Algorithmique des graphes

2 — Parcours des graphes orientés

Marie-Pierre Béal (Cours d'Anthony Labarre)

Université
Gustave Eiffel

Aperçu

- Une des tâches les plus basiques pour n'importe quelle structure de données consiste à la parcourir ;

Aperçu

- Une des tâches les plus basiques pour n'importe quelle structure de données consiste à la parcourir ;
- Les graphes se parcourent principalement de deux façons : en *profondeur* ou en *largeur* ;

Aperçu

- Une des tâches les plus basiques pour n'importe quelle structure de données consiste à la parcourir ;
- Les graphes se parcourent principalement de deux façons : en *profondeur* ou en *largeur* ;
- Le choix du type de parcours dépendra de ce qu'on veut en faire ;

Aperçu

- Une des tâches les plus basiques pour n'importe quelle structure de données consiste à la parcourir ;
- Les graphes se parcourent principalement de deux façons : en *profondeur* ou en *largeur* ;
- Le choix du type de parcours dépendra de ce qu'on veut en faire ;
- Un nombre surprenant d'algorithmes résolvant des problèmes très variés sont de simples variantes de ces parcours.

Échauffement : parcours d'arbres

- En guise d'échauffement, examinons les parcours sur des arbres (binaires) ;

Échauffement : parcours d'arbres

- En guise d'échauffement, examinons les parcours sur des arbres (binaires) ;
- Les arbres peuvent être vus comme des graphes orientés très particuliers :

Échauffement : parcours d'arbres

- En guise d'échauffement, examinons les parcours sur des arbres (binaires) ;
- Les arbres peuvent être vus comme des graphes orientés très particuliers :
  1. ils ne possèdent pas de cycles ;

Échauffement : parcours d'arbres

- En guise d'échauffement, examinons les parcours sur des arbres (binaires) ;
- Les arbres peuvent être vus comme des graphes orientés très particuliers :
  1. ils ne possèdent pas de cycles ;
  2. le point de départ est fixé (c'est la racine) ;

Échauffement : parcours d'arbres

- En guise d'échauffement, examinons les parcours sur des arbres (binaires) ;
- Les arbres peuvent être vus comme des graphes orientés très particuliers :
  1. ils ne possèdent pas de cycles ;
  2. le point de départ est fixé (c'est la racine) ;
- On les parcourt fréquemment de deux manières :

Échauffement : parcours d'arbres

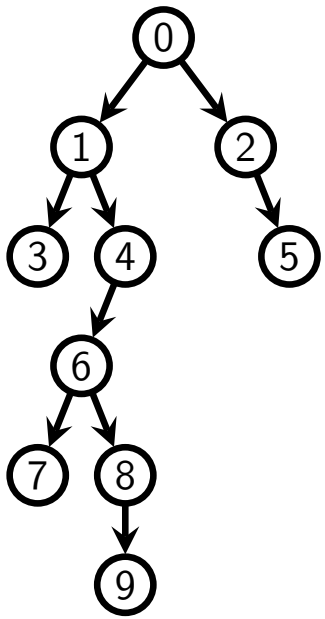
- En guise d'échauffement, examinons les parcours sur des arbres (binaires) ;
- Les arbres peuvent être vus comme des graphes orientés très particuliers :
  1. ils ne possèdent pas de cycles ;
  2. le point de départ est fixé (c'est la racine) ;
- On les parcourt fréquemment de deux manières :
  - en **profondeur** : la racine, puis le sous-arbre gauche, puis le sous-arbre droit ;

Échauffement : parcours d'arbres

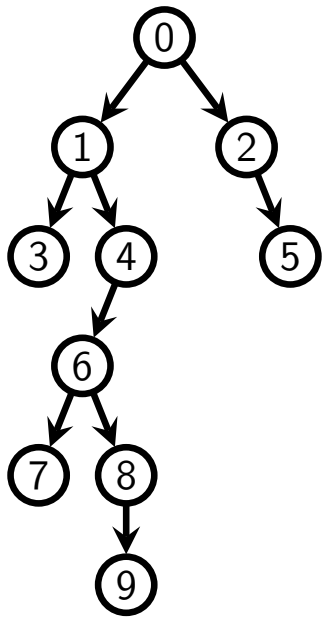
- En guise d'échauffement, examinons les parcours sur des arbres (binaires) ;
- Les arbres peuvent être vus comme des graphes orientés très particuliers :
  1. ils ne possèdent pas de cycles ;
  2. le point de départ est fixé (c'est la racine) ;
- On les parcourt fréquemment de deux manières :
  - en **profondeur** : la racine, puis le sous-arbre gauche, puis le sous-arbre droit ;
  - en **largeur** : la racine, puis ses fils, puis les fils de ces fils, ...

Parcours d'arbres en profondeur et en largeur

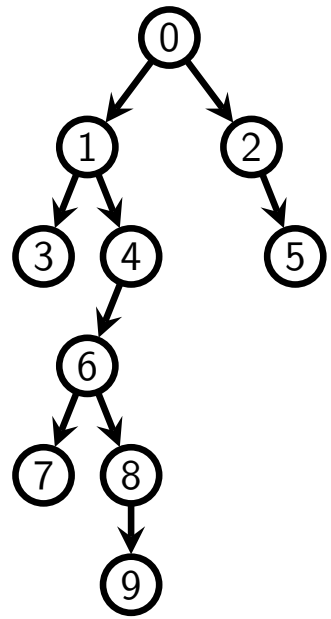
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

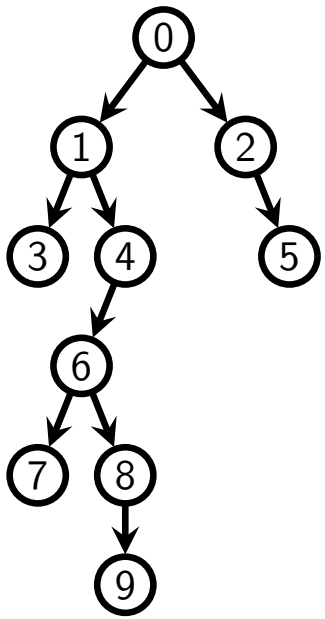


en largeur

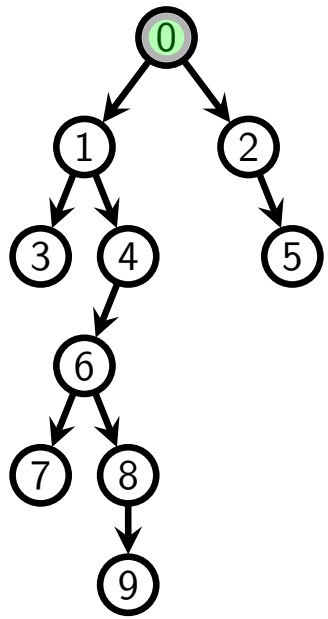
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

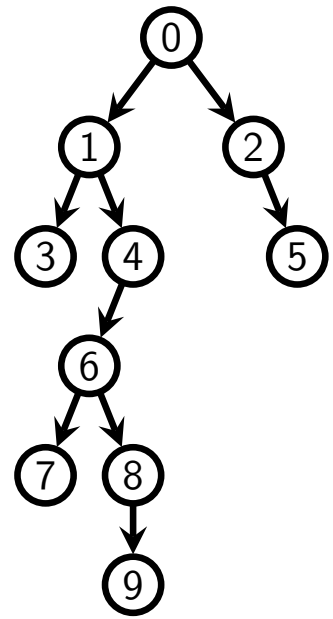
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

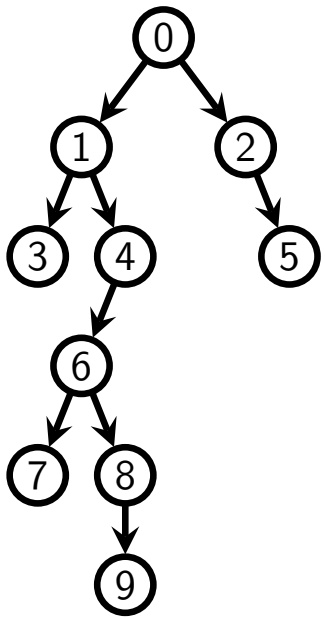


en largeur

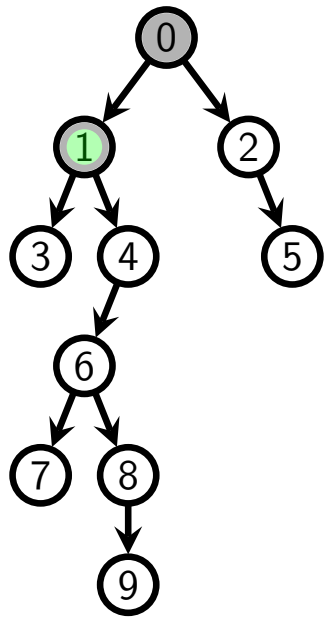
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

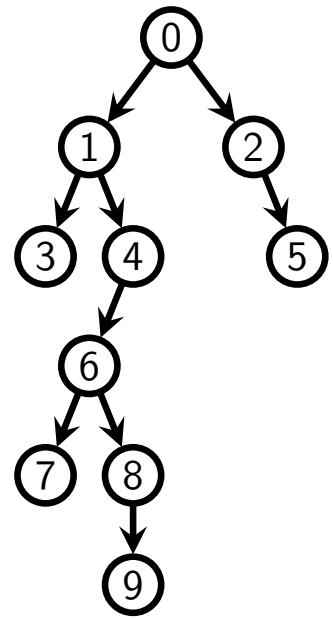
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

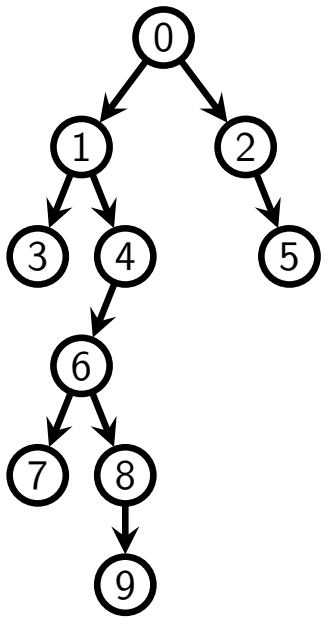


en largeur

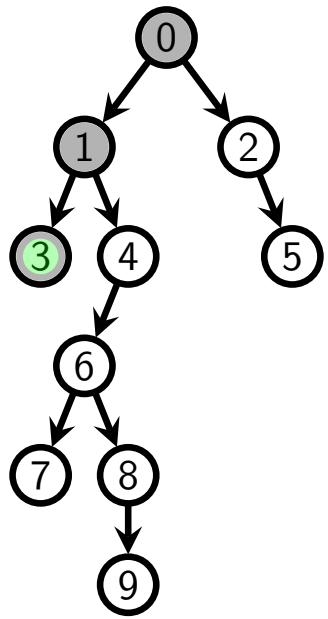
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

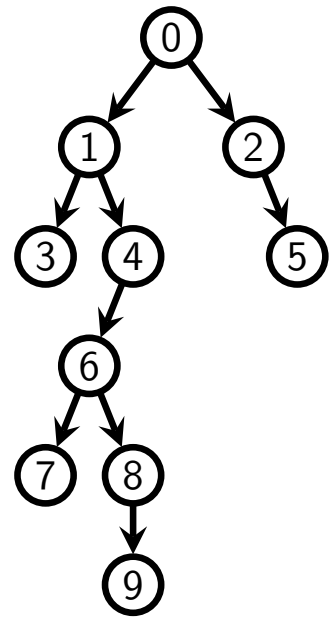
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

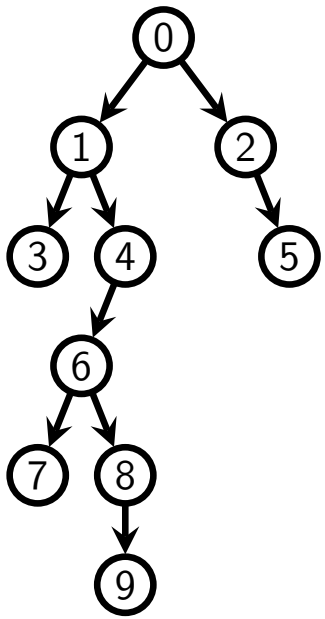


en largeur

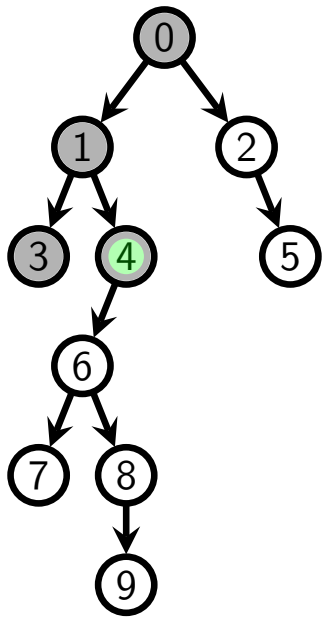
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

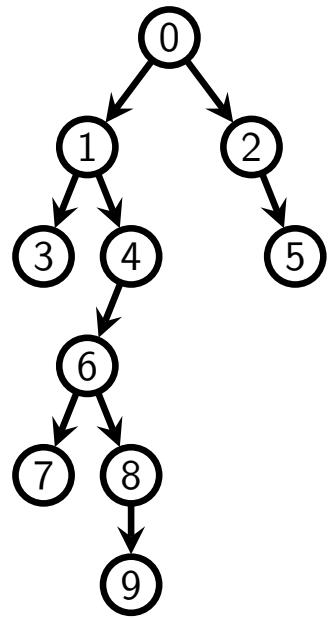
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

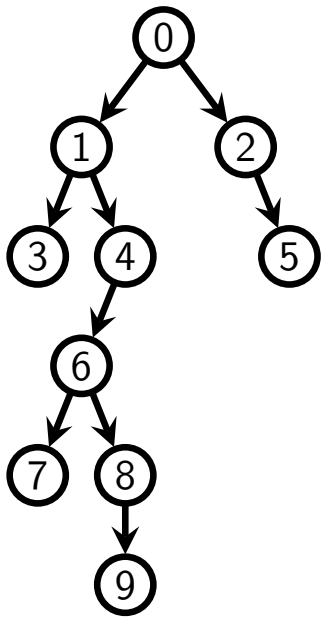


en largeur

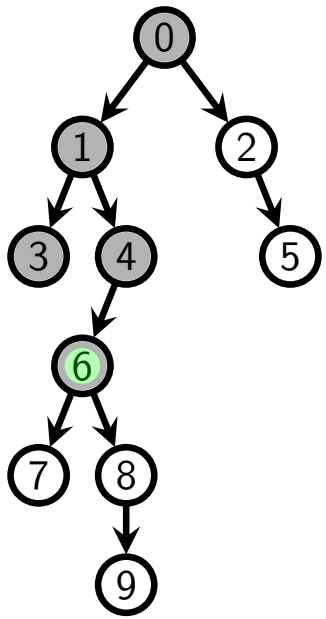
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

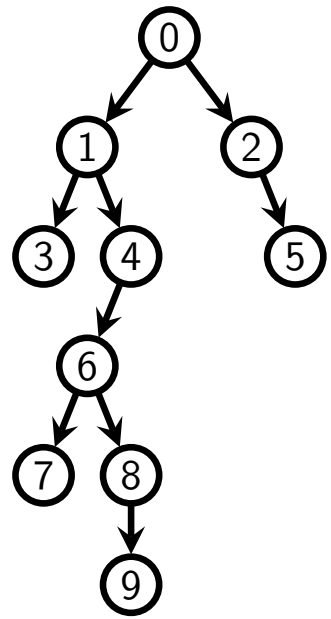
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

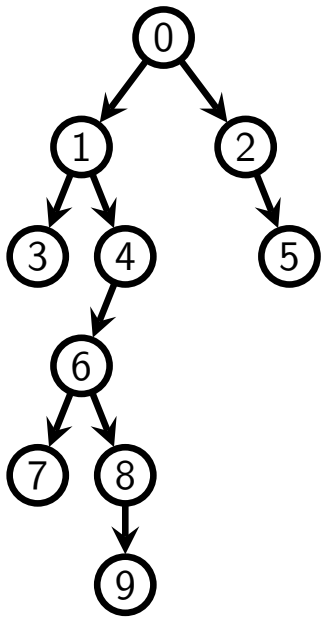


en largeur

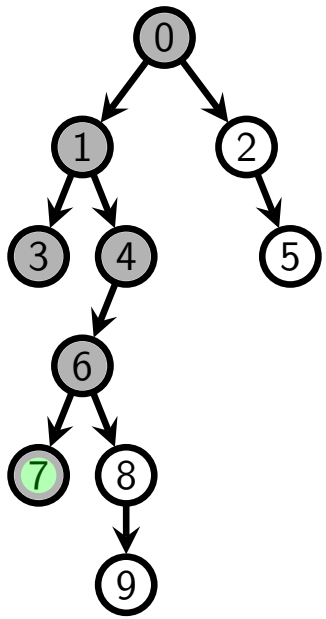
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

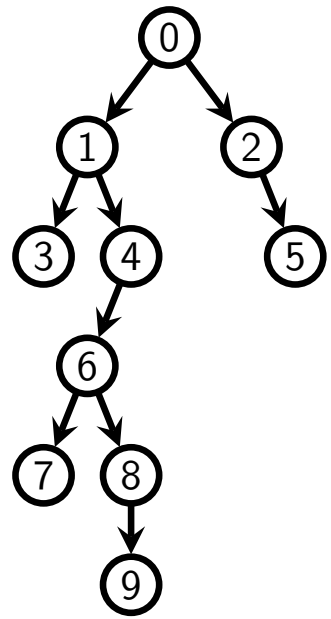
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

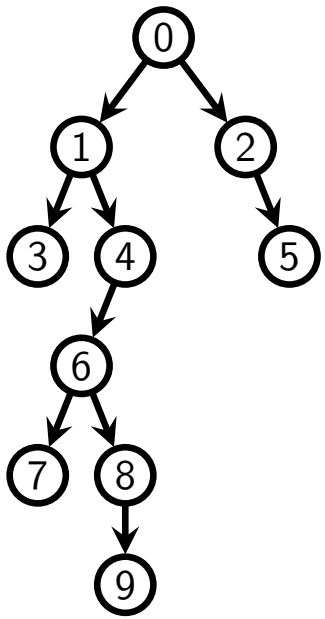


en largeur

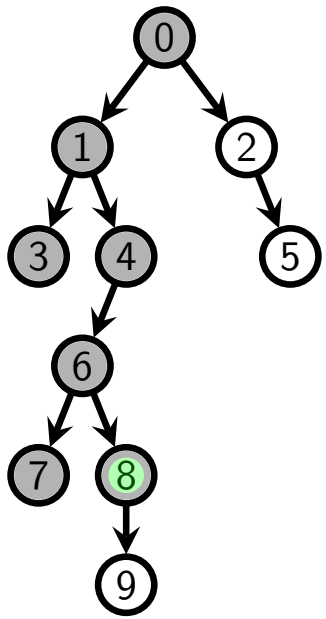
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

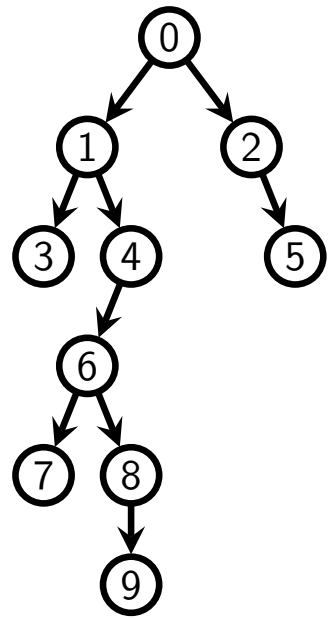
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

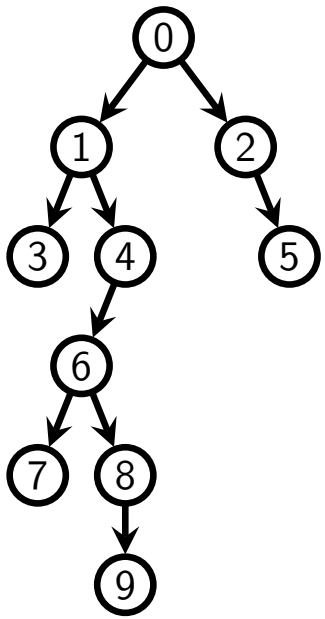


en largeur

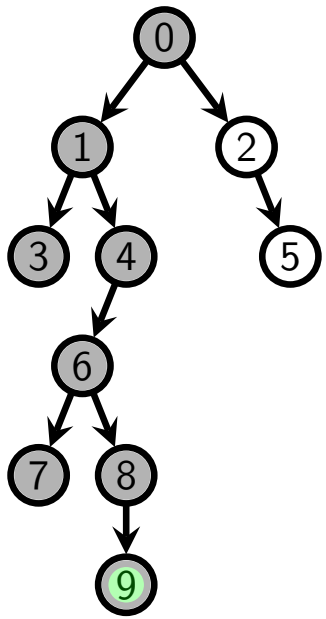
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

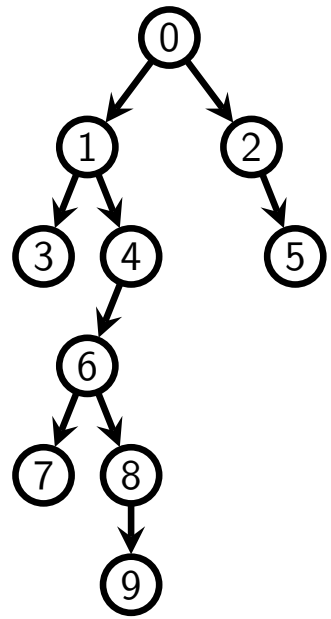
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

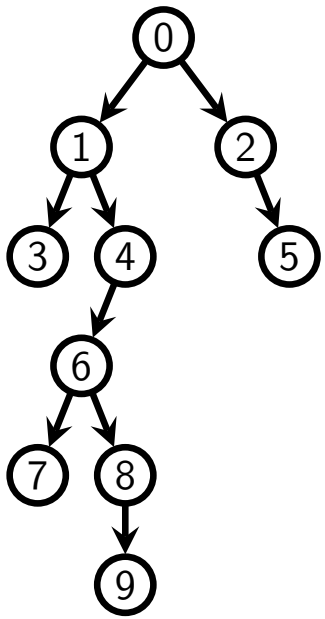


en largeur

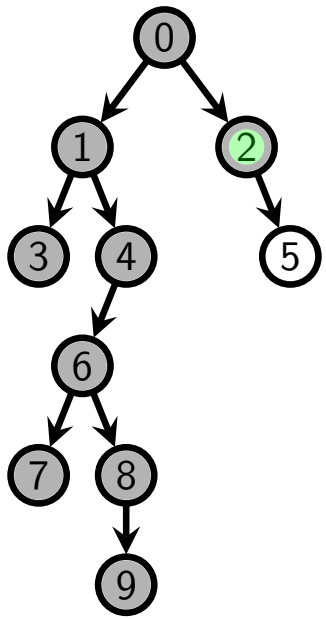
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

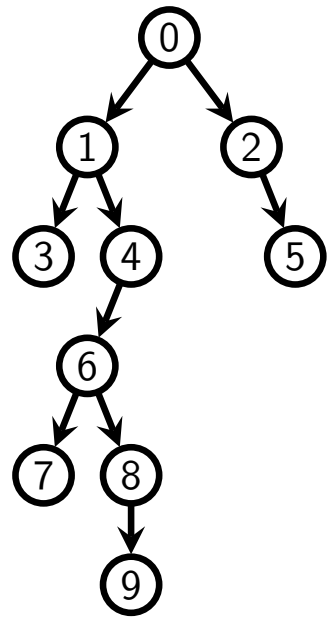
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

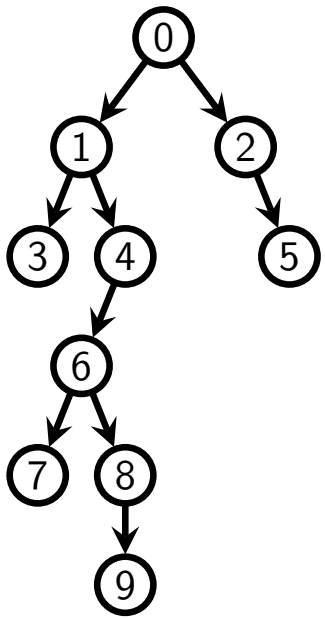


en largeur

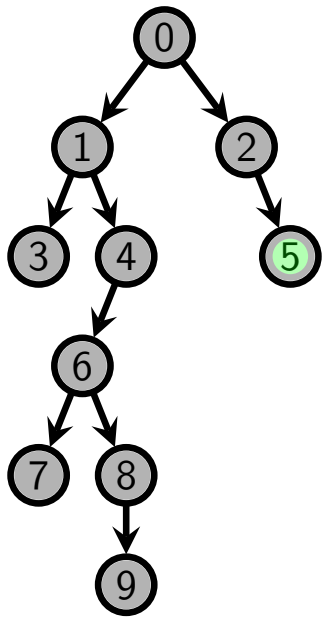
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

