Parcours d'un graphe

ISN 2013

Parcours en profondeur

Parcours en profondeur : principe de l'algorithme

Vous devez parcourir toutes les pages d'un site web. Les pages sont les sommets d'un graphe et un lien entre deux pages est une arête entre ces deux sommets.

Parcours en profondeur : principe de l'algorithme

Vous devez parcourir toutes les pages d'un site web. Les pages sont les sommets d'un graphe et un lien entre deux pages est une arête entre ces deux sommets.

1. Dans le parcours en profondeur, on utilise une pile. On empile le sommet de départ (on visite la page index du site).

Parcours en profondeur : principe de l'algorithme

Vous devez parcourir toutes les pages d'un site web. Les pages sont les sommets d'un graphe et un lien entre deux pages est une arête entre ces deux sommets.

1. Dans le parcours en profondeur, on utilise une pile. On empile le sommet de départ (on visite la page index du site).
2. Si le sommet de la pile présente des voisins qui ne sont pas dans la pile, ni déjà passés dans la pile :

Parcours en profondeur : principe de l'algorithme

Vous devez parcourir toutes les pages d'un site web. Les pages sont les sommets d'un graphe et un lien entre deux pages est une arête entre ces deux sommets.

1. Dans le parcours en profondeur, on utilise une pile. On empile le sommet de départ (on visite la page index du site).
2. Si le sommet de la pile présente des voisins qui ne sont pas dans la pile, ni déjà passés dans la pile :
   - alors on sélectionne l'un de ces voisins et on l'empile (en le marquant de son numéro de découverte),

Parcours en profondeur : principe de l'algorithme

Vous devez parcourir toutes les pages d'un site web. Les pages sont les sommets d'un graphe et un lien entre deux pages est une arête entre ces deux sommets.

1. Dans le parcours en profondeur, on utilise une pile. On empile le sommet de départ (on visite la page index du site).
2. Si le sommet de la pile présente des voisins qui ne sont pas dans la pile, ni déjà passés dans la pile :
   - alors on sélectionne l'un de ces voisins et on l'empile (en le marquant de son numéro de découverte),
   - sinon on dépile (c'est à dire on supprime l'élément du sommet de la pile).

Parcours en profondeur : principe de l'algorithme

Vous devez parcourir toutes les pages d'un site web. Les pages sont les sommets d'un graphe et un lien entre deux pages est une arête entre ces deux sommets.

1. Dans le parcours en profondeur, on utilise une pile. On empile le sommet de départ (on visite la page index du site).
2. Si le sommet de la pile présente des voisins qui ne sont pas dans la pile, ni déjà passés dans la pile :
   - alors on sélectionne l'un de ces voisins et on l'empile (en le marquant de son numéro de découverte),
   - sinon on dépile (c'est à dire on supprime l'élément du sommet de la pile).
3. On recommence au point 2 (tant que la pile n'est pas vide).

Parcours en profondeur : principe de l'algorithme

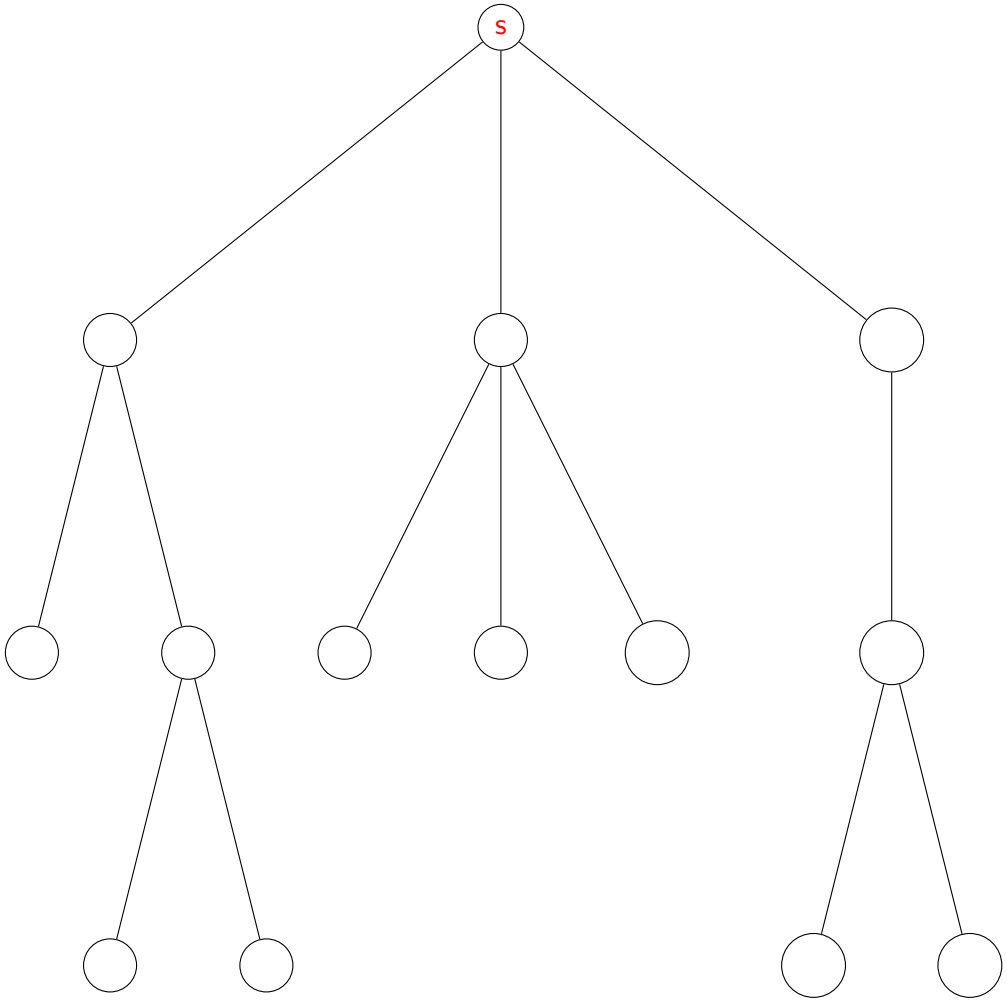
Vous devez parcourir toutes les pages d'un site web. Les pages sont les sommets d'un graphe et un lien entre deux pages est une arête entre ces deux sommets.

1. Dans le parcours en profondeur, on utilise une pile. On empile le sommet de départ (on visite la page index du site).
2. Si le sommet de la pile présente des voisins qui ne sont pas dans la pile, ni déjà passés dans la pile :
   - alors on sélectionne l'un de ces voisins et on l'empile (en le marquant de son numéro de découverte),
   - sinon on dépile (c'est à dire on supprime l'élément du sommet de la pile).
3. On recommence au point 2 (tant que la pile n'est pas vide).

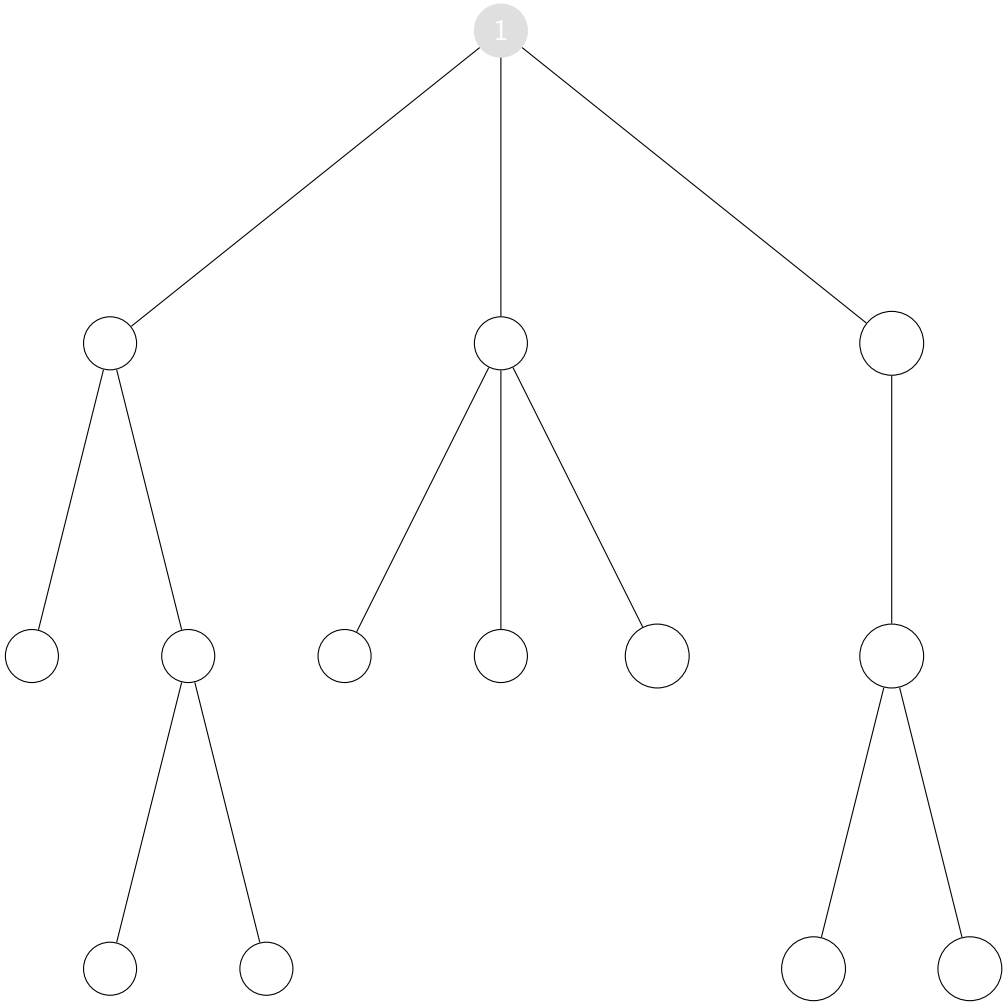
En d'autres termes, on traite toujours en priorité les liens des pages les plus tard découvertes.

Parcours en profondeur d'un arbre

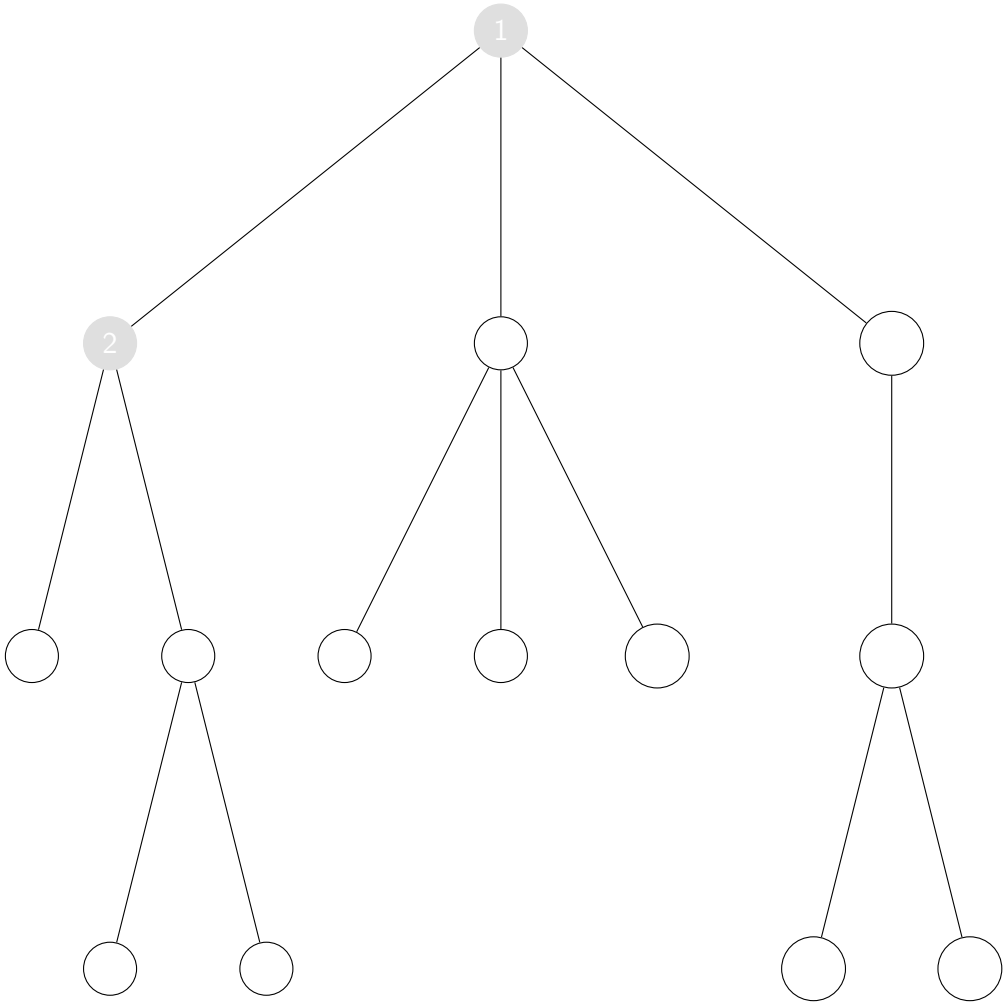
Parcourir en profondeur le graphe ci-dessous à partir du sommet s :



