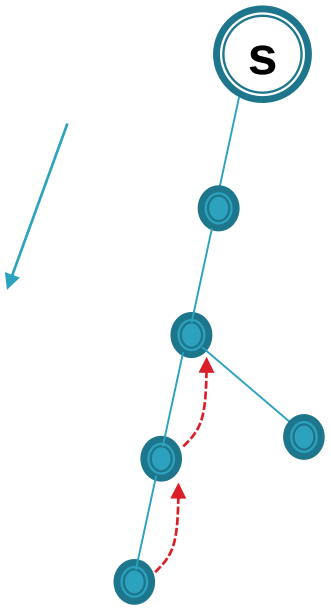
Parcurgerea în adâncime

Parcurgerea în adâncime

La un pas:

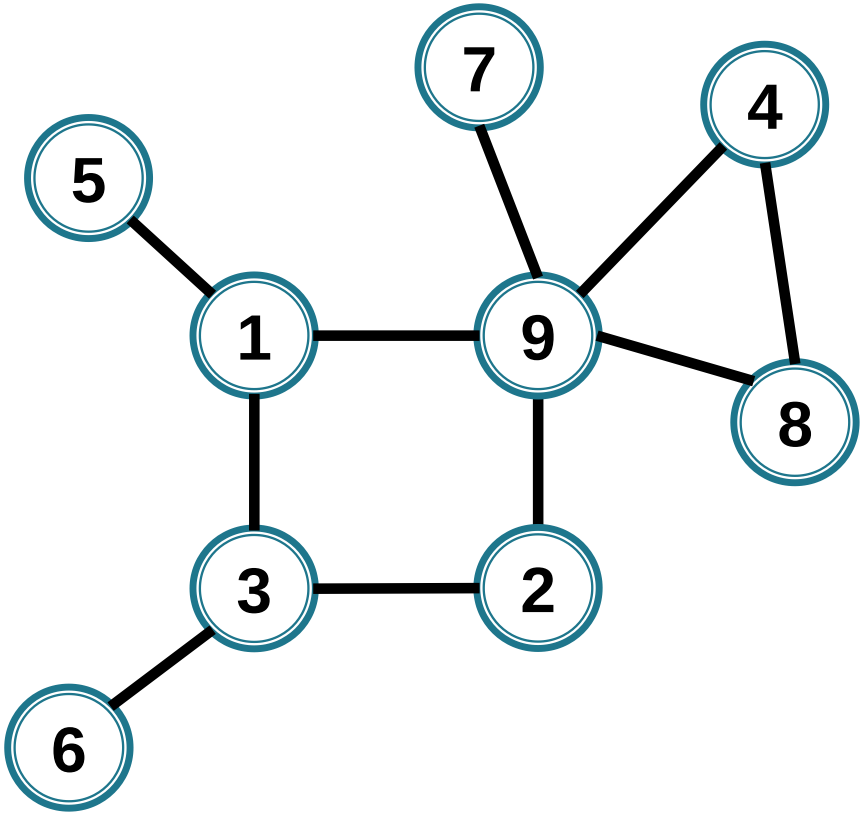
Coborâm adânc în graf,
mereu în primul vecin nevizitat
(ca o încercare de a ieși din labirint)

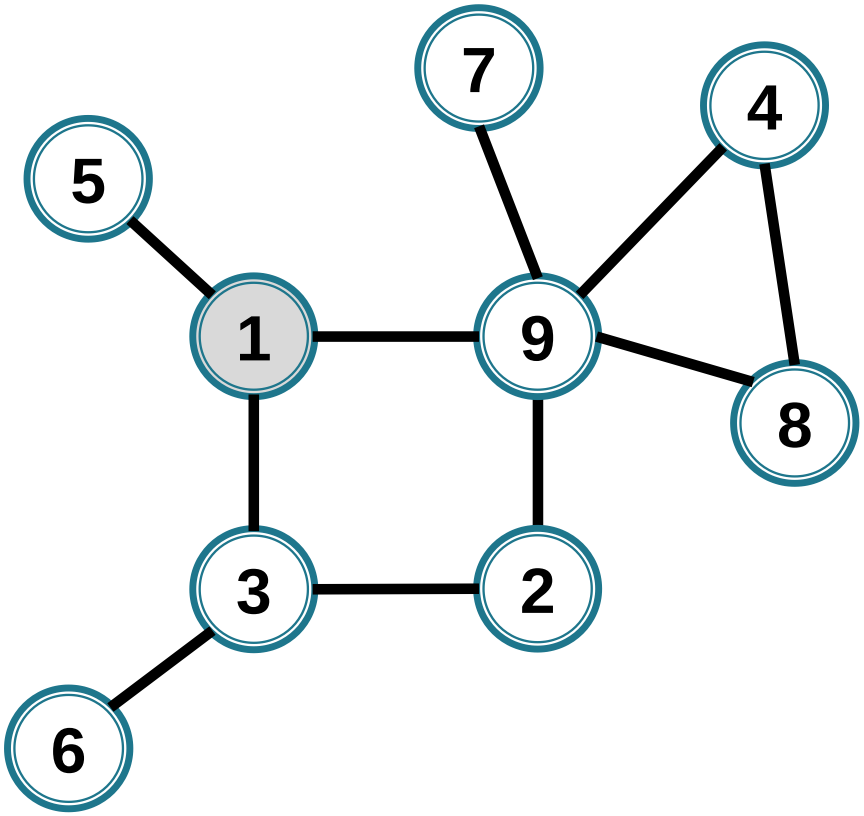


Când nu mai putem înainta
(vârful curent nu mai are vecini nevizitați)
ne întoarcem pe drumul pe care am venit (tată, bunic etc)
până găsim un vârf care mai are vecini nevizitați și reluăm coborârea

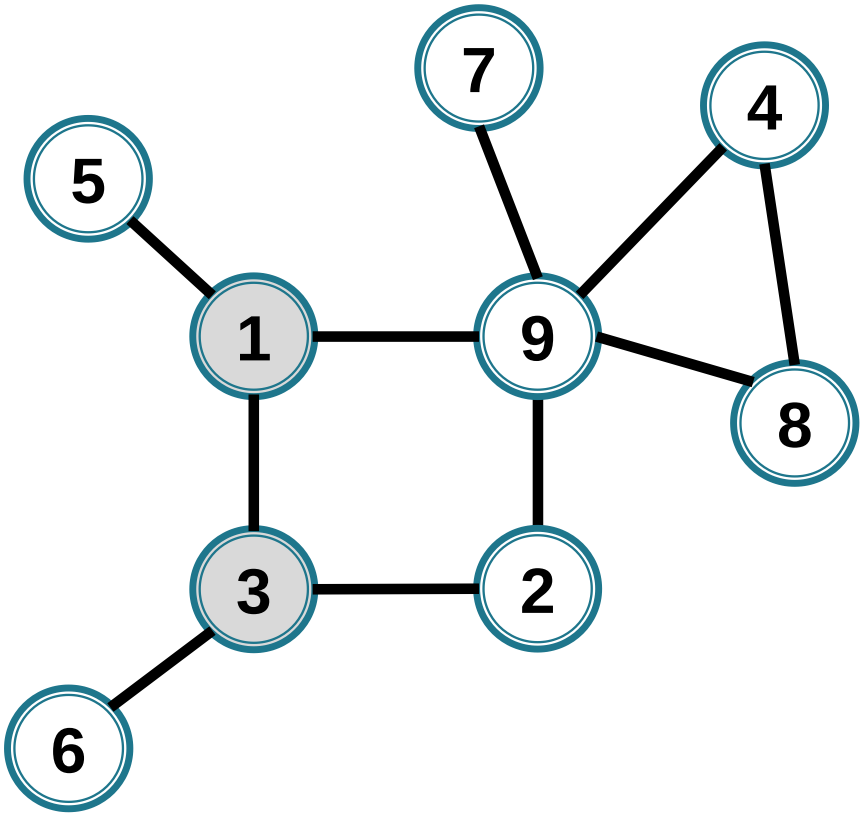
Parcurgerea în adâncime

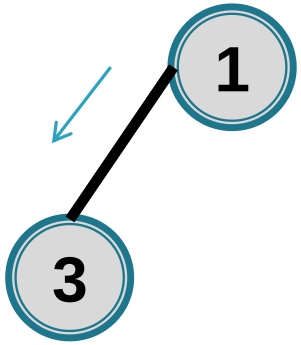
- **Inițial**: vârful de start s – devine vârf curent
- **La un pas**:
  - se trece la primul vecin nevizitat al vârfului curent, **dacă există**
  - altfel
    - **se merge înapoi pe drumul de la s la vârful curent, până se ajunge la un vârf cu vecini nevizitați (dacă nu mai există un astfel de vârf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

