Algorithmique des graphes

2 — Parcours et applications

Anthony Labarre

3 février 2021

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 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 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 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 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

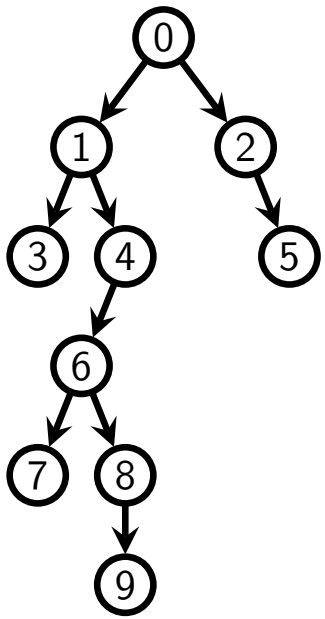
- En guise d'échauffement, examinons les parcours sur des arbres (binaires) ;
- Les arbres peuvent être vus comme des graphes très particuliers :
  1. ils ne possèdent pas de cycles ;
  2. le point de départ est fixé (c'est la racine) ;
- On les parcourt fréquemment de deux manières :
  - en **profondeur**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

Échauffement : parcours d'arbres

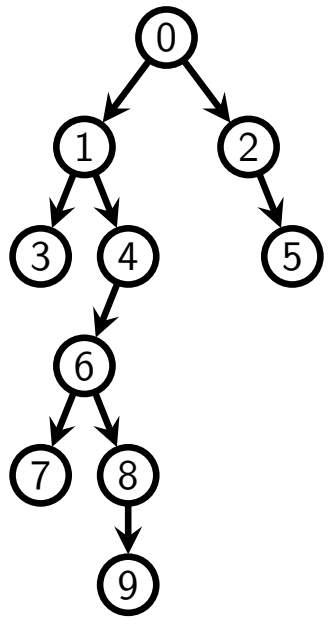
- En guise d'échauffement, examinons les parcours sur des arbres (binaires) ;
- Les arbres peuvent être vus comme des graphes très particuliers :
  1. ils ne possèdent pas de cycles ;
  2. le point de départ est fixé (c'est la racine) ;
- On les parcourt fréquemment de deux manières :
  - en **profondeur**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

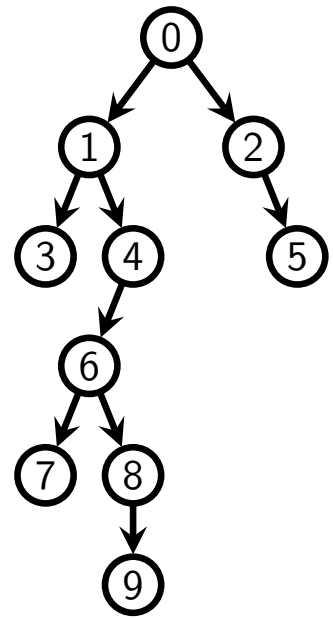
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

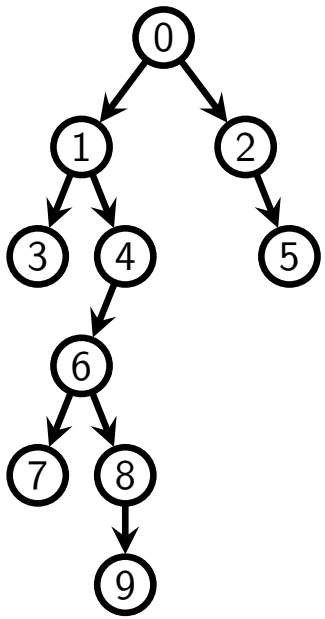


en largeur

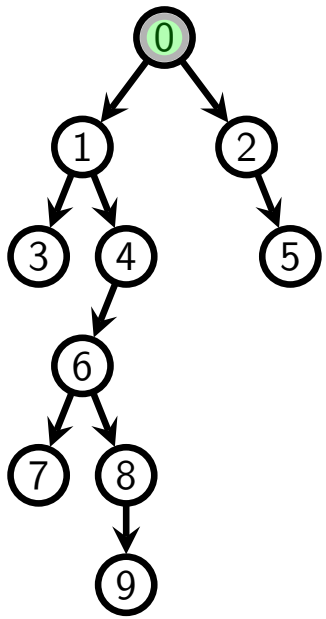
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

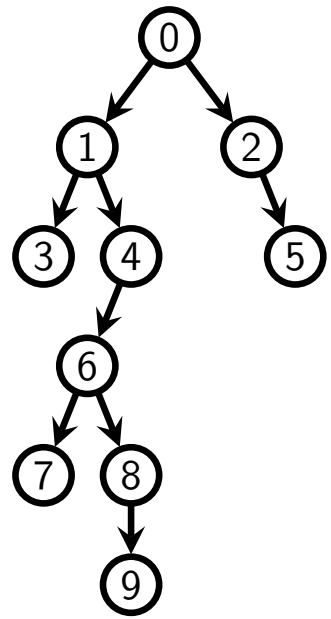
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

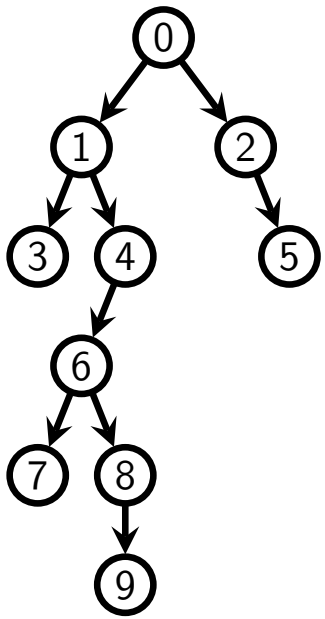


en largeur

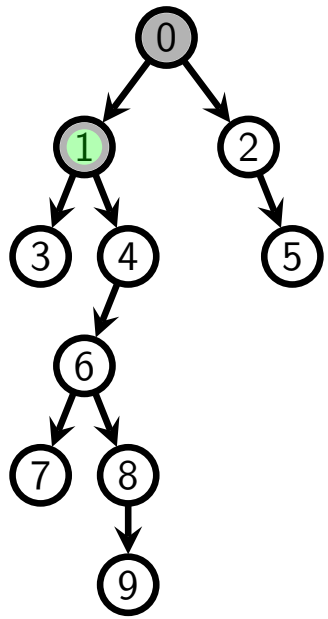
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

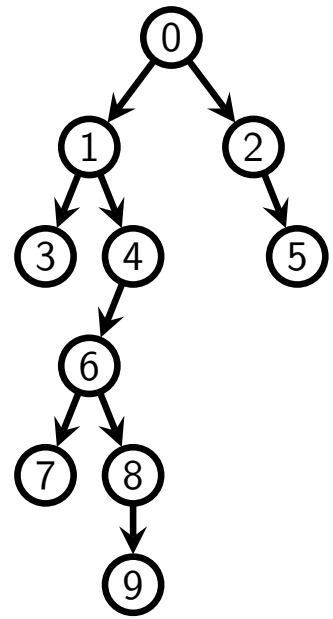
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

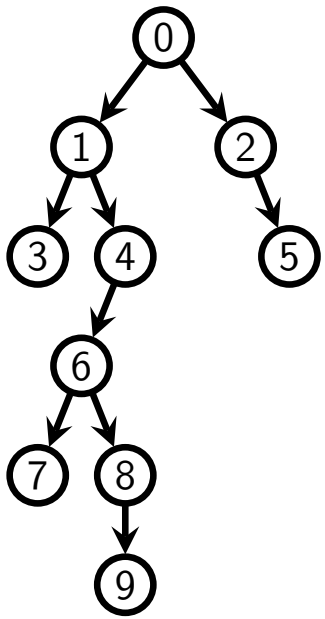


en largeur

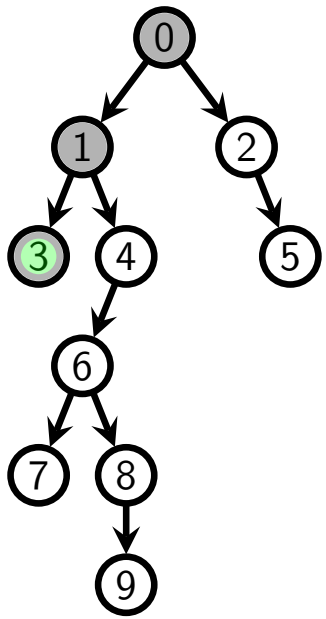
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

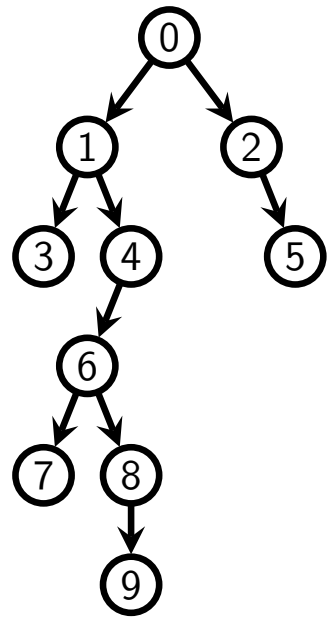
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

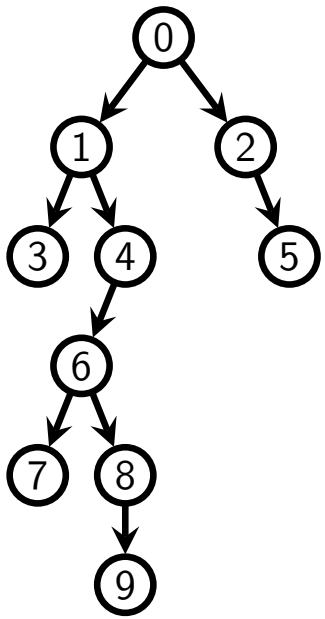


en largeur

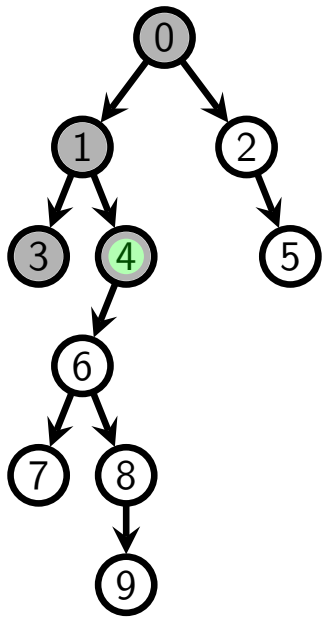
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

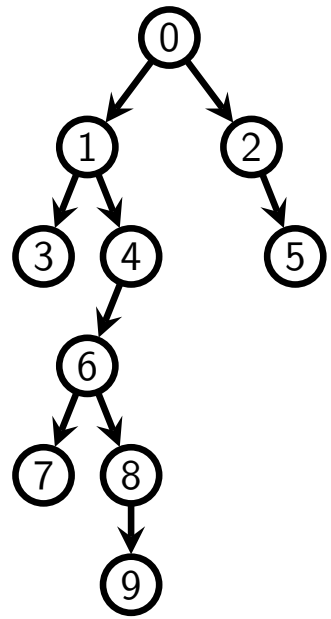
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