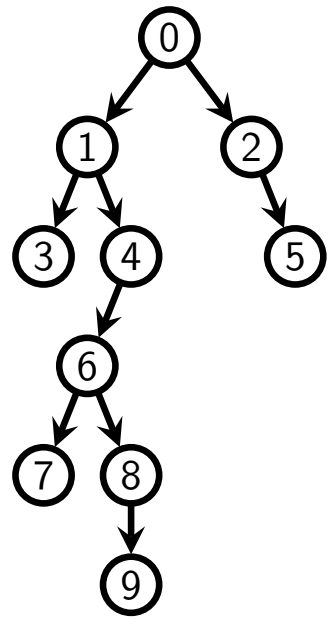
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

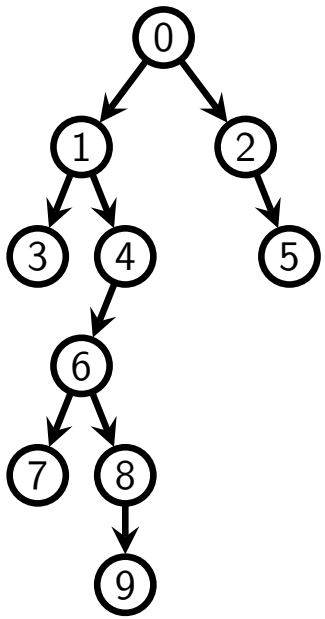


en largeur

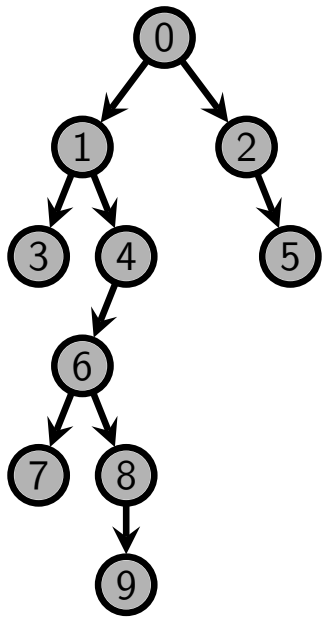
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

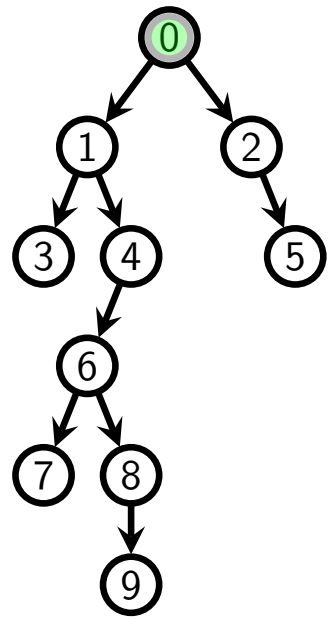
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

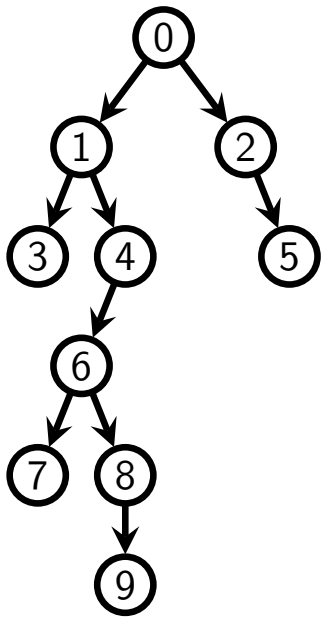


en largeur

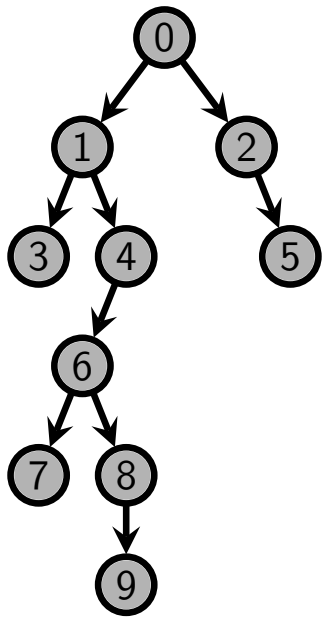
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

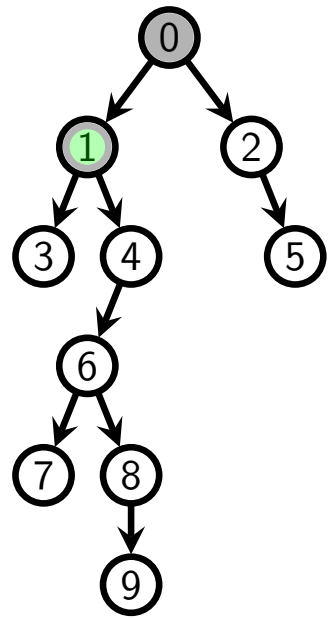
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

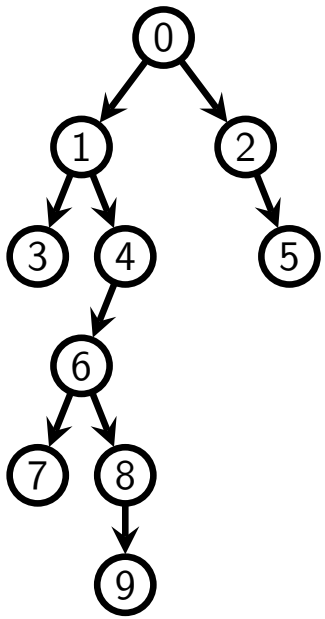


en largeur

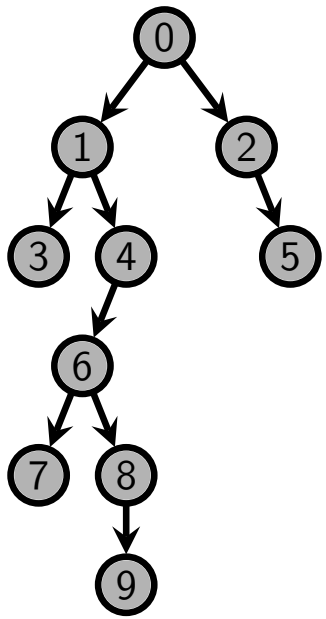
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

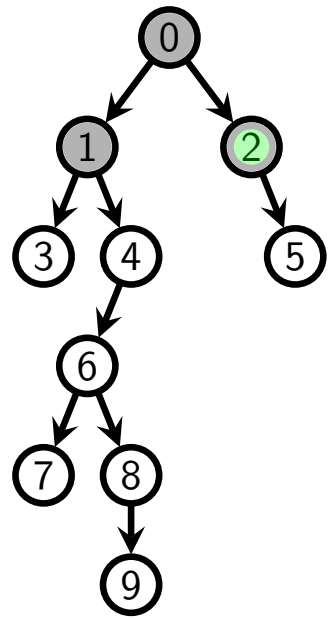
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

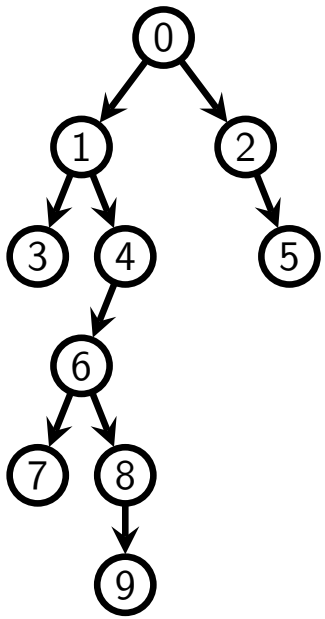


en largeur

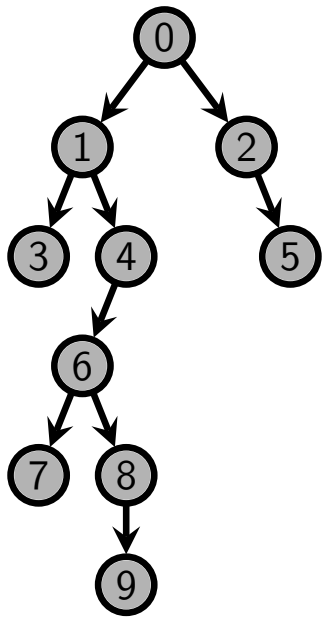
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

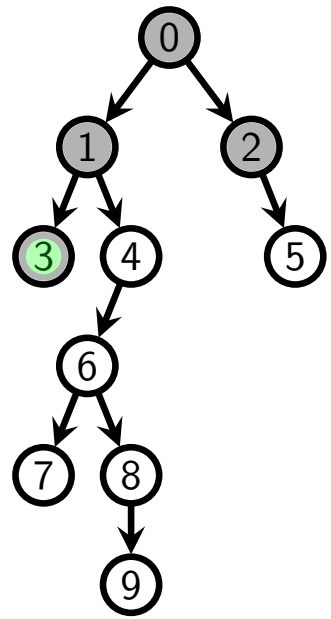
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

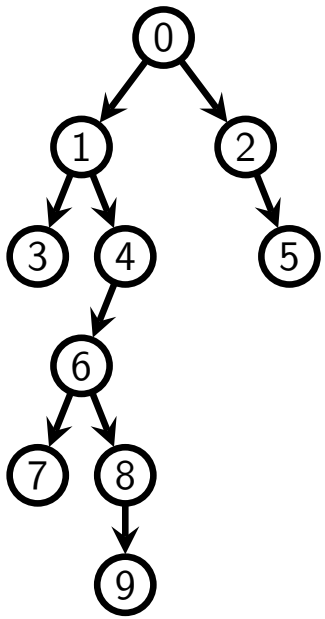


en largeur

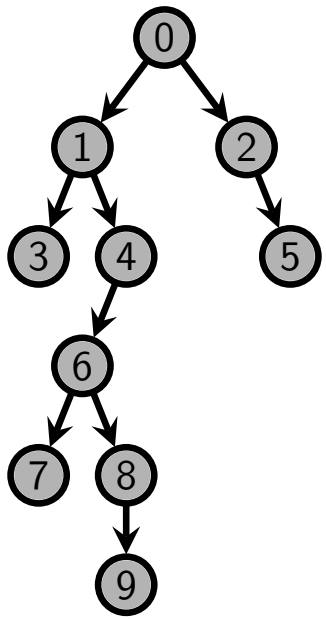
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

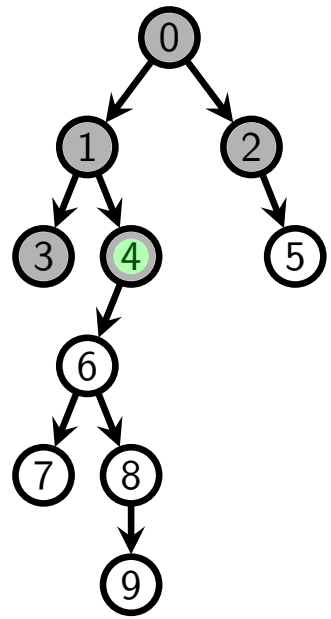
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

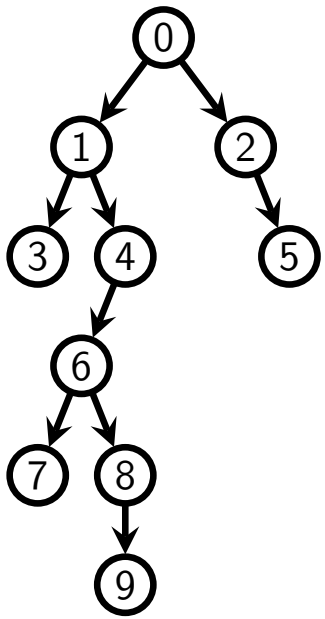


en largeur

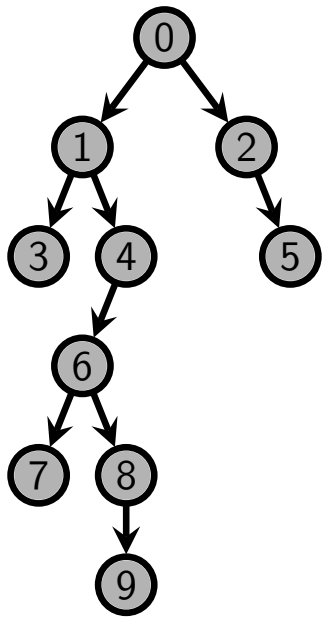
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

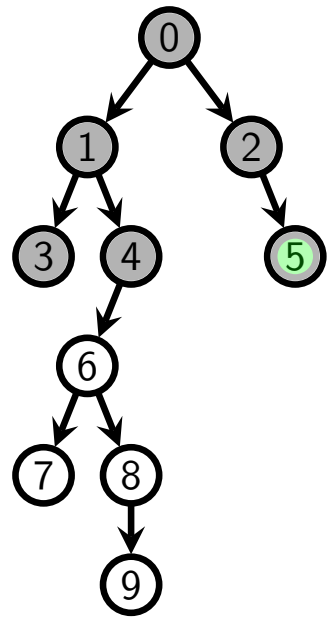
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

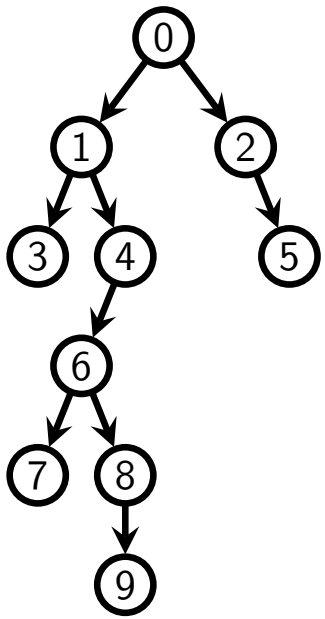


en largeur

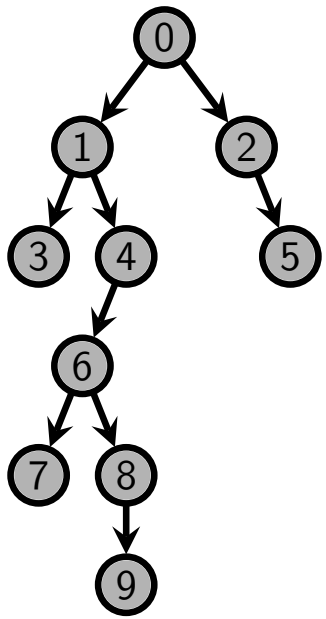
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

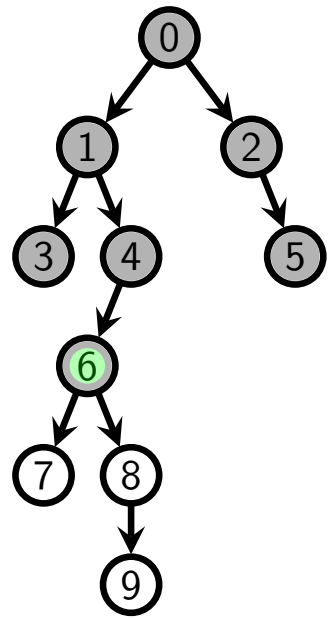
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

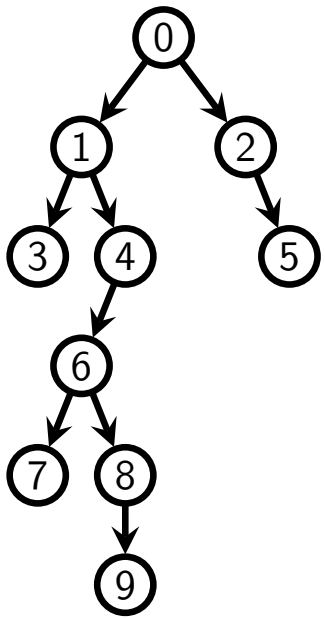


en largeur

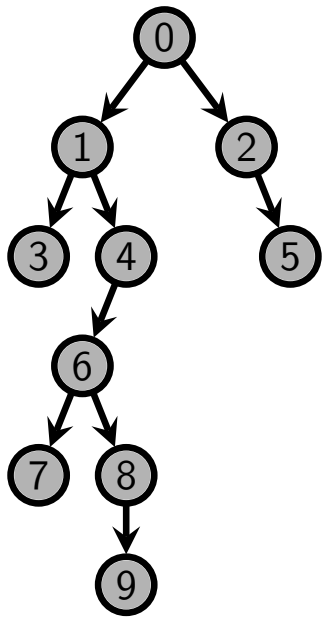
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

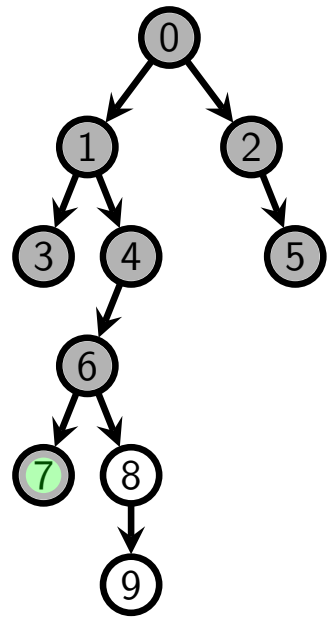
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

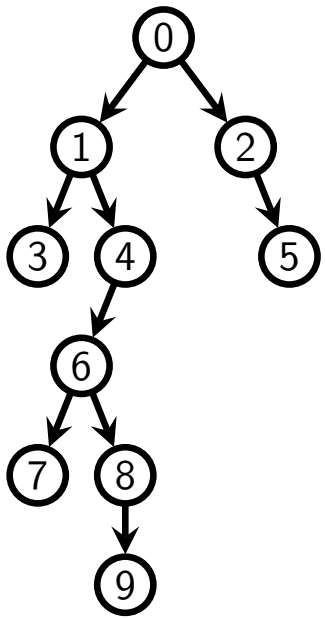


en largeur

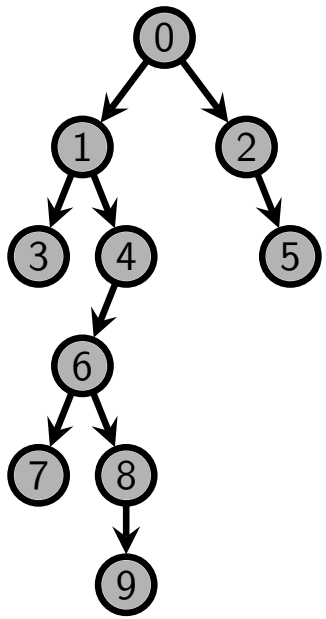
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

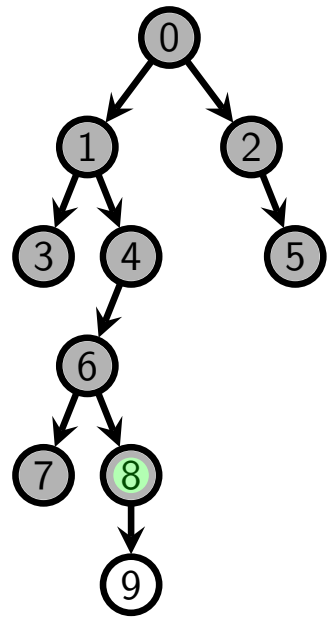
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

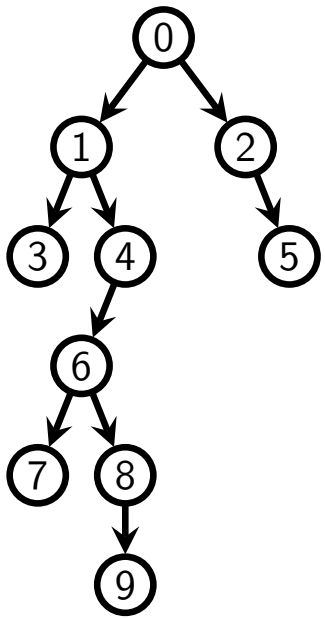


en largeur

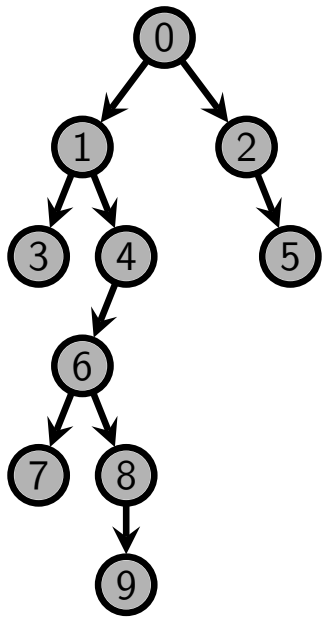
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

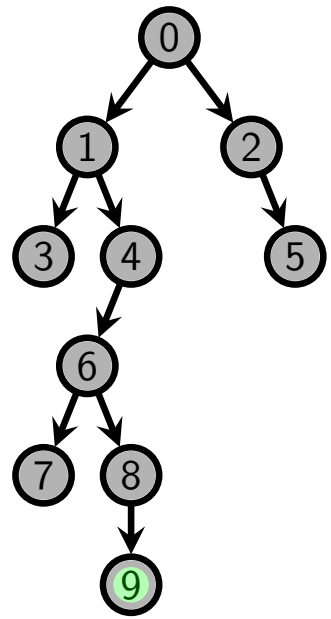
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

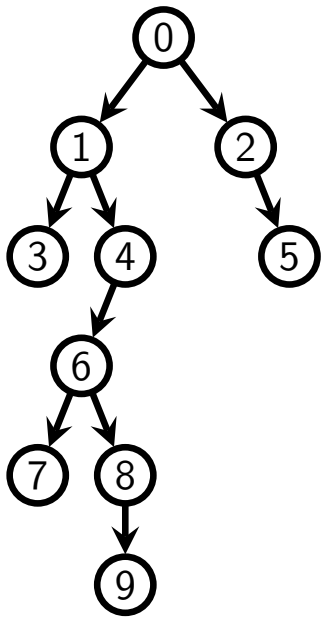


en largeur

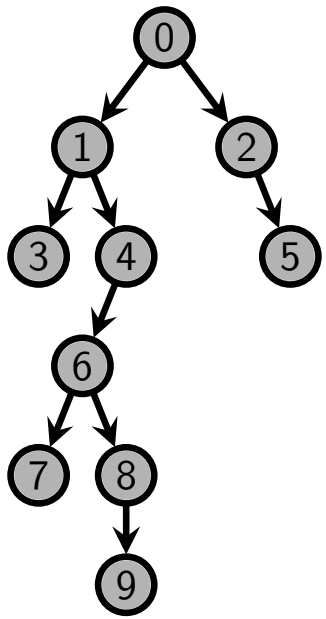
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

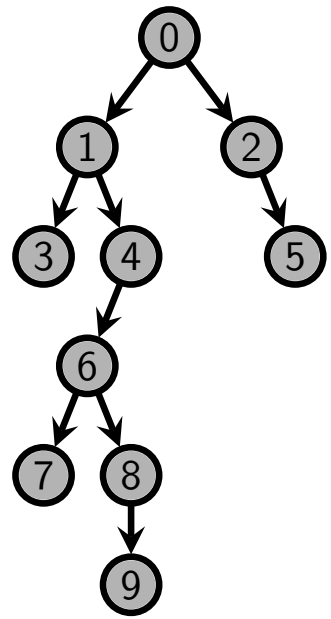
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur



en largeur

- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Piles et files

On aura besoin dans la suite de deux structures de données bien connues :

Piles et files

On aura besoin dans la suite de deux structures de données bien connues :

Pile

Last **I**n **F**irst **O**ut

- empiler(x) : rajoute x en haut de la pile ;
- dépiler() : retire et renvoie le haut de la pile ;

Piles et files

On aura besoin dans la suite de deux structures de données bien connues :

Pile

Last **I**n **F**irst **O**ut

- empiler(x) : rajoute x en haut de la pile ;
- dépiler() : retire et renvoie le haut de la pile ;

File

First **I**n **F**irst **O**ut

- enfiler(x) : rajoute x à la fin de la file ;
- défiler() : retire et renvoie le début de la file ;

Piles et files

On aura besoin dans la suite de deux structures de données bien connues :

Pile

Last **I**n **F**irst **O**ut

- empiler(x) : rajoute x en haut de la pile ;
- dépiler() : retire et renvoie le haut de la pile ;

File

First **I**n **F**irst **O**ut

- enfiler(x) : rajoute x à la fin de la file ;
- défiler() : retire et renvoie le début de la file ;

- Les deux classes possèdent aussi les méthodes est_vide() et pas_vide() ;

Piles et files

On aura besoin dans la suite de deux structures de données bien connues :

Pile

Last **I**n **F**irst **O**ut

- empiler(x) : rajoute x en haut de la pile ;
- dépiler() : retire et renvoie le haut de la pile ;

File

First **I**n **F**irst **O**ut

- enfiler(x) : rajoute x à la fin de la file ;
- défiler() : retire et renvoie le début de la file ;

- Les deux classes possèdent aussi les méthodes est_vide() et pas_vide() ;
- Toutes ces méthodes s'exécutent en $O(1)$;

Piles et files

On aura besoin dans la suite de deux structures de données bien connues :

Pile

Last **I**n **F**irst **O**ut

- empiler(x) : rajoute x en haut de la pile ;
- dépiler() : retire et renvoie le haut de la pile ;

File

First **I**n **F**irst **O**ut

- enfiler(x) : rajoute x à la fin de la file ;
- défiler() : retire et renvoie le début de la file ;

- Les deux classes possèdent aussi les méthodes est_vide() et pas_vide() ;
- Toutes ces méthodes s'exécutent en $O(1)$;
- Pour être plus concis, on suppose que empiler() et enfiler() acceptent aussi plusieurs éléments ;

Parcours d'arbres en profondeur

Le parcours d'arbre en profondeur s'écrit facilement de manière récursive :

Algorithme 1 : ProfondeurArbre(A)

Entrées : un arbre binaire enraciné A.
Résultat : l'affichage des sommets de A suivant un parcours en profondeur à partir de la racine.

```
1 si A.racine() ≠ NIL alors
2 |   afficher(A.racine());
3 |   ProfondeurArbre(A.sous_arbre_gauche());
4 |   ProfondeurArbre(A.sous_arbre_droit());
```

Parcours d'arbre en largeur

Algorithme 2 : PARCOURSLARGEURARBRE(A)

Entrées : un arbre enraciné A.
Sortie : la liste des sommets de l'arbre ordonné selon un parcours en largeur à partir de la racine.

```
1 a_traiter ← file();
2 résultat ← liste();
3 a_traiter.enfiler(A.racine());
4 tant que a_traiter.pas_vide() faire
5 |   sommet ← a_traiter.défiler();
6 |   résultat.ajouter_en_fin(sommet);
7 |   pour chaque fils dans A.fils(sommet) faire
8 |   |   a_traiter.enfiler(fils);
9 renvoyer résultat;
```

Parcours de graphes orientés

Parcourir un graphe présente plusieurs difficultés par rapport aux arbres (binaires ou non) :

1. tout sommet peut servir de point de départ ;

Parcours de graphes orientés

Parcourir un graphe présente plusieurs difficultés par rapport aux arbres (binaires ou non) :

1. tout sommet peut servir de point de départ ;
2. il n'y a pas d'ordre sur les successeurs ;

Parcours de graphes orientés

Parcourir un graphe présente plusieurs difficultés par rapport aux arbres (binaires ou non) :

1. tout sommet peut servir de point de départ ;
2. il n'y a pas d'ordre sur les successeurs ;
3. certains sommets sont accessibles par plusieurs chemins, il faudra donc se rappeler de ce qu'on a déjà examiné. Il peut y avoir des circuits. Il ne faut pas boucler indéfiniment.