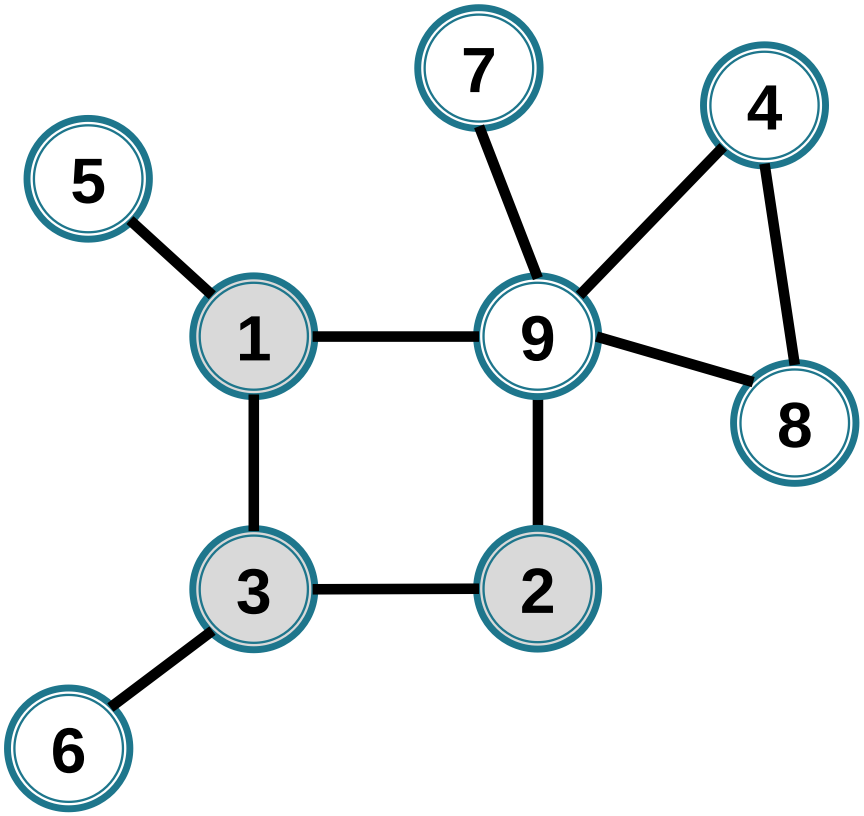

7
4
5
1
9
8
3
2
6

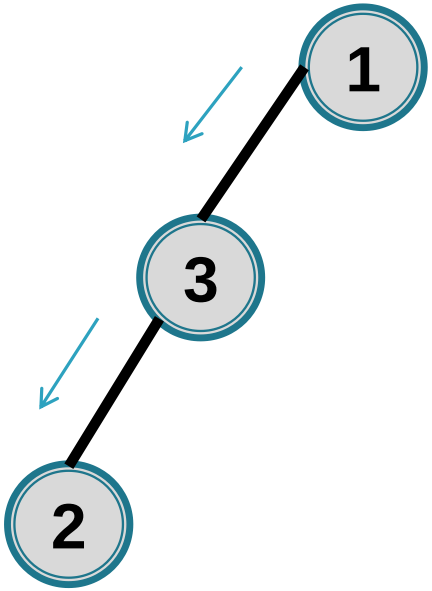


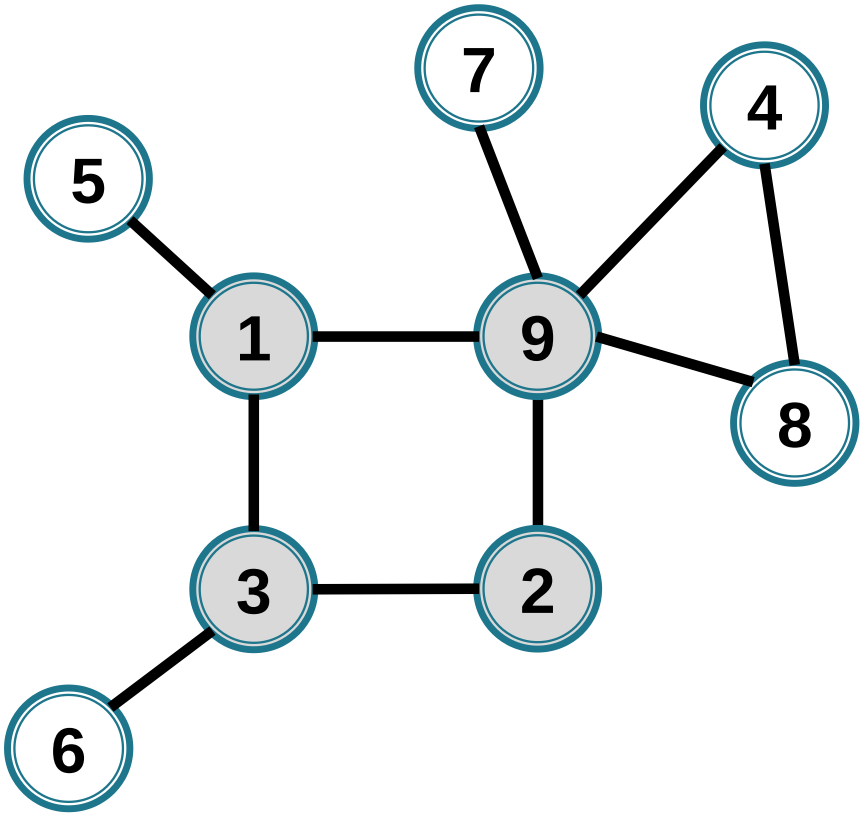
1

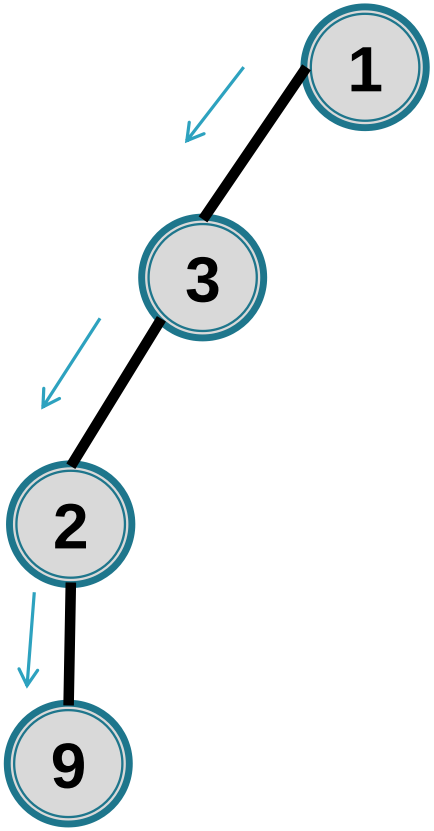

7
4
5
1
9
8
3
2
6

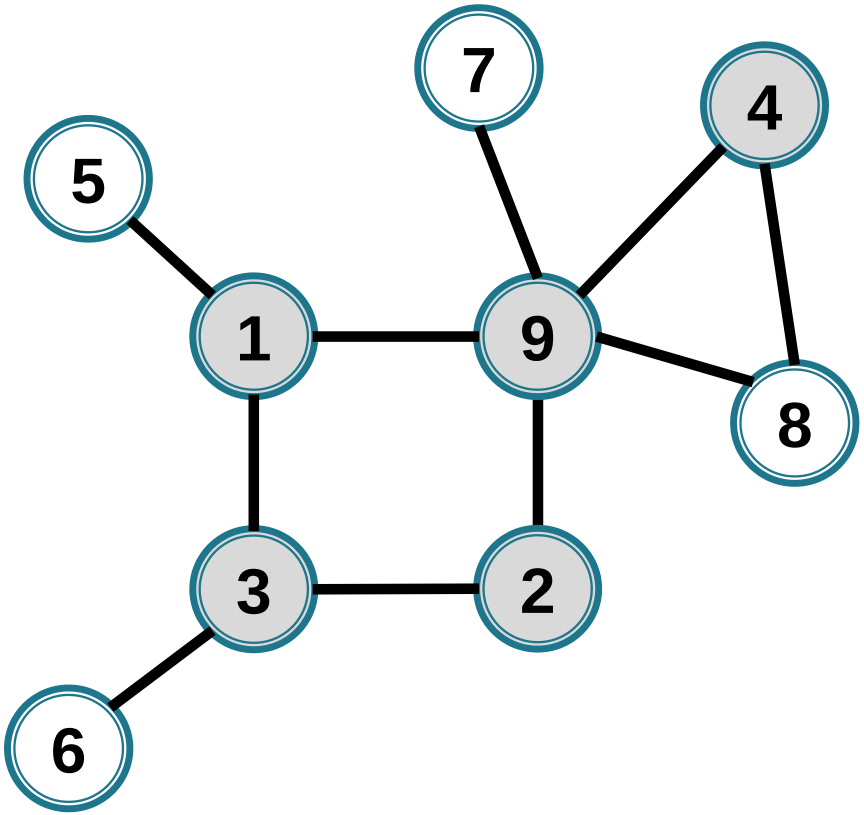

1
3

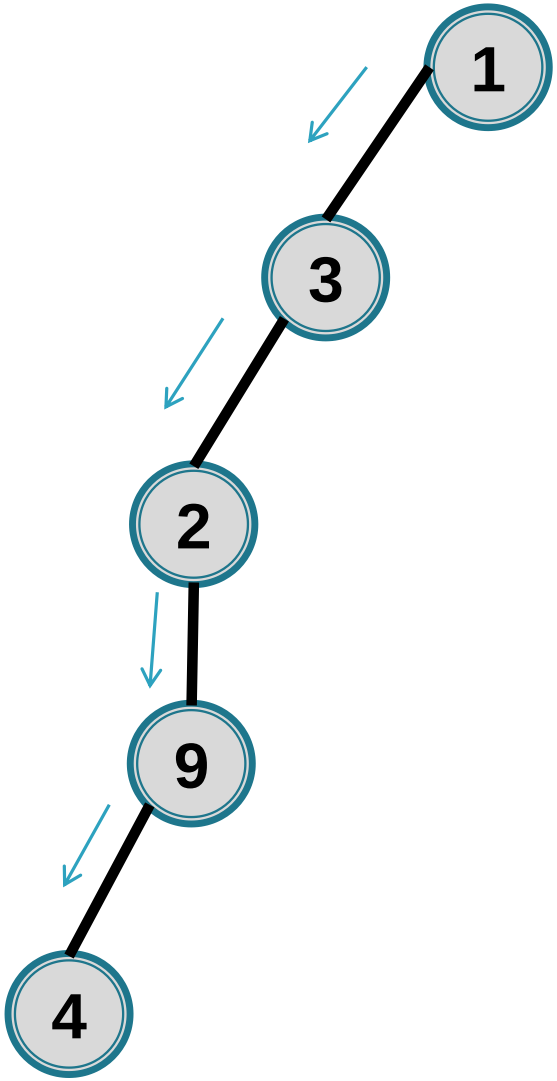

7
4
5
1
9
8
3
2
6

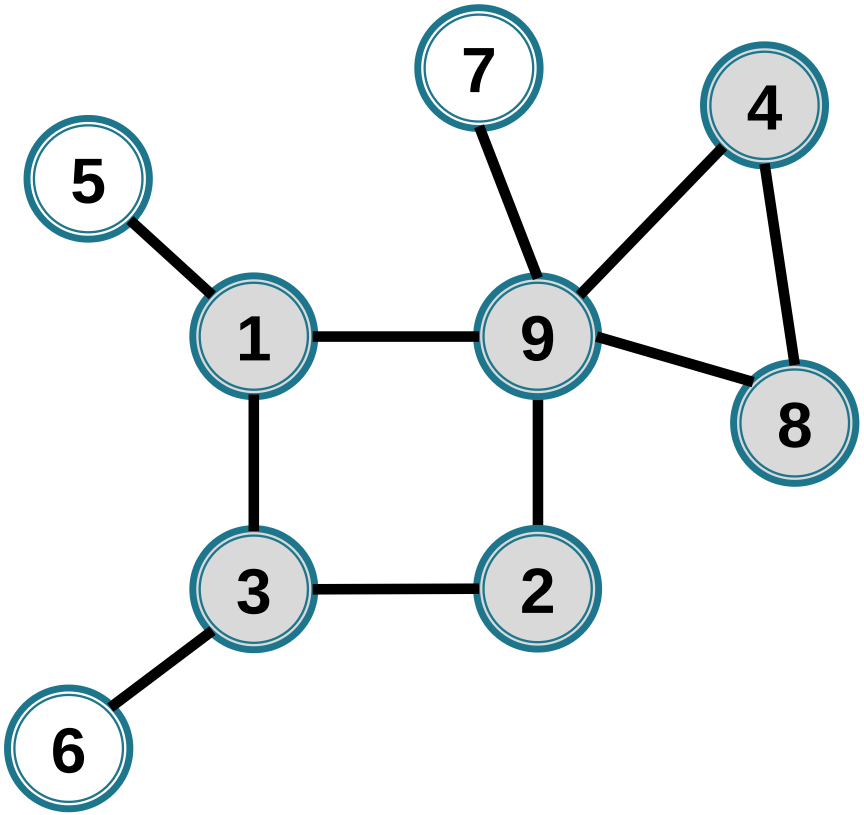

1
3
2

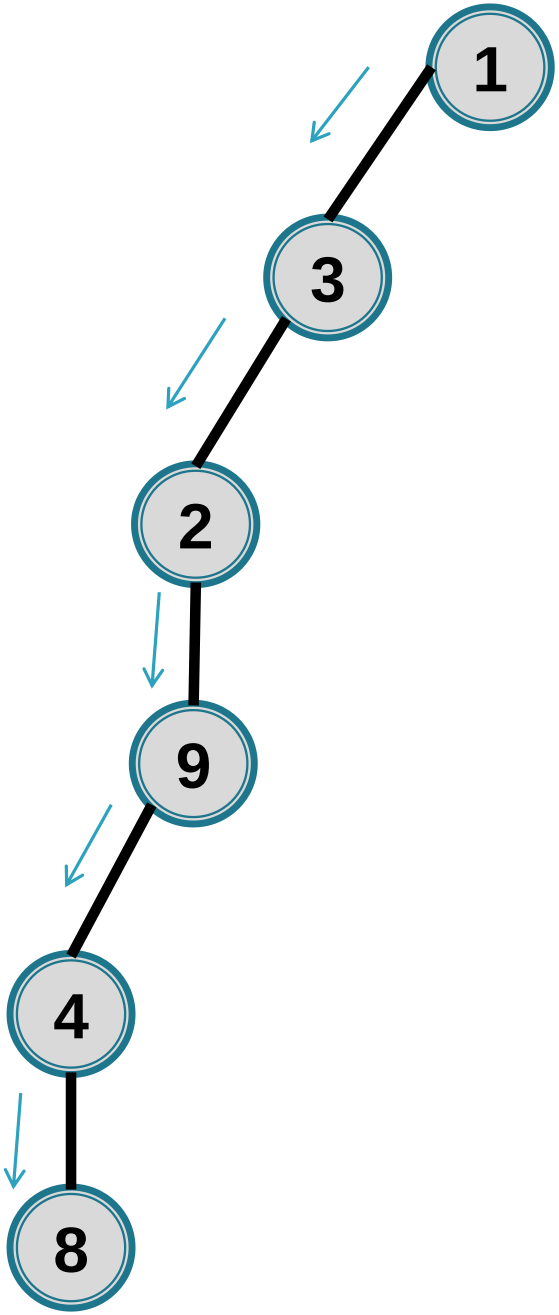

7
4
5
1
9
8
3
2
6

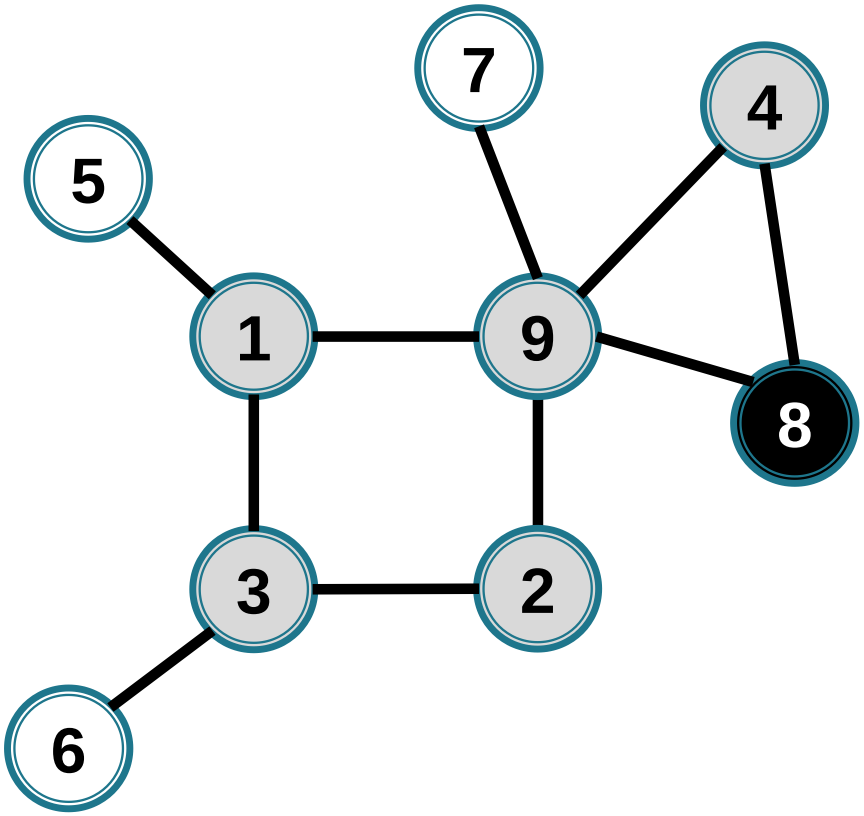

1
3
2
9

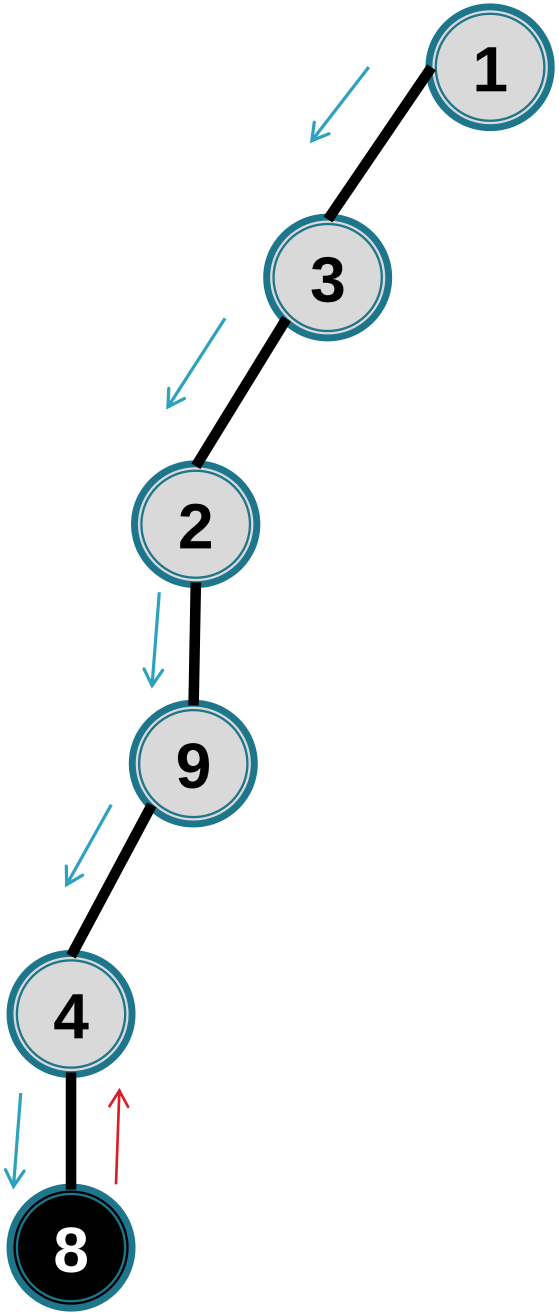

7
4
5
1
9
8
3
2
6

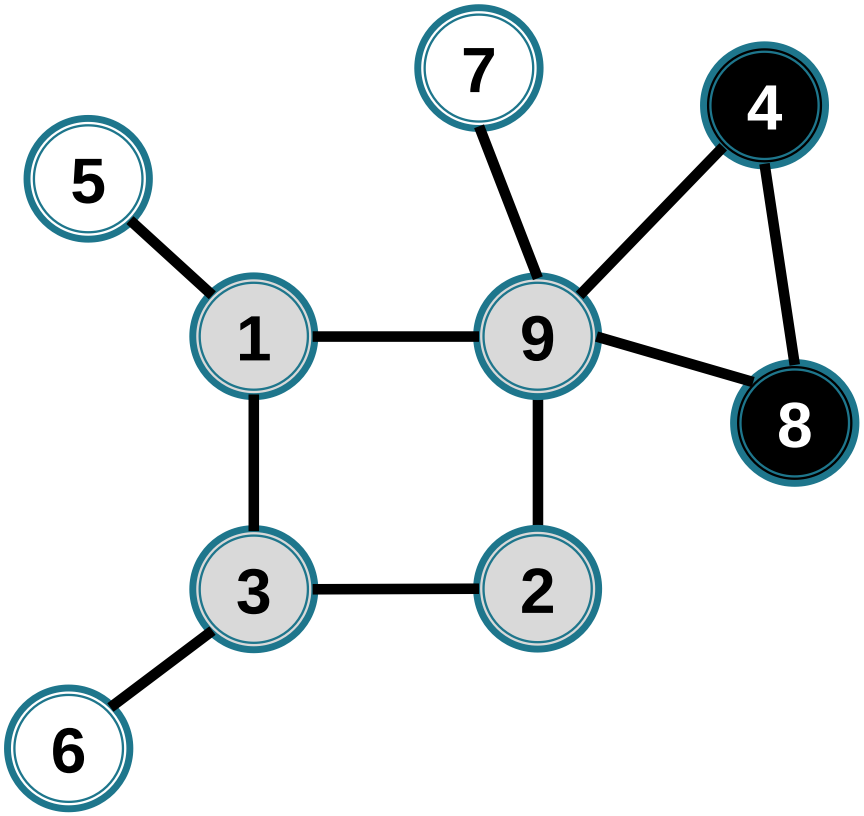

1
3
2
9
4

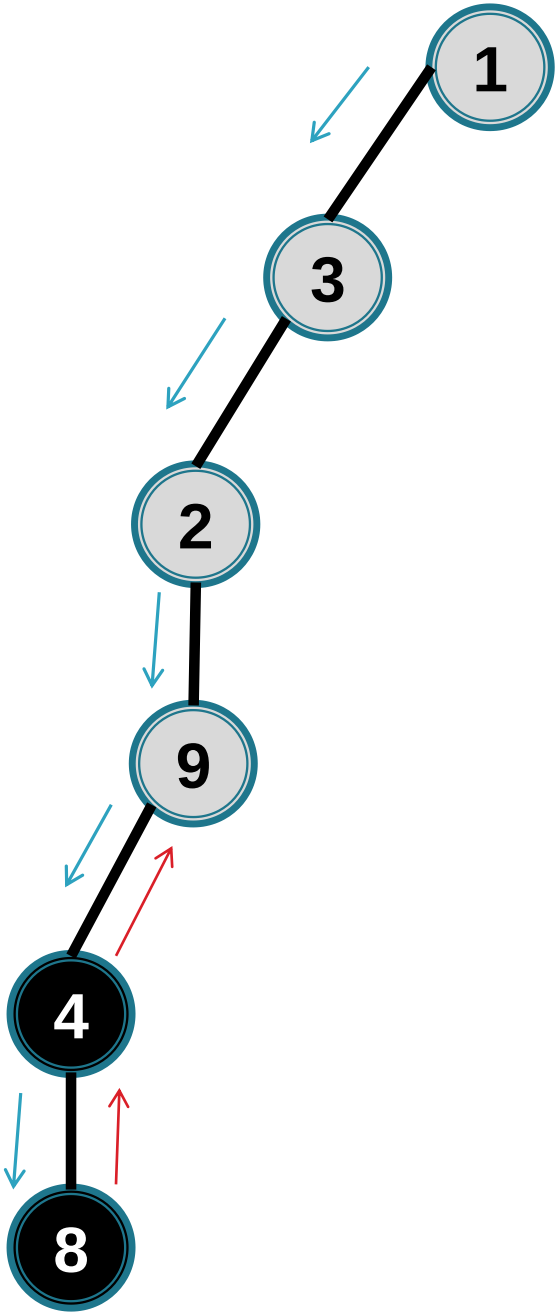

7
4
5
1
9
8
3
2
6

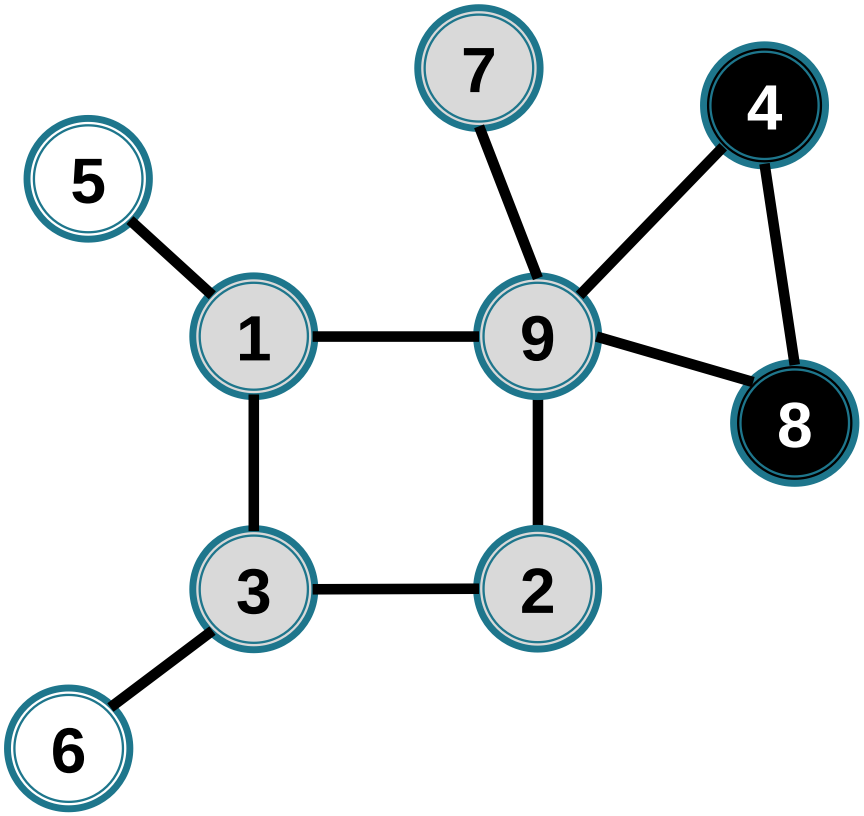

1
3
2
9
4
8

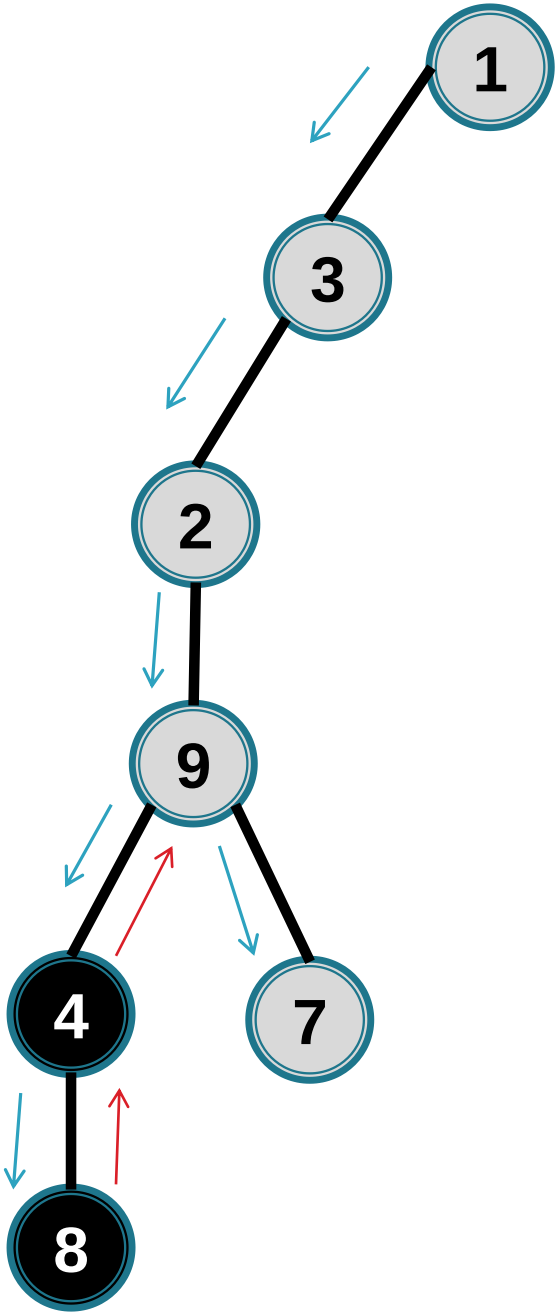

7
4
5
1
9
8
3
2
6

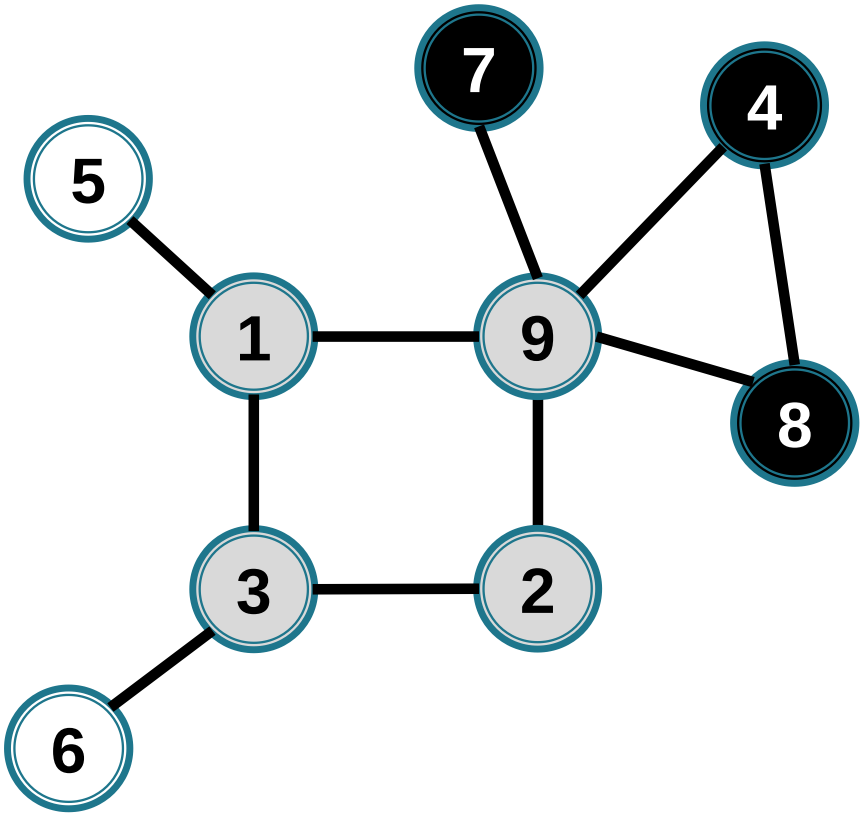

1
3
2
9
4
8

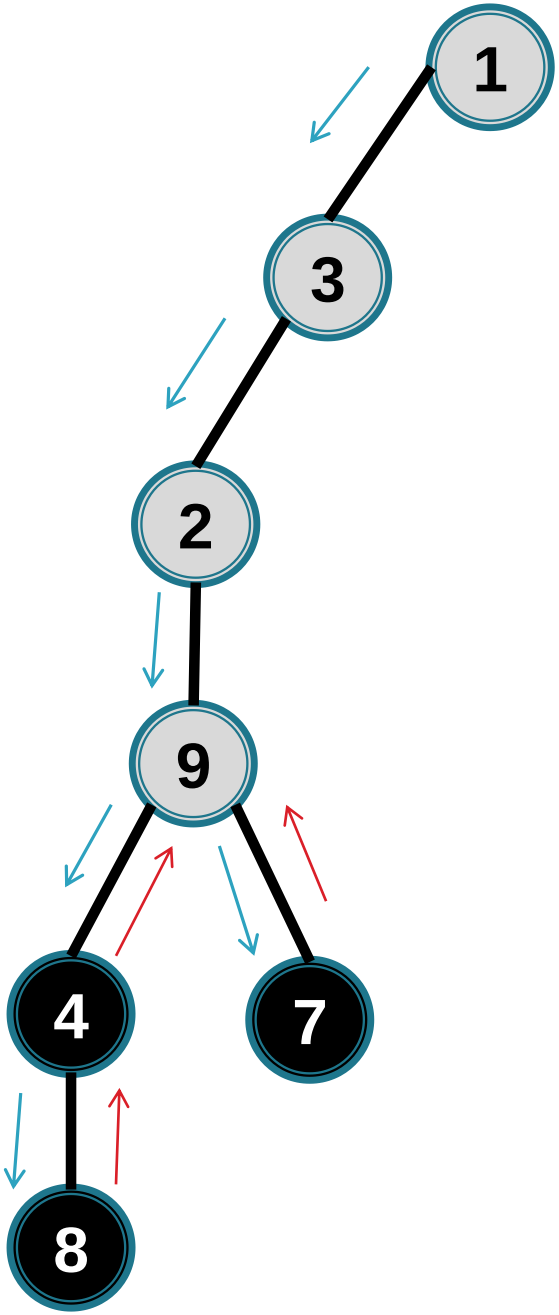

7
4
5
1
9
8
3
2
6

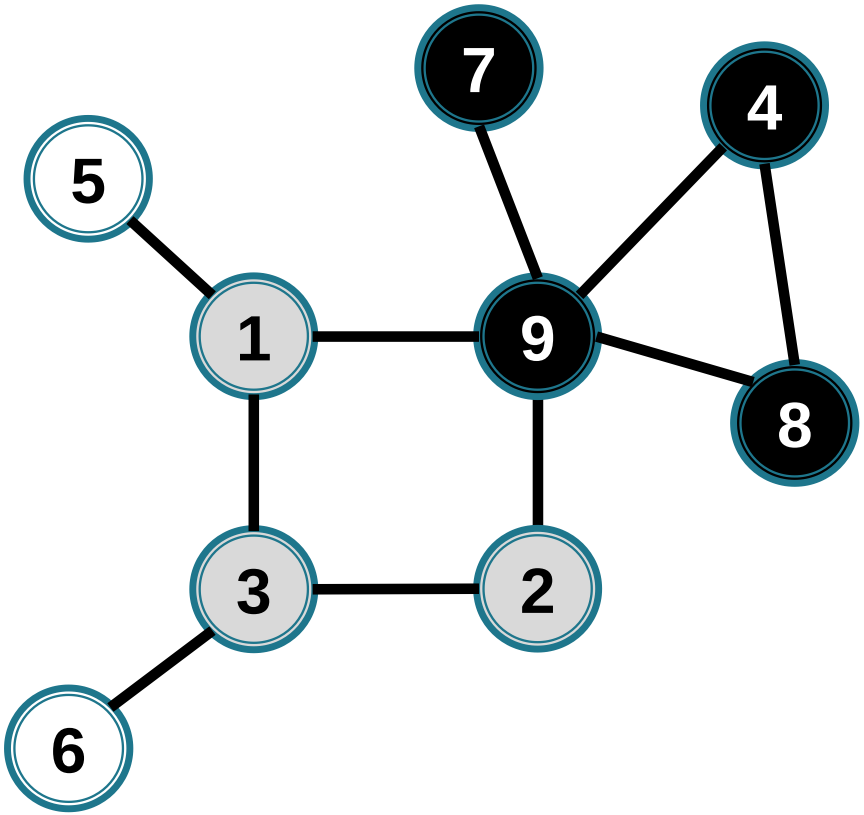

1
3
2
9
4
8

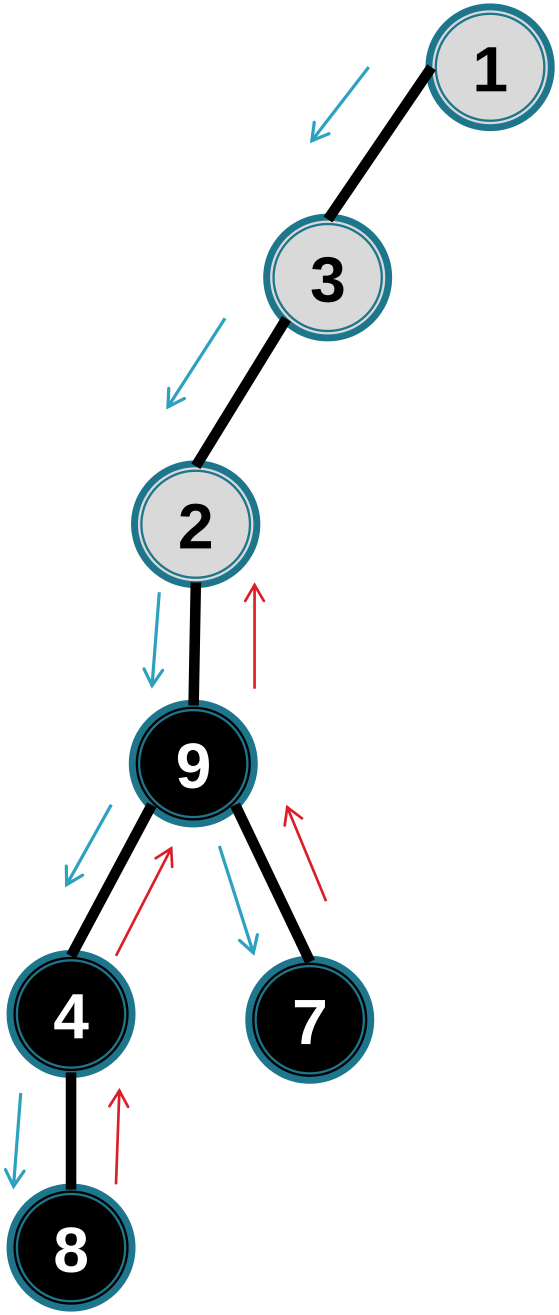

7
4
5
1
9
8
3
2
6

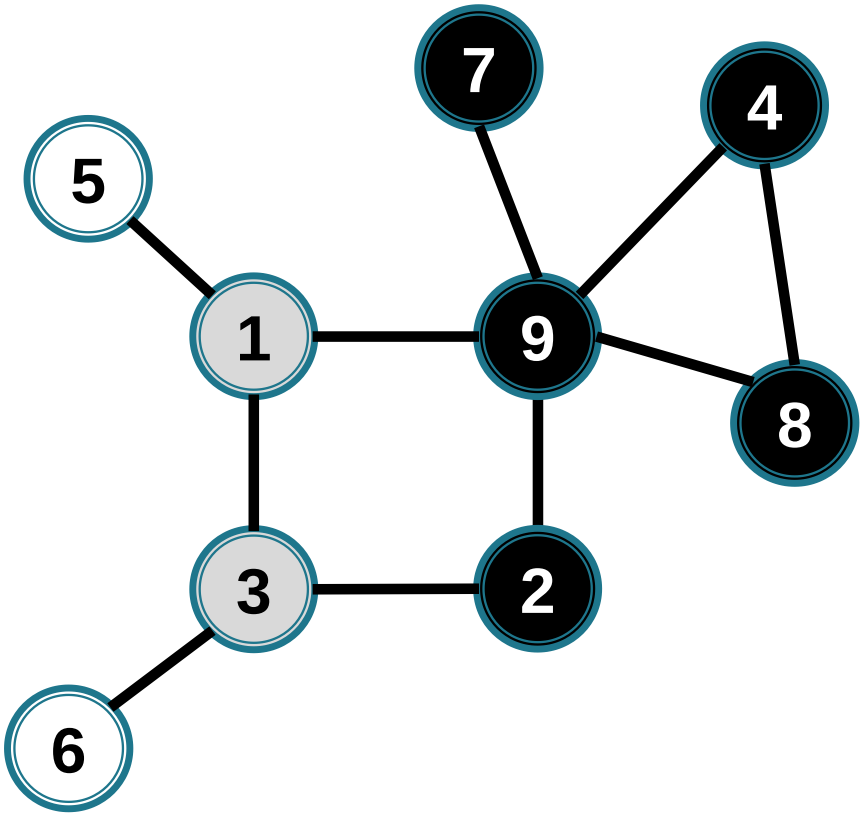

1
3
2
9
4
7
8

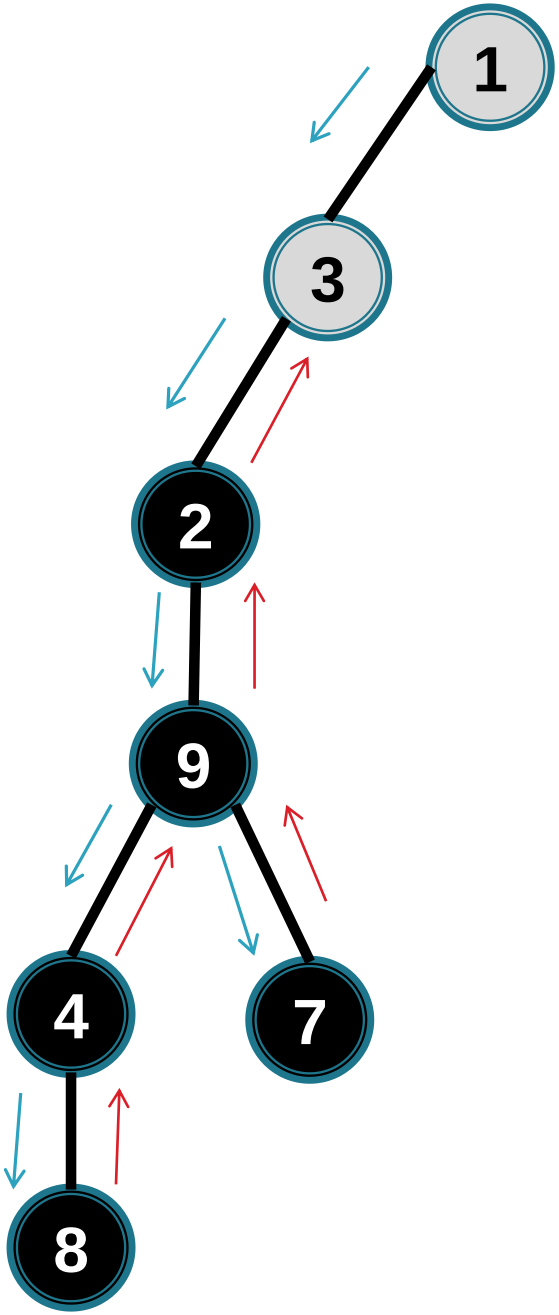

7
4
5
1
9
8
3
2
6

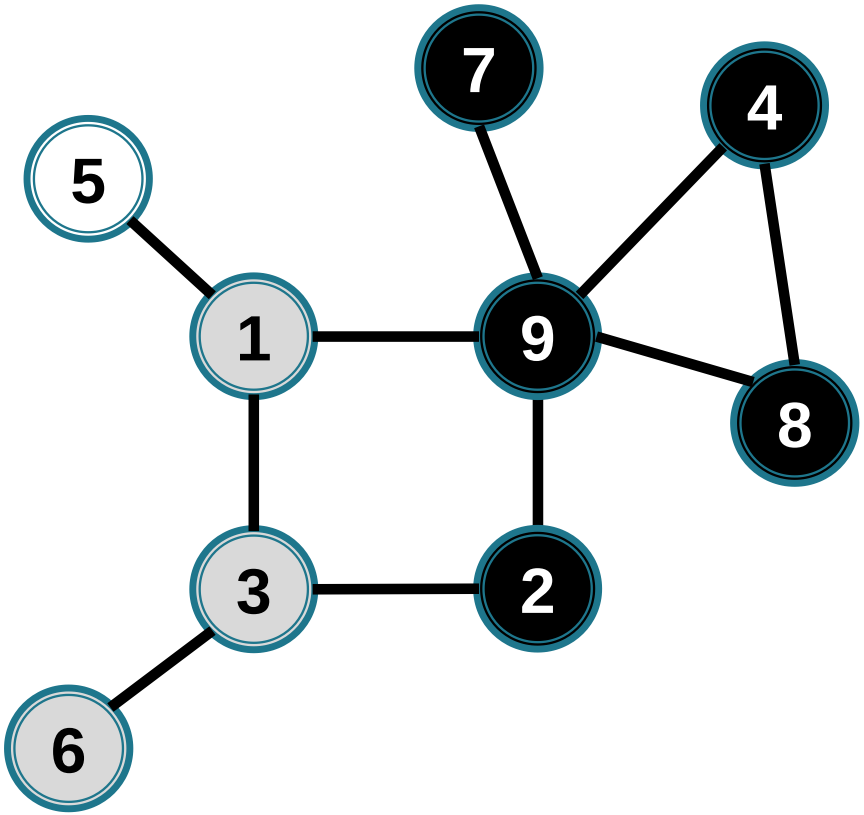

1
3
2
9
4
7
8

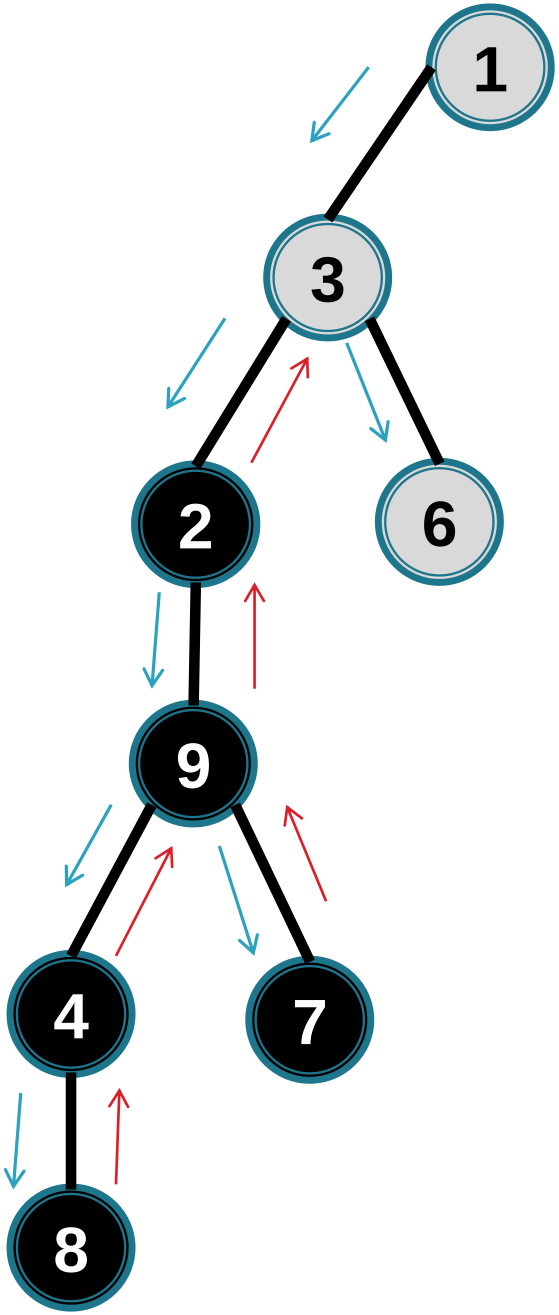

7
4
5
1
9
8
3
2
6

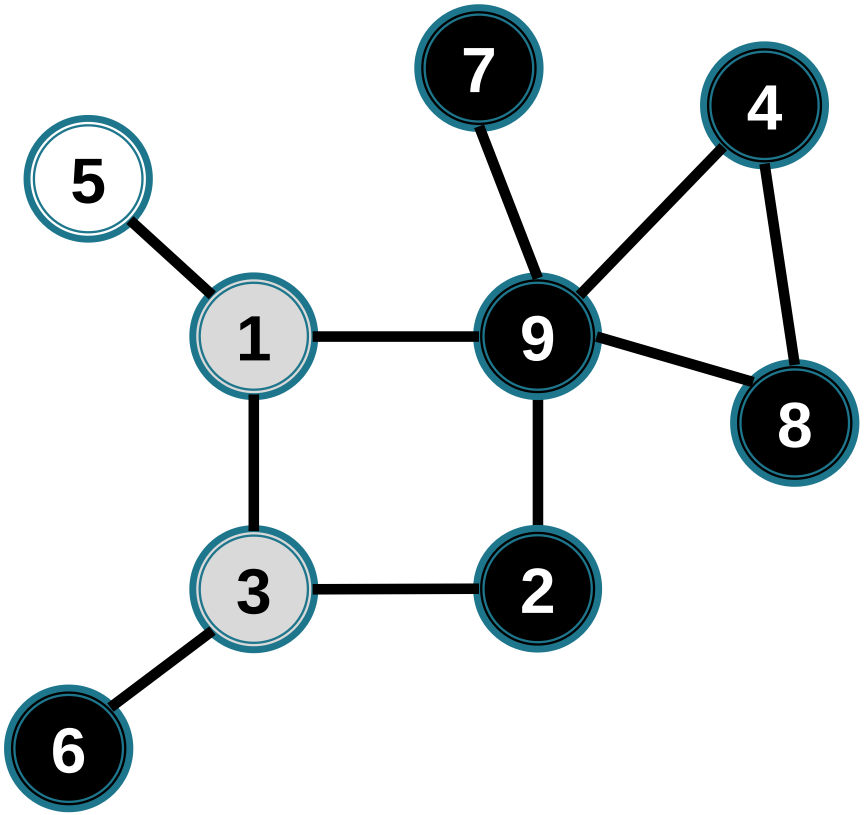

1
3
2
9
4
7
8

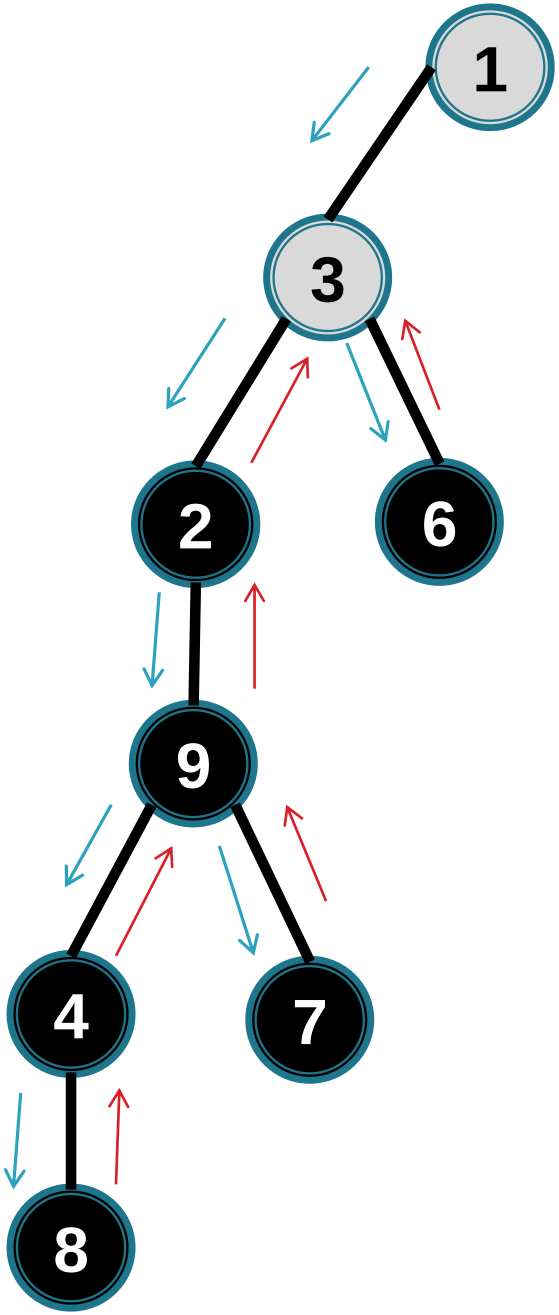

7
4
5
1
9
8
3
2
6

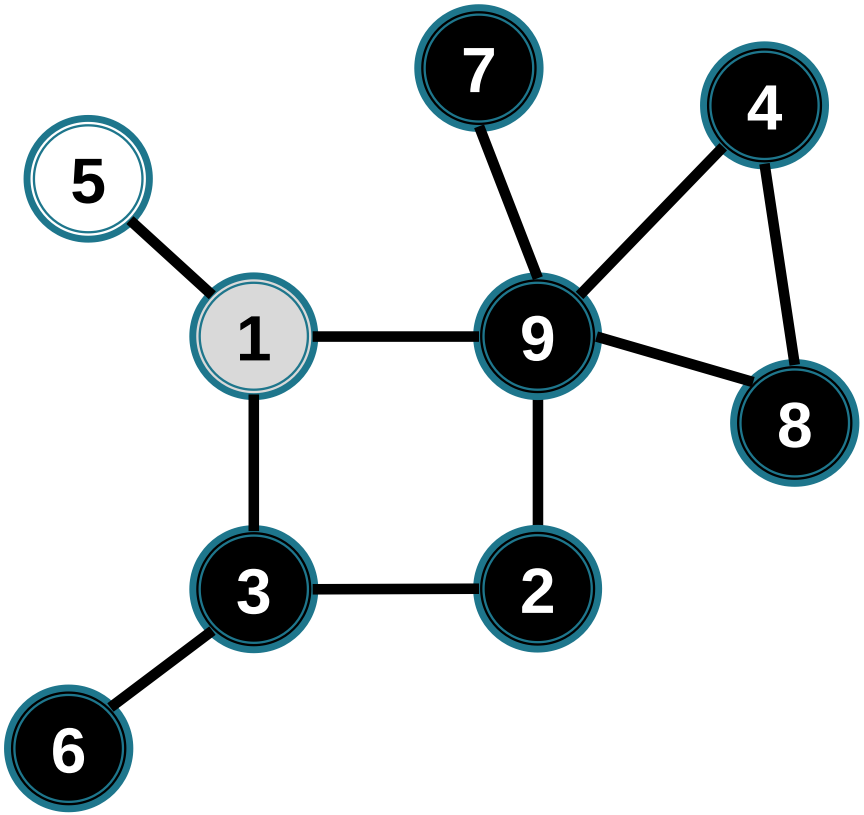

1
3
2
9
4
7
8

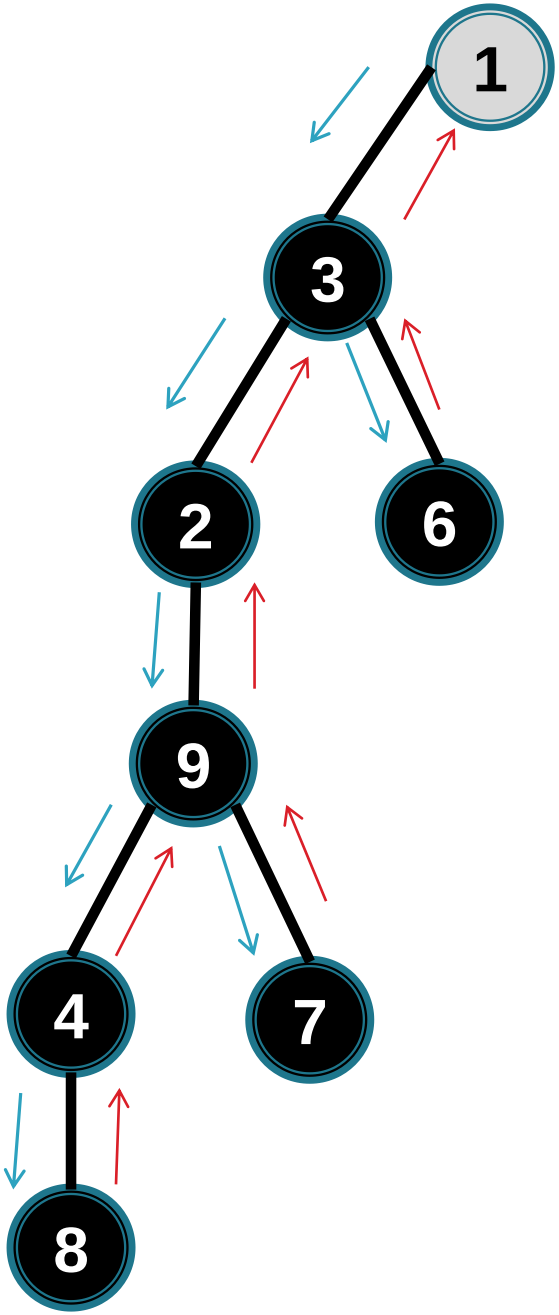

7
4
5
1
9
8
3
2
6

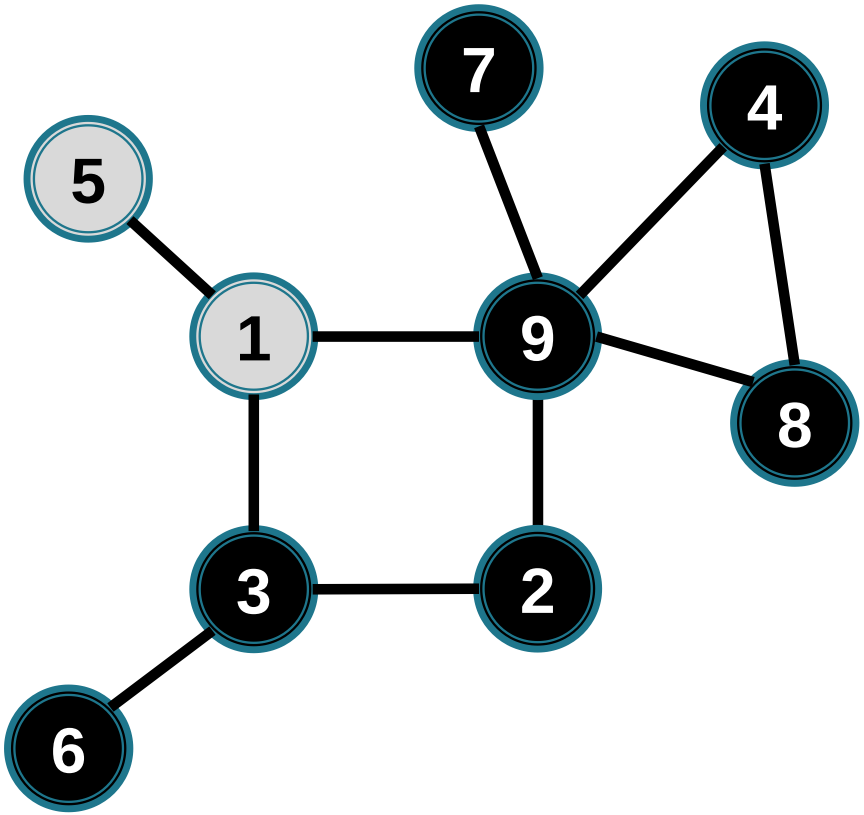

1
3
2
6
9
4
7
8

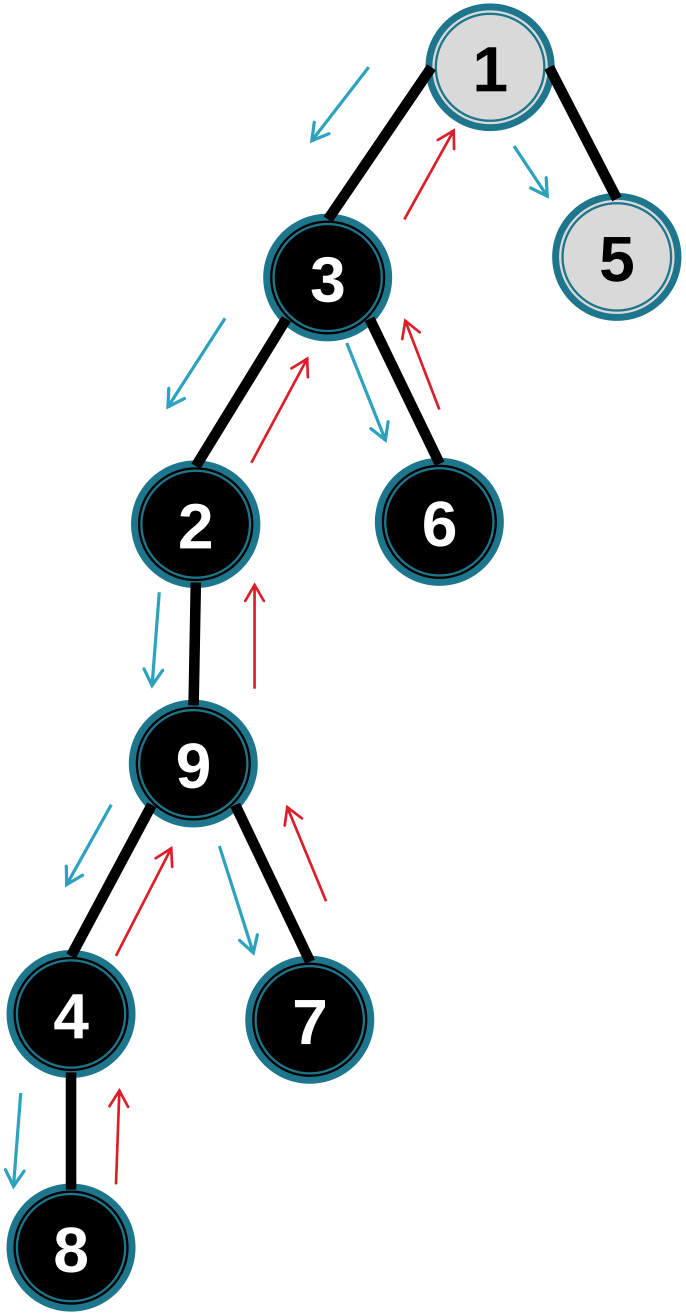

7
4
5
1
9
8
3
2
6

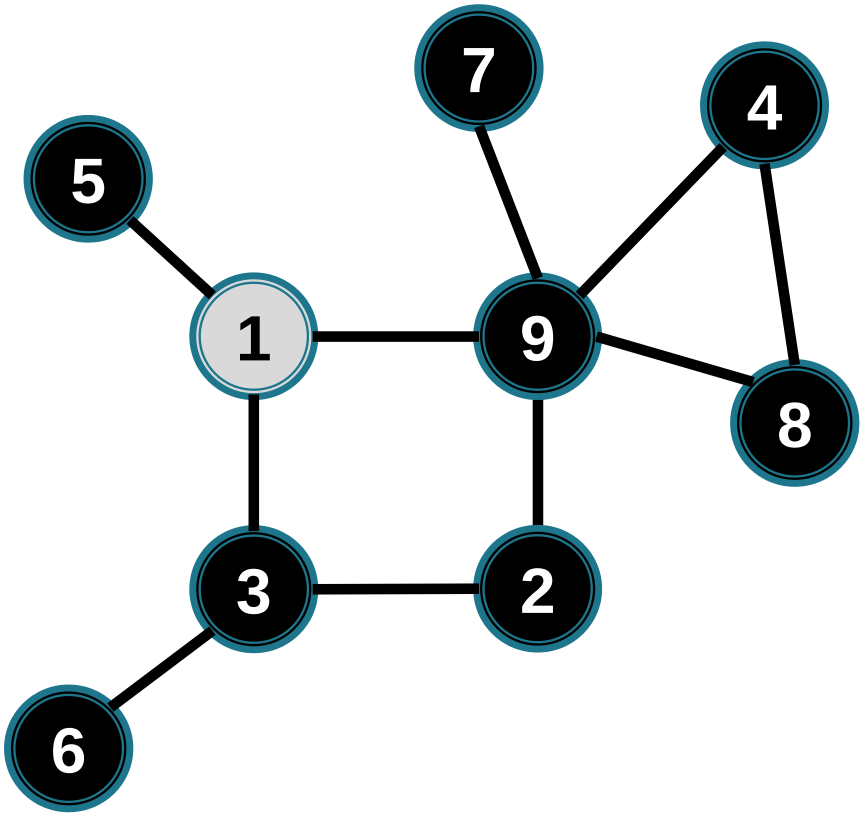

1
3
2
6
9
4
7
8

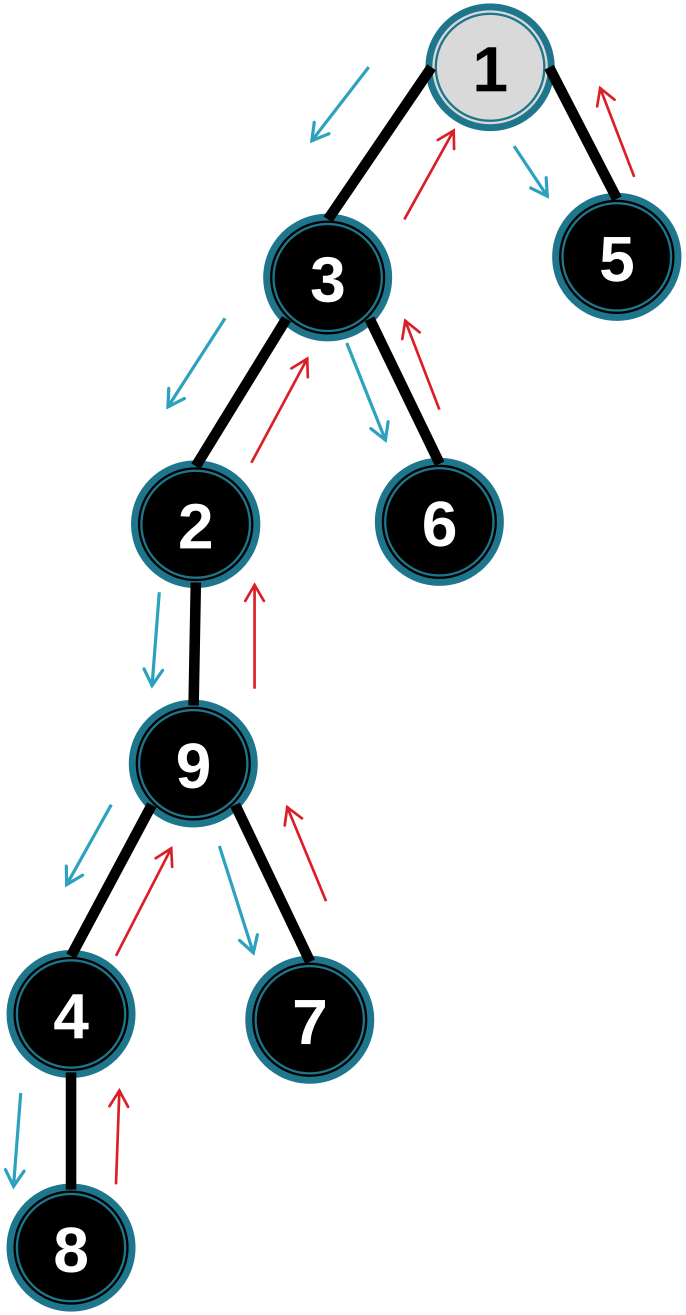

7
4
5
1
9
8
3
2
6

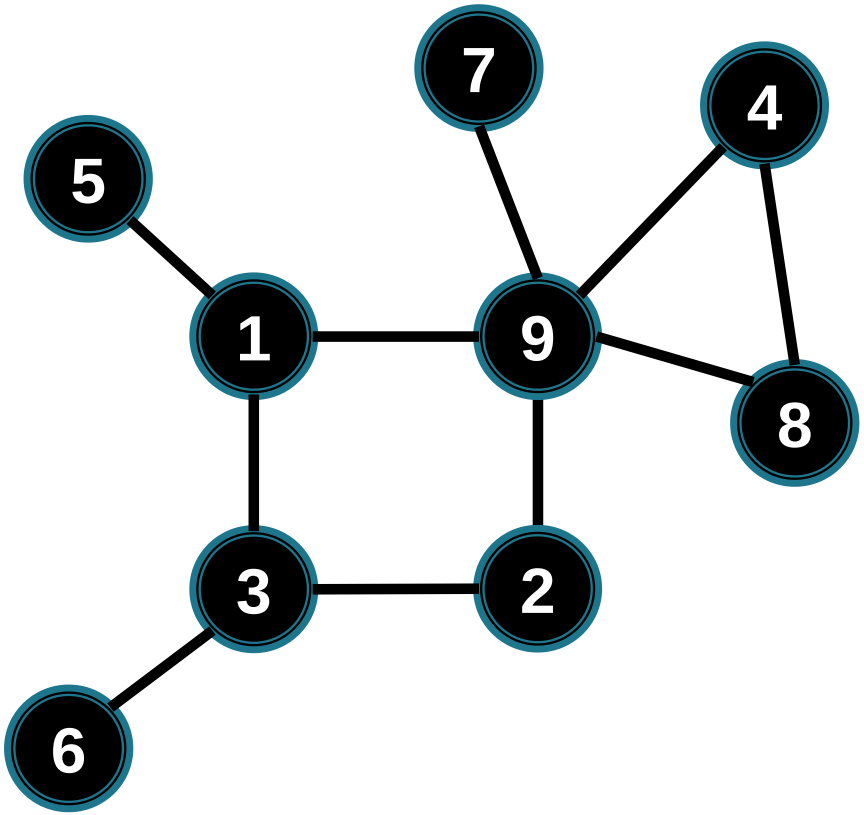

1
3
2
6
9
4
7
8

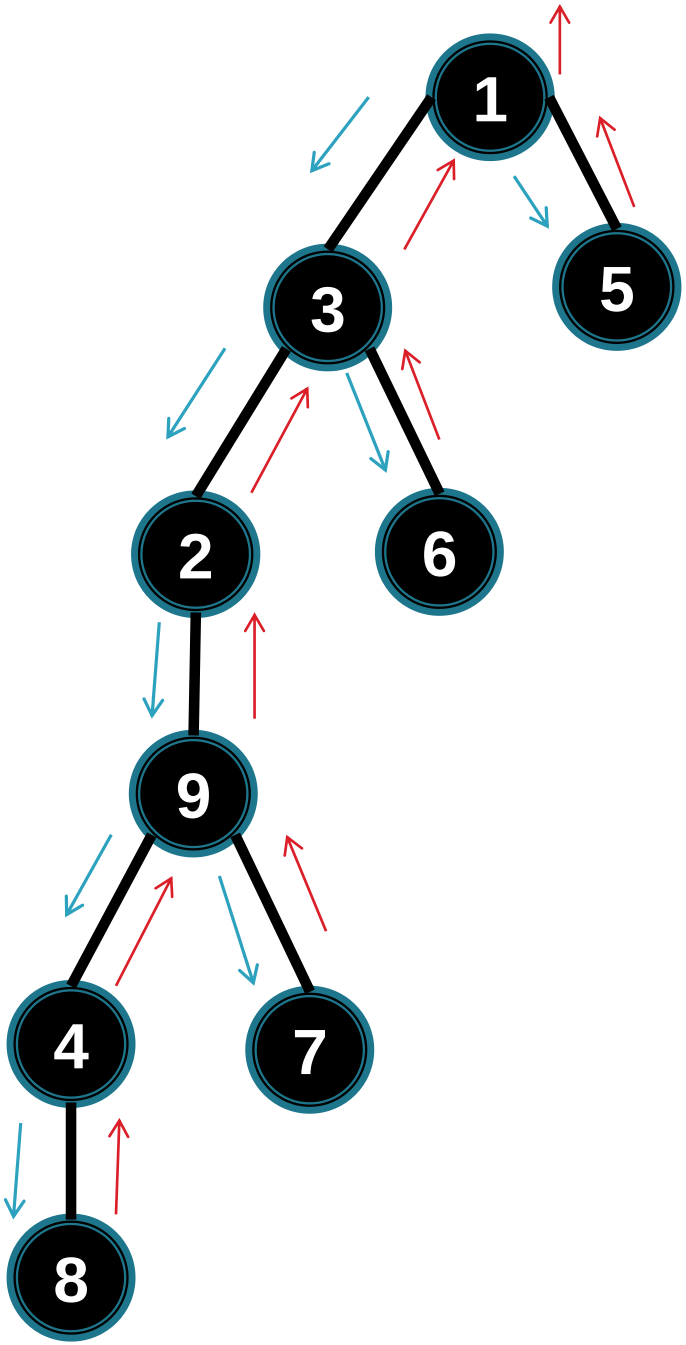

7
4
5
1
9
8
3
2
6

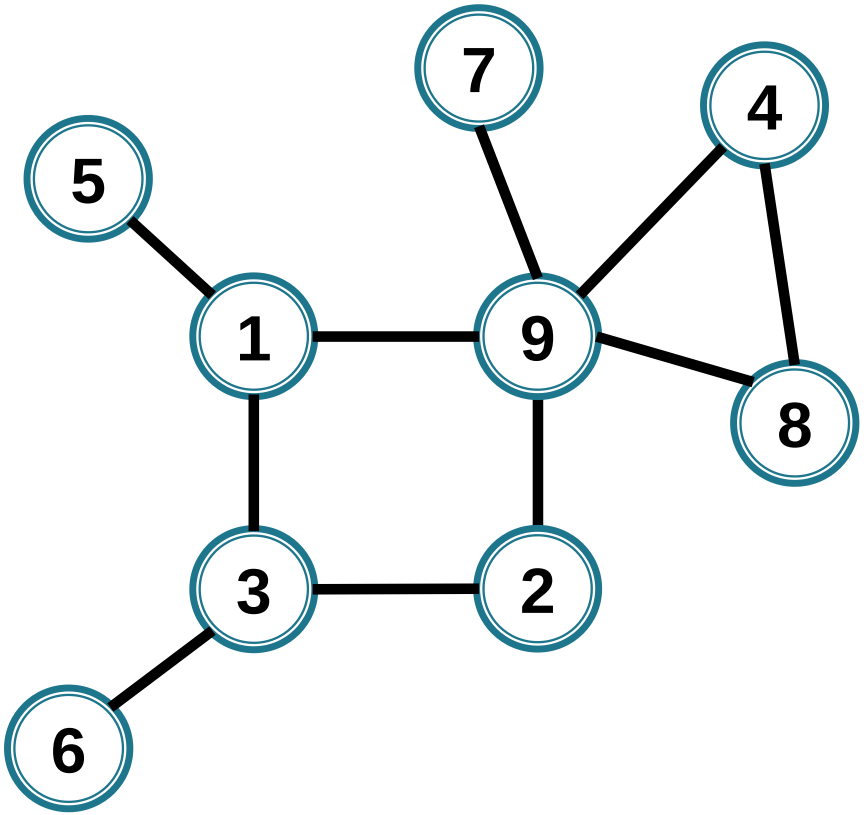

1
5
3
2
6
9
4
7
8

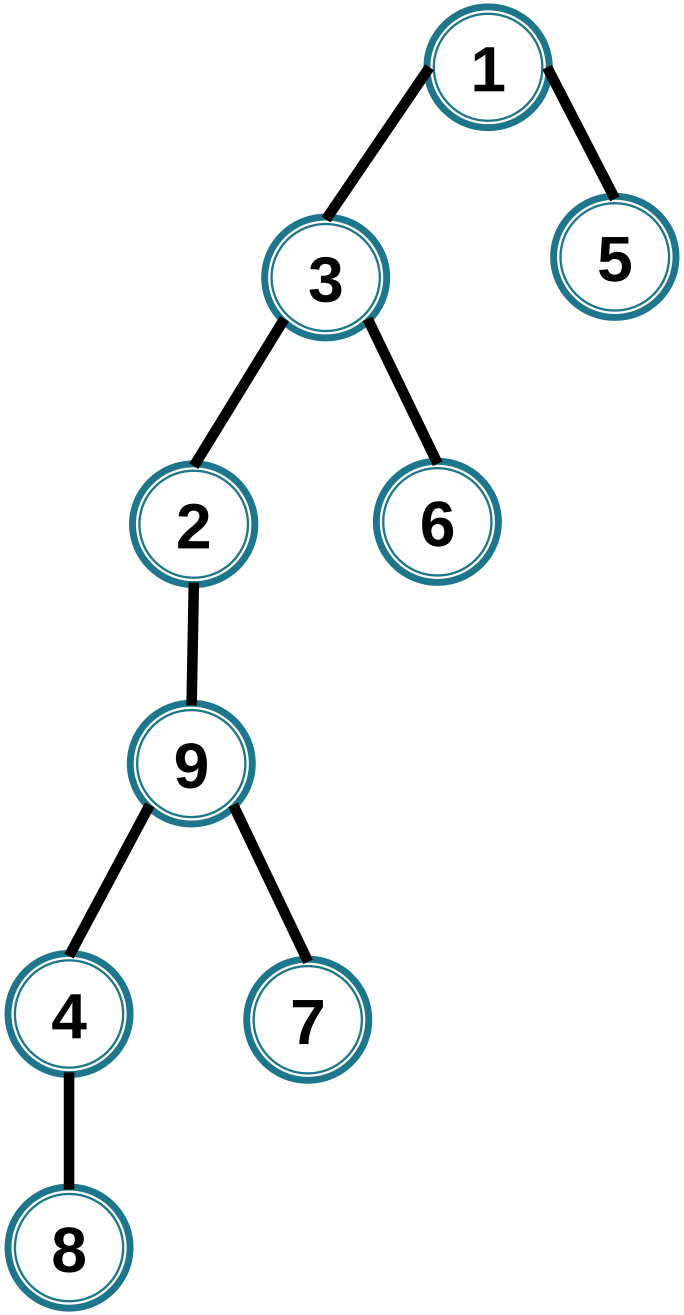

7
4
5
1
9
8
3
2
6


1
3
5
2
6
9
4
7
8


7
4
5
1
9
8
3
2
6


1
3
5
2
6
9
4
7
8


7
4
5
1
9
8
3
2
6


1
3
5
2
6
9
4
7
8

```
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
```

x alb

x gri

x negru

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

`viz[x] = 0 ⇔ culoare[x] = 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

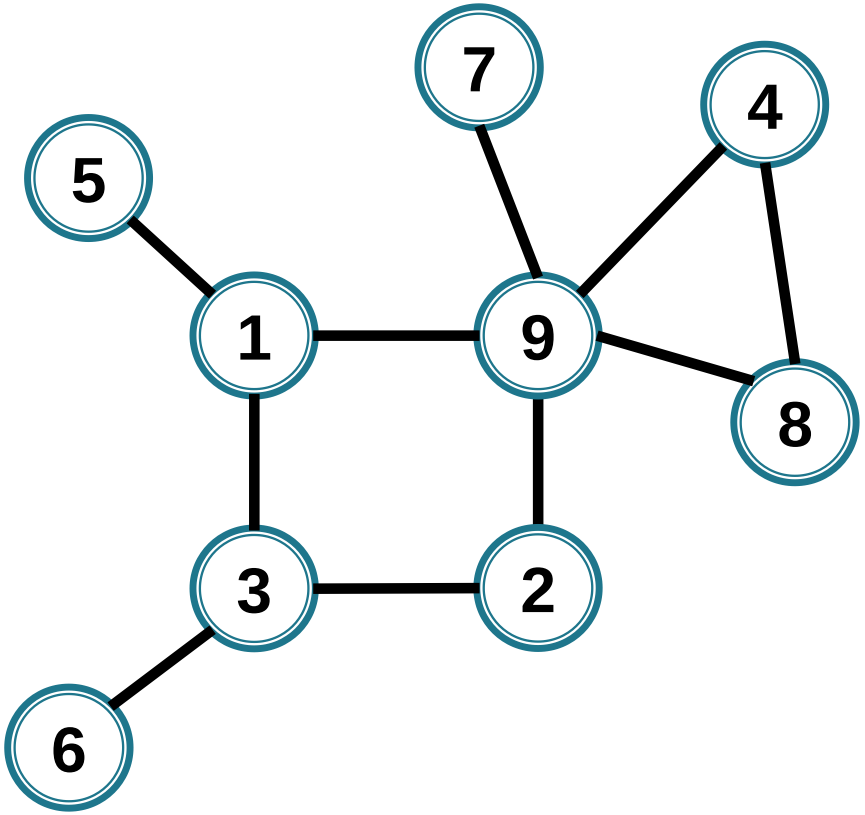
Dacă dorim determinarea unor elemente de structură în graf precum circuite, componente tare conexe etc poate fi util să memorăm în parcurgerea DFS culoarea fiecărui vârf și/sau momentul în care a fost descoperit sau finalizat.