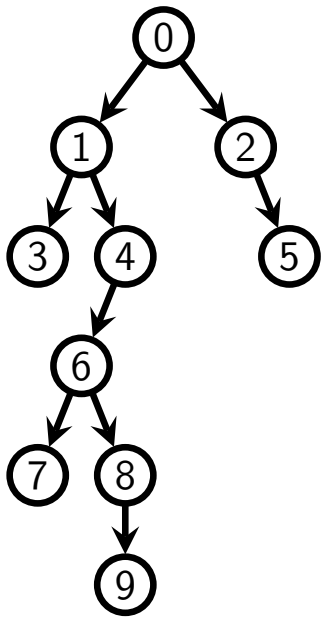


en largeur

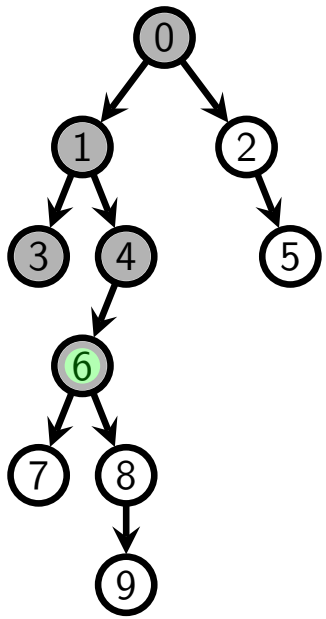
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

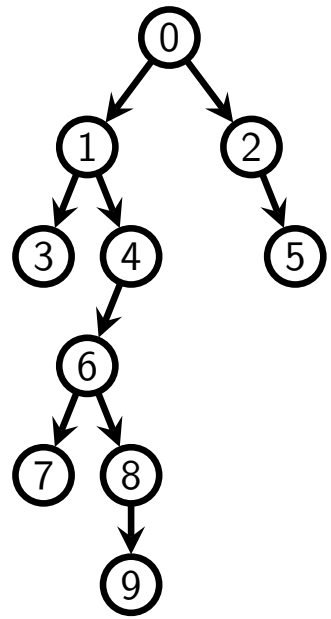
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

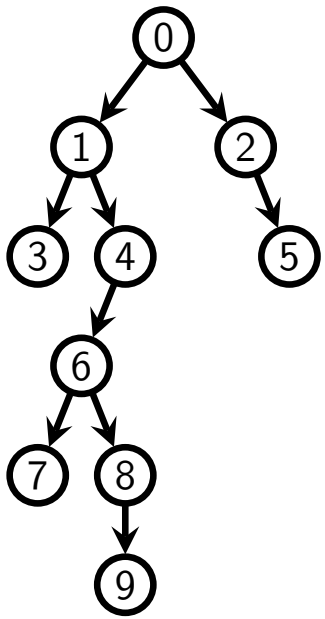


en largeur

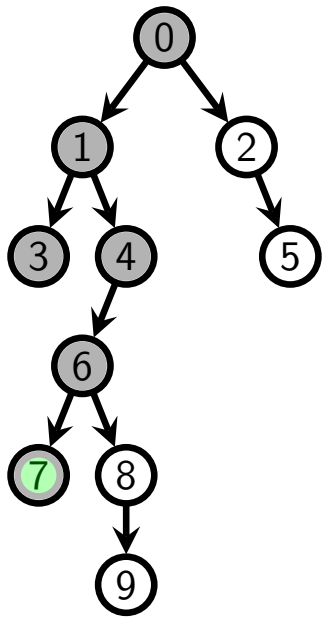
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

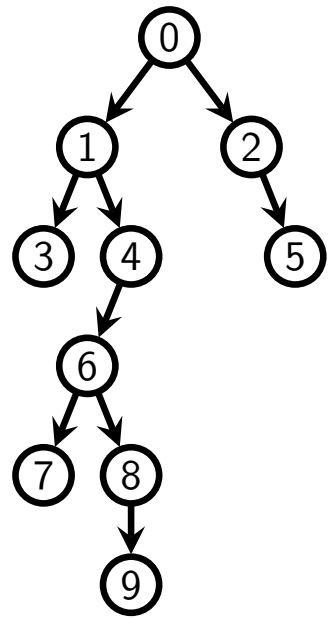
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

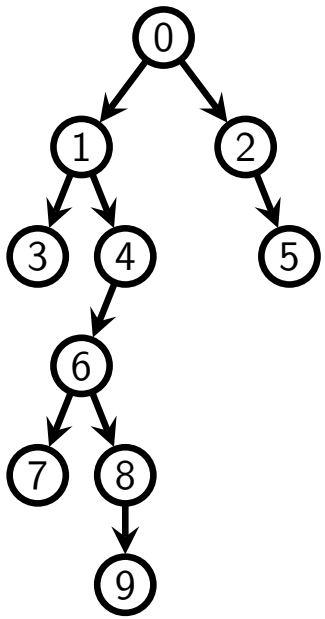


en largeur

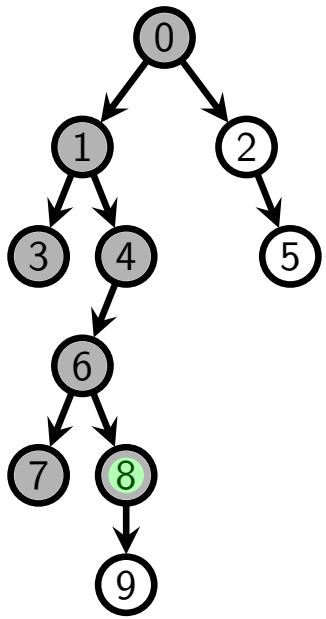
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

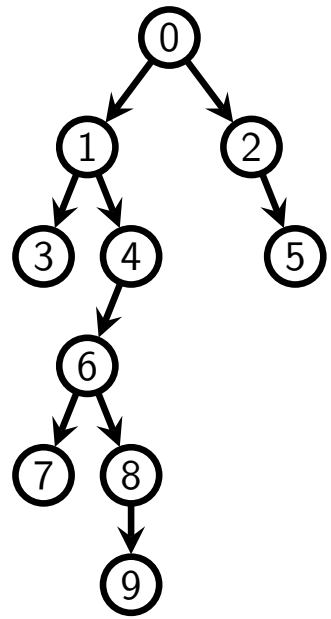
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

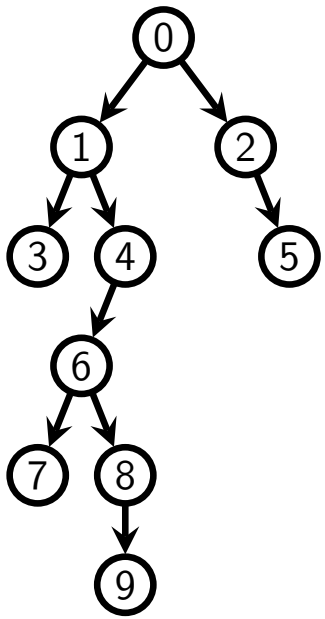


en largeur

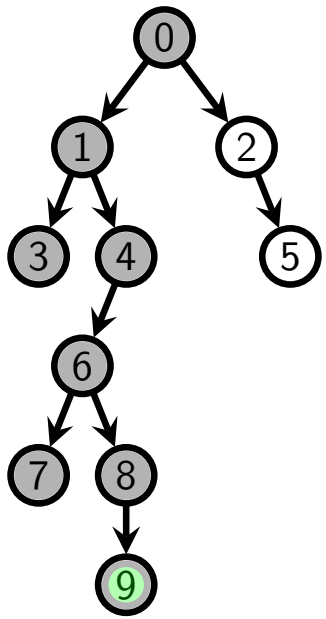
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

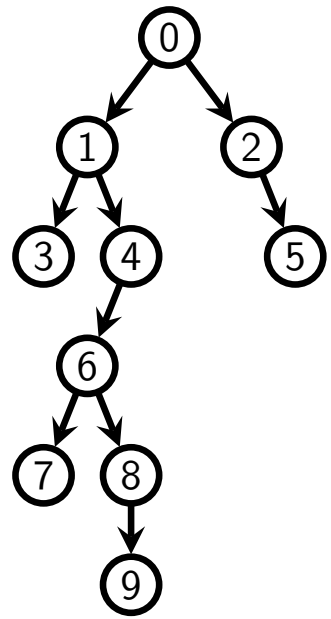
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

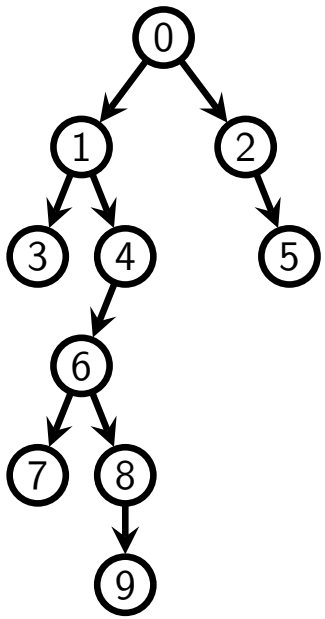


en largeur

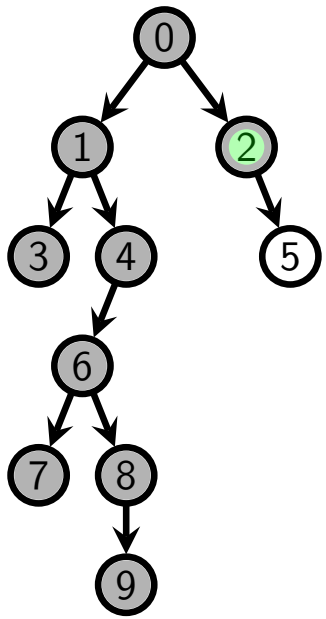
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

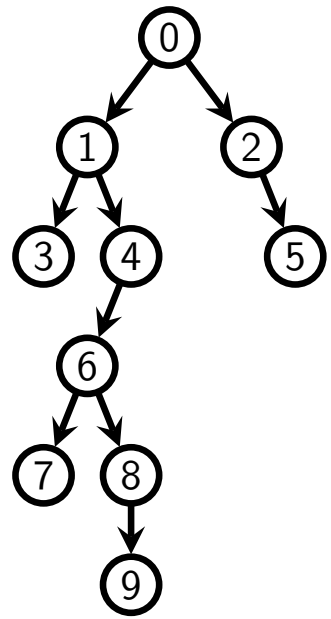
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

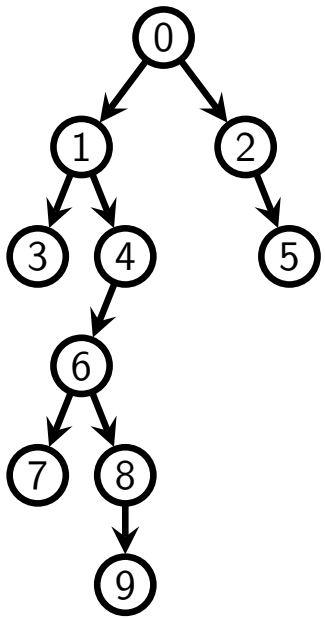


en largeur

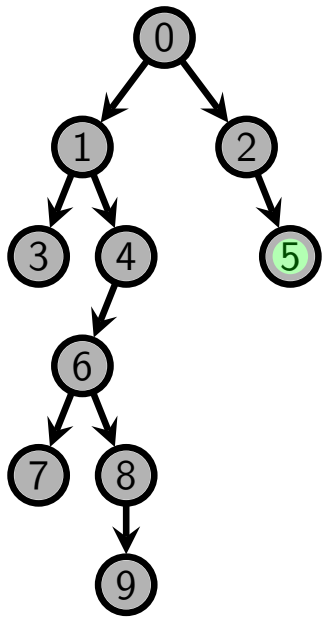
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