Parcours de graphes orientés

Parcourir un graphe présente plusieurs difficultés par rapport aux arbres (binaires ou non) :

1. tout sommet peut servir de point de départ ;
2. il n'y a pas d'ordre sur les successeurs ;
3. certains sommets sont accessibles par plusieurs chemins, il faudra donc se rappeler de ce qu'on a déjà examiné. Il peut y avoir des circuits. Il ne faut pas boucler indéfiniment.

Conventions : on suppose que :

- **la méthode G.successeurs(v) renvoie les successeurs de v par ordre croissant d'identifiant ;**
- **en cas d'ambiguïté, on sélectionne les sommets d'indice minimal.**

Parcours de graphe orienté en profondeur

- Le principe du parcours en profondeur des arbres se généralise comme suit aux graphes orientés :

Parcours de graphe orienté en profondeur

- Le principe du parcours en profondeur des arbres se généralise comme suit aux graphes orientés :
  1. si le sommet actuel u n'a pas encore été visité, l'afficher ;

Parcours de graphe orienté en profondeur

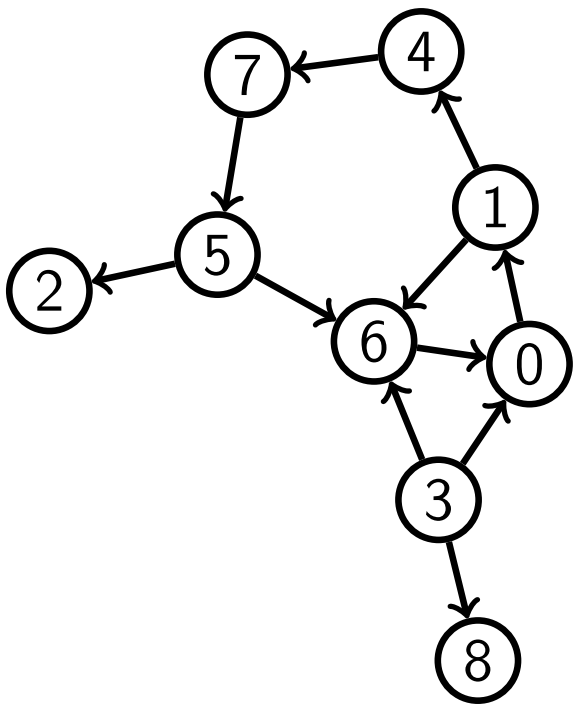
- Le principe du parcours en profondeur des arbres se généralise comme suit aux graphes orientés :
  1. si le sommet actuel u n'a pas encore été visité, l'afficher ;
  2. pour chaque successeur v non encore visité de u : parcourir v et ses descendants en profondeur.

Parcours de graphe orienté en profondeur

- Le principe du parcours en profondeur des arbres se généralise comme suit aux graphes orientés :
  1. si le sommet actuel u n'a pas encore été visité, l'afficher ;
  2. pour chaque successeur v non encore visité de u : parcourir v et ses descendants en profondeur.
- Examinons les étapes de ce parcours sur un exemple.