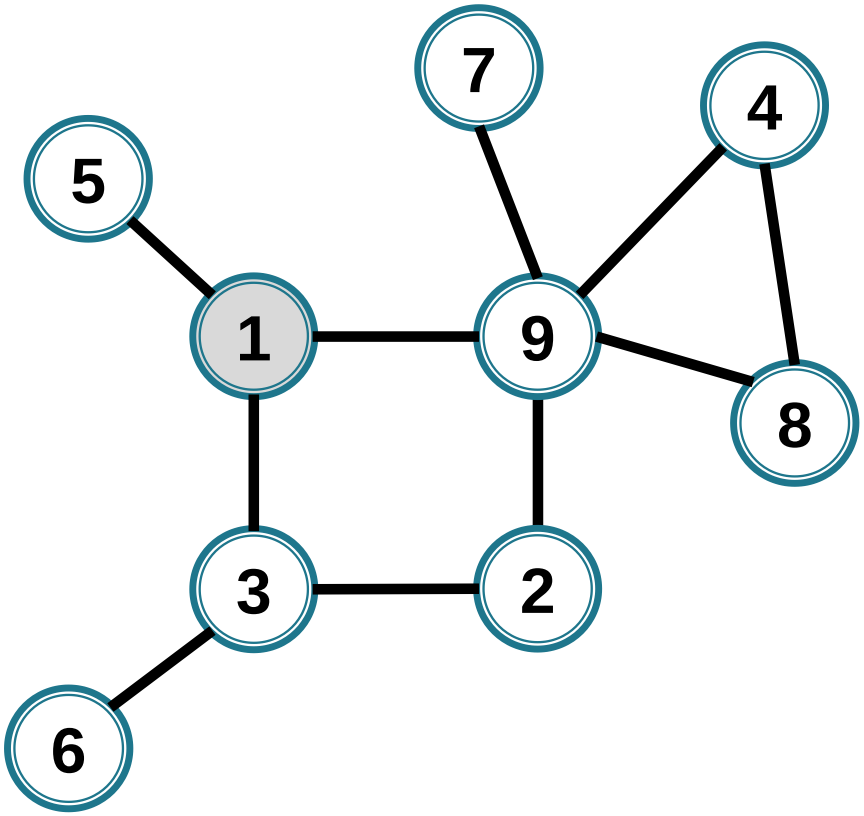
Un pseudocod complet în acest caz este următorul:

```
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
```

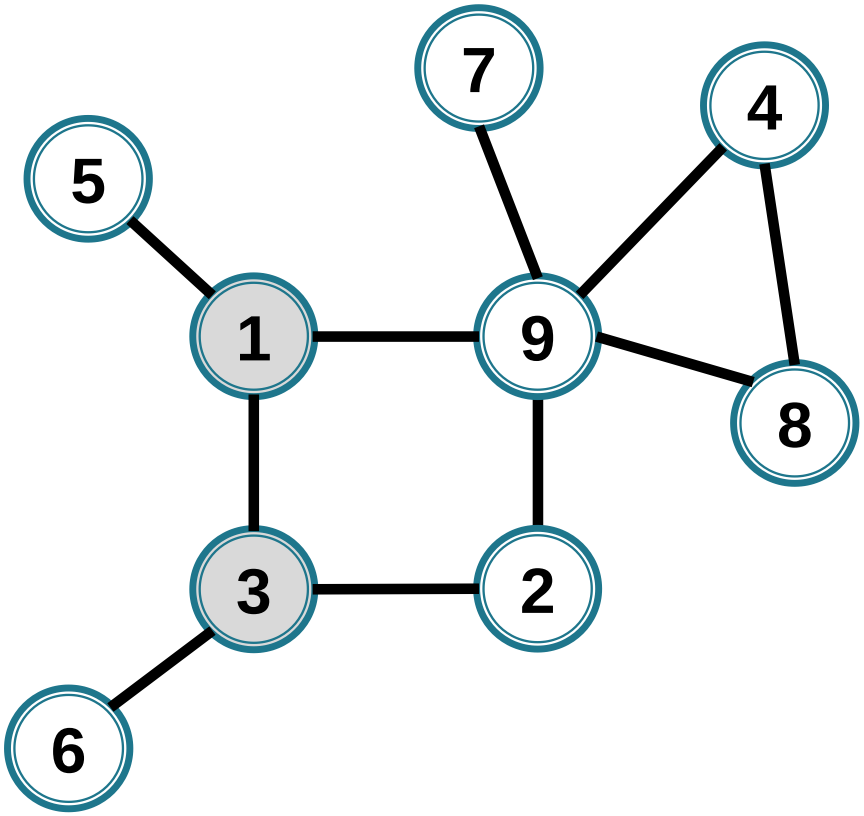
Apel:

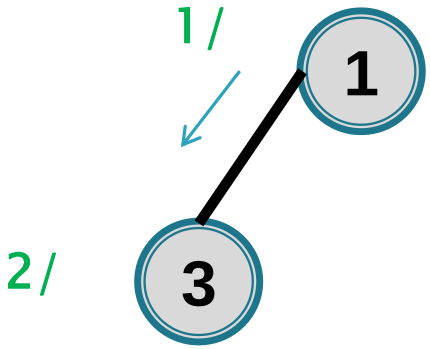
```
pentru fiecare x in V executa
      culoare[x] = alb
timp = 0
pentru fiecare x in V executa
        daca culoare[x] == alb atunci
           DFS(x)
```