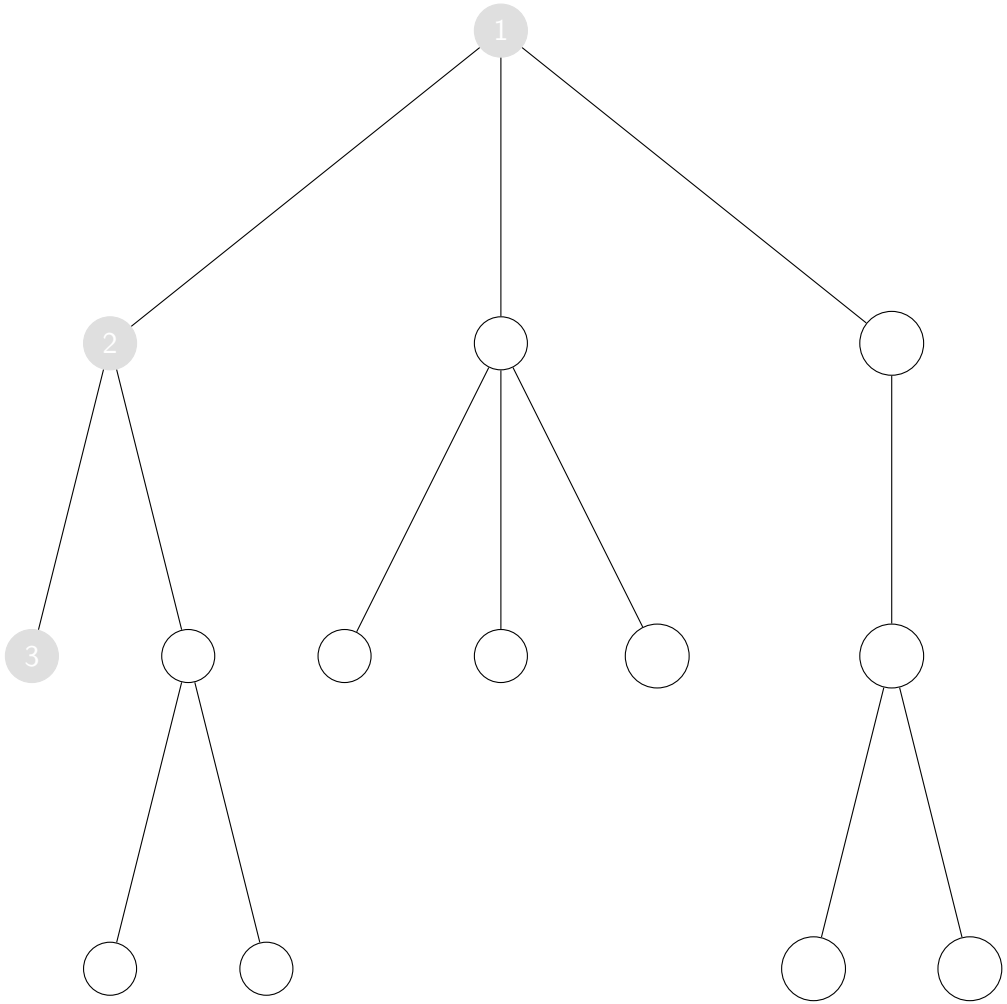
Parcours en profondeur d'un arbre



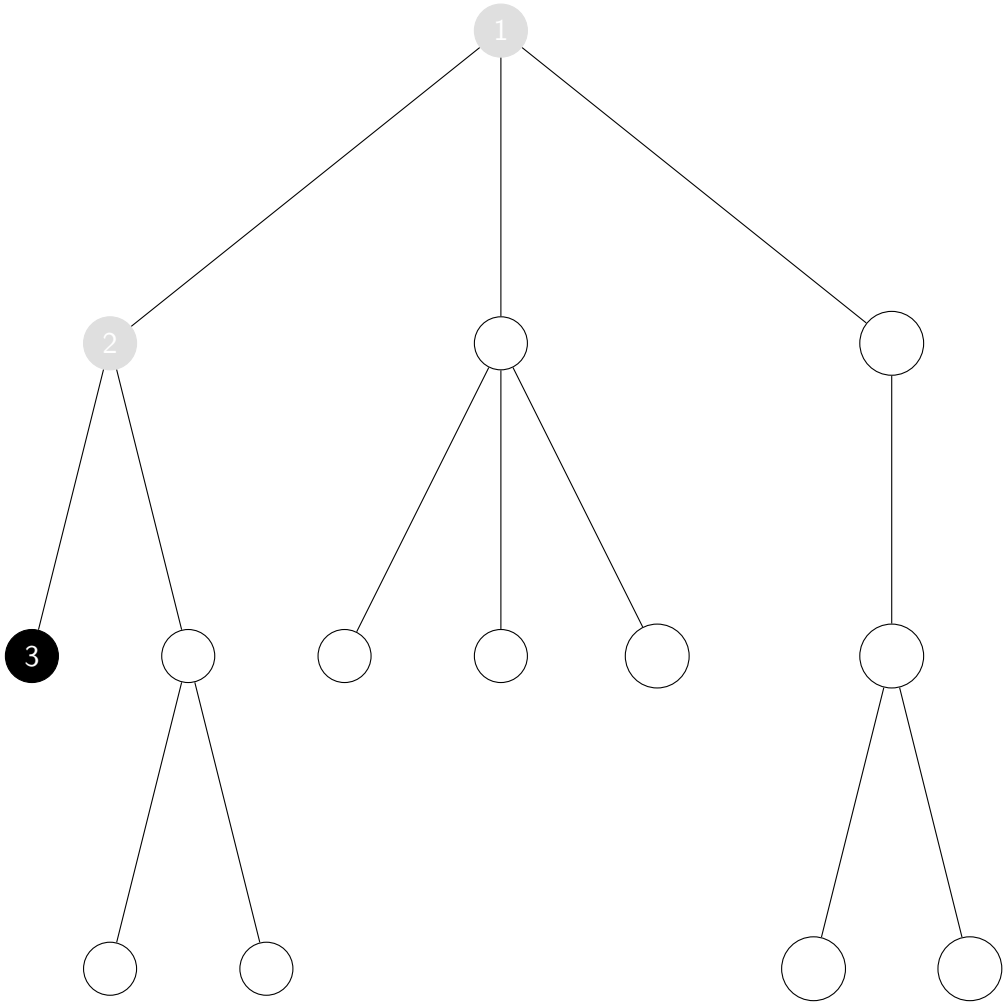
Parcours en profondeur d'un arbre



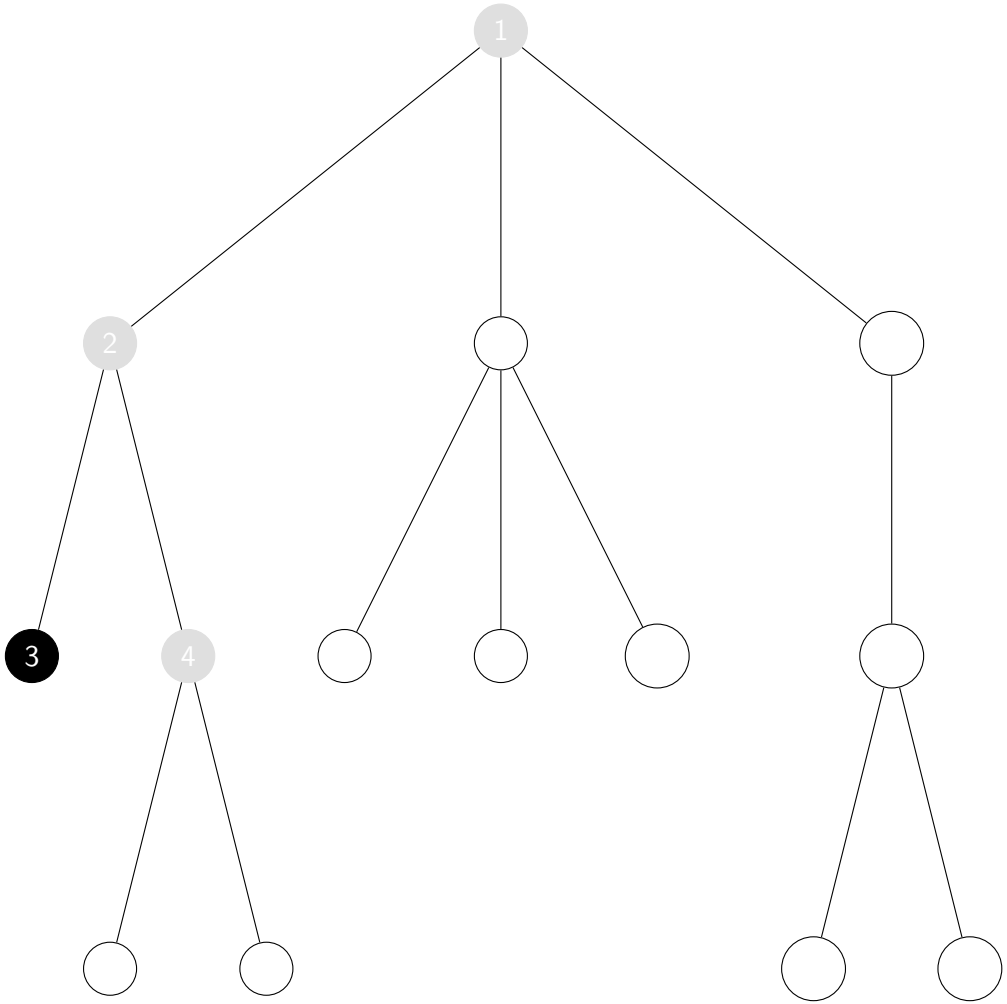
Parcours en profondeur d'un arbre



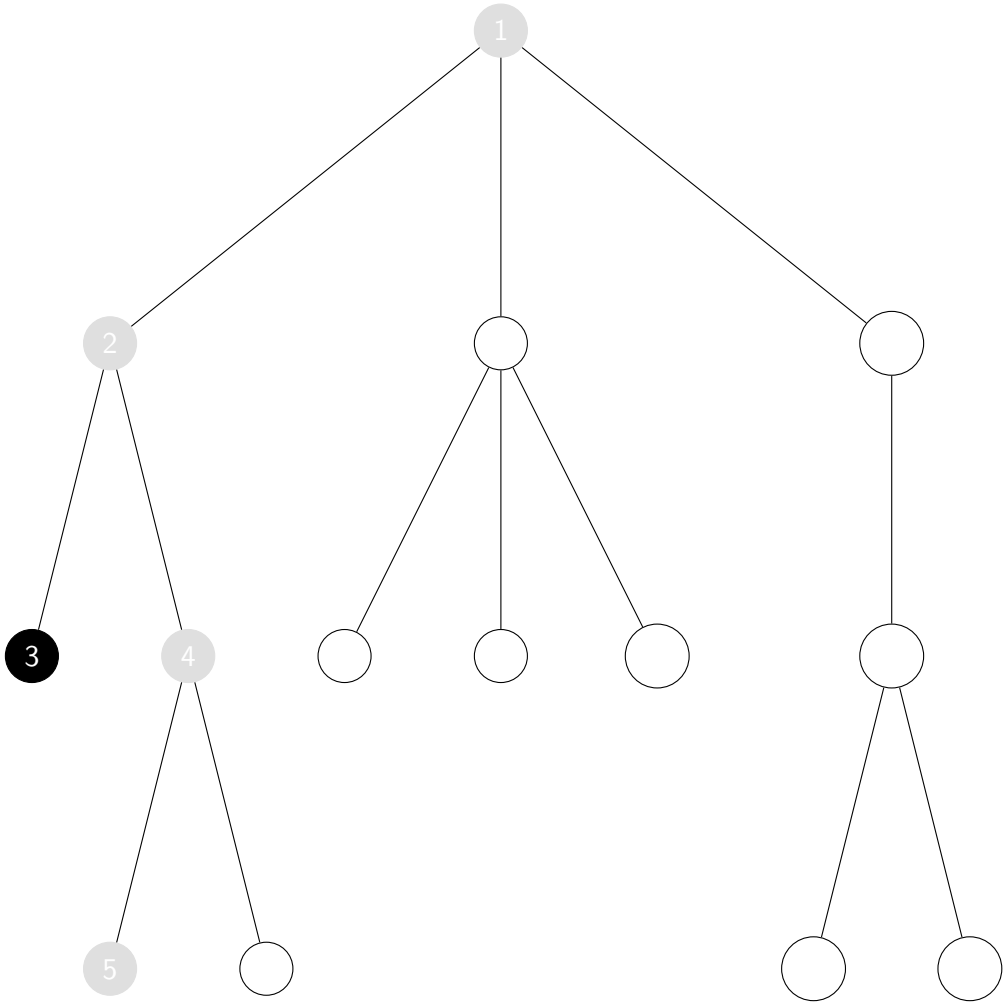
Parcours en profondeur d'un arbre



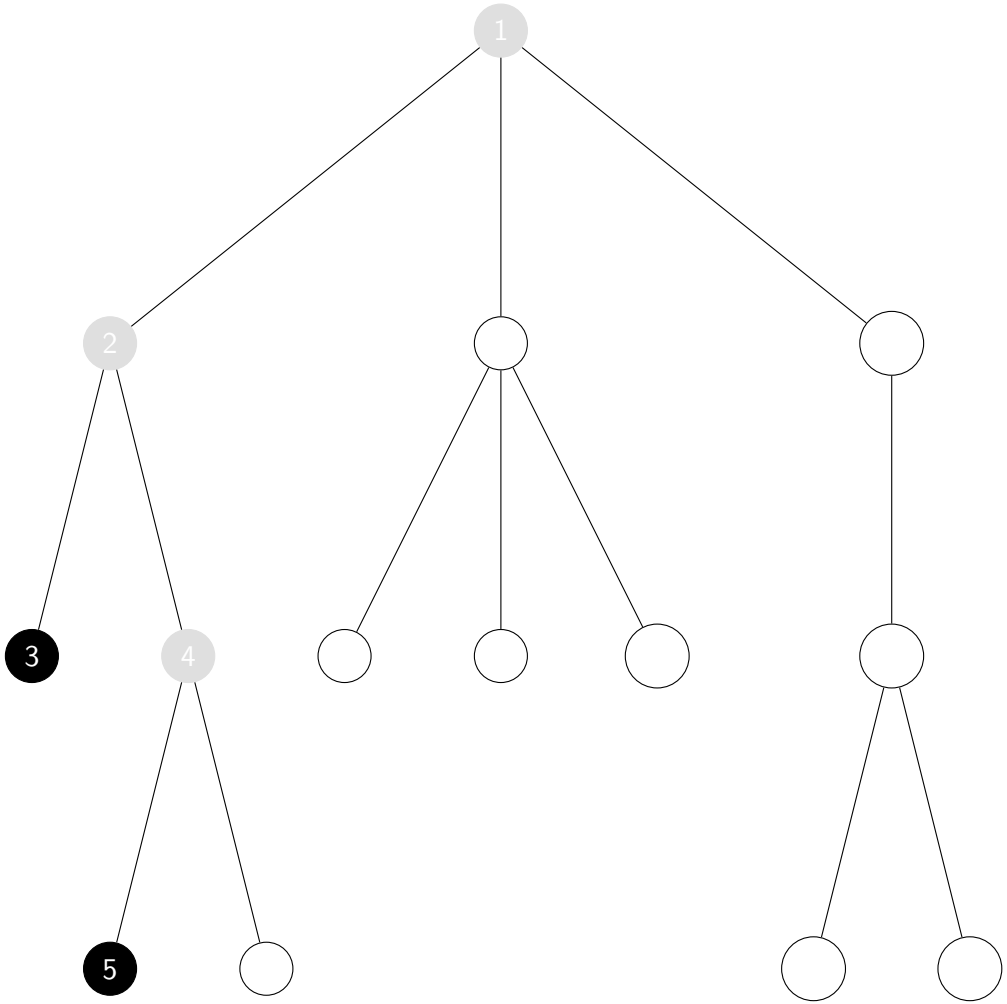
Parcours en profondeur d'un arbre



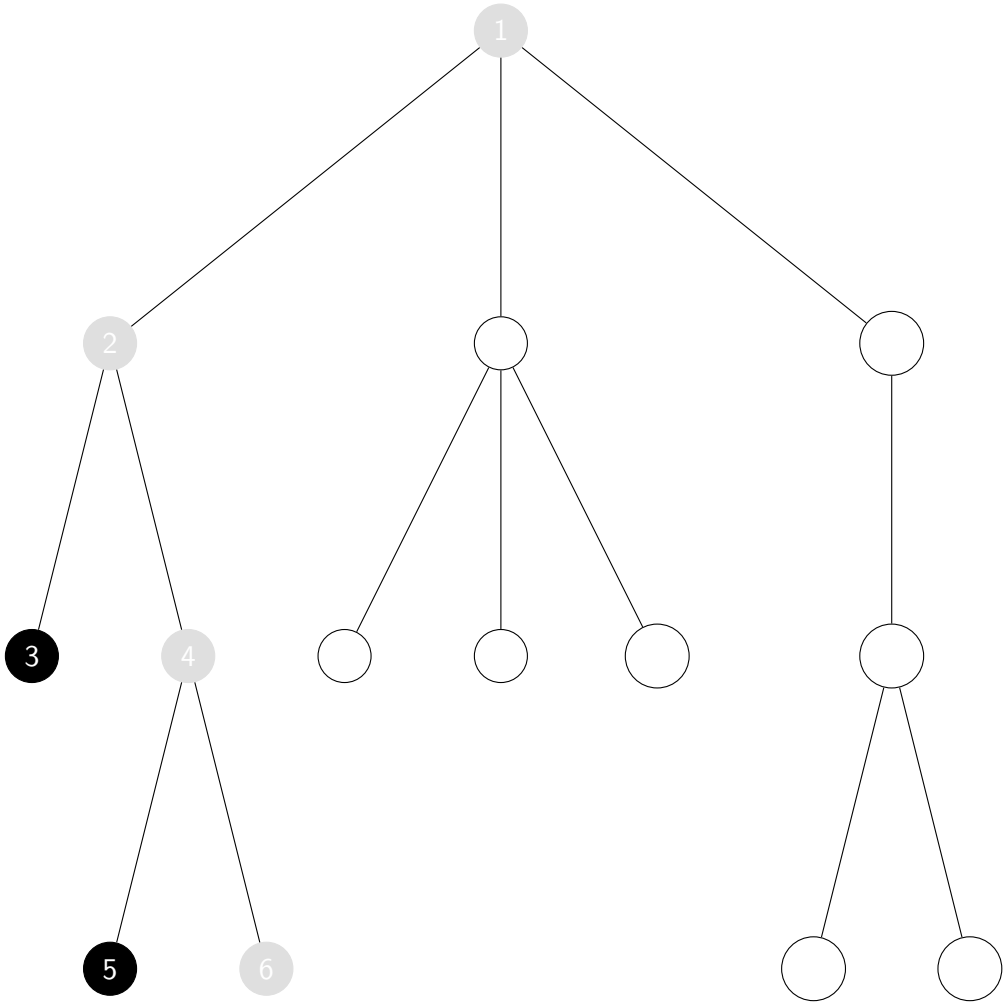
Parcours en profondeur d'un arbre



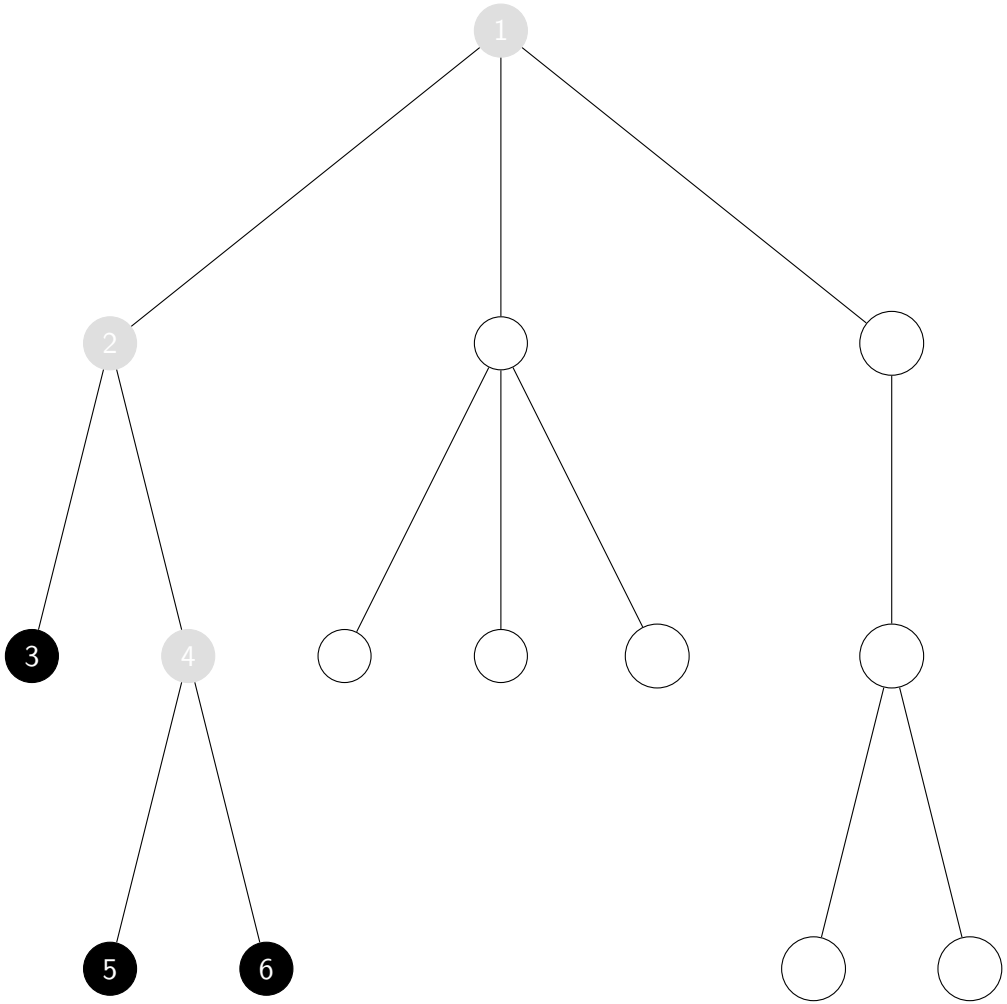
Parcours en profondeur d'un arbre



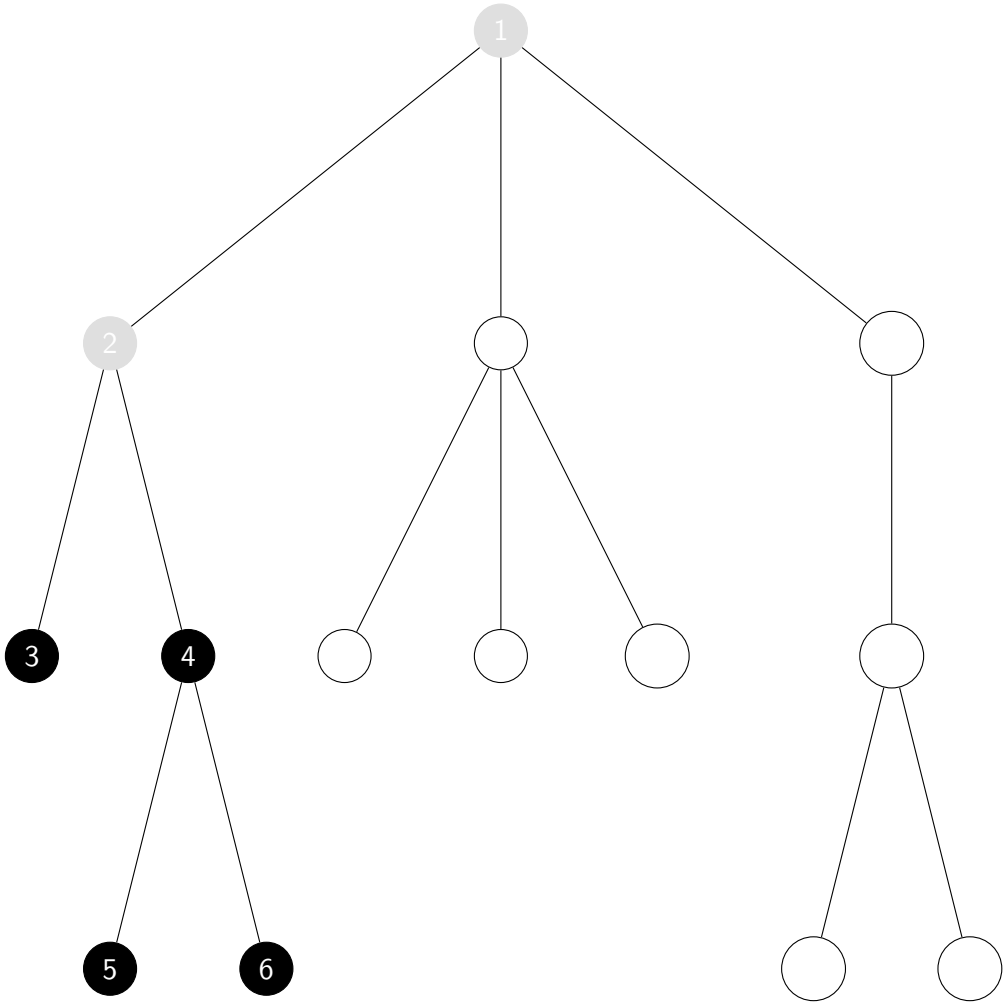
Parcours en profondeur d'un arbre