Parcours en profondeur : exemple

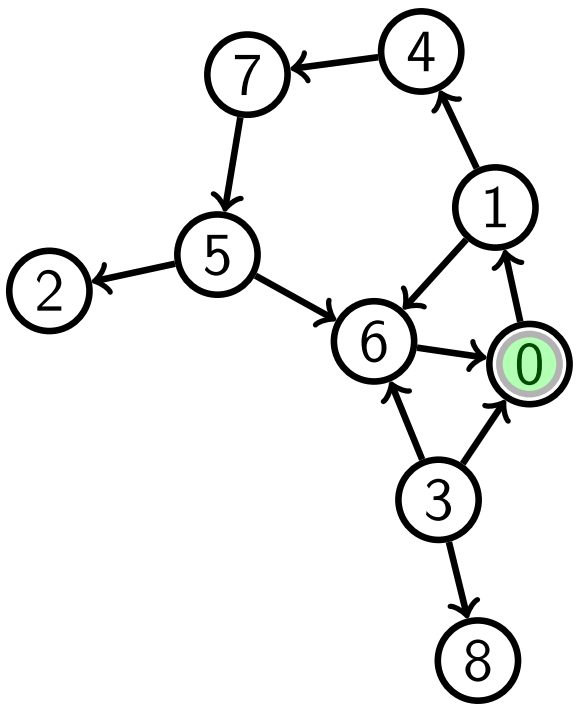
Exemple 2



Ordre de l'exploration

Parcours en profondeur : exemple

Exemple 2

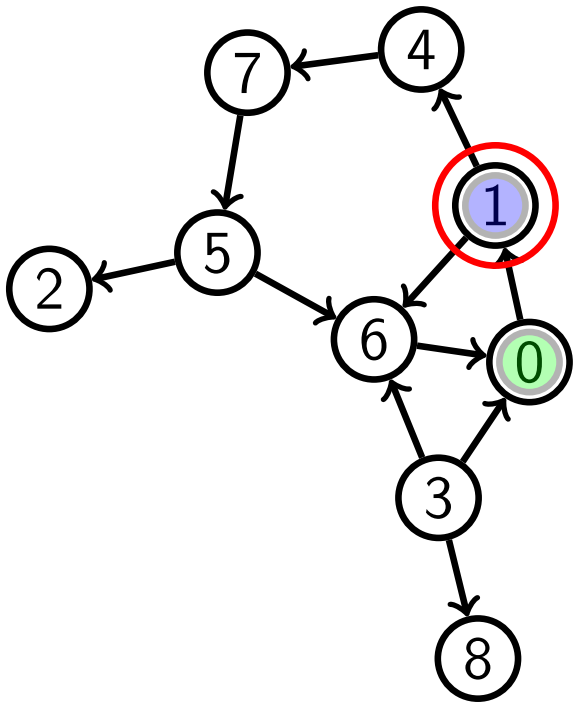


Ordre de l'exploration

0

Parcours en profondeur : exemple

Exemple 2

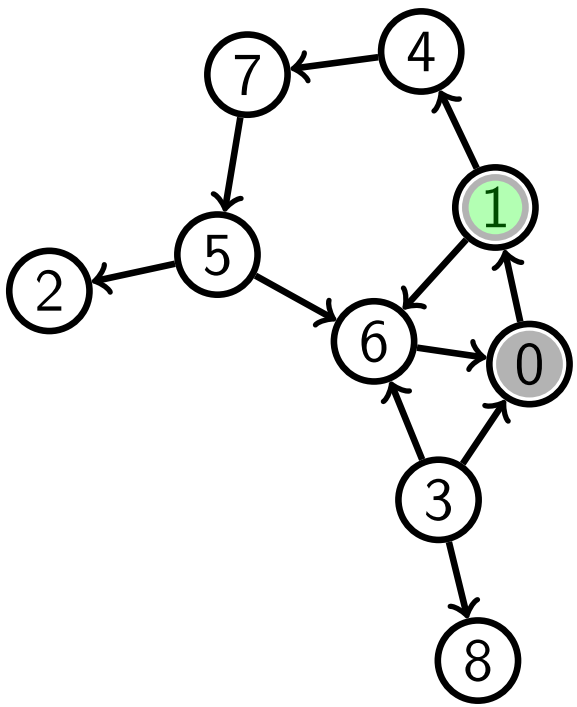


Ordre de l'exploration

0

Parcours en profondeur : exemple

Exemple 2

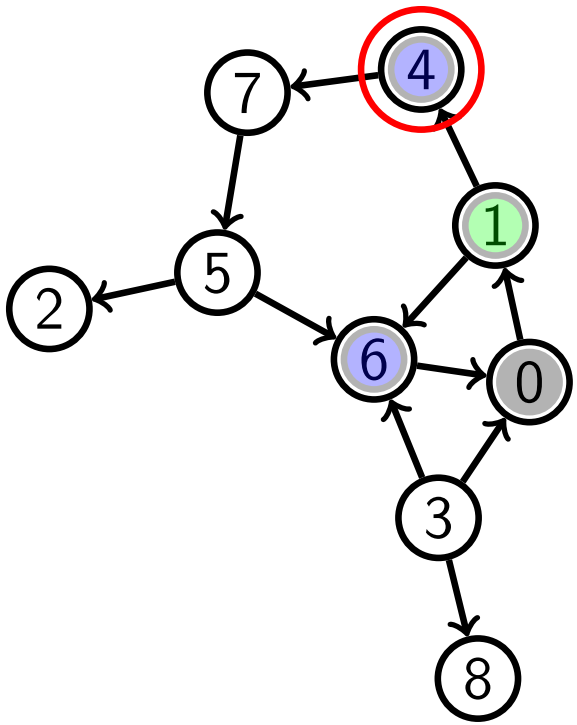


Ordre de l'exploration

0 1

Parcours en profondeur : exemple

Exemple 2

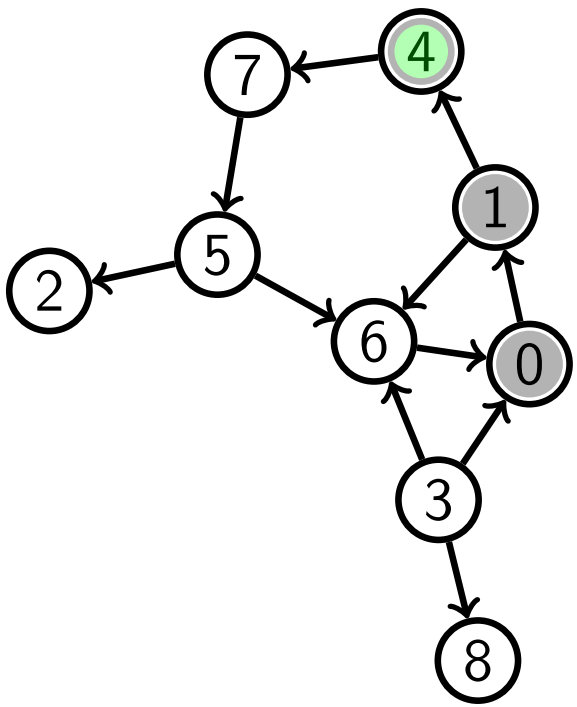


Ordre de l'exploration

0 1

Parcours en profondeur : exemple

Exemple 2

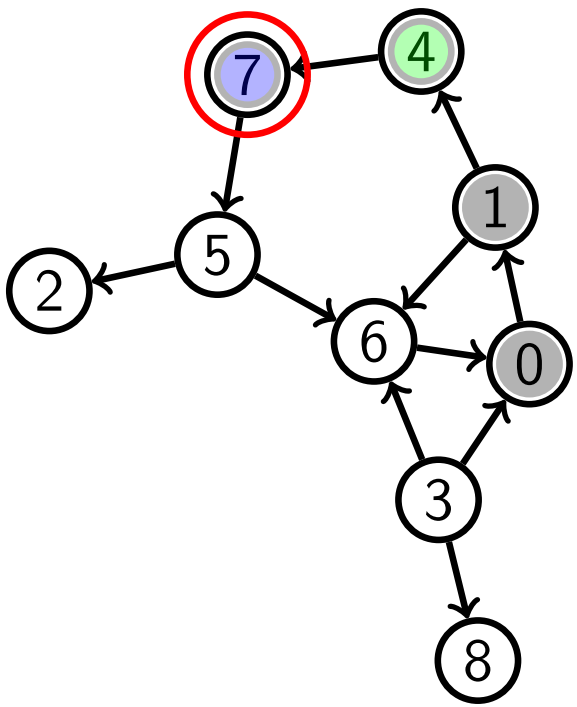


Ordre de l'exploration

0 1 4

Parcours en profondeur : exemple

Exemple 2

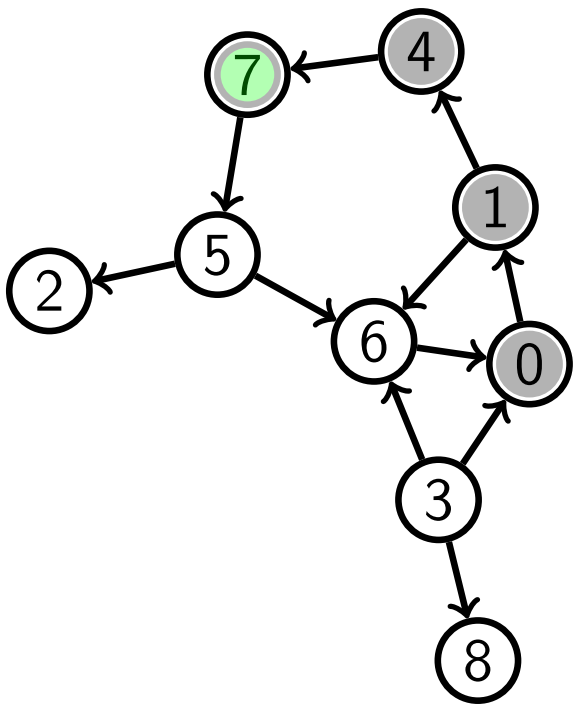


Ordre de l'exploration

0 1 4

Parcours en profondeur : exemple

Exemple 2

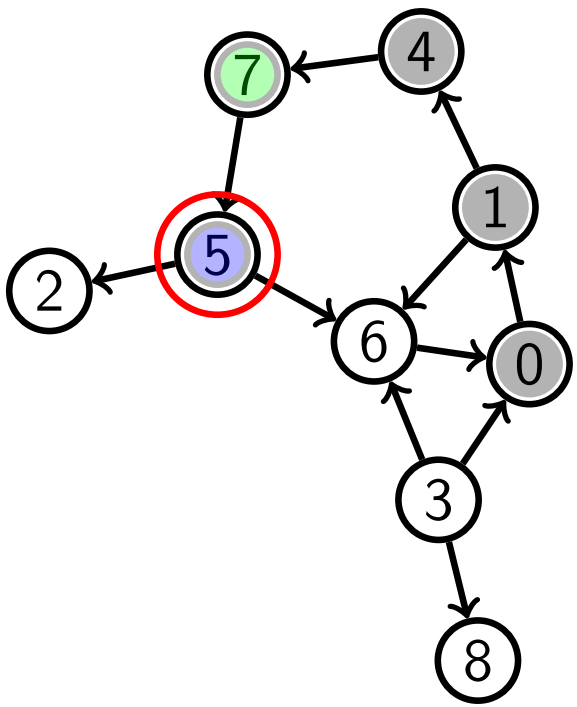


Ordre de l'exploration

0 1 4 7

Parcours en profondeur : exemple

Exemple 2

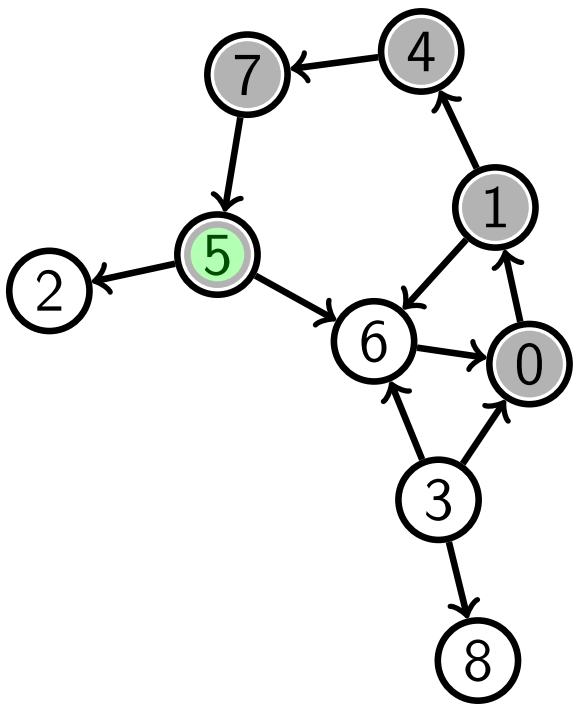


Ordre de l'exploration

0 1 4 7

Parcours en profondeur : exemple

Exemple 2

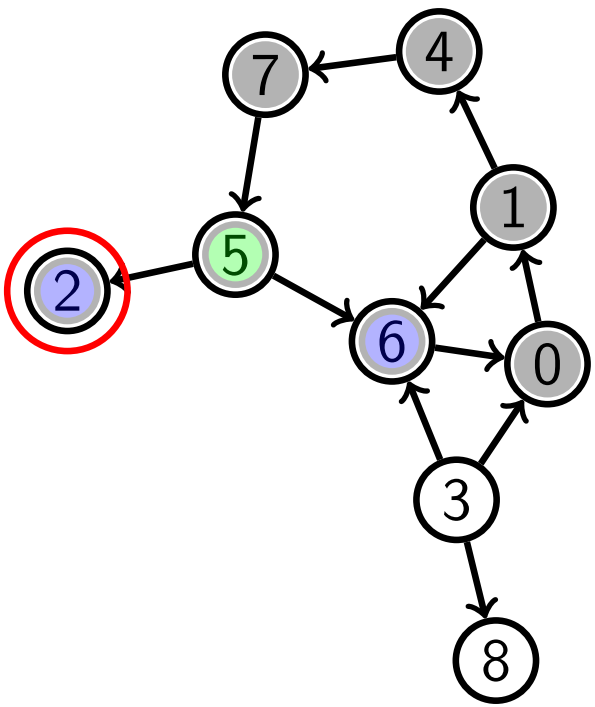


Ordre de l'exploration

0 1 4 7 5

Parcours en profondeur : exemple

Exemple 2

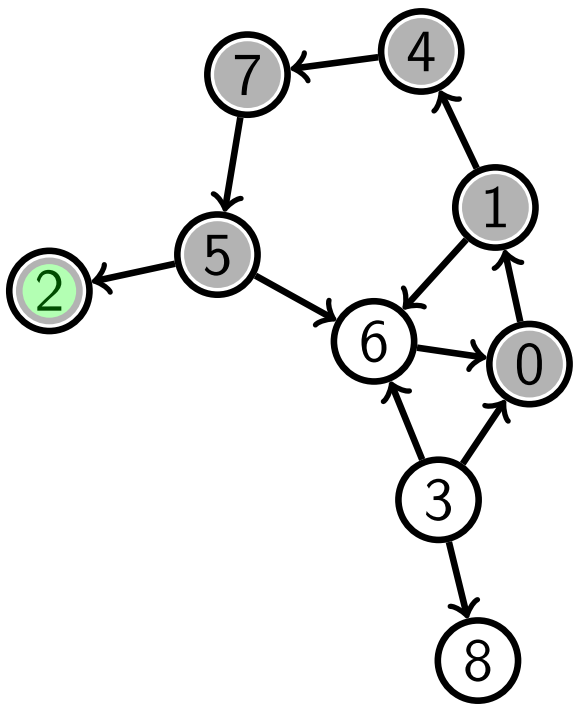


Ordre de l'exploration

0 1 4 7 5

Parcours en profondeur : exemple

Exemple 2

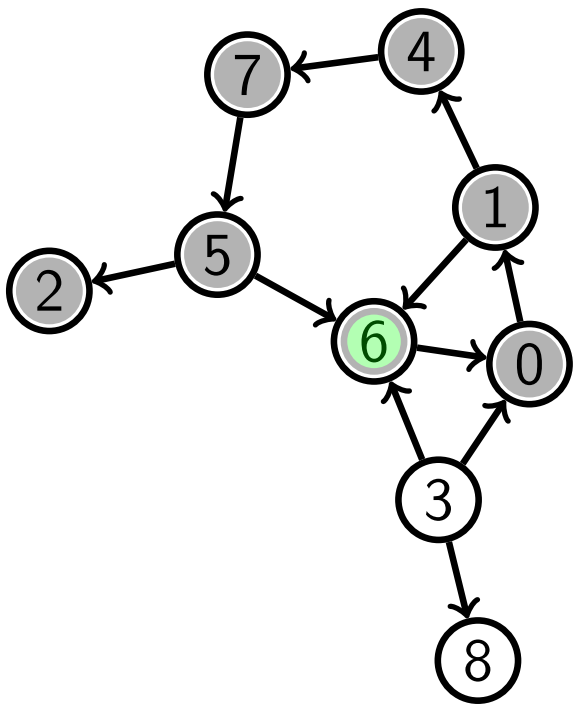


Ordre de l'exploration

0 1 4 7 5 2

Parcours en profondeur : exemple

Exemple 2

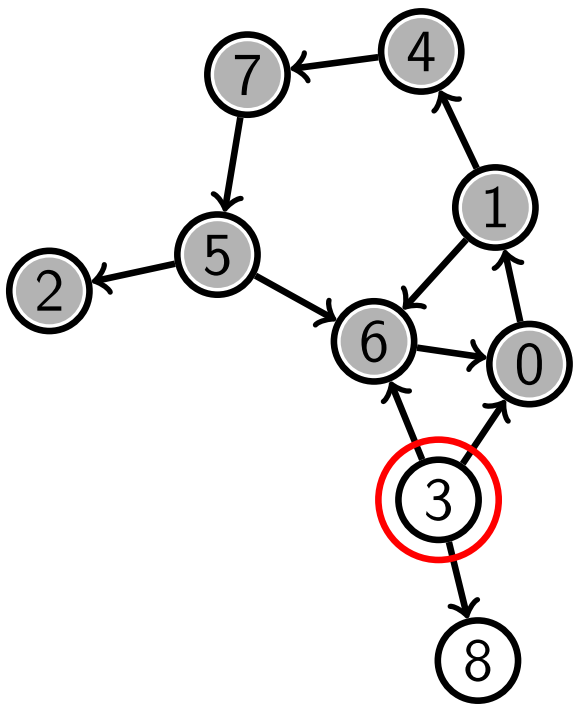


Ordre de l'exploration

0 1 4 7 5 2 6

Parcours en profondeur : exemple

Exemple 2

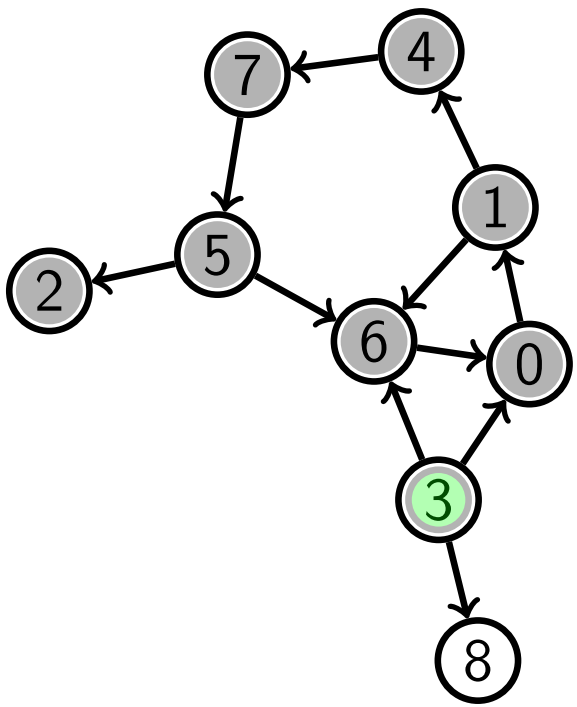


Ordre de l'exploration

0 1 4 7 5 2 6

Parcours en profondeur : exemple

Exemple 2

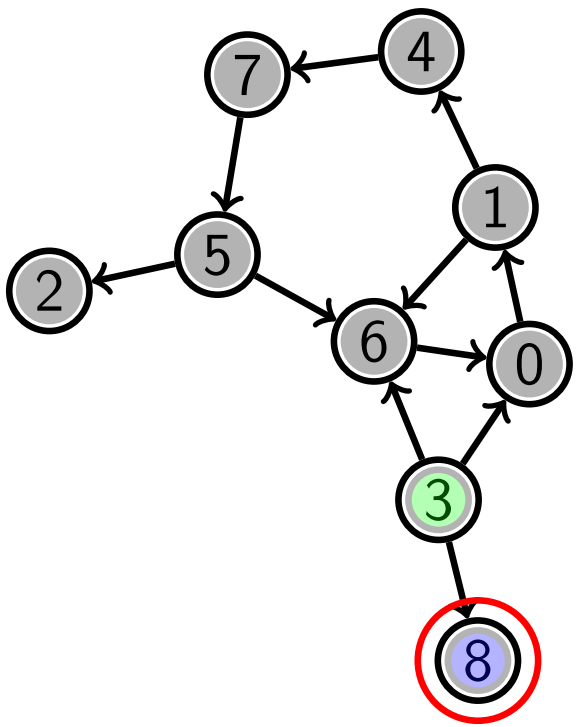


Ordre de l'exploration

0 1 4 7 5 2 6 3

Parcours en profondeur : exemple

Exemple 2

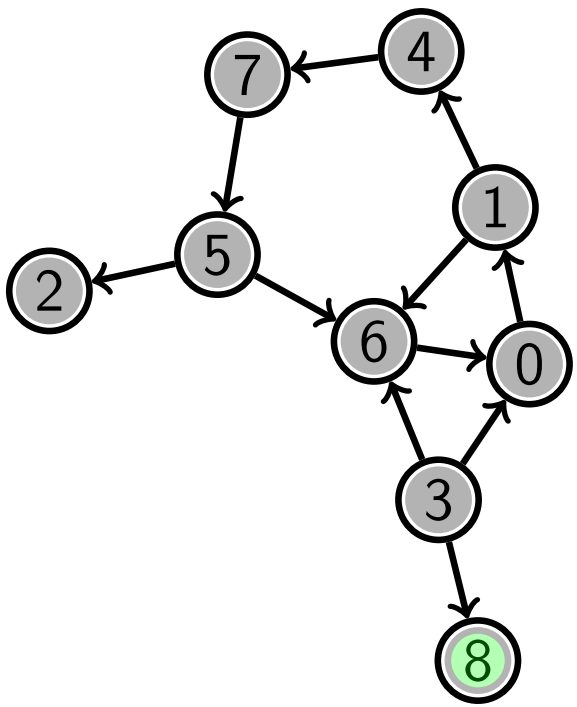


Ordre de l'exploration

0 1 4 7 5 2 6 3

Parcours en profondeur : exemple

Exemple 2

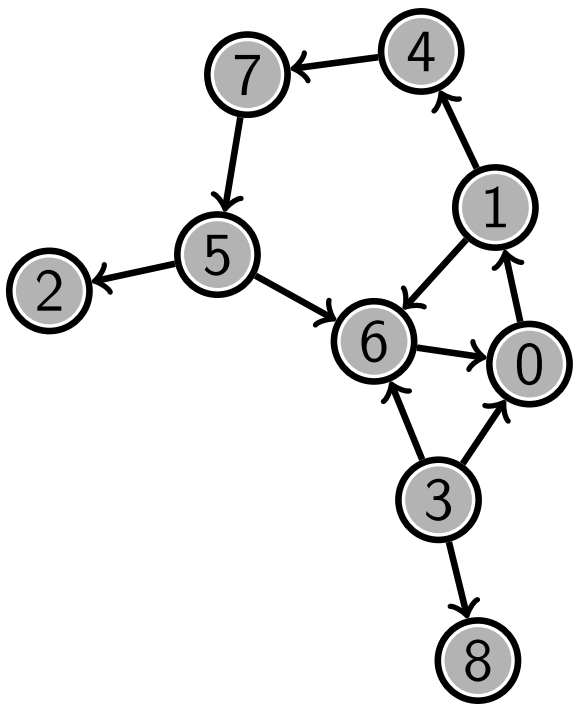


Ordre de l'exploration

0 1 4 7 5 2 6 3 8

Parcours en profondeur : exemple

Exemple 2

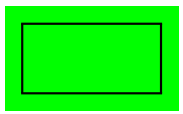


Ordre de l'exploration

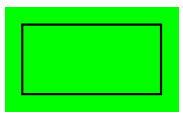
0 1 4 7 5 2 6 3 8

Exploration en profondeur récursive des graphes orientés

Pour certaines opérations on a aura besoin de trois couleurs :
Exploration en trois couleurs :

1.  : l'exploration du sommet n'a pas commencé ;
2.  : l'exploration a commencé ;
3.  : l'exploration est terminée.

Pour certaines opérations on a aura besoin de seulement deux couleurs :

1.  : l'exploration du sommet n'a pas commencé ;
2.  . l'exploration a commencé ou est terminée.
   Correspond à rouge ou noir avec les trois couleurs.

Exploration en profondeur récursive des graphes orientés

Algorithme 3 : PROFONDEUR(G)

Entrées : un graphe orienté G
Sortie : la liste des sommets de G dans l'ordre où le parcours en profondeur les a découverts (ordre de début de l'exploration).

```
1 résultat ← liste();
2 visités ←tableau(G.nombre_sommets(), "green");
3 pour chaque sommet dans G.sommets() faire
4 |   si visités[sommet] == "green" alors
5 |   |   ProfondeurRec(G, sommet, visités, résultat)
6 renvoyer résultat;
```

Exploration en profondeur récursive des graphes orientés

Algorithme 4 : ProfondeurRec(G, sommet, visités, résultat)

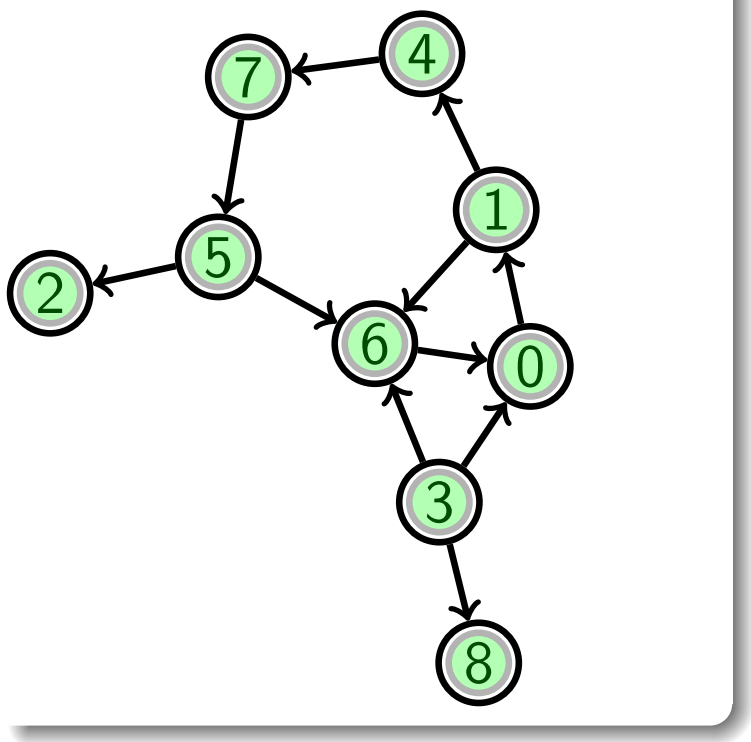
Entrées : un graphe orienté G, un sommet de G, un tableau visités, une liste résultat

Sortie : void

```
1 visités[sommet] ← "red";
2 résultat.add(sommet);
3 pour chaque successeur dans G.successeurs(sommet) faire
4 |   si visités[successeur] == "green" alors
5 |   |   ProfondeurRec(G, successeur, visités, résultat)
6 visités[sommet] ← "black";
```

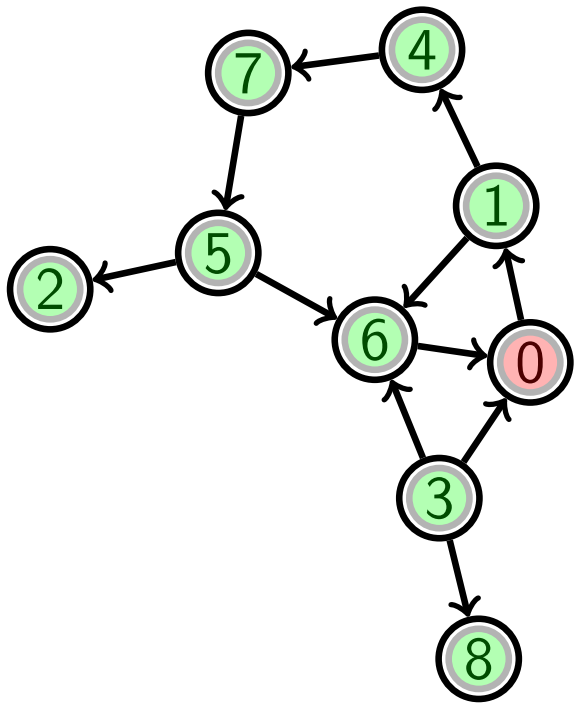
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



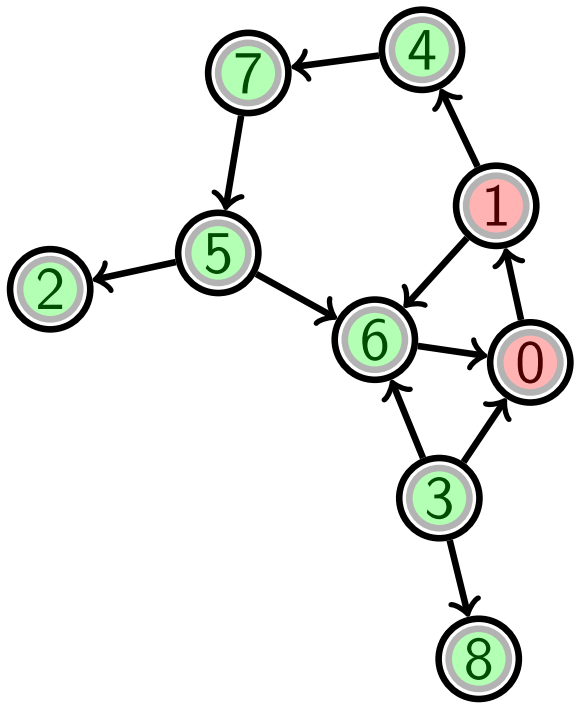
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



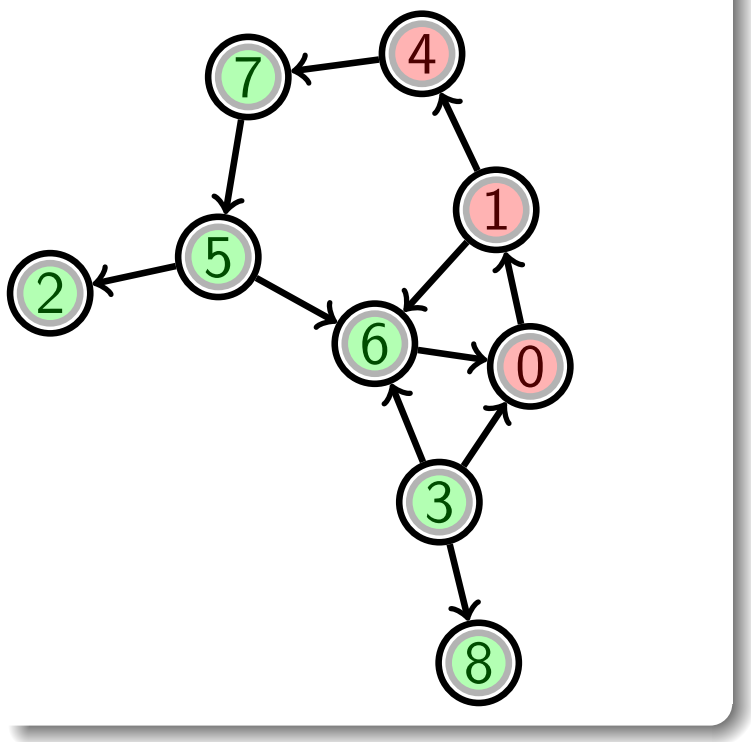
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



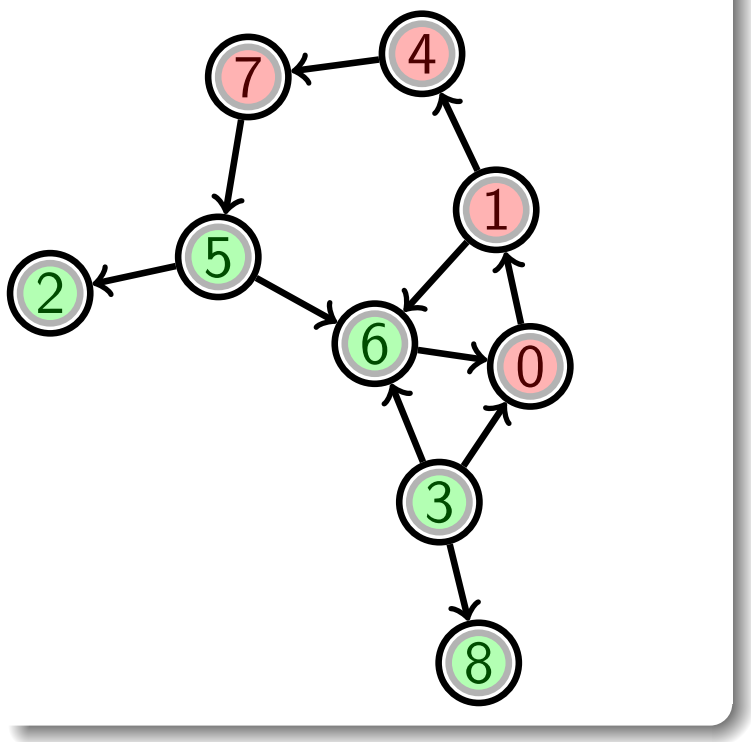
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



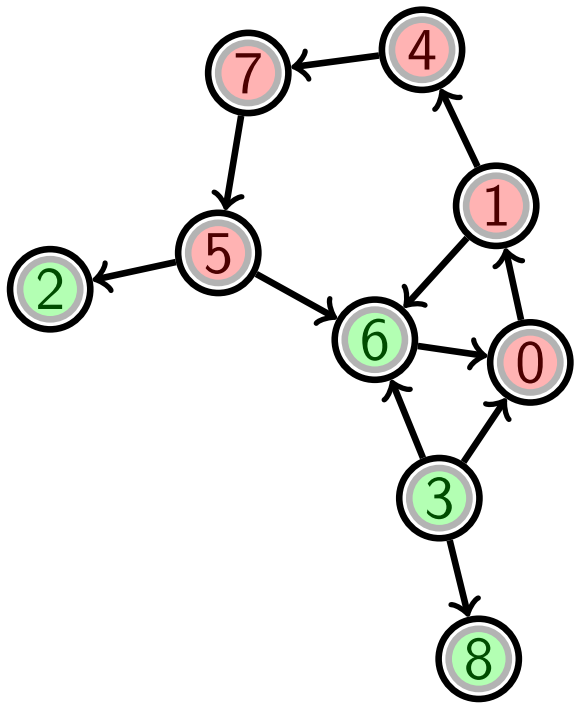
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



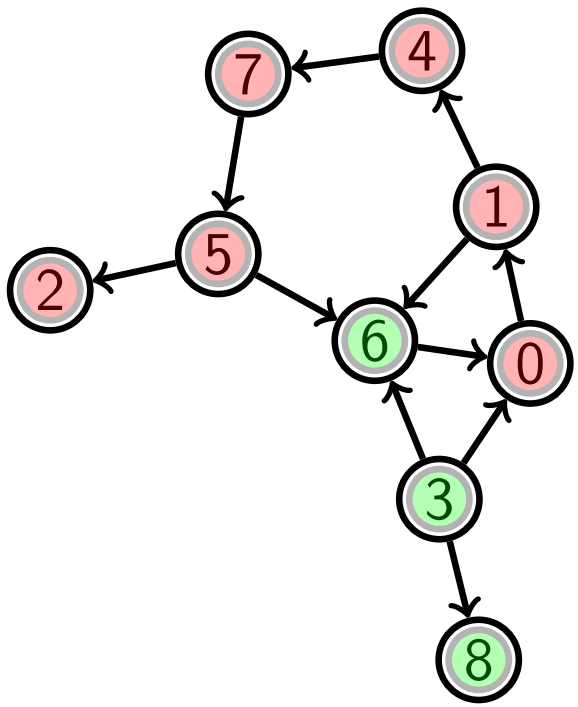
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



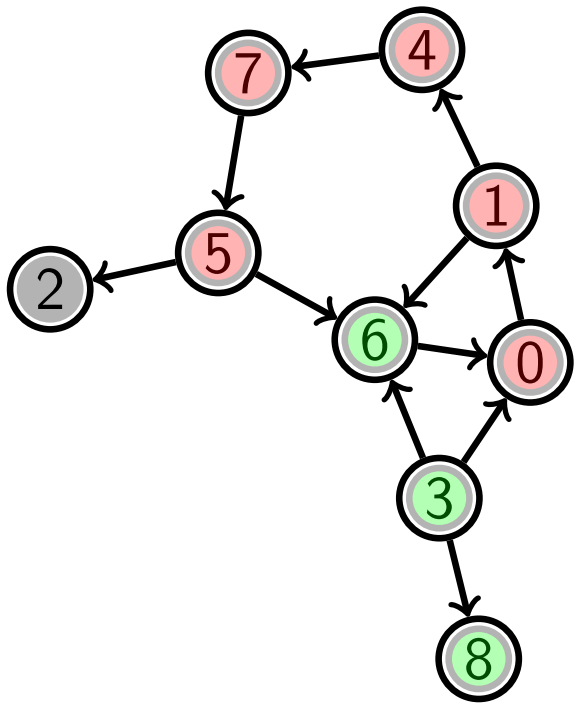
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



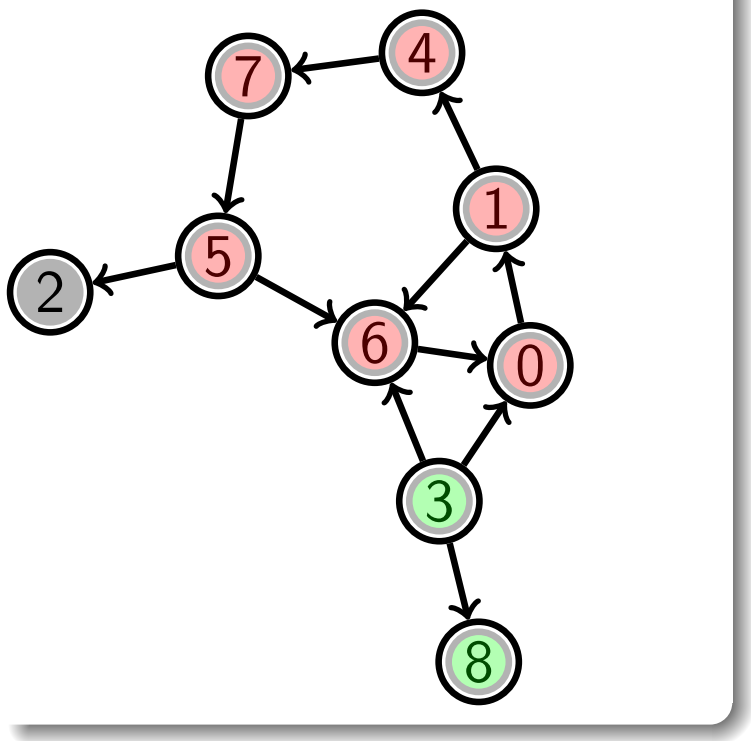
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



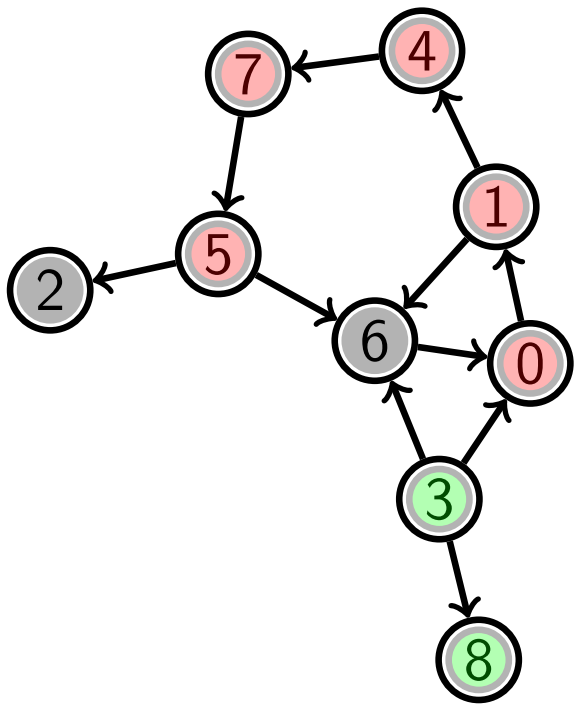
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



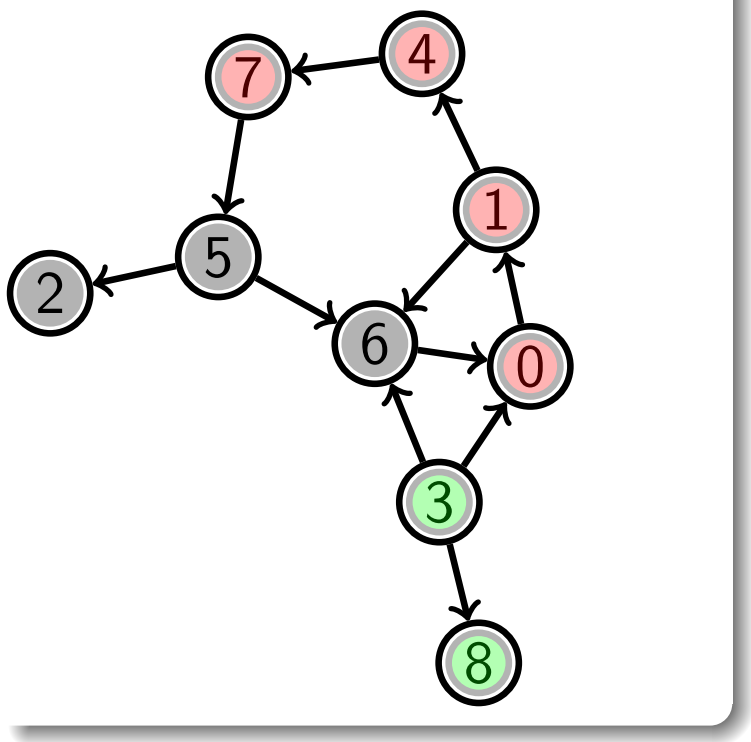
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