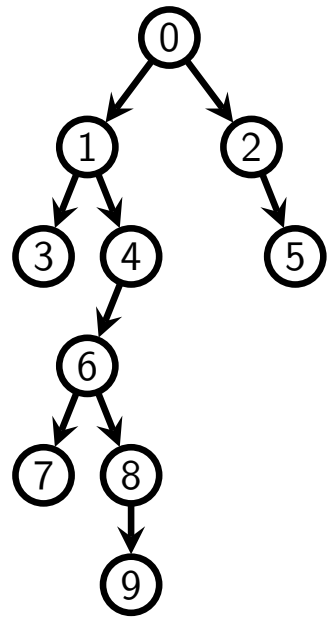
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

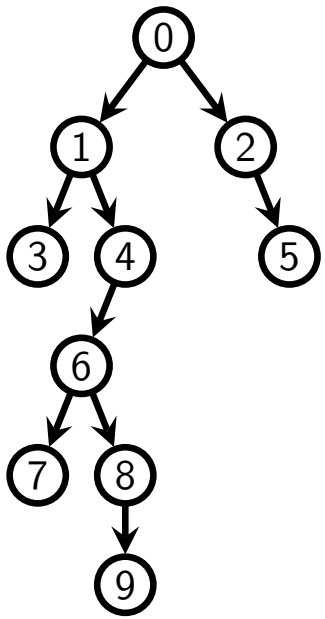


en largeur

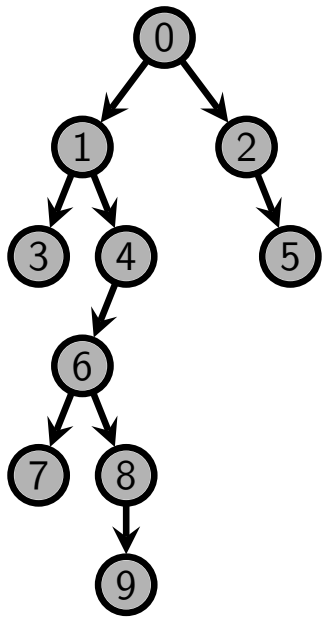
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

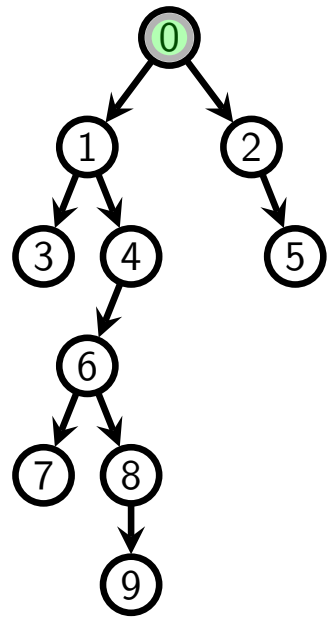
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

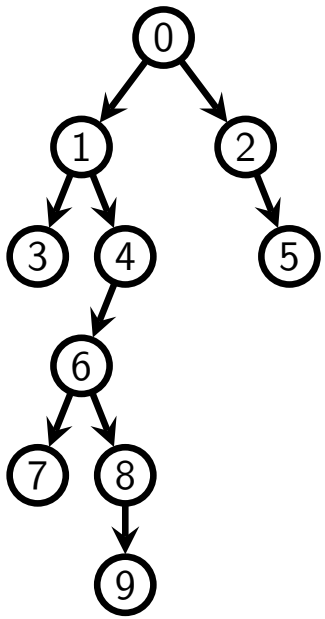


en largeur

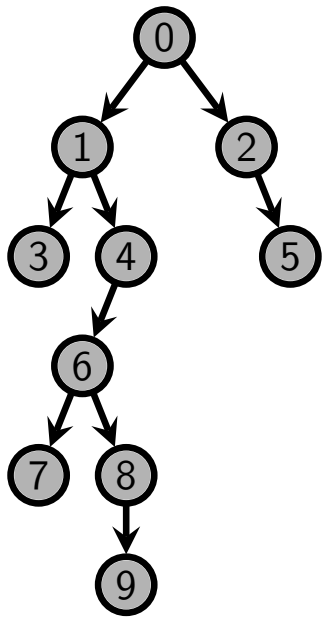
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

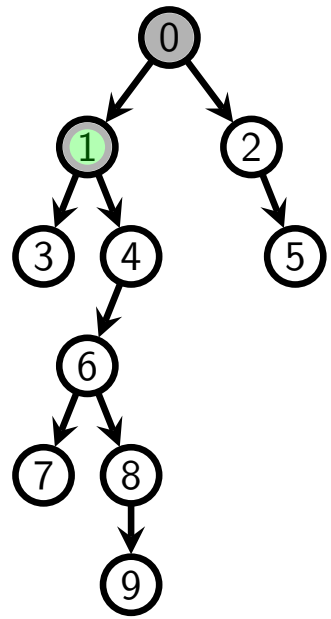
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

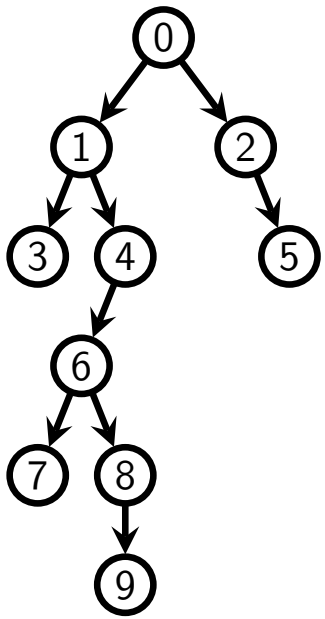


en largeur

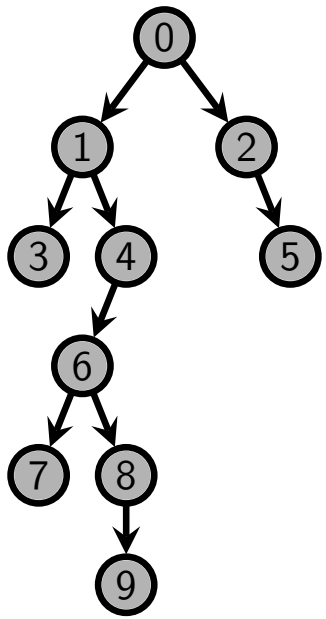
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

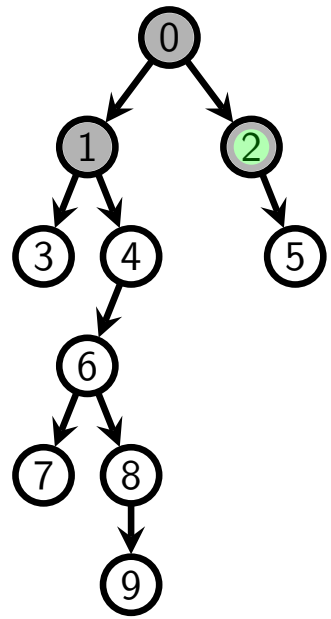
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

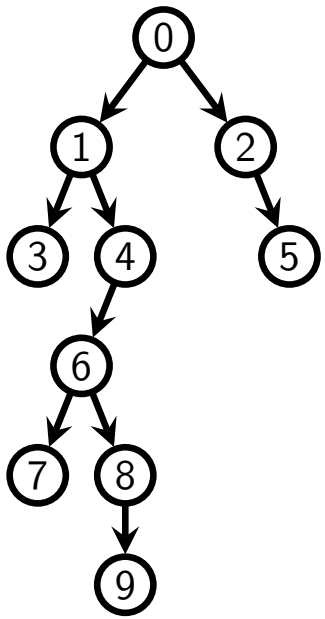


en largeur

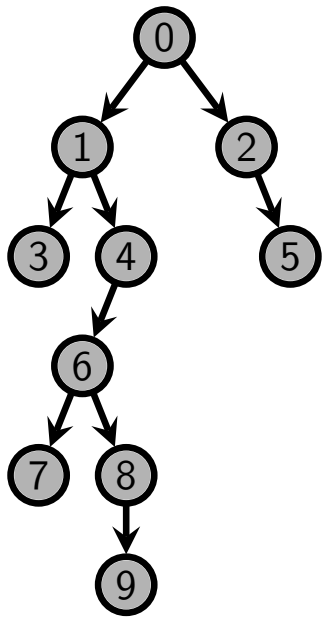
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

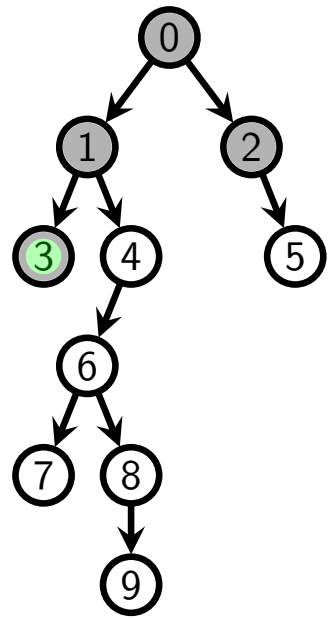
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

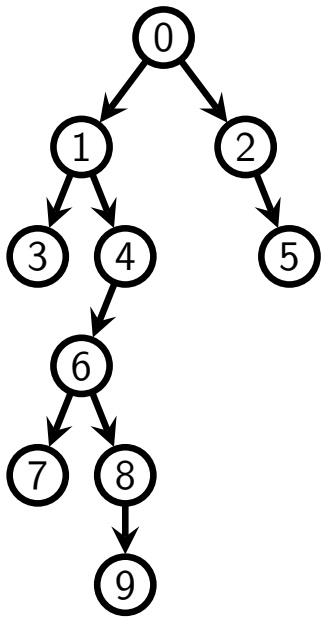


en largeur

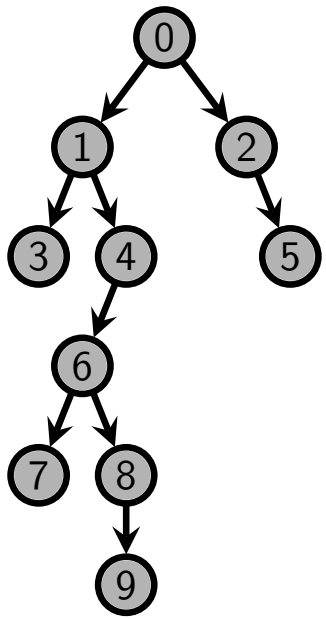
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

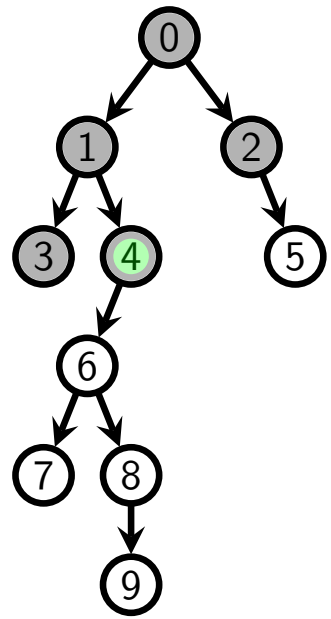
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

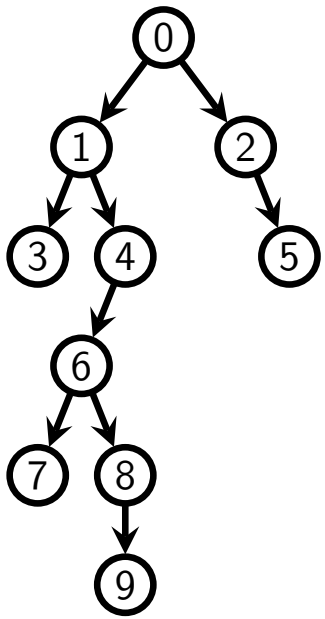


en largeur

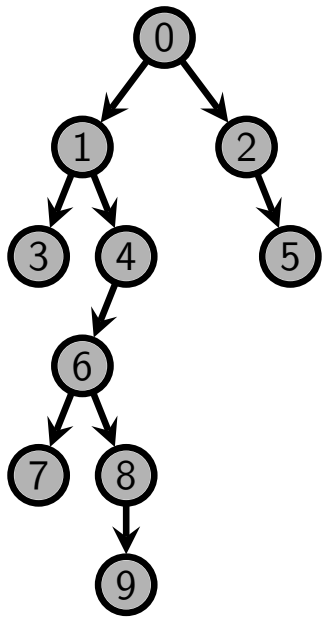
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