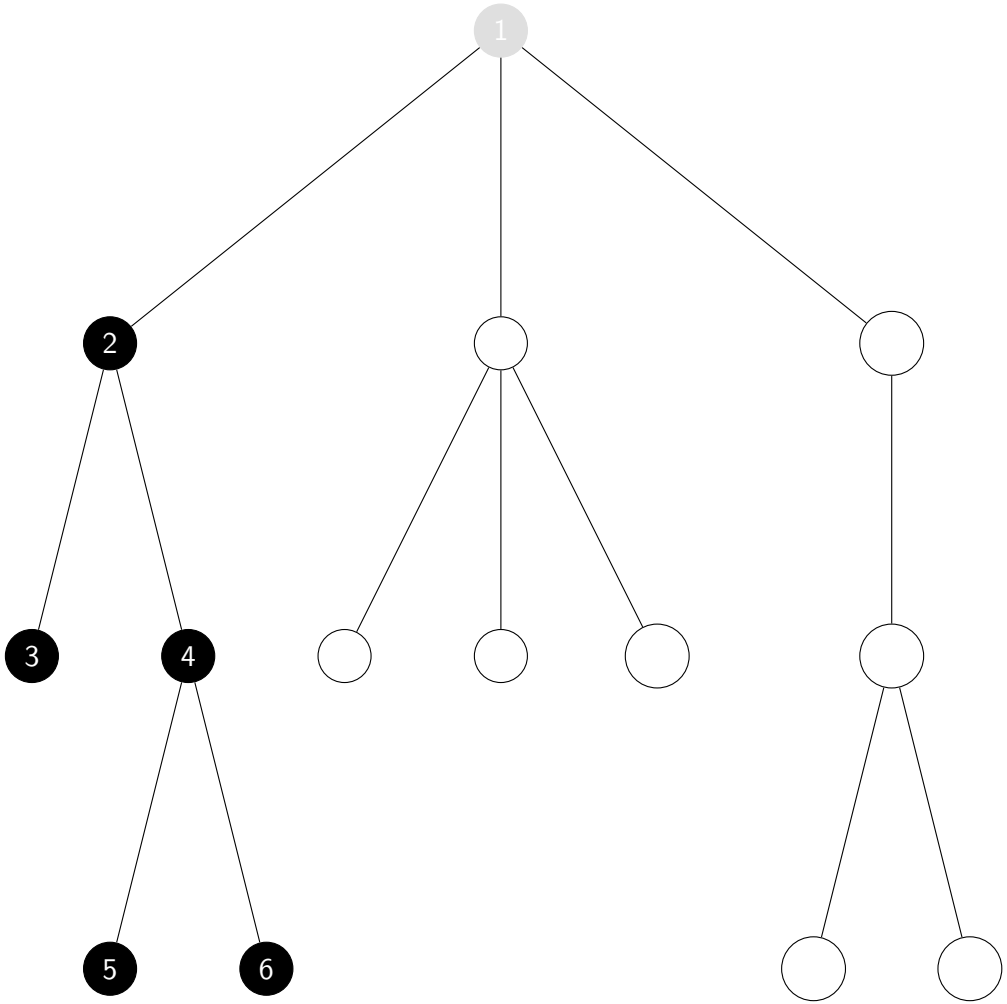
Parcours en profondeur d'un arbre



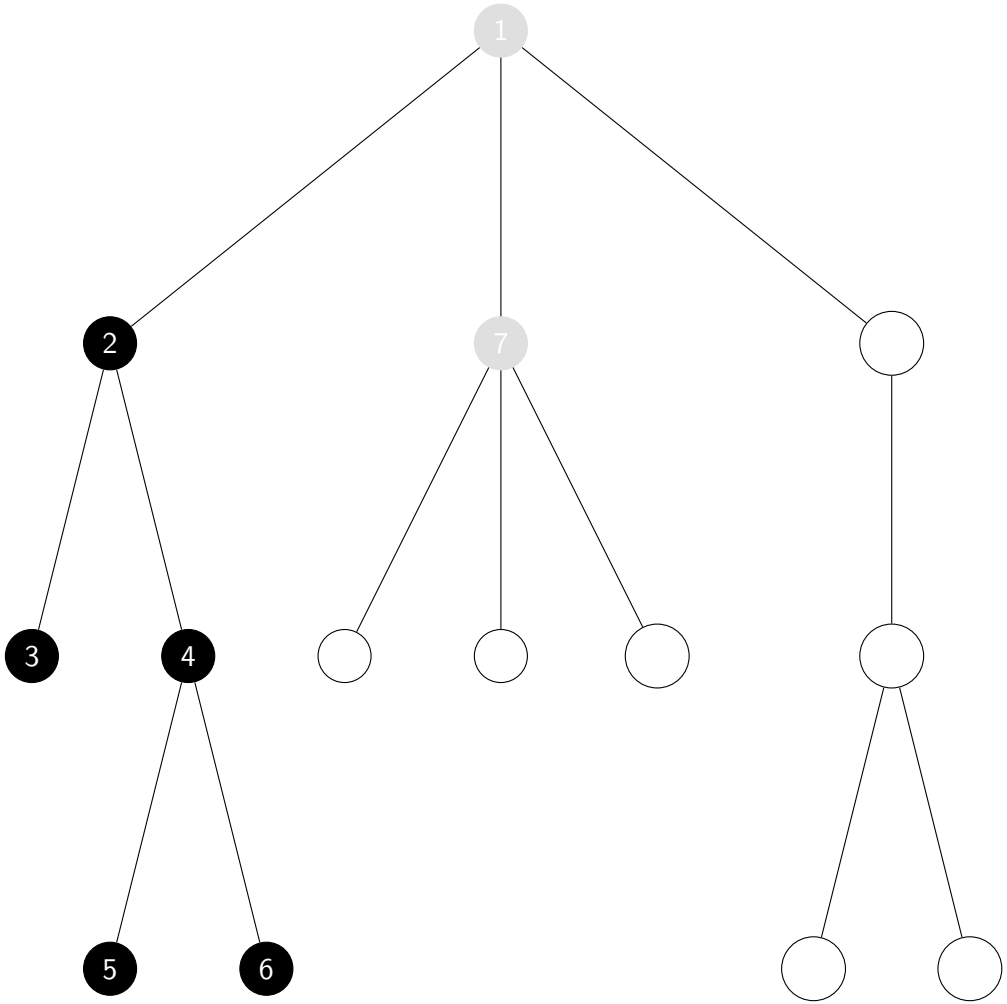
Parcours en profondeur d'un arbre



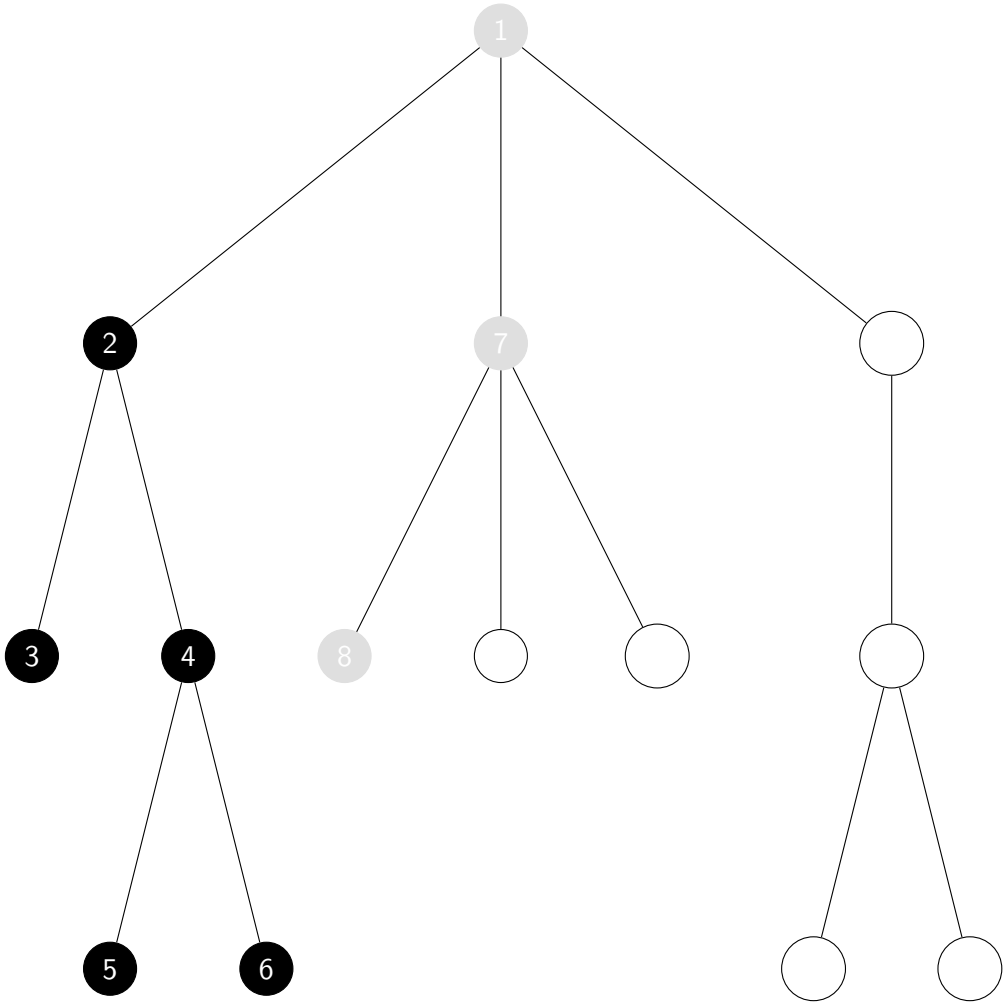
Parcours en profondeur d'un arbre



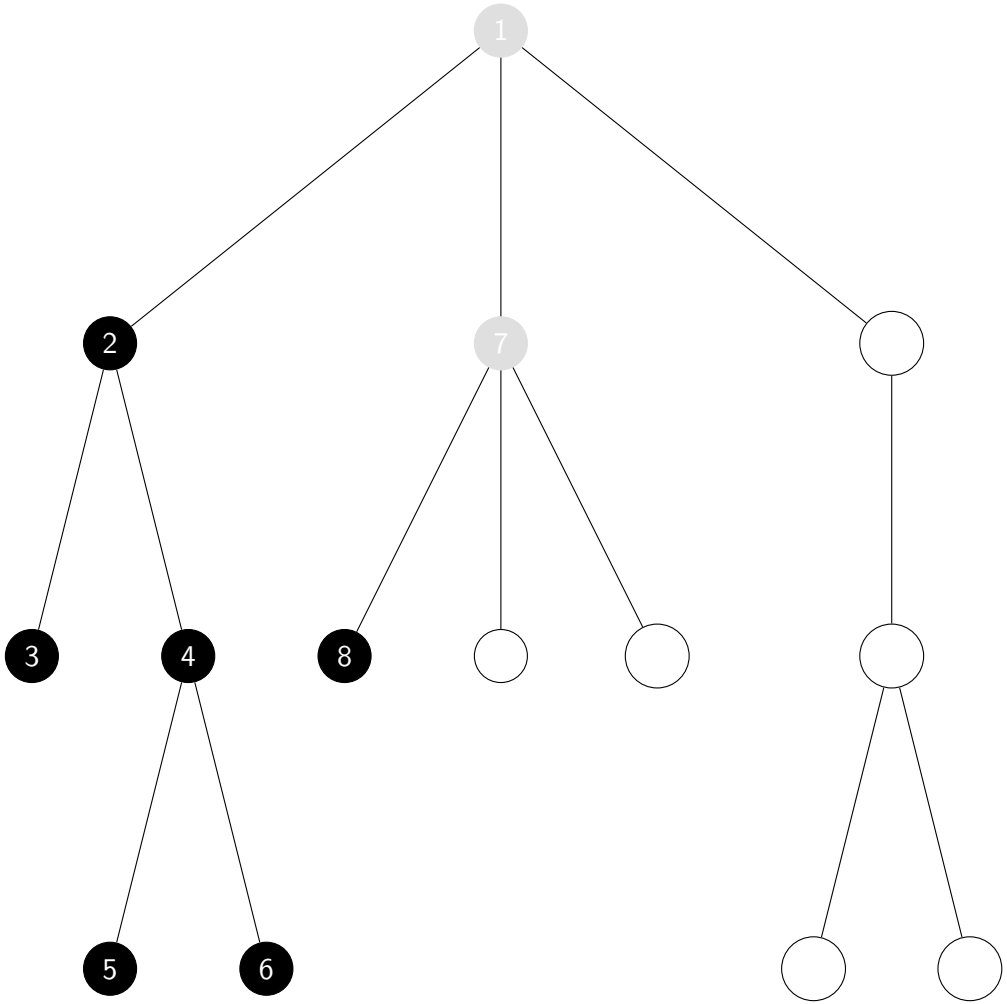
Parcours en profondeur d'un arbre



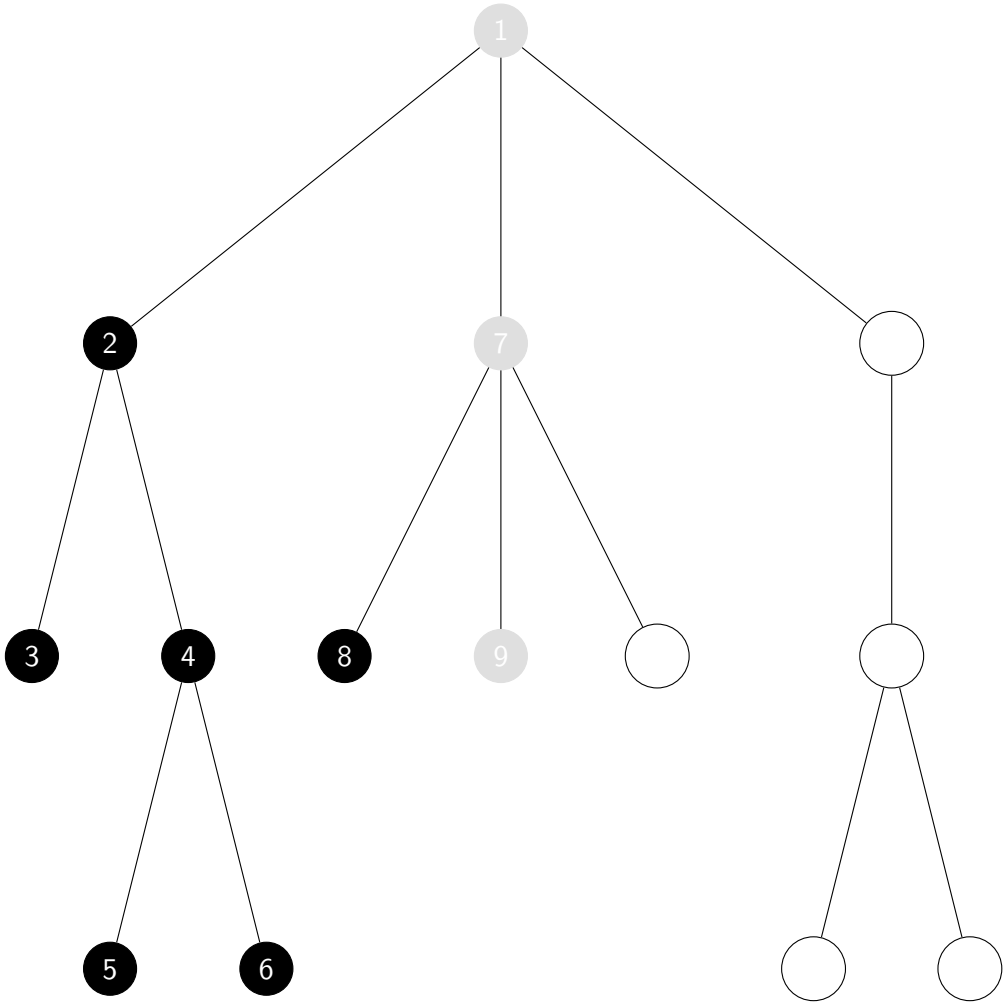
Parcours en profondeur d'un arbre



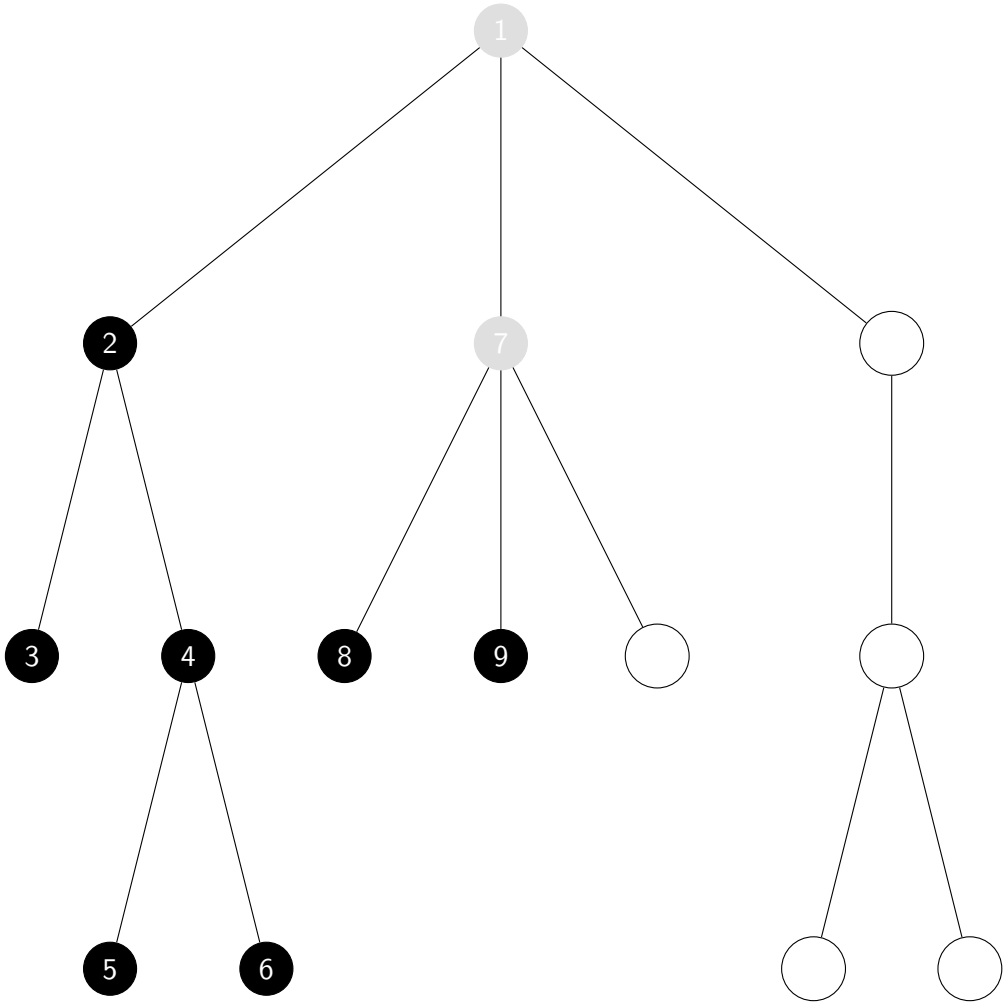
Parcours en profondeur d'un arbre



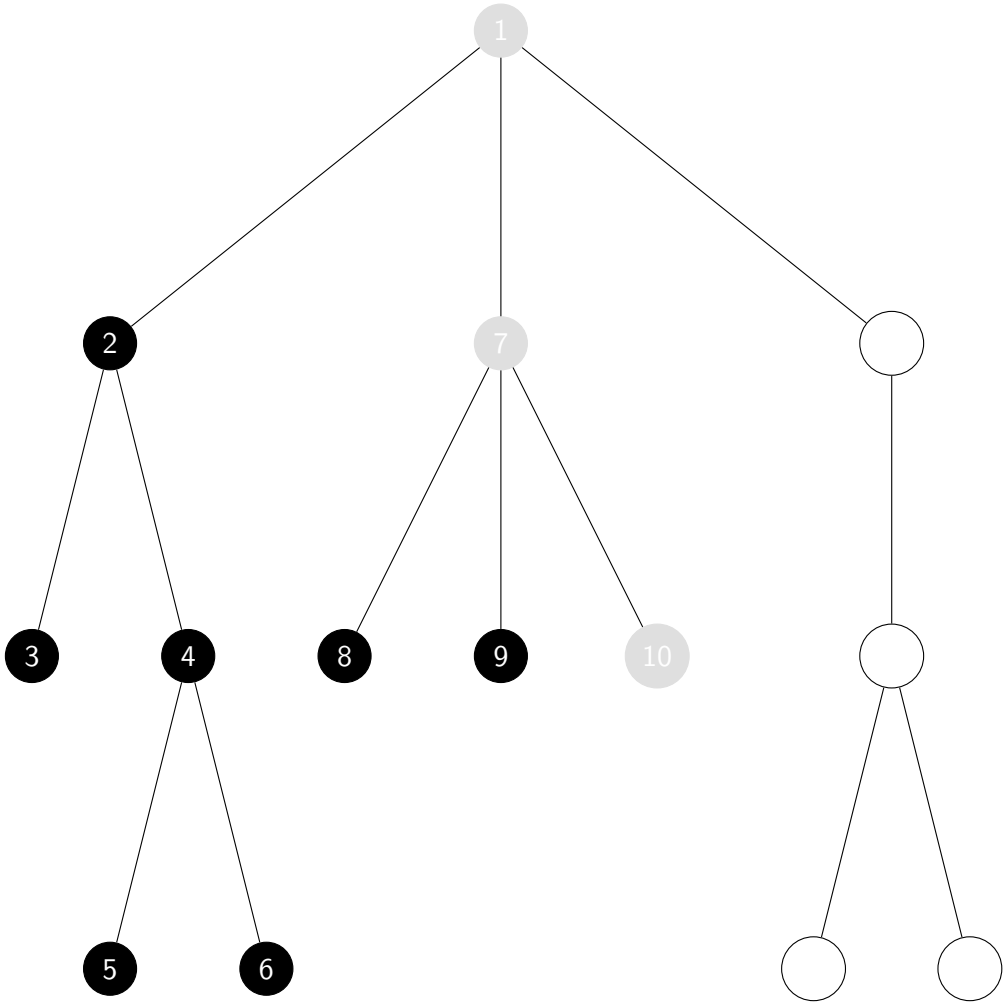
Parcours en profondeur d'un arbre



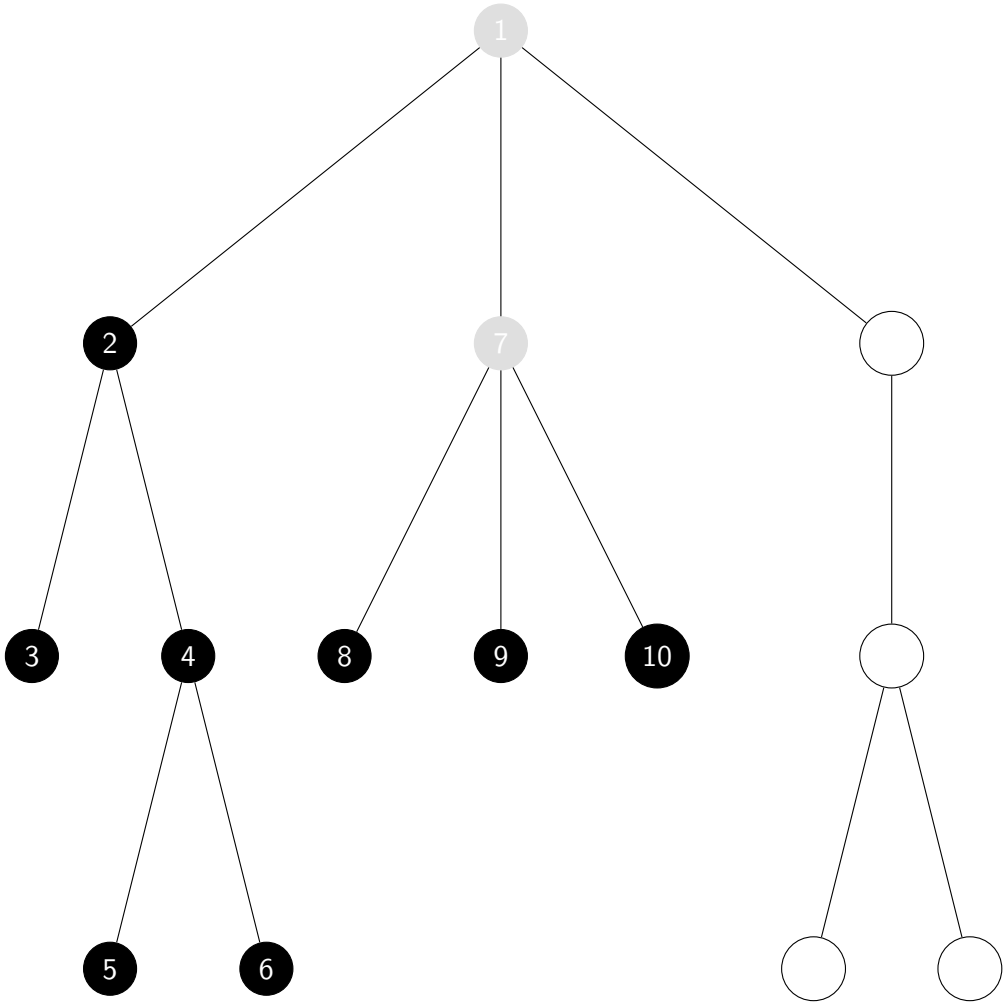
Parcours en profondeur d'un arbre