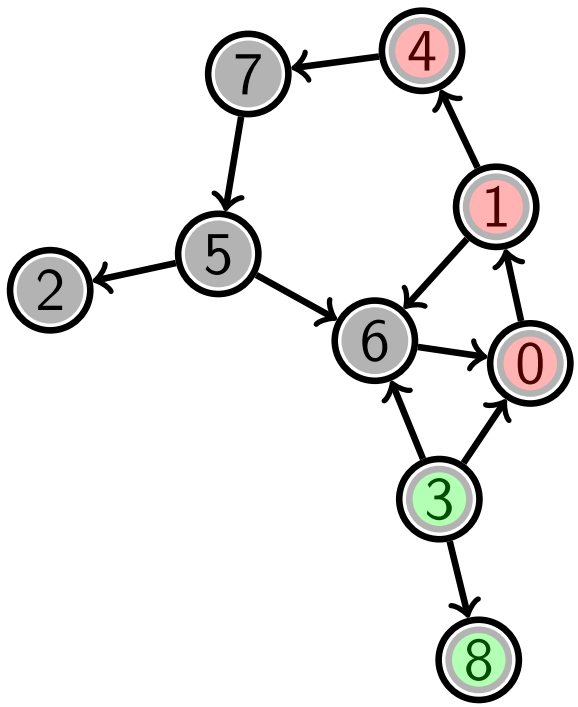
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



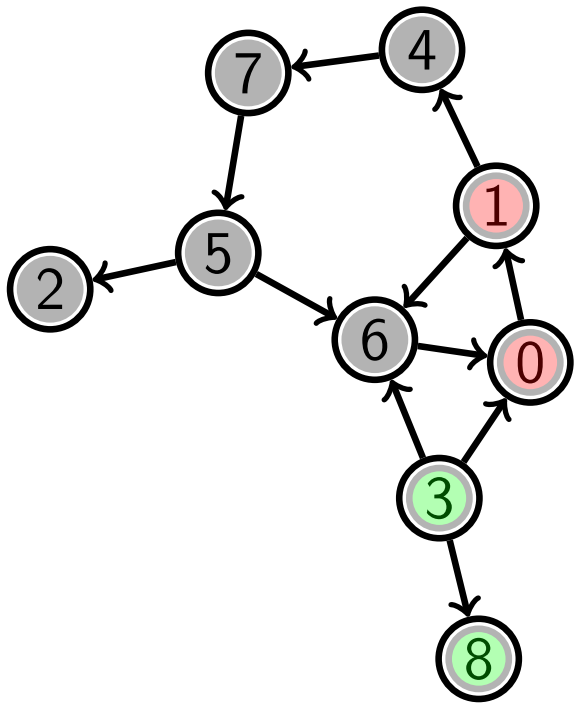
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



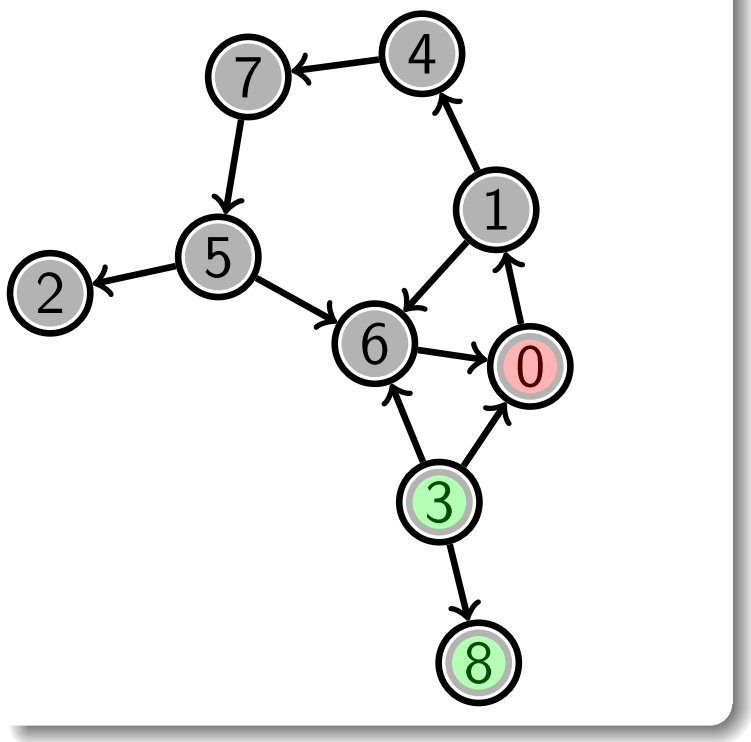
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



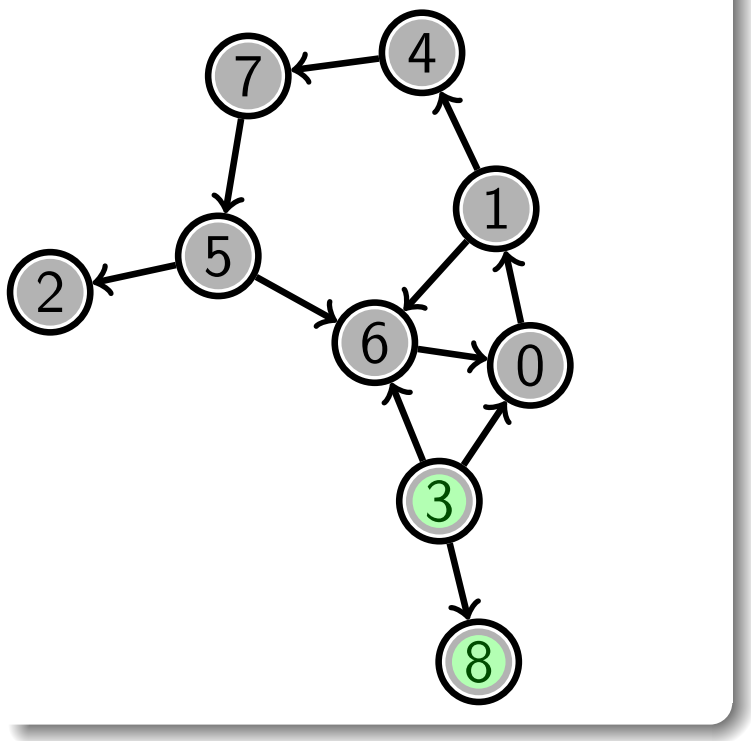
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



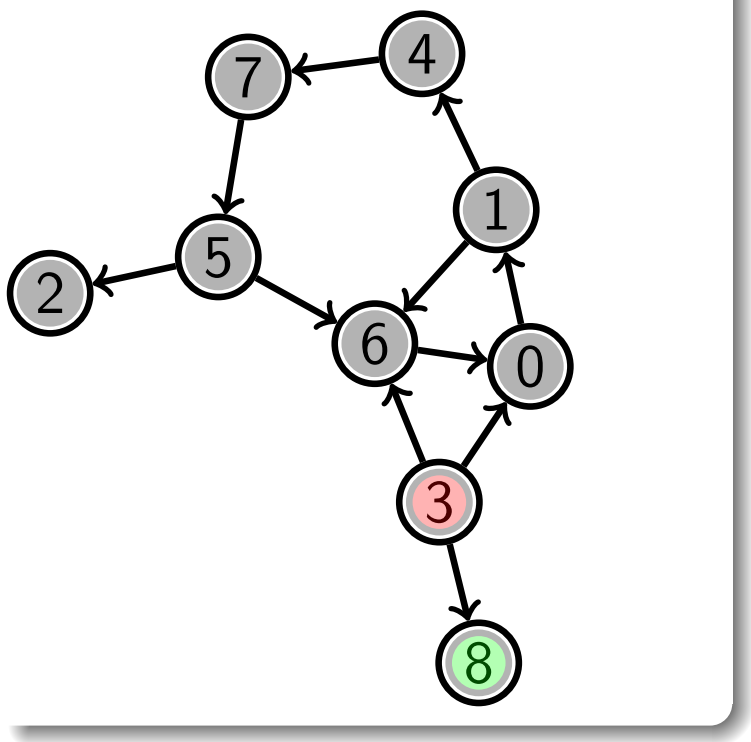
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



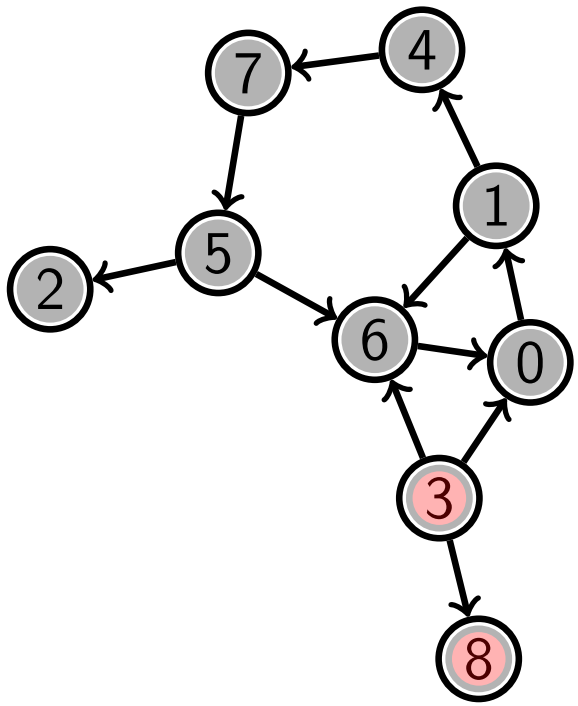
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



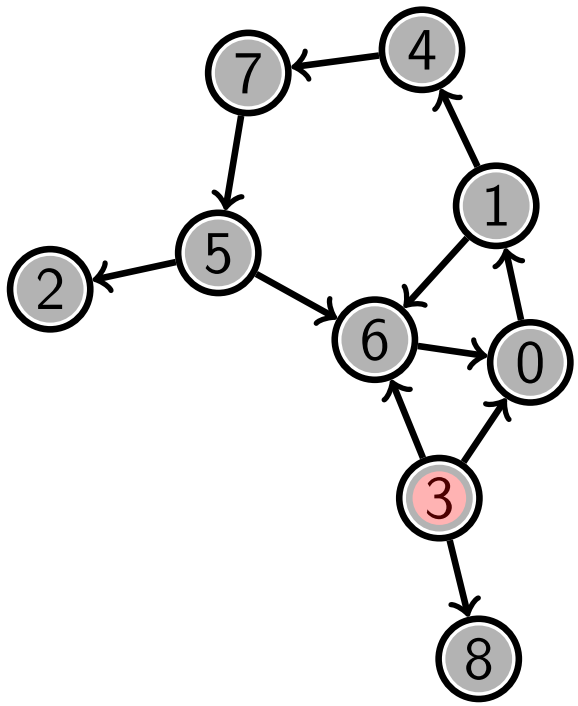
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



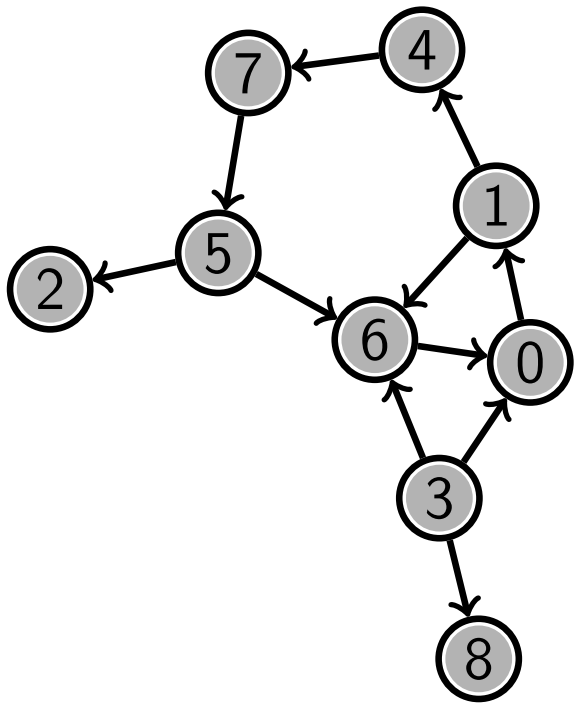
Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



Parcours en profondeur récursif avec trois couleurs : exemple

Exemple 3



Complexité de l'exploration récursive

- Toutes les méthodes des structures de données auxiliaires sont en $O(1)$;

Complexité de l'exploration récursive

- Toutes les méthodes des structures de données auxiliaires sont en $O(1)$;
- La complexité des algorithmes de parcours dépend donc directement de l'implémentation du graphe ;

Complexité de l'exploration récursive

- Toutes les méthodes des structures de données auxiliaires sont en $O(1)$;
- La complexité des algorithmes de parcours dépend donc directement de l'implémentation du graphe ;
- Pour chaque sommet v, on doit accéder à tous ses successeurs ;

Complexité de l'exploration récursive

- Toutes les méthodes des structures de données auxiliaires sont en $O(1)$;
- La complexité des algorithmes de parcours dépend donc directement de l'implémentation du graphe ;
- Pour chaque sommet v, on doit accéder à tous ses successeurs ;
  - matrice d'adjacence : G.successeurs(v) est en $O(|V|)$;

Complexité de l'exploration récursive

- Toutes les méthodes des structures de données auxiliaires sont en $O(1)$;
- La complexité des algorithmes de parcours dépend donc directement de l'implémentation du graphe ;
- Pour chaque sommet v, on doit accéder à tous ses successeurs ;
  - matrice d'adjacence : G.successeurs(v) est en $O(|V|)$;
  - listes d'adjacence : G.successeurs(v) est en $O(\deg^+(v))$.

Complexité de l'exploration récursive

- On examine chaque sommet une fois et une seule. Pour chaque sommet on accède une fois et une seule à chacun de ses successeurs ;

Complexité de l'exploration récursive

- On examine chaque sommet une fois et une seule. Pour chaque sommet on accède une fois et une seule à chacun de ses successeurs ;
- Le parcours en profondeur récursif a donc une complexité de :

Complexité de l'exploration récursive

- On examine chaque sommet une fois et une seule. Pour chaque sommet on accède une fois et une seule à chacun de ses successeurs ;
- Le parcours en profondeur récursif a donc une complexité de :
  - $O(|V|^2)$ pour une matrice d'adjacence ;

Complexité de l'exploration récursive

- On examine chaque sommet une fois et une seule. Pour chaque sommet on accède une fois et une seule à chacun de ses successeurs ;
- Le parcours en profondeur récursif a donc une complexité de :
  - $O(|V|^2)$ pour une matrice d'adjacence ;
  - $O(|V| + |A|)$ pour des listes d'adjacence ($\sum_{v \in V} \deg^+(v) = |A|$).

Exploration en profondeur récursive des graphes orientés

Pour une exploration en profondeur donnée, on définit :

- Les arcs directs : arcs (u, v) de G tels que lors de l'exploration de u, v est vert et on lance l'exploration de v à partir de u.
- Les arcs retours : arcs (u, v) de G tels que lors de l'exploration de u on examine le successeur v qui est rouge.

Exploration en profondeur récursive des graphes orientés

Algorithme 5 : ProfondeurRec(G, sommet, visités, résultat)

Entrées : un graphe orienté G, un sommet de G, un tableau visités, une liste résultat

Sortie : void

```
1 visités[sommet] ← "red";
2 pour chaque successeur dans G.successeurs(sommet) faire
3 |   si visités[successeur] == "green" alors
4 |   |   (sommet, successsseur) est un arc direct ;
  |   |    ProfondeurRec(G, successeur, visités, résultat)
5 |   si visités[successeur] == "red" alors
6 |   |   (sommet, successsseur) est un arc retour ;
7 visités[sommet] ← "black";
```

Forêt recouvrante pour le parcours en profondeur

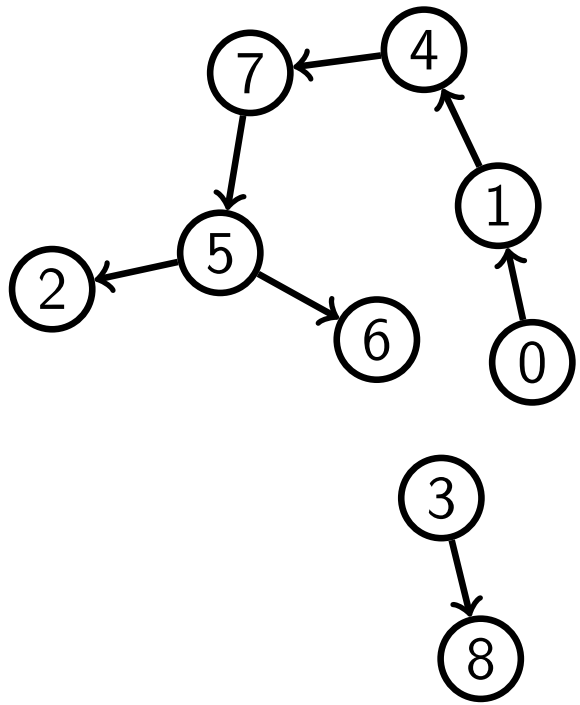
Étant donnée une exploration en profondeur, le graphe obtenu à partir de G en gardant uniquement les arcs directs s'appelle la forêt recouvrante.

- La forêt recouvrante comprend tous les sommets du graphe G.
- C'est une union fini d'arbres.

Forêt recouvrante sur l'exploration précédente

Ici la forêt est constitué de deux arbres.

Exemple 4



Détection de cycles orientés

Proposition 1

Un graphe orienté admet un cycle orienté si et seulement il admet au moins un arc retour lors d'une exploration en profondeur.

Preuve :

- Supposons qu'il existe un arc retour (u, v). Lors de l'exploration de u on trouve un successeur v rouge. Comme v est rouge, il existe un chemin de v à u dans un arbre de la forêt recouvrante. D'autre part il existe un chemin de u à v dans G. Donc u et v sont sur un cycle.

Détection de cycles orientés

Proposition 1

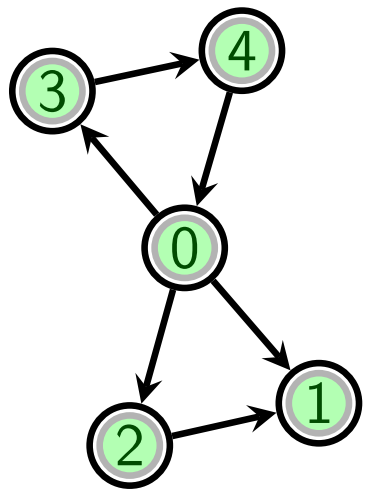
Un graphe orienté admet un cycle orienté si et seulement il admet au moins un arc retour lors d'une exploration en profondeur.

Preuve :

- Supposons qu'il existe un arc retour (u, v). Lors de l'exploration de u on trouve un successeur v rouge. Comme v est rouge, il existe un chemin de v à u dans un arbre de la forêt recouvrante. D'autre part il existe un chemin de u à v dans G. Donc u et v sont sur un cycle.
- S'il existe un cycle $C = (u_0, u_1, \ldots, u_{p-1} = u_0)$. Soit i tel que u_i est le premier exploré dans C. Au cours de l'exploration de u_i on atteint tous les sommets de C. Donc on atteint $u_{i-1 \mod p}$. Lors de l'exploration de $u_{i-1 \mod p}$ l'arc $(u_{i-1 \mod p}, u_i)$ est alors un arc retour.

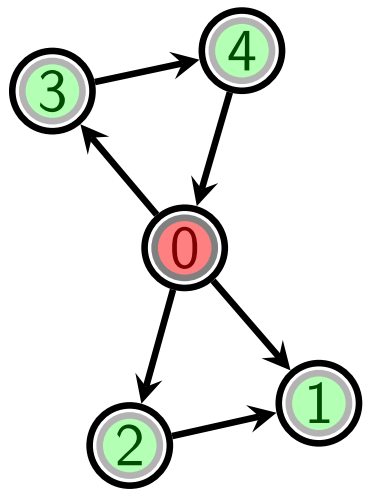
Détection de cycles orientés

Exemple 5 (parcours tricolore)



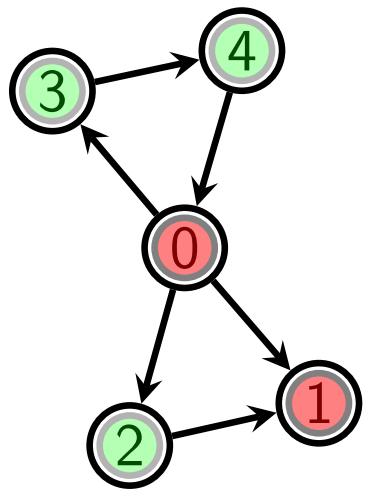
Détection de cycles orientés

Exemple 5 (parcours tricolore)



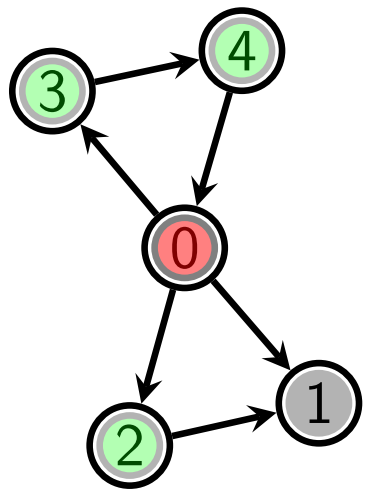
Détection de cycles orientés

Exemple 5 (parcours tricolore)



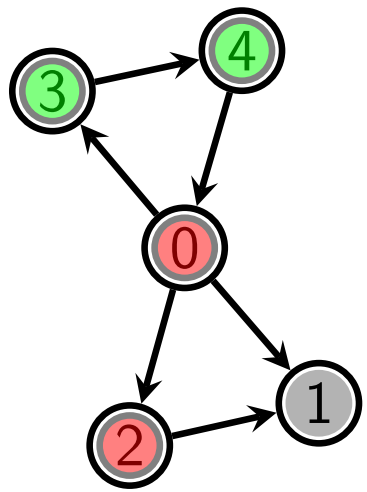
Détection de cycles orientés

Exemple 5 (parcours tricolore)



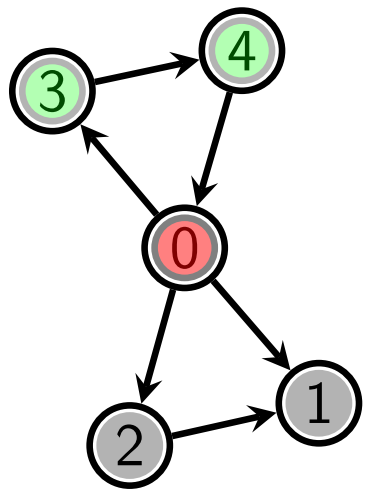
Détection de cycles orientés

Exemple 5 (parcours tricolore)



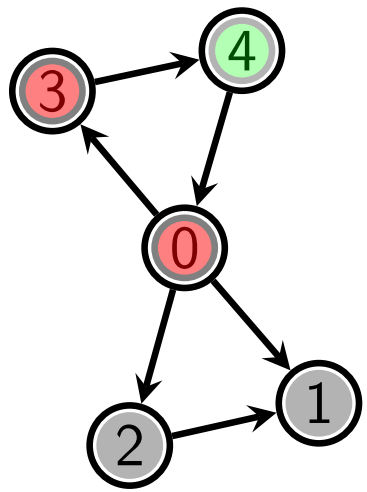
Détection de cycles orientés

Exemple 5 (parcours tricolore)



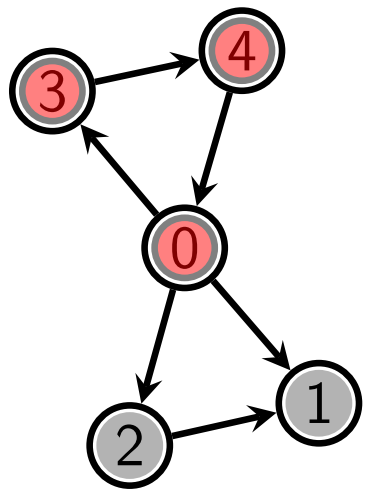
Détection de cycles orientés

Exemple 5 (parcours tricolore)



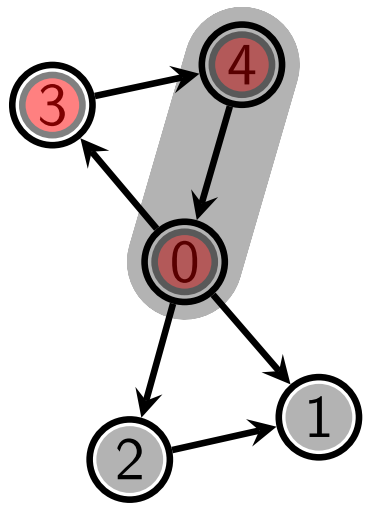
Détection de cycles orientés

Exemple 5 (parcours tricolore)



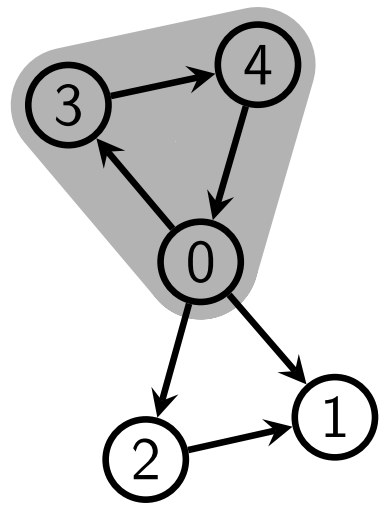
Détection de cycles orientés

Exemple 5 (parcours tricolore)



Détection de cycles orientés

Exemple 5 (parcours tricolore)



L'algorithme de détection d'un cycle orienté

Une légère adaptation du parcours en profondeur récursif nous permet d'arriver à nos fins :

Algorithme 6 : ContientCycleOrienté(G)

Entrées : un graphe orienté G
Sortie : VRAI si G contient un cycle orienté FAUX sinon.

```
1 visités ←tableau(G.nombre_sommets(), "green");
2 pour chaque sommet dans G.sommets() faire
3 |   si visités[sommet] == "green" alors
4 |   |   si ContientCycleOrientéRec(G, sommet, visités) alors
5 |   |   |   renvoyer VRAI;
6 renvoyer FAUX;
```

L'algorithme de détection d'un cycle orienté

Algorithme 7 : ContientCycleOrientéRec(G, sommet, visités)

Entrées : un graphe orienté G, un sommet de G, un tableau visités, une liste résultat

Sortie : VRAI si G contient un cycle orienté accessible à partir de `sommet` FAUX sinon.

```
1 visités[sommet] ← "red";
2 pour chaque successeur dans G.successeurs(sommet) faire
3     si visités[successeur] == "green" alors
4         si ContientCycleOrientéRec(G, successeur, visités) alors
5             renvoyer VRAI;
6     si visités[successeur] == "red" alors
7         renvoyer VRAI;
8 visités[sommet] ← "black";
9 renvoyer FAUX;
```

Parcours en profondeur itératif

Algorithme 8 : PROFONDEUR(G, départ, visités=NIL)

Entrées : un graphe orienté G, un sommet de départ, un tableau (facultatif) visités de taille $|V|$.