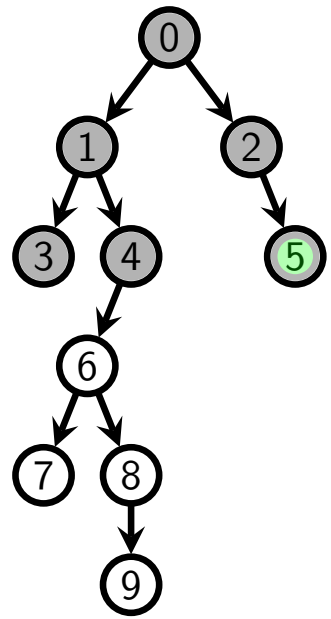
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

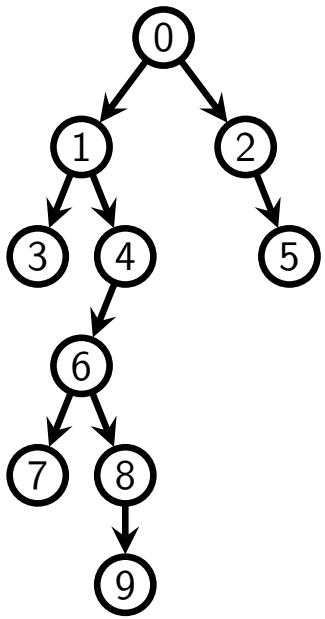


en largeur

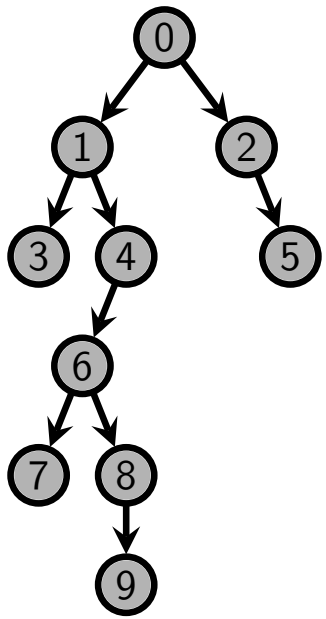
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

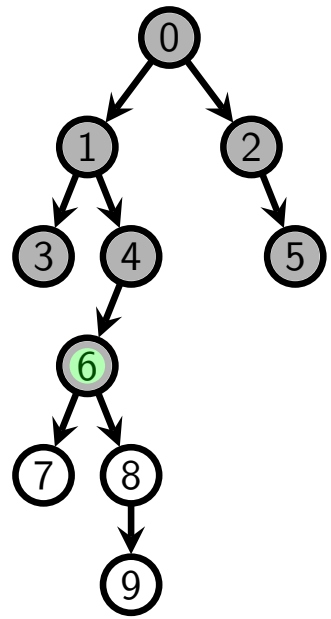
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

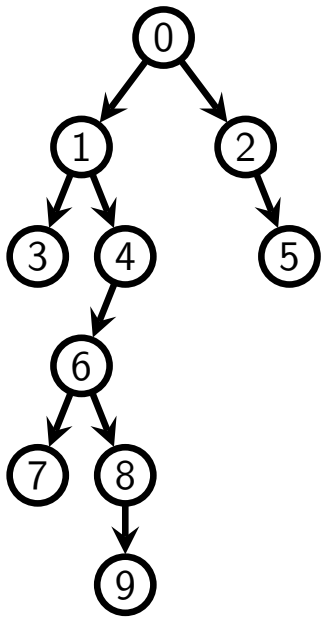


en largeur

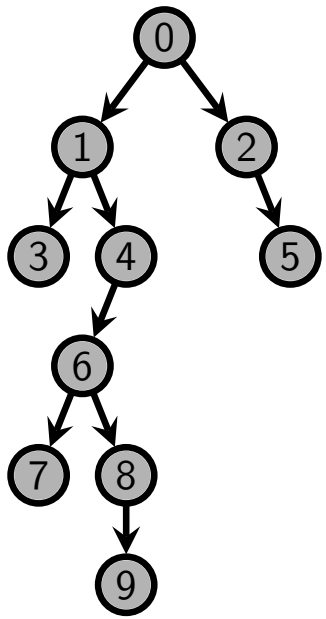
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

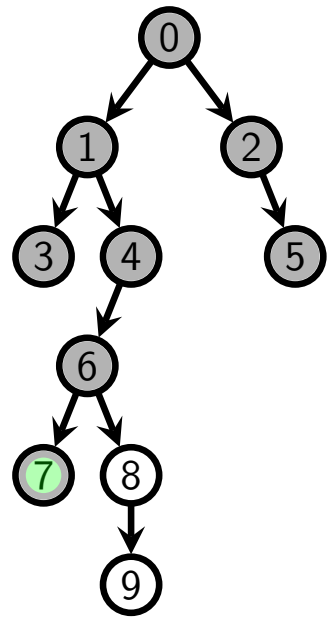
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

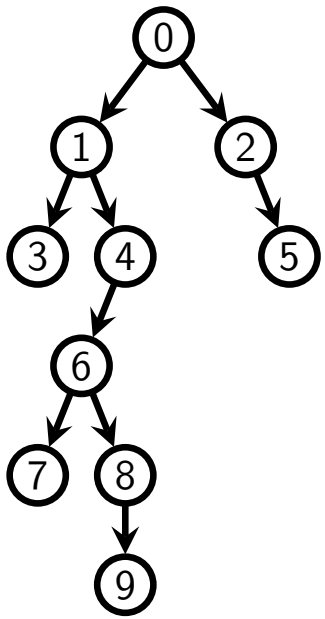


en largeur

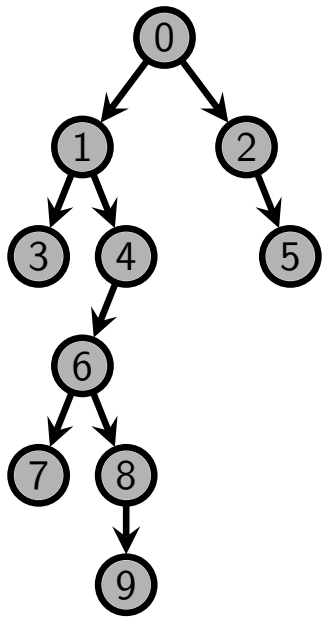
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

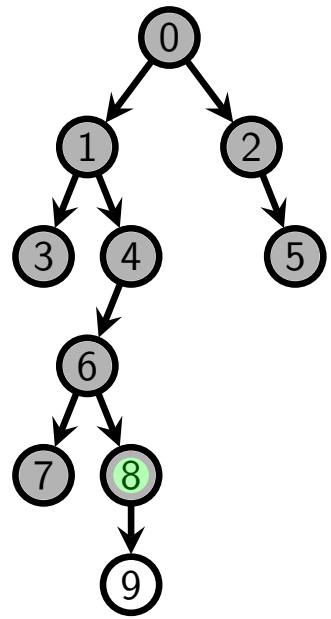
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

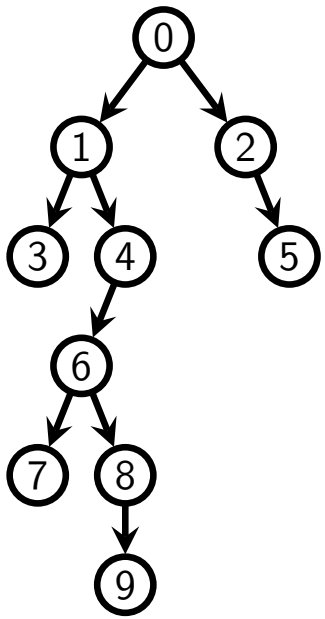


en largeur

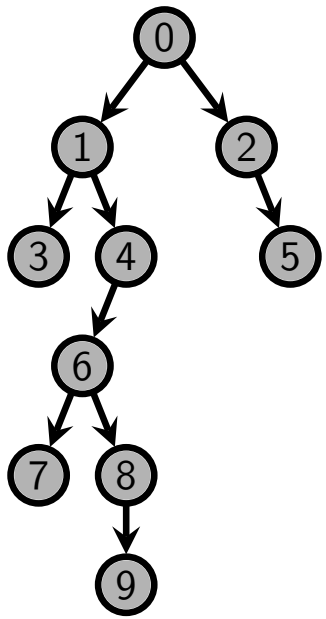
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

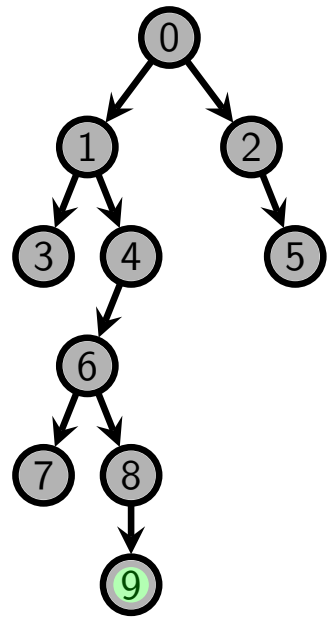
Exemple 1 (parcours d'un arbre binaire)



arbre



en profondeur

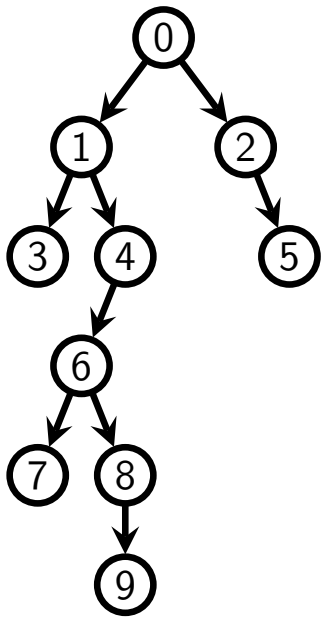


en largeur

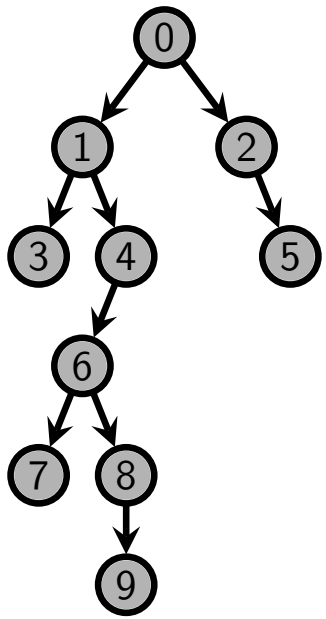
- en profondeur : 0 1 3 4 6 7 8 9 2 5 ;
- en largeur : 0 1 2 3 4 5 6 7 8 9 ;

