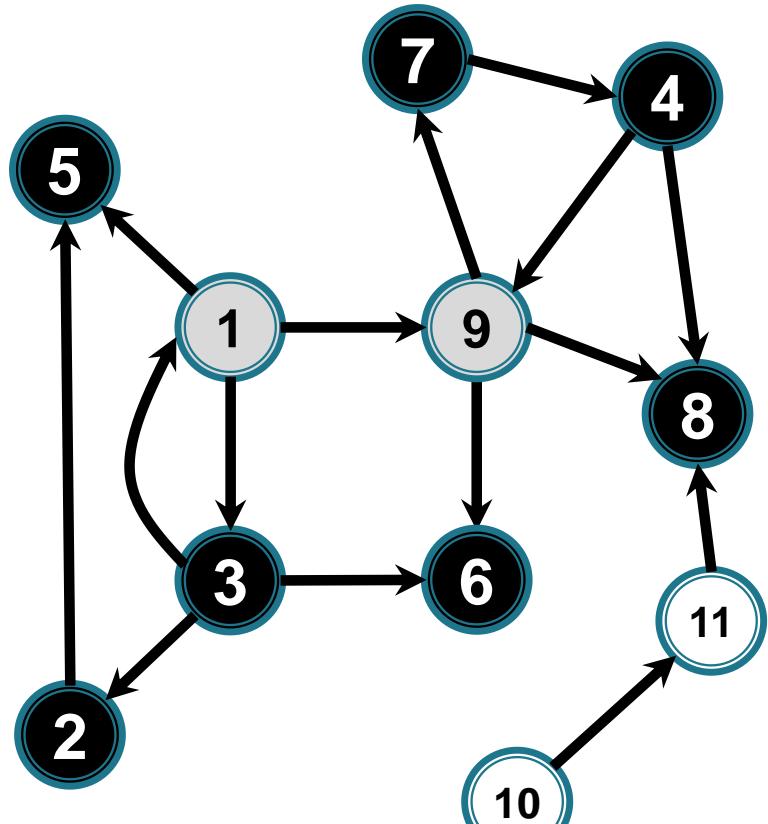
Exemplu DF graf orientat



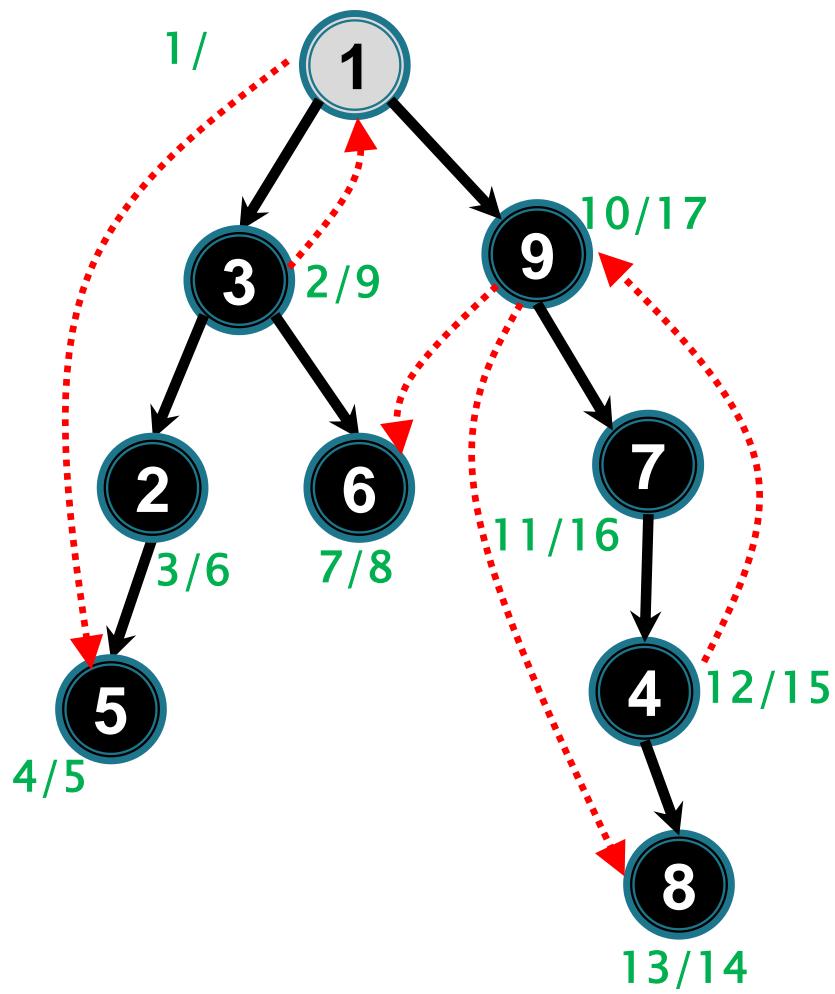
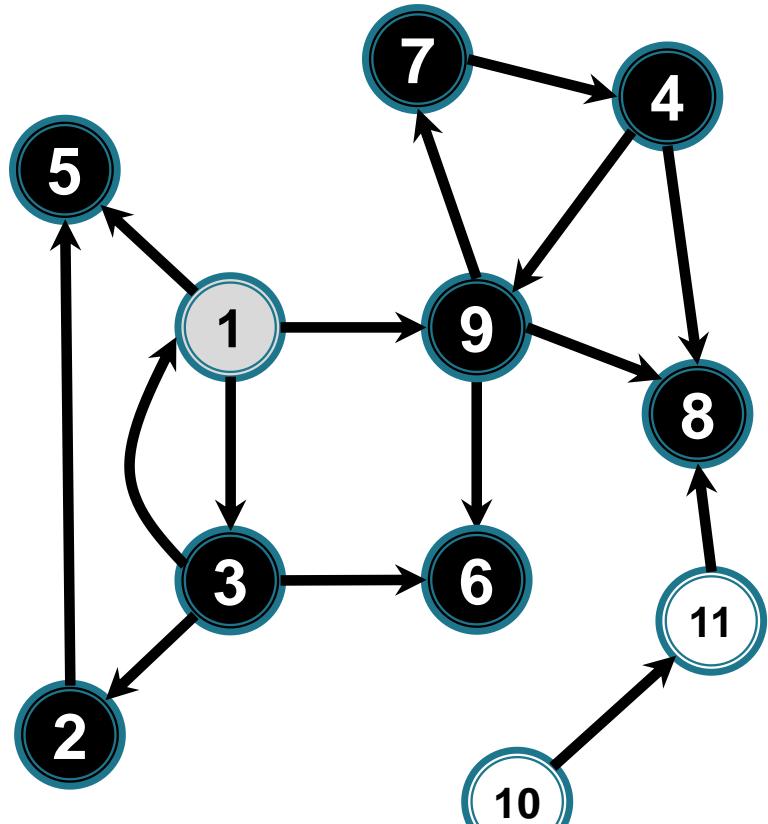
Exemplu DF graf orientat



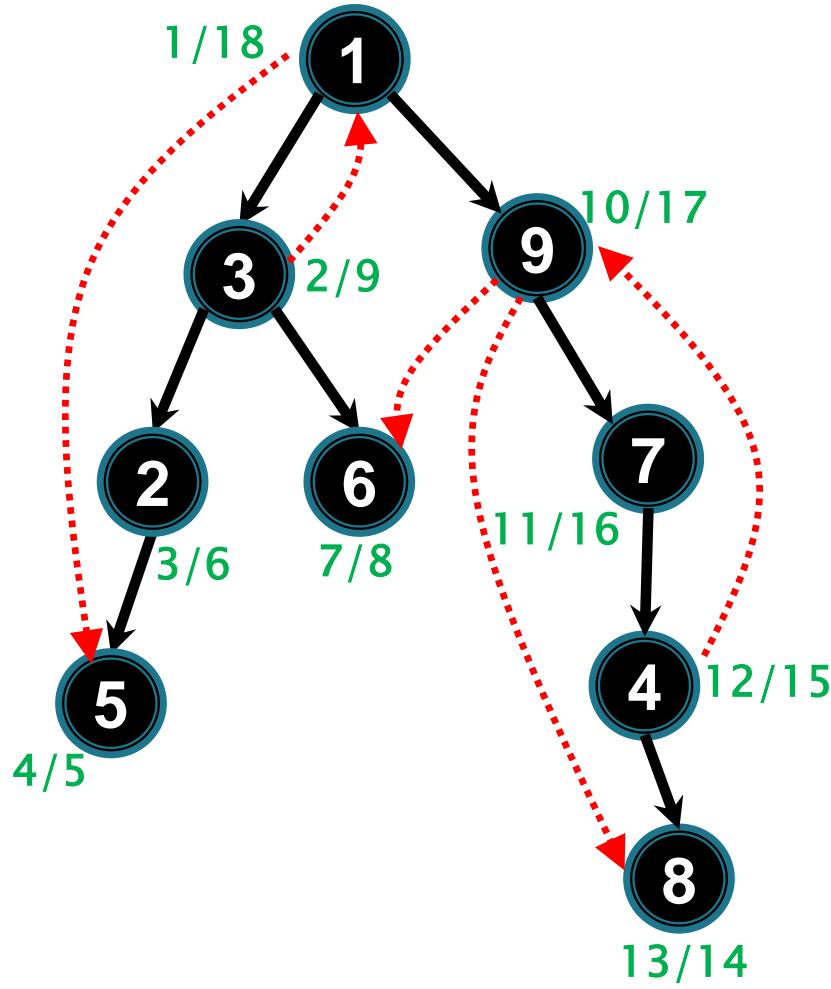
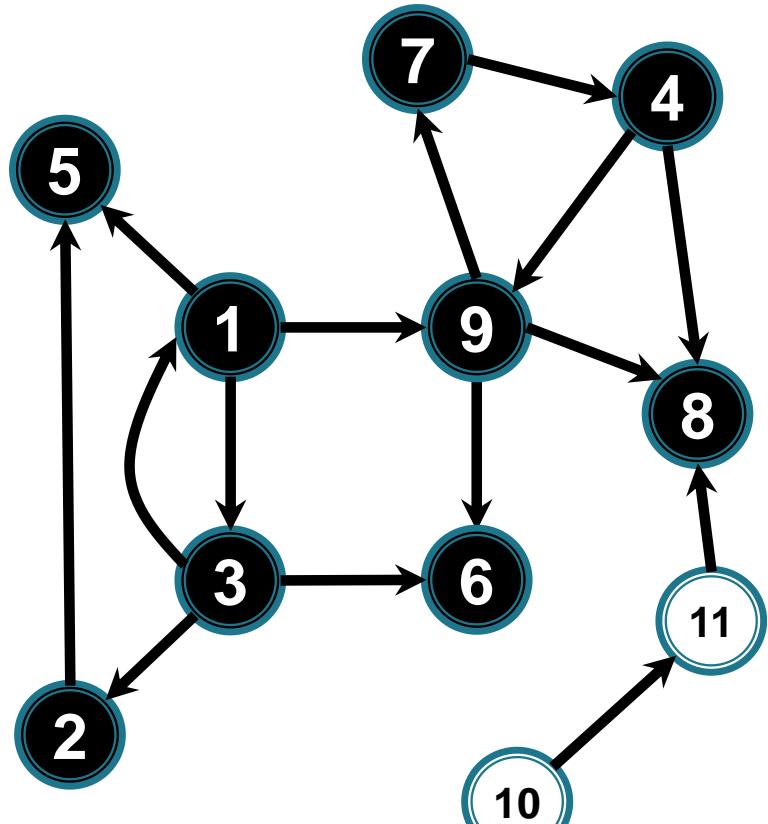
Exemplu DF graf orientat