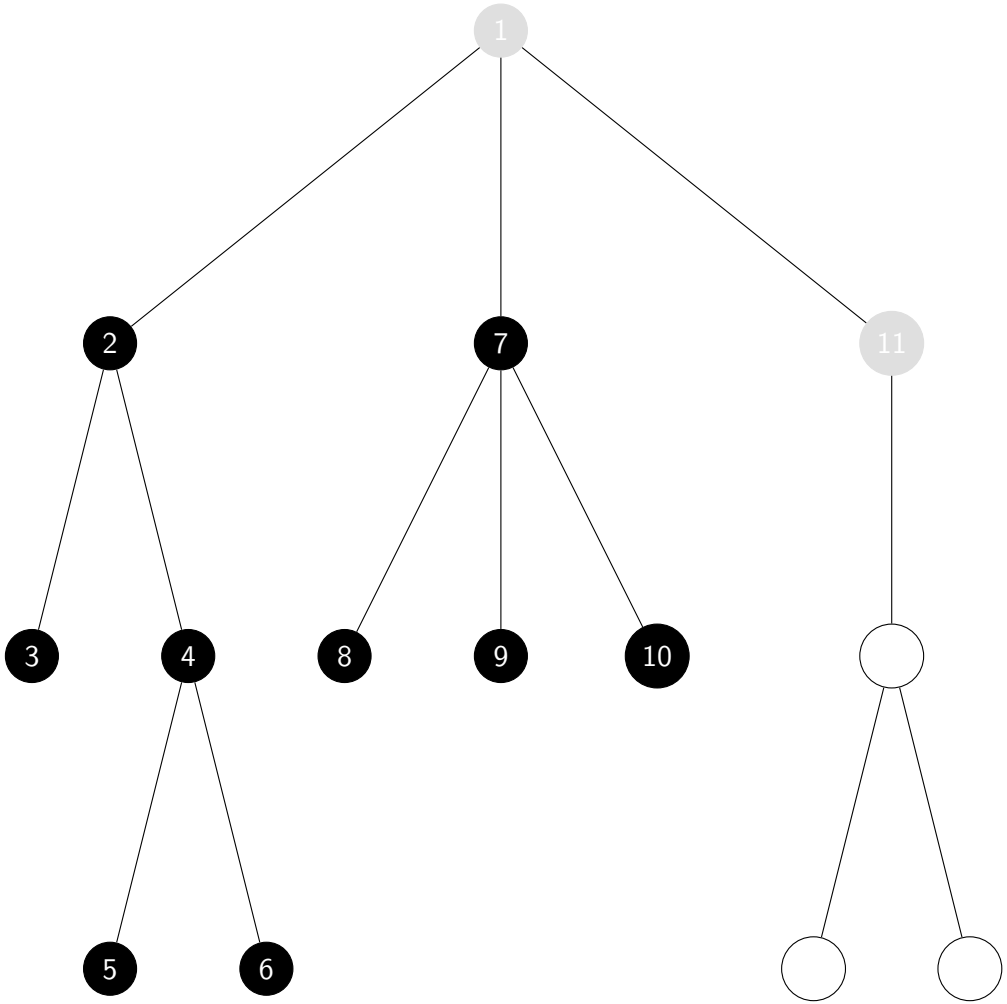
Parcours en profondeur d'un arbre



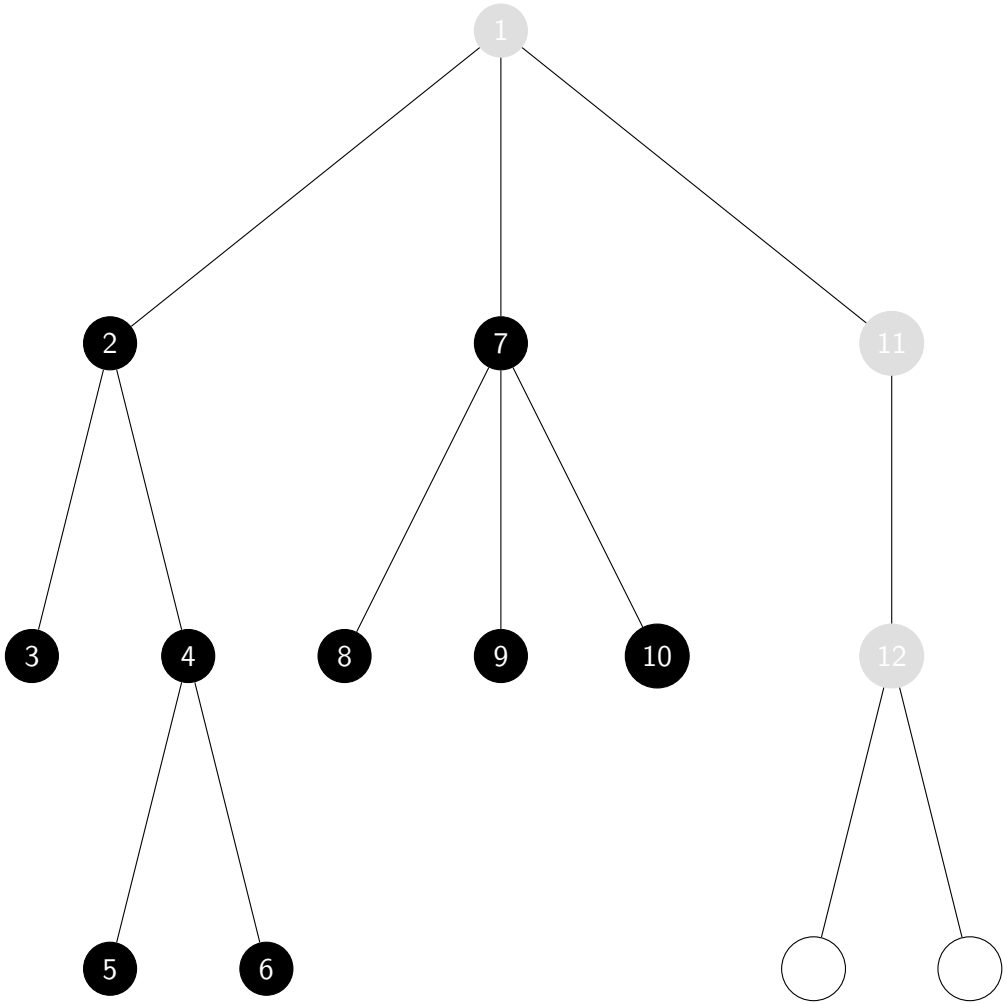
Parcours en profondeur d'un arbre



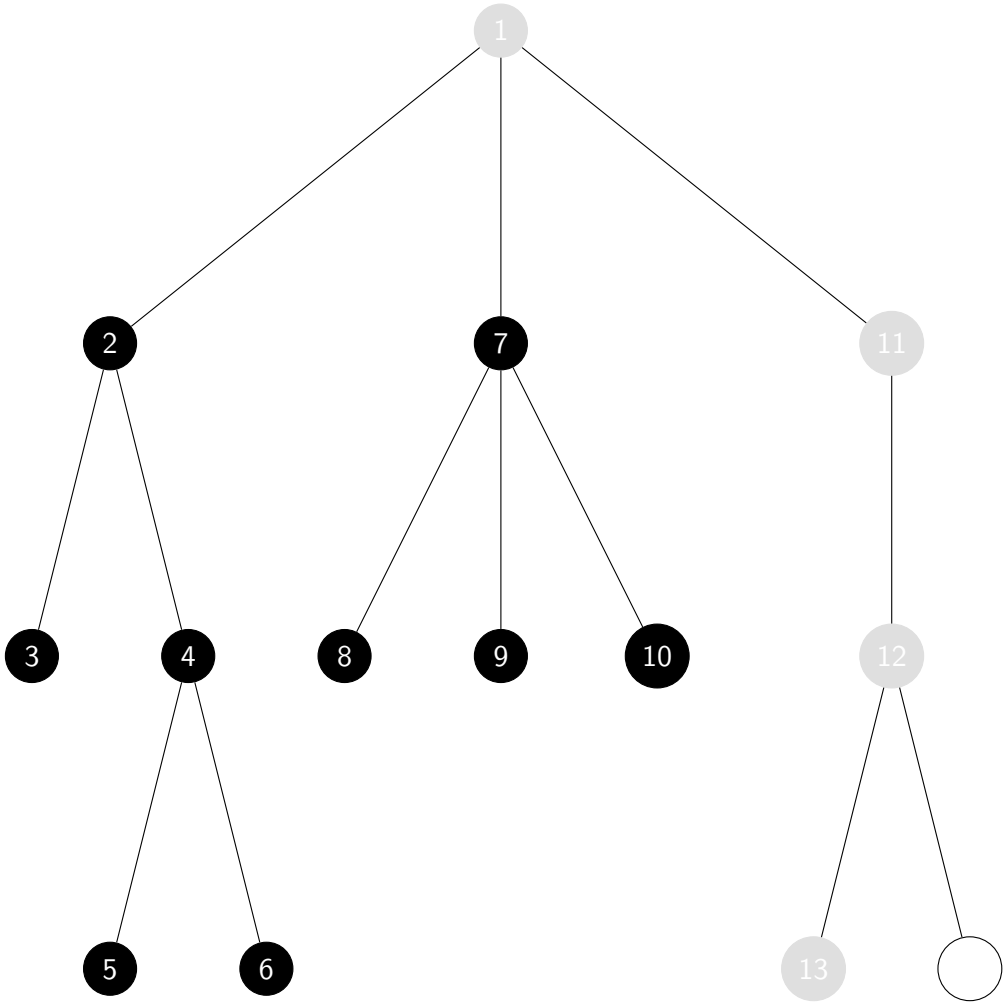
Parcours en profondeur d'un arbre



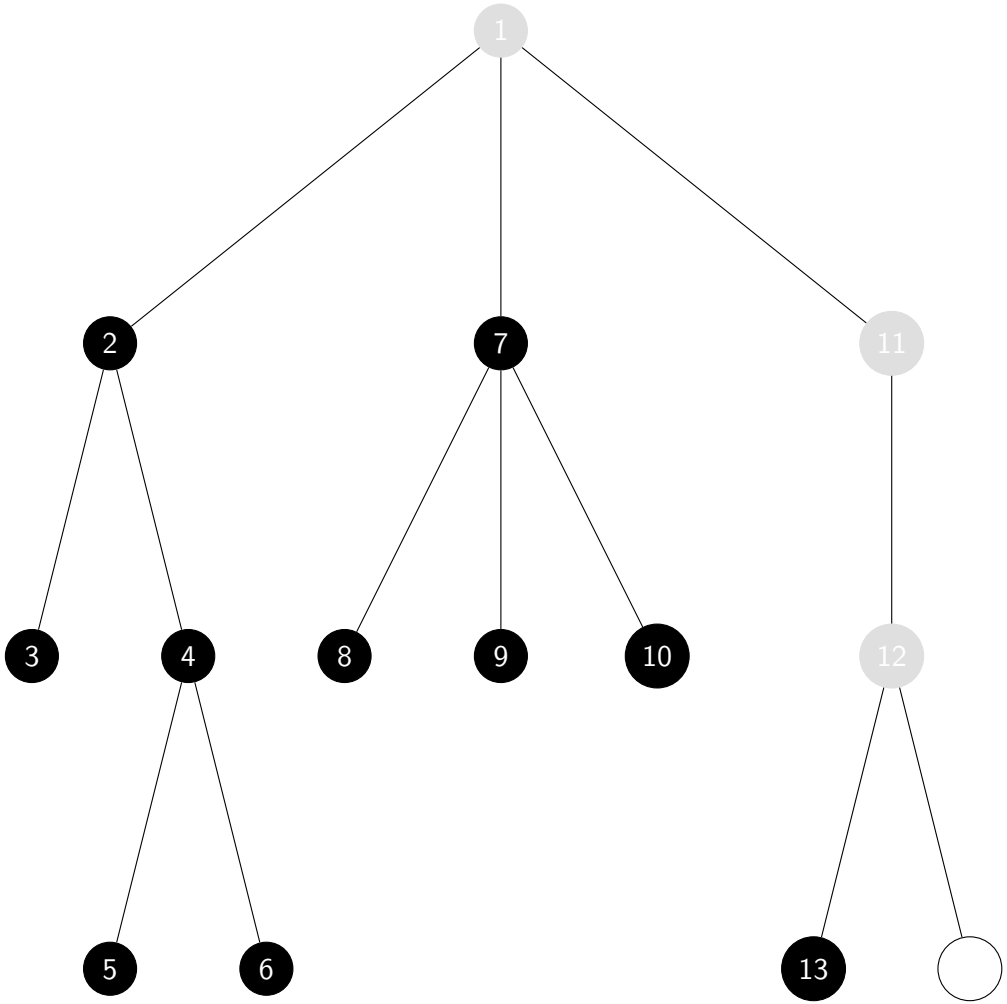
Parcours en profondeur d'un arbre



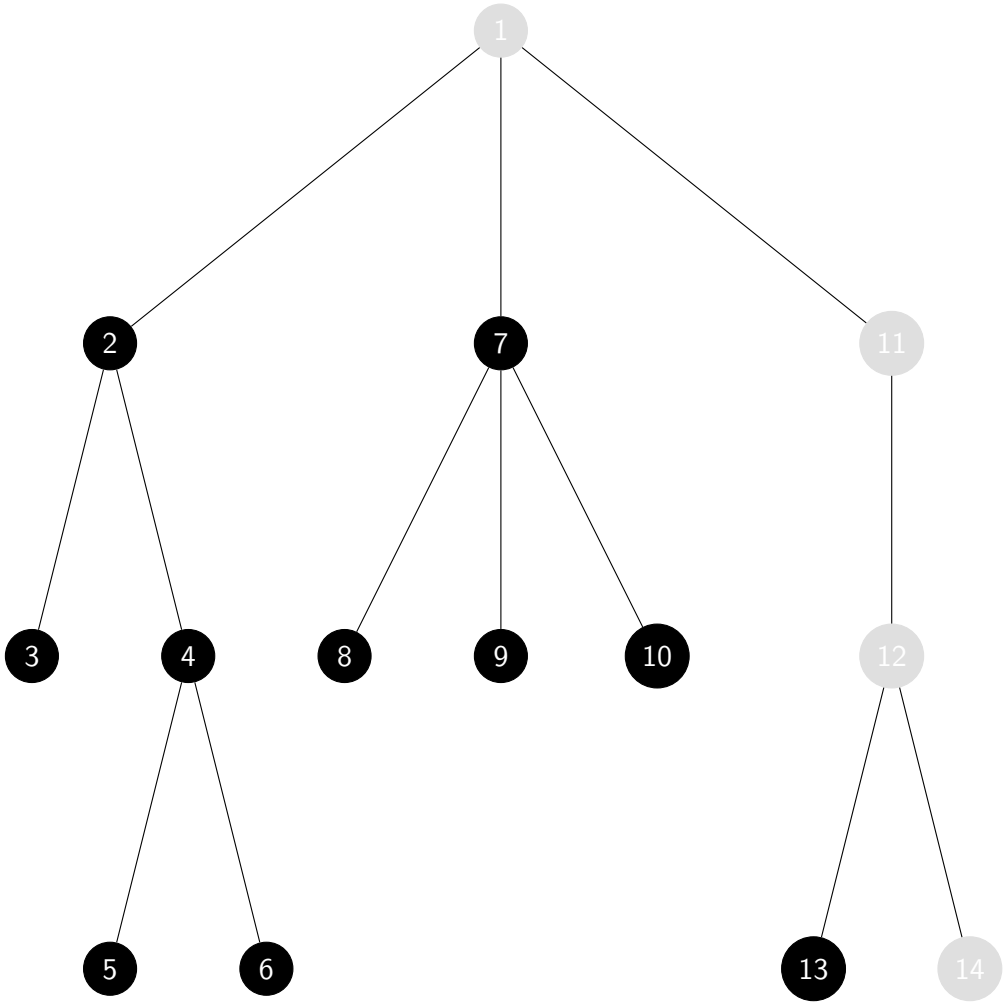
Parcours en profondeur d'un arbre



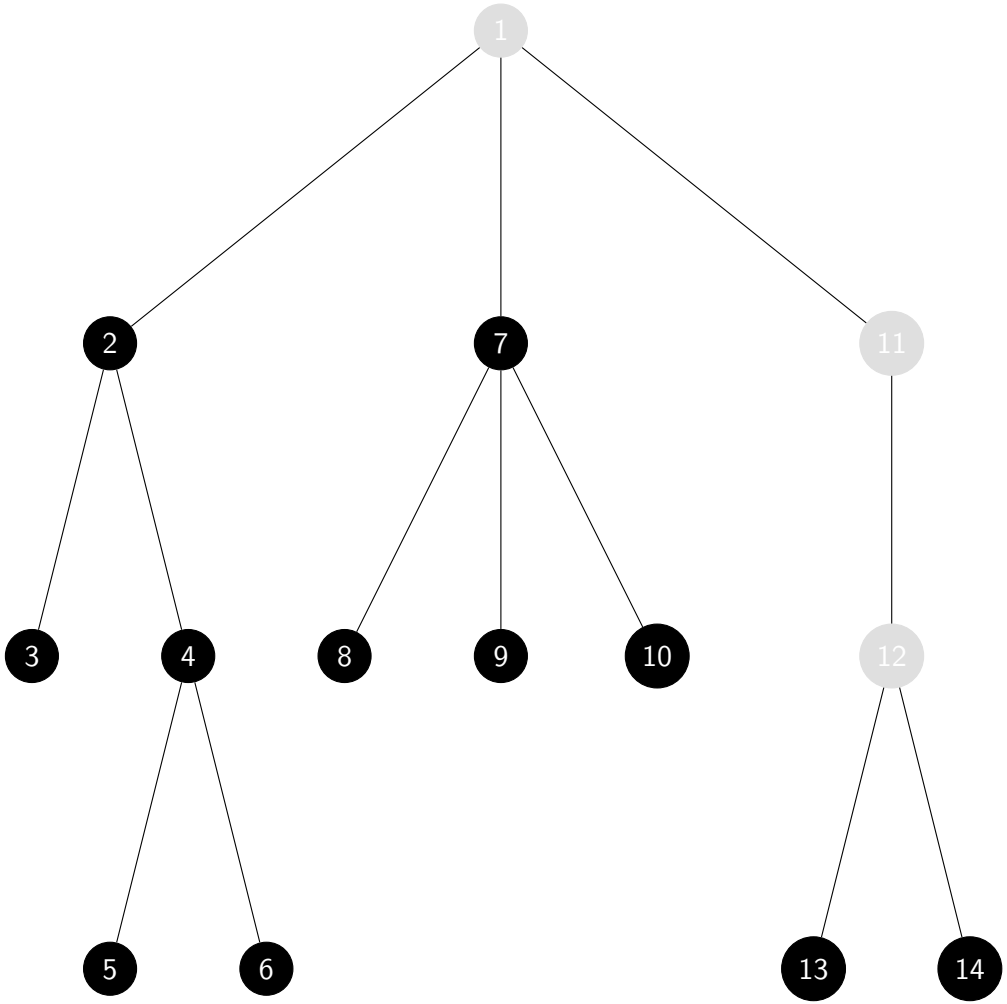
Parcours en profondeur d'un arbre



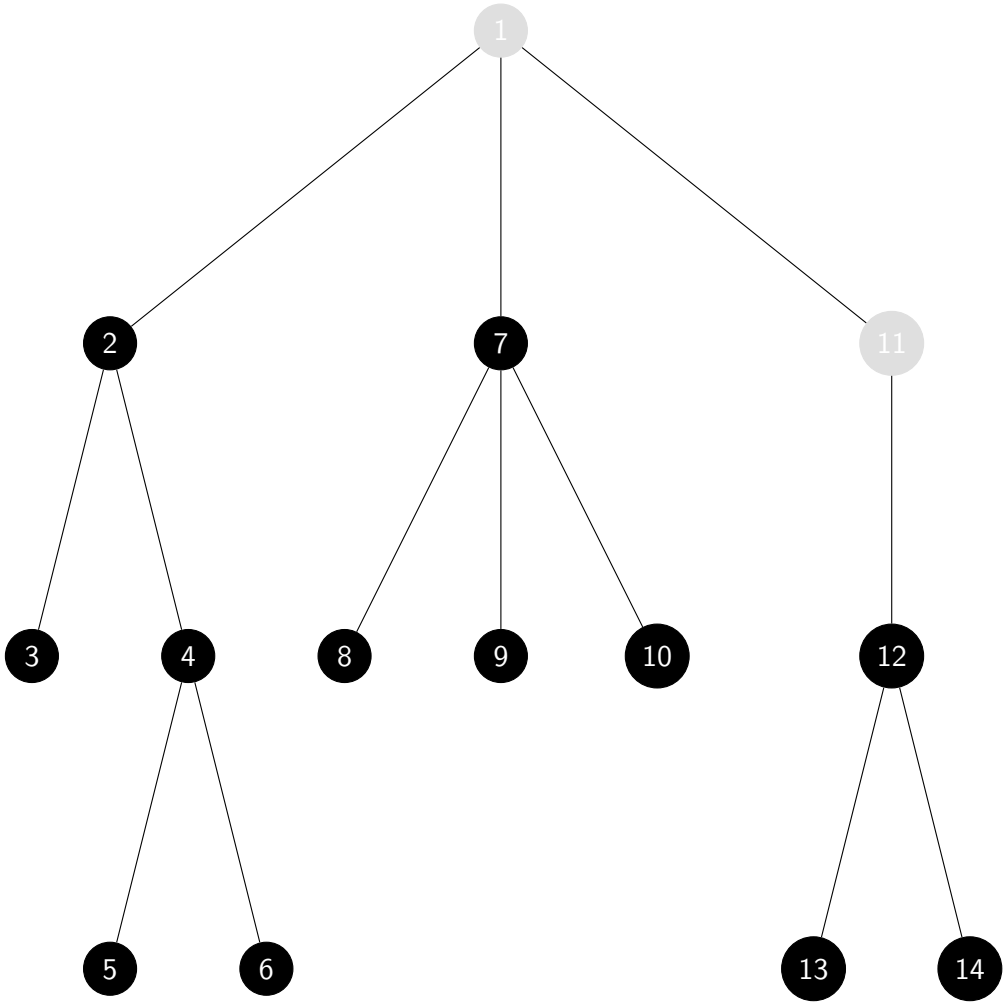
Parcours en profondeur d'un arbre



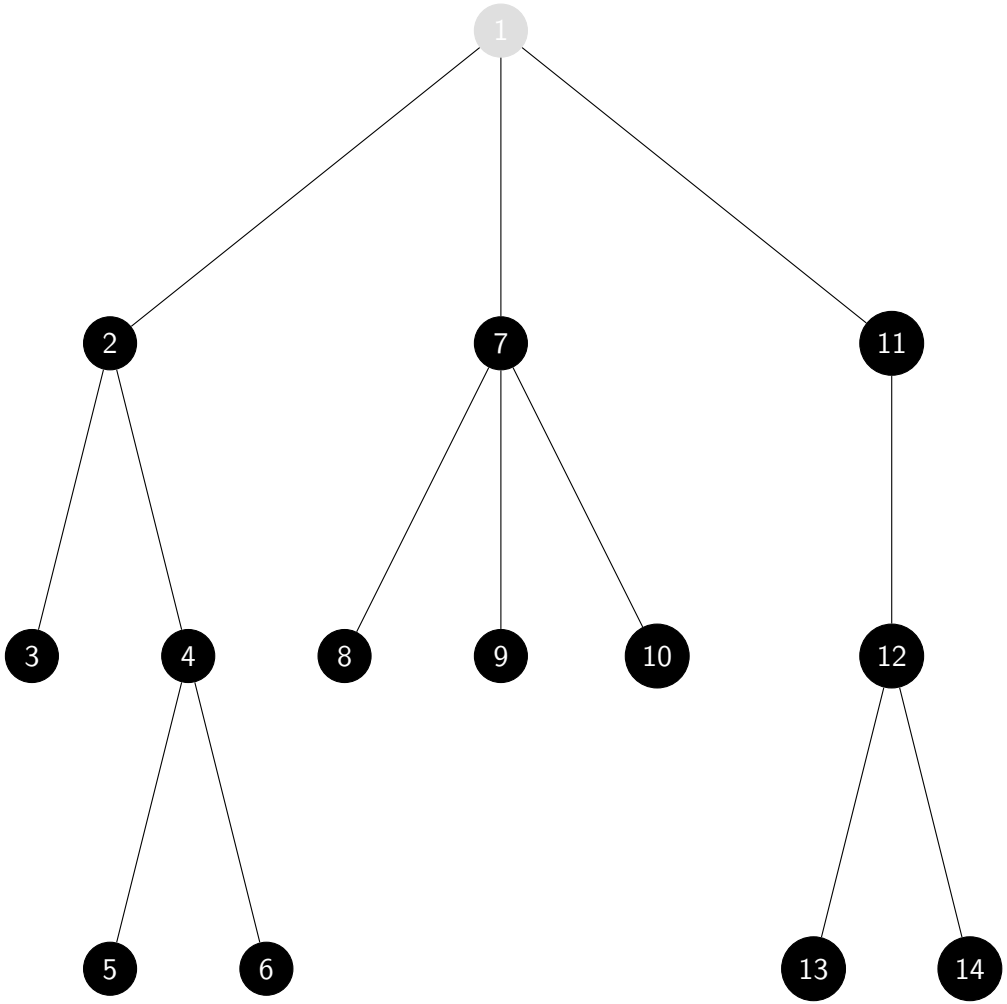
Parcours en profondeur d'un arbre