Sortie : la liste des sommets de G accessibles depuis le départ dans l'ordre où le parcours en profondeur les a découverts.

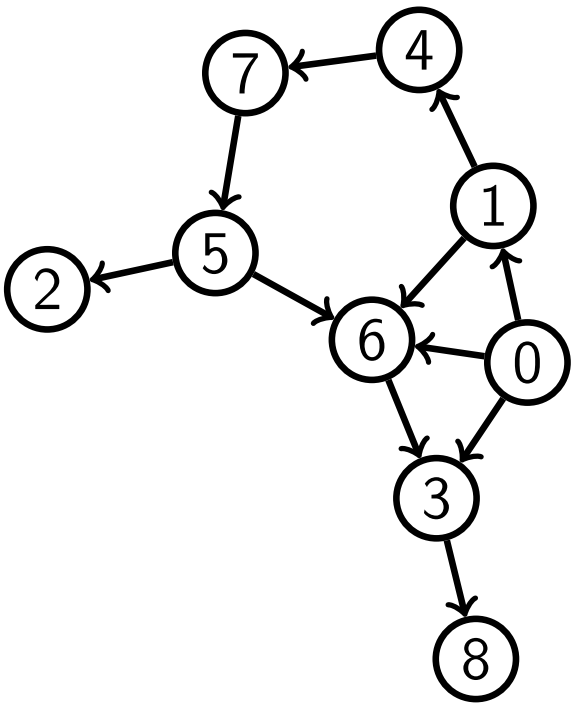
```
1  résultat ← liste();
2  si visités = NIL alors visités ← tableau(G.nombre_sommets(), FAUX);
3  a_traiter ← pile();
4  a_traiter.empiler(départ);
5  tant que a_traiter.pas_vide() faire
6  |   sommet ← a_traiter.dépiler();
7  |   si ¬ visités[sommet] alors
8  |   |   résultat.ajouter_en_fin(sommet);
9  |   |   visités[sommet] ← VRAI;
10 |   |   pour chaque successeur dans
   |   |     renverser(G.successeurs(sommet)) faire
11 |   |   |   si ¬ visités[successeur] alors a_traiter.empiler(successeur) ;
12 renvoyer résultat;
```

Parcours en profondeur itératif

Note : il peut y avoir plusieurs occurrences d'un même sommet dans la pile. Ne pas l'interdire !

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

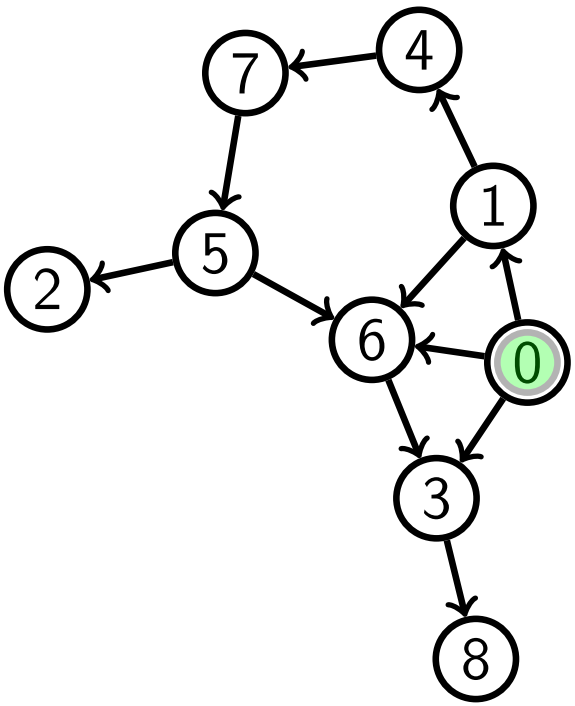
visités : 0 1 2 3 4 5 6 7 8

à traiter (pile)

résultat :

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

0 1 2 3 4 5 6 7 8

visités : | | | | | | | | | |

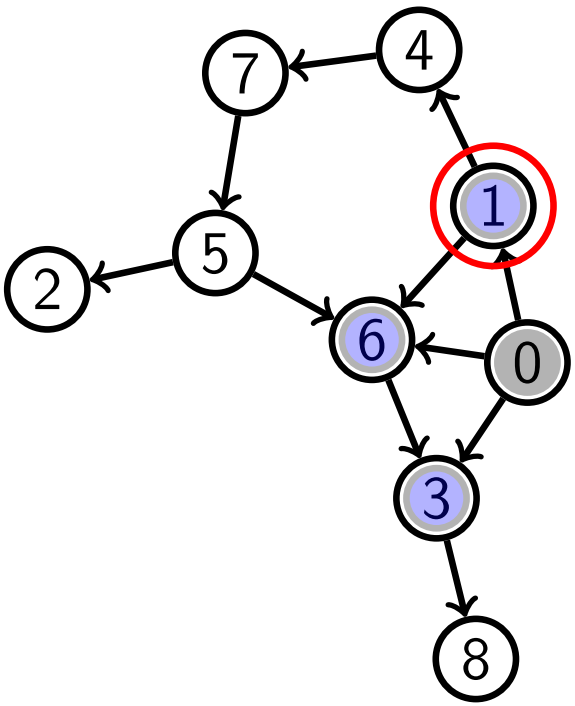
0

à traiter (pile)

résultat :

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓								

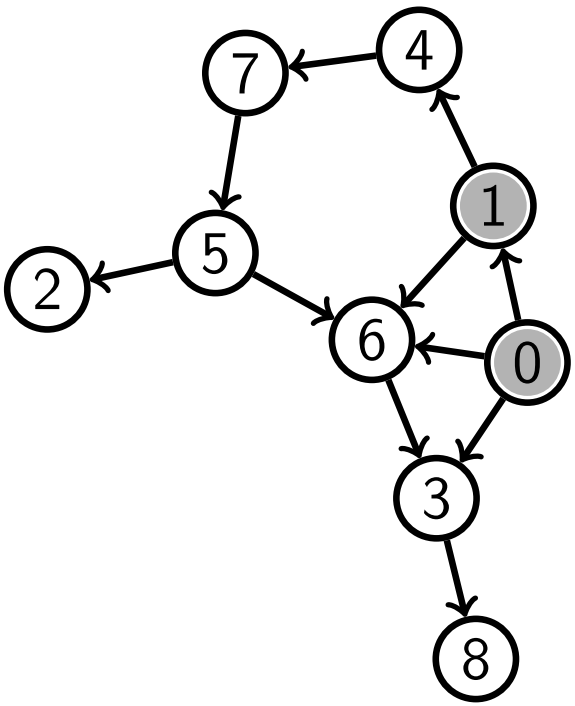
1
3
6

à traiter (pile)

résultat : 0

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓							

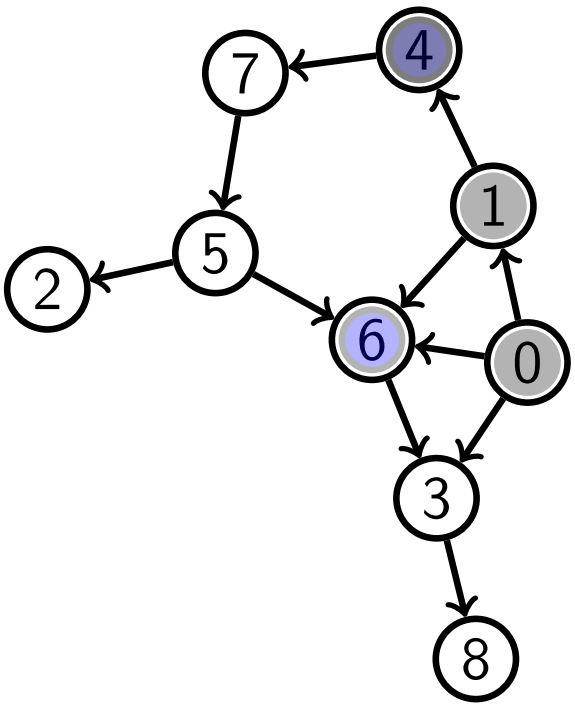
3
6

à traiter (pile)

résultat : 0 1

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓							

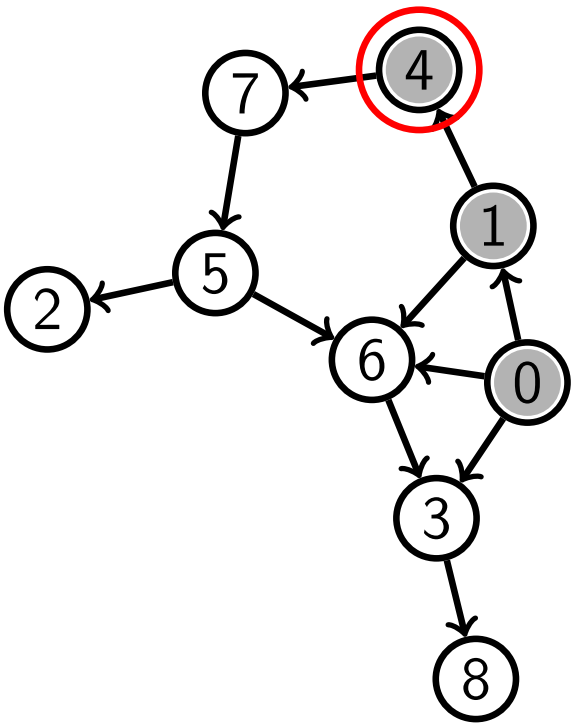
4
6
3
6

à traiter (pile)

résultat : 0 1

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓			✓				

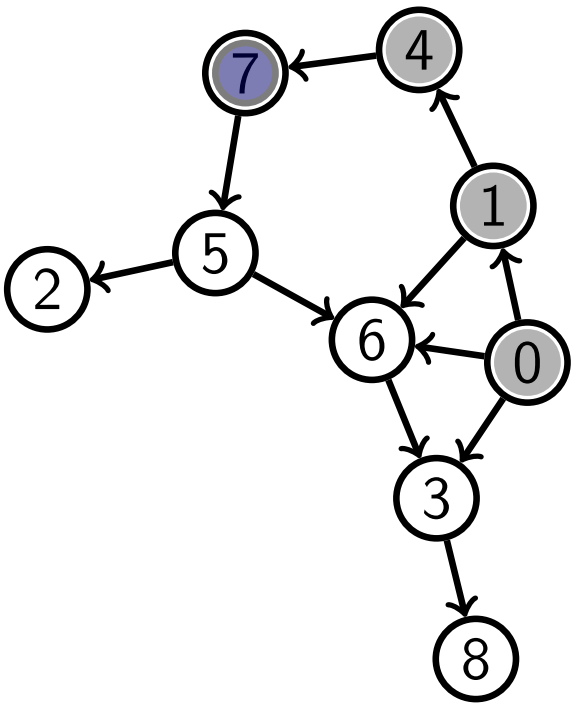
6
3
6

à traiter (pile)

résultat : 0 1 4

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓			✓				

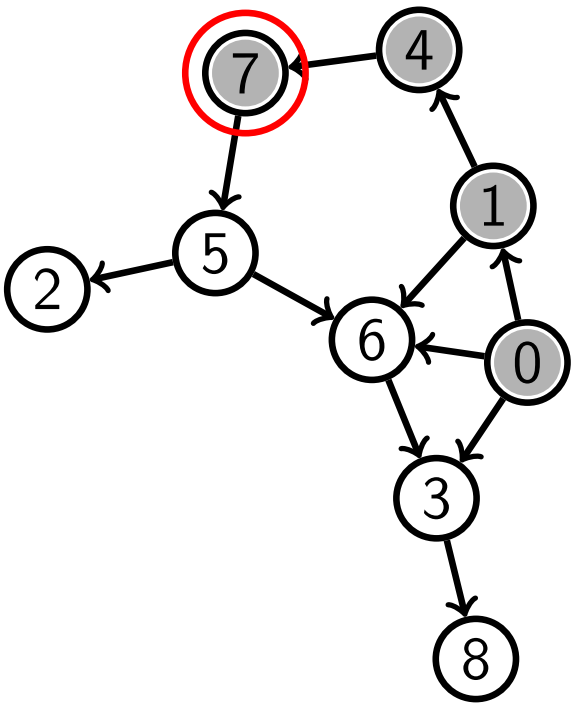
7
6
3
6

à traiter (pile)

résultat : 0 1 4

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓			✓			✓	

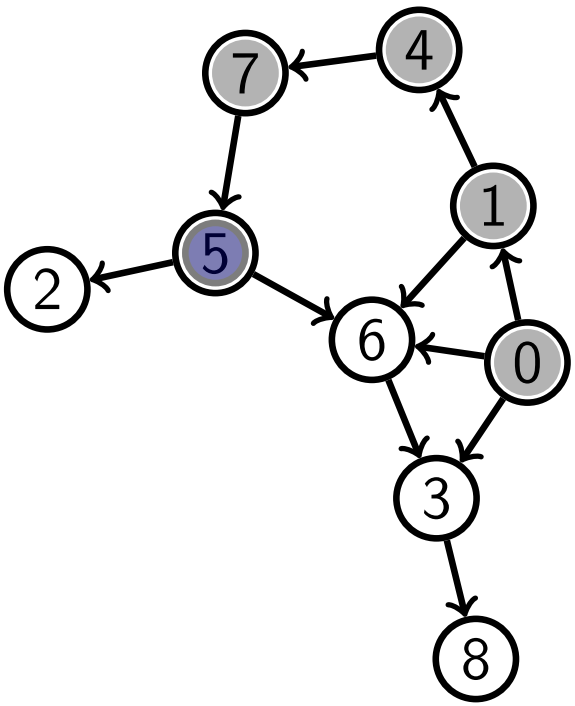
6
3
6

à traiter (pile)

résultat : 0 1 4 7

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓			✓			✓	

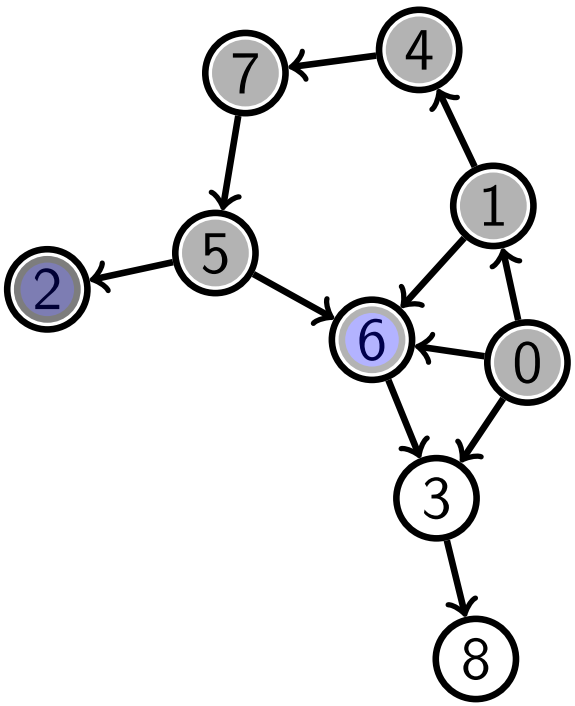
5
6
3
6

à traiter (pile)

résultat : 0 1 4 7

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

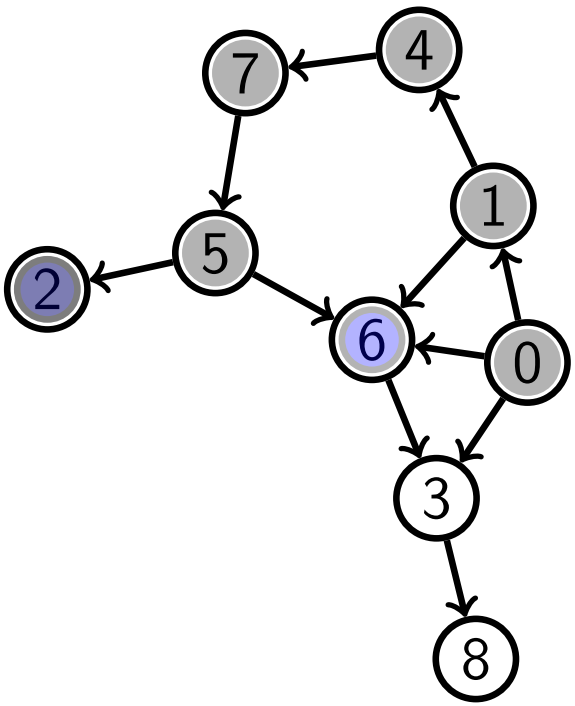
	0	1	2	3	4	5	6	7	8
visités :	✓	✓			✓	✓		✓	

à traiter (pile)
6
3
6

résultat : 0 1 4 7 5

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

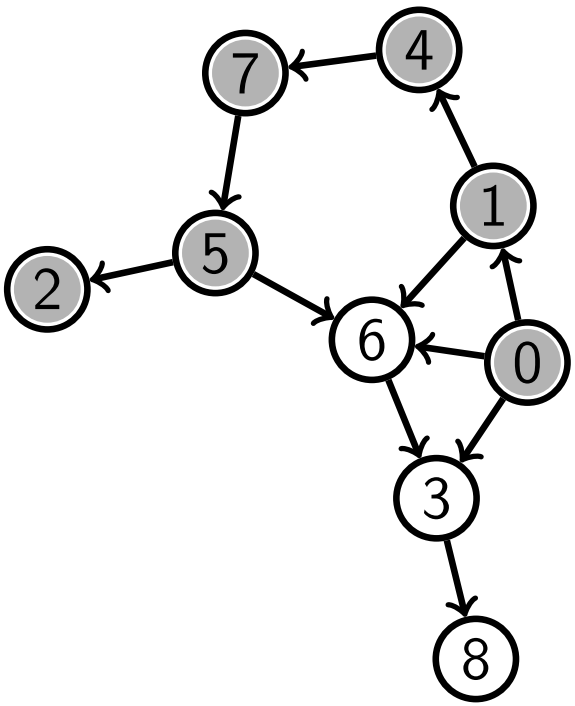
	0	1	2	3	4	5	6	7	8
visités :	✓	✓			✓	✓		✓	

à traiter (pile)
2
6
6
3
6

résultat : 0 1 4 7 5

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓	✓		✓	✓		✓	

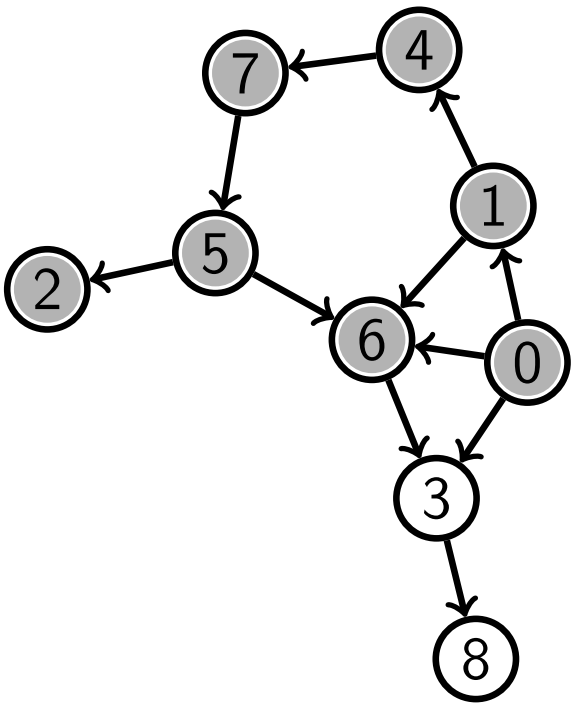
6
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓	✓		✓	✓	✓	✓	

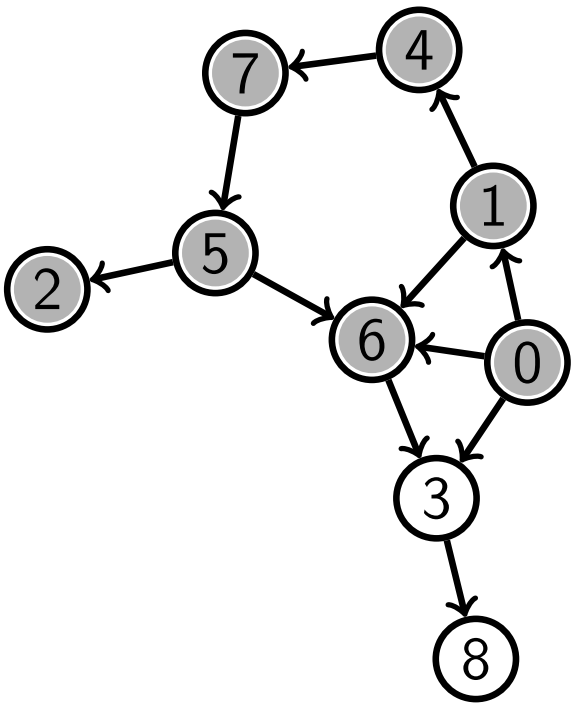
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓	✓		✓	✓	✓	✓	

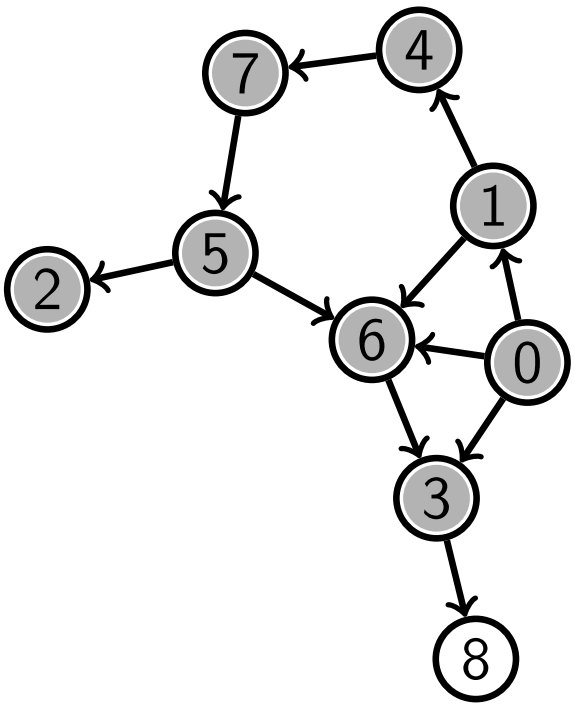
3
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓	✓	✓	✓	✓	✓	✓	

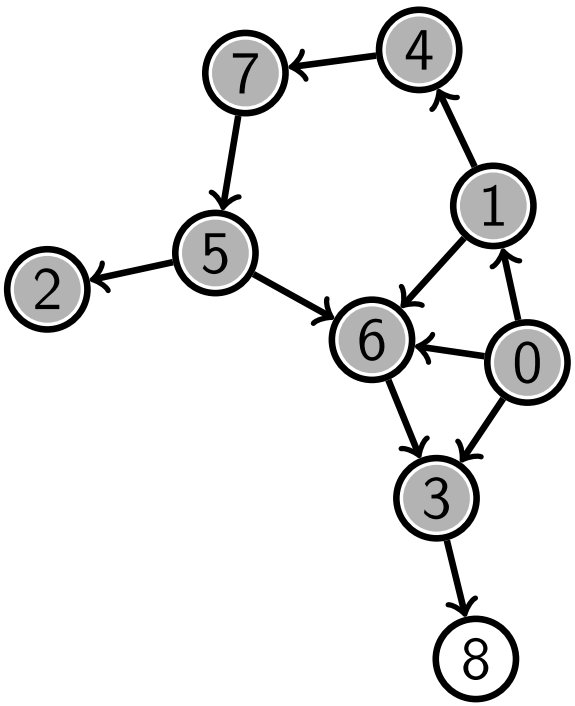
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓	✓	✓	✓	✓	✓	✓	