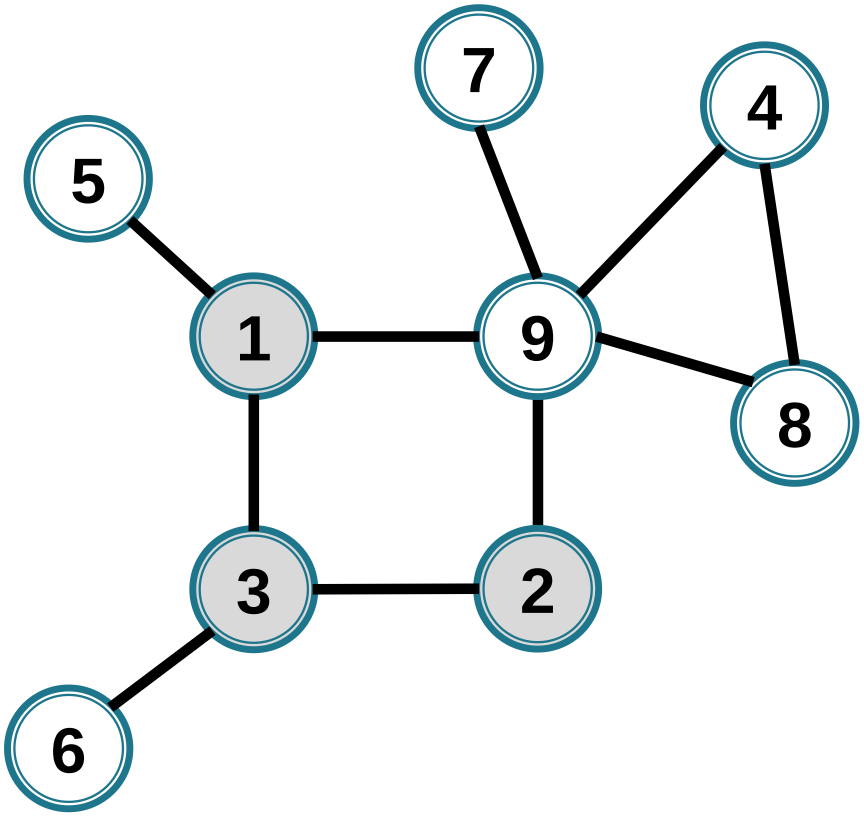

7
4
5
1
9
8
3
2
6

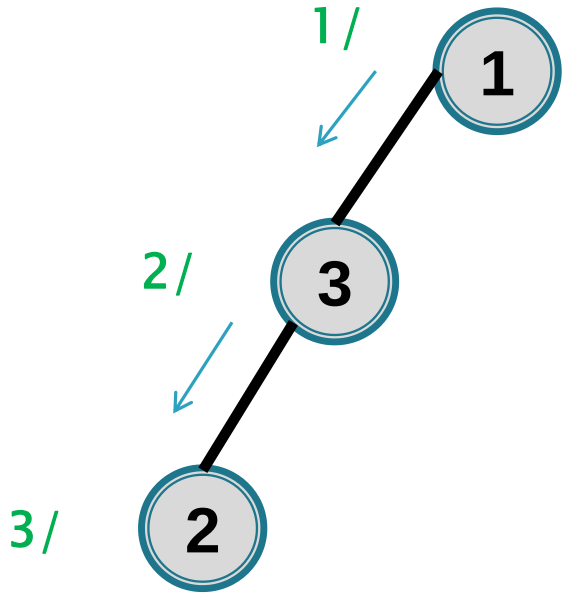


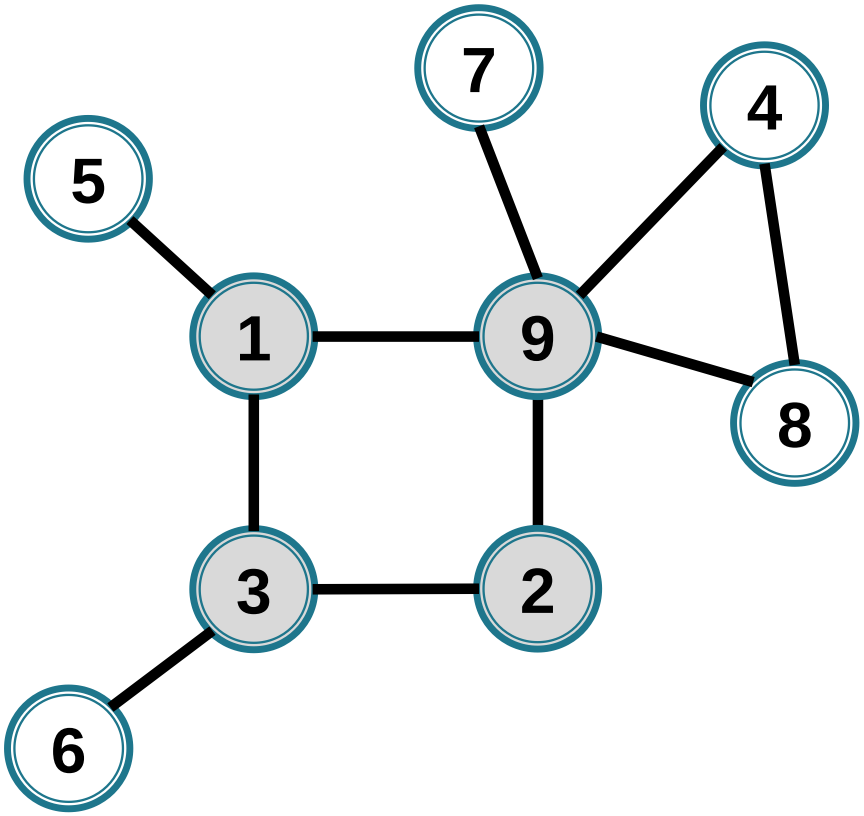
1 /

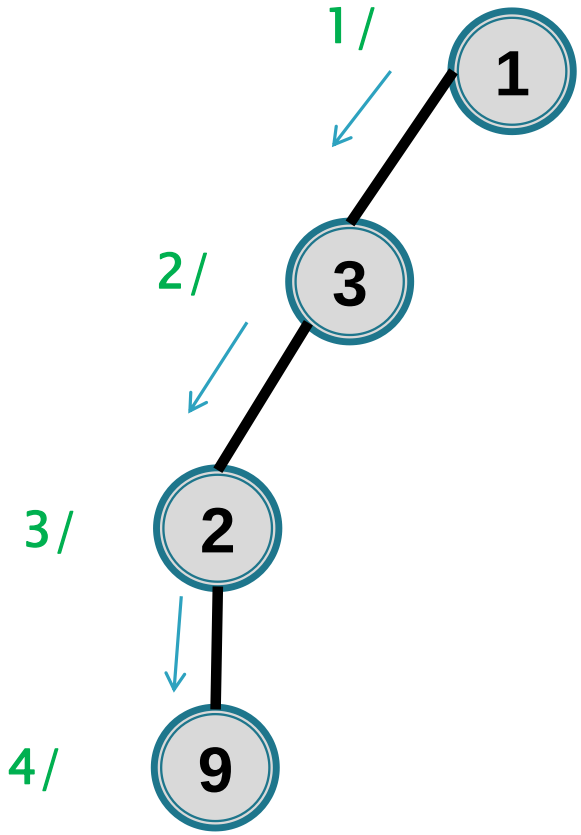

7
4
5
1
9
8
3
2
6

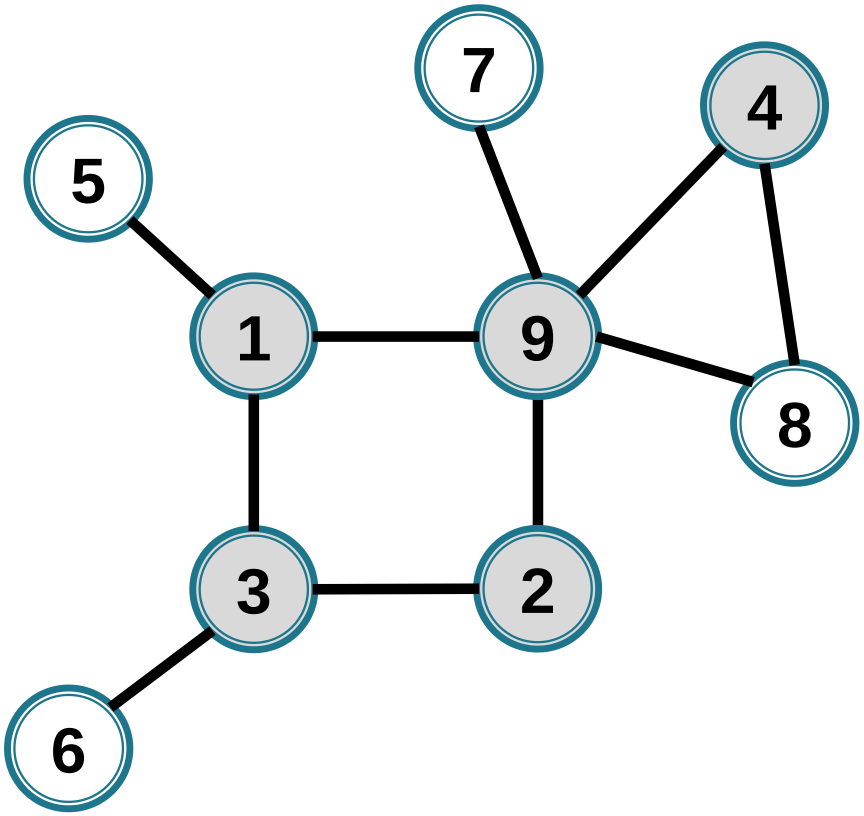

1 /
1
2 /
3

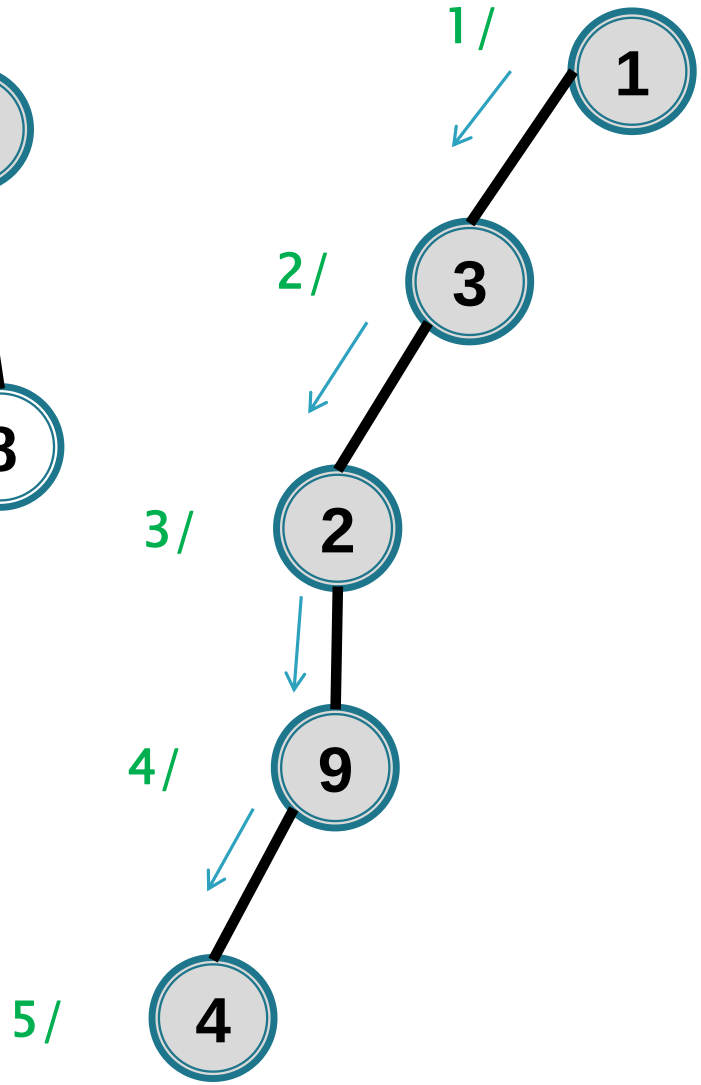

7
4
5
1
9
8
3
2
6

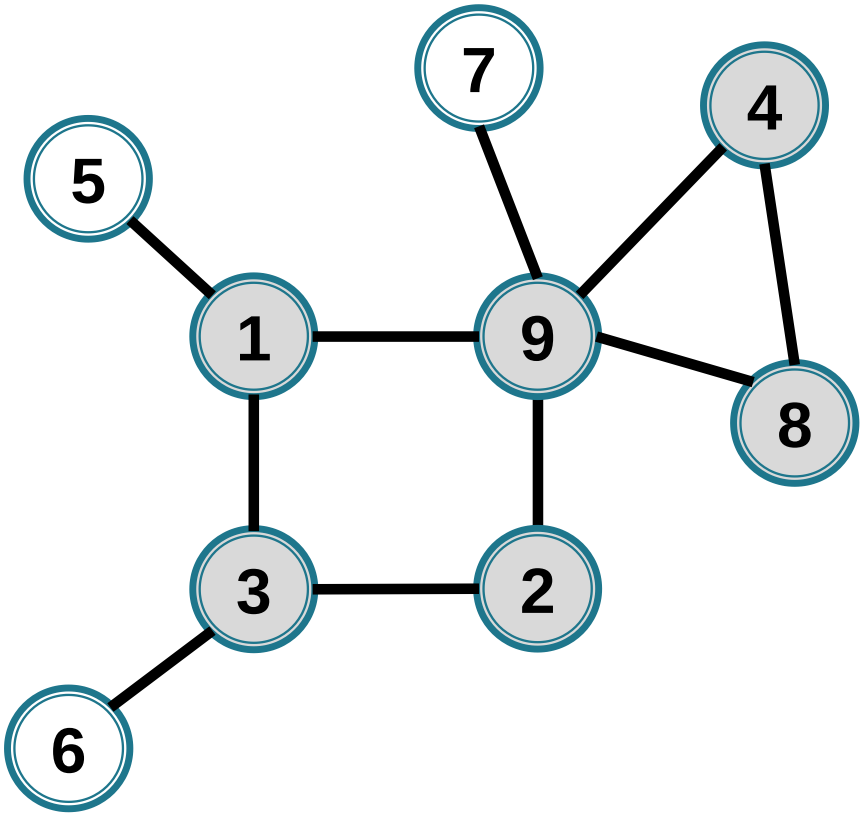

1 /
1
2 /
3
3 /
2

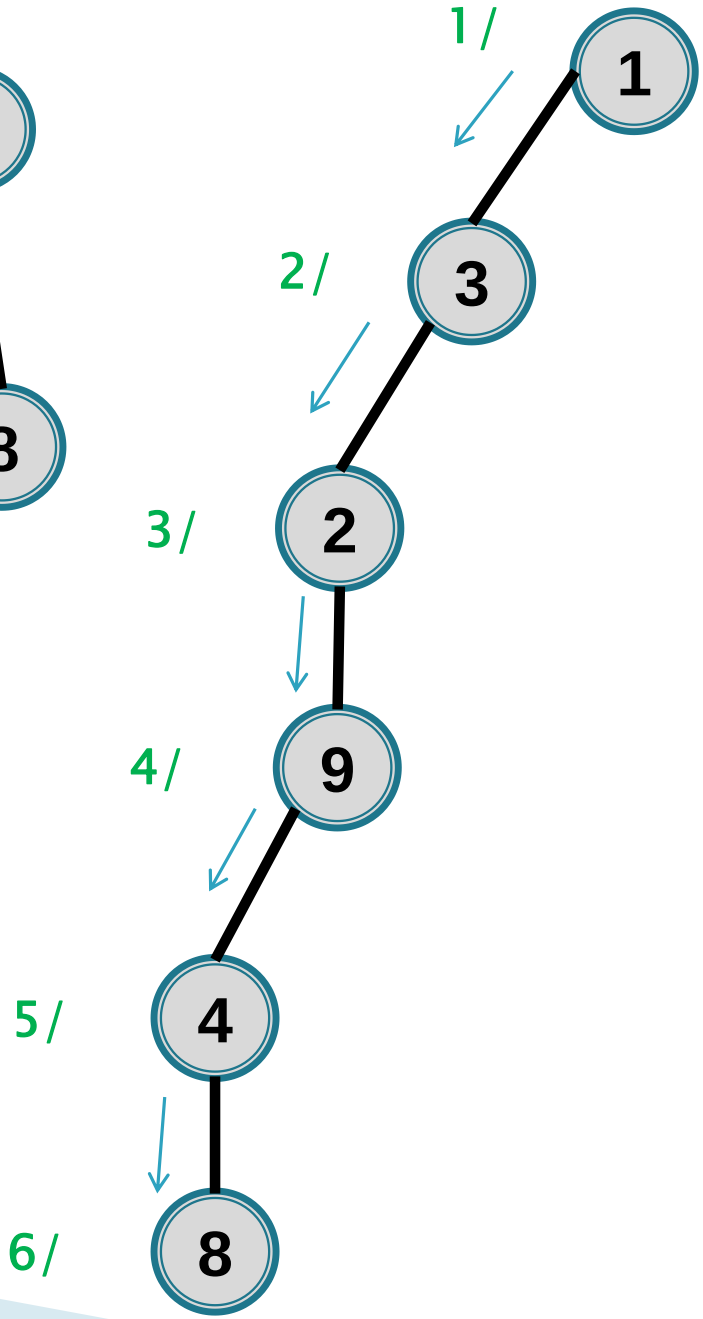

7
4
5
1
9
8
3
2
6

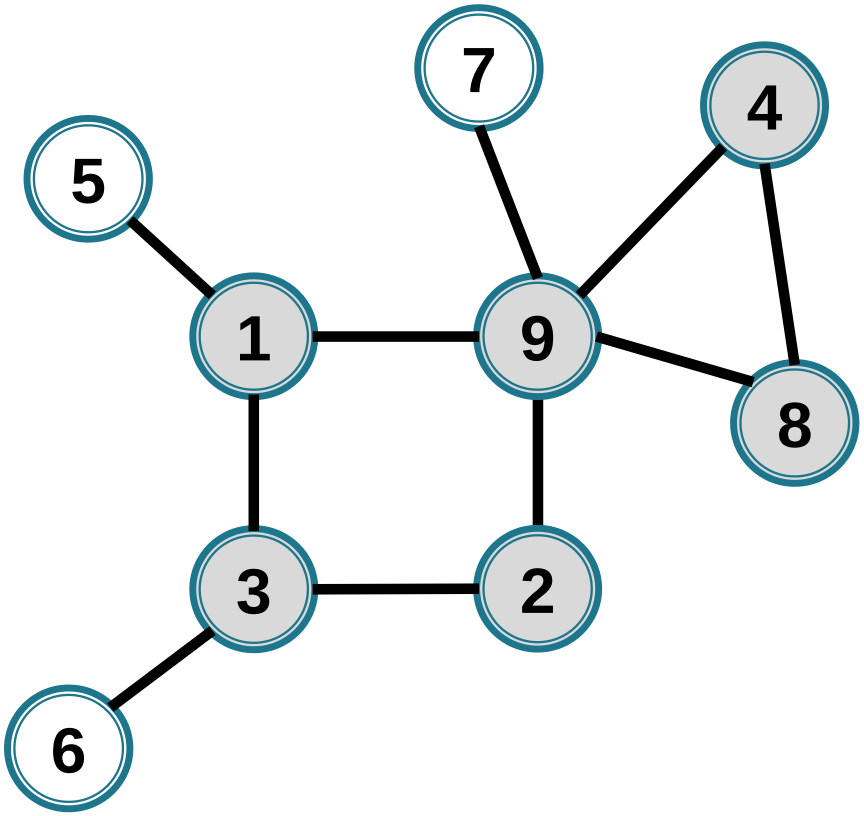

1 /
1
2 /
3
3 /
2
4 /
9

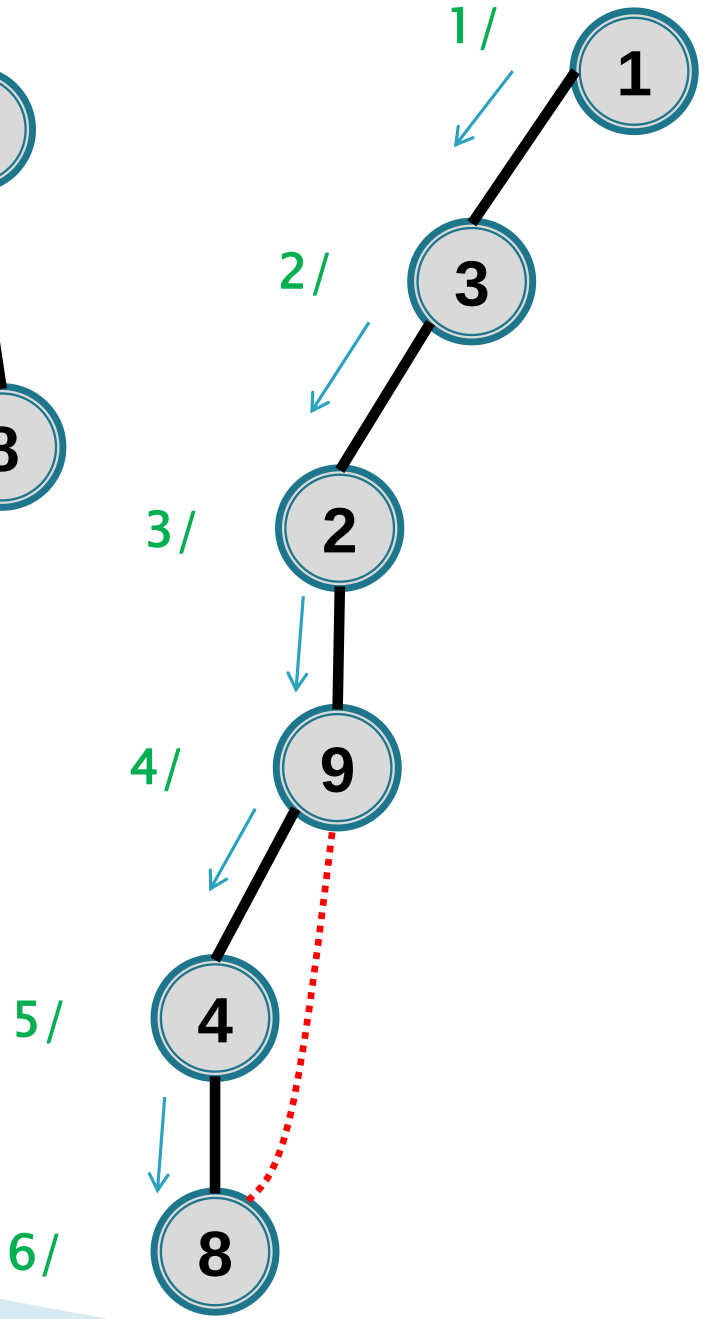

7
4
5
1
9
8
3
2
6

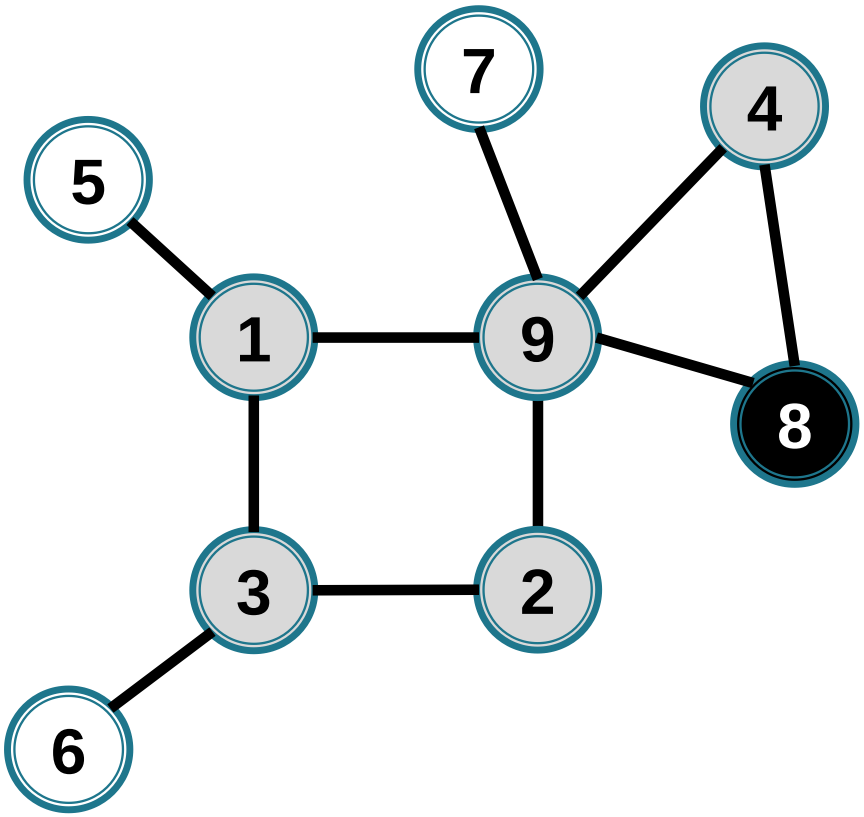

1 /
1
2 /
3
3 /
2
4 /
9
5 /
4

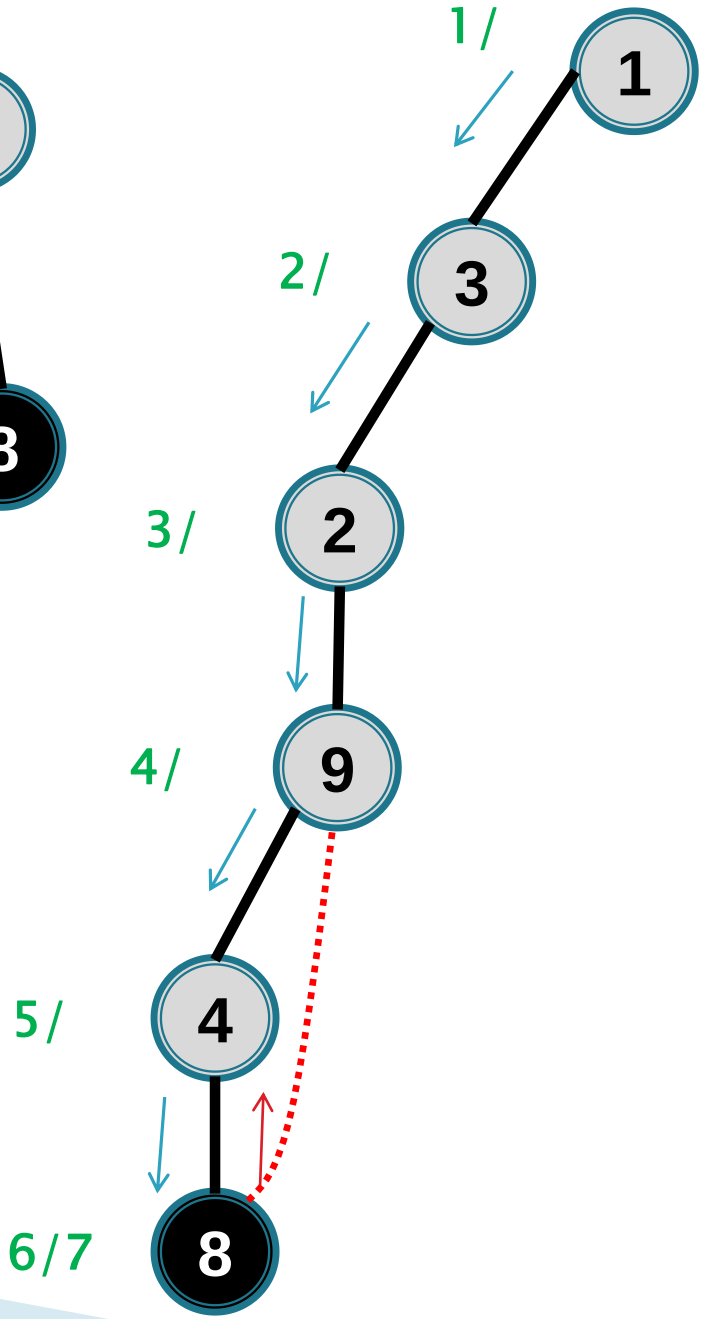

7
4
5
1
9
8
3
2
6

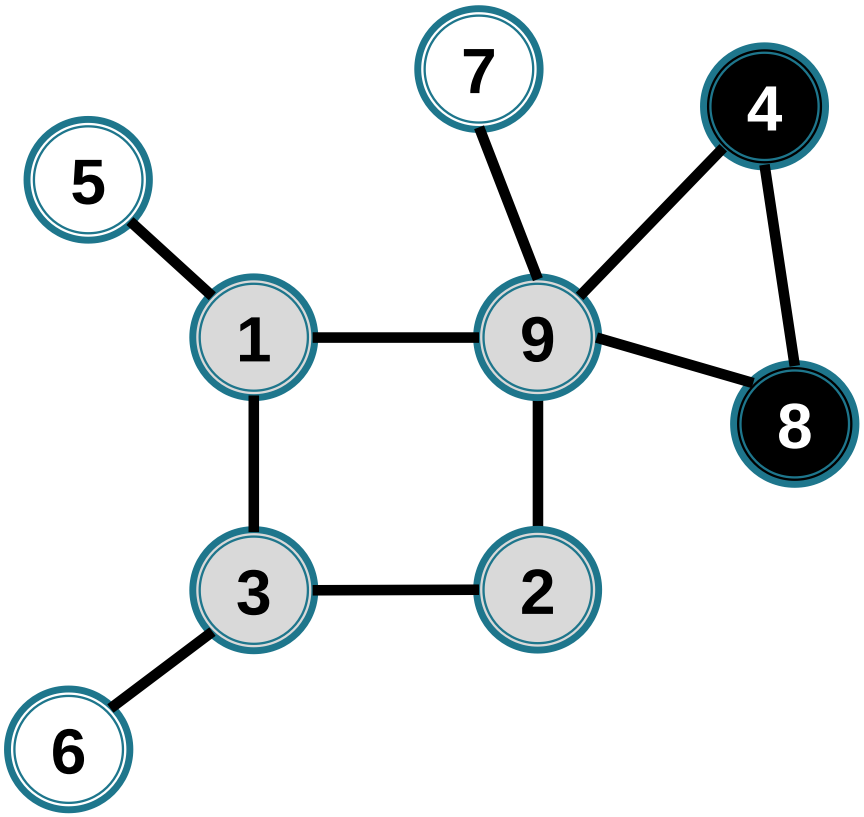

1/
1
2/
3
3/
2
4/
9
5/
4
6/
8

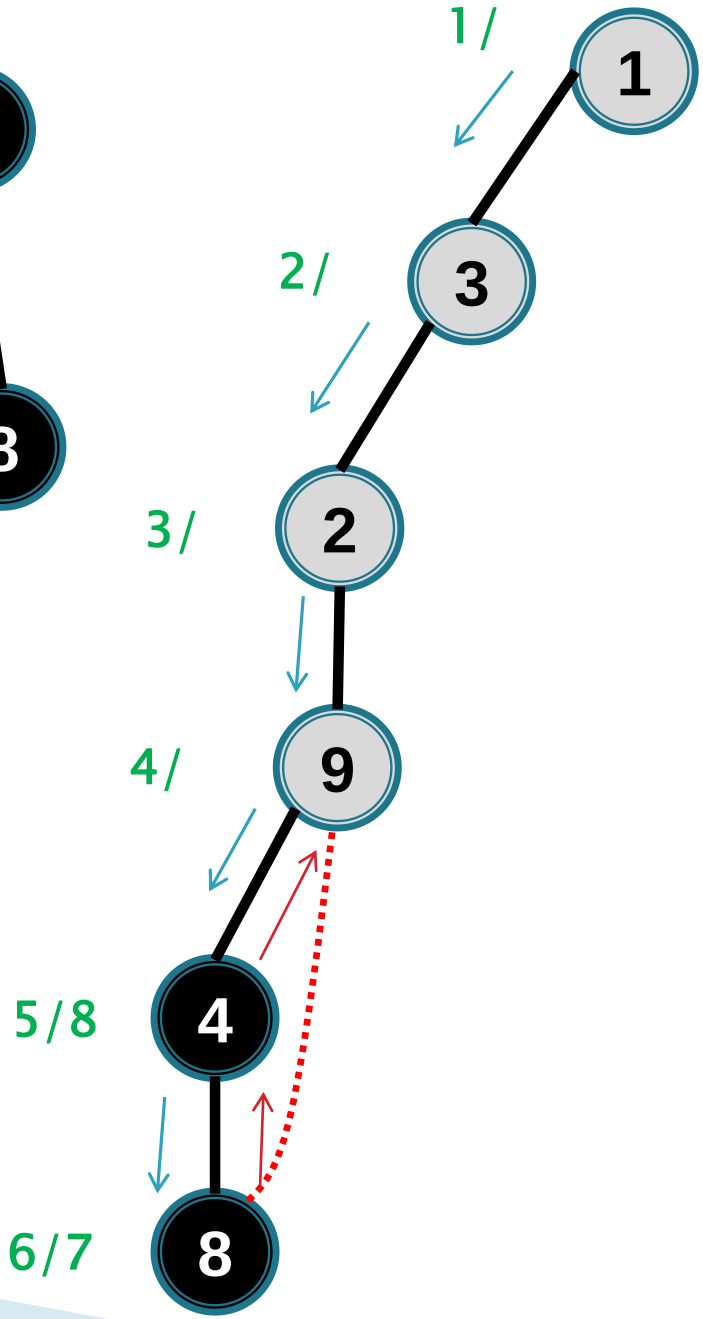

7
4
5
1
9
8
3
2
6

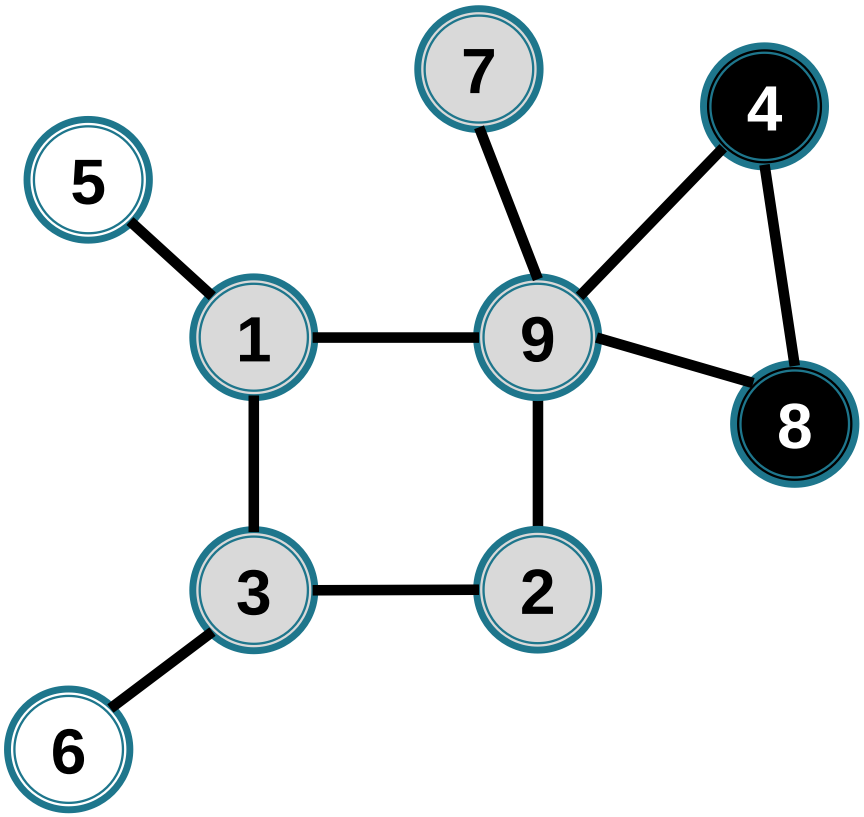

1/
1
2/
3
3/
2
4/
9
5/
4
6/
8

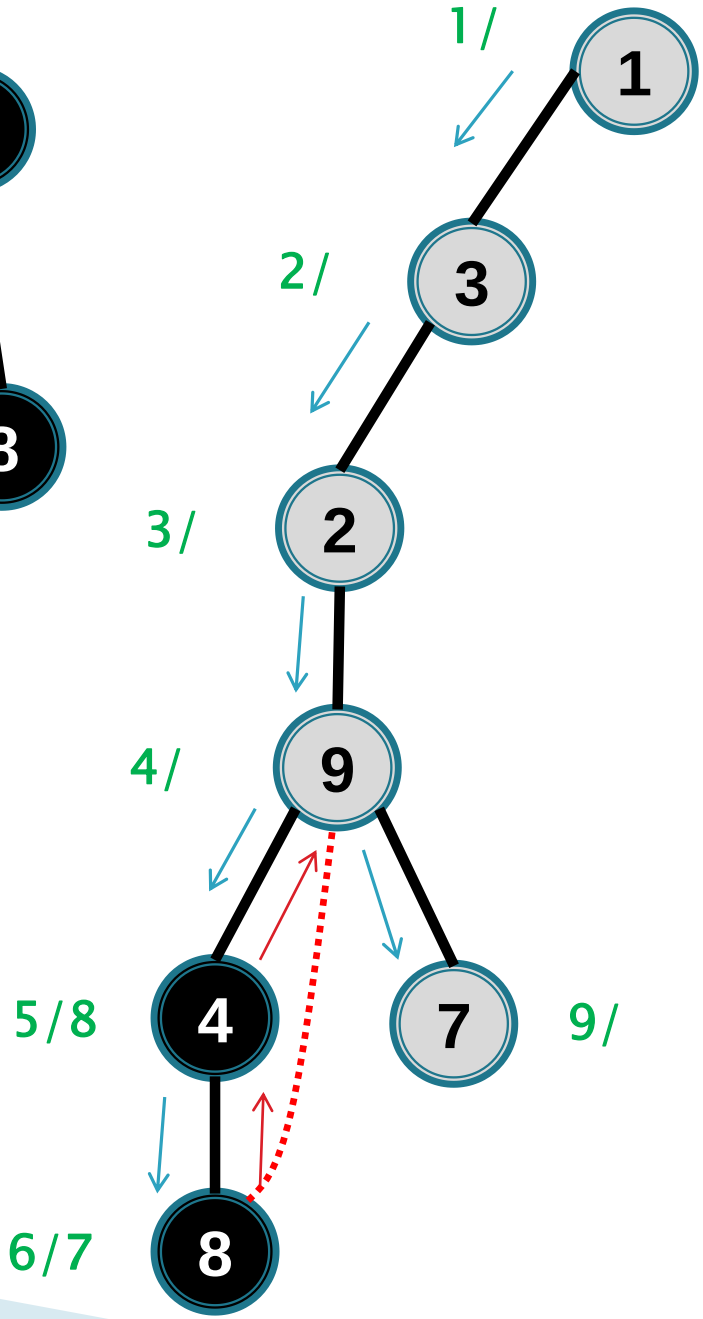

7
4
5
1
9
8
3
2
6

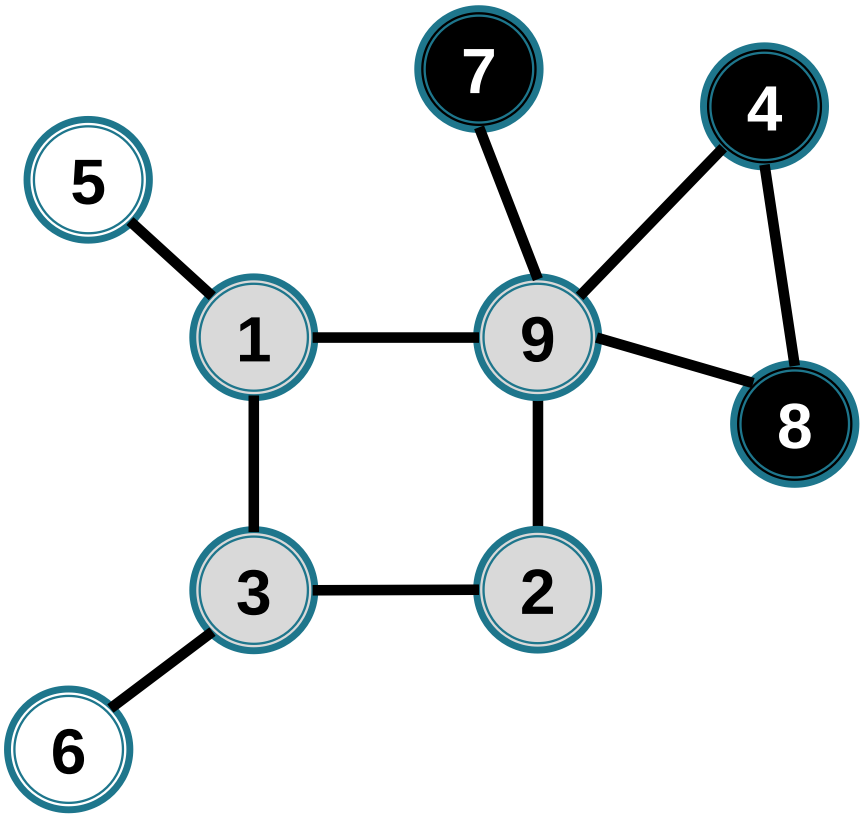

1/
1
2/
3
3/
2
4/
9
5/
4
6/7
8

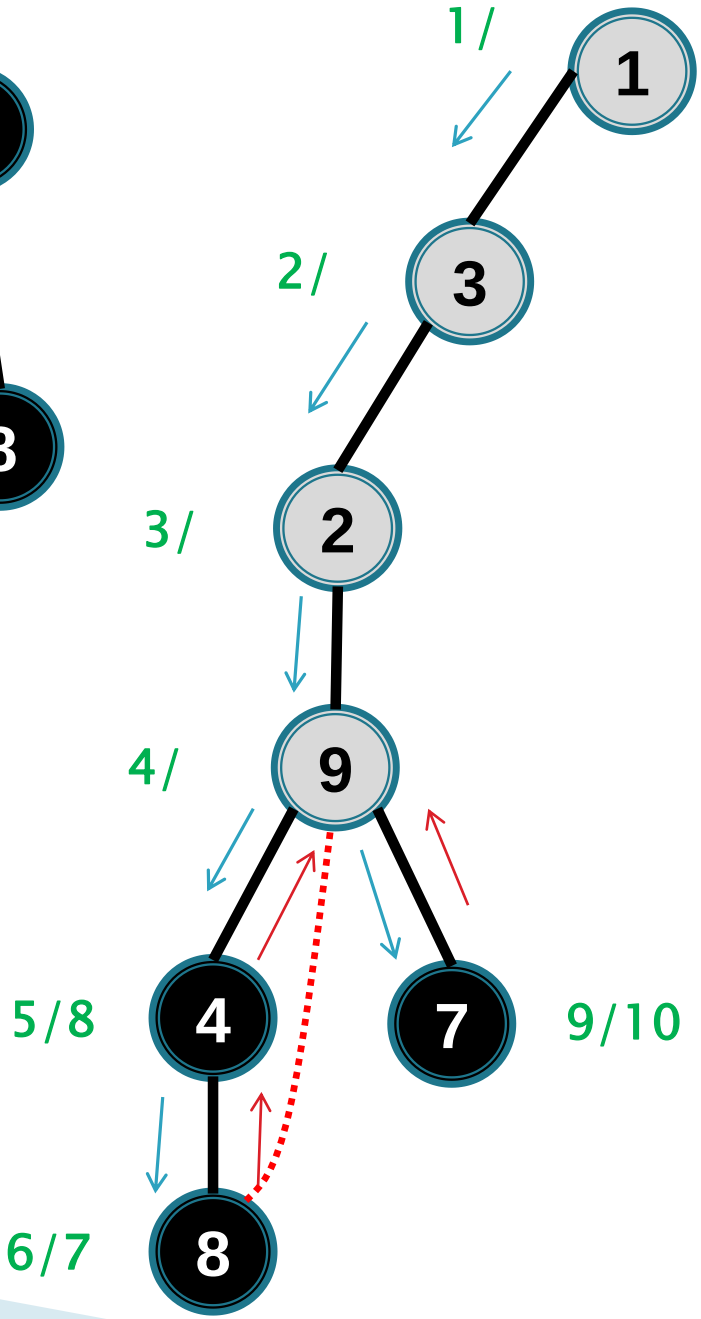

7
4
5
1
9
8
3
2
6

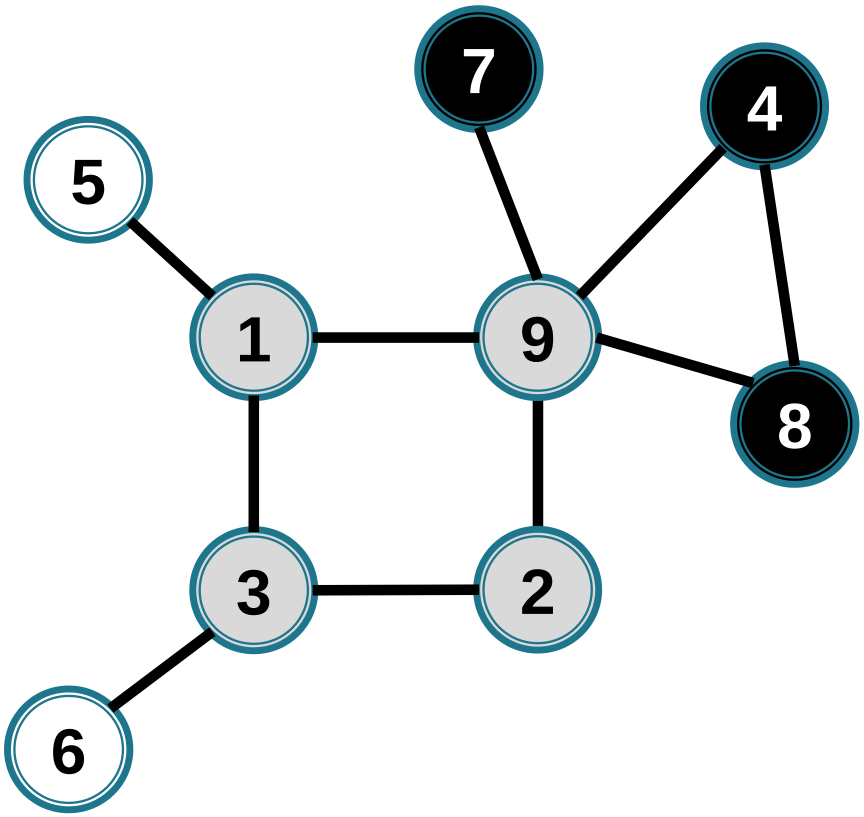

1/
1
2/
3
3/
2
4/
9
5/8
4
6/7
8

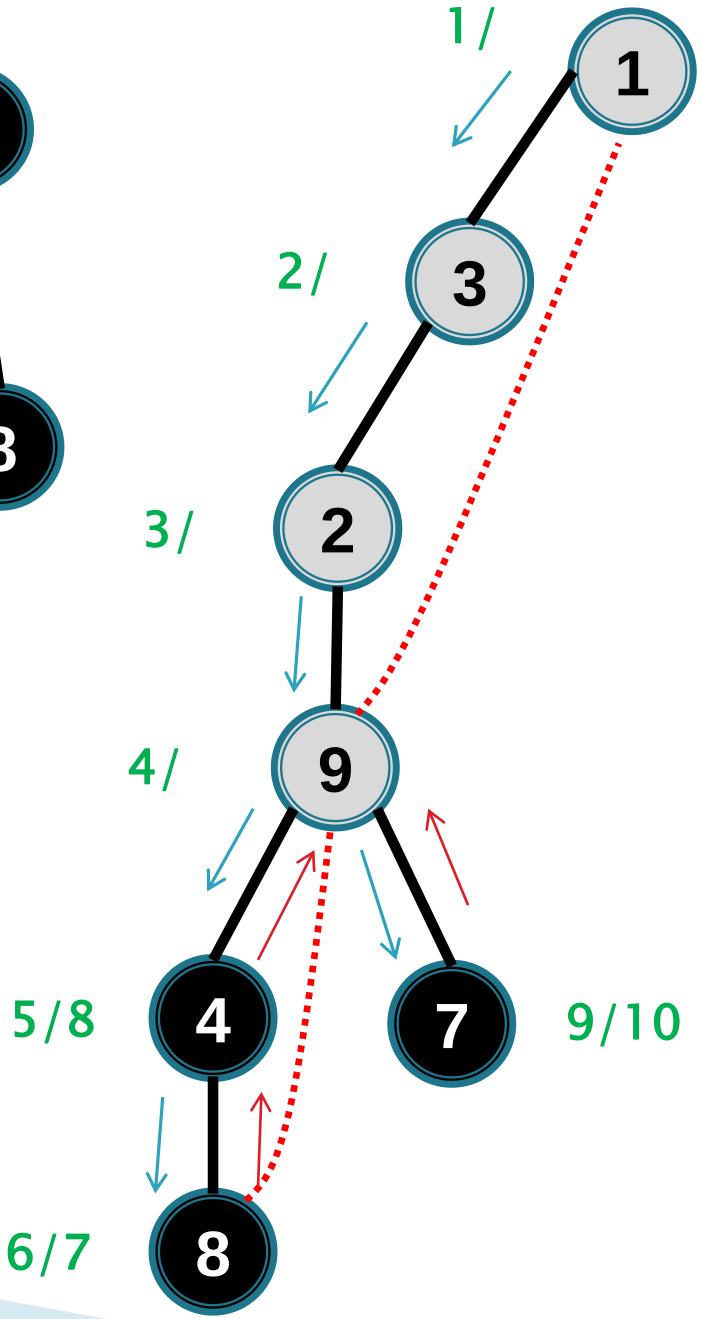

7
4
5
1
9
8
3
2
6

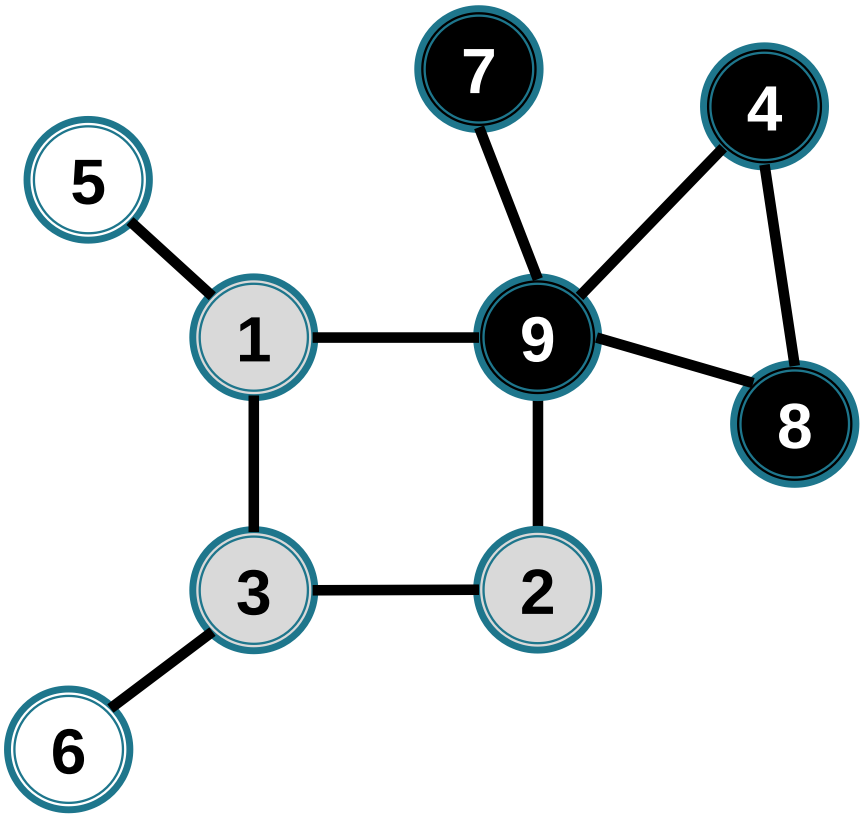

1/
1
2/
3
3/
2
4/
9
5/8
4
7
9/
6/7
8

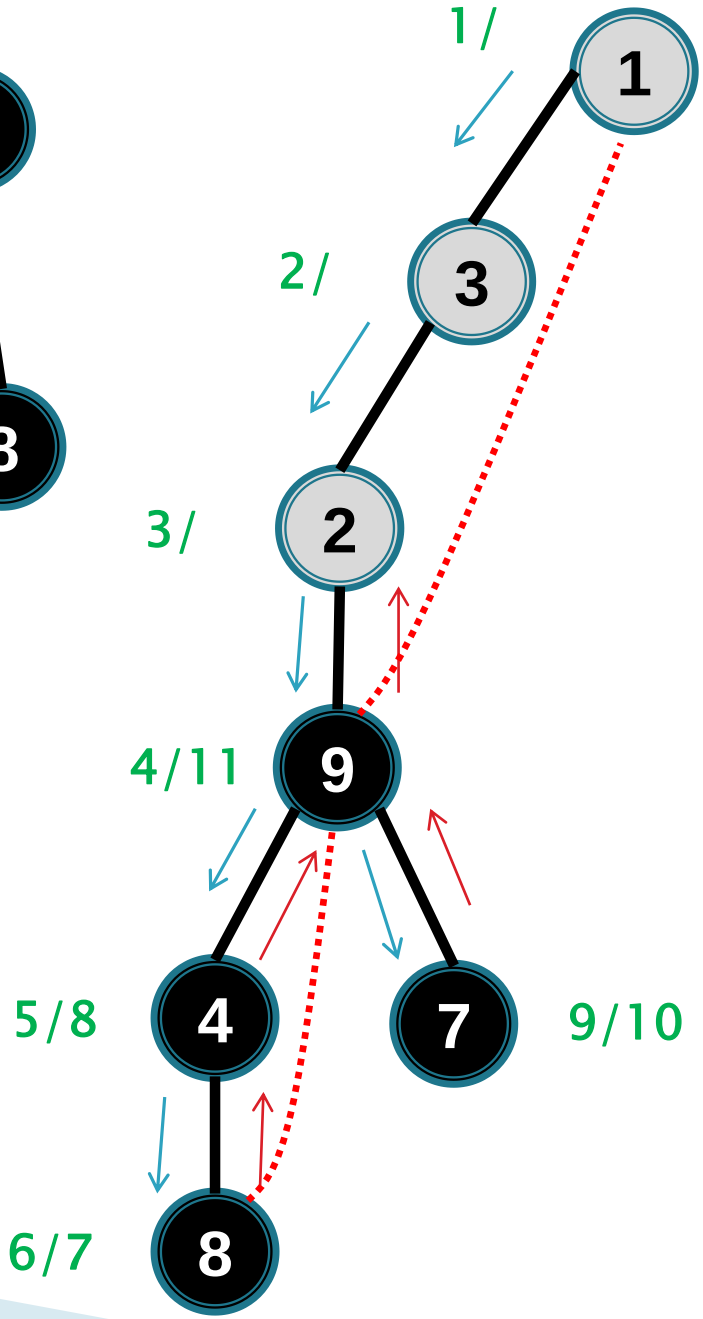

7
4
5
1
9
8
3
2
6

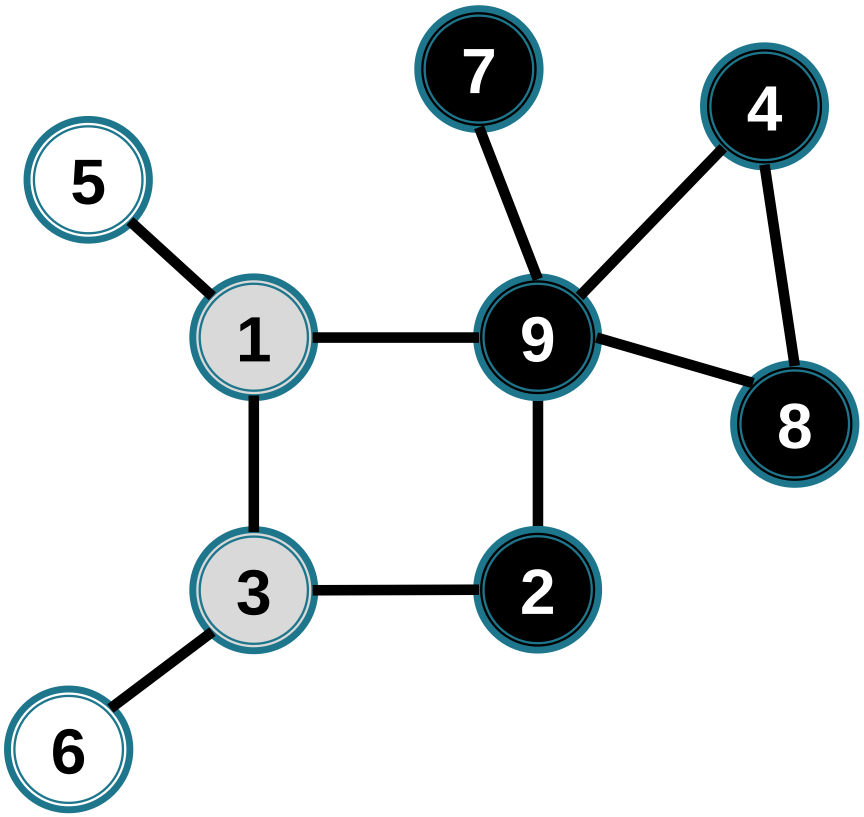

1/
1
2/
3
3/
2
4/
9
5/8
4
7
9/10
6/7
8

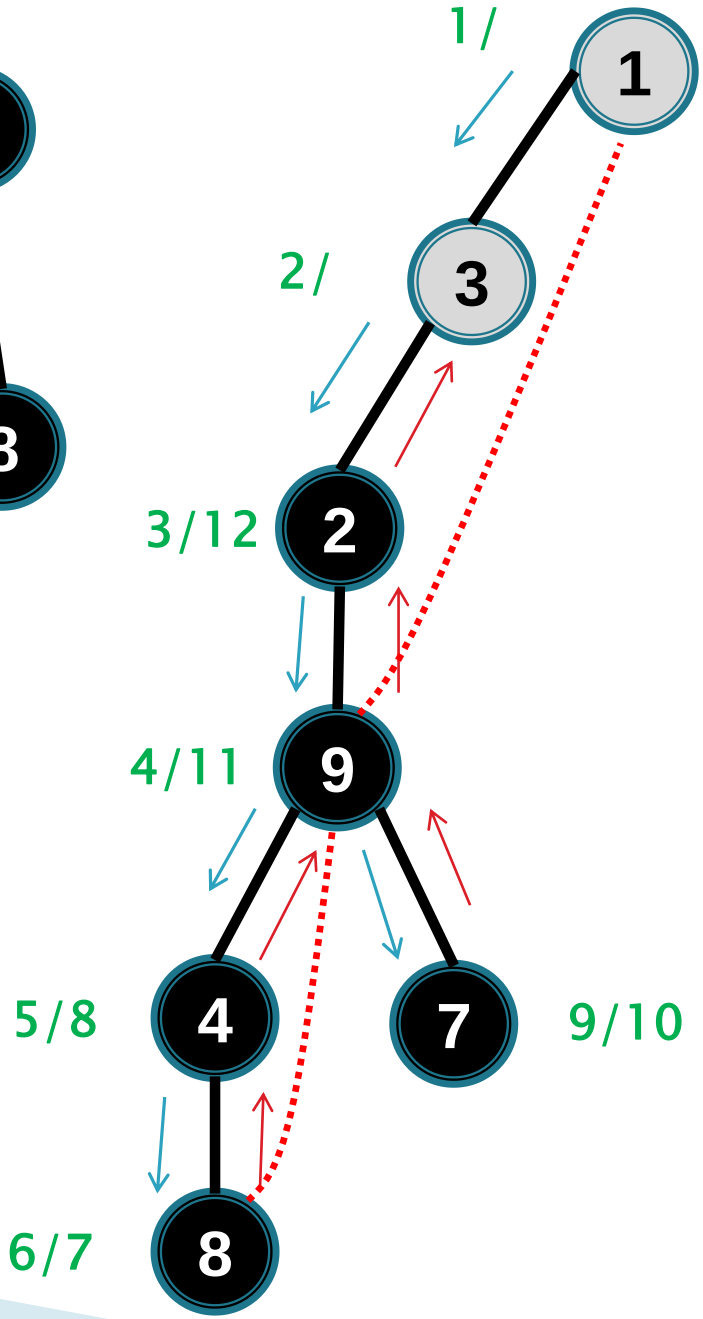

7
4
5
1
9
8
3
2
6

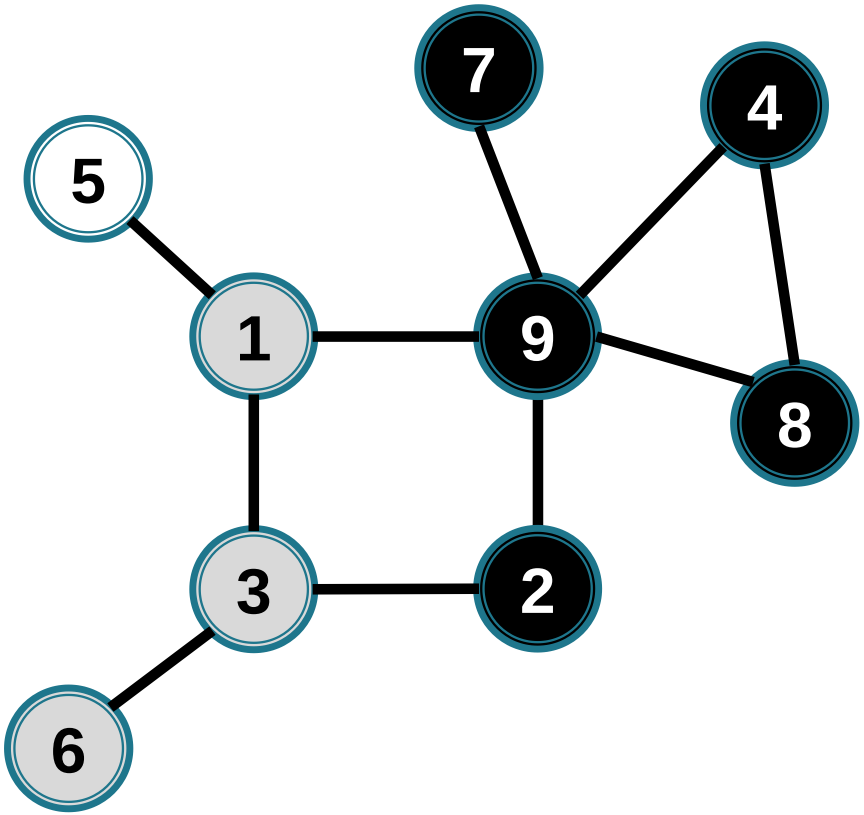

1/
1
2/
3
3/
2
4/
9
5/8
4
7
9/10
6/7
8

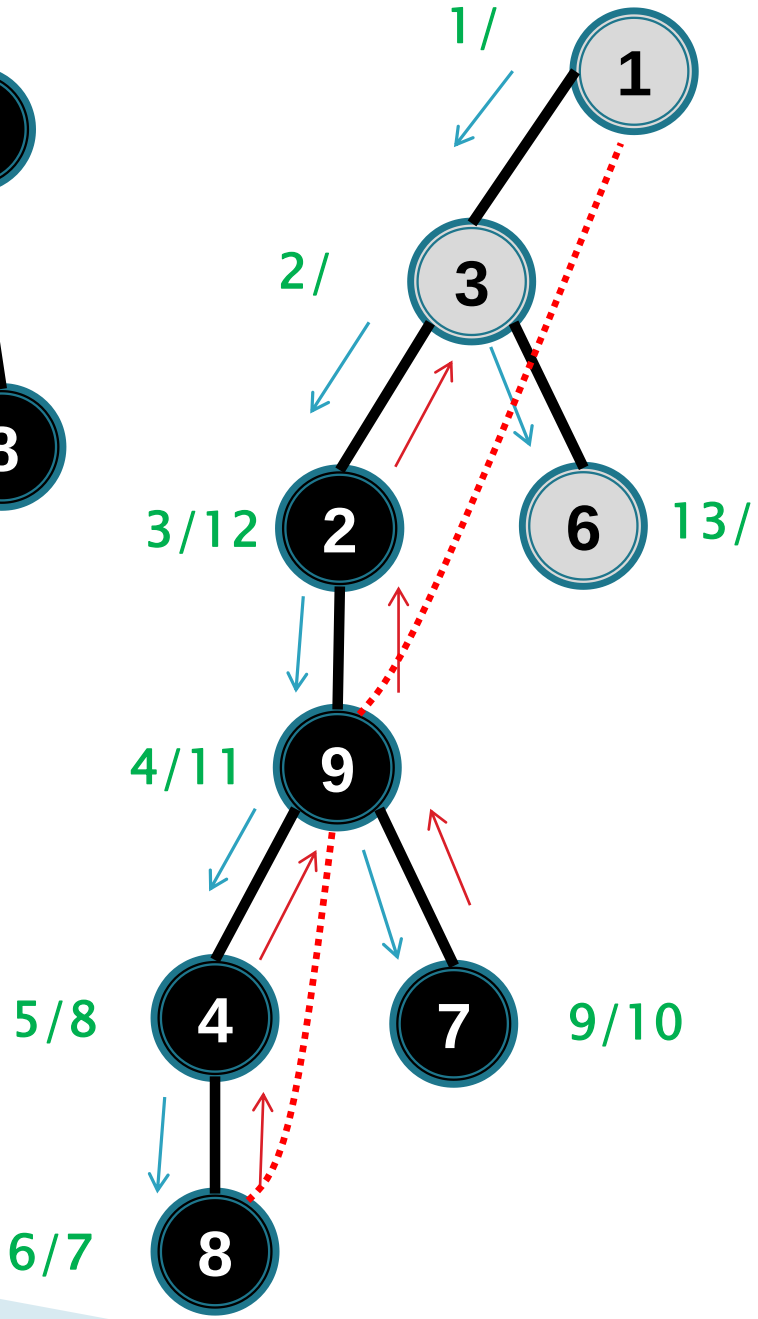

7
4
5
1
9
8
3
2
6

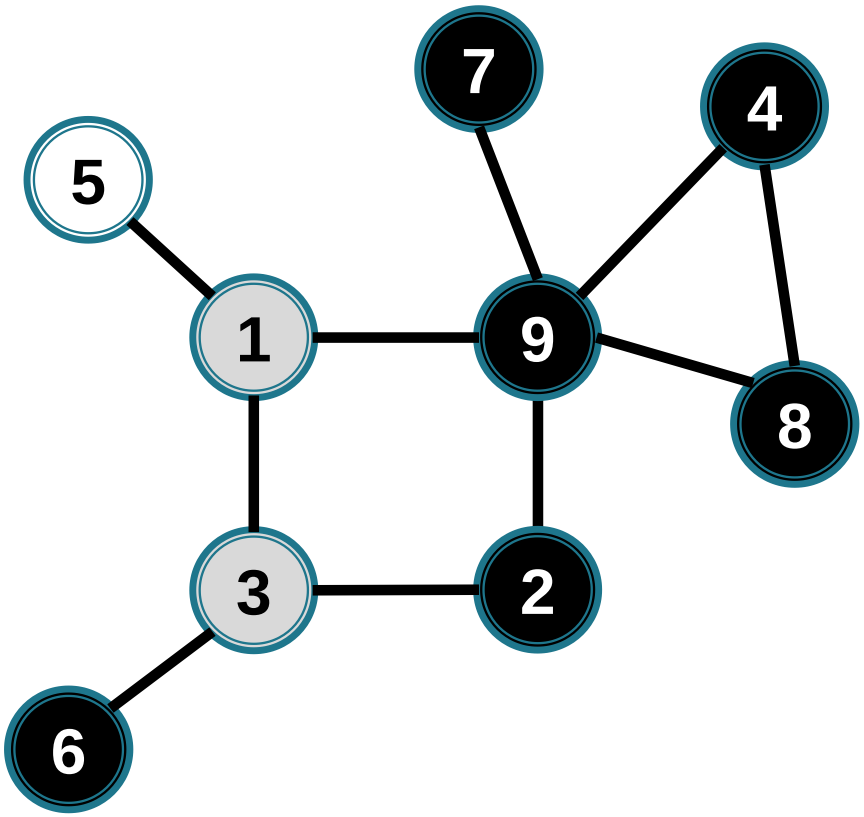

1/
1
2/
3
3/
2
4/11
9
5/8
4
7
9/10
6/7
8

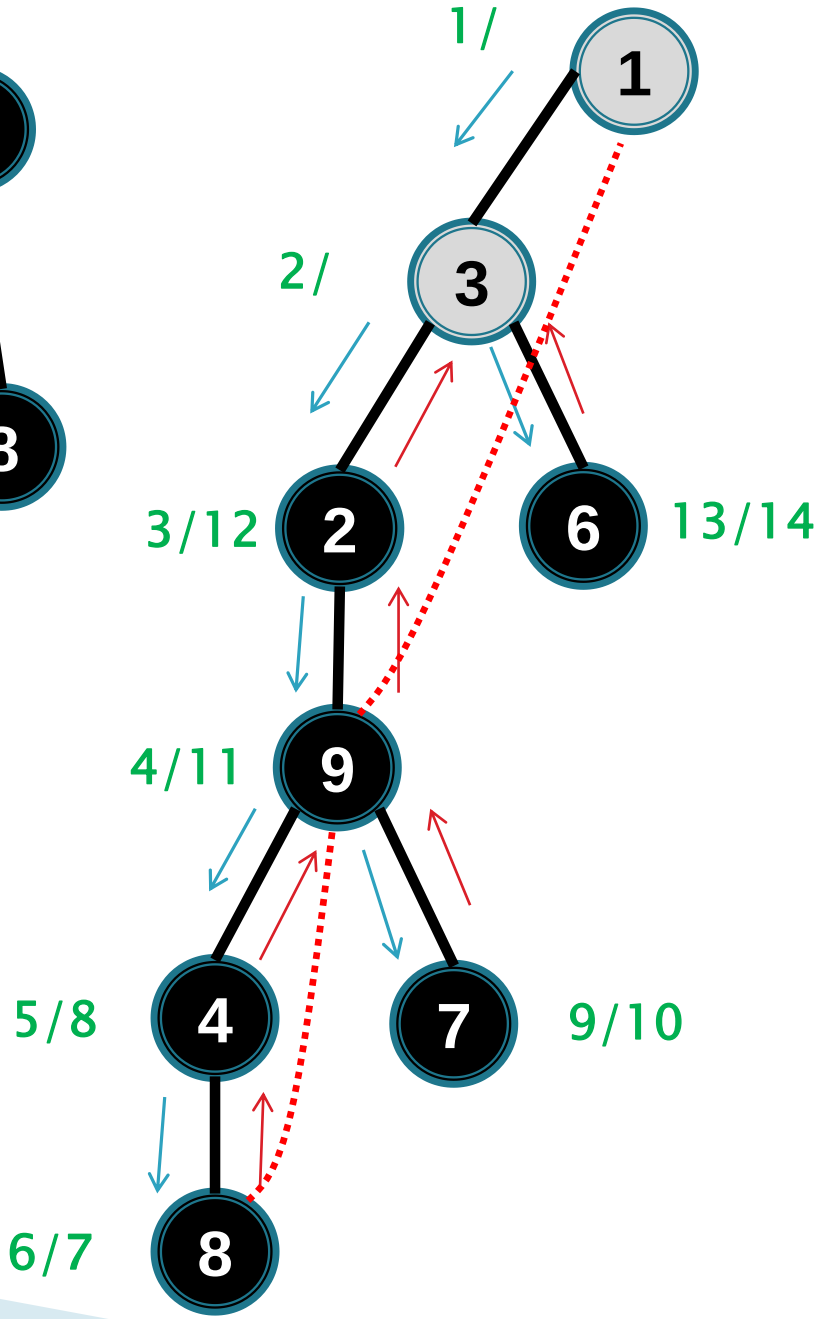

7
4
5
1
9
8
3
2
6

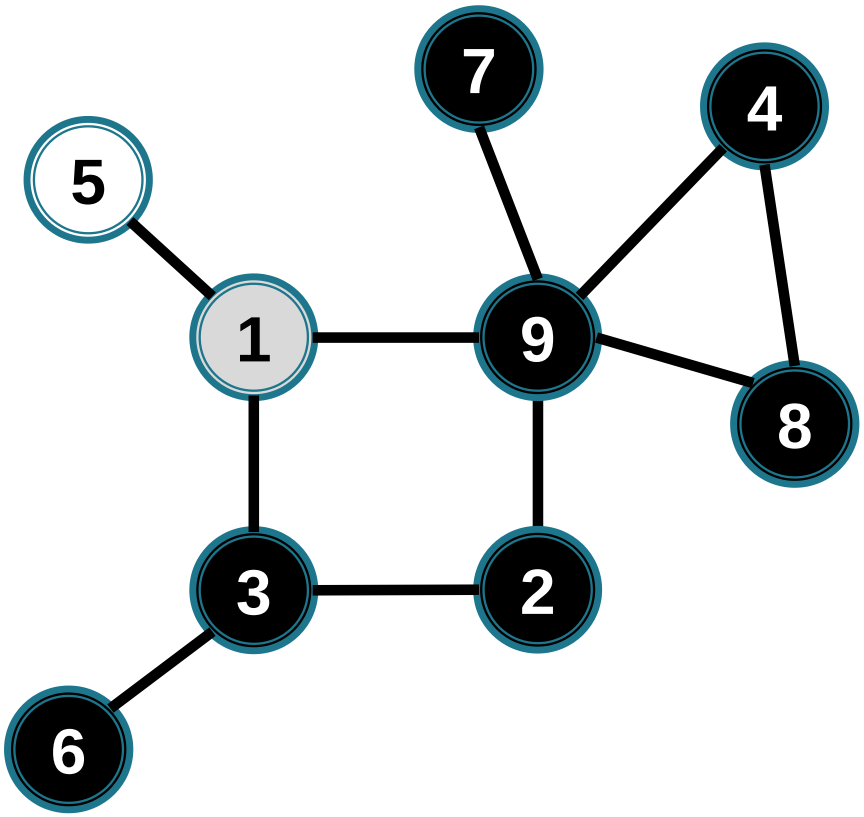

1/
1
2/
3
3/12
2
4/11
9
5/8
4
7
9/10
6/7
8