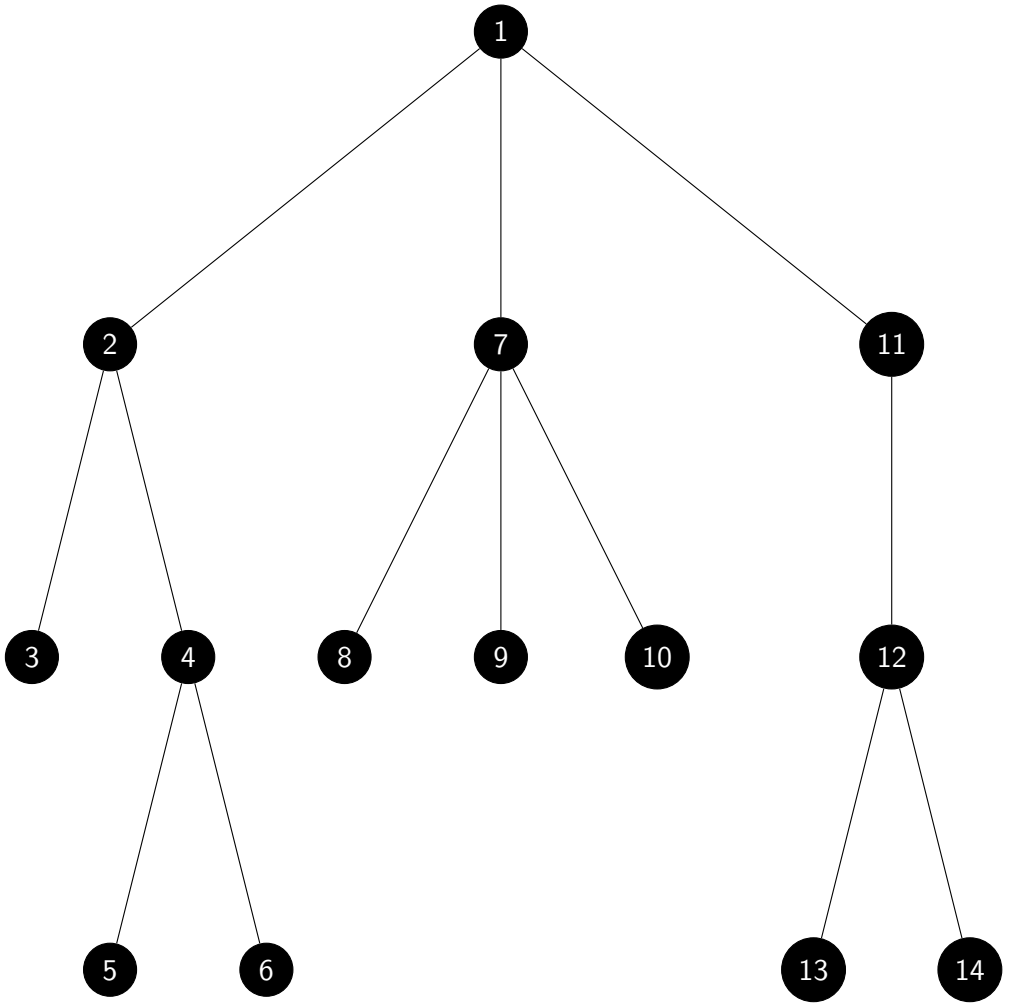
Parcours en profondeur d'un arbre



Parcours en profondeur d'un arbre



Parcours en profondeur d'un arbre



arbre DFS en python

DFS (depth first search) : programmation python

Exercice à rendre 3.

Avec la représentation d'un graphe par un dictionnaire comme précédemment, programmer en langage python le DFS avec les variables suivantes :

- Un dictionnaire P. En fin de parcours, pour tout sommet s du graphe P[s] sera le père de s, c'est à dire le sommet à partir duquel le sommet s a été découvert lors du parcours.

DFS (depth first search) : programmation python

Exercice à rendre 3.
Avec la représentation d'un graphe par un dictionnaire comme précédemment, programmer en langage python le DFS avec les variables suivantes :

- Un dictionnaire P. En fin de parcours, pour tout sommet s du graphe P[s] sera le père de s, c'est à dire le sommet à partir duquel le sommet s a été découvert lors du parcours.
- Un dictionnaire couleur. Pour tout sommet s, couleur[s] vaut blanc si le sommet s n'est pas encore découvert, gris s'il est déjà découvert mais non encore fermé (c'est à dire si l'algorithme n' a pas encore découvert tous ses voisins), noir lorsque ce sommet est fermé.