Parcours d'arbres en profondeur et en largeur

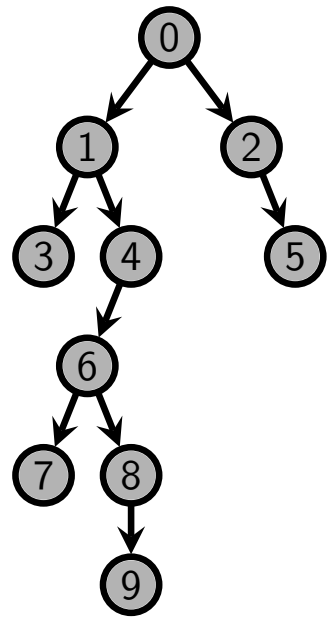
Exemple 1 (parcours d'un arbre binaire)



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 descendant dans A.successeurs(sommet) faire
8 |   |   a_traiter.enfiler(descendant);
9 renvoyer résultat;
```

Parcours de graphes

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

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 (encore) d'orientation ou d'ordre sur les voisins ;

Parcours de graphes

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 (encore) d'orientation ou d'ordre sur les voisins ;
3. certains sommets sont accessibles par plusieurs chemins, il faudra donc se rappeler de ce qu'on a déjà examiné ;

Parcours de graphes

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 (encore) d'orientation ou d'ordre sur les voisins ;
3. certains sommets sont accessibles par plusieurs chemins, il faudra donc se rappeler de ce qu'on a déjà examiné ;

Conventions : on suppose que :

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

Parcours de graphe en profondeur

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

Parcours de graphe en profondeur

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

Parcours de graphe en profondeur

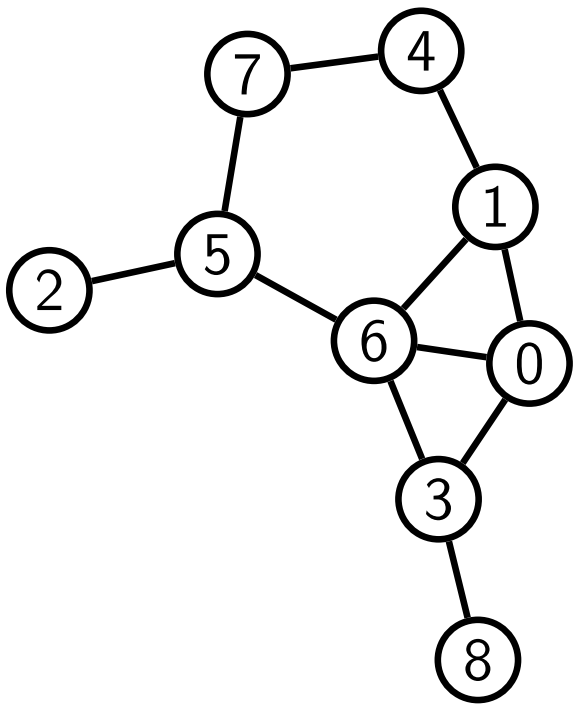