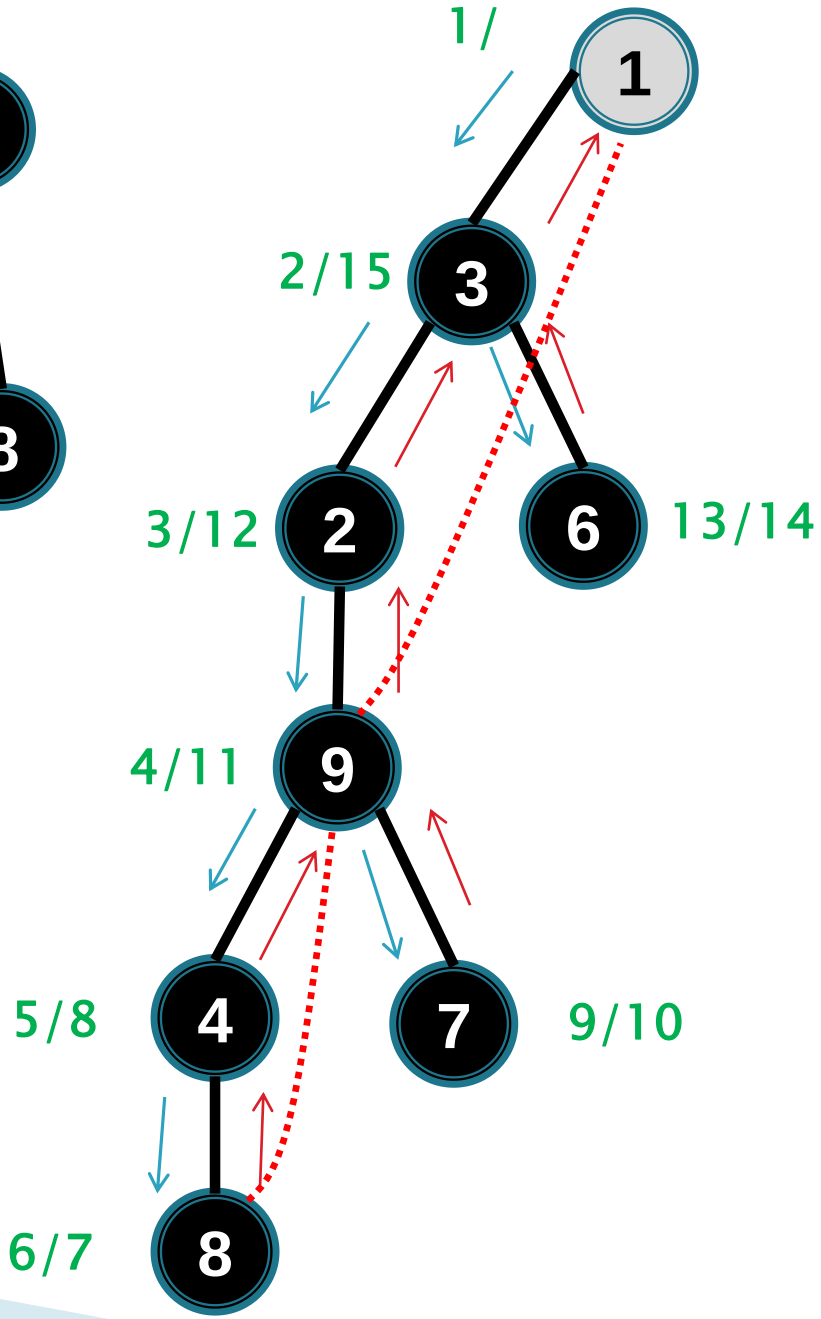

7
4
5
1
9
8
3
2
6

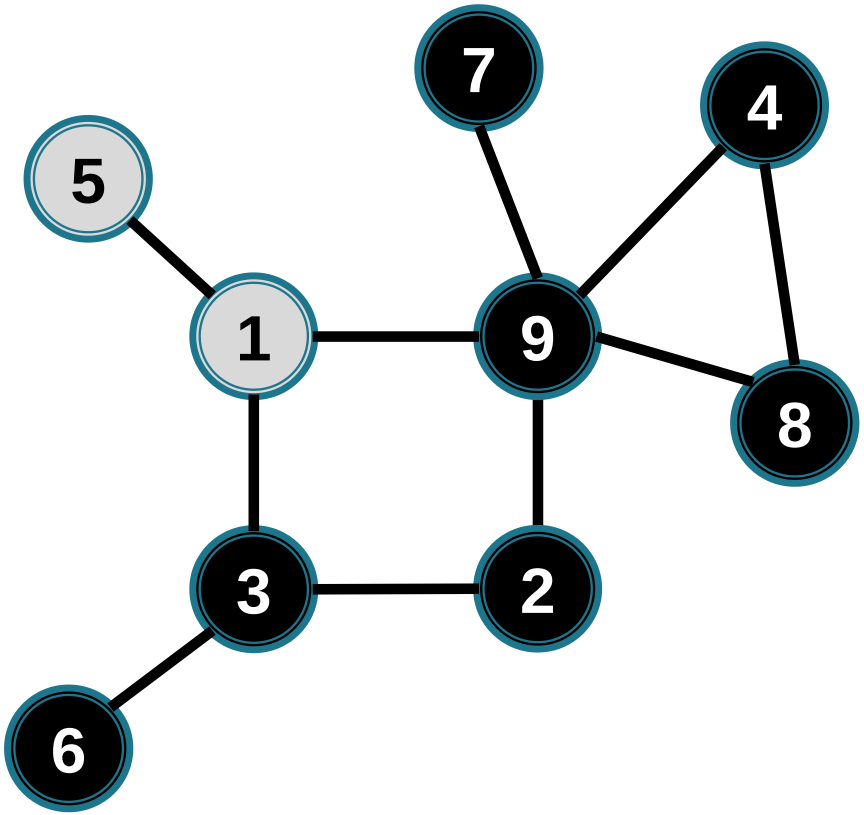

1/
1
2/
3
3/12
2
6
13/
4/11
9
5/8
4
7
9/10
6/7
8

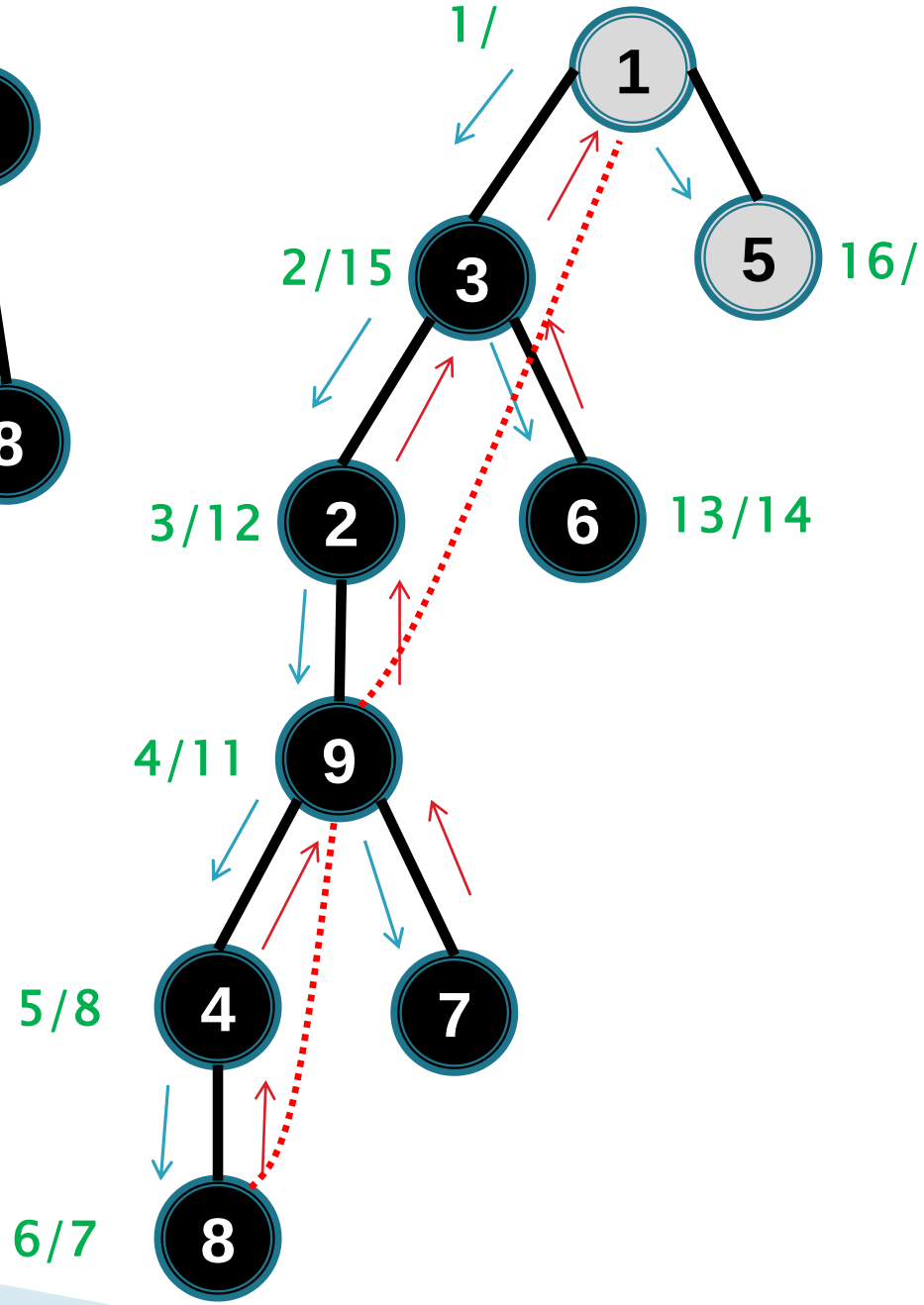

7
4
5
1
9
8
3
2
6

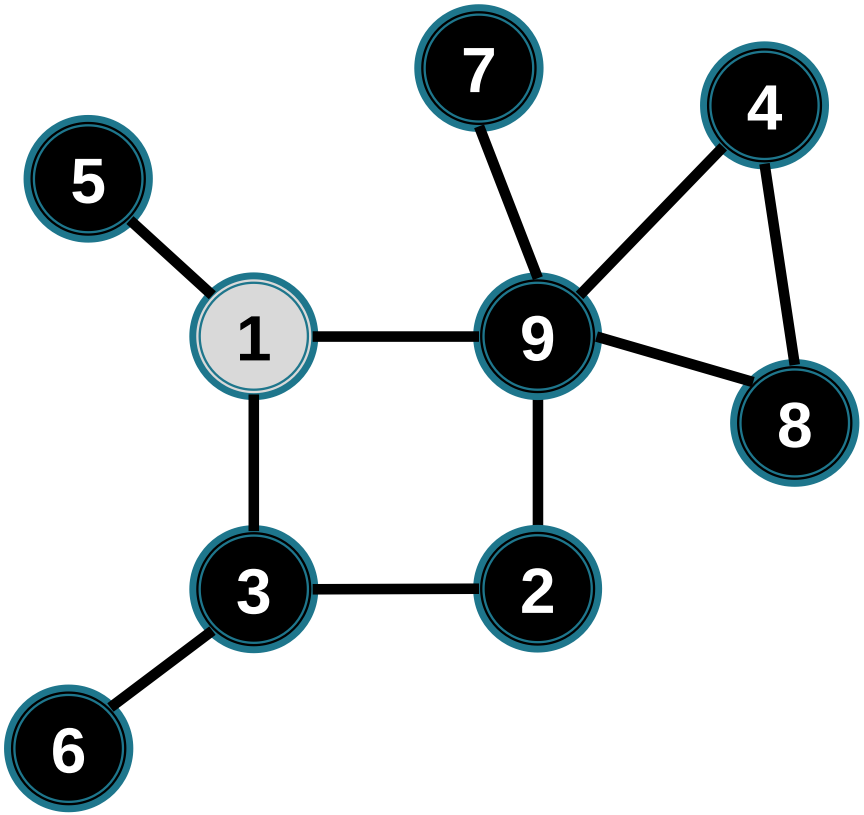

1/
1
2/
3
3/12
2
6
13/14
4/11
9
5/8
4
7
9/10
6/7
8

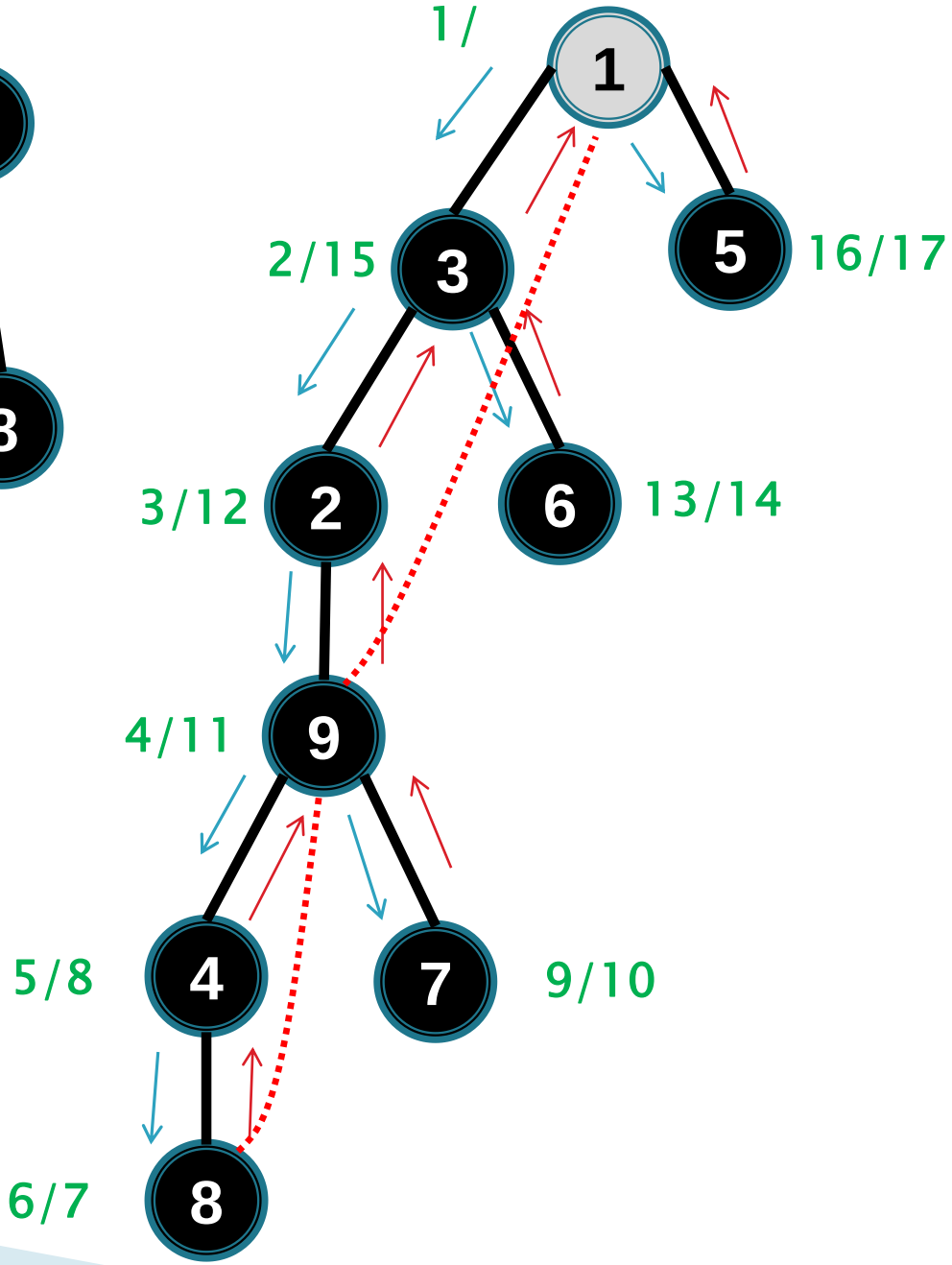

7
4
5
1
9
8
3
2
6

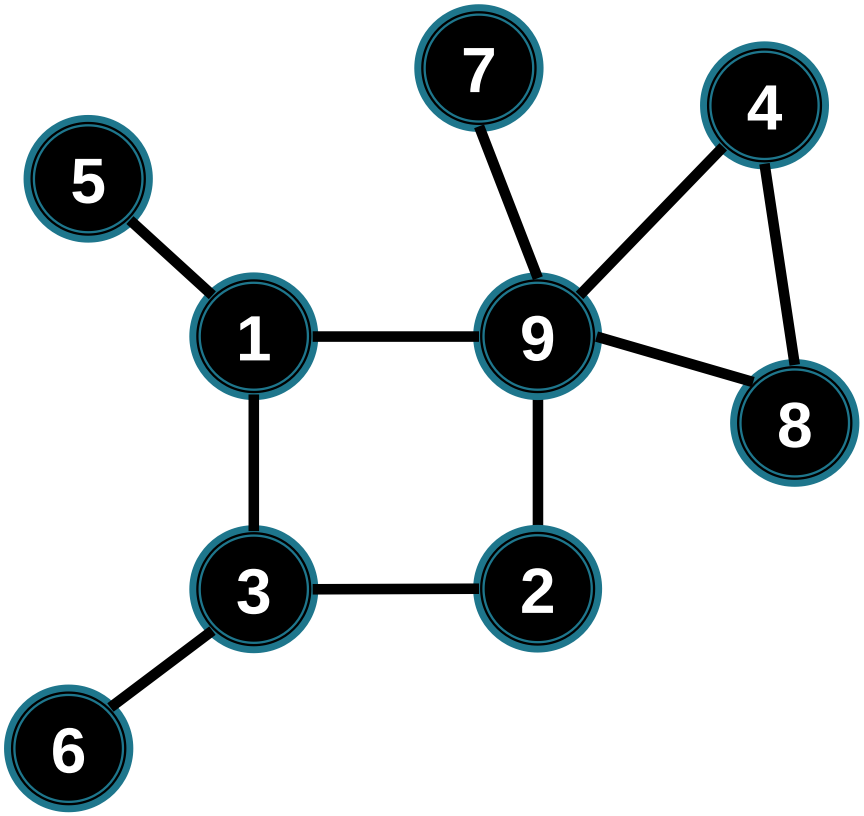

1/
1
2/15
3
3/12
2
6
13/14
4/11
9
5/8
4
7
9/10
6/7
8

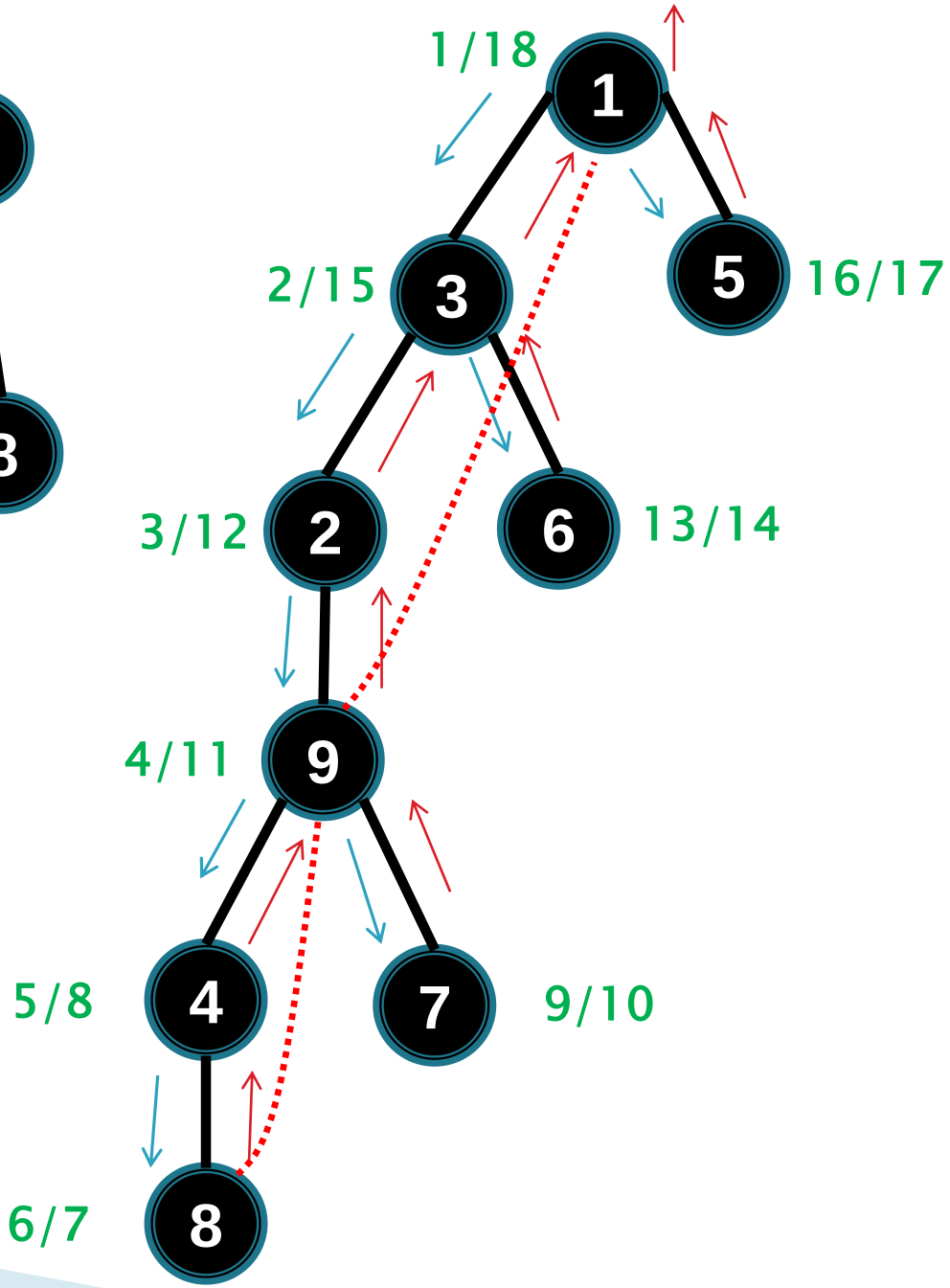

7
4
5
1
9
8
3
2
6


1/
1
2/15
3
5
16/
3/12
2
6
13/14
4/11
9
5/8
4
7
6/7
8

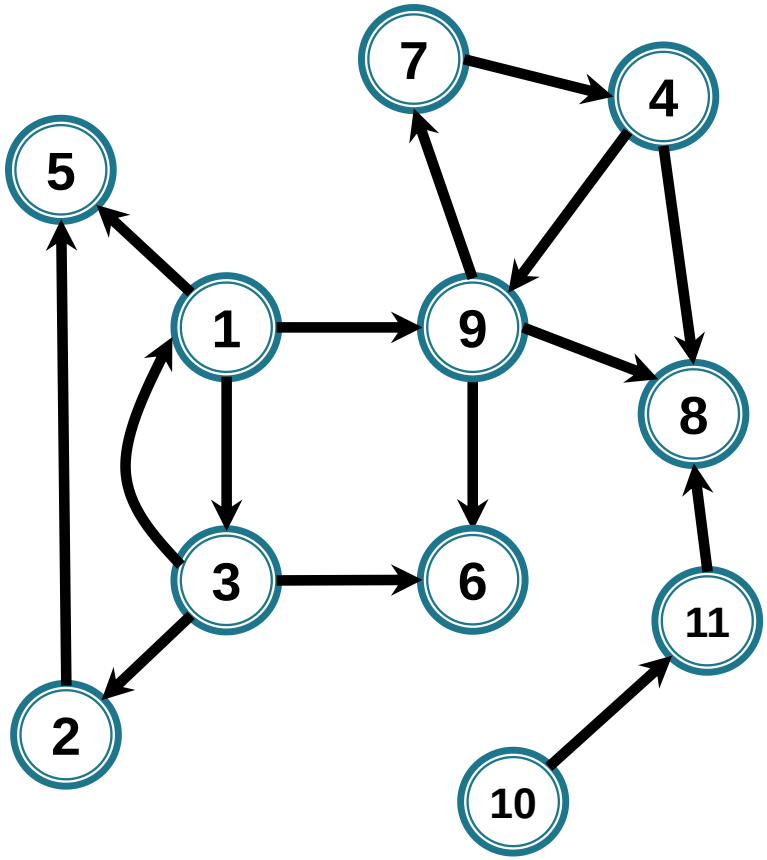

7
4
5
1
9
8
3
2
6


1/
1
2/15
3
5
16/17
3/12
2
6
13/14
4/11
9
5/8
4
7
9/10
6/7
8

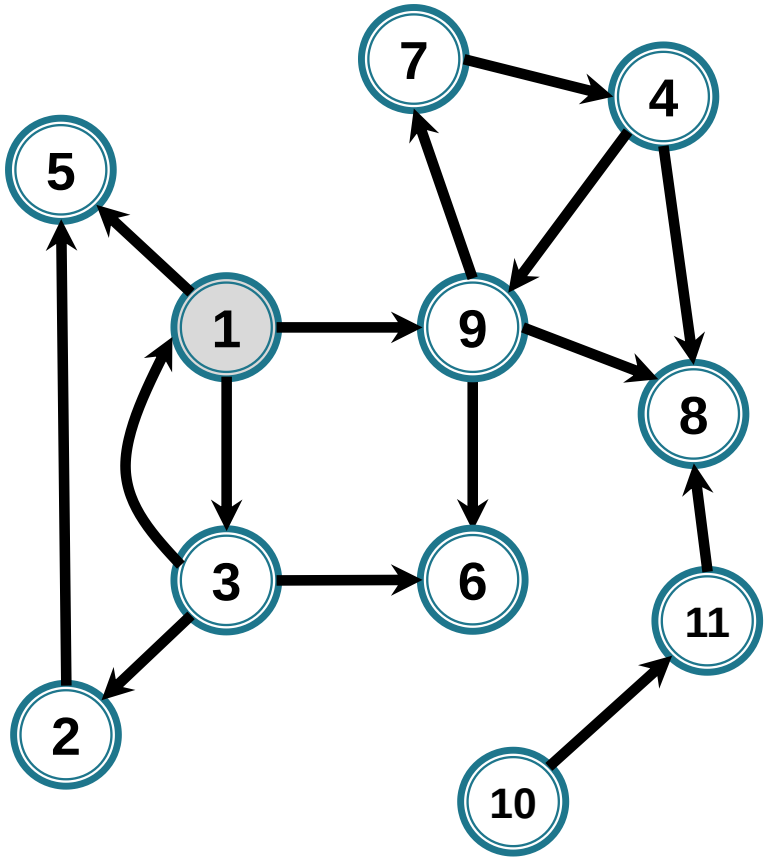

7
4
5
1
9
8
3
2
6


1/18
1
2/15
3
5
16/17
3/12
2
6
13/14
4/11
9
5/8
4
7
9/10
6/7
8

Exemplu DF graf orientat

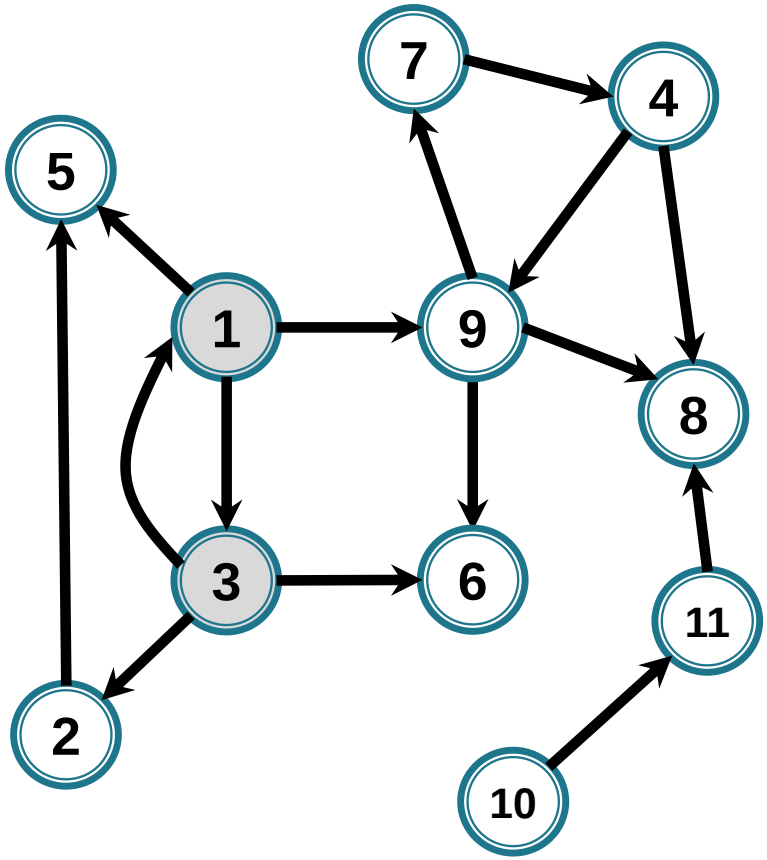


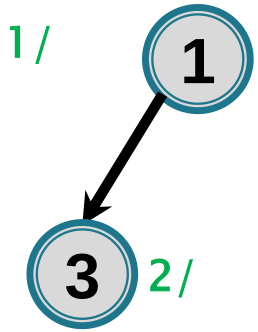
Exemplu DF graf orientat



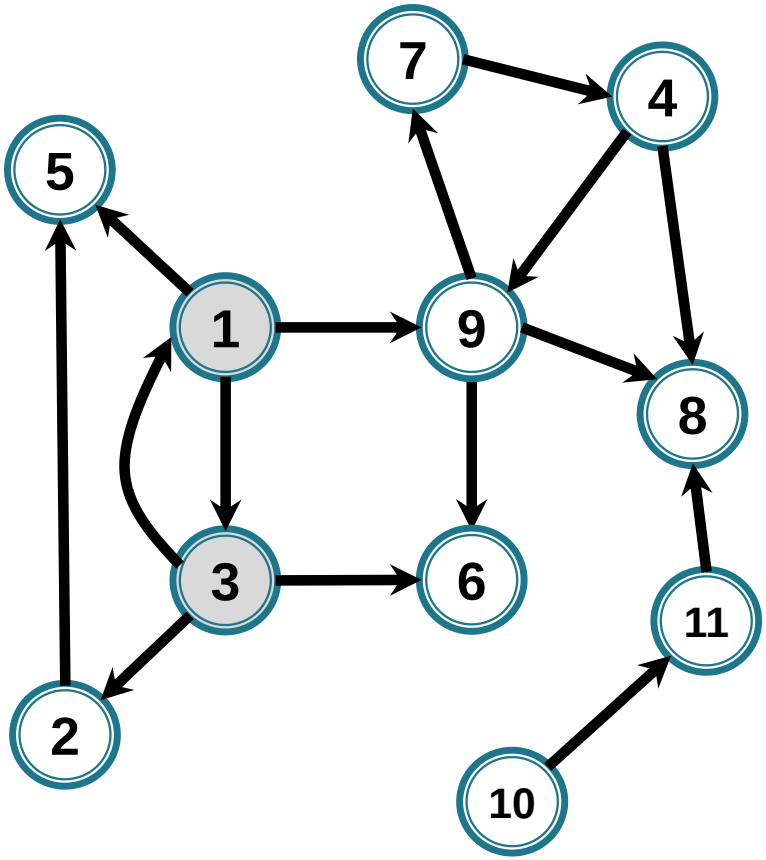
1 /

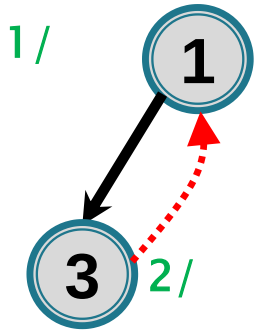
Exemplu DF graf orientat



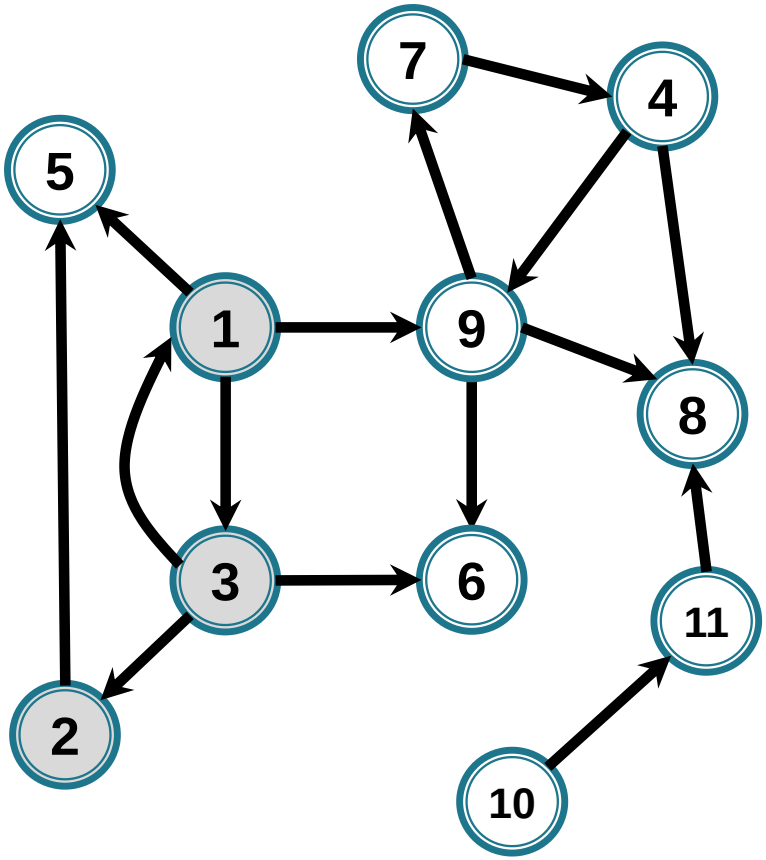


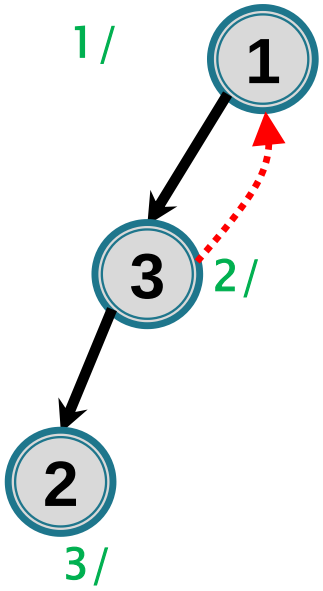
Exemplu DF graf orientat



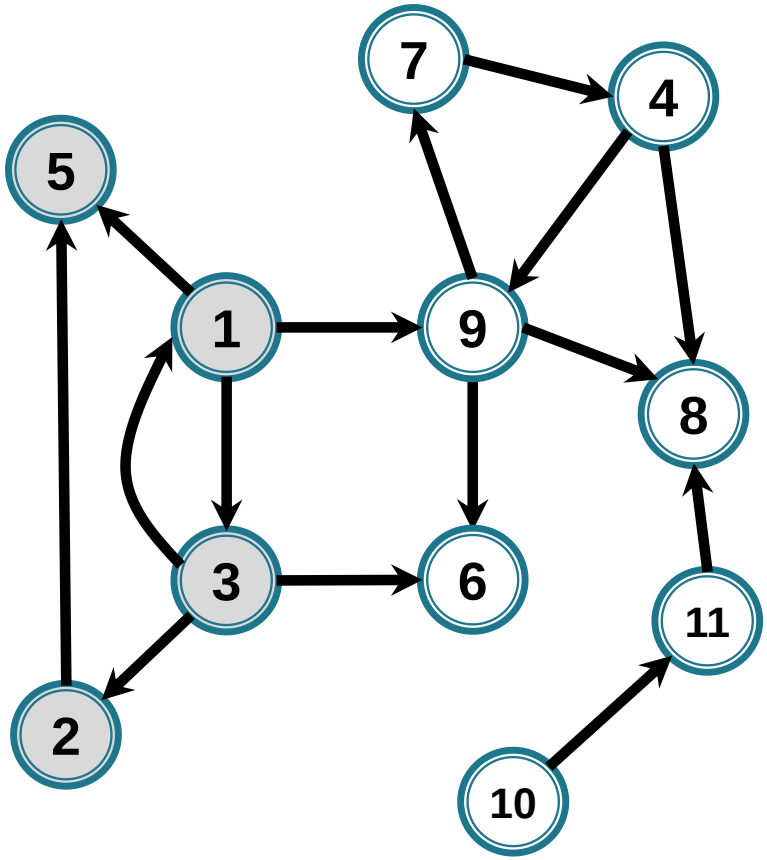


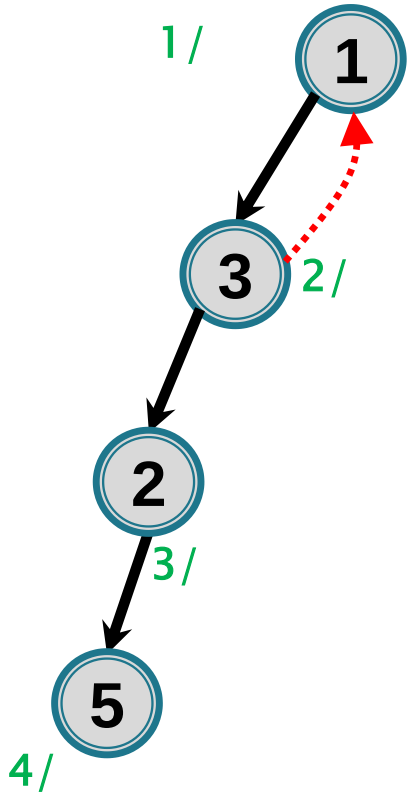
Exemplu DF graf orientat



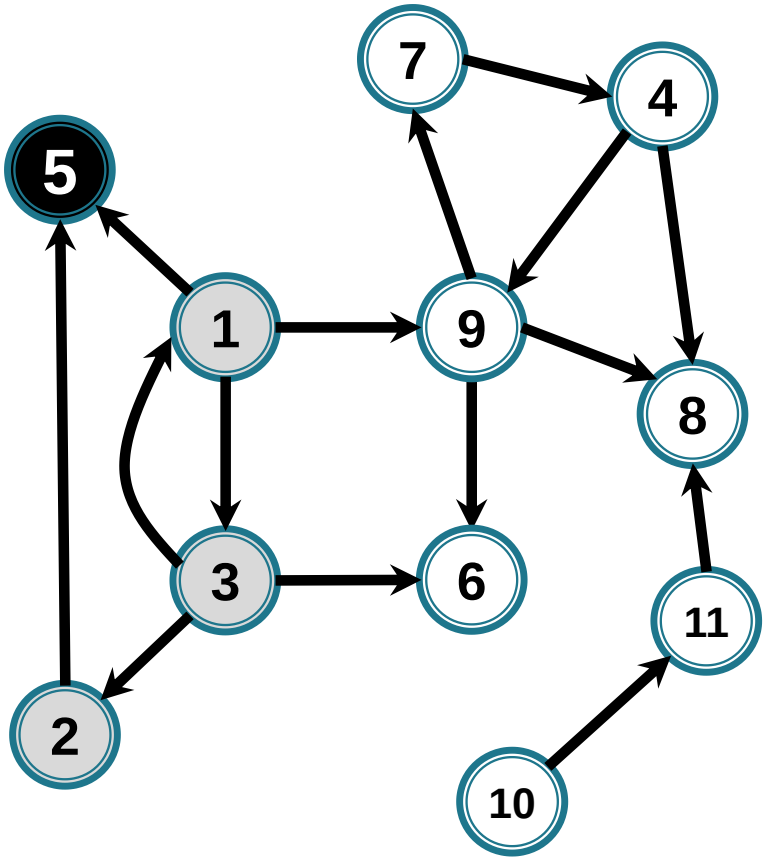


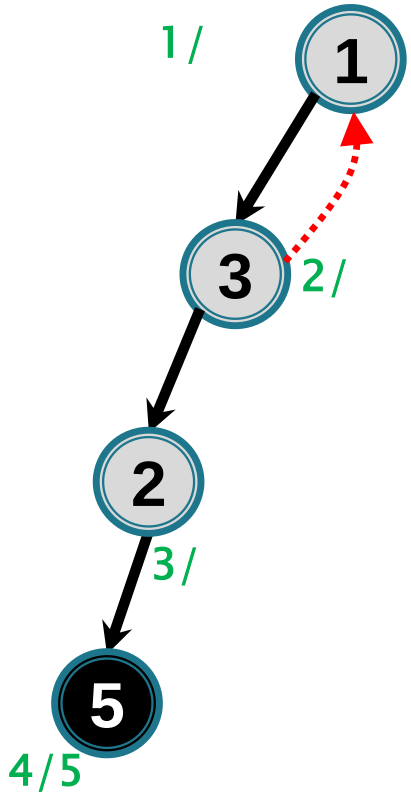
Exemplu DF graf orientat



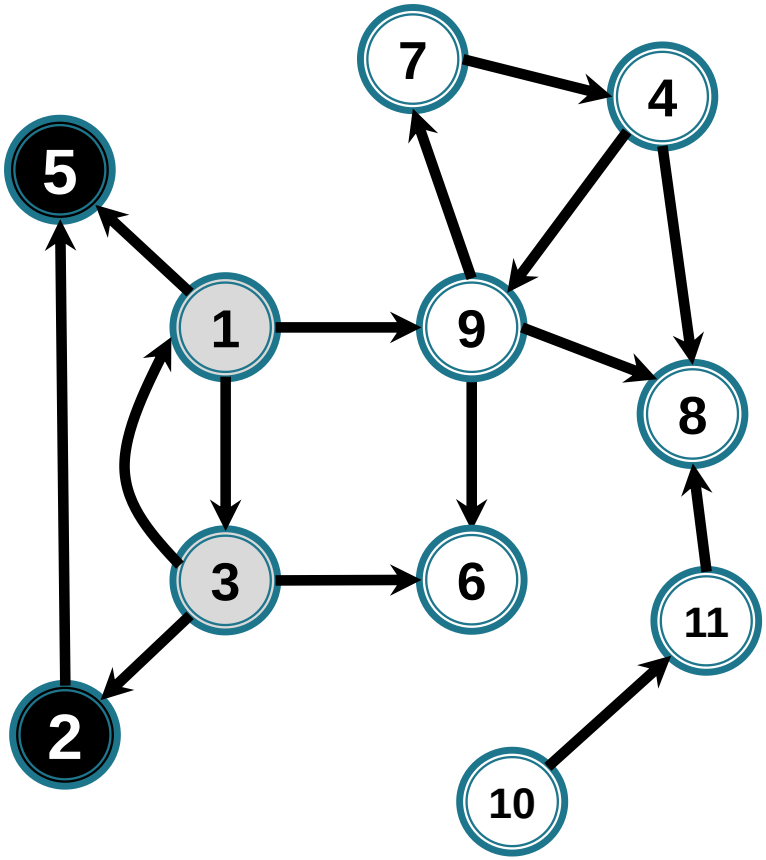


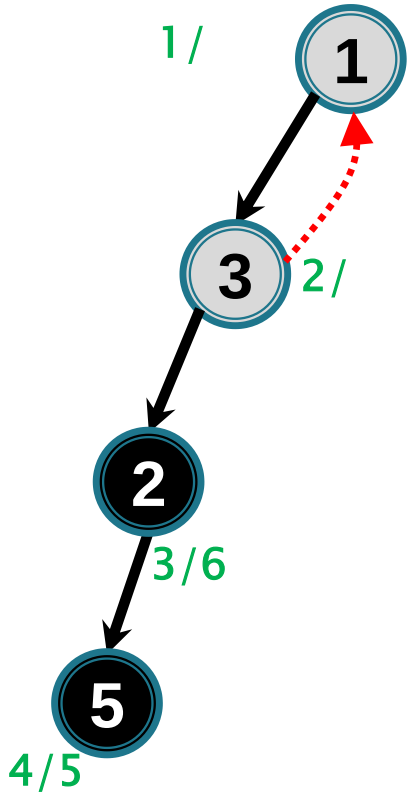
Exemplu DF graf orientat



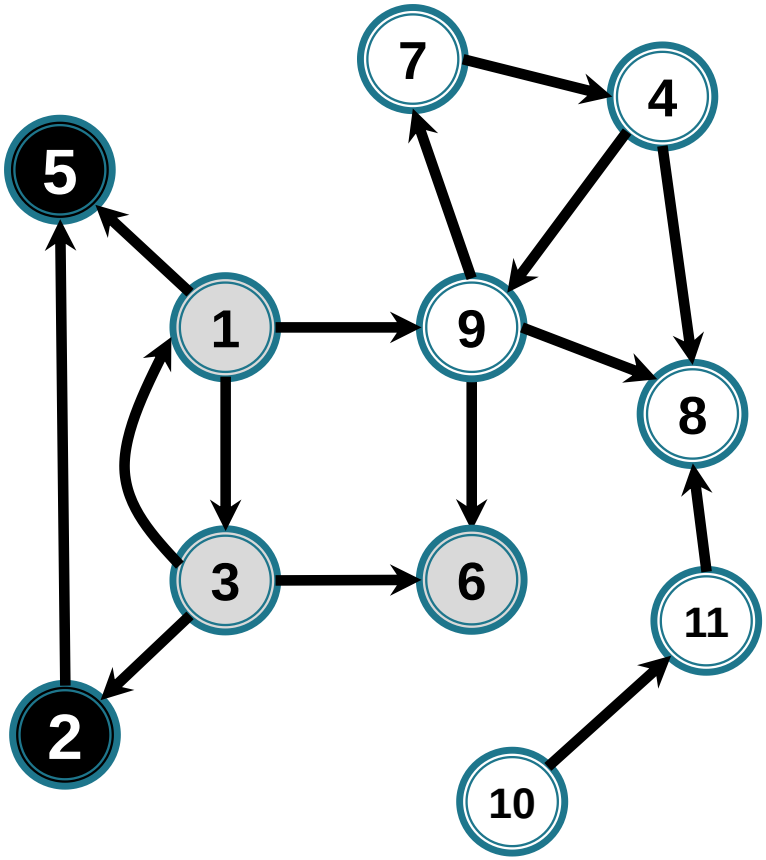


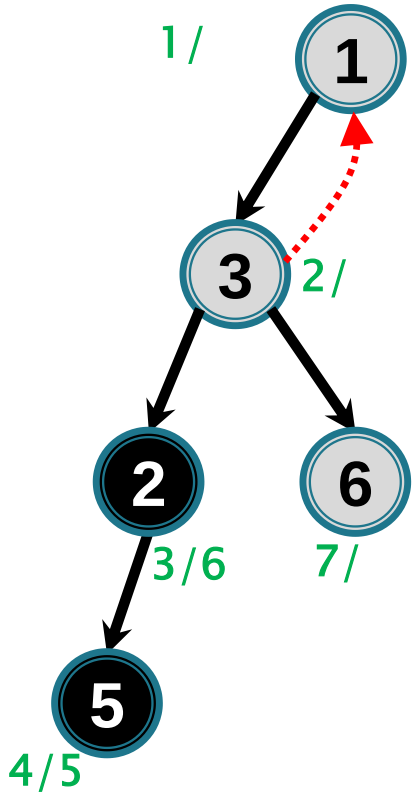
Exemplu DF graf orientat



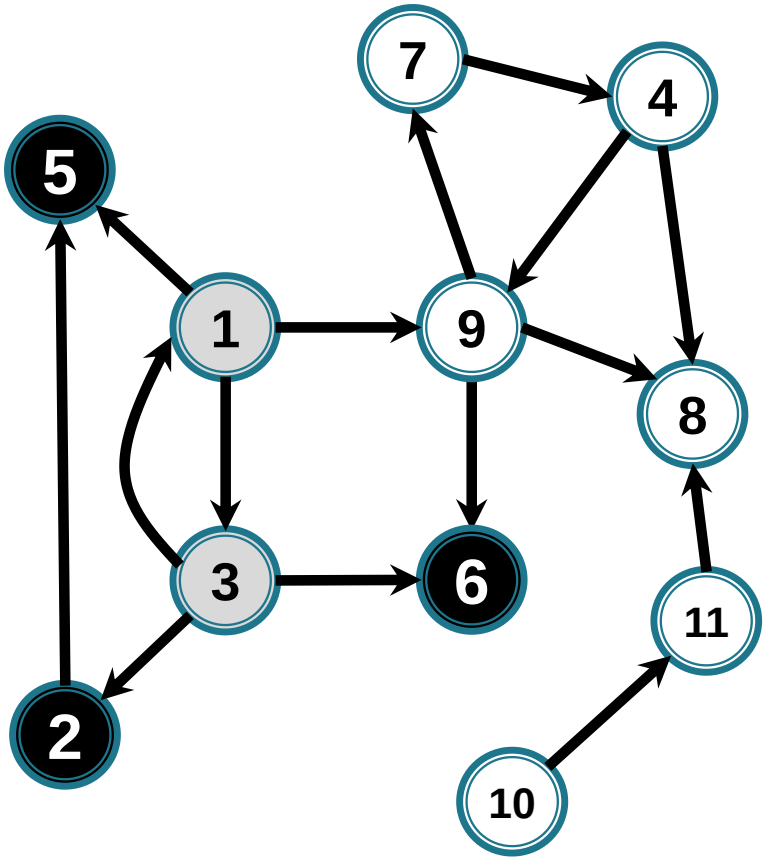


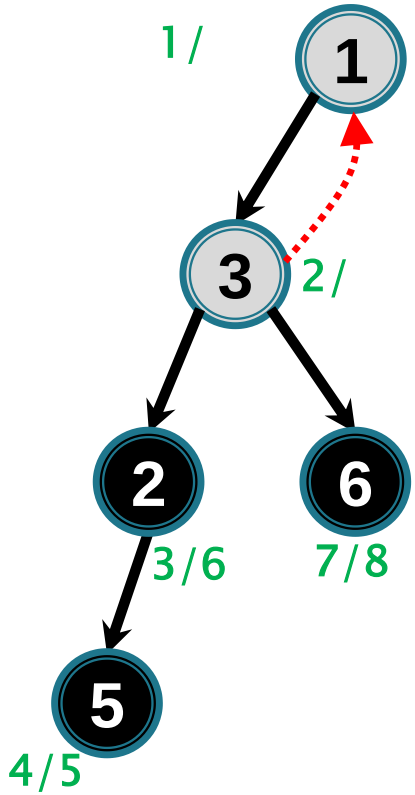
Exemplu DF graf orientat



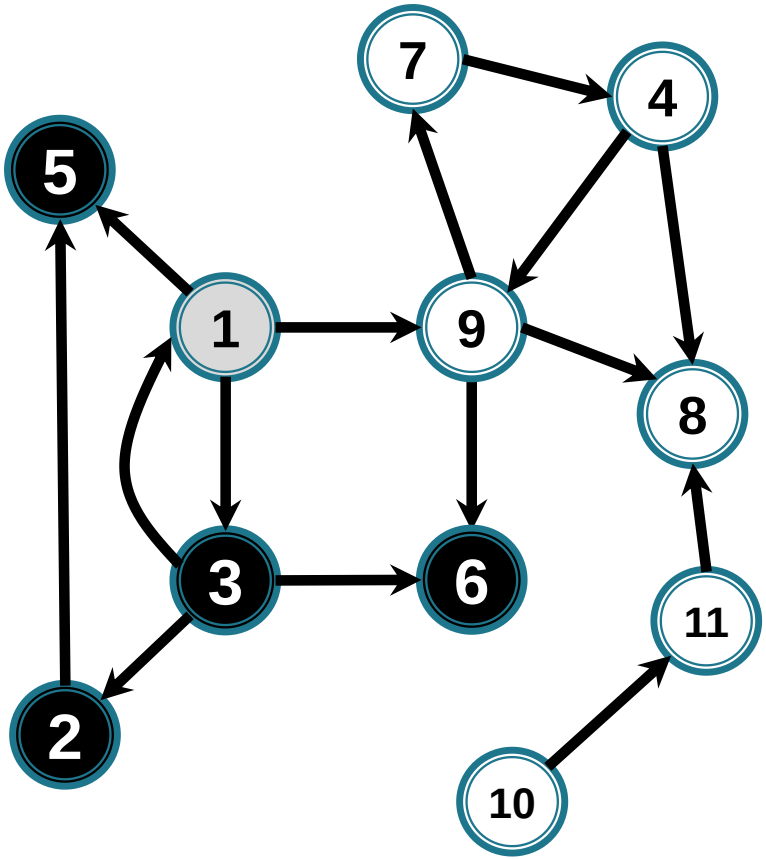


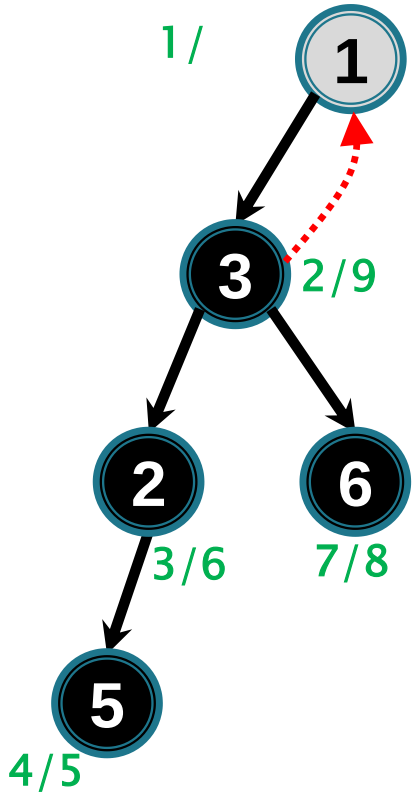
Exemplu DF graf orientat



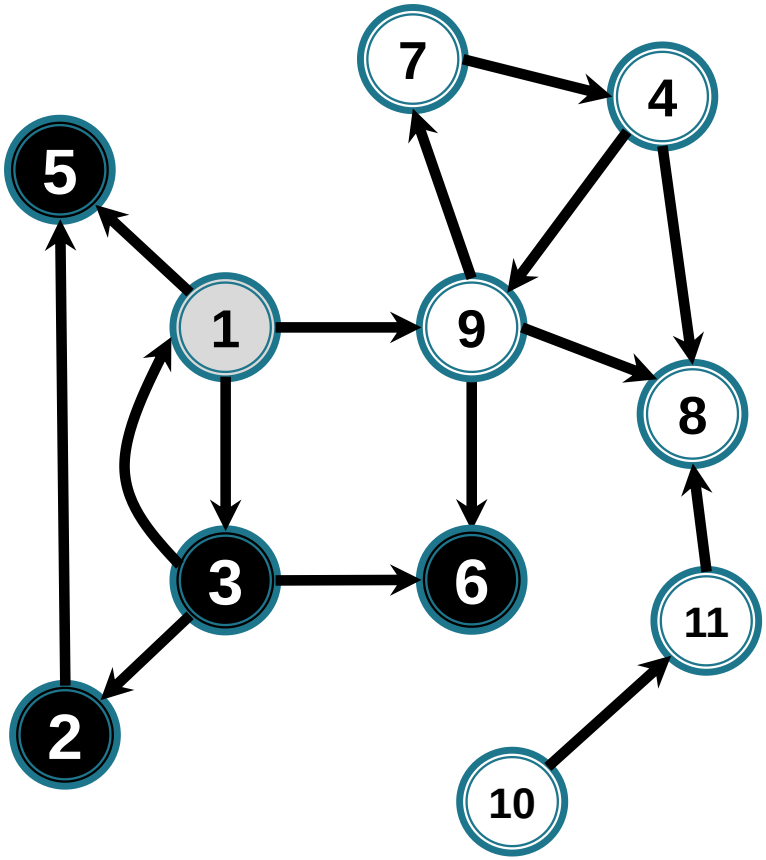


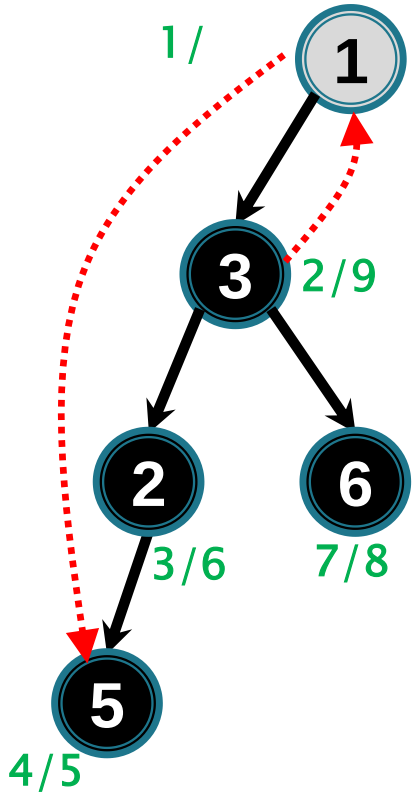