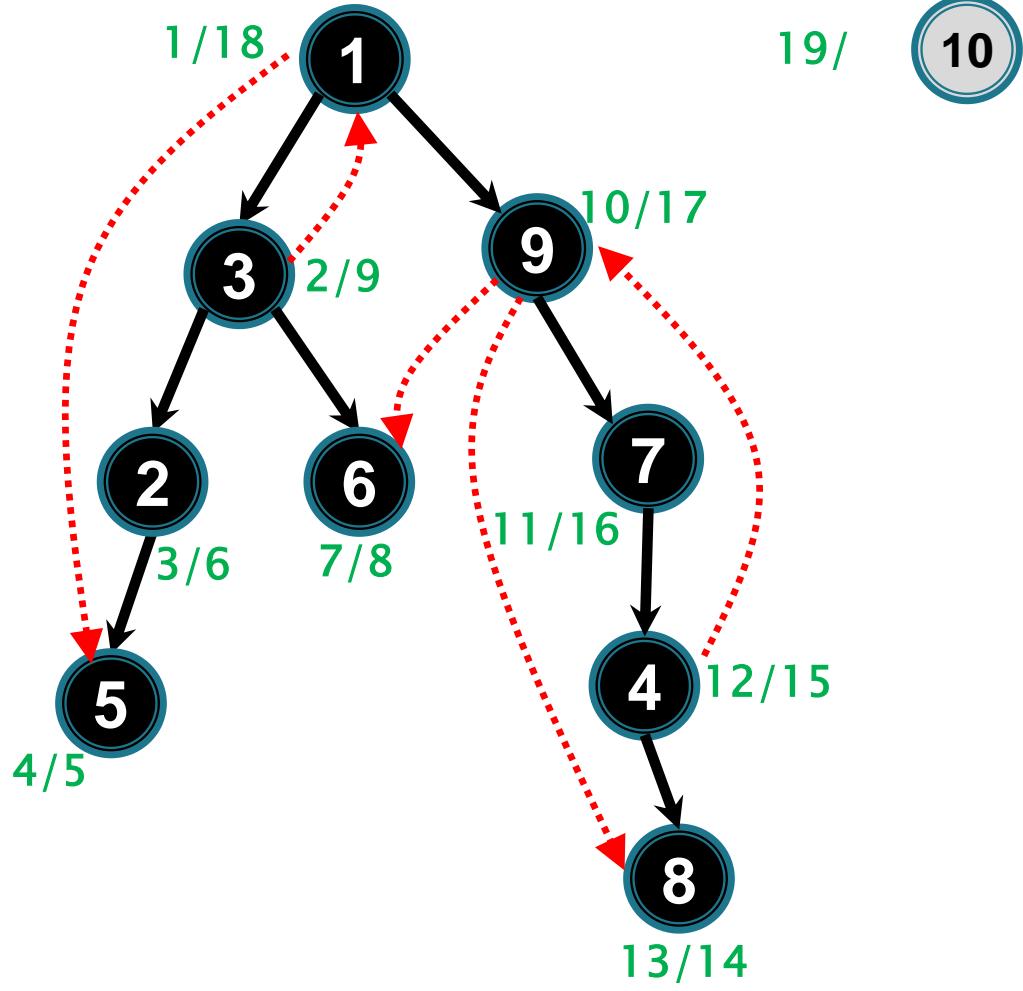
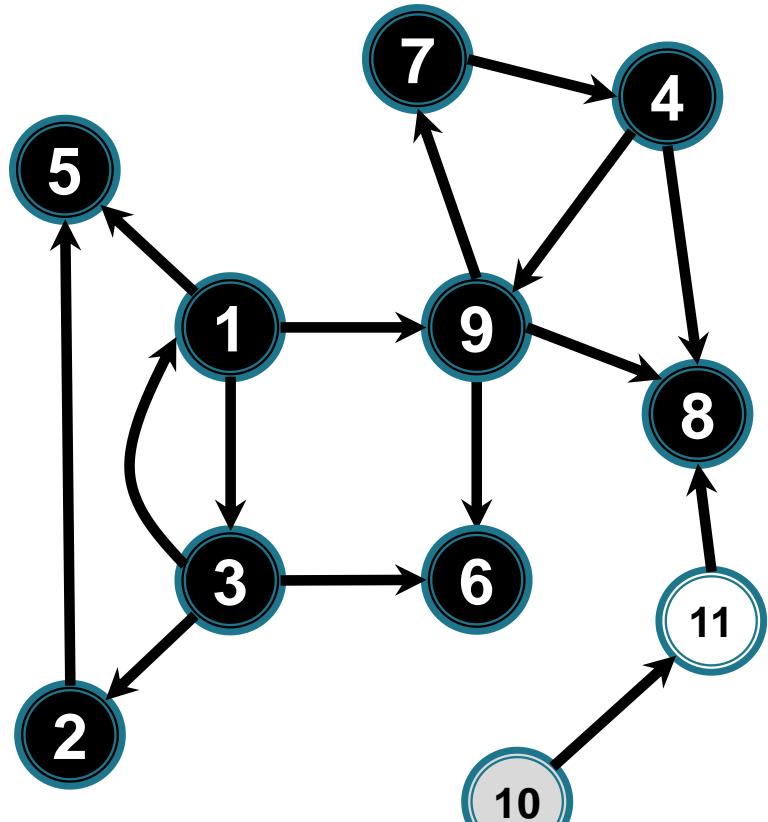
Exemplu DF graf orientat



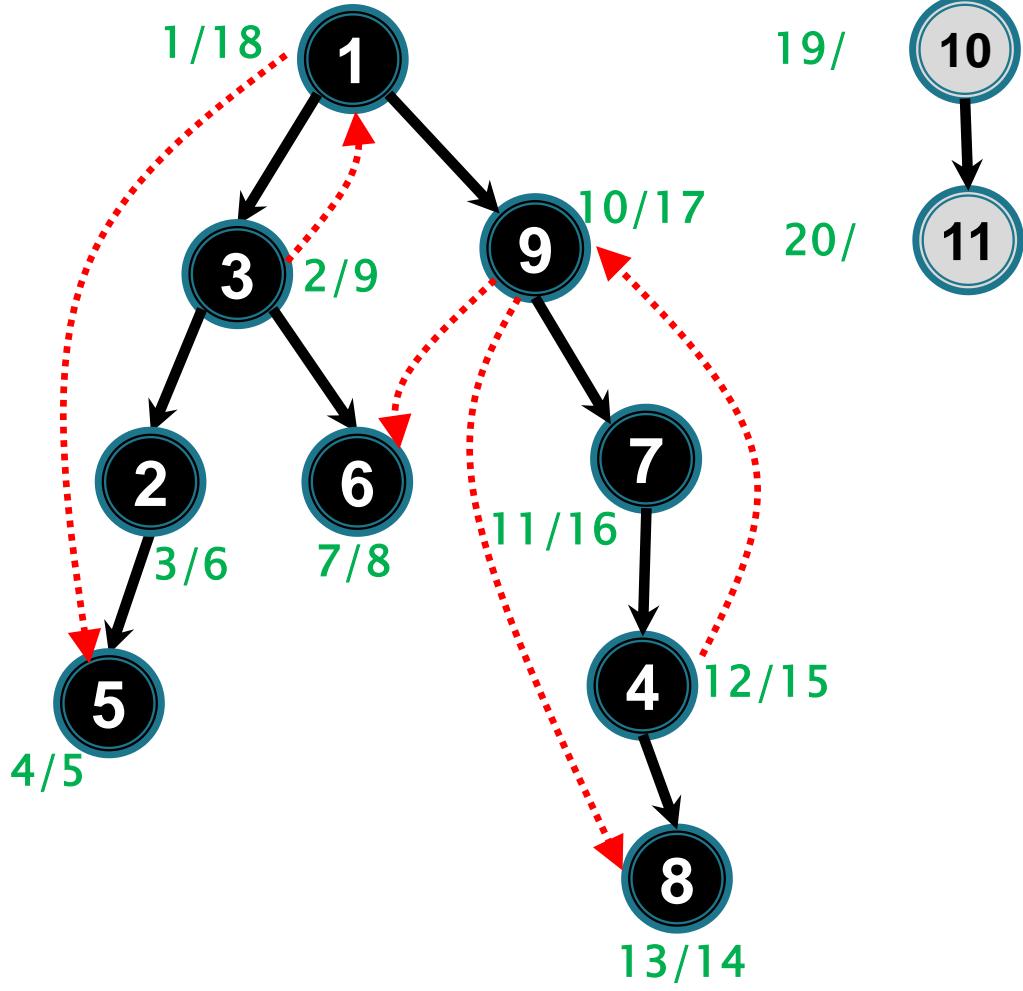
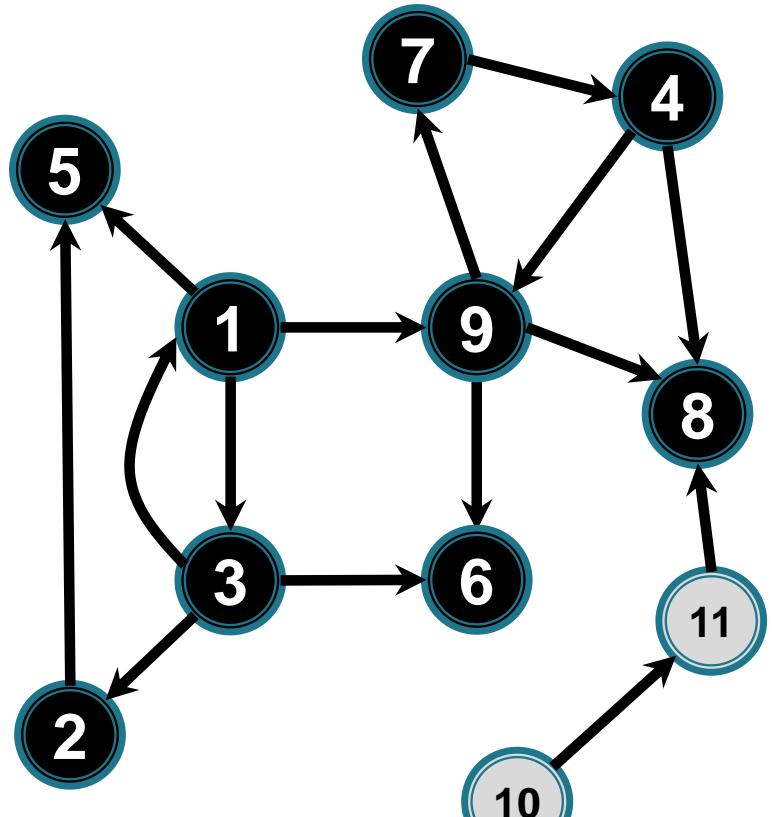
Exemplu DF graf orientat