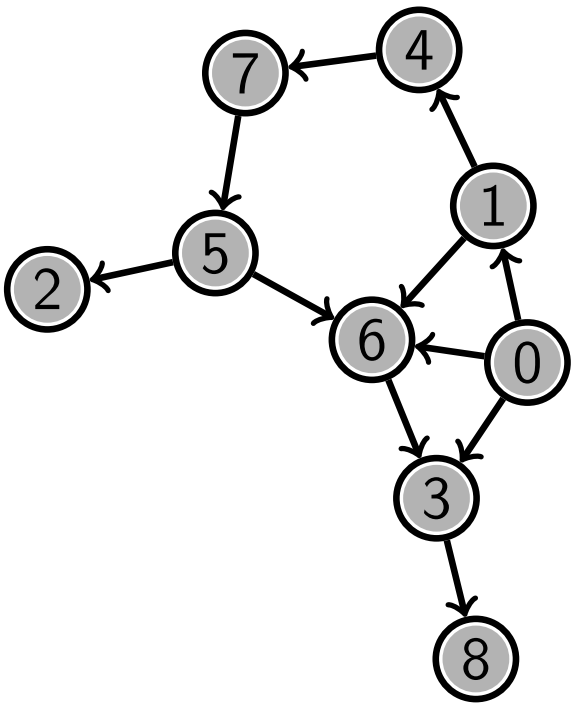
8
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓	✓	✓	✓	✓	✓	✓	✓

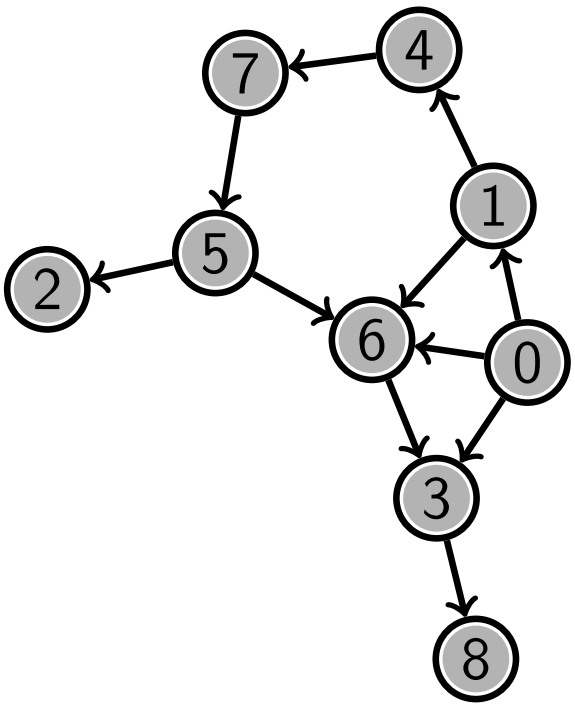
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3 8

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓	✓	✓	✓	✓	✓	✓	✓

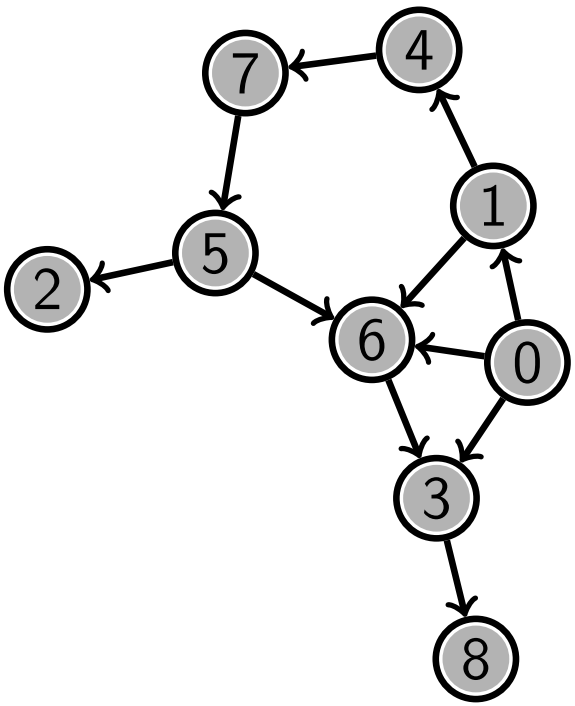
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3 8

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓	✓	✓	✓	✓	✓	✓	✓

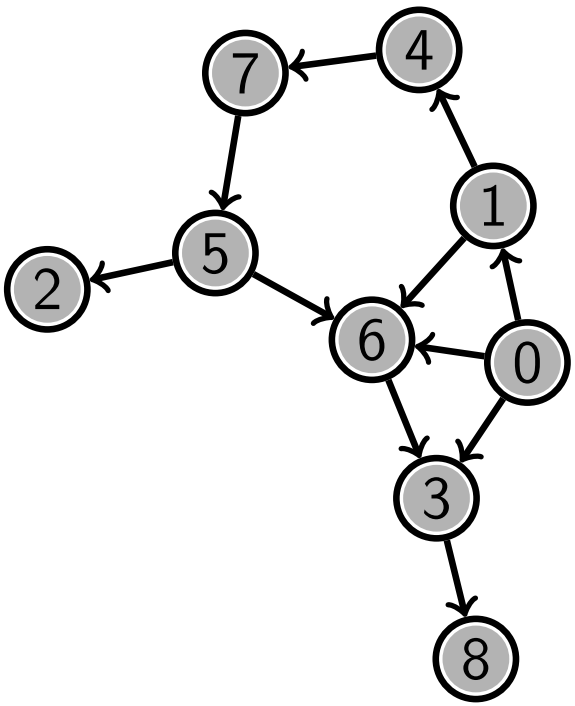
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3 8

Parcours en profondeur itératif : exemple

Exemple 6 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓	✓	✓	✓	✓	✓	✓	✓

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3 8

Parcours de graphe orienté en largeur

- si le sommet actuel u n'a pas encore été visité, l'afficher ;

Parcours de graphe orienté en largeur

- si le sommet actuel u n'a pas encore été visité, l'afficher ;
- pour chaque successeur v non encore visité de u : placer v dans la file d'attente ;

Parcours de graphe orienté en largeur

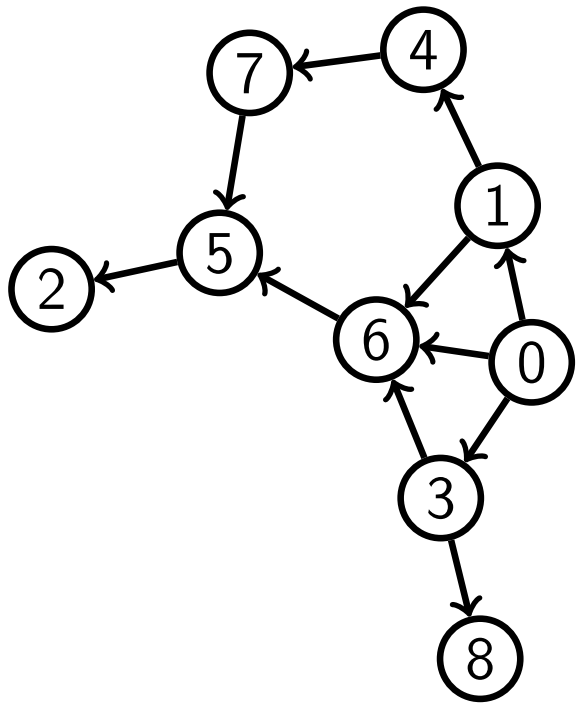
- si le sommet actuel u n'a pas encore été visité, l'afficher ;
- pour chaque successeur v non encore visité de u : placer v dans la file d'attente ;
- appliquer le même traitement aux éléments de la file jusqu'à ce qu'elle soit vide ;

Parcours de graphe orienté en largeur

- si le sommet actuel u n'a pas encore été visité, l'afficher ;
- pour chaque successeur v non encore visité de u : placer v dans la file d'attente ;
- appliquer le même traitement aux éléments de la file jusqu'à ce qu'elle soit vide ;
- Ce parcours partitionne les sommets de G en fonction de leur **distance** au sommet de départ.

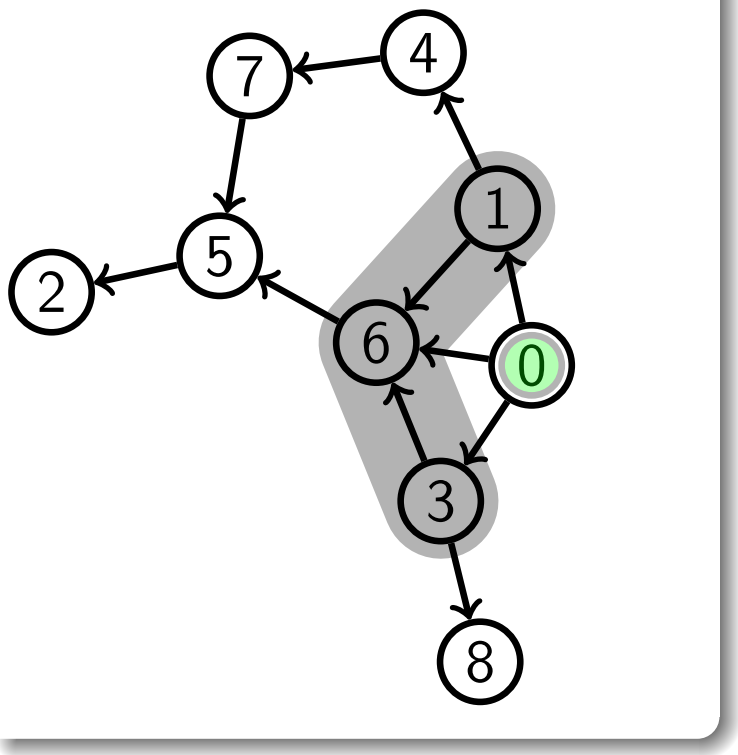
Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



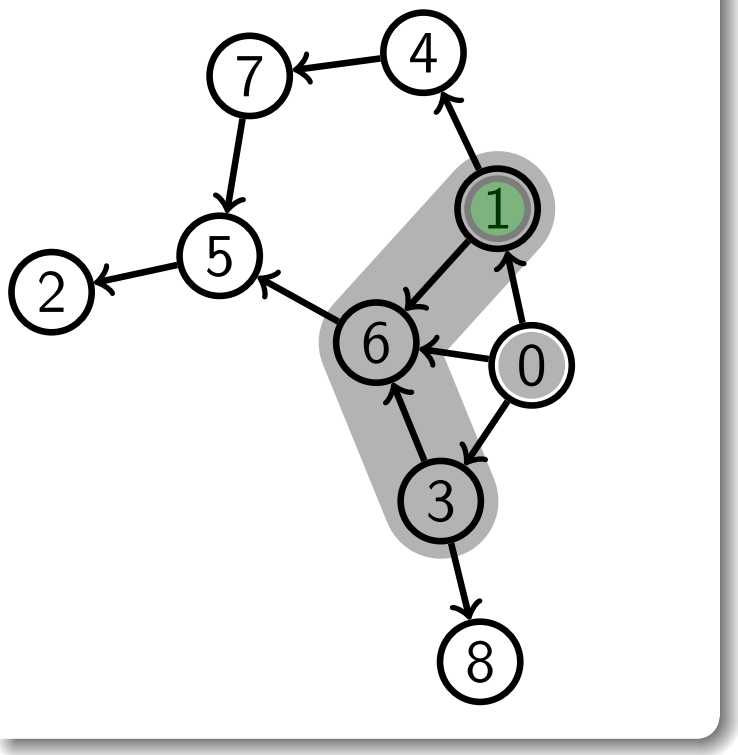
Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



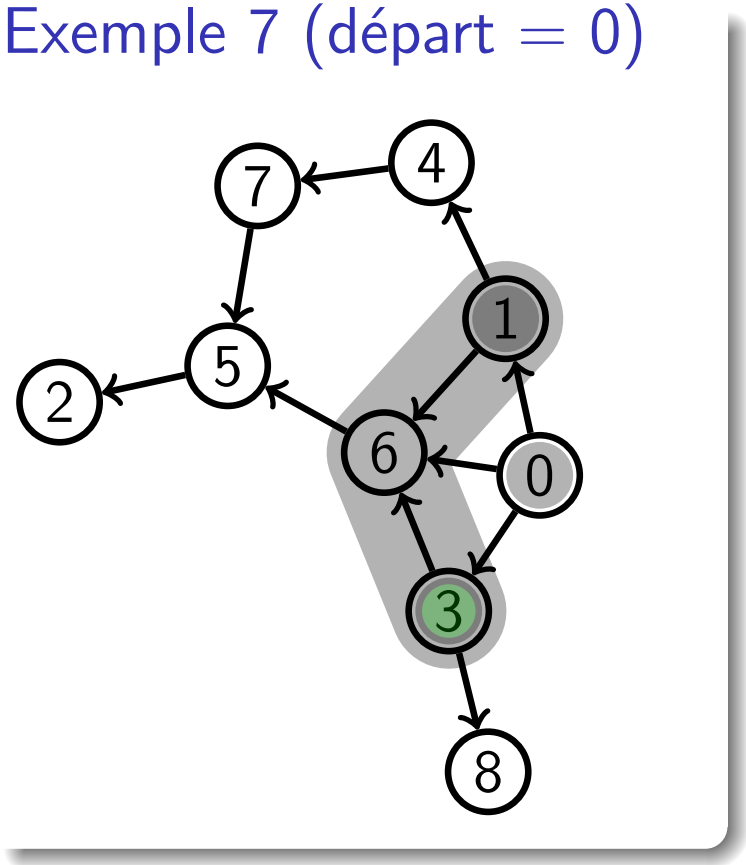
Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



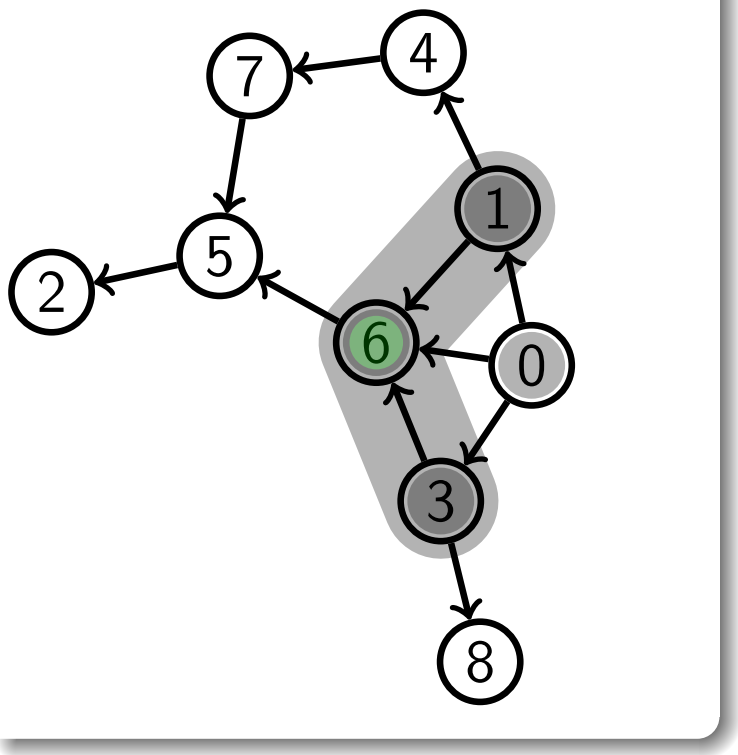
Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



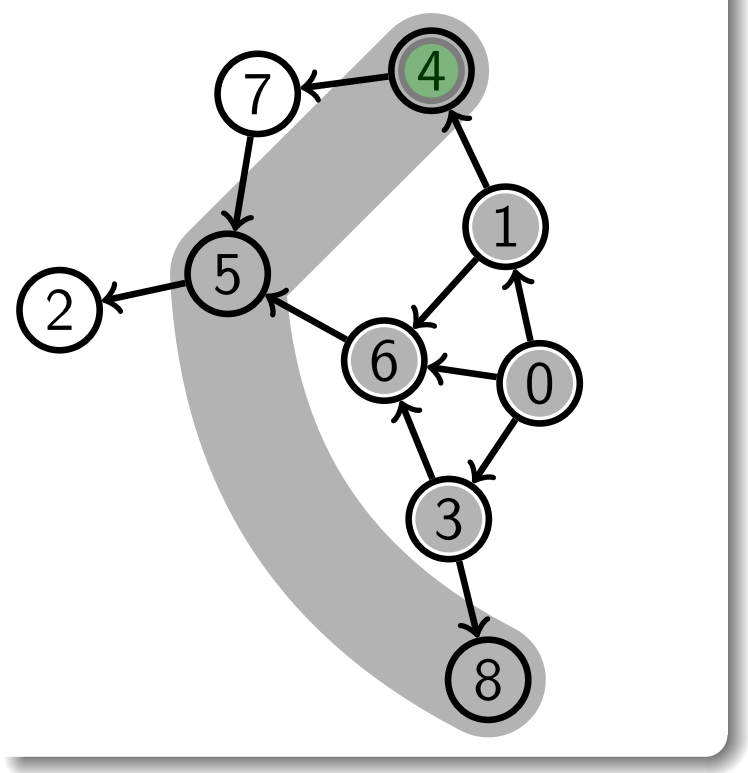
Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



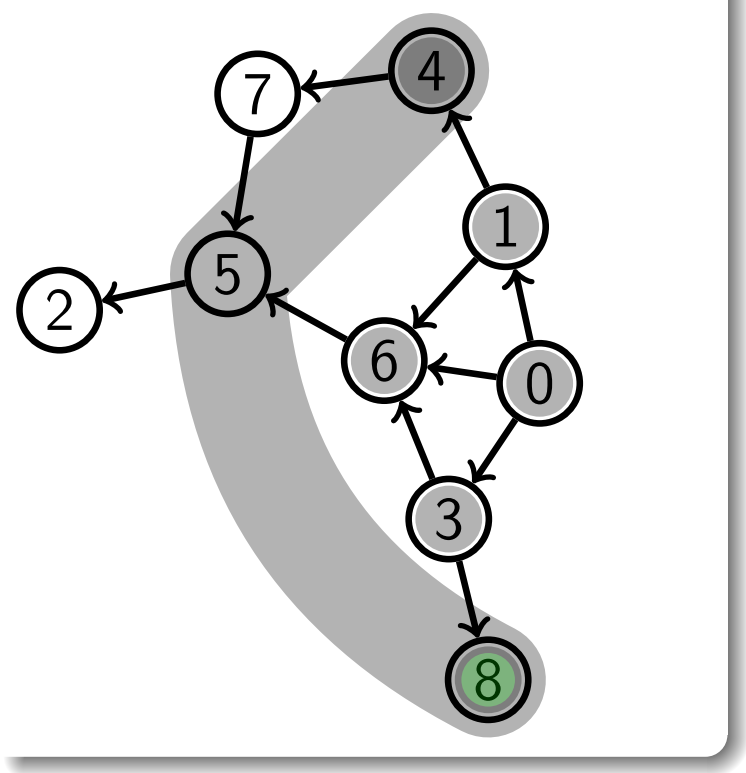
Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



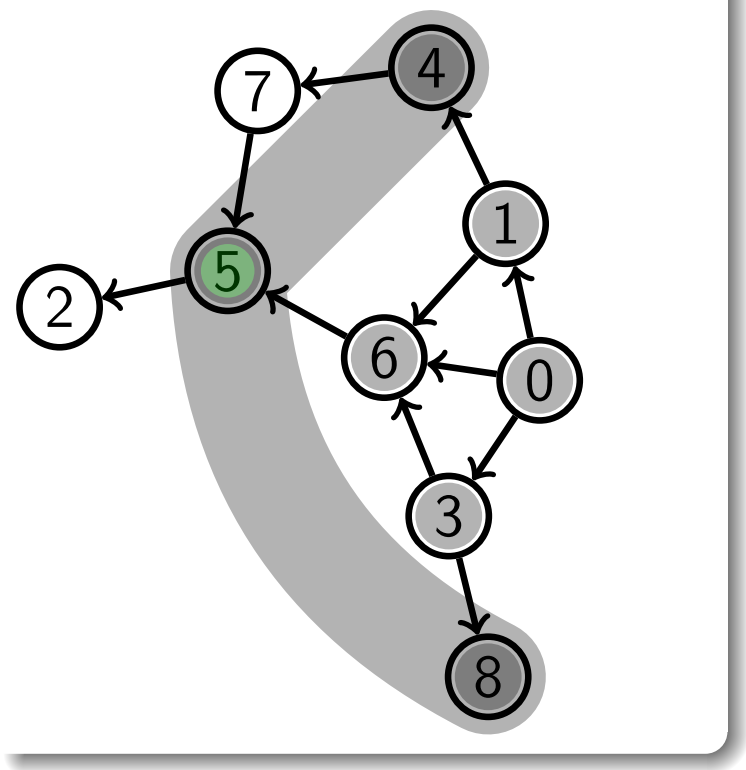
Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



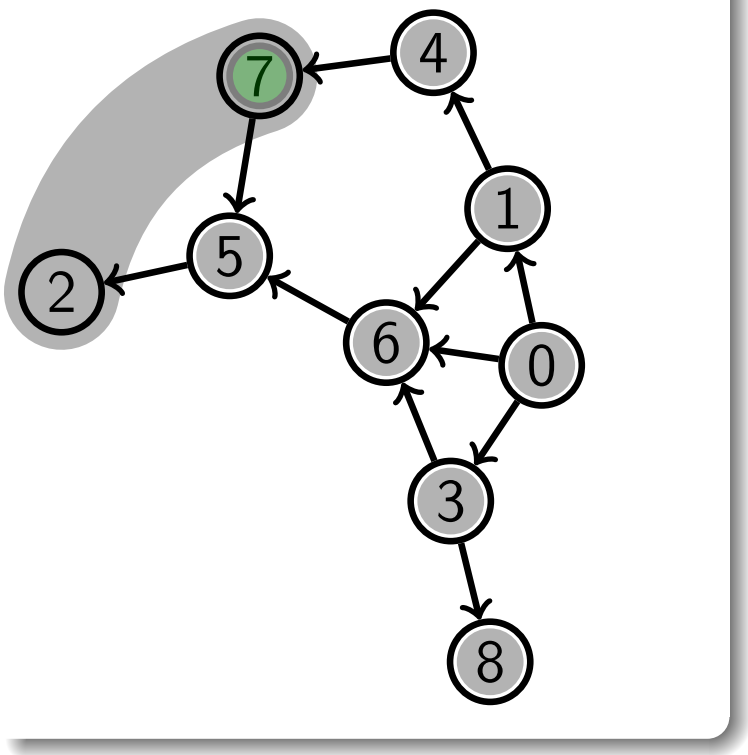
Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



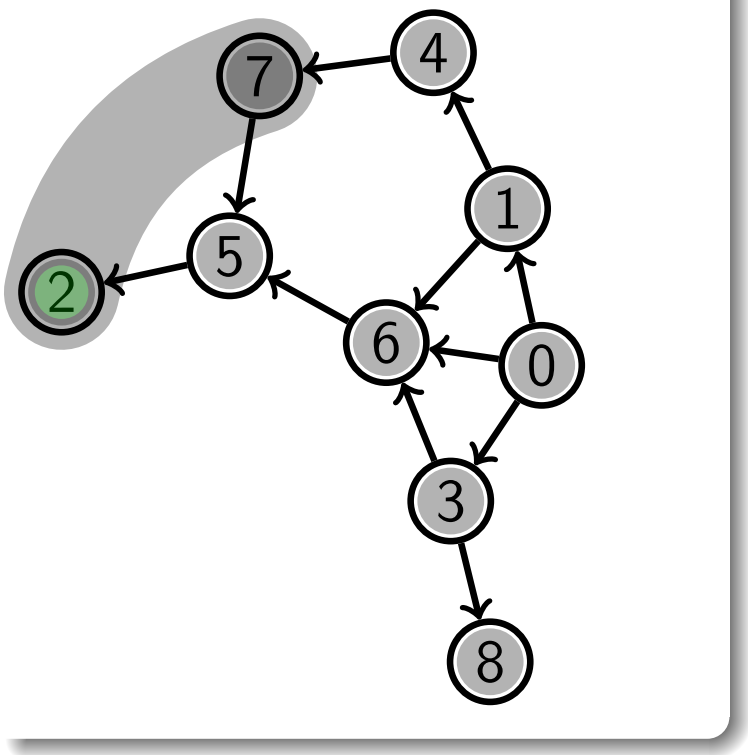
Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