- Le principe du parcours en profondeur se généralise comme suit aux graphes :
  1. si le sommet actuel u n'a pas encore été visité, l'afficher ;
  2. pour chaque voisin v non encore visité de u : parcourir v et ses descendants en profondeur.

Parcours de graphe en profondeur

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

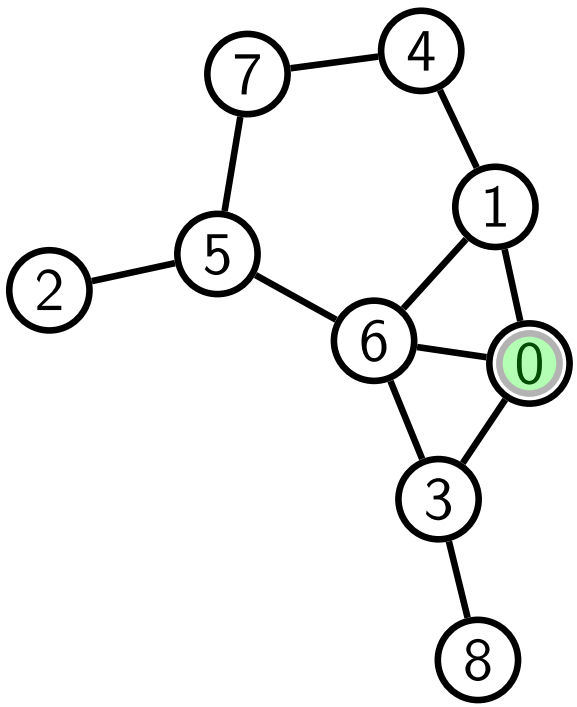
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



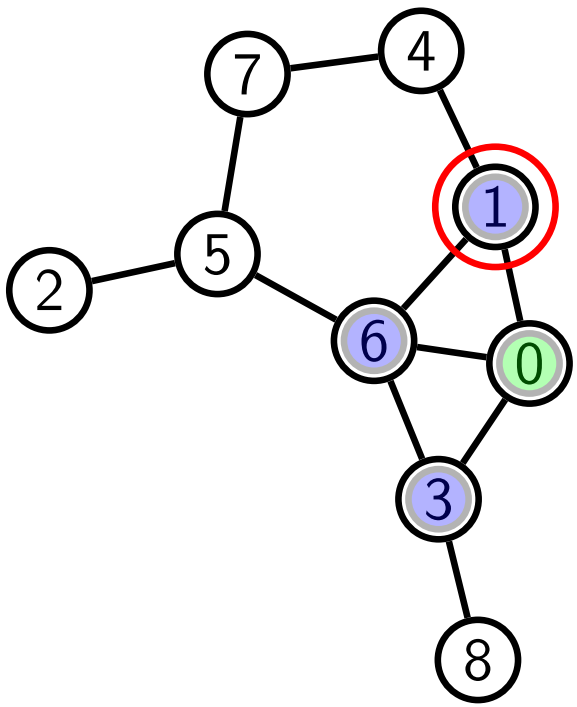
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



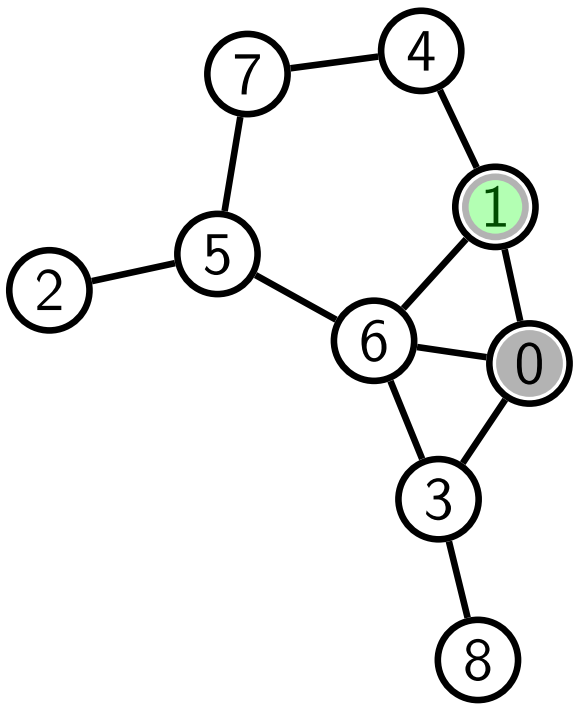
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



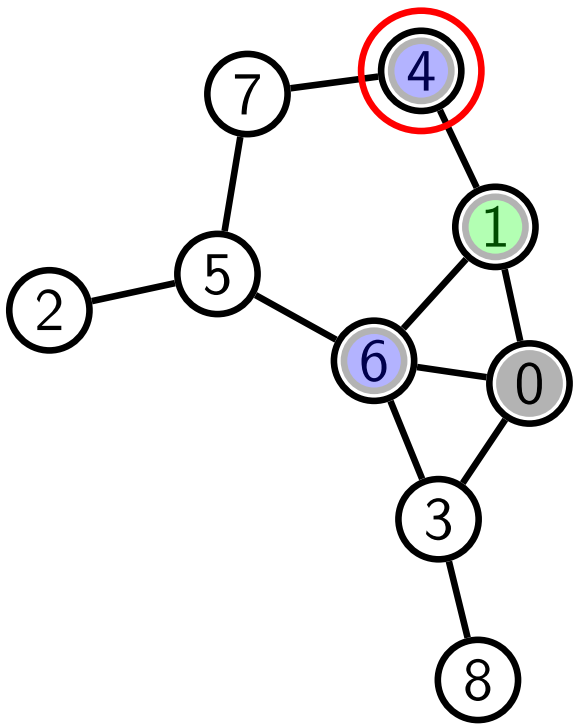
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



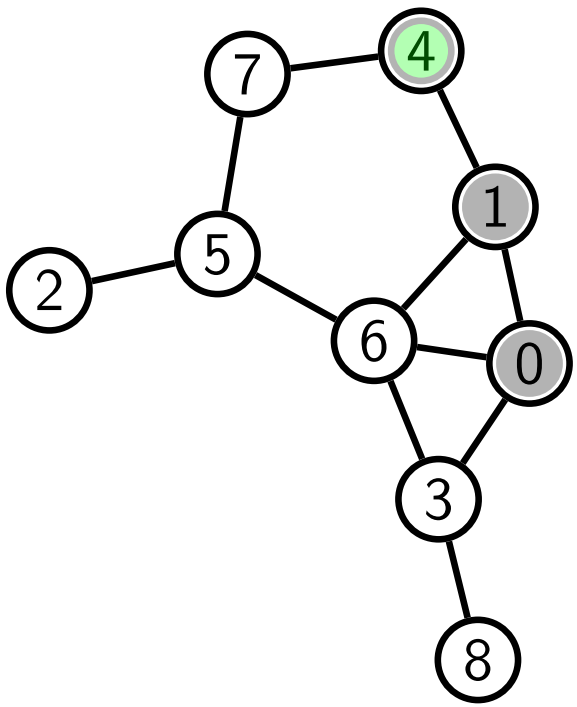
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



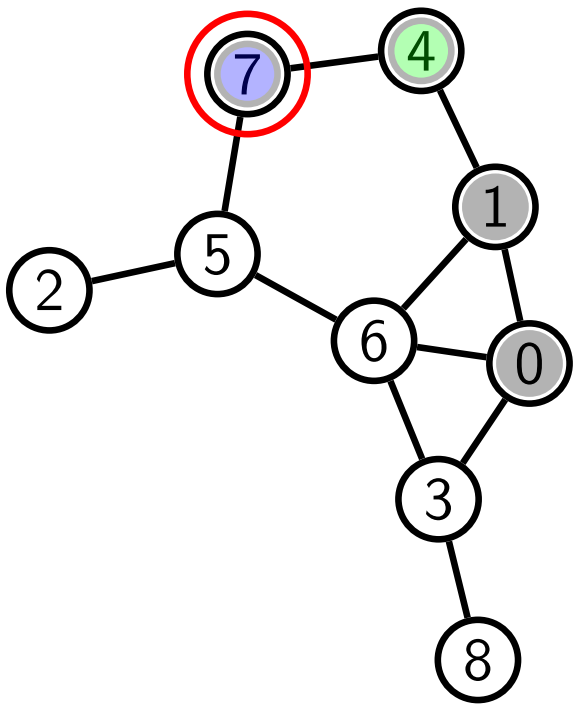
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



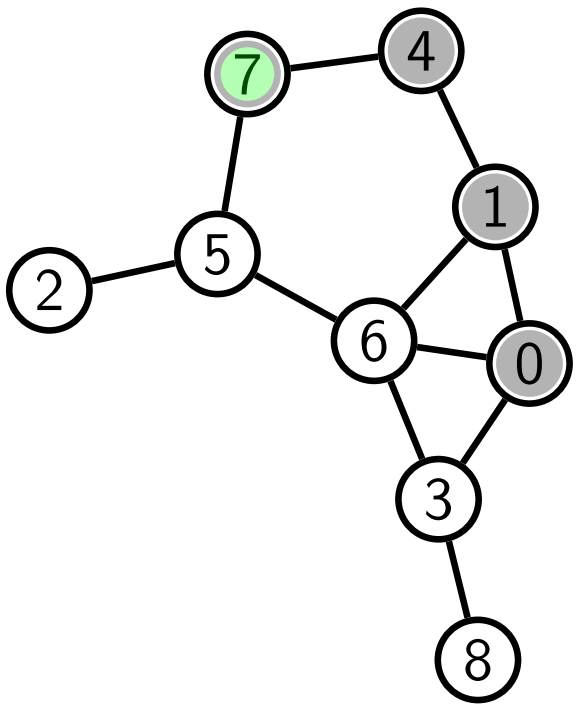
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



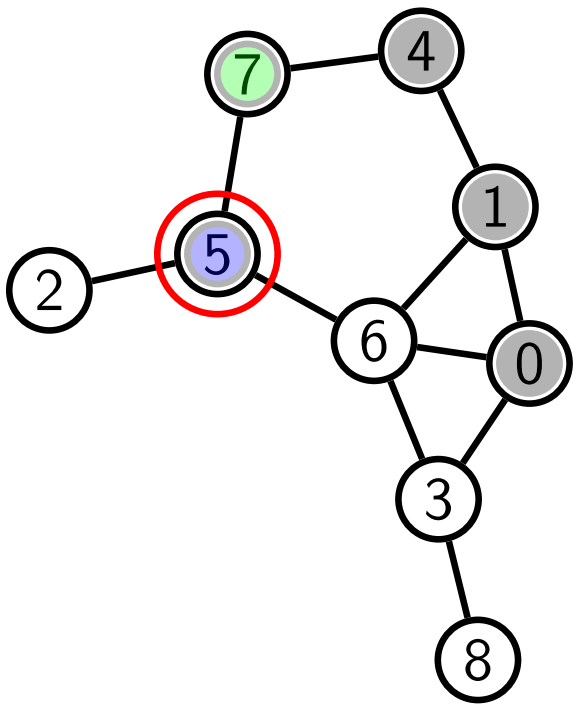
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



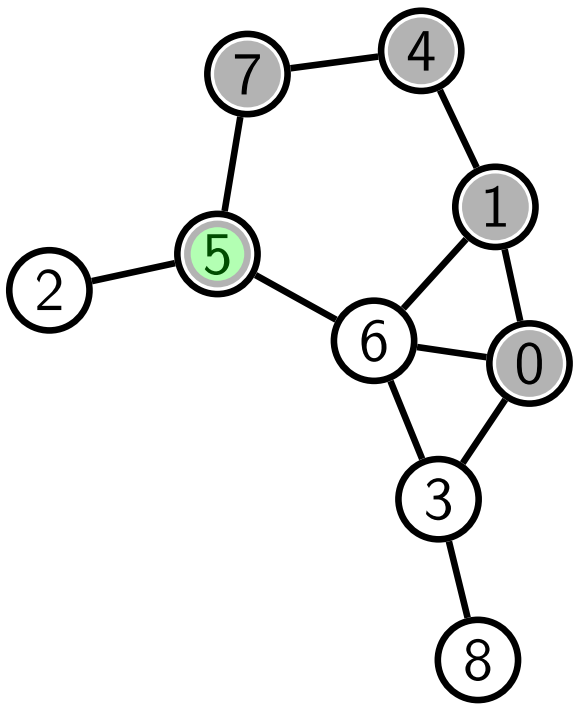
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



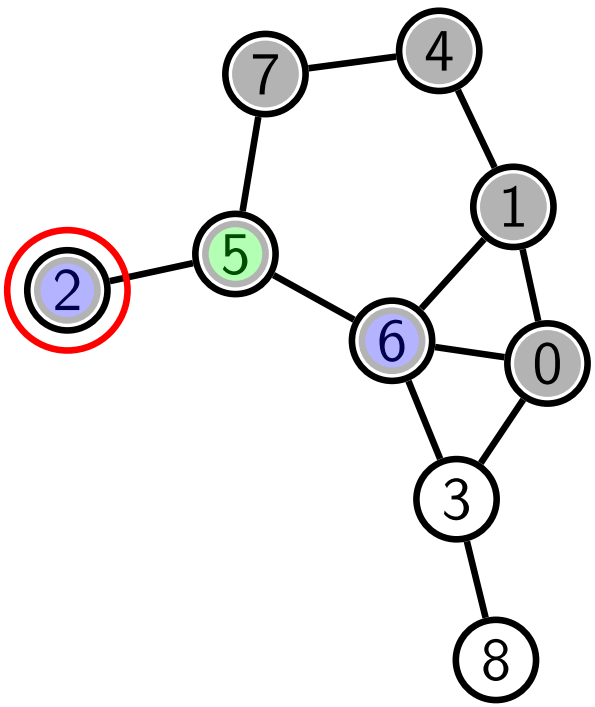