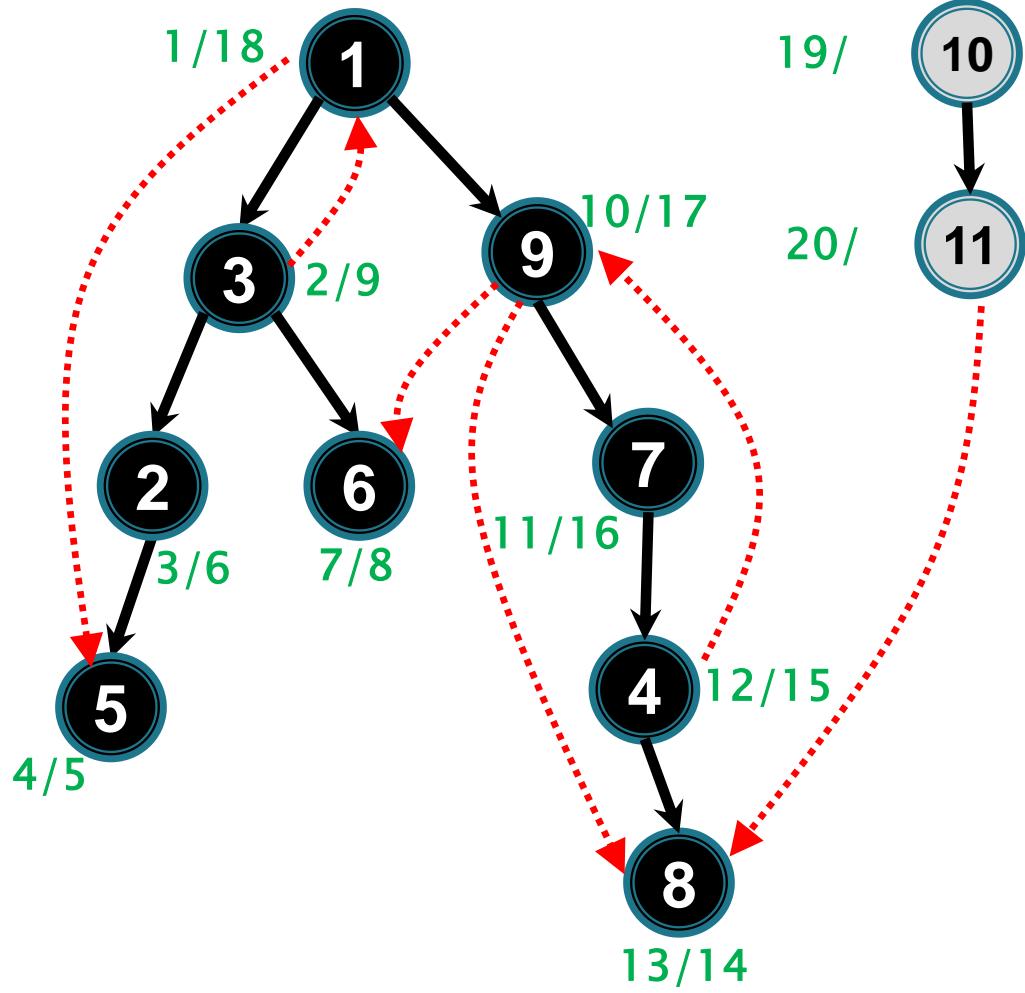
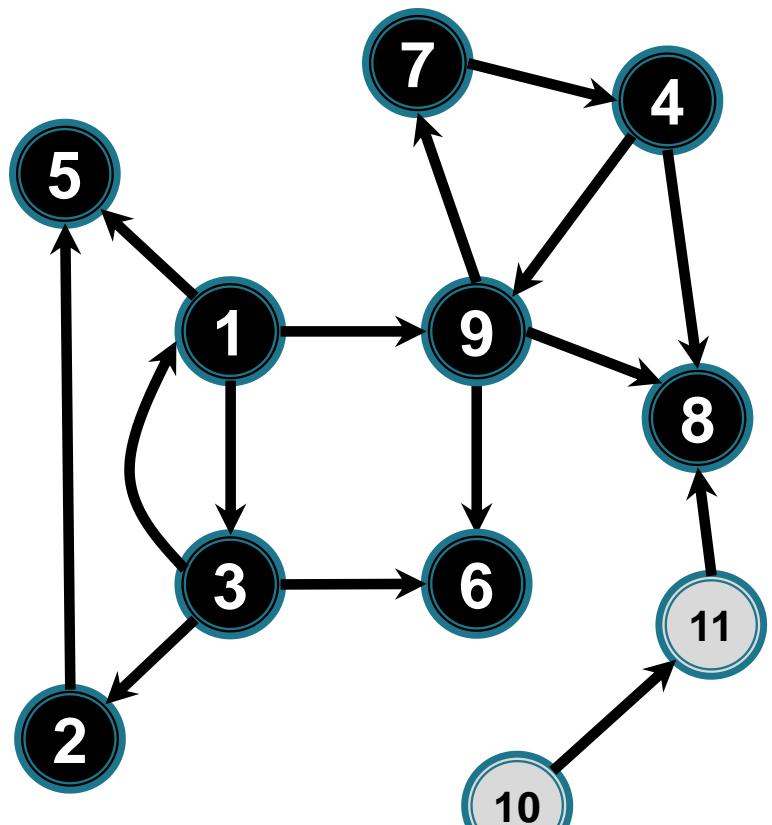
Exemplu DF graf orientat



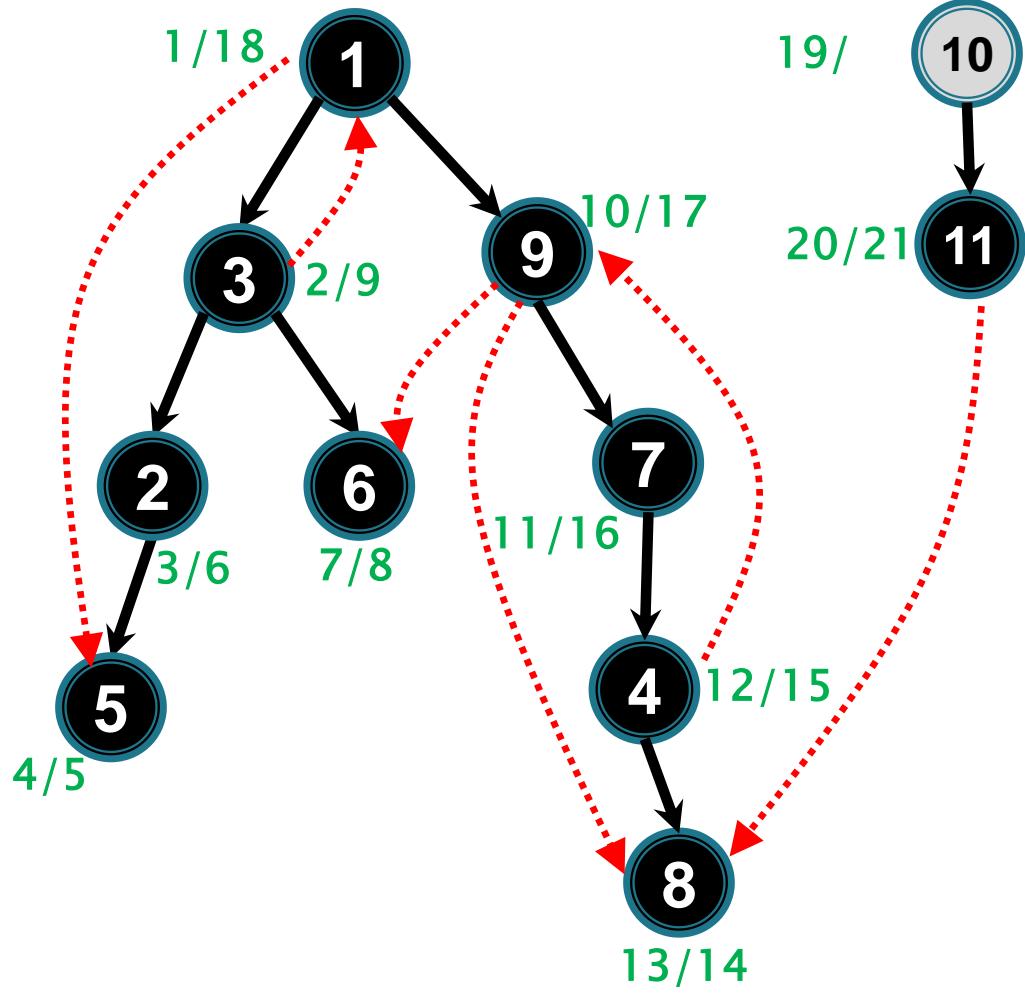
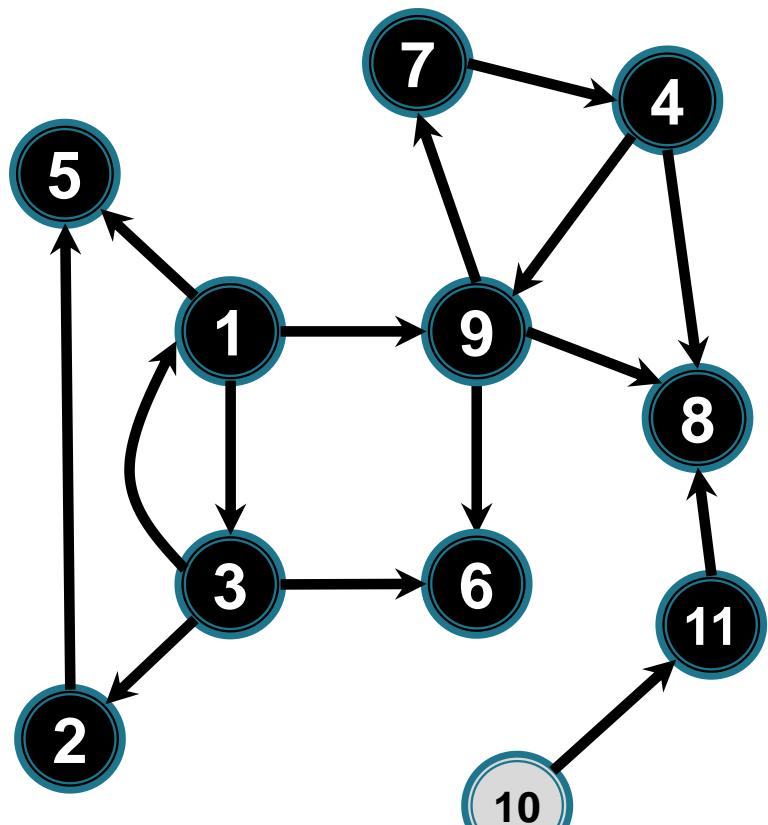
Exemplu DF graf orientat



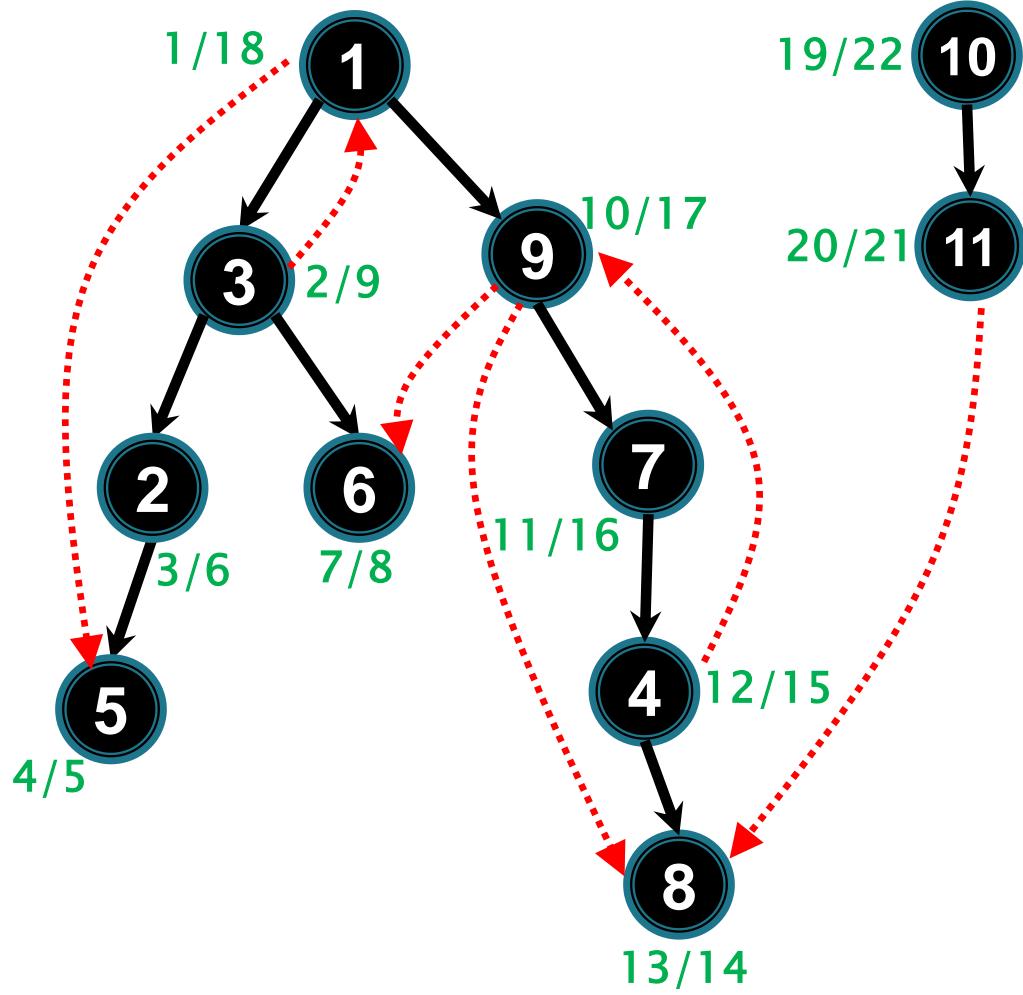
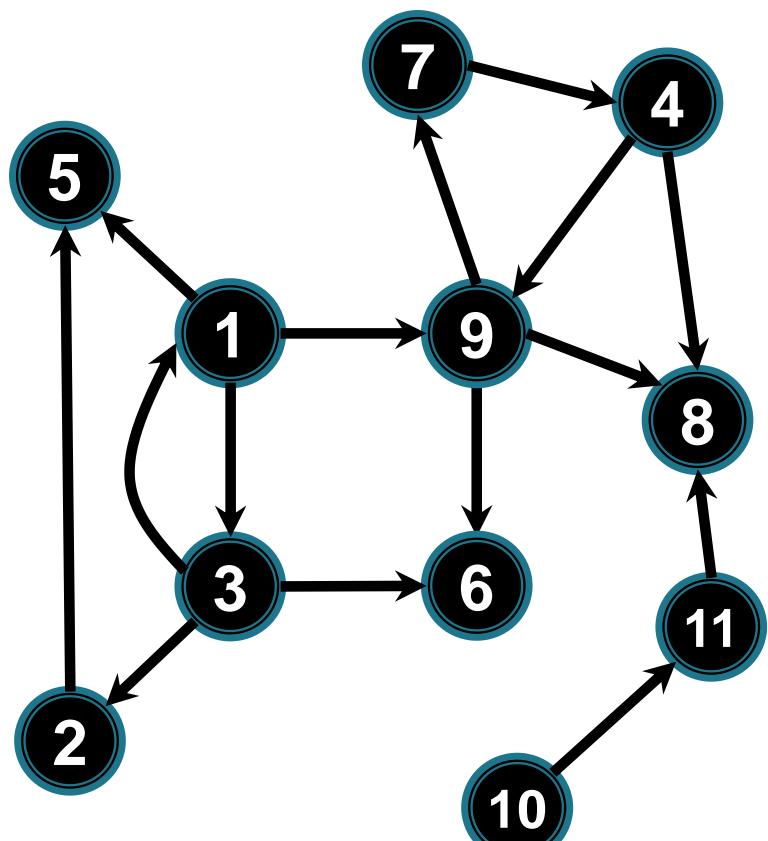
Exemplu DF graf orientat