Exemplu DF graf orientat



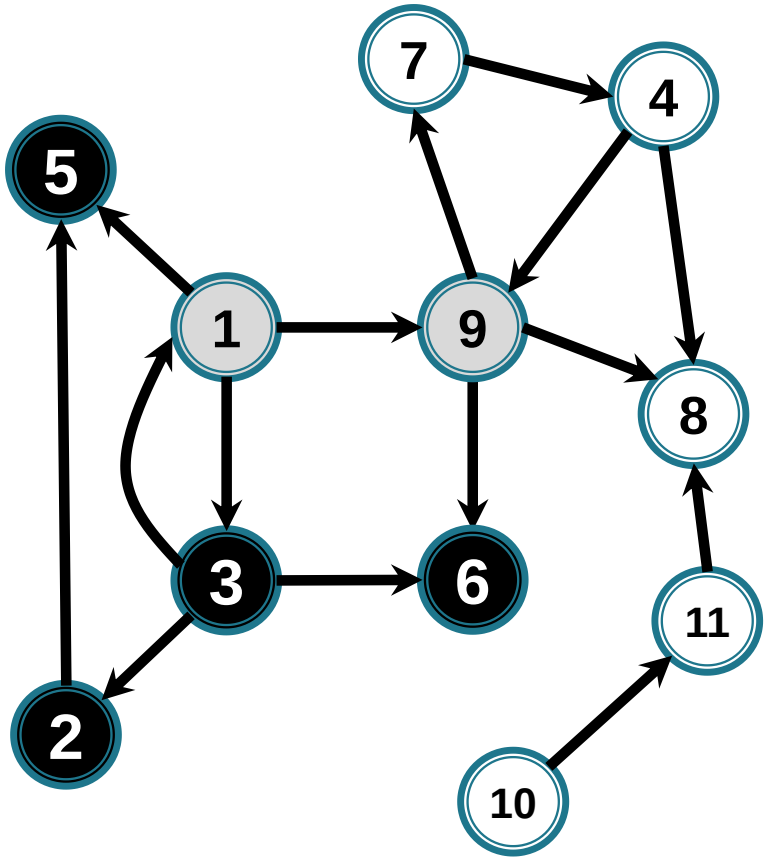


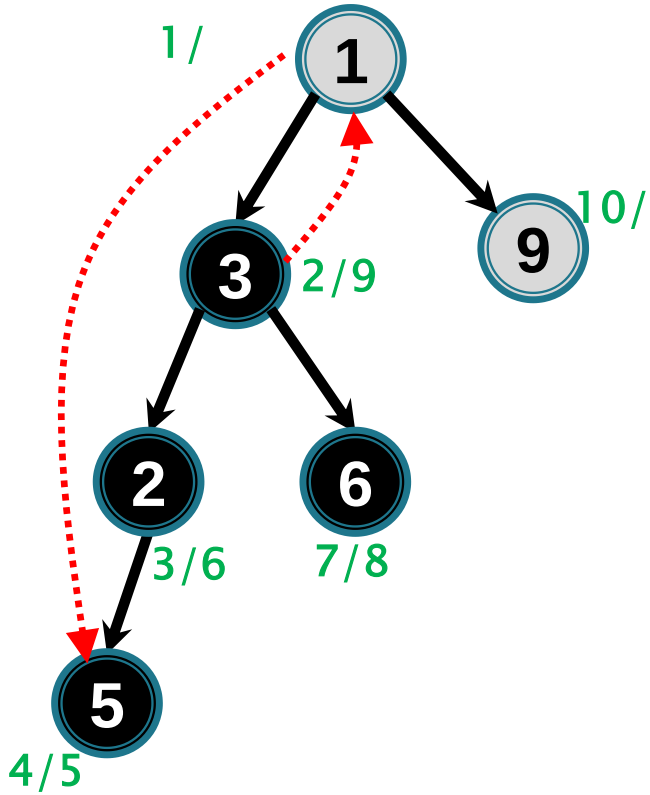
Exemplu DF graf orientat



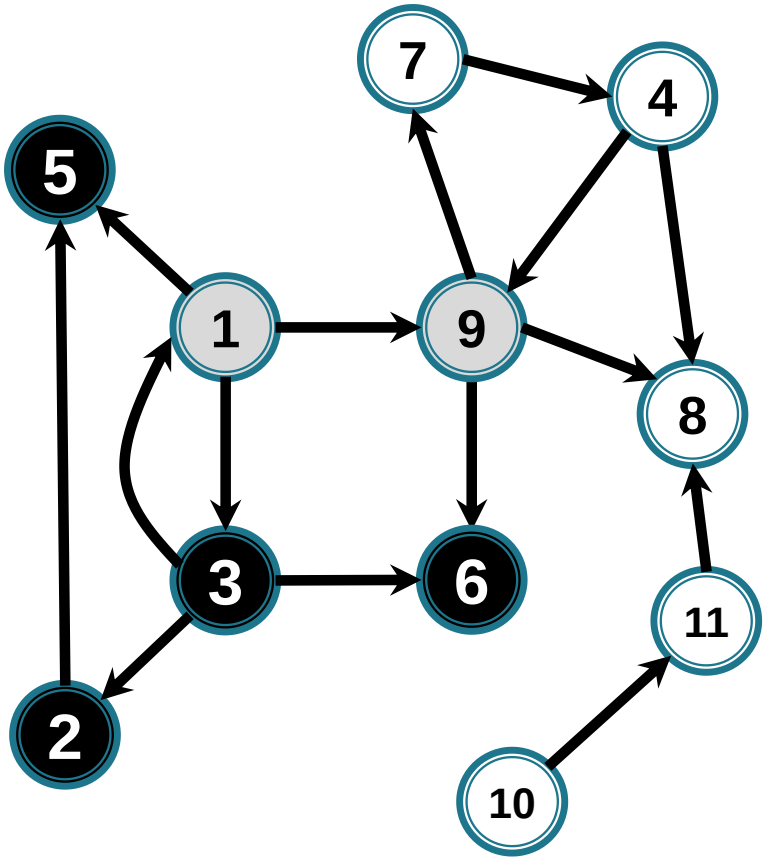


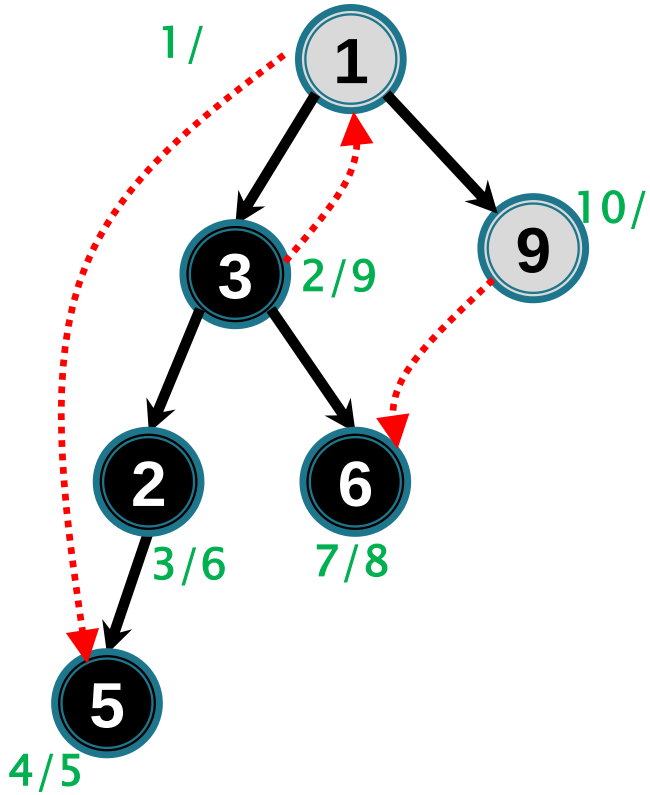
Exemplu DF graf orientat



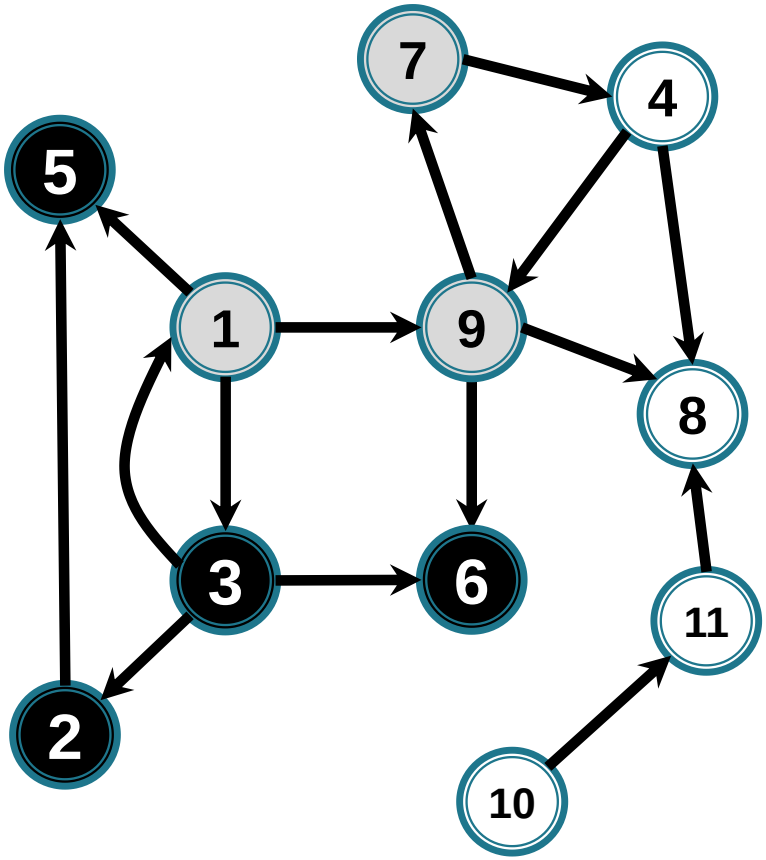


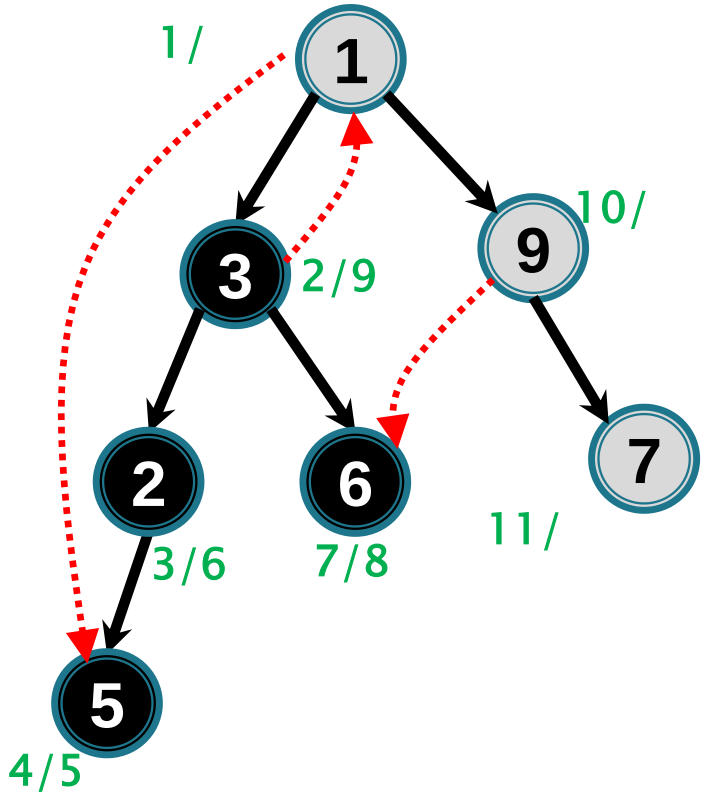
Exemplu DF graf orientat



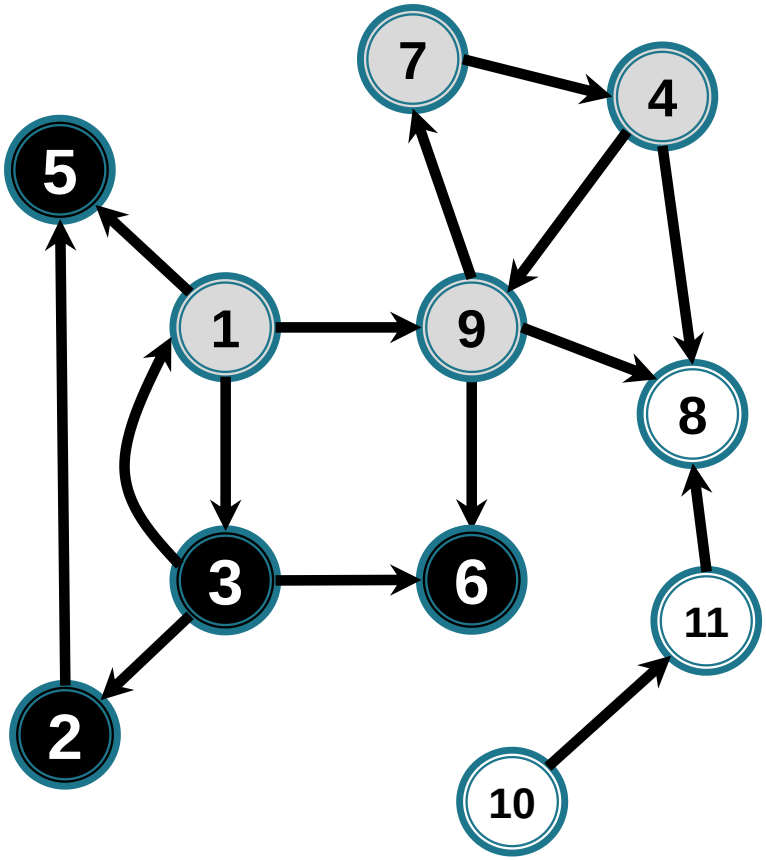


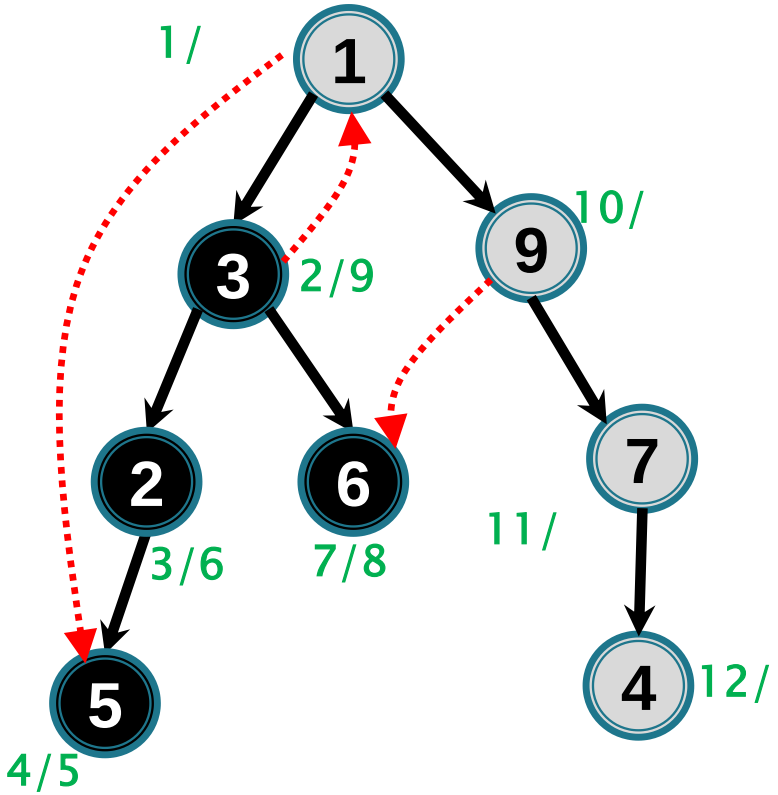
Exemplu DF graf orientat



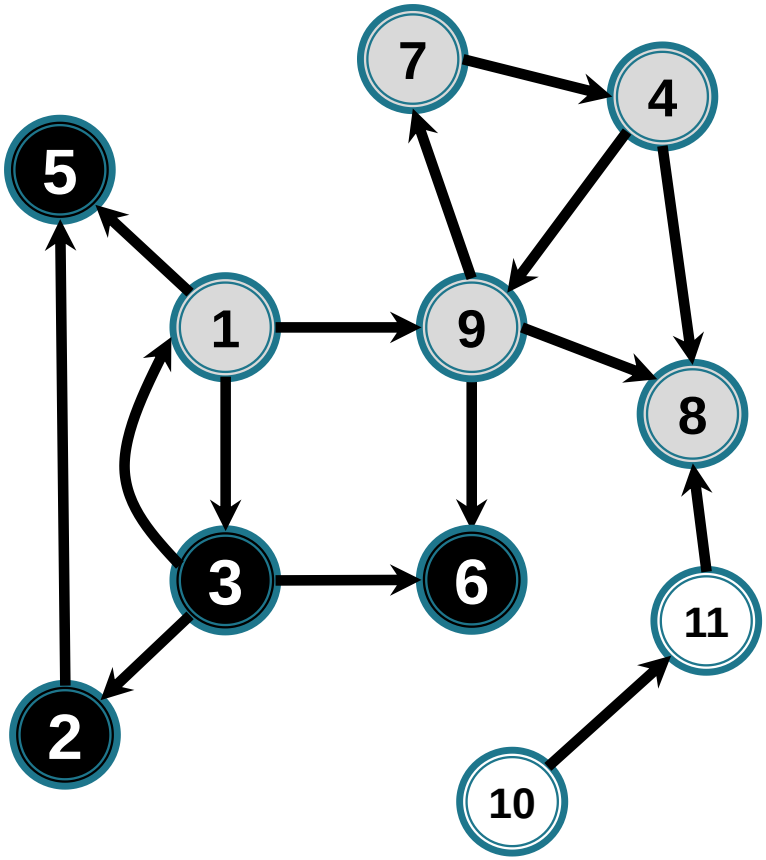


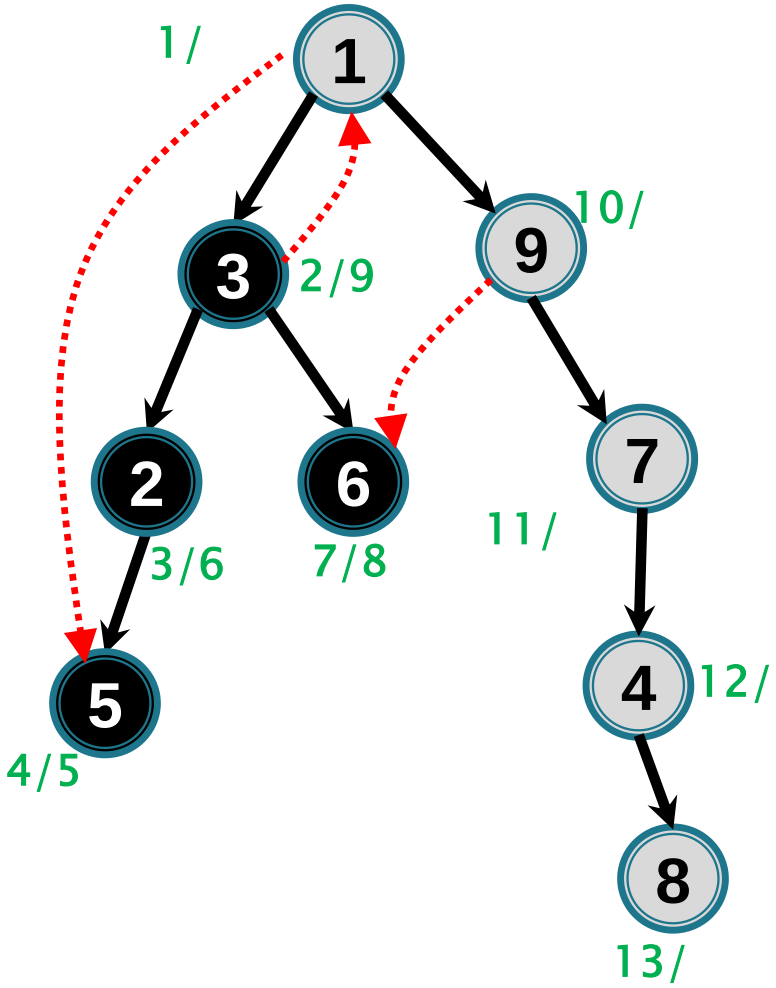
Exemplu DF graf orientat



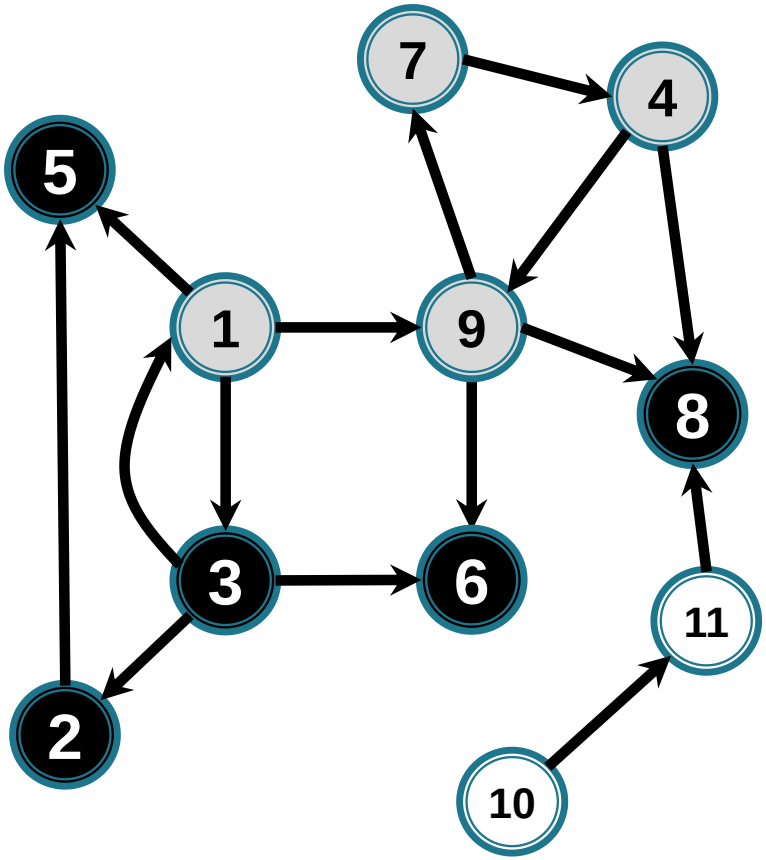


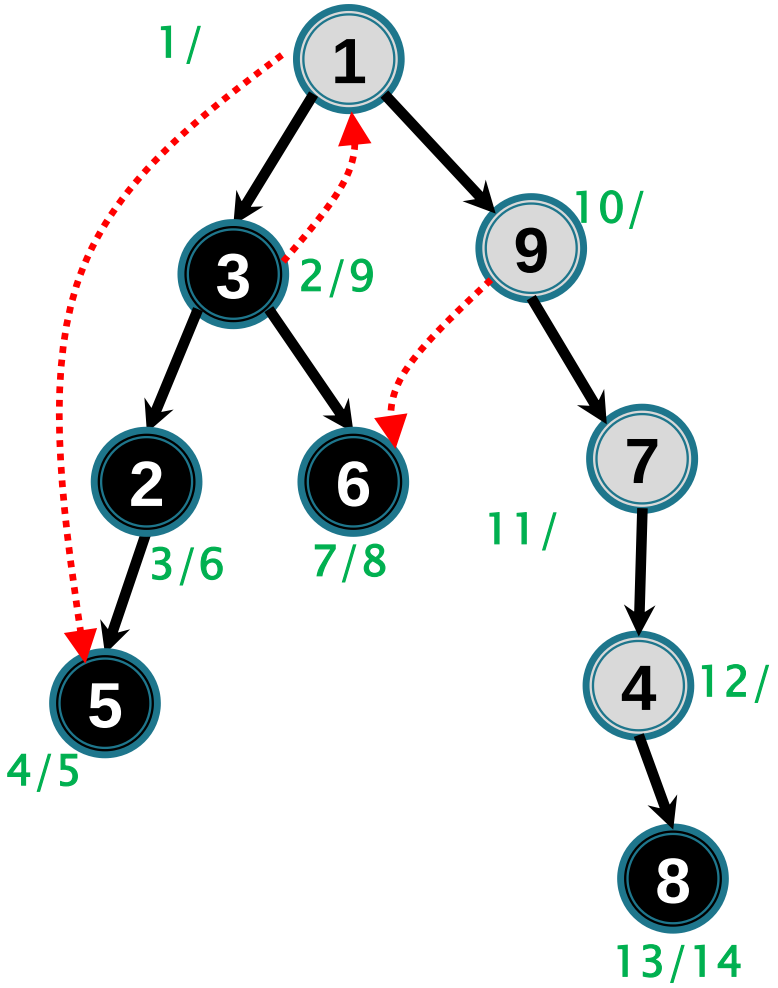
Exemplu DF graf orientat



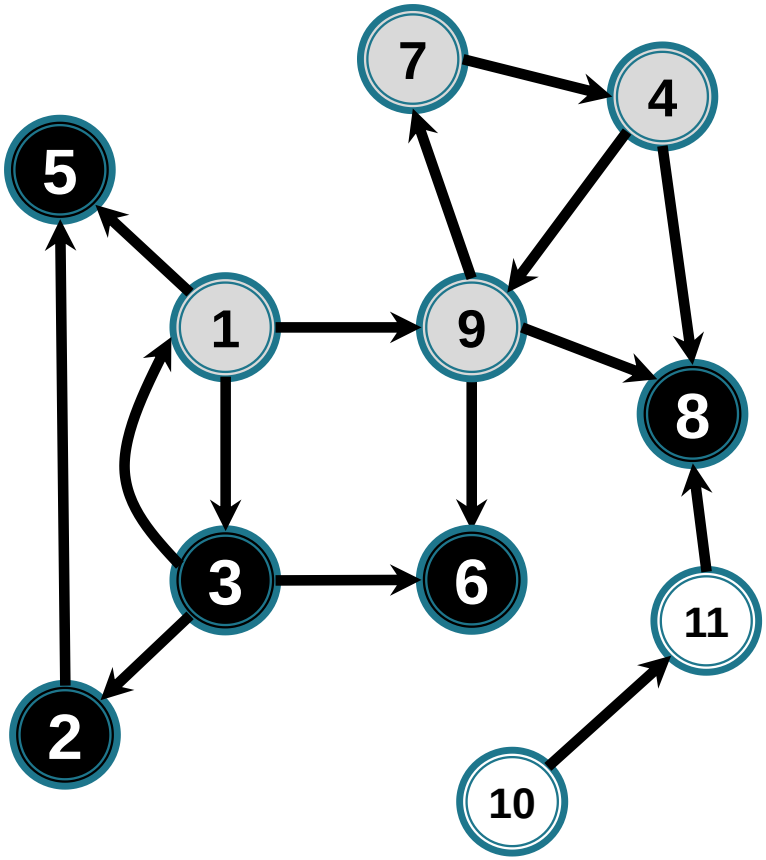


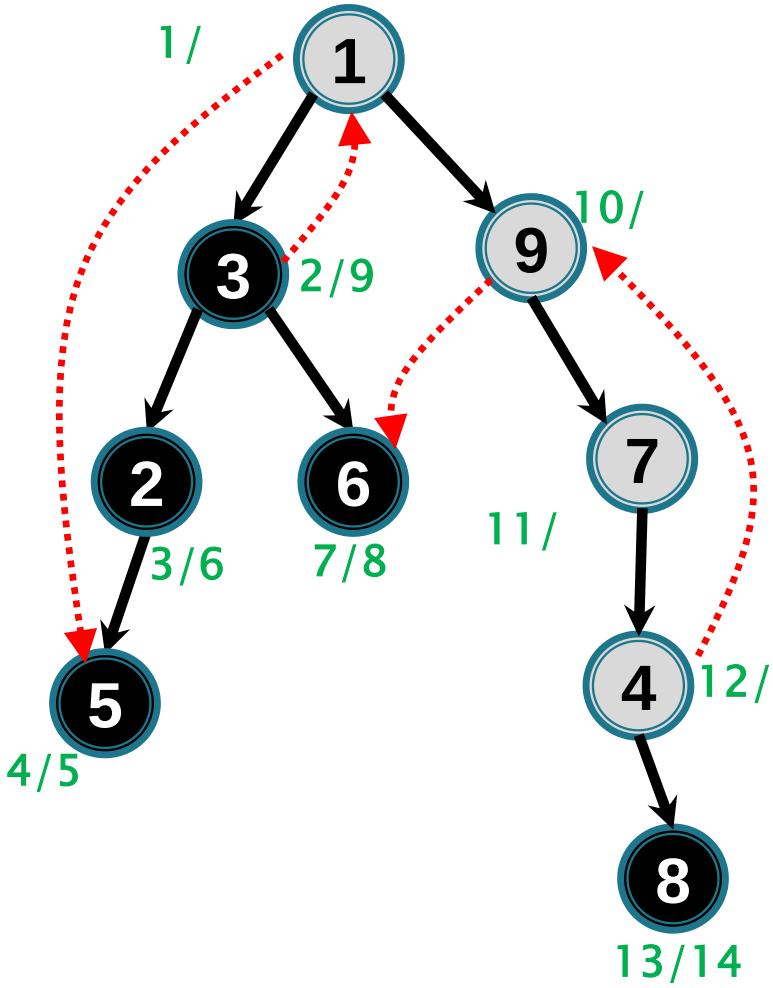
Exemplu DF graf orientat



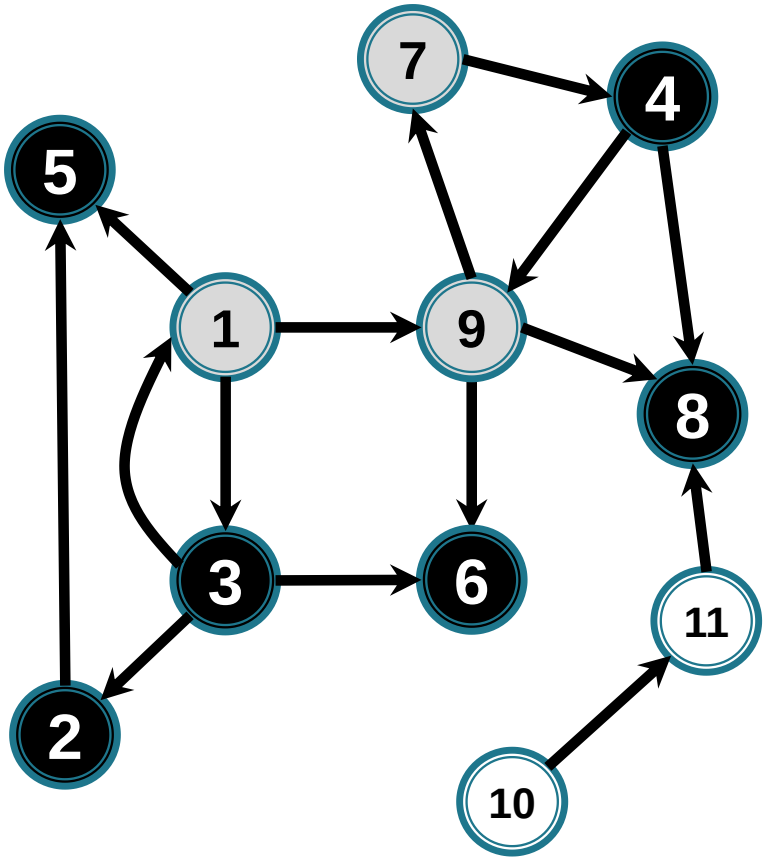


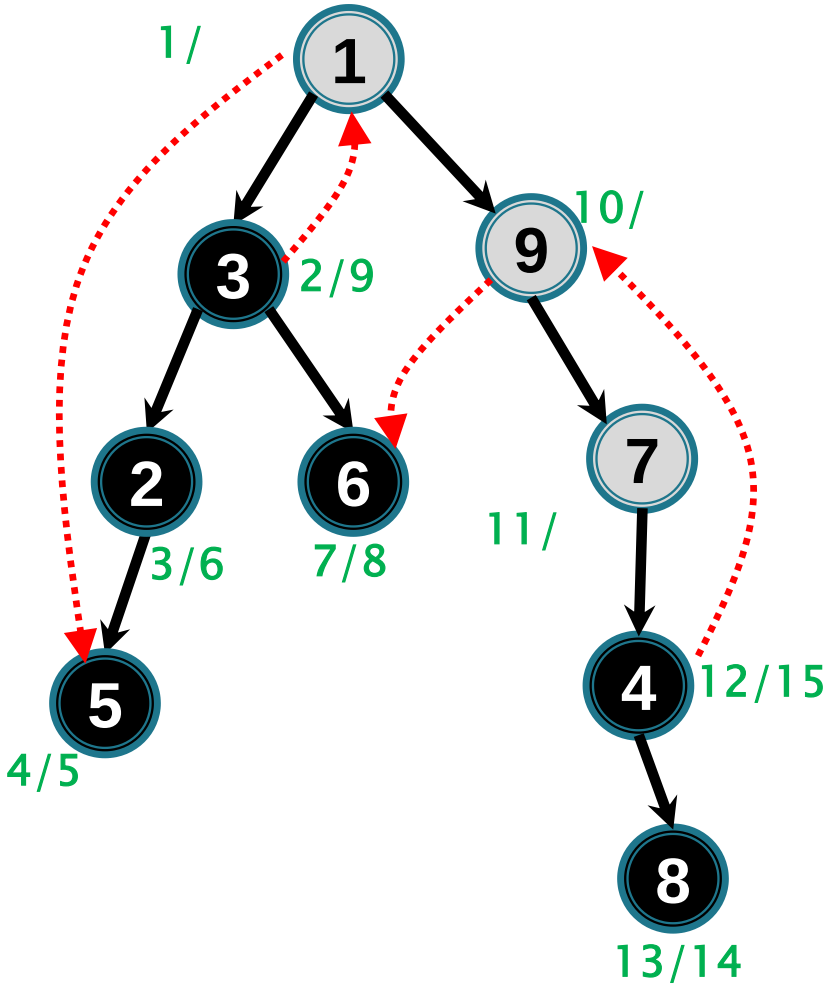
Exemplu DF graf orientat



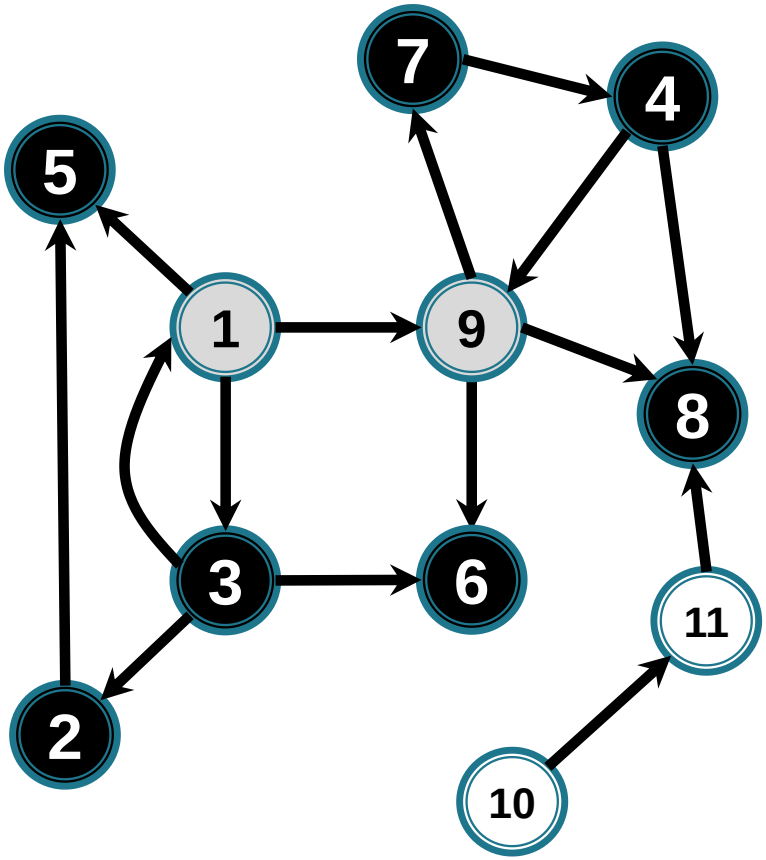


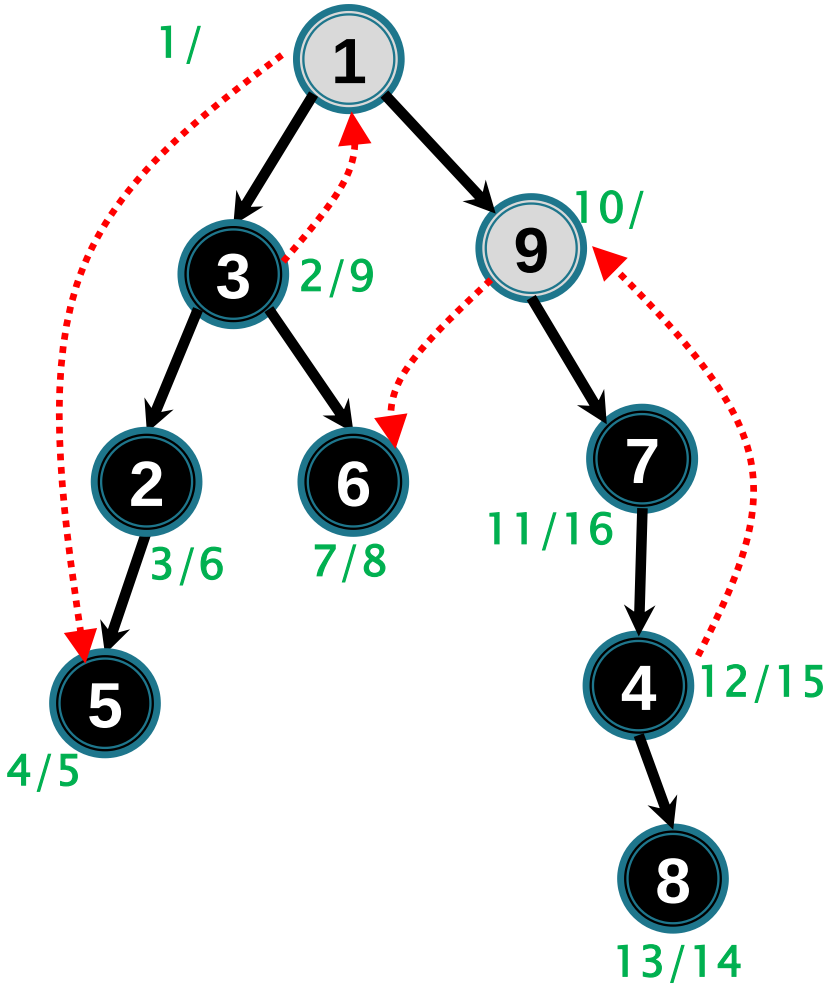
Exemplu DF graf orientat



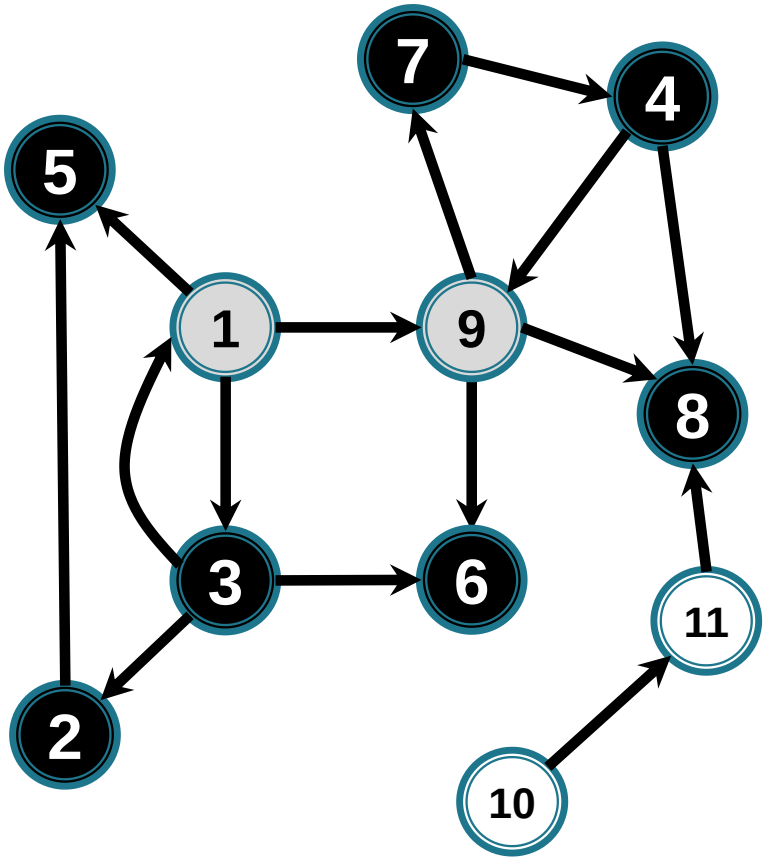


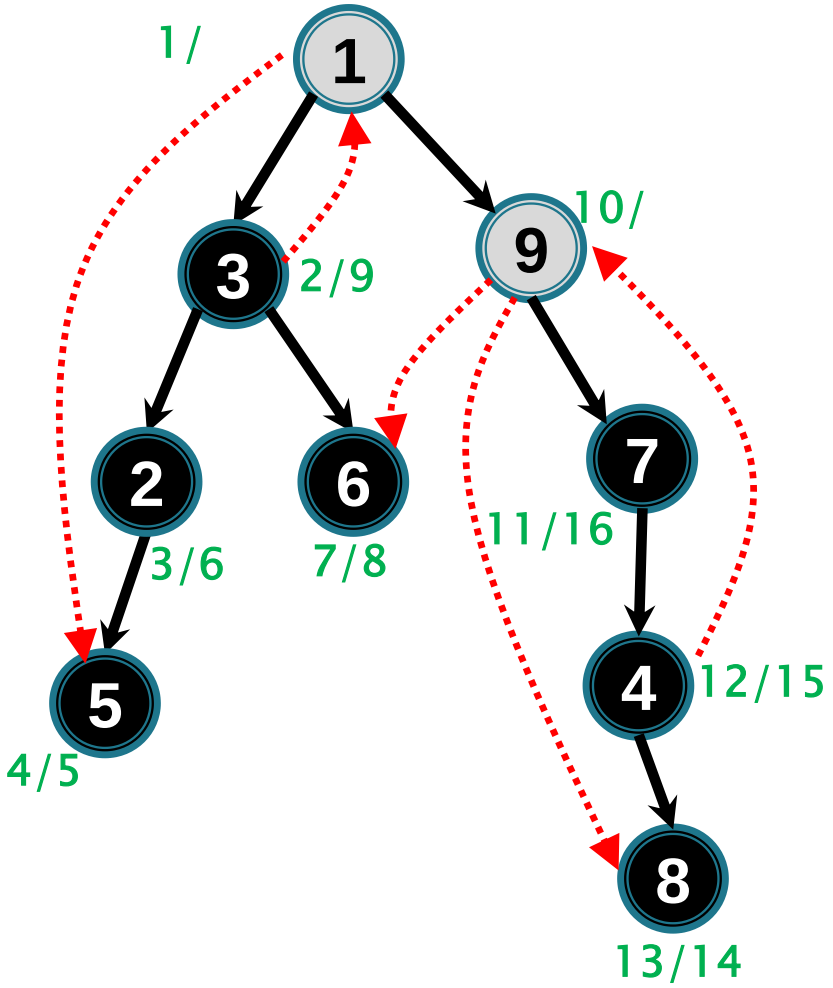
Exemplu DF graf orientat



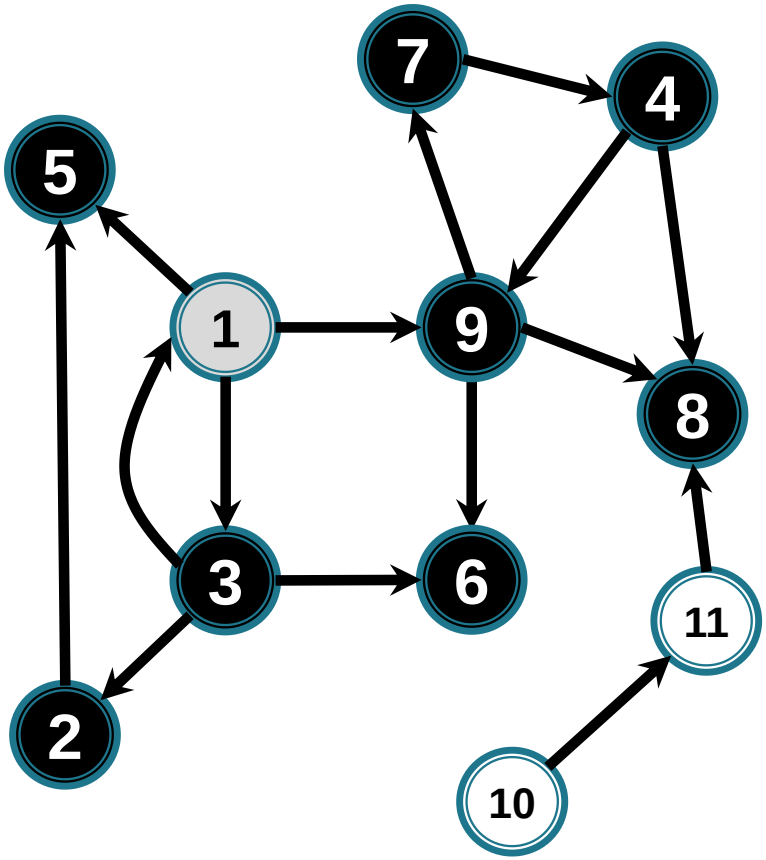


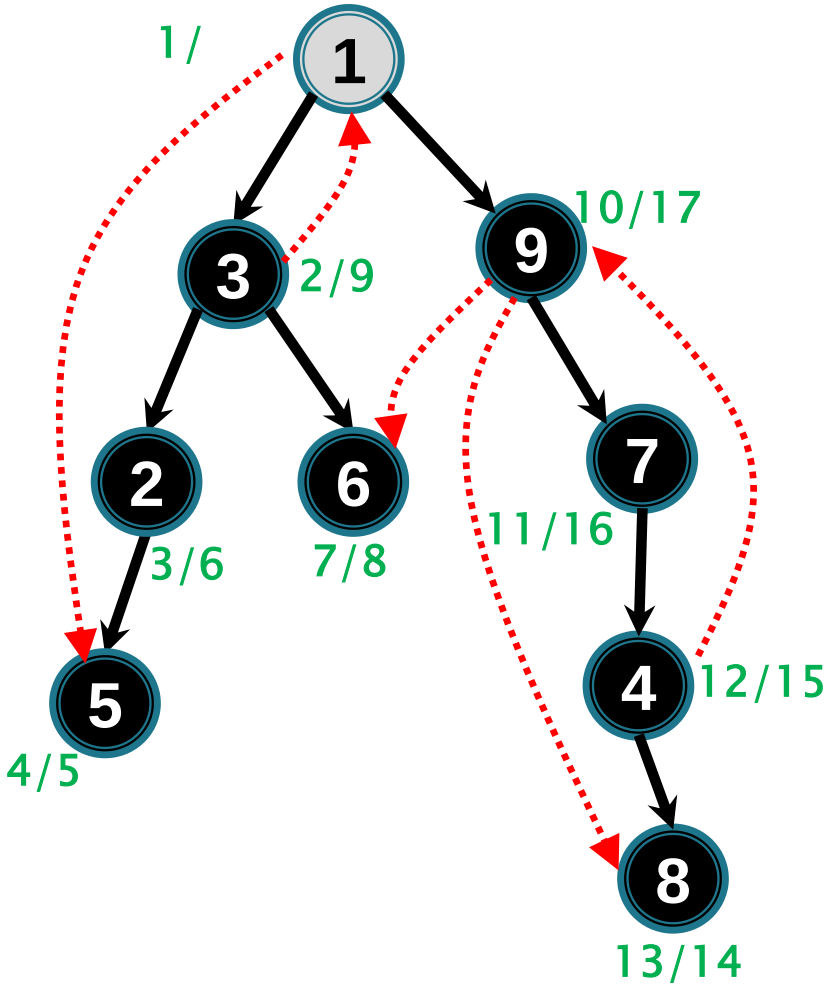
Exemplu DF graf orientat



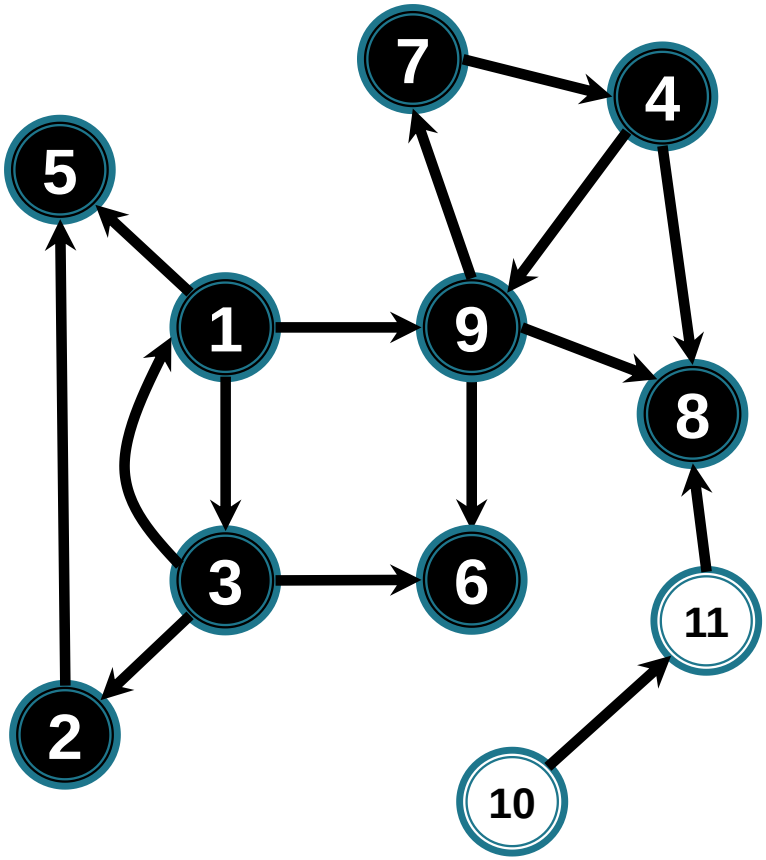


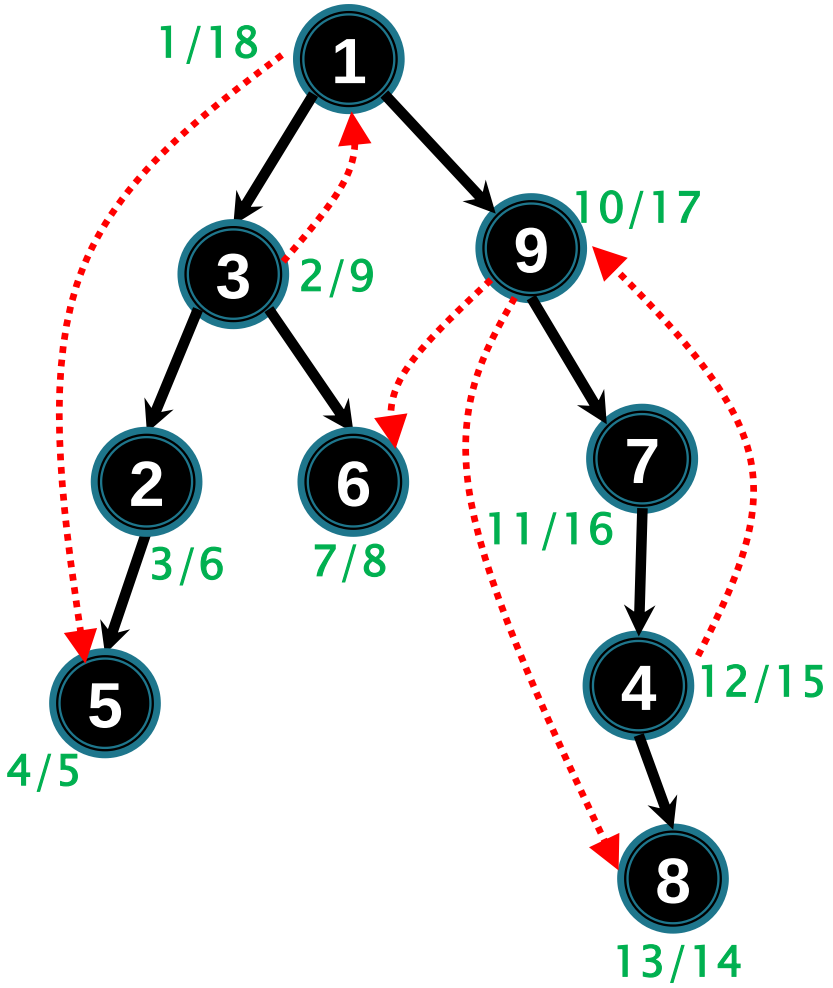
Exemplu DF graf orientat



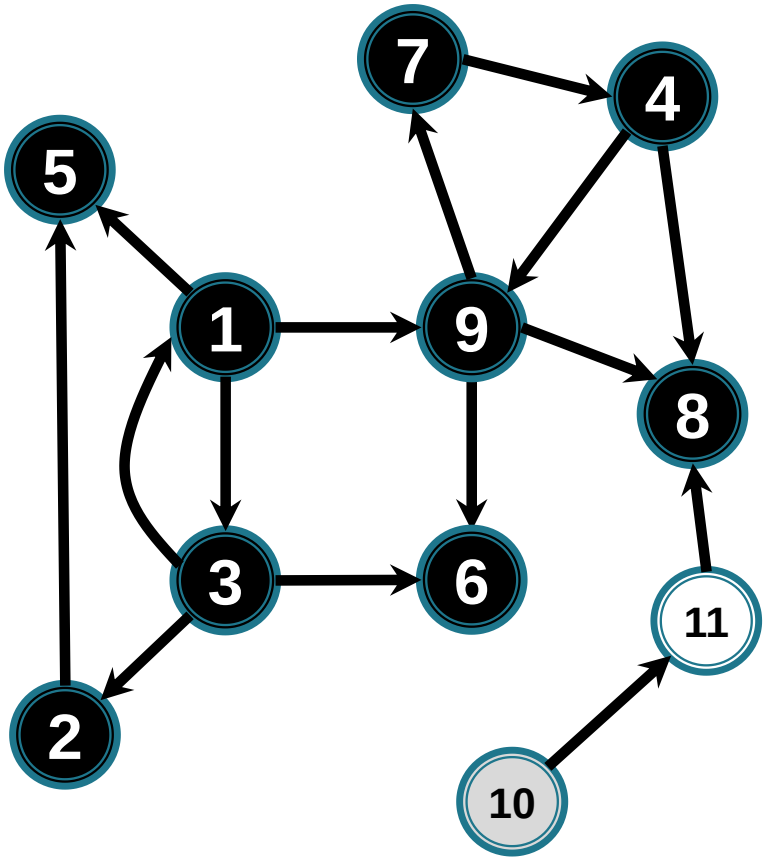


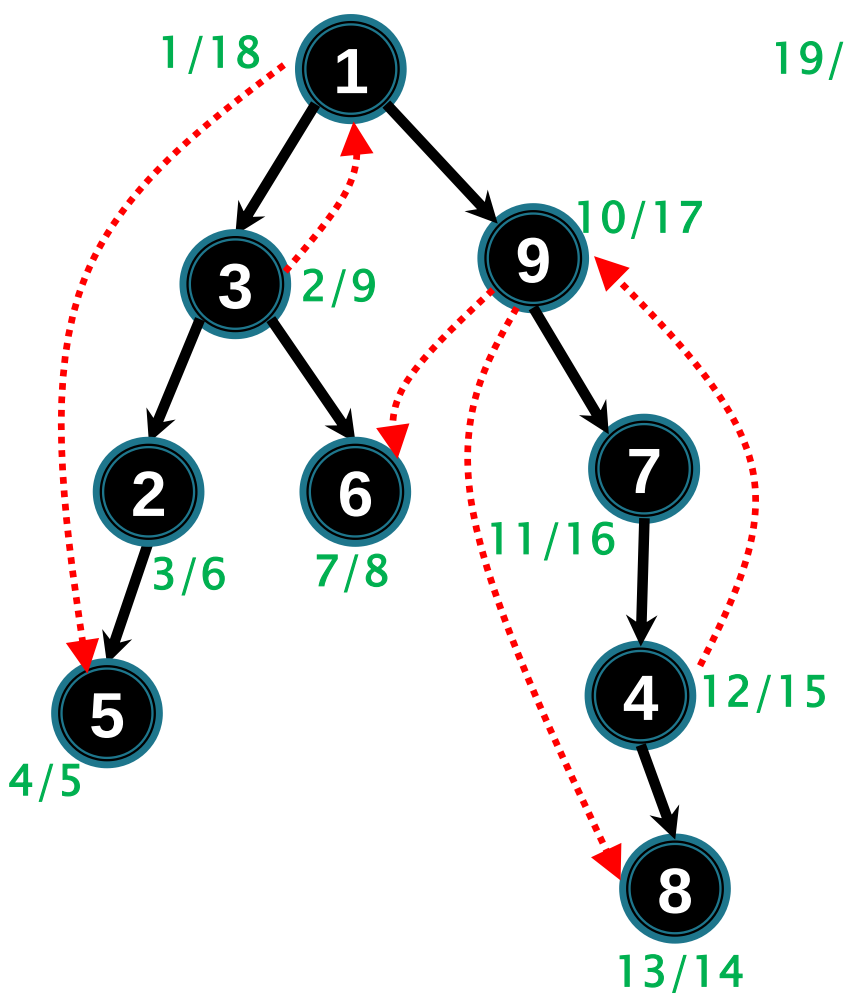
Exemplu DF graf orientat



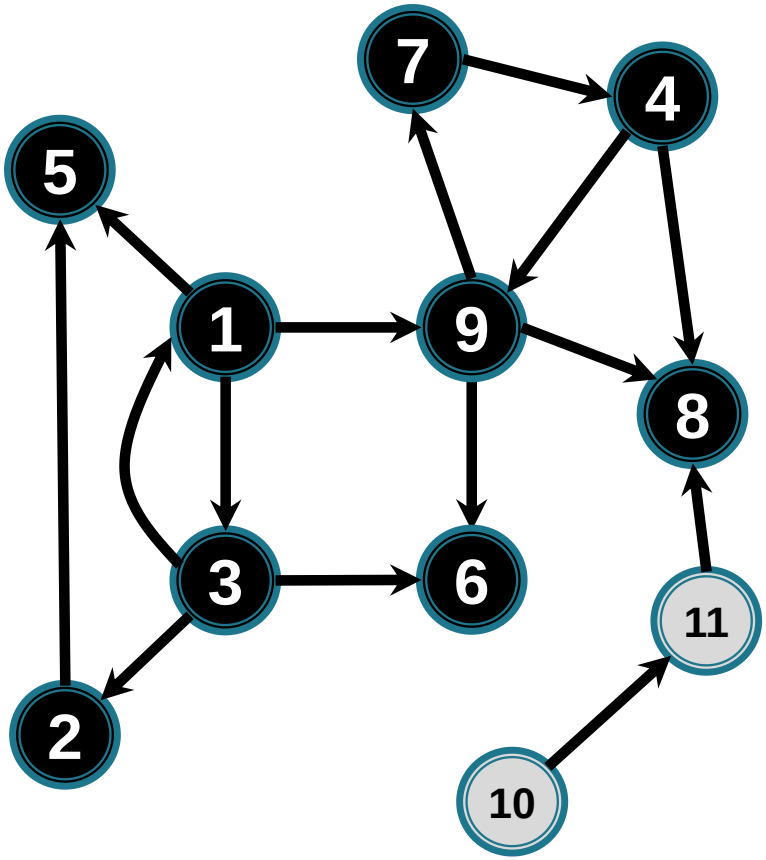


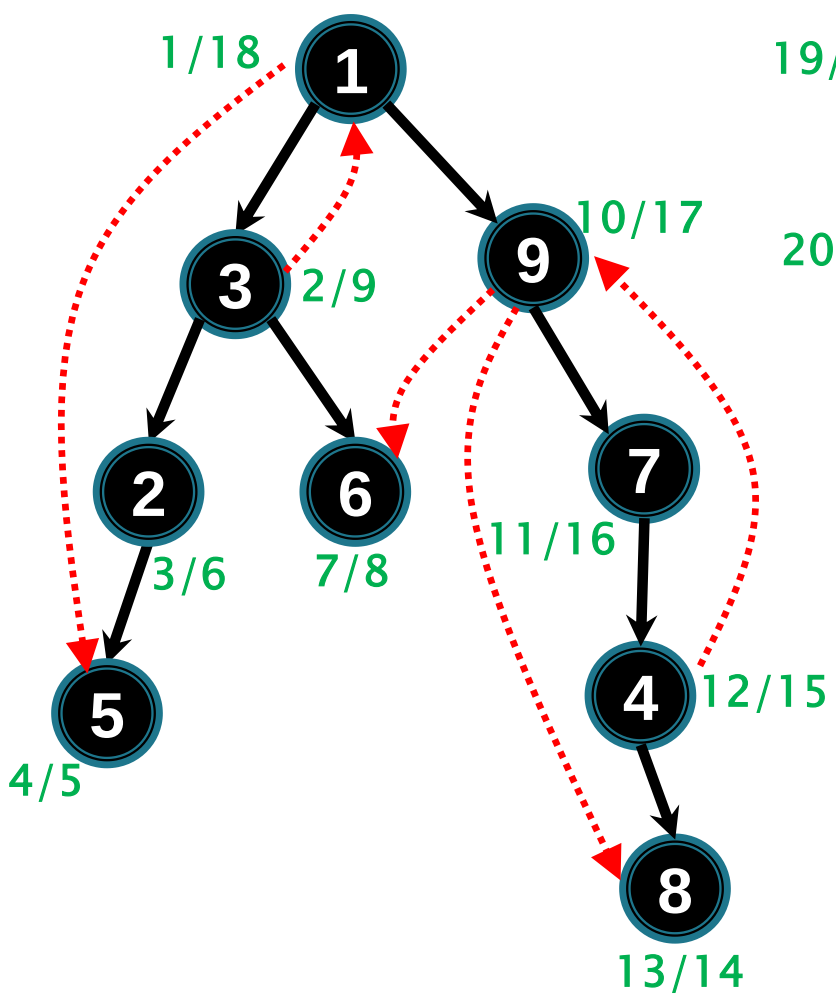
Exemplu DF graf orientat



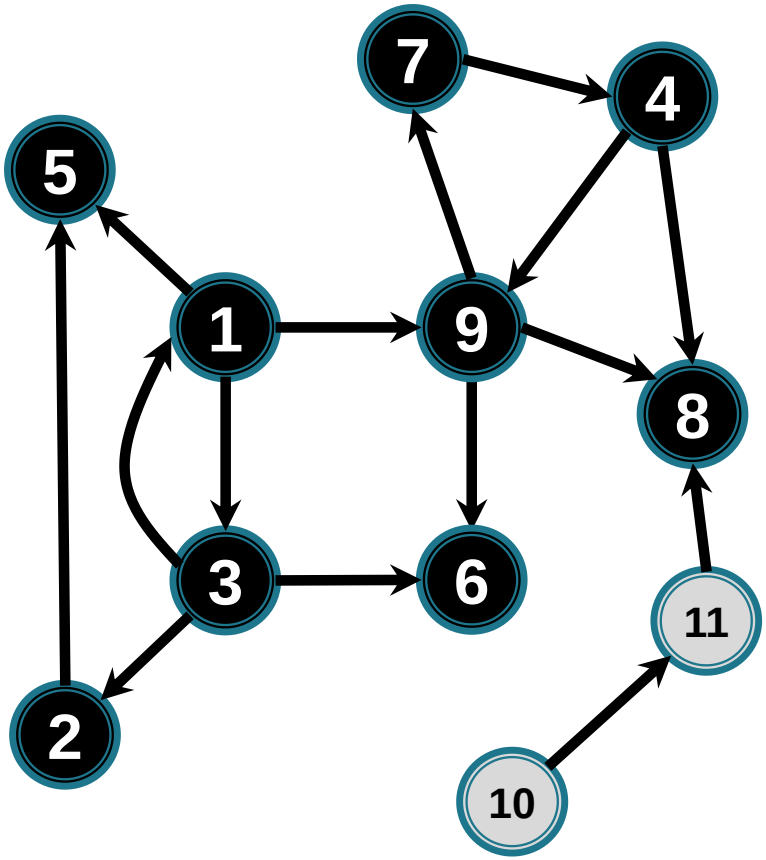


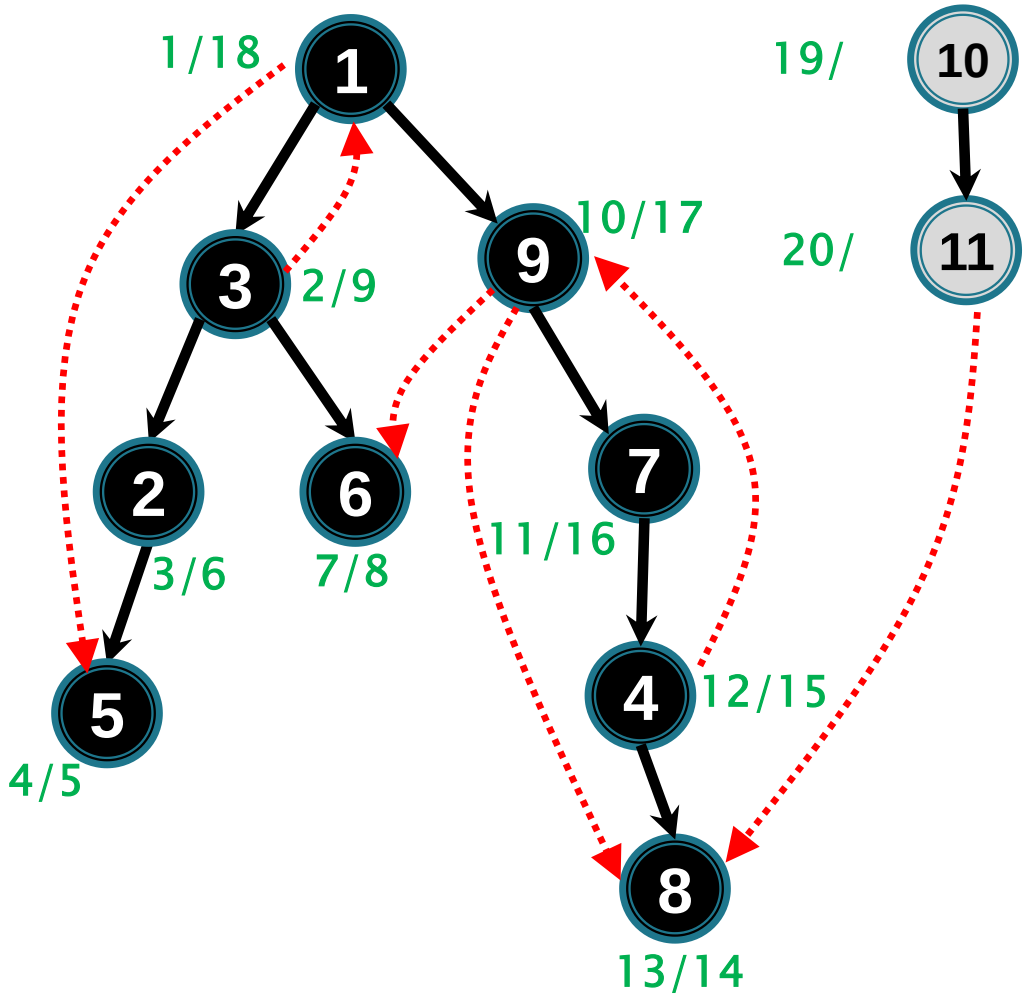
Exemplu DF graf orientat



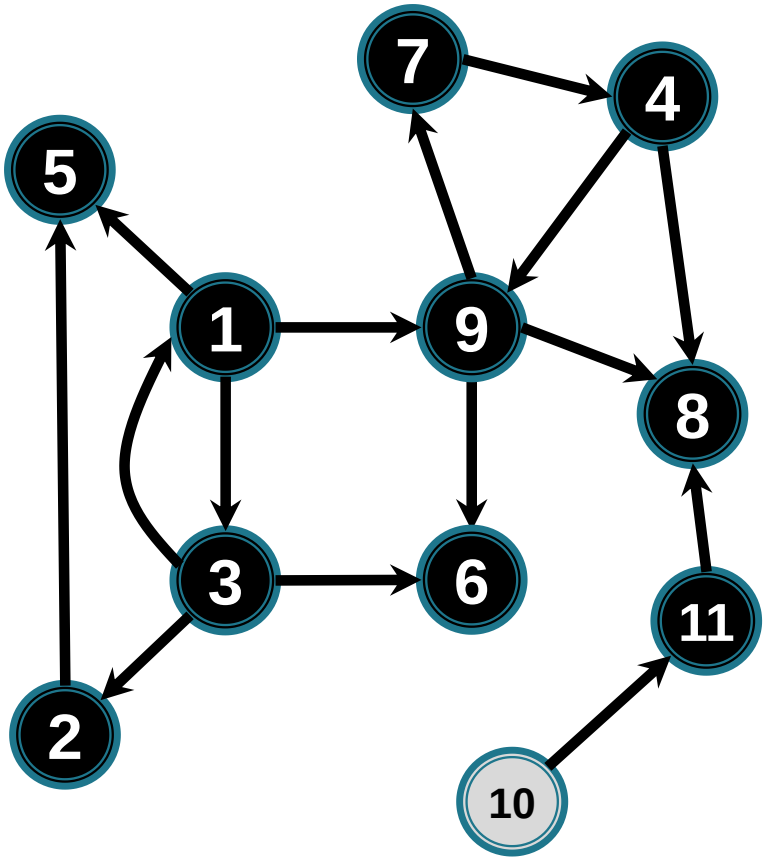


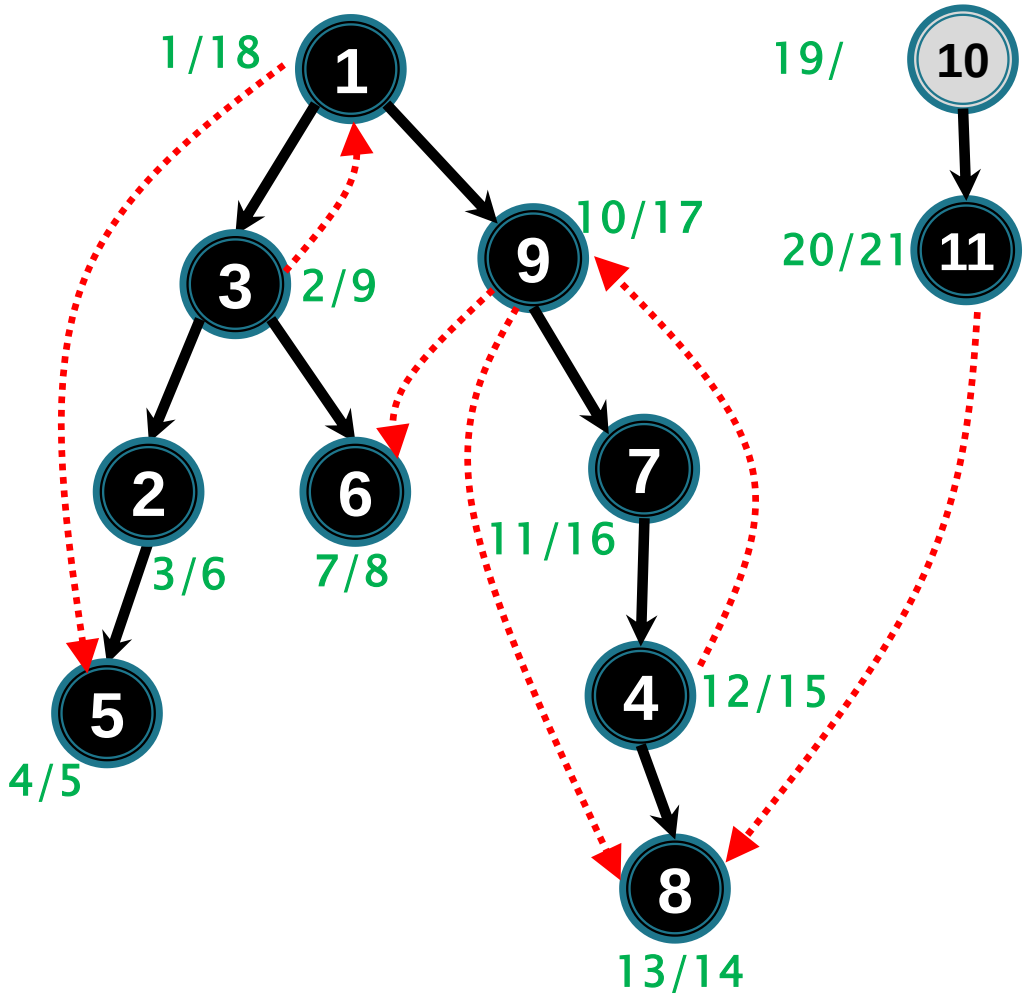
Exemplu DF graf orientat



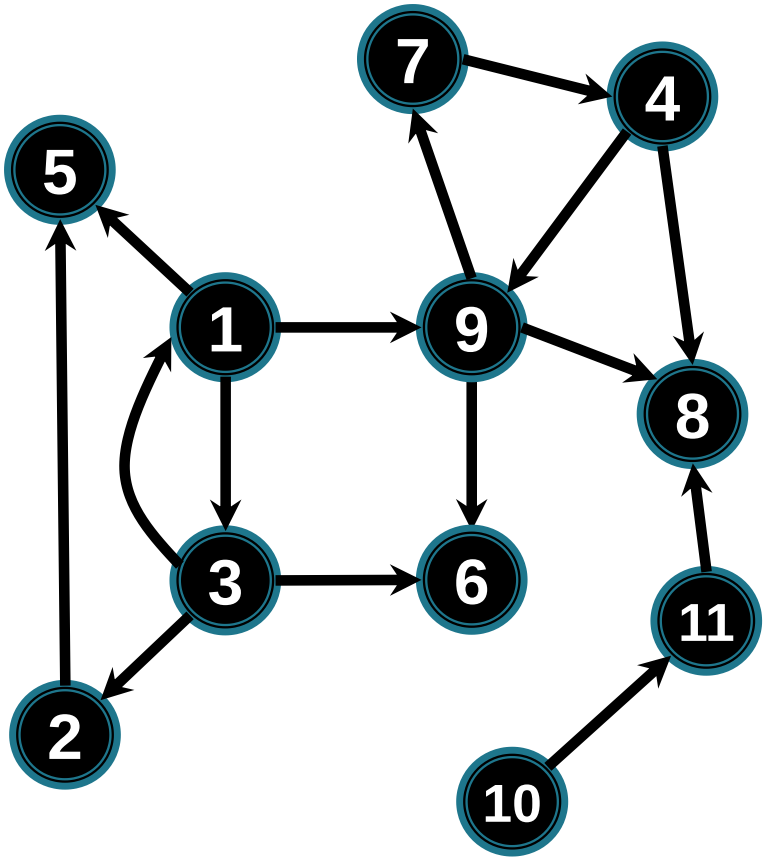


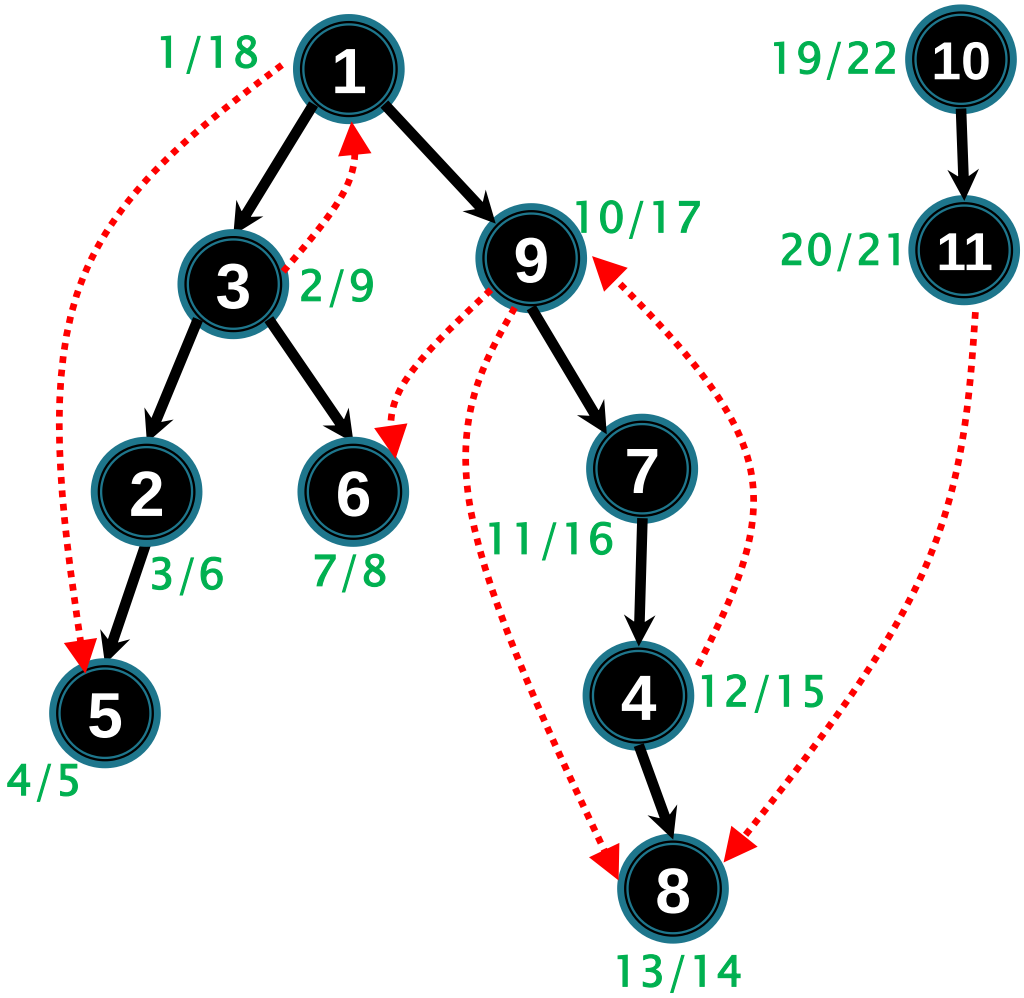