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



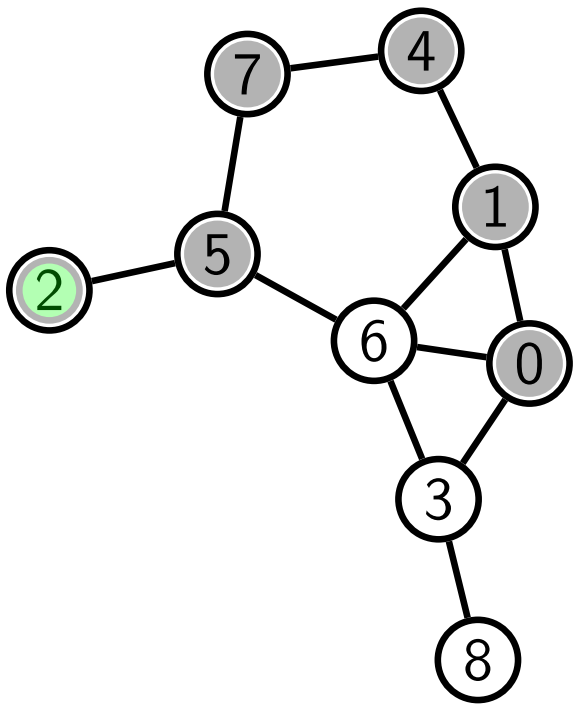
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



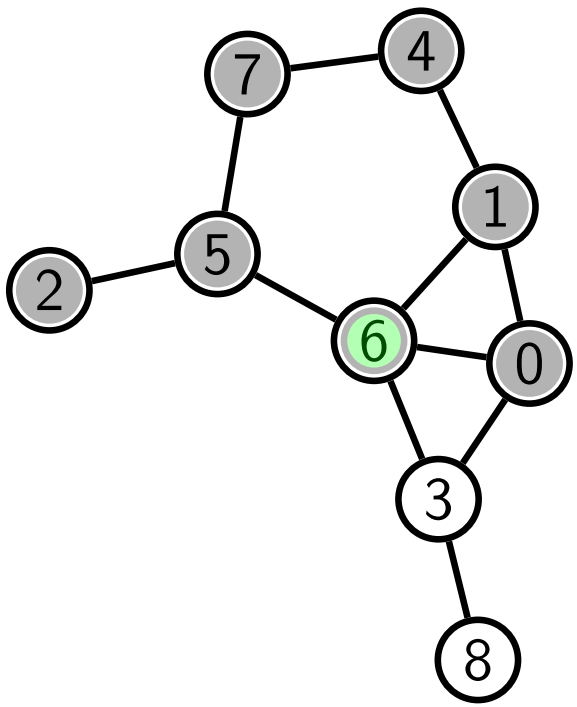
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



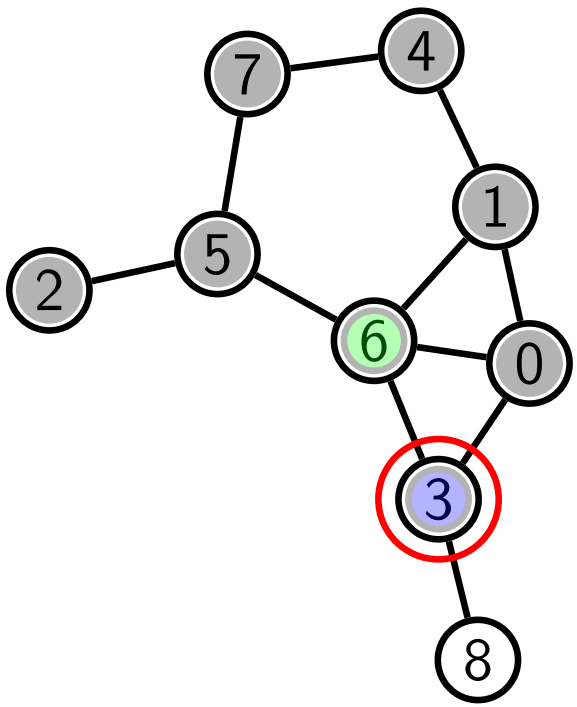
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



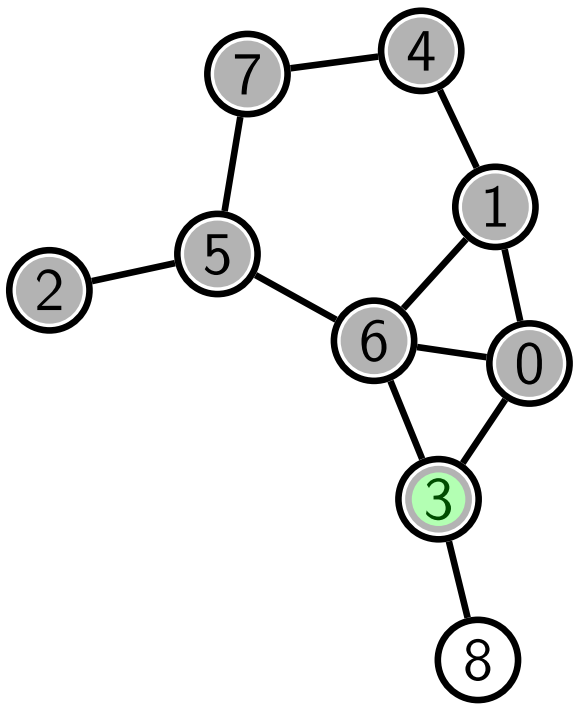
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



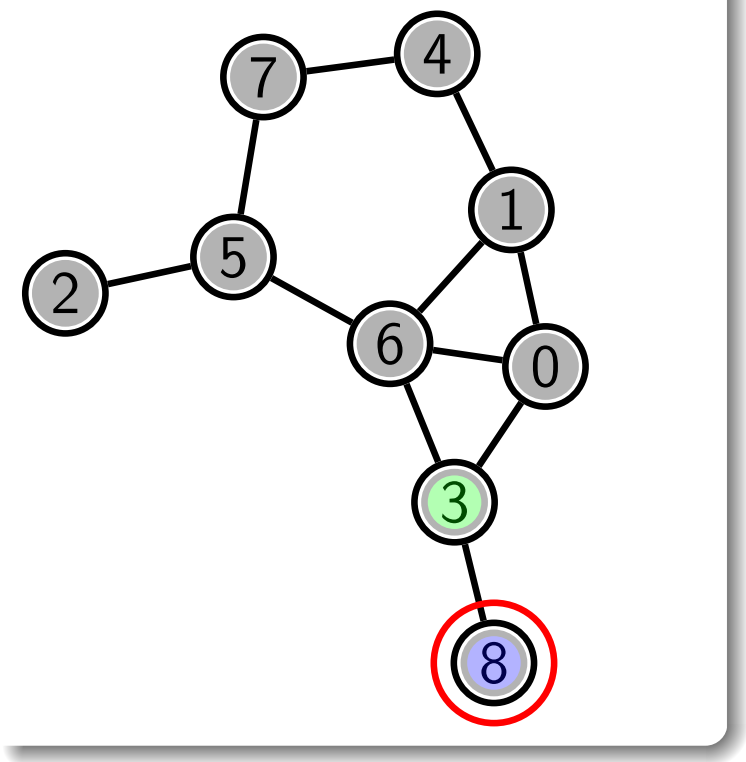
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



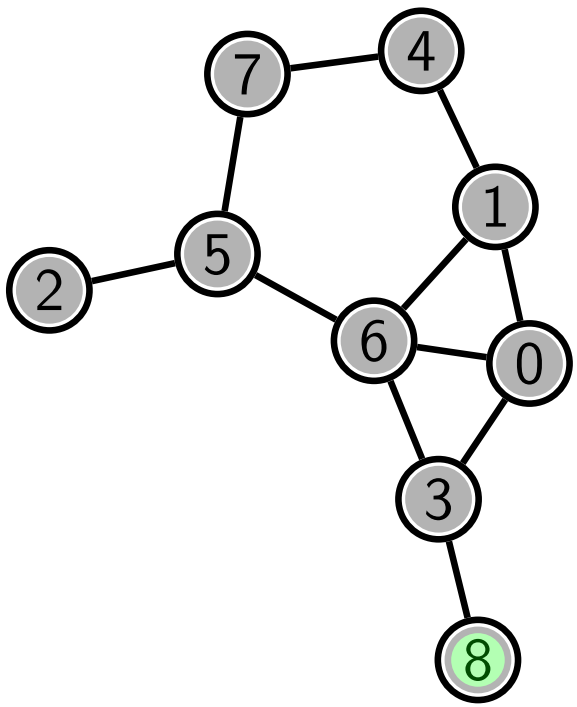
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



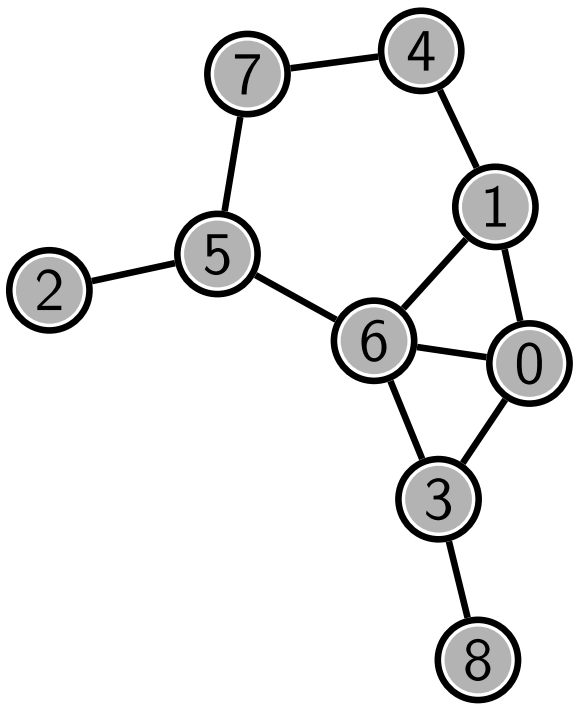
Parcours en profondeur : exemple

Exemple 2 (départ = 0)



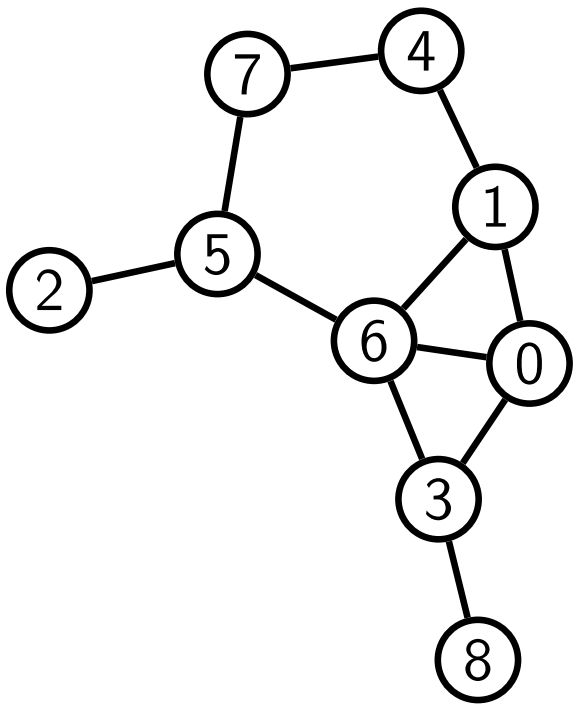
Parcours en profondeur : exemple

Exemple 2 (départ = 0)

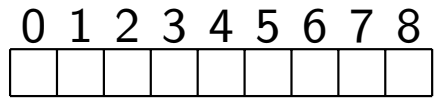


Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

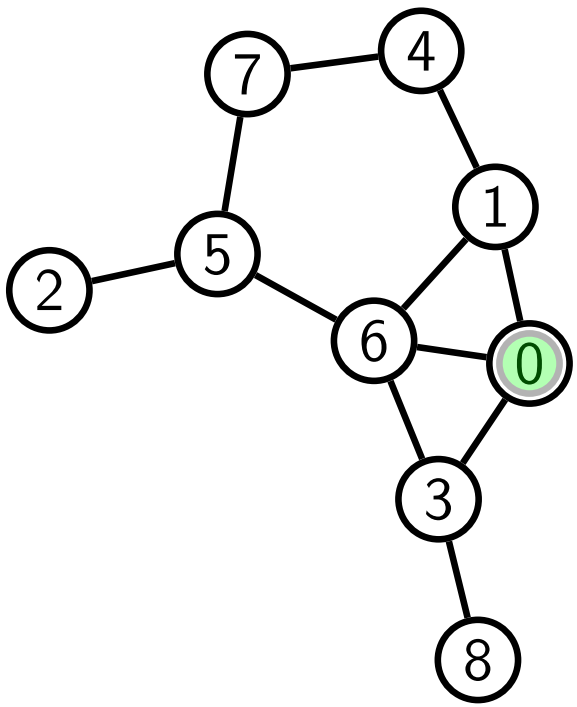
visités : 

à traiter (pile)

résultat :

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

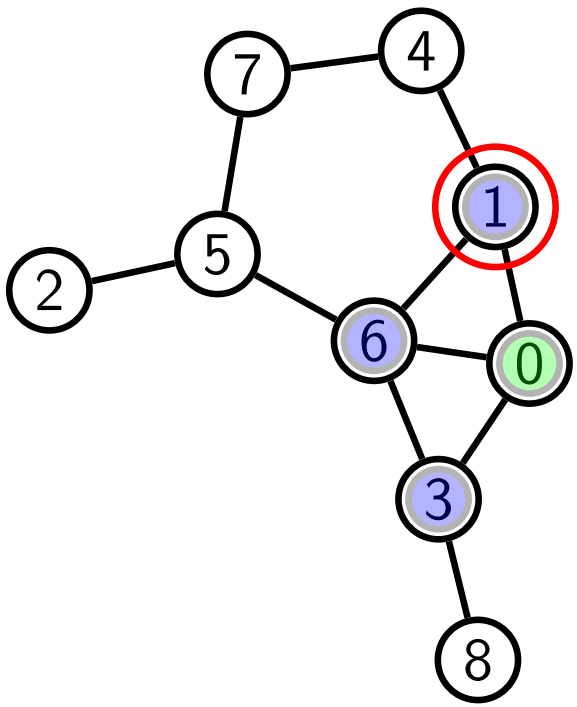
	0	1	2	3	4	5	6	7	8
visités :	✓								

à traiter (pile)

résultat : 0

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

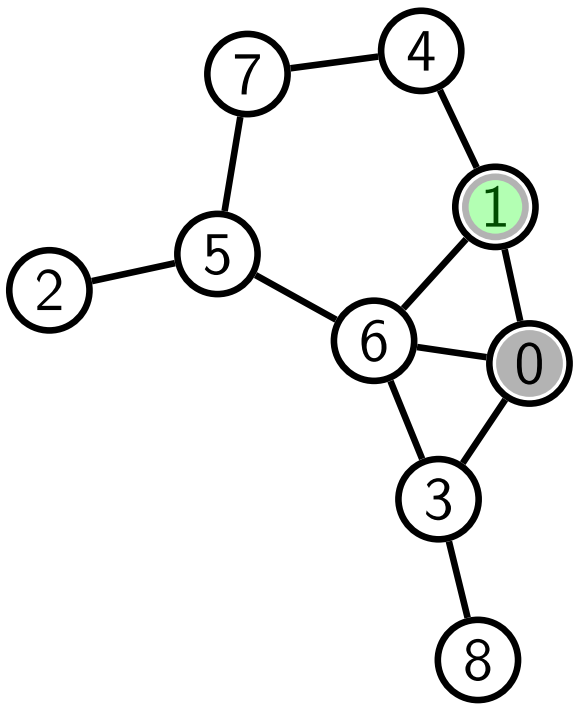
1
3
6

à traiter (pile)