Exemplu DF graf orientat



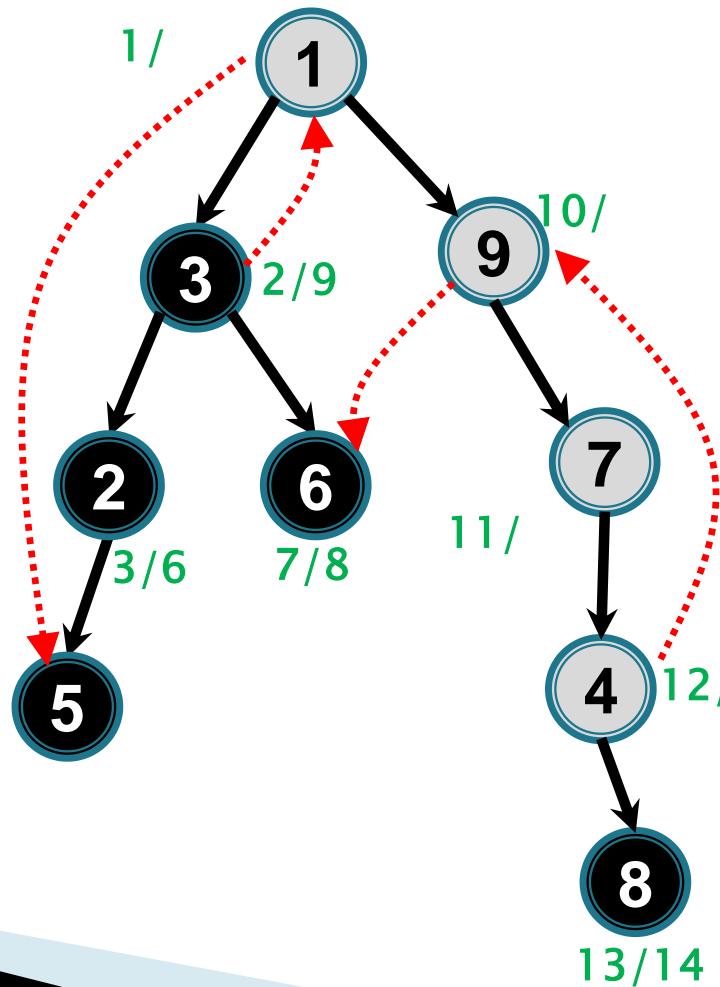
Exemplu DF graf orientat



Proprietăți DF

Vârfurile gri (descoperite, încă în explorare, “în stivă”)

- = vârfurile de pe drumul de la vârful de start la cel curent
- = vârfurile din stiva (de recursivitate)



Proprietăți DF

Proprietate

Vârful v este un **descendent** al lui u în pădurea de adâncime DF pentru un graf G (orientat sau neorientat)

dacă și numai dacă

vârful v este descoperit în timp ce u este gri (este încă în explorare)

Au loc rezultate mai generale:

Proprietăți DF

Teorema parantezelor

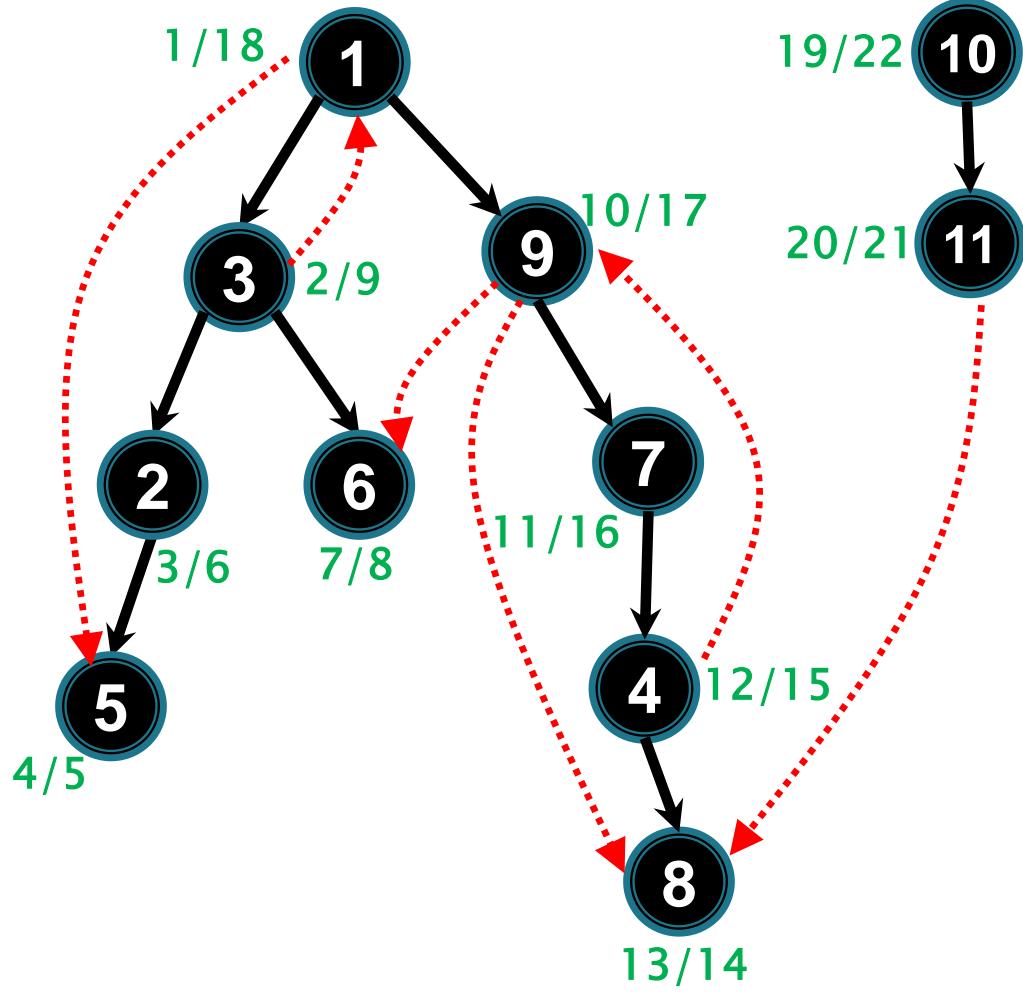
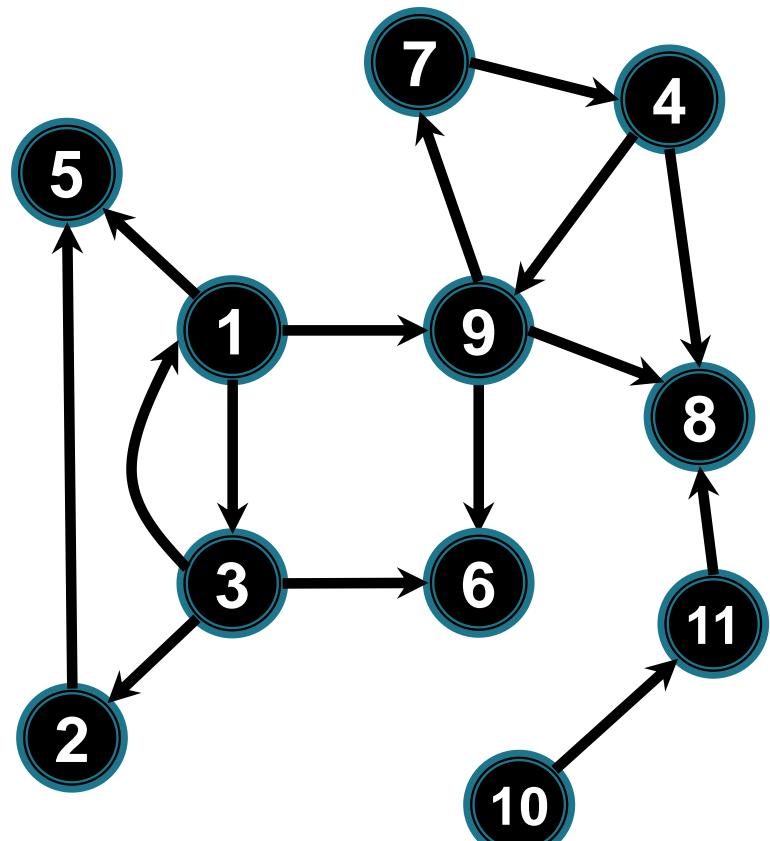
În orice căutare în adâncime a unui graf (orientat sau neorientat) $G = (V, E)$, pentru orice două vârfuri u și v cu

$$\text{desc}[u] < \text{desc}[v]$$

exact una din următoarele condiții este adevărată:

-

-



Proprietăți DF

Teorema parantezelor

În orice căutare în adâncime a unui graf (orientat sau neorientat) $G = (V, E)$, pentru orice două vârfuri u și v cu

$$\text{desc}[u] < \text{desc}[v]$$

exact una din următoarele condiții este adevărată:

- intervalele $[\text{desc}[u], \text{fin}[u]]$ și $[\text{desc}[v], \text{fin}[v]]$ sunt **disjuncte** și u și v nu sunt unul **descendentul celuilalt** în arborele/pădurea DF
- $[\text{desc}[v], \text{fin}[v]] \subset [\text{desc}[u], \text{fin}[u]]$,
caz în care v este un **descendent al lui u** în arborele/pădurea DF