DFS (depth first search) : programmation python

Exercice à rendre 3.
Avec la représentation d'un graphe par un dictionnaire comme précédemment, programmer en langage python le DFS avec les variables suivantes :

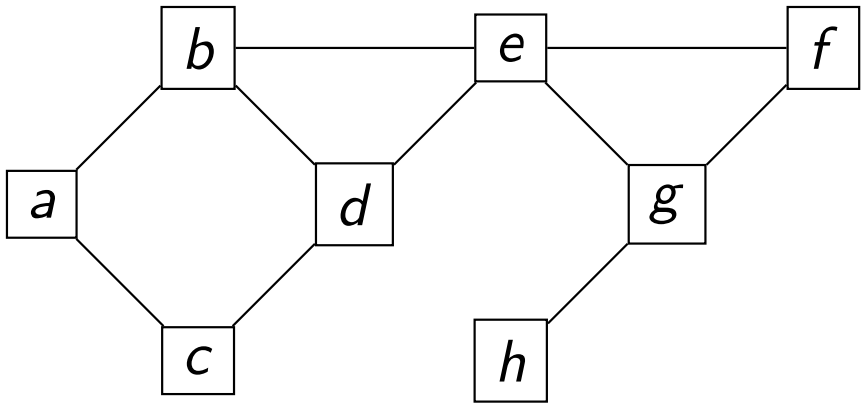
- Un dictionnaire P. En fin de parcours, pour tout sommet s du graphe P[s] sera le père de s, c'est à dire le sommet à partir duquel le sommet s a été découvert lors du parcours.
- Un dictionnaire couleur. Pour tout sommet s, couleur[s] vaut blanc si le sommet s n'est pas encore découvert, gris s'il est déjà découvert mais non encore fermé (c'est à dire si l'algorithme n' a pas encore découvert tous ses voisins), noir lorsque ce sommet est fermé.
- Une liste Q utilisée comme pile (lifo) : on empile un sommet lorsqu'il est découvert, on le dépile lorsqu'il est terminé (traitement prioritaire des sommets découverts au plus tard).

Exemple d'exécution

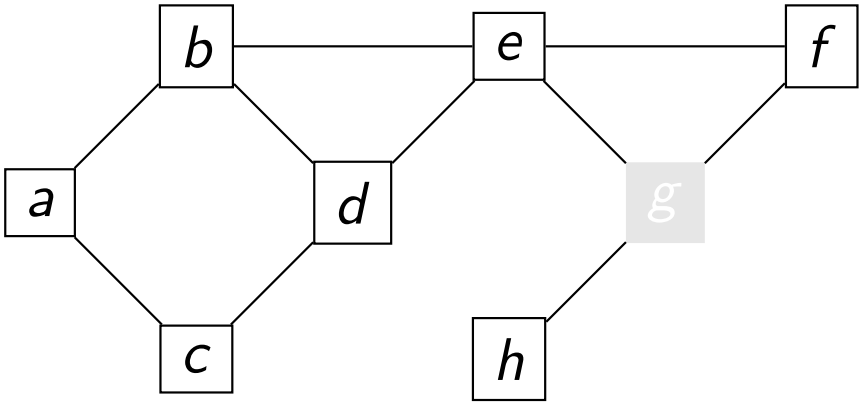
Déroulement attendu du programme, avec l'appel dfs(G,'g'), sur le graphe ci-dessous :

Python

```
G=dict()
G['a']=['b','c']
G['b']=['a','d','e']
G['c']=['a','d']
G['d']=['b','c','e']
G['e']=['b','d','f','g']
G['f']=['e','g']
G['g']=['e','f','h']
G['h']=['g']
```

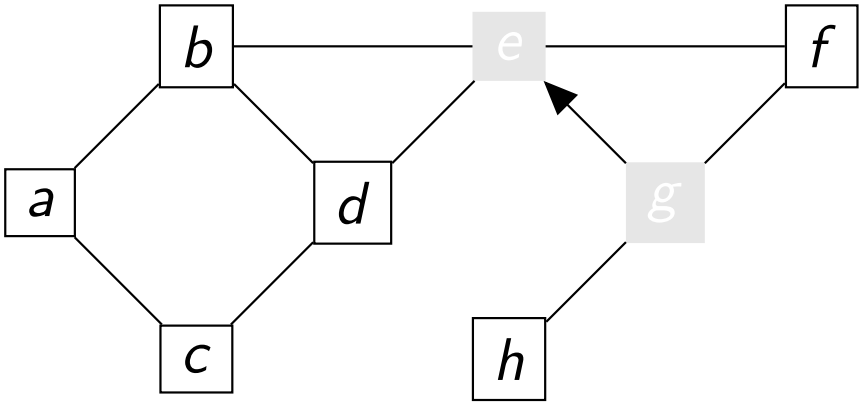


Déroulement



P={ g : None}
Q=[g]
Découverts (gris ou noirs)=[g]
Fermés (noirs) =[]

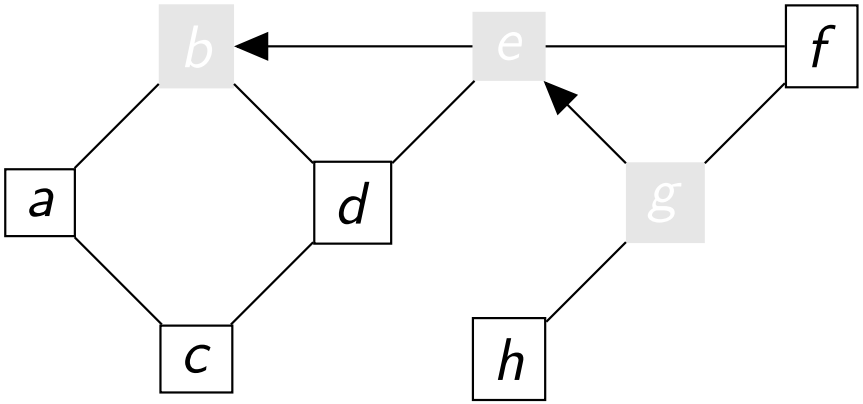
Déroulement



u=g, R=[e,f,h],v=e,

P={ g : None, e : g}
Q=[g,e]
Découverts=[g,e]
Fermés=[]

Déroulement



u=e, R=[b,d,f],v=b,

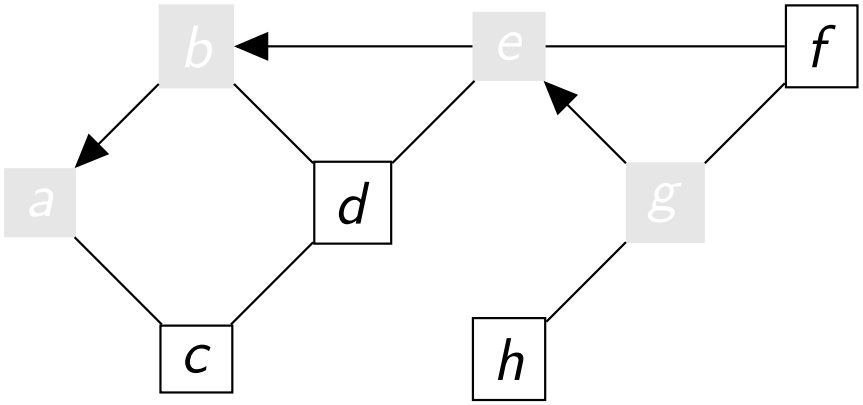
P={ g : None, e : g,b : e}

Q=[g,e,b]

Découverts=[g,e,b]

Fermés=[]

Déroulement



u=b, R=[a,d],v=a,

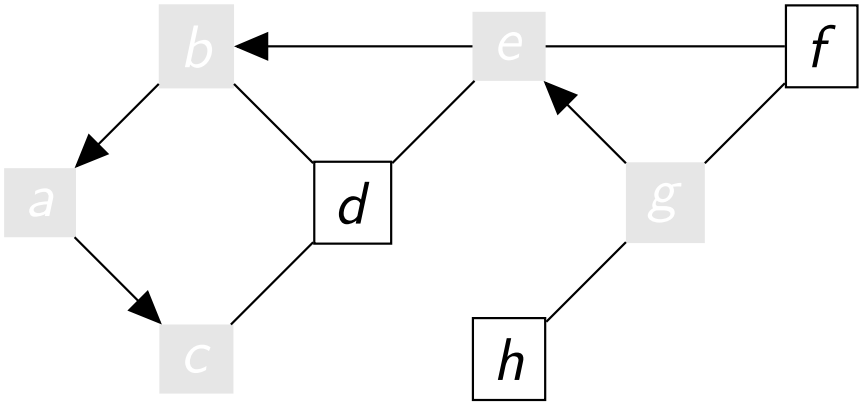
P={ g : None, e : g,b : e, a :b}

Q=[g,e,b,a]

Découverts=[g,e,b,a]

Fermés=[]

Déroulement



u=a, R=[c],v=c,

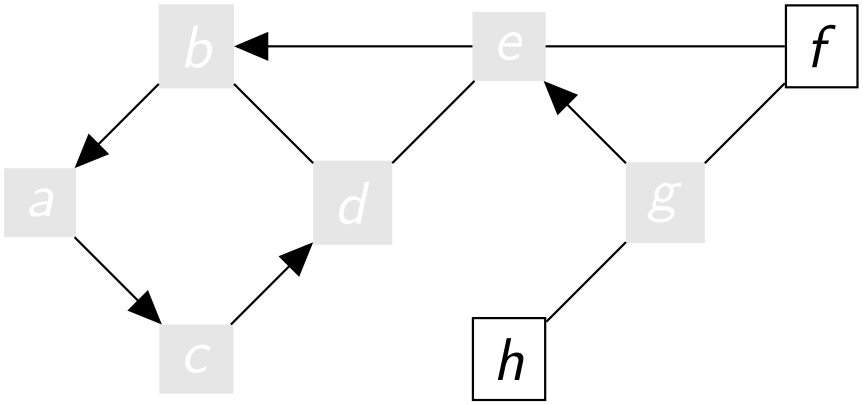
P={ g : None, e : g,b : e, a :b,c :a}

Q=[g,e,b,a,c]

Découverts=[g,e,b,a,c]

Fermés=[]

Déroulement



u=c, R=[d],v=d,

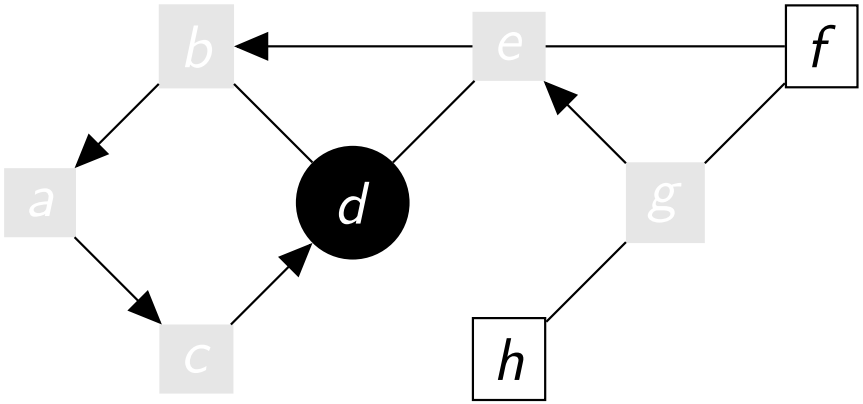
P={ g : None, e : g,b : e, a :b,c :a, d :c}

Q=[g,e,b,a,c,d]

Découverts=[g,e,b,a,c,d]

Fermés=[]

Déroulement



u=d, R=[],

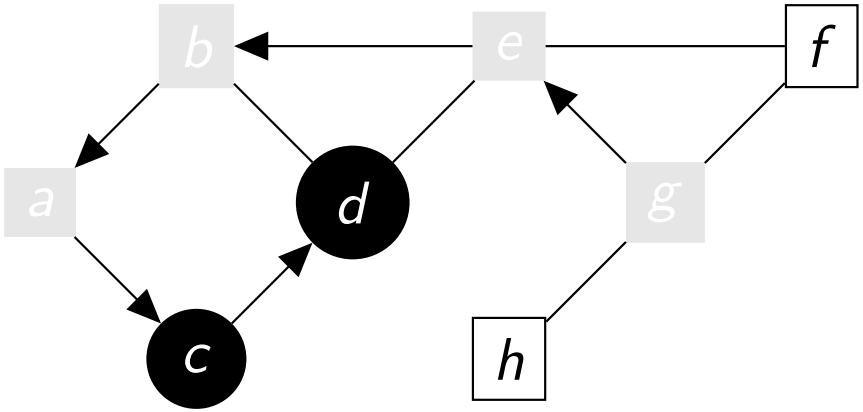
P={ g : None, e : g,b : e, a :b,c :a, d :c}

Q=[g,e,b,a,c]

Découverts=[g,e,b,a,c,d]

Fermés=[d]

Déroulement



u=c, R=[],

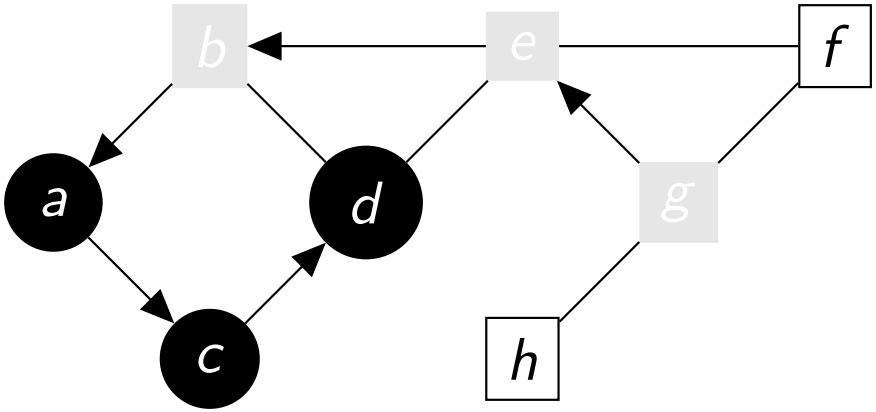
P={ g : None, e : g,b : e, a :b,c :a, d :c}

Q=[g,e,b,a]

Découverts=[g,e,b,a,c,d]

Fermés=[d,c]

Déroulement



u=a, R=[],

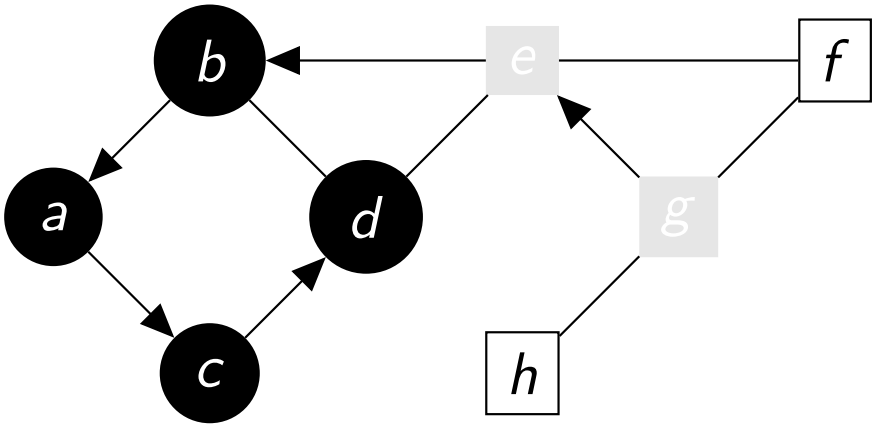
P={ g : None, e : g,b : e, a :b,c :a, d :c}