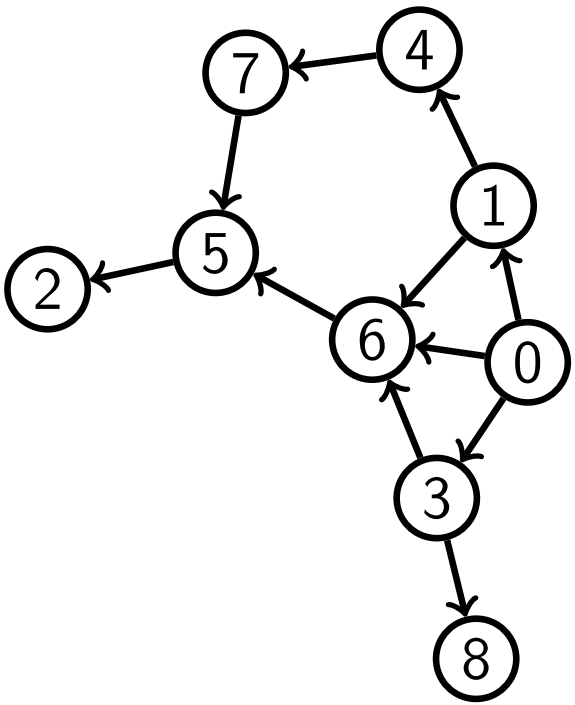
Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)

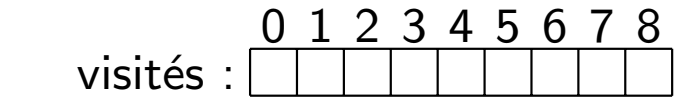


Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



Les coulisses



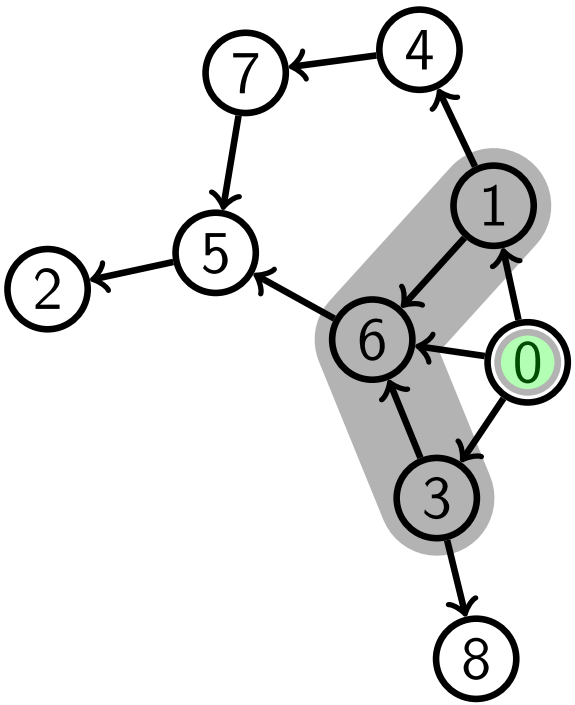
à traiter (file) : 0

résultat :

(file : fin à gauche, début à droite)

Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓								

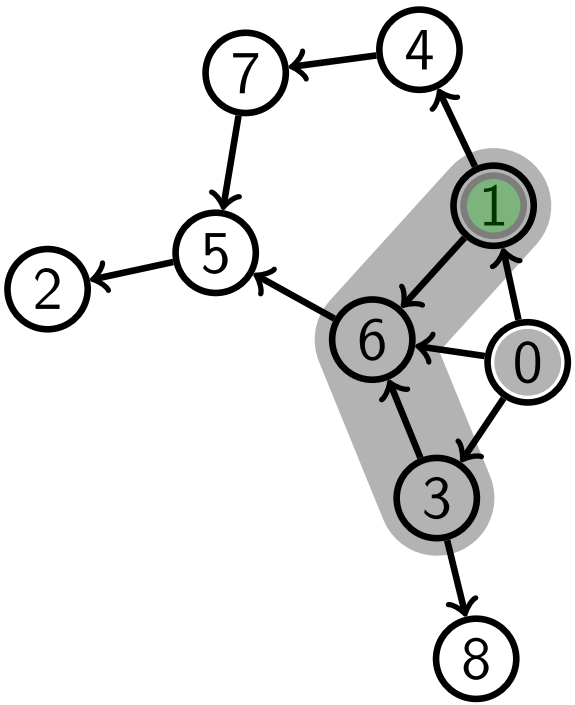
à traiter (file) : | 6 | 3 | 1 |

résultat : 0

(file : fin à gauche, début à droite)

Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓							

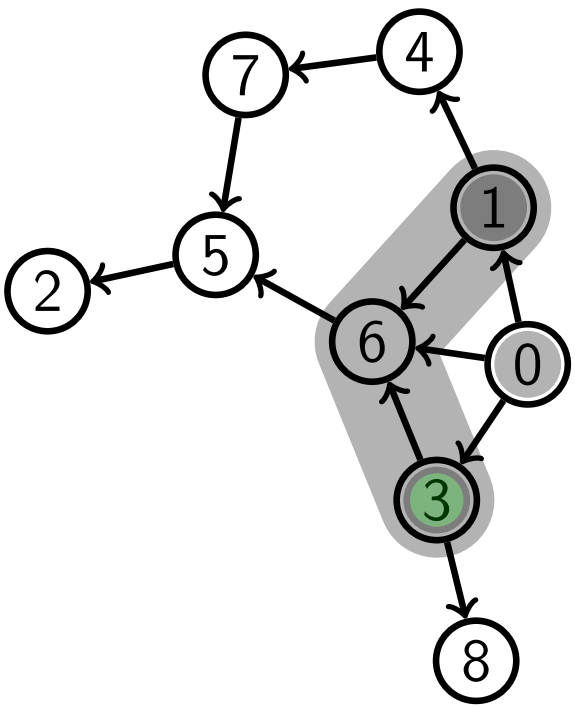
à traiter (file) :	6	4	6	3

résultat : 0 1

(file : fin à gauche, début à droite)

Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓		✓					

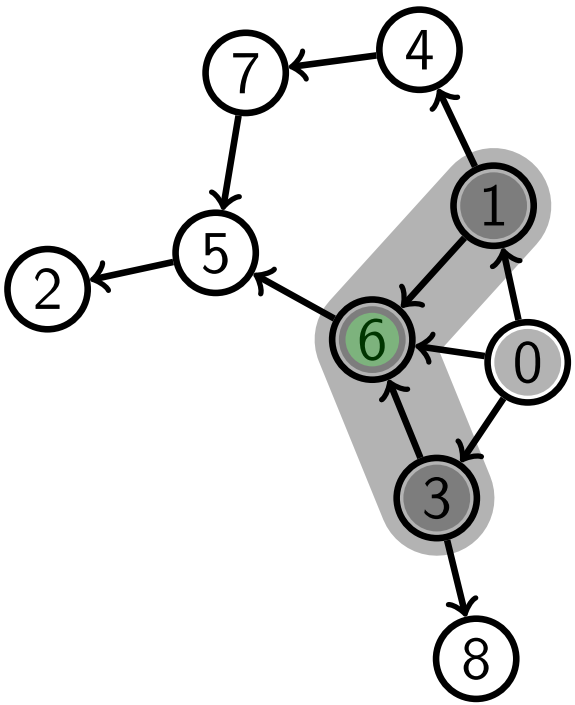
à traiter (file) :	8	6	6	4	6

résultat : 0 1 3

(file : fin à gauche, début à droite)

Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓		✓			✓		

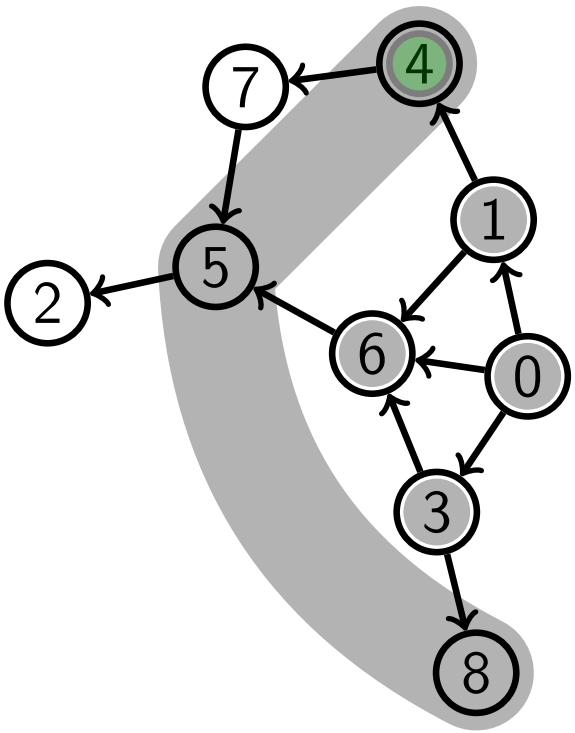
à traiter (file) :	5	8	6	6	4

résultat : 0 1 3 6

(file : fin à gauche, début à droite)

Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓		✓	✓		✓		

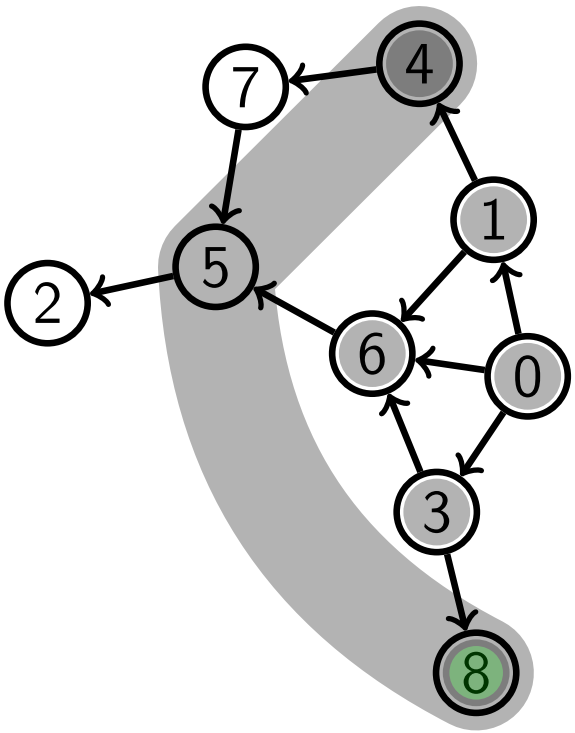
à traiter (file) :	7	5	8	6	6

résultat : 0 1 3 6 4

(file : fin à gauche, début à droite)

Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓		✓	✓		✓		✓

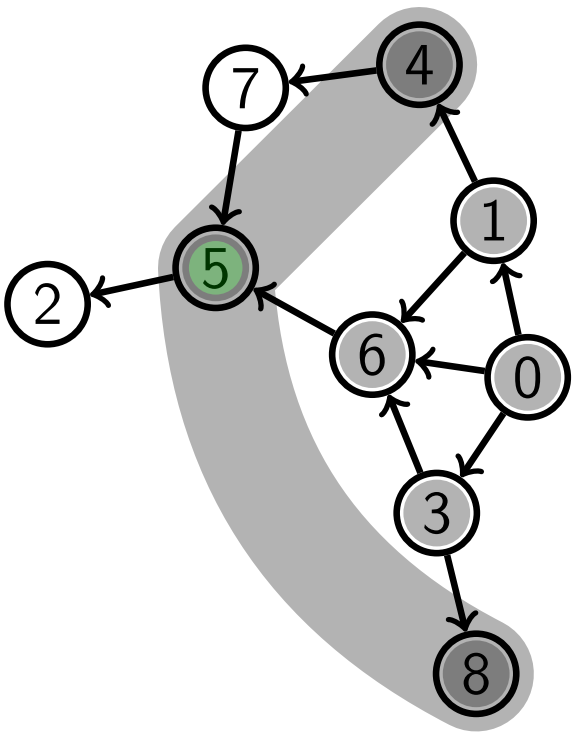
à traiter (file) :	5	5	7	5

résultat : 0 1 3 6 4 8

(file : fin à gauche, début à droite)

Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓		✓	✓	✓	✓		✓

à traiter (file) :

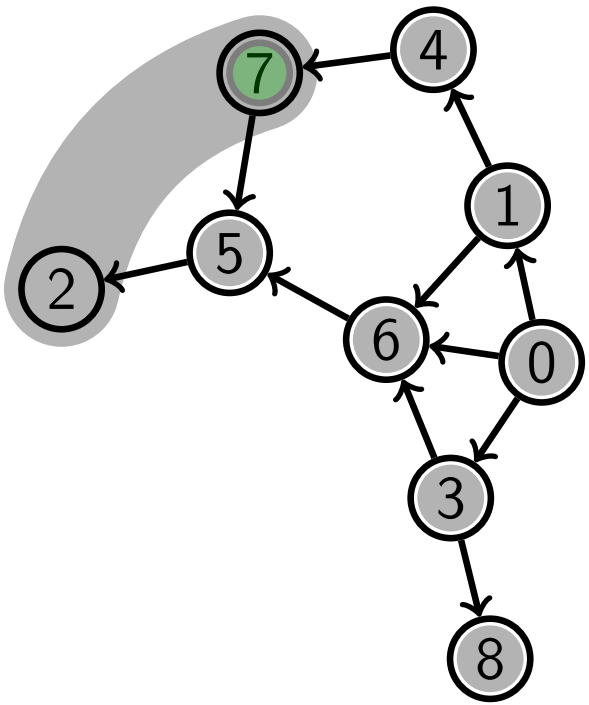
2	5	5	7

résultat : 0 1 3 6 4 8 5

(file : fin à gauche, début à droite)

Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓		✓	✓	✓	✓	✓	✓

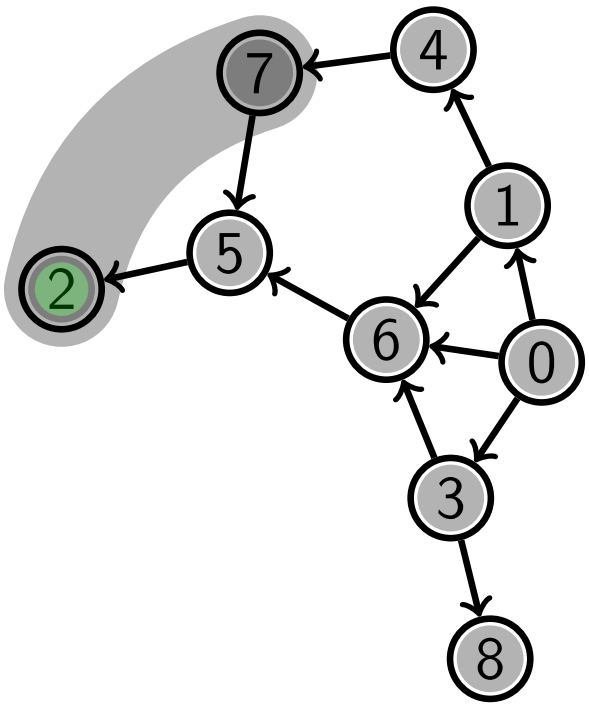
à traiter (file) : | 2 | 2 | 2 |

résultat : 0 1 3 6 4 8 5 7

(file : fin à gauche, début à droite)

Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓	✓	✓	✓	✓	✓	✓	✓

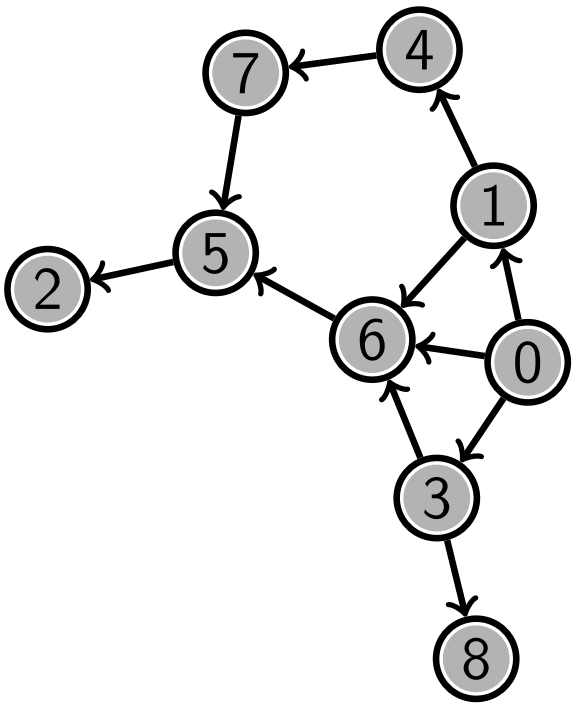
à traiter (file) :

résultat : 0 1 3 6 4 8 5 7 2

(file : fin à gauche, début à droite)

Parcours en largeur itératif : exemple

Exemple 7 (départ = 0)



Les coulisses

	0	1	2	3	4	5	6	7	8
visités :	✓	✓	✓	✓	✓	✓	✓	✓	✓

à traiter (file) :

résultat : 0 1 3 6 4 8 5 7 2

(file : fin à gauche, début à droite)

Algorithme 9 : LARGEUR(G, départ, visités=NIL)

Entrées : un graphe orienté G, un sommet de départ, un tableau (facultatif) visités de $|V|$ cases indiquant les sommets déjà traités.

Sortie : les sommets de G accessibles depuis le départ dans l'ordre où le parcours en largeur les a découverts.

```
1  résultat ← liste();
2  si visités = NIL alors visités ← tableau(G.nombre_sommets(), FAUX);
3  a_traiter ← file();
4  a_traiter.enfiler(départ);
5  tant que a_traiter.pas_vide() faire
6  |   sommet ← a_traiter.défiler();
7  |   si ¬ visités[sommet] alors
8  |   |   résultat.ajouter_en_fin(sommet);
9  |   |   visités[sommet] ← VRAI;
10 |   |   pour chaque successeur dans G.successeurs(sommet) faire
11 |   |   |   si ¬ visités[successeur] alors a_traiter.enfiler(successeur) ;
12 renvoyer résultat;
```

Parcours en largeur itératif

Note : il peut y avoir plusieurs occurrences d'un même sommet dans la file. On peut l'interdire !

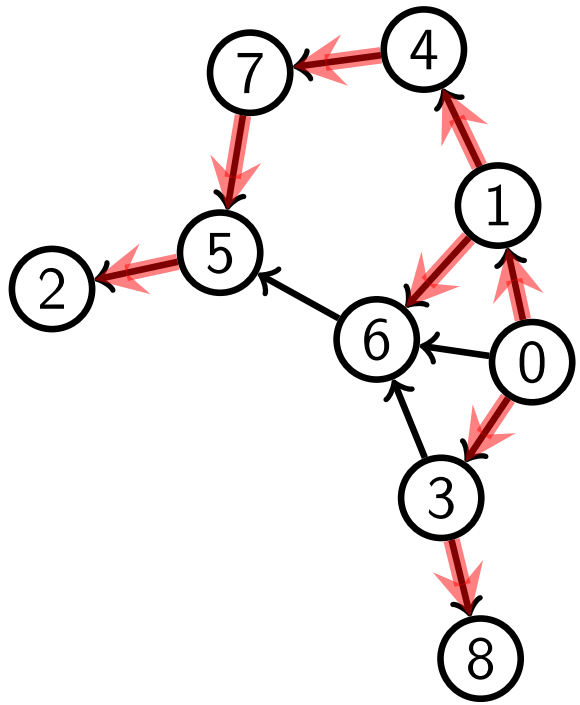
Forêt recouvrante pour les parcours

Étant donné une exploration, le graphe obtenu à partir de G en gardant uniquement les arcs qui donne accès à un nouveau sommet découvert lors du parcours est la forêt recouvrante.

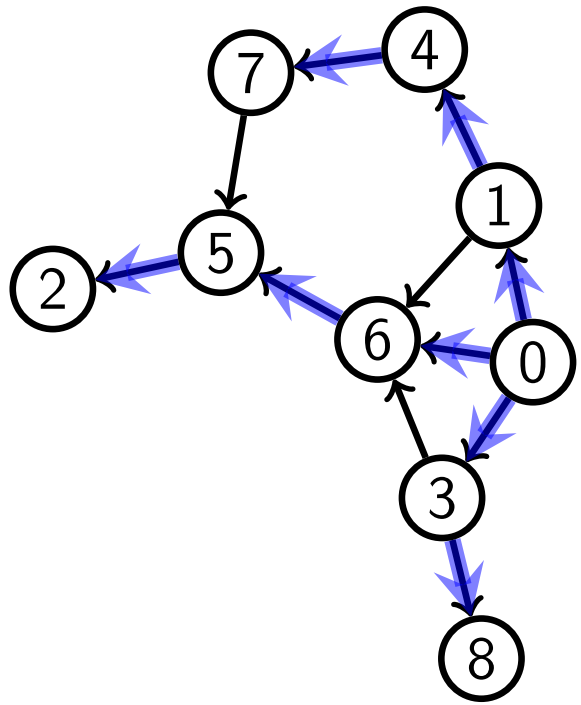
- La forêt recouvrante comprend tous les sommets du graphe G.
- C'est une union fini d'arbres, appelés arbres de parcours.

Forêt recouvrante pour les parcours

Exemple 8 (arbres de parcours partant de 0)



profondeur



largeur

Complexité des parcours itératifs

- Pour chaque sommet v, on doit accéder à tous ses successeurs ;

Complexité des parcours itératifs

- Pour chaque sommet v, on doit accéder à tous ses successeurs ;
  - matrice d'adjacence : G.successeurs(v) est en $O(|V|)$;

Complexité des parcours itératifs

- Pour chaque sommet v, on doit accéder à tous ses successeurs ;
  - matrice d'adjacence : G.successeurs(v) est en $O(|V|)$;
  - listes d'adjacence : G.successeurs(v) est en $O(\deg^+(v))$;

Complexité des parcours itératifs

- Pour chaque sommet v, on doit accéder à tous ses successeurs ;
  - matrice d'adjacence : G.successeurs(v) est en $O(|V|)$;
  - listes d'adjacence : G.successeurs(v) est en $O(\deg^+(v))$;
- Chaque sommet est marqué visité une seule fois et chacun de ses successeurs est alors entré dans la pile ou la file.

Complexité des parcours itératifs

- Pour chaque sommet v, on doit accéder à tous ses successeurs ;
  - matrice d'adjacence : G.successeurs(v) est en $O(|V|)$;
  - listes d'adjacence : G.successeurs(v) est en $O(\deg^+(v))$;
- Chaque sommet est marqué visité une seule fois et chacun de ses successeurs est alors entré dans la pile ou la file.
- Nos parcours ont donc une complexité de :

Complexité des parcours itératifs

- Pour chaque sommet v, on doit accéder à tous ses successeurs ;
  - matrice d'adjacence : G.successeurs(v) est en $O(|V|)$;
  - listes d'adjacence : G.successeurs(v) est en $O(\deg^+(v))$;
- Chaque sommet est marqué visité une seule fois et chacun de ses successeurs est alors entré dans la pile ou la file.
- Nos parcours ont donc une complexité de :
  - $O(|V|^2)$ pour une matrice d'adjacence ;

Complexité des parcours itératifs

- Pour chaque sommet v, on doit accéder à tous ses successeurs ;
  - matrice d'adjacence : G.successeurs(v) est en $O(|V|)$;
  - listes d'adjacence : G.successeurs(v) est en $O(\deg^+(v))$;
- Chaque sommet est marqué visité une seule fois et chacun de ses successeurs est alors entré dans la pile ou la file.
- Nos parcours ont donc une complexité de :
  - $O(|V|^2)$ pour une matrice d'adjacence ;
  - $O(|V| + |A|)$ pour des listes d'adjacence ($\sum_{v \in V} \deg^+(v) = |A|$) ;

Comparaison des algorithmes

Comparons les deux algorithmes de parcours itératifs :

Profondeur

```
1  résultat ← liste();
2  si visités = NIL alors visités ←
     tableau(G.nombre_sommets(), FAUX);
3  a_traiter ← pile();
4  a_traiter.empiler(départ);
5  tant que a_traiter.pas_vide() faire
6      sommet ← a_traiter.dépiler();
7      si ¬ visités[sommet] alors
8          résultat.ajouter_en_fin(sommet);
9          visités[sommet] ← VRAI;
10         pour chaque successeur dans
             renverser(G.successeurs(sommet))
             faire
11             si ¬ visités[successeur] alors
                 a_traiter.empiler(successeur)
                 ;
12 renvoyer résultat;
```

Largeur

```
1  résultat ← liste();
2  si visités = NIL alors visités ←
     tableau(G.nombre_sommets(), FAUX);
3  a_traiter ← file();
4  a_traiter.enfiler(départ);
5  tant que a_traiter.pas_vide() faire
6      sommet ← a_traiter.défiler();
7      si ¬ visités[sommet] alors
8          résultat.ajouter_en_fin(sommet);
9          visités[sommet] ← VRAI;
10         pour chaque successeur dans
             G.successeurs(sommet) faire
11             si ¬ visités[successeur] alors
                 a_traiter.enfiler(successeur) ;
12 renvoyer résultat;
```

Seul le type de la structure de données a_traiter change !