Exemplu DF graf orientat





Exemplu DF graf orientat



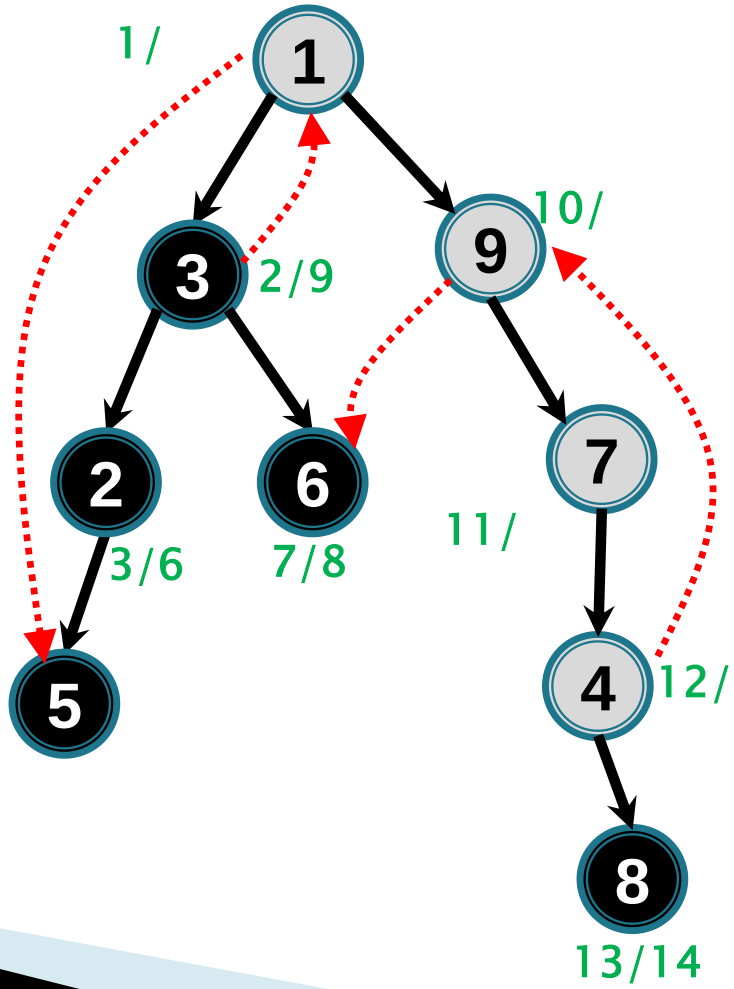


Proprietăți DF

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 şi 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

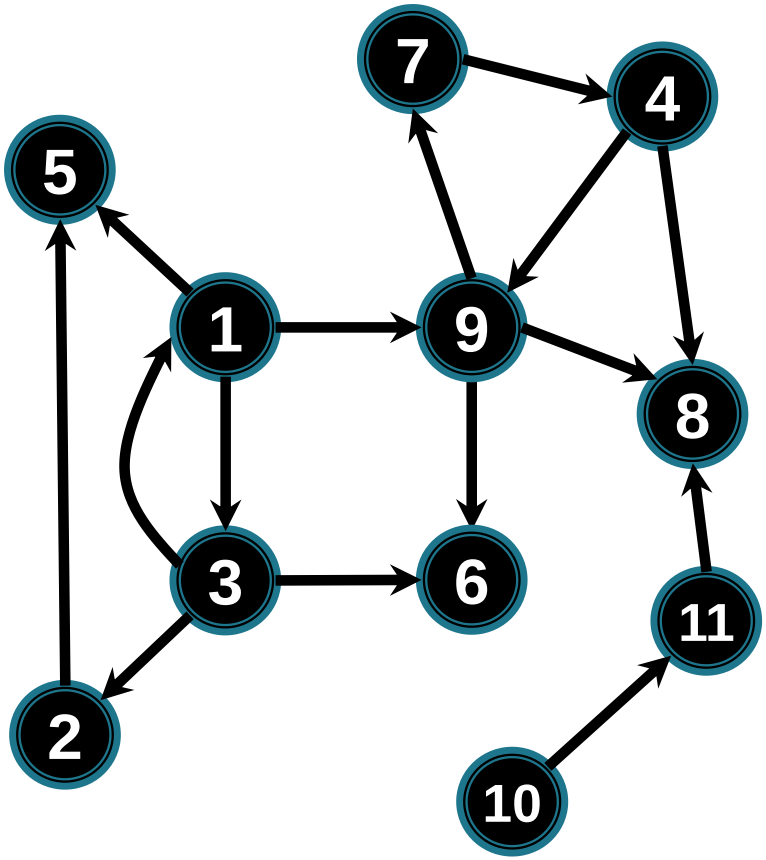
În orice căutare în adâncime a unui graf (orientat sau neorientat) G = (V, E), pentru orice două vârfuri u şi v cu

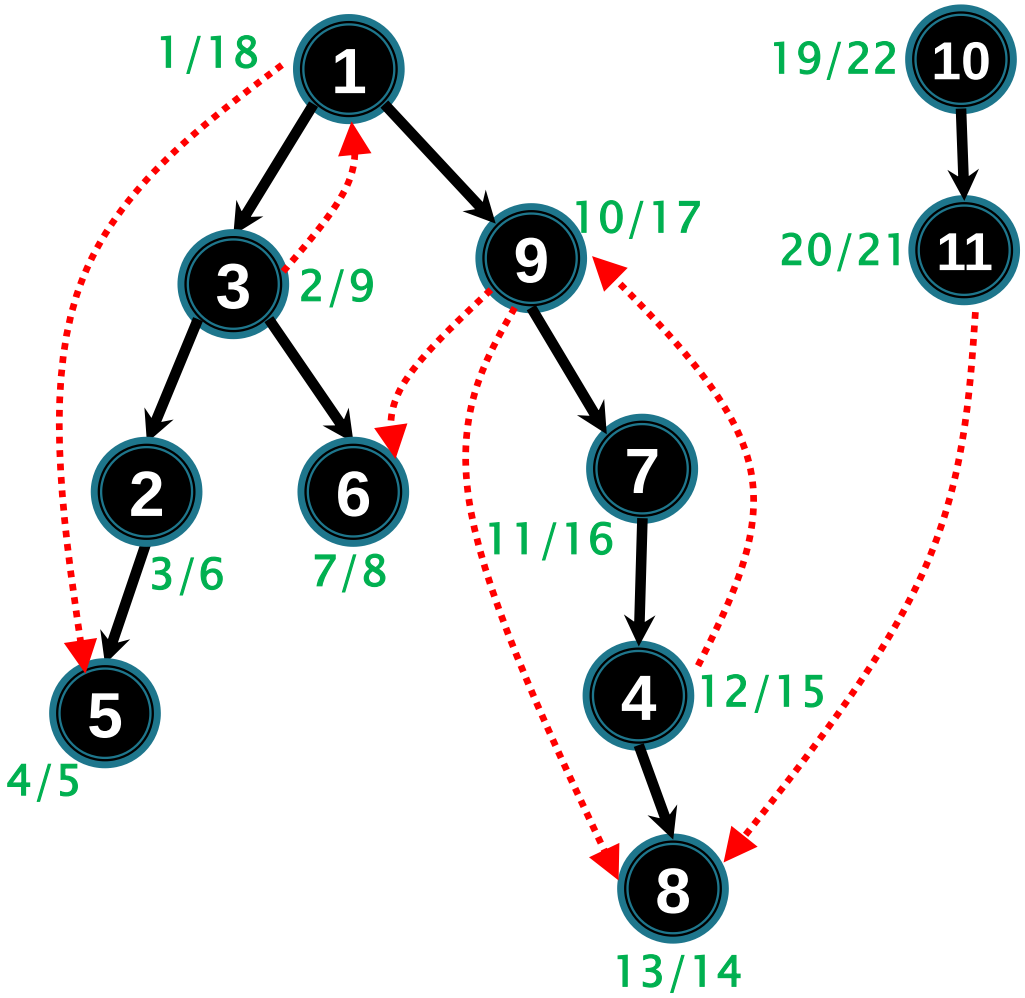
```
desc[u] < desc[v]
```

exact una din următoarele condiții este adevărată:

.

.


7
4
5
1
9
8
3
6
11
2
10


1/18
1
19/22
10
10/17
3
2/9
9
20/21
11
2
6
7
3/6
7/8
11/16
5
4
12/15
4/5
8
13/14

Proprietăți DF

Teorema parantezelor

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

`desc[u] < desc[v]`

exact una din următoarele condiții este adevărată:

- intervalele `[desc[u], fin[u]]` şi `[desc[v], fin[v]]` sunt **disjuncte** **și u și v nu sunt unul descendentul celuilalt** în arborele/pădurea DF

- `[desc[v], fin[v]]` ⊂ `[desc[u], fin[u]]`,

caz în care v este un descendent al lui u în arborele/pădurea DF

Proprietăți DF

Idee de demonstrație

Avem

$$desc[u] < desc[v]$$

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

$$[desc[v], fin[v]] \subset [desc[u], fin[u]]$$

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)

⇔

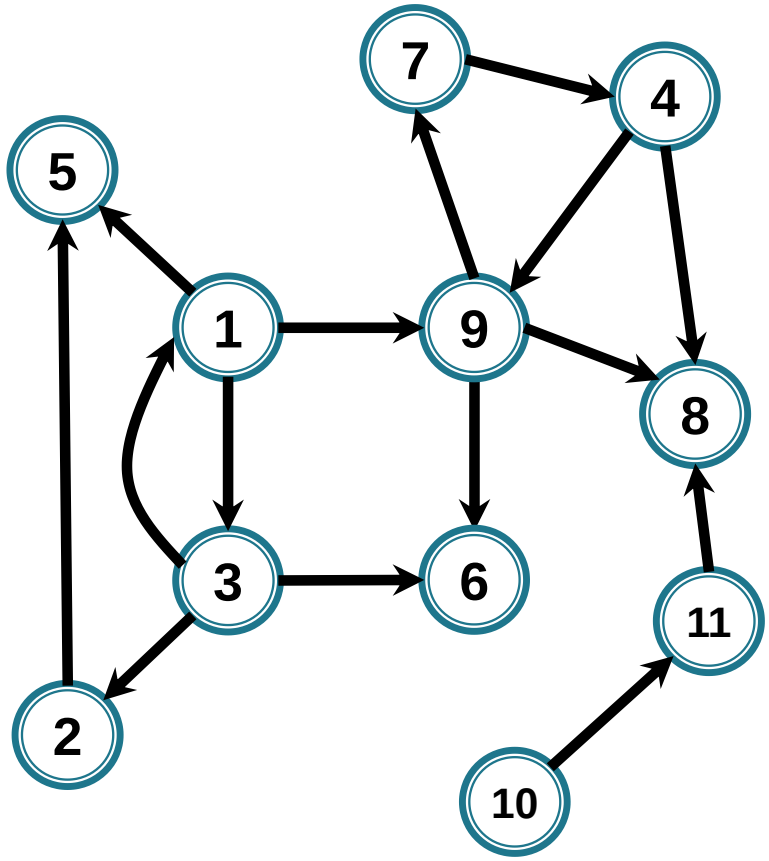
vârful v este descoperit în timp ce u este gri (este încă în explorare)

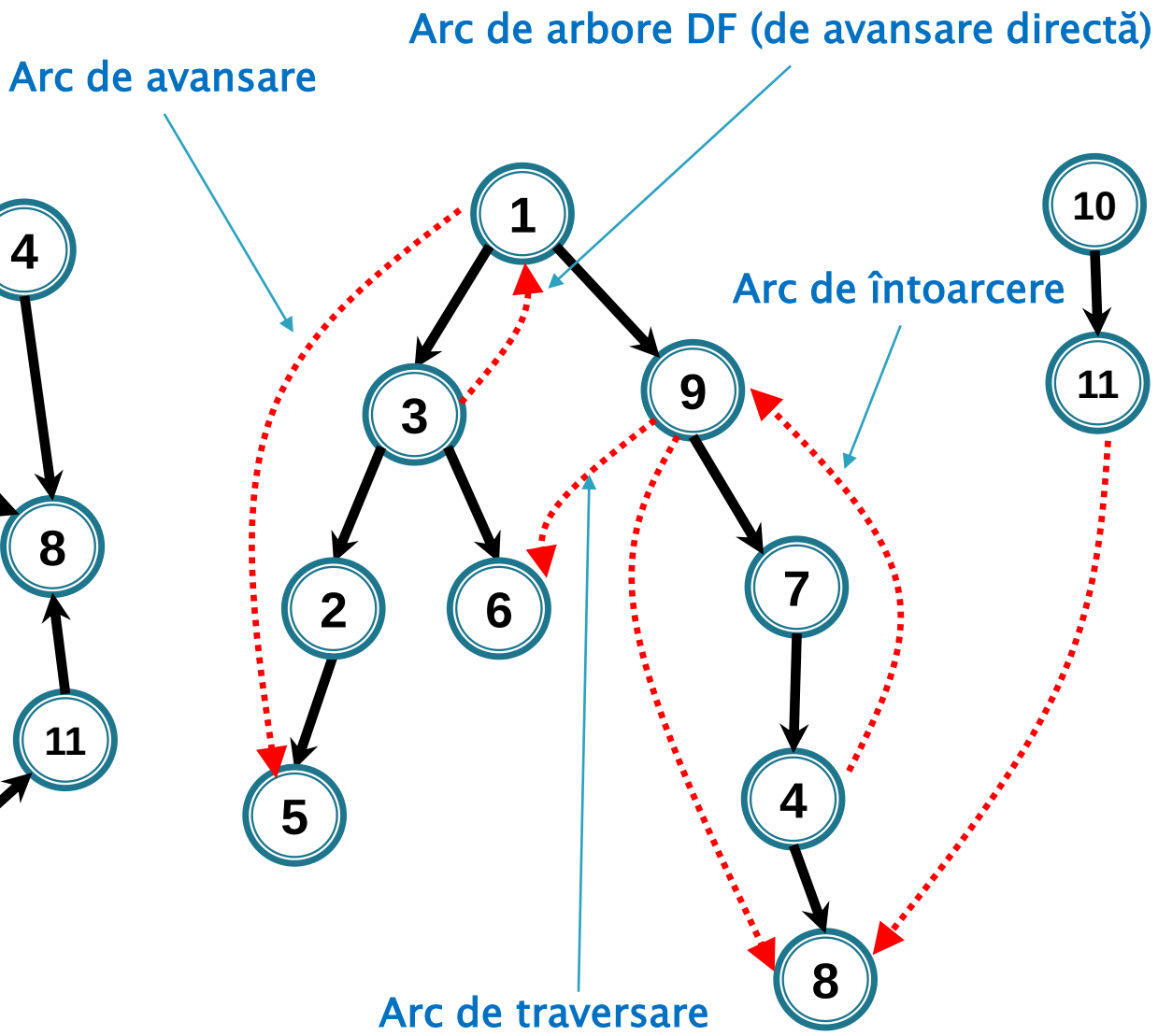
⇔

`[desc[v], fin[v]] ⊂ [desc[u], fin[u]]`

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 descendent al lui u care nu este însă fiu al lui

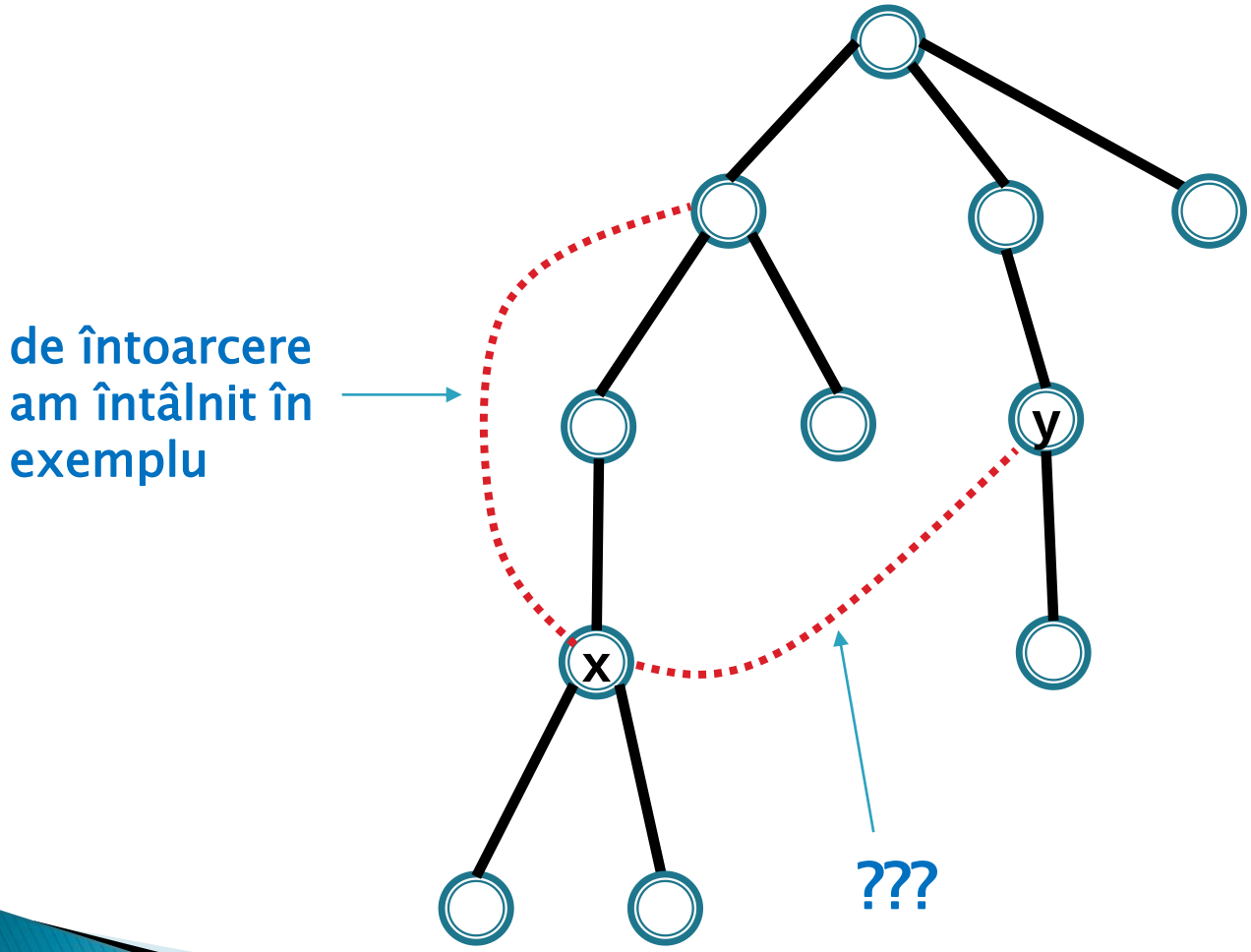
- **de traversare (transversale) :**

  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

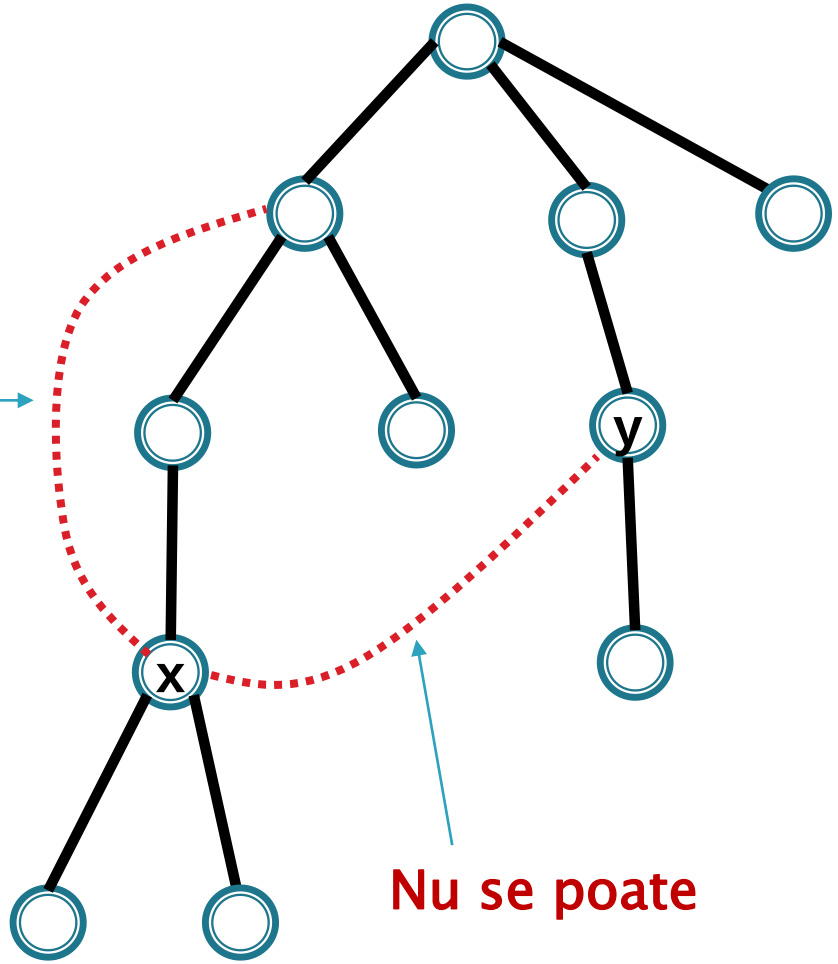
Proprietăți DF

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



Proprietăți DF

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