Proprietăți DF

Idee de demonstrație

Avem

$$\text{desc}[u] < \text{desc}[v]$$

- $\text{desc}[v] > \text{fin}[u]$: intervalele sunt disjuncte; u este deja negru când v a fost descoperit, deci v nu este descendant al lui u și nici invers
- $\text{desc}[v] < \text{fin}[u]$: v este descoperit în timp ce u este gri (în explorare), deci este descendant al lui u; atunci v va fi finalizat înaintea lui u , deci

$$[\text{desc}[v], \text{fin}[v]] \subset [\text{desc}[u], \text{fin}[u]]$$

Proprietăți DF

Proprietate (Incluziunea intervalelor descendenților)

Vârful v este un **descendent** al lui u în pădurea de adâncime DF pentru un graf G (orientat sau neorientat)

\Leftrightarrow

vârful v este descoperit în timp ce u este gri (este încă în explorare)

\Leftrightarrow

$$[\text{desc}[v], \text{fin}[v]] \subset [\text{desc}[u], \text{fin}[u]]$$

Proprietăți DF

Teorema drumului alb

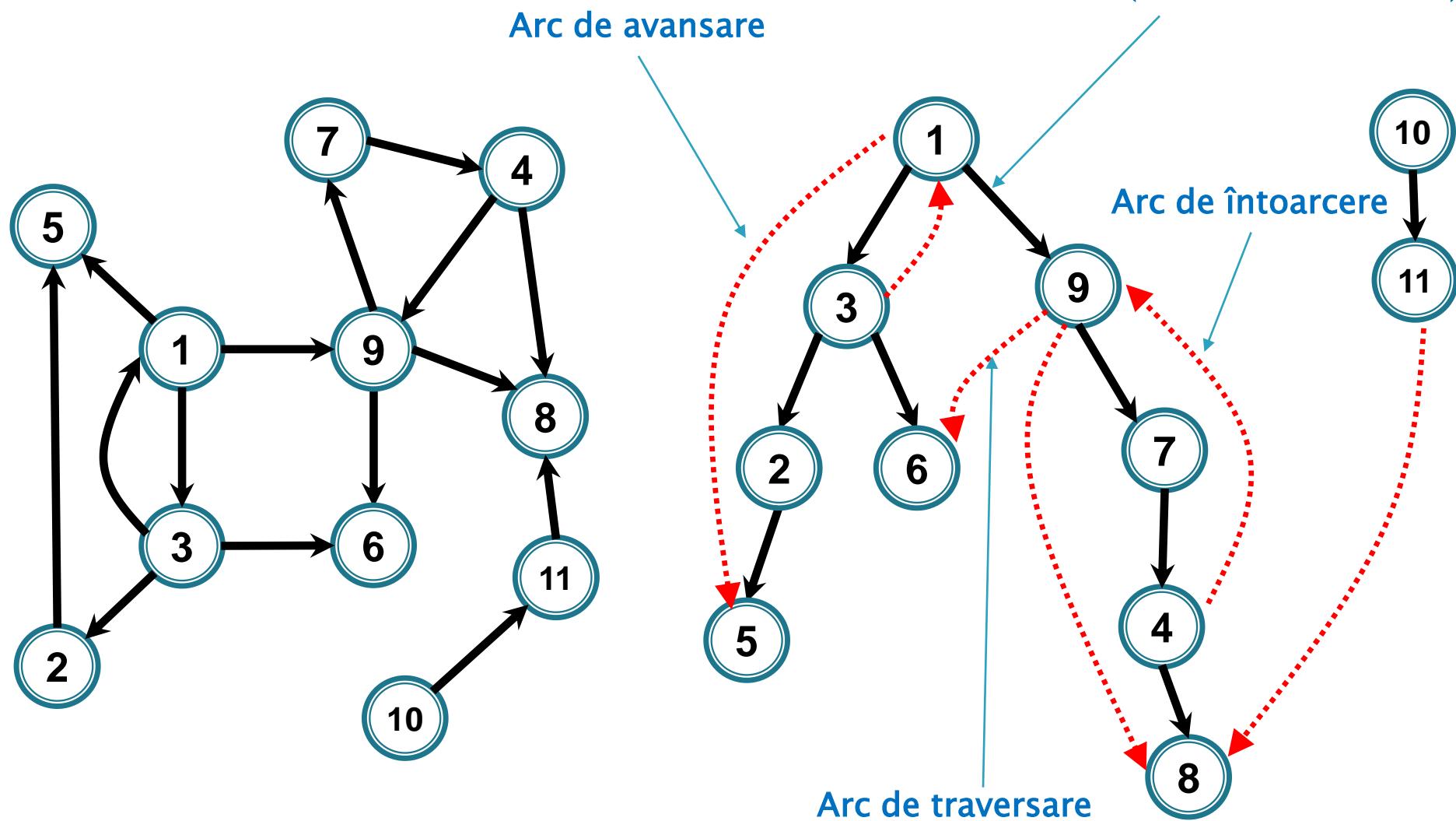
Vârful v este un **descendent** al lui u în pădurea de adâncime DF pentru un graf G (orientat sau neorientat)

\Leftrightarrow

la momentul descoperirii lui u există un drum de la u la v cu vârfurile albe

Proprietăți DF

- În raport cu pădurea DF muchiile/arcele se împart în 4 categorii:



Proprietăți DF

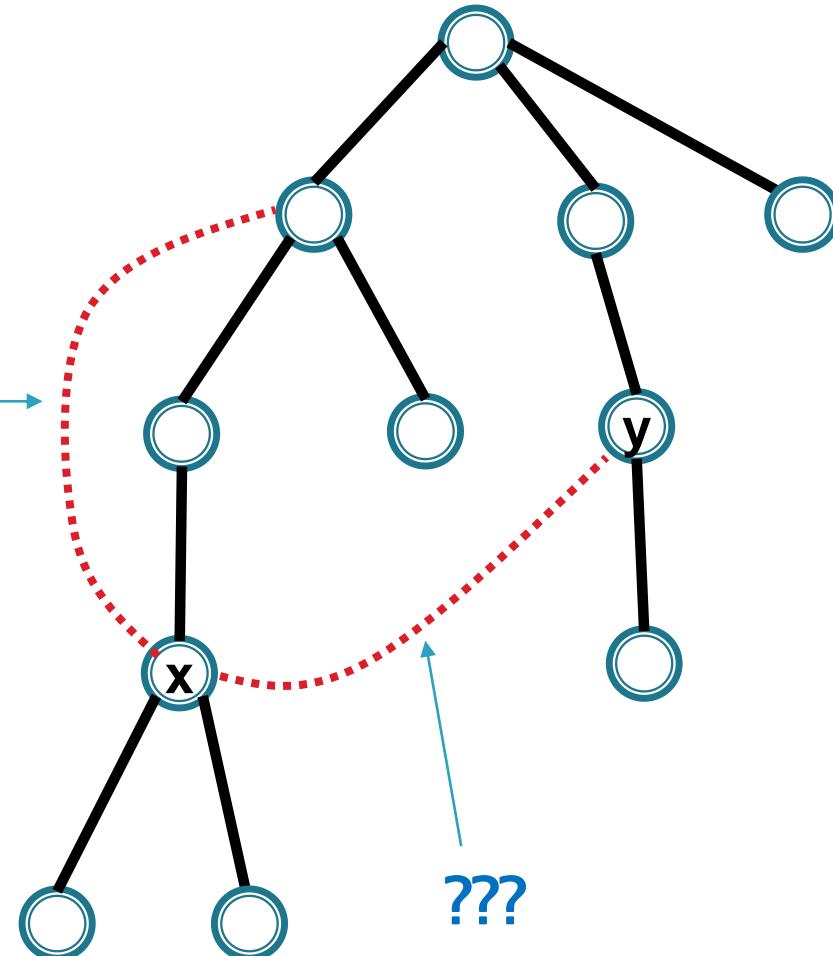
În raport cu pădurea DF muchiile/arcele se împart în 4 categorii:

- ▶ **de arbore DF (de avansare directă/ de arbore):** din pădurea DF
(u, v) cu v fiu al lui u (v a fost descoperit din u)
- ▶ **de întoarcere (înapoi) :**
(u, v) cu v strămoș al lui u (și care nu este de arbore)
- ▶ **de avansare (de avansare înainte) :**
(u, v) cu v descendant al lui u care nu este însă fiu al lui
- ▶ **de traversare (transversale) :**
toate celelalte muchii;
pot fi între vârfuri din același arbore din pădurea DF cu condiția ca unul să nu fie strămoșul celuilalt sau pot uni vârfuri din arbori diferiți

Proprietăți DF

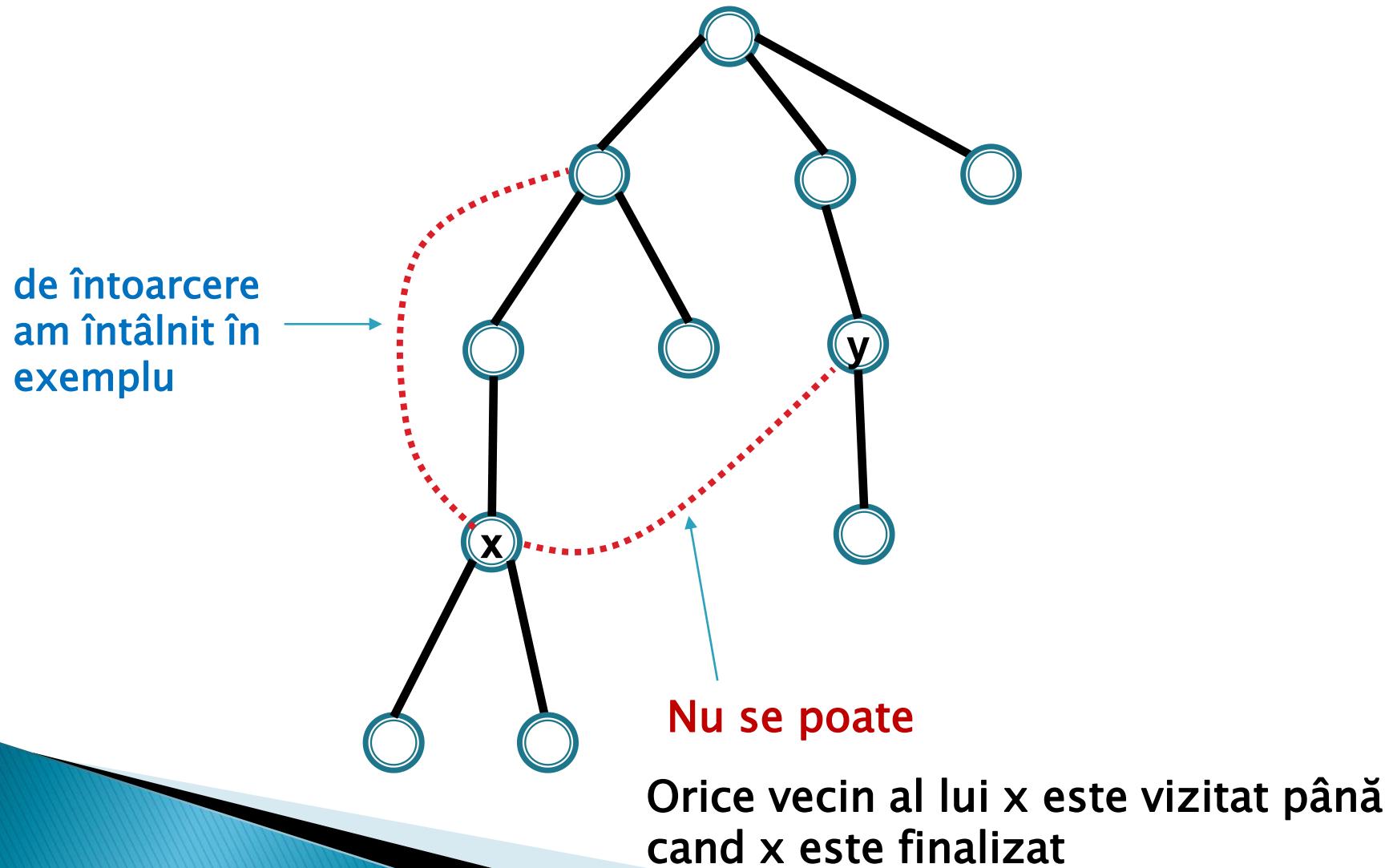
Ce tipuri de muchii pot apărea în cazul unui graf **neorientat** ?

de întoarcere
am întâlnit în
exemplu



Proprietăți DF

Ce tipuri de muchii pot apărea în cazul unui graf **neorientat** ?