résultat : 0

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

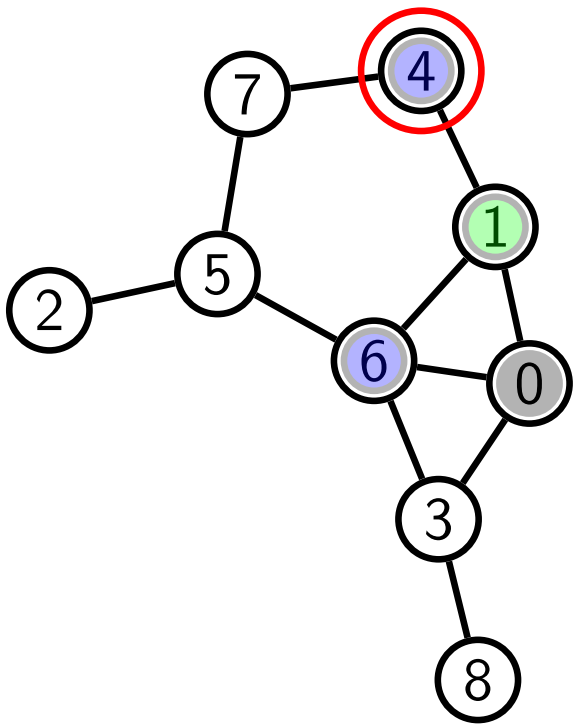
3
6

à traiter (pile)

résultat : 0 1

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

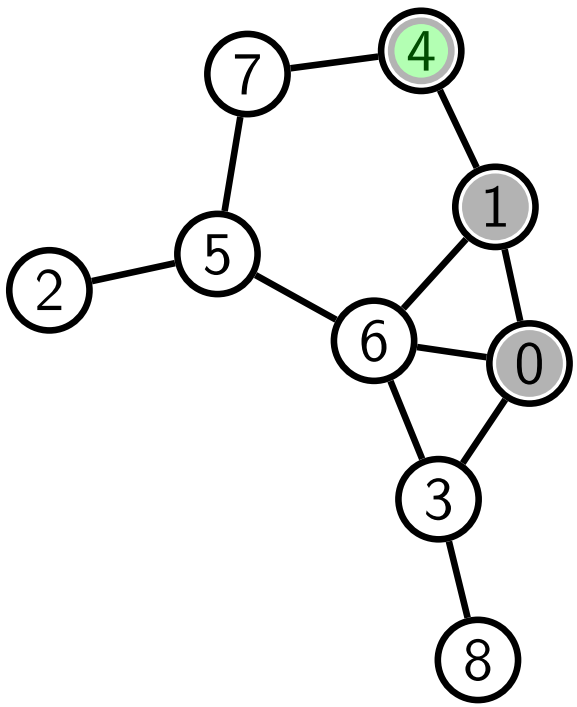
4
6
3
6

à traiter (pile)

résultat : 0 1

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

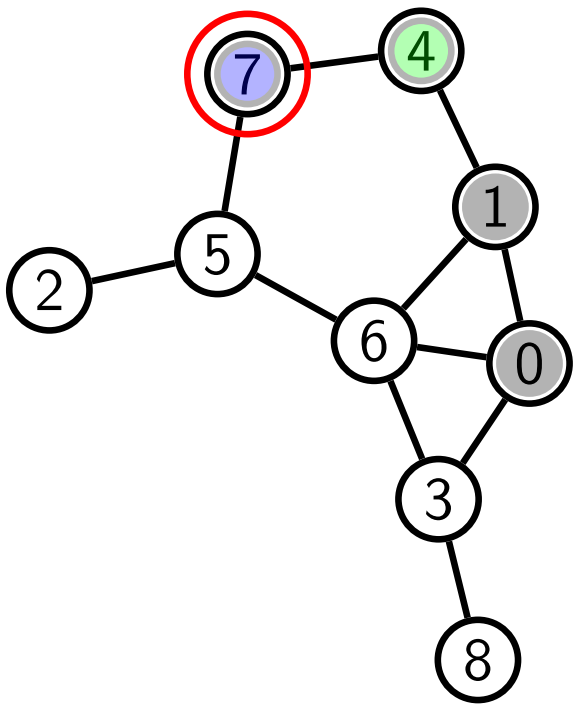
6
3
6

à traiter (pile)

résultat : 0 1 4

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

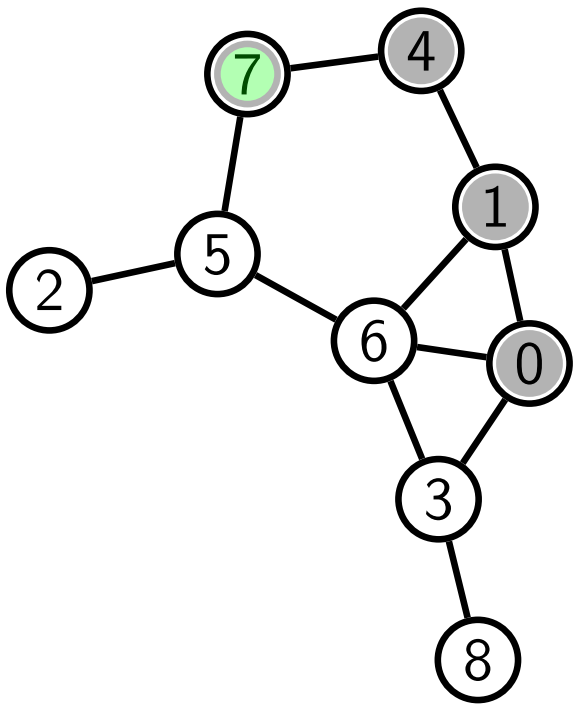
7
6
3
6

à traiter (pile)

résultat : 0 1 4

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

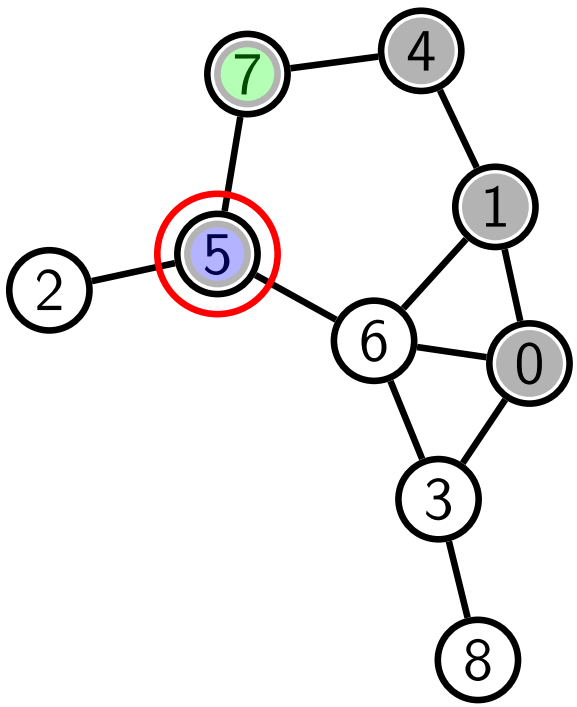
6
3
6

à traiter (pile)

résultat : 0 1 4 7

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

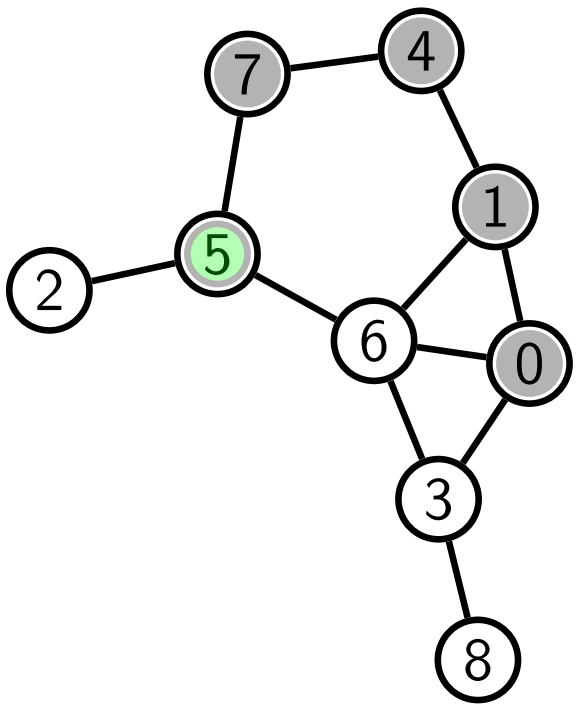
5
6
3
6

à traiter (pile)

résultat : 0 1 4 7

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

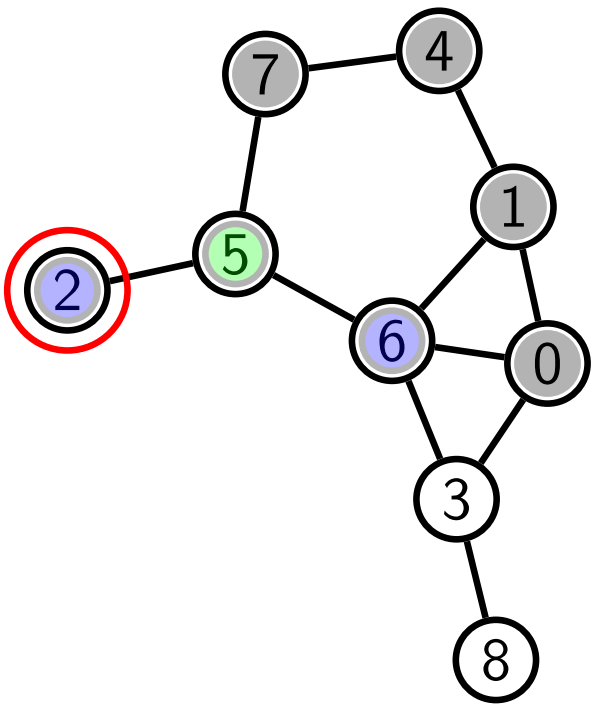
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

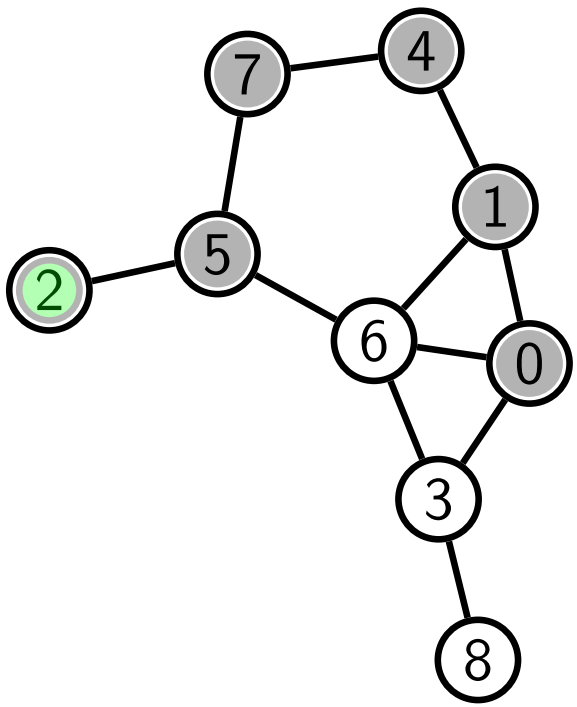
2
6
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

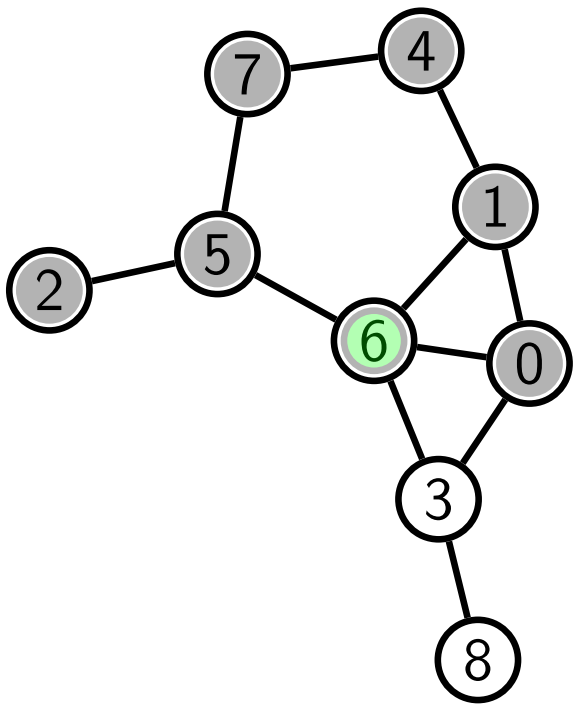
6
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

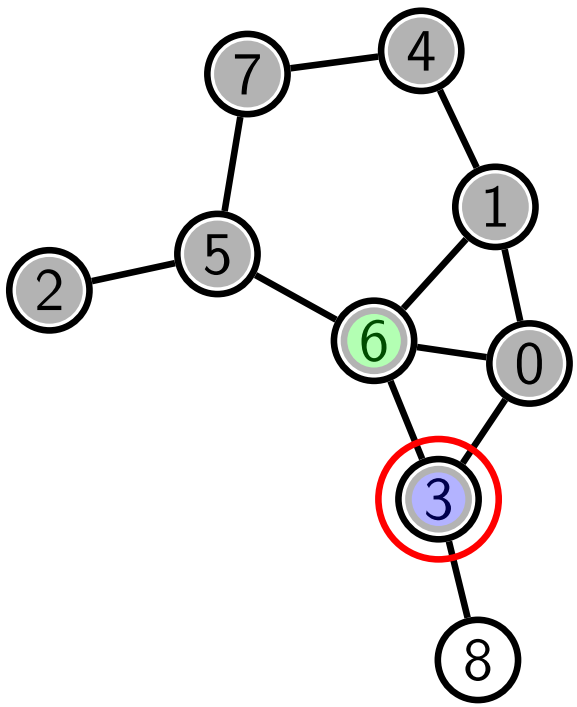
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