de întoarcere am întâlnit în exemplu

Nu se poate

Orice vecin al lui x este vizitat până cand x este finalizat

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

Proprietăți DF

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

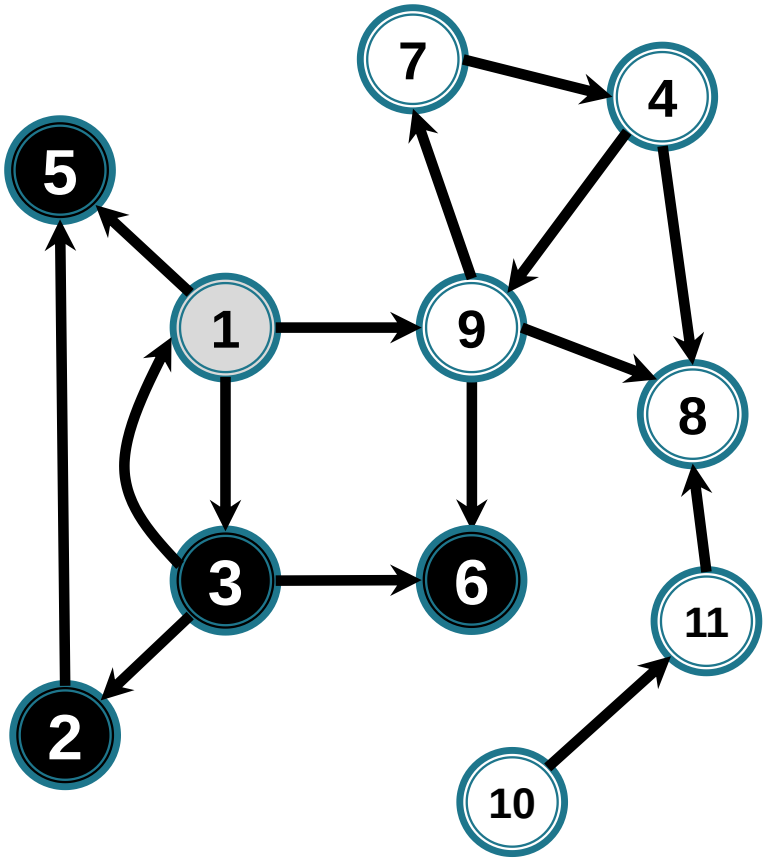
Cum se determină tipul unui arc în parcurgerea DF?

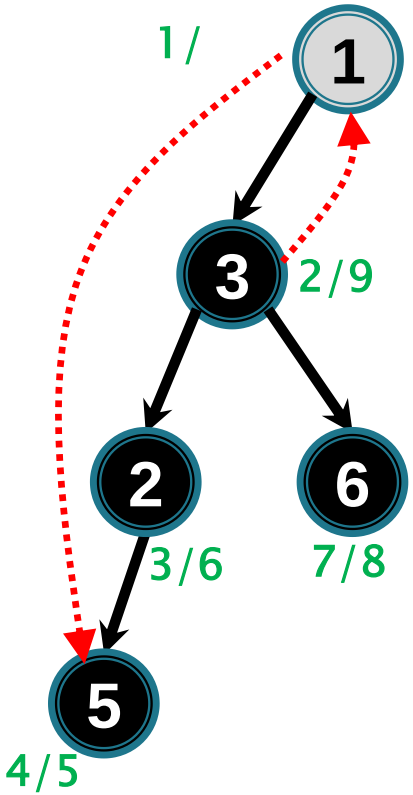
Un arc (u,v) este detectat ca fiind

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

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

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

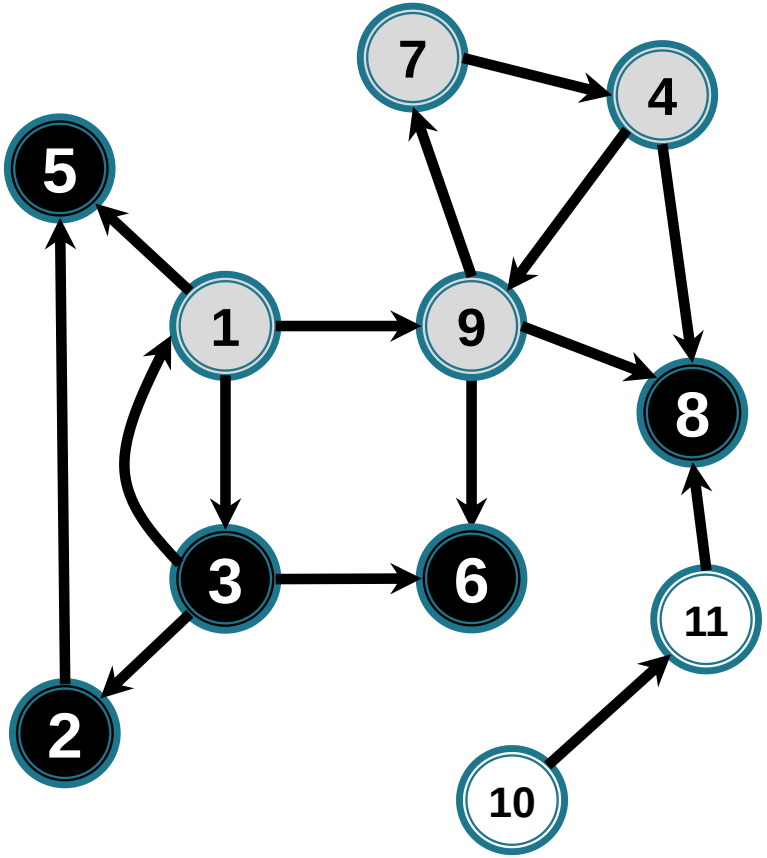


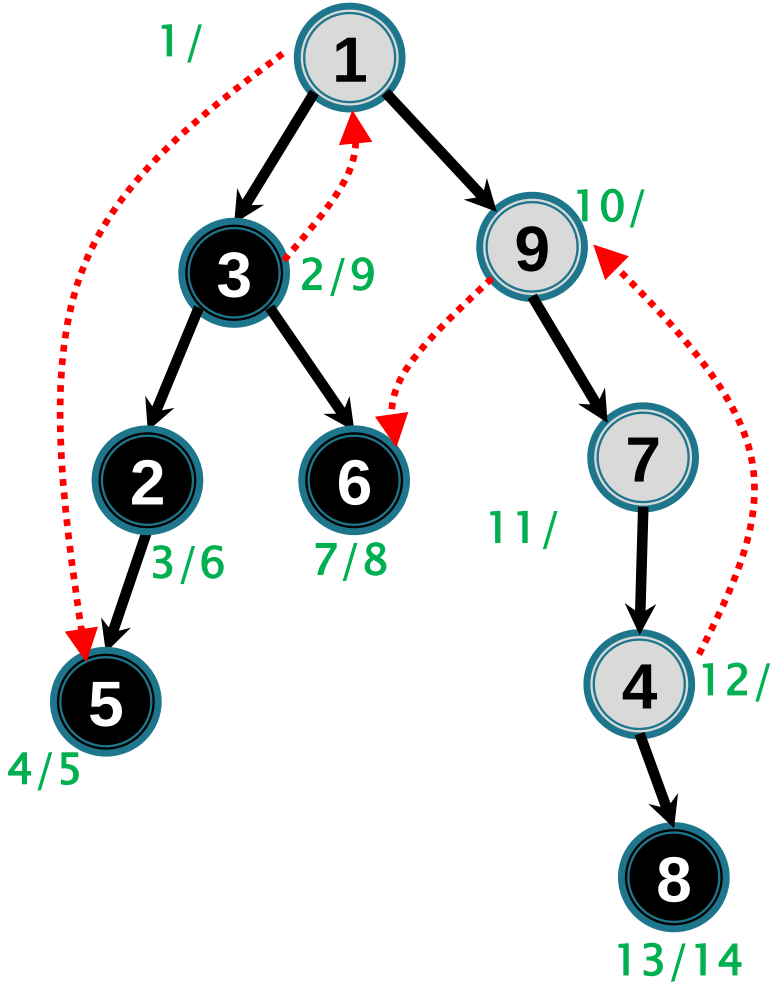


- **de întoarcere:**

`desc[v] < desc[u] < fin[u] < fin[v]` ⇔

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

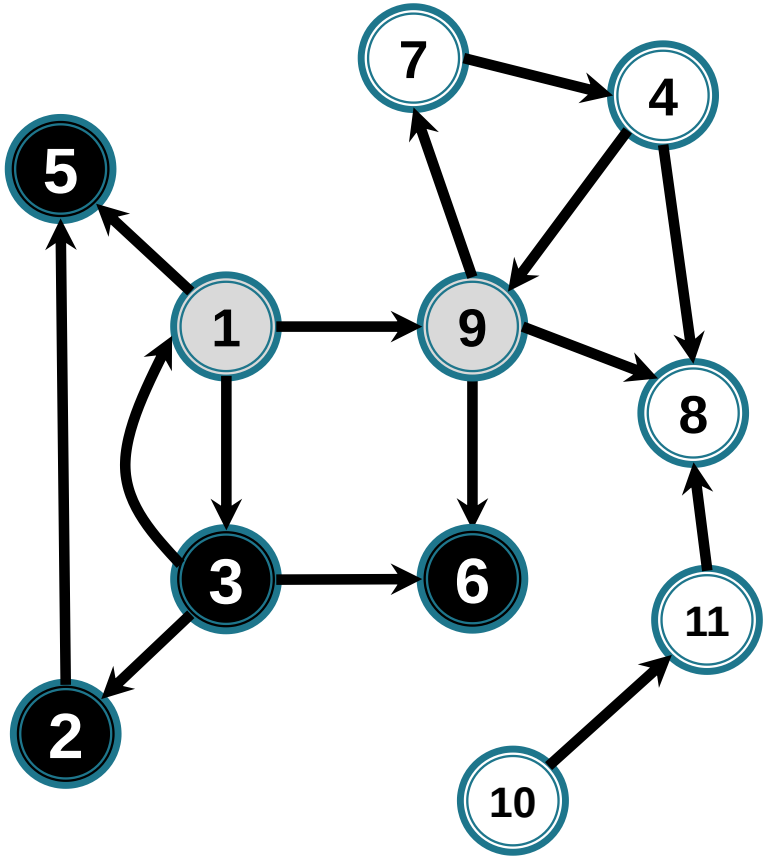


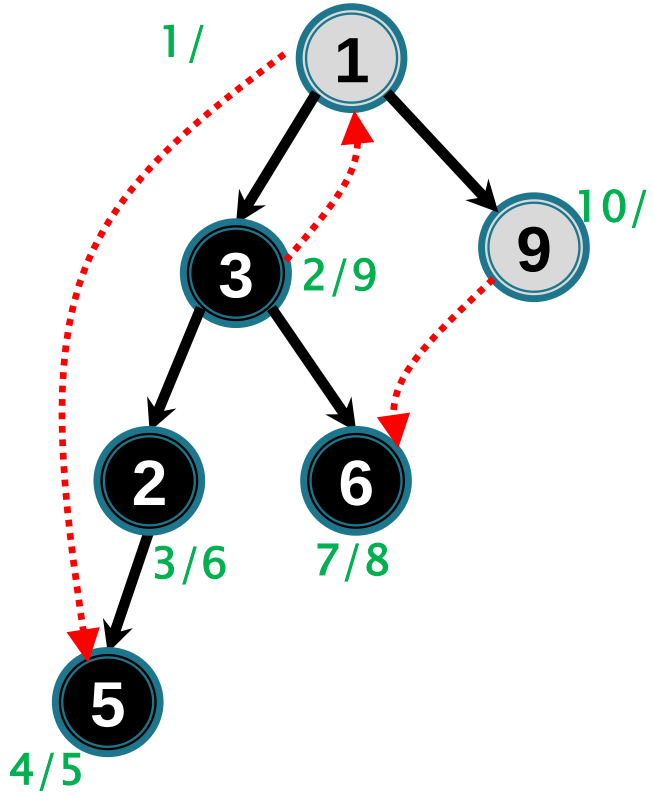


- **de traversare:**

`desc[v] < fin[v]  < desc[u] < fin[u]` ⇔

când este parcurs arcul uv, **v este negru** și **nu este descendent** al lui u, adică `desc[u] > desc[v]`

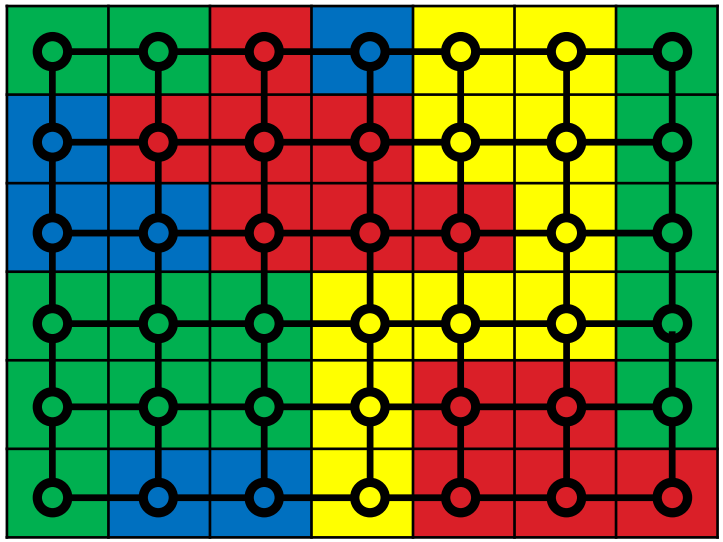




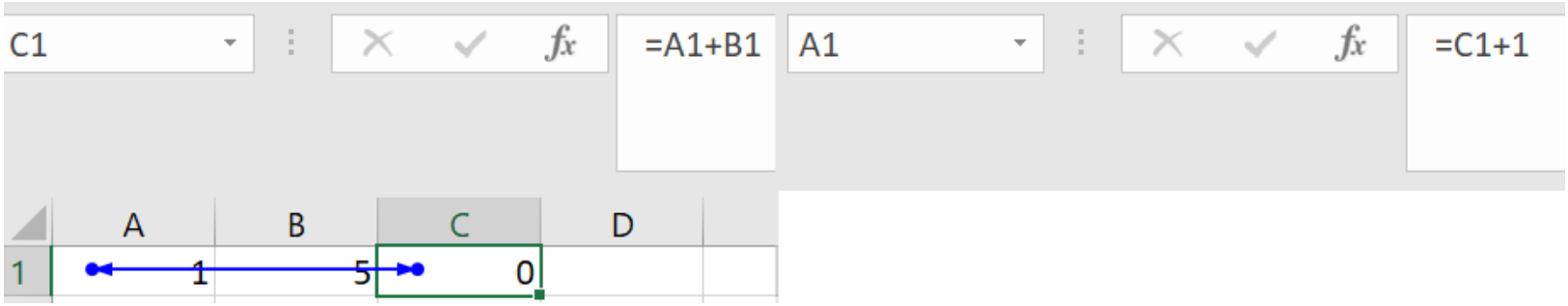
Parcurgerea în adâncime – Aplicații

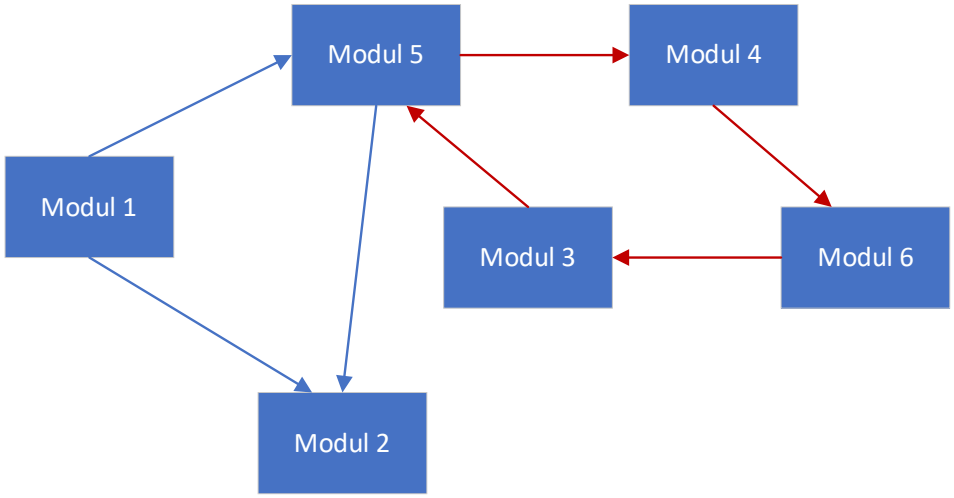
Determinarea componentelor conexe
Determinarea unui arbore parțial

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



Determinarea de cicluri / circuite





Determinarea de cicluri / circuite

- **Ce fel de muchii/arce închid cicluri/circuite?**

Determinarea de cicluri / circuite

- **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)**

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, 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 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


                cat timp v!=y executa
                    adauga v la circuit
                    v = tata[v]
                adauga y,x la circuit
                inverseaza circuit
                STOP
    fin[x] = 1
```