Proprietăți DF

În cazul unui graf **neorientat** există doar două tipuri de muchii în raport cu pădurea DF:

- ▶ muchii de arbore (de avansare)
- ▶ muchii de întoarcere (către un ascendent)
 - !! care nu este tata
 - **închide un ciclu**

O muchie este detectată în timpul parcurgerii DF ca fiind de întoarcere dacă unește vârful curent cu un vârf **gri** (=vizitat + ascendent al său) **care nu este tatăl lui** => **CICLU ÎN GRAF**

Determinarea de cicluri - graf neorientat

DFS (x)

viz[x] = 1

pentru fiecare $xy \in E$

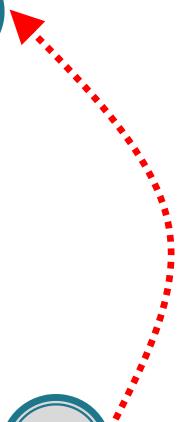
daca $\text{viz}[y]==0$ atunci

$\text{tata}[y] = x$

DFS(y)

altfel

daca $\text{tata}[x] != y$ atunci //xy nu e muchie din arbore=>
//este de intoarcere, y e gri sigur



Determinarea de cicluri - graf neorientat

DFS(x)

```
viz[x] = 1  
pentru fiecare xy ∈ E  
    daca viz[y]==0 atunci  
        tata[y] = x  
        DFS(y)  
    altfel  
        daca tata[x] != y atunci //xy nu e muchie din arbore=>  
            //este de intoarcere  
        v = x;  
        cat timp v!=y execută  
            adauga v la ciclu  
            v = tata[v]  
        adauga y,x la ciclu  
STOP
```

Proprietăți DF

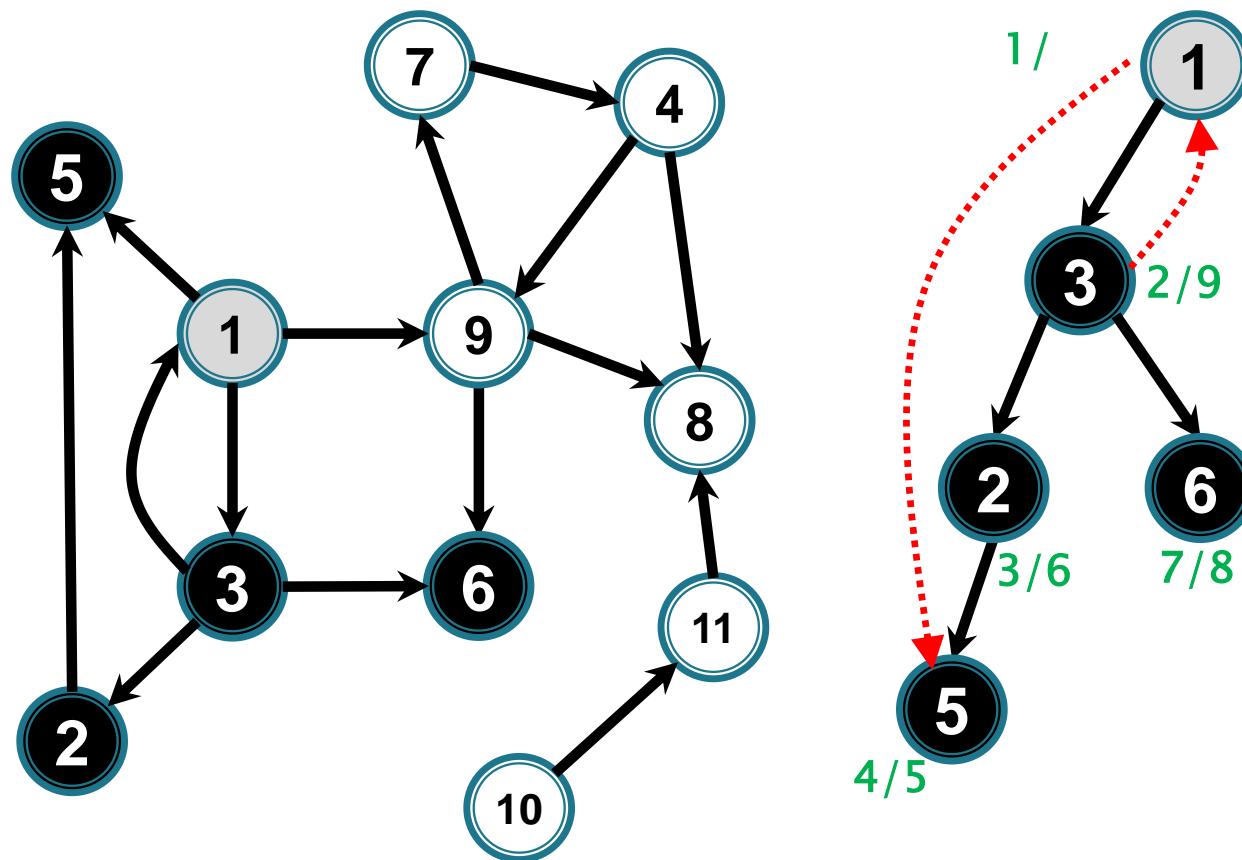
În cazul unui graf **orientat** pot exista toate cele 4 tipuri de muchii în raport cu pădurea DF.



Cum se determină tipul unui arc în parcurgerea DF?

Un arc (u,v) este detectat ca fiind

- de avansare directă (de arbore): v este nevizitat (alb)
- de avansare înainte:

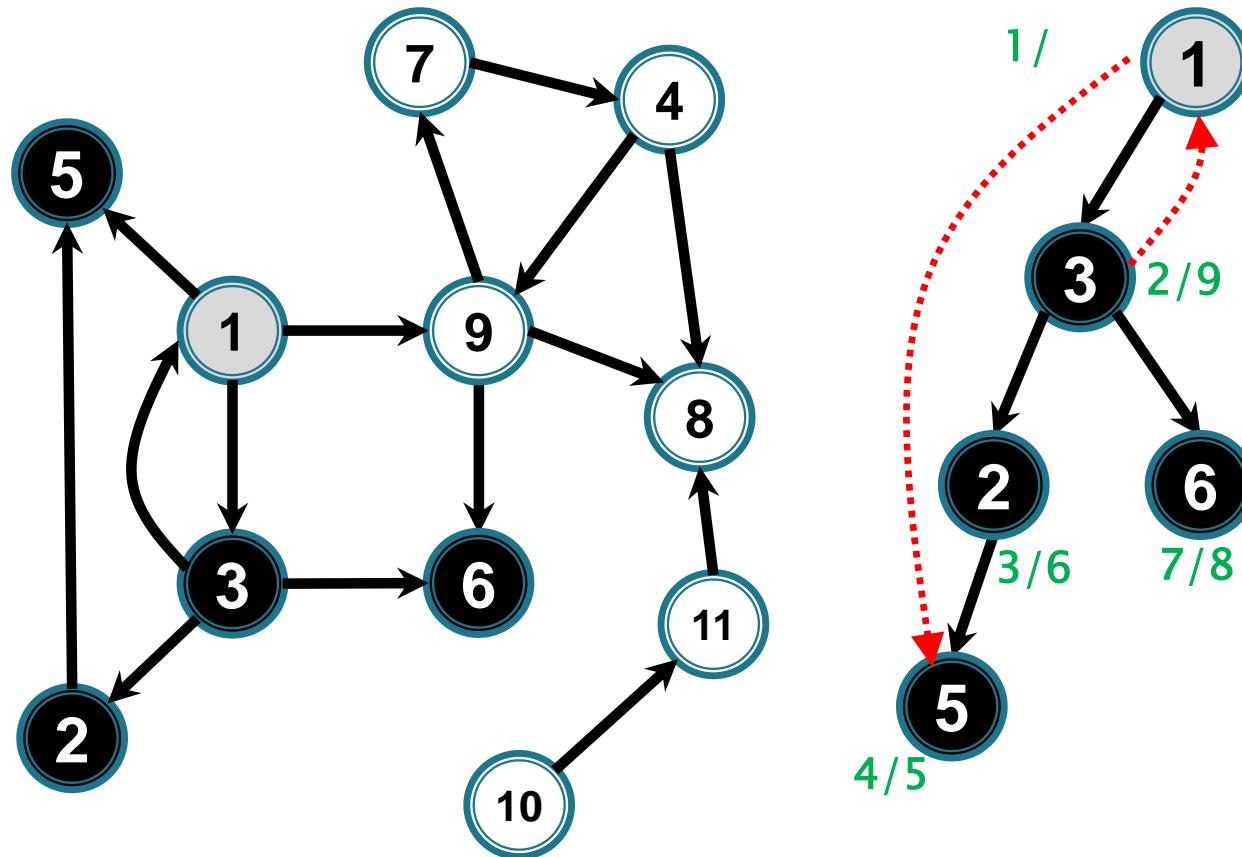


Un arc (u,v) este detectat ca fiind

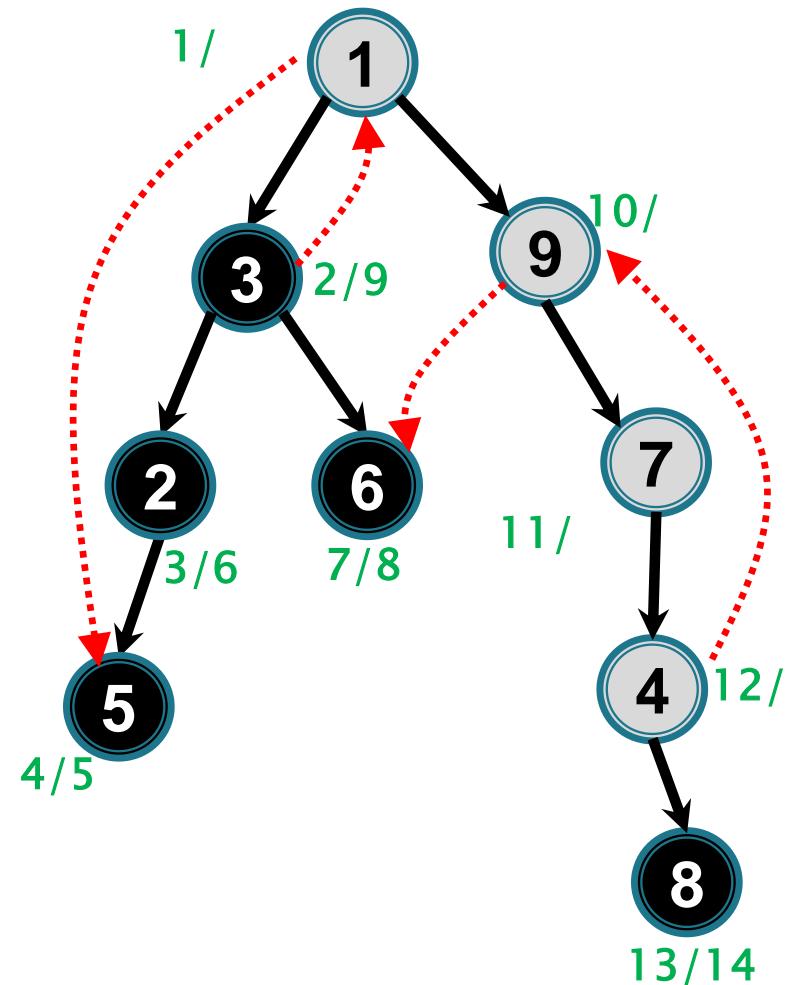
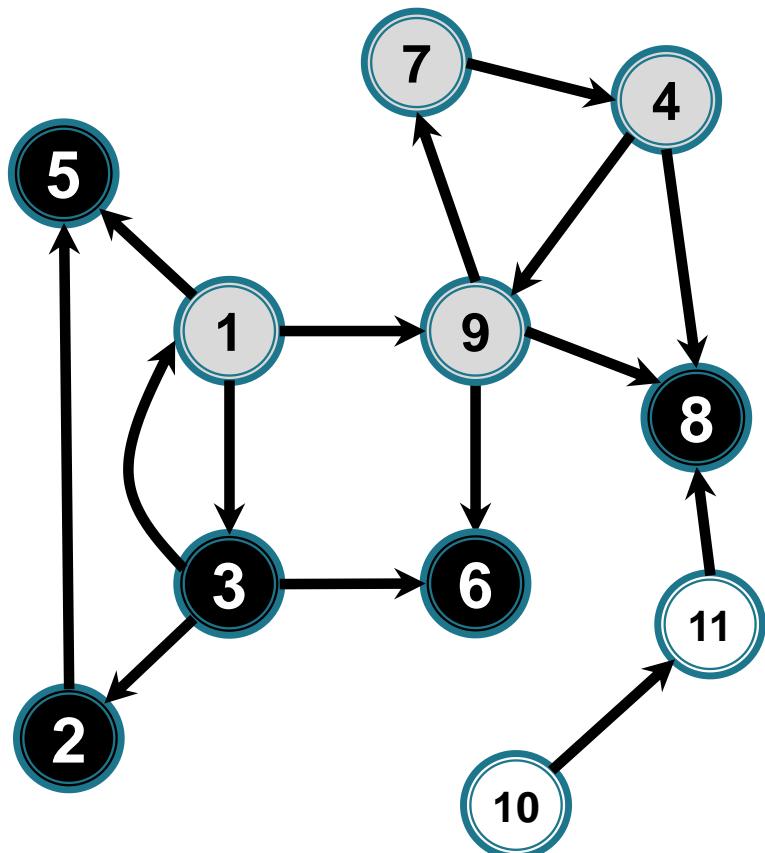
- de avansare directă (de arbore): v este nevizitat (alb)
- de avansare înainte:

`desc[u] < desc[v] < fin[v] < fin[u]` și v nu e fiu \Leftrightarrow

când este parcurs arcul uv , **v este negru** (finalizat) și este **descendent al lui u** , adică $desc[u] < desc[v]$



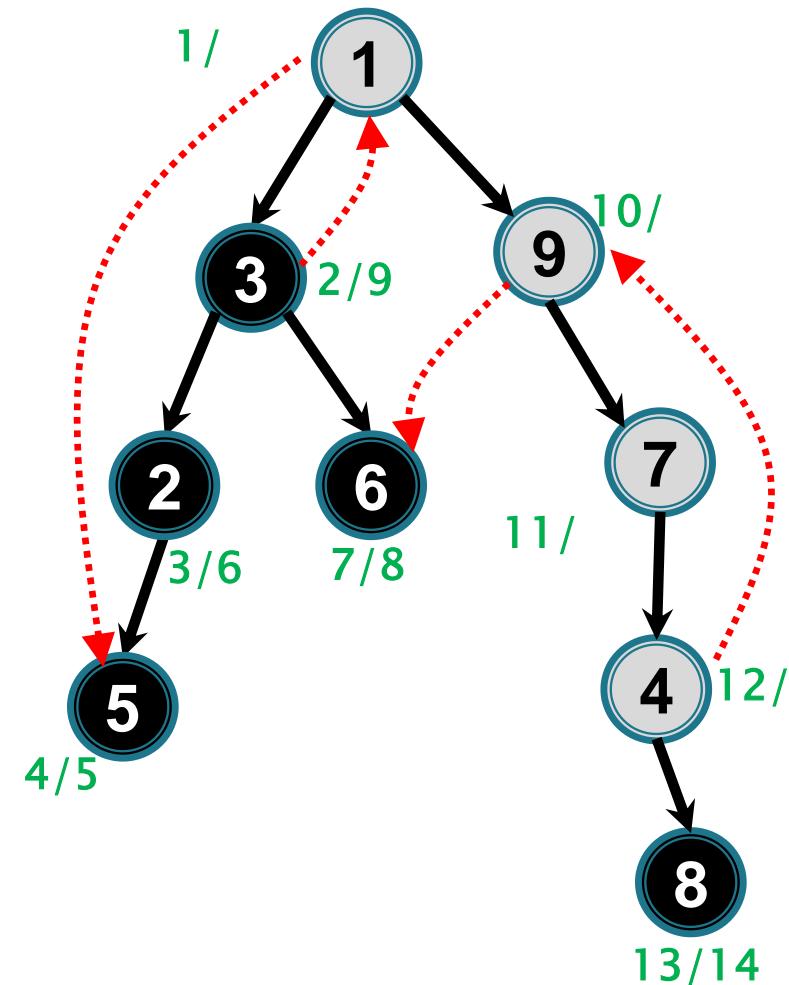
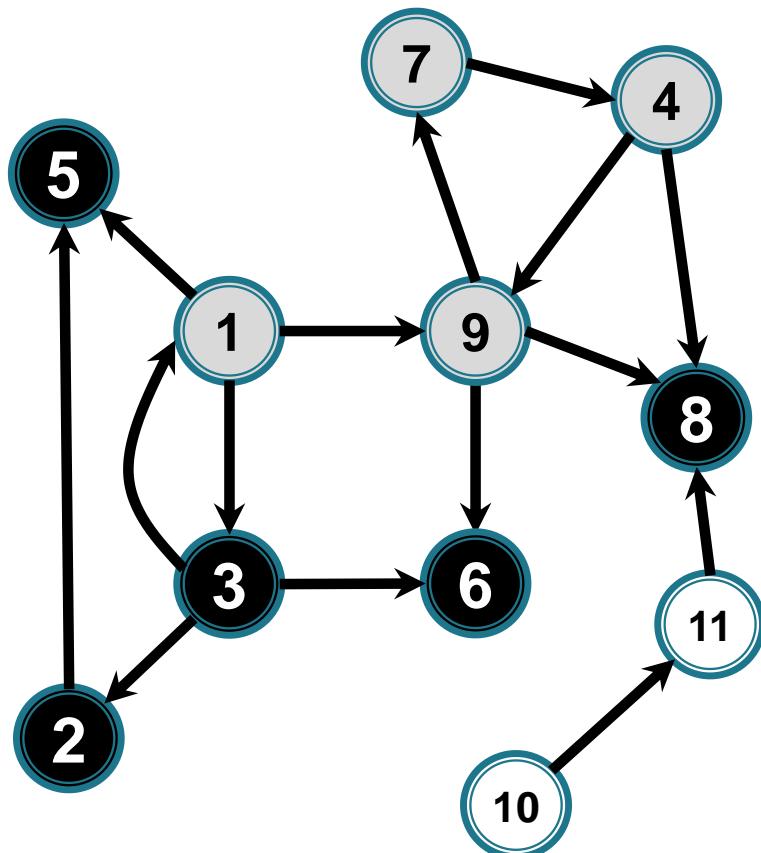
- de întoarcere:



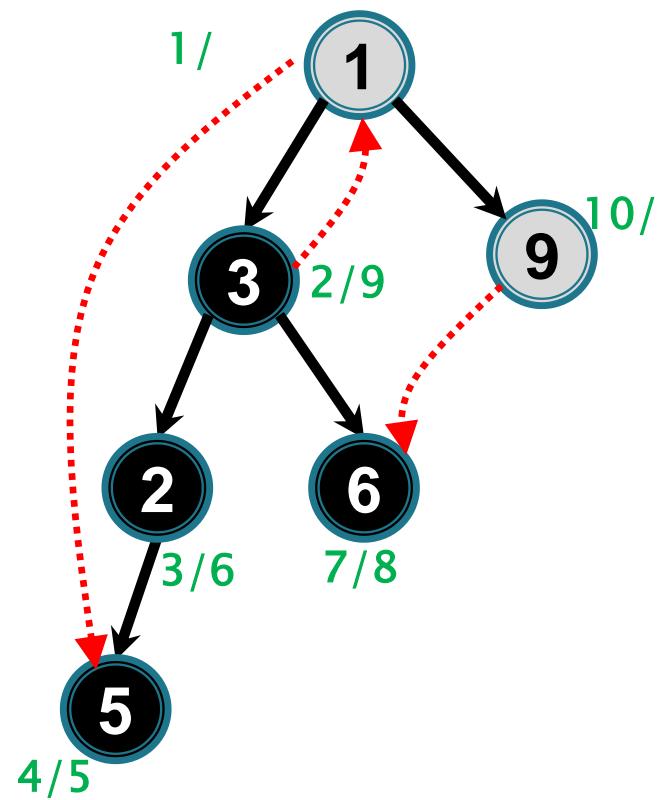
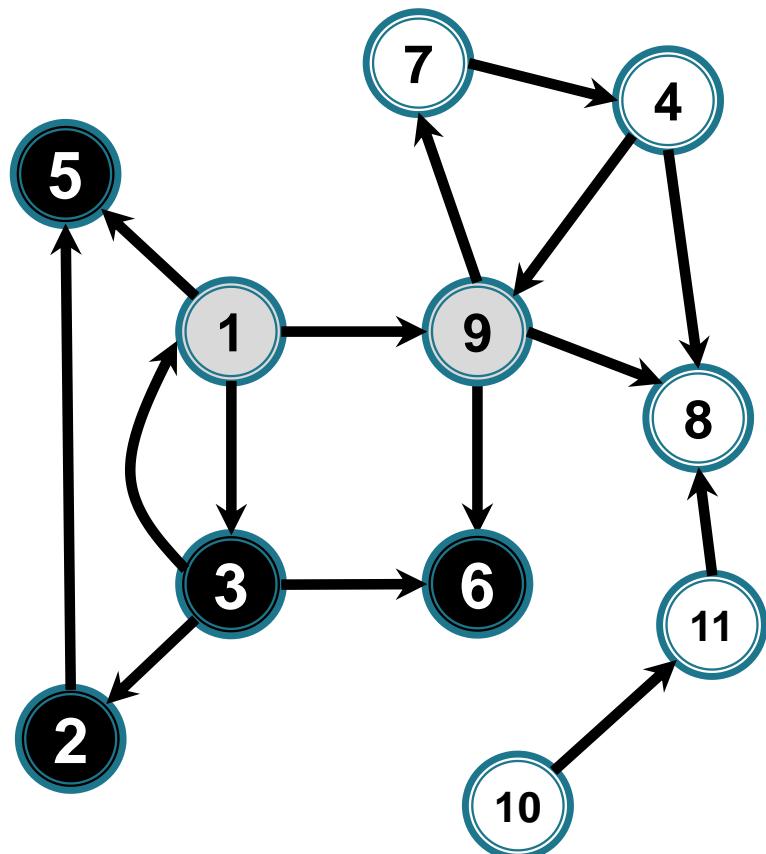
- de întoarcere:

`desc[v] < desc[u] < fin[u] < fin[v] \Leftrightarrow`

când este parcurs arcul uv , v este încă gri (nefinalizat, în stivă)



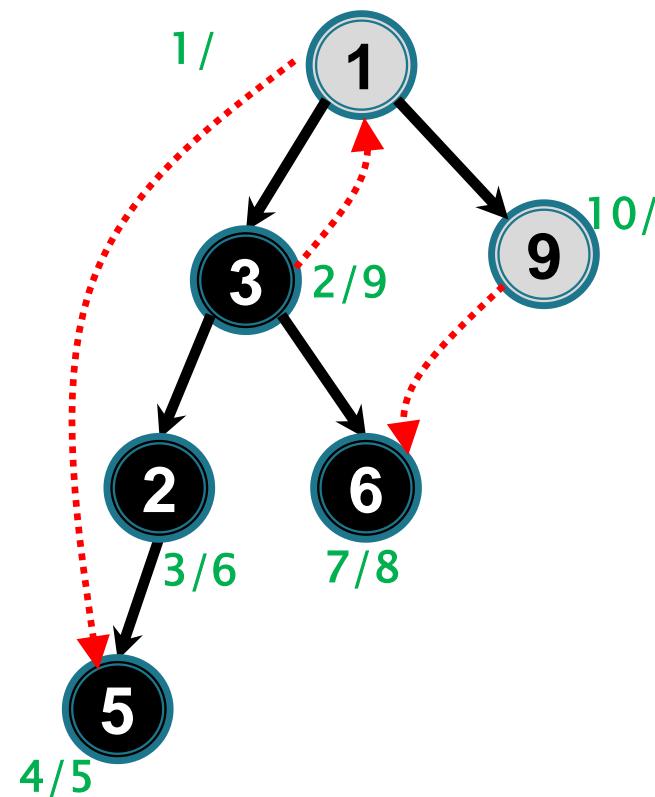
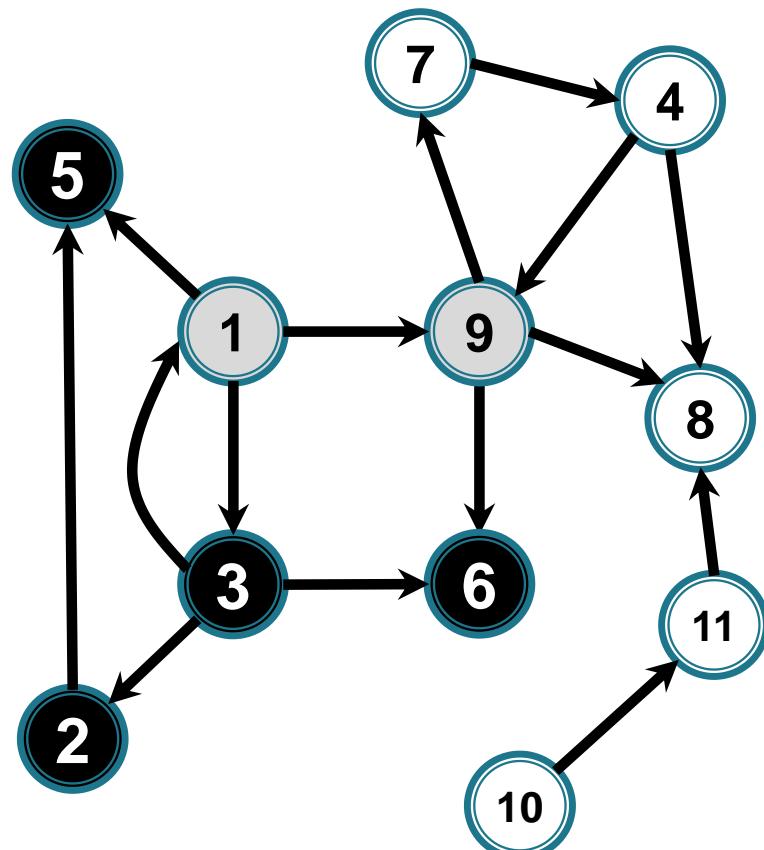
- de traversare:



- de traversare:

$\text{desc}[v] < \text{fin}[v] < \text{desc}[u] < \text{fin}[u] \Leftrightarrow$

când este parcurs arcul uv , v este negru și nu este descendenta al lui u , adică $\text{desc}[u] > \text{desc}[v]$



Proprietăți DF

- ▶ Ce fel de arce închid circuite cu arcele din arborele DFS?

Proprietăți DF

În cazul unui graf **orientat** doar un arc de întoarcere închide un circuit cu arcele din arborele DF

Un arc este detectat în timpul parcurgerii DF ca fiind de întoarcere dacă unește vârful curent cu un vârf **gri**, adică **nefinalizat**= **în stivă** (!!poate fi tatăl său) => **CIRCUIT ÎN GRAF**

Determinarea de cicluri / circuite

Concluzii

- ▶ Ciclu/circuit – închis doar de muchii/arce de întoarcere

Un graf neorientat/orientat G are ciclu/circuit dacă și numai dacă există muchie/arc de întoarce uv în pădurea DFS

= de la un vârf u la un vârf gri (nefinalizat)

(!! muchie/arc care nu este din pădurea DFS)

Determinarea de cicluri - graf neorientat

DFS (x)

```
viz[x] = 1  
pentru fiecare xy ∈ E  
daca viz[y]==0 atunci
```

```
tata[y] = x
```

```
DFS(y)
```

```
altfel
```

```
daca tata[x] != y atunci //xy nu e muchie din arbore=>  
//este de intoarcere
```

```
v = x;
```

```
cat timp v!=y executa
```

```
adauga v la ciclu
```

```
v = tata[v]
```

```
adauga y,x la ciclu
```

```
STOP
```

Determinarea de circuit - graf orientat

DFS (x)

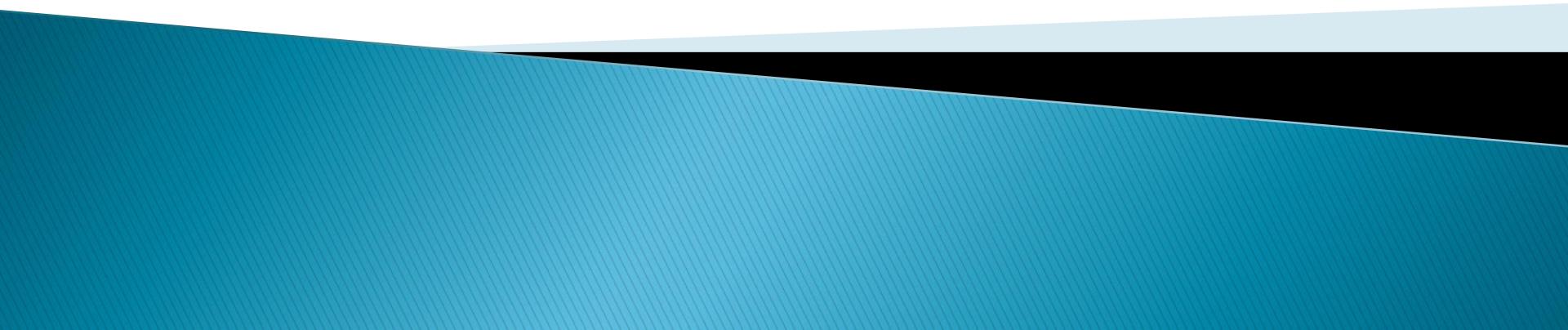
```
viz[x] = 1  
pentru fiecare xy ∈ E  
    daca viz[y]==0 atunci  
        tata[y] = x  
        DFS(y)  
    altfel  
        daca fin[y] == 0 atunci //y e gri,in explorare,in stiva
```

Determinarea de circuit - graf orientat

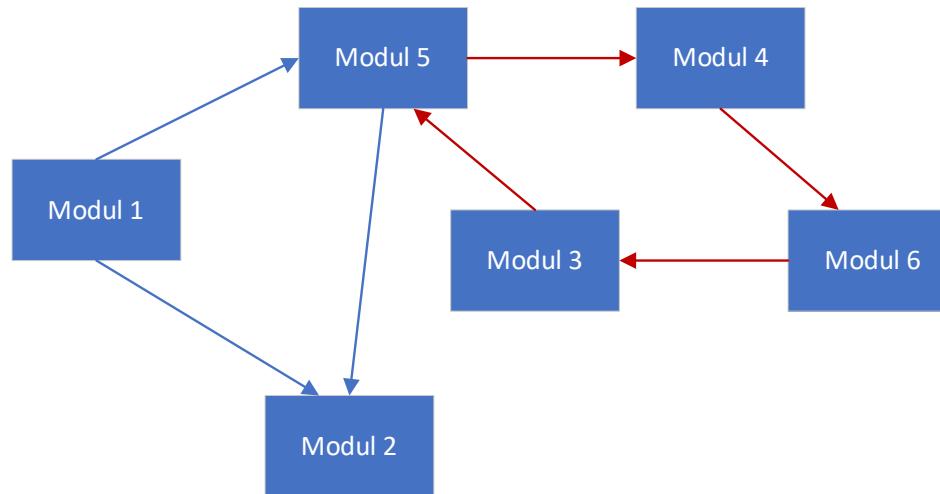
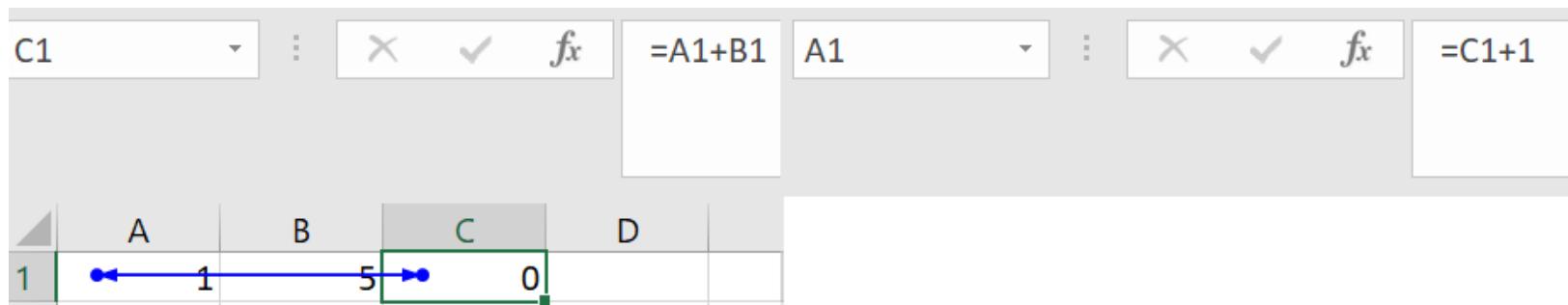
DFS(x)

```
viz[x] = 1
pentru fiecare xy ∈ E
    daca viz[y]==0 atunci
        tata[y] = x
        DFS(y)
    altfel
        daca fin[y] == 0 atunci //y e gri,in explorare,in stiva
            v = x;
            cat timp v!=y executa
                adauga v la circuit
                v = tata[v]
            adauga y,x la circuit
            inverseaza circuit
        STOP
fin[x] = 1
```

Parcurgerea în adâncime – Aplicații



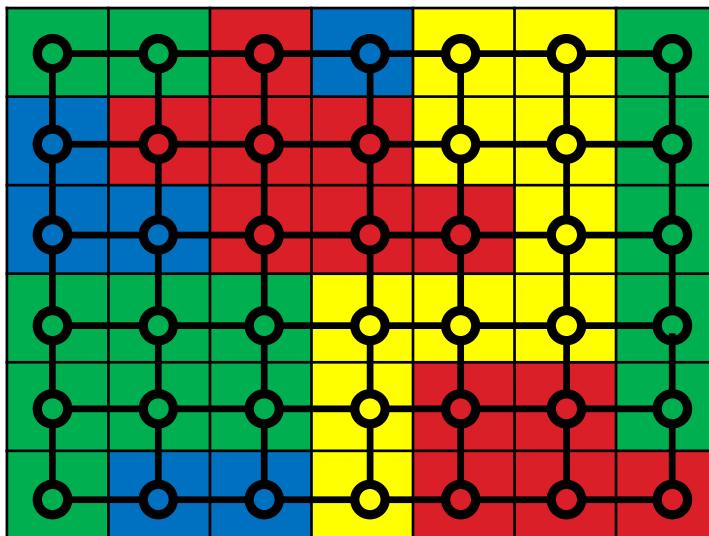
Determinarea de cicluri / circuite



Determinarea componentelor conexe

Determinarea unui arbore parțial

- ▶ Ca și la parcurgerea în lățime
- ▶ Aplicații – algoritm de umplere (fill)



Alte aplicații

- ▶ Sortare topologică (ordinea de executare a unor activități care depind una de cealaltă)
- ▶ Detectarea de componente tare conexe (“comunități”)