Q=[g,e,b]

Découverts=[g,e,b,a,c,d]

Fermés=[d,c,a]

Déroulement



u=b, R=[],

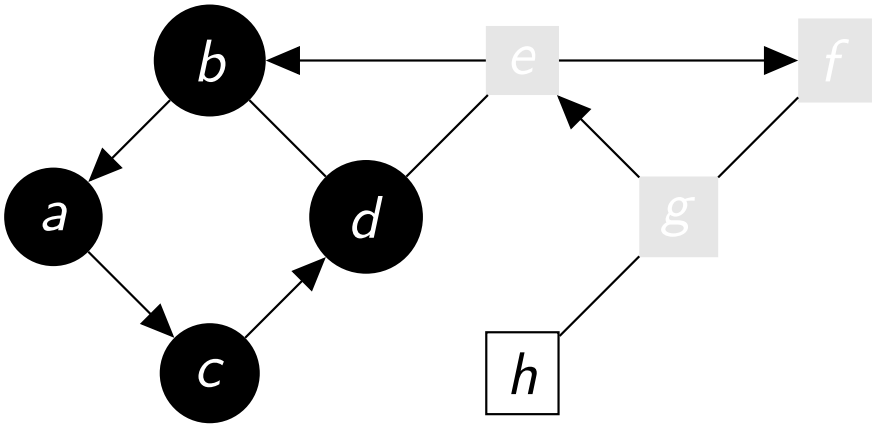
P={ g : None, e : g,b : e, a :b,c :a, d :c}

Q=[g,e]

Découverts=[g,e,b,a,c,d]

Fermés=[d,c,a,b]

Déroulement



u=e, R=[f], v=f,

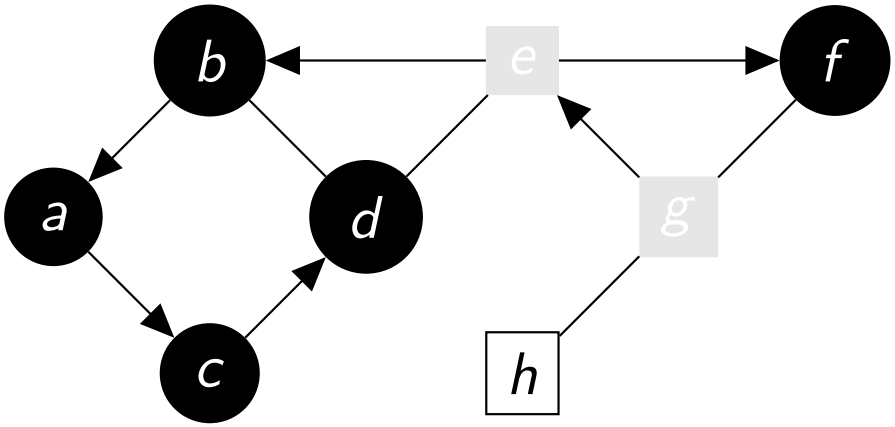
P={ g : None, e : g,b : e, a :b,c :a, d :c,f :e}

Q=[g,e,f]

Découverts=[g,e,b,a,c,d,f]

Fermés=[d,c,a,b]

Déroulement



u=f, R=[],

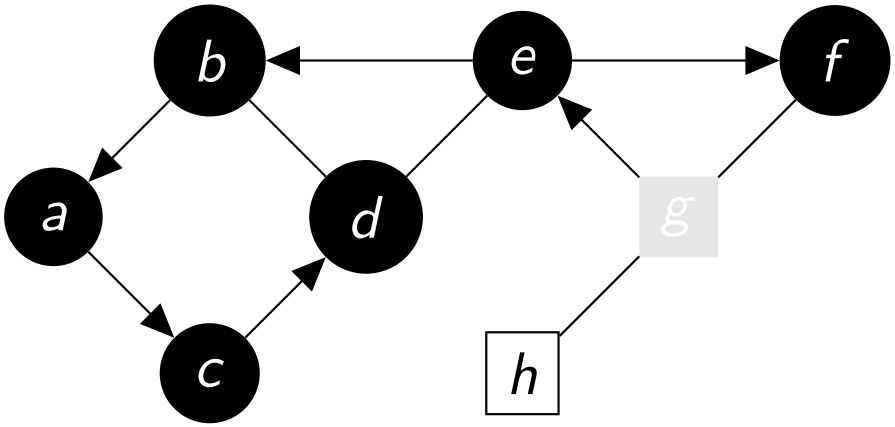
P={ g : None, e : g,b : e, a :b,c :a, d :c,f :e}

Q=[g,e]

Découverts=[g,e,b,a,c,d,f]

Fermés=[d,c,a,b,f]

Déroulement



u=e, R=[],

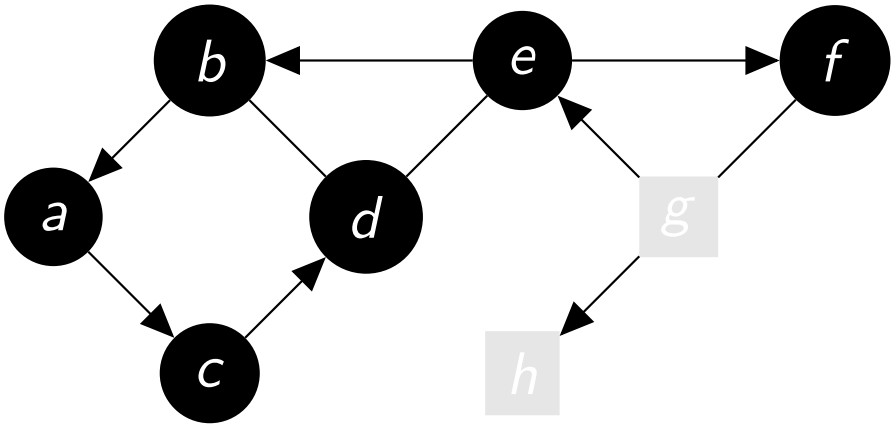
P={ g : None, e : g,b : e, a :b,c :a, d :c,f :e}

Q=[g]

Découverts=[g,e,b,a,c,d,f]

Fermés=[d,c,a,b,f,e]

Déroulement



u=g, R=[h], v=h,

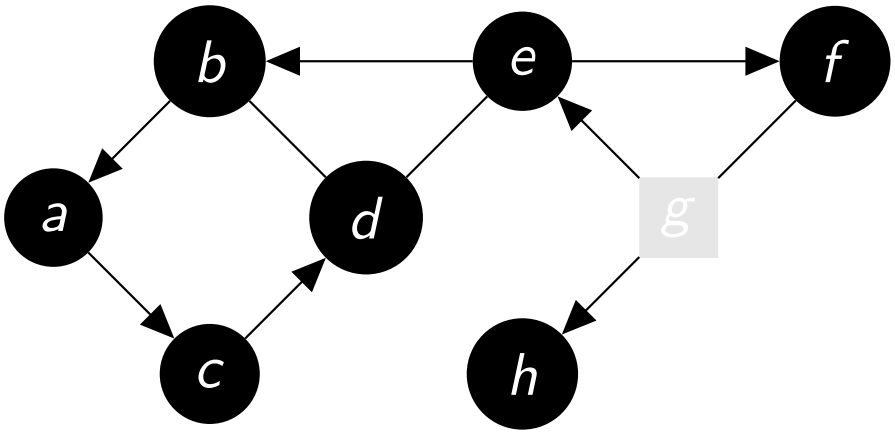
P={ g : None, e : g,b : e, a :b,c :a, d :c,f :e,h :g}

Q=[g,h]

Découverts=[g,e,b,a,c,d,f,h]

Fermés=[d,c,a,b,f,e]

Déroulement



u=h, R=[],

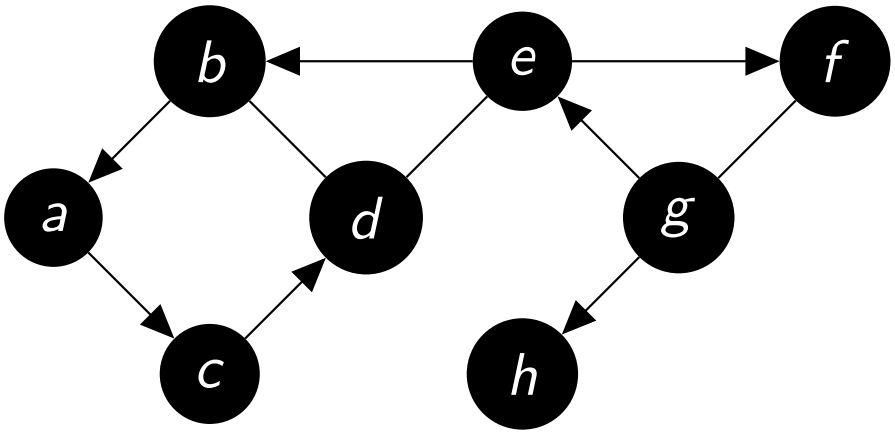
P={ g : None, e : g,b : e, a :b,c :a, d :c,f :e,h :g}

Q=[g]

Découverts=[g,e,b,a,c,d,f,h]

Fermés=[d,c,a,b,f,e,h]

Déroulement



u=g, R=[],

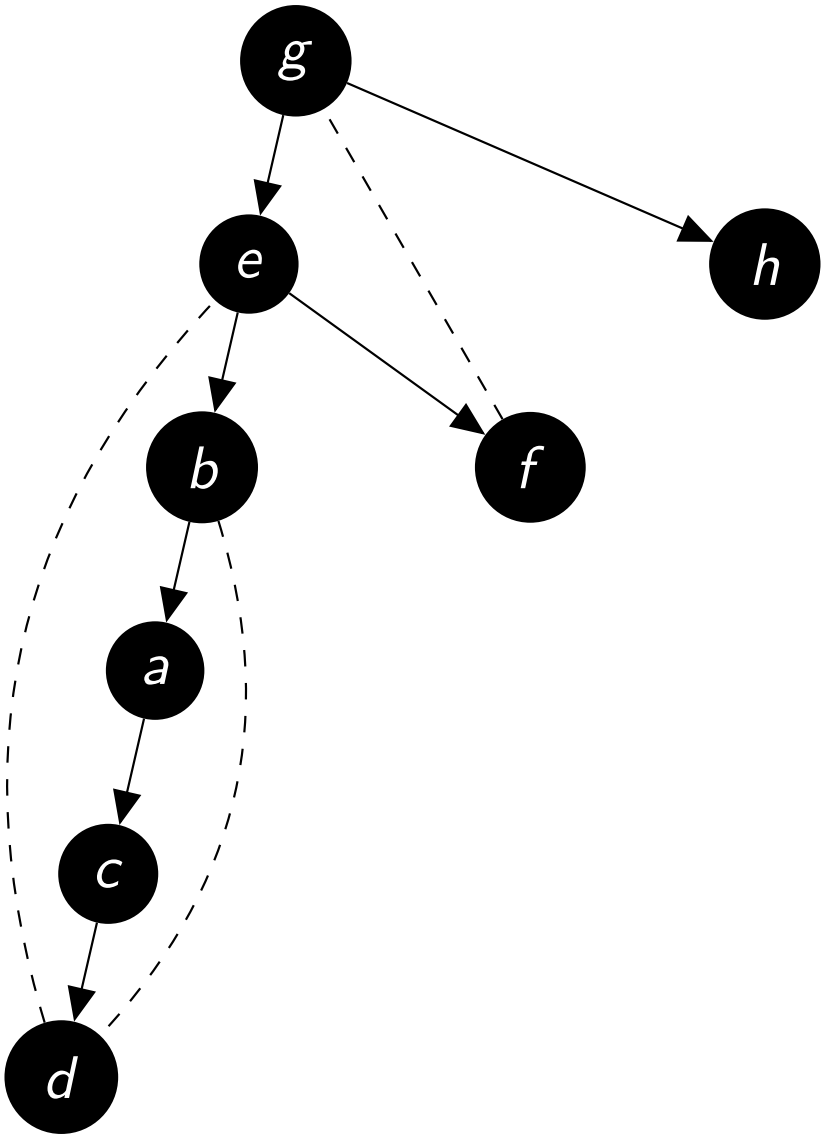
P={ g : None, e : g,b : e, a :b,c :a, d :c,f :e,h :g}

Q=[ ]

Découverts=[g,e,b,a,c,d,f,h]

Fermés=[d,c,a,b,f,e,h,g]

Arborescence associée au parcours



DFS : une programmation en python

Python

```
def dfs(G,s) :
    couleur=dict()
    for v in G :couleur[v]='blanc'
    P=dict()
    P[s]=None
    couleur[s]='gris'
    Q=[s]
    while Q :
        u=Q[-1]
        R=[y for y in G[u] if couleur[y]=='blanc']
        if R :
            v=R[0]
            couleur[v]='gris'
            P[v]=u
            Q.append(v)
        else :
            Q.pop()
            couleur[u]='noir'
    return P
```

DFS en python

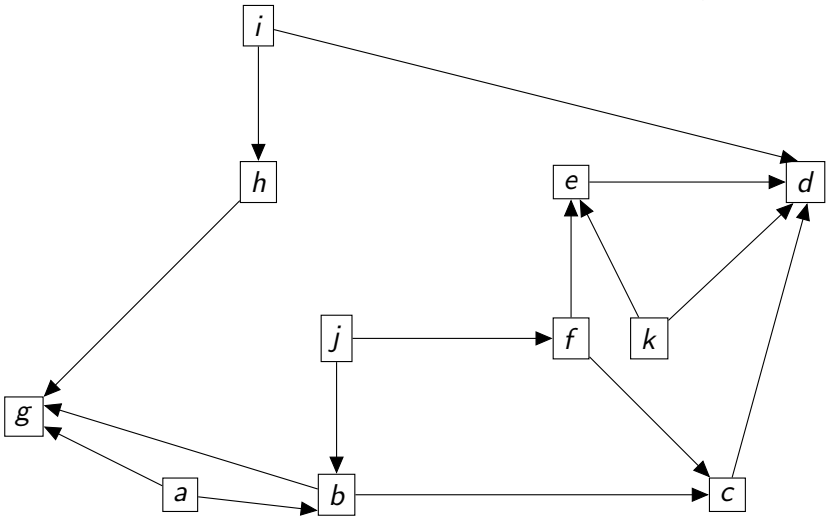
La même programmation, moins colorée

Python

```
def dfs(G,s) :
    P,Q={s :None},[s]
    while Q :
        u=Q[-1]
        R=[y for y in G[u] if y not in P]
        if R :
            v=random.choice(R)
            P[v]=u
            Q.append(v)
        else :
            Q.pop()
    return P
```

tri topologique

Exemple d'application du parcours en profondeur.
Début d'année, vos enseignants préparent le cours d'ISN.
Pour cela, ils découpent les notions en chapitres a, b, c, d...
Sur le graphe ci-dessous, une flèche du chapitre x vers le chapitre y signifie que x doit être traité avant y.



Pour une meilleure organisation et savoir où commencer , on aimerait réorganiser la représentation du graphe de façon à ce que tous les sommets soient dessinés alignés, toutes les flèches du schéma devant être orientées vers la droite.

tri topologique

L'énumération des sommets du graphe en ordre inverse de leur date de fermeture dans l'algorithme de parcours en profondeur permet d'obtenir un ordre des sommets satisfaisant la demande.

Python

```
def lancement(G) :
    for s in G :
        if couleur[s]=='blanc' : parcours(G,s)
def parcours(G,s) :
    couleur[s]='gris'
    for v in G[s] :
        if couleur[v]=='blanc' : parcours(G,v)
    P.append(s)
# # # # # #
couleur=dict()
for v in G :couleur[v]='blanc'
P=list()
lancement(G)
P.reverse()
print P
```

tri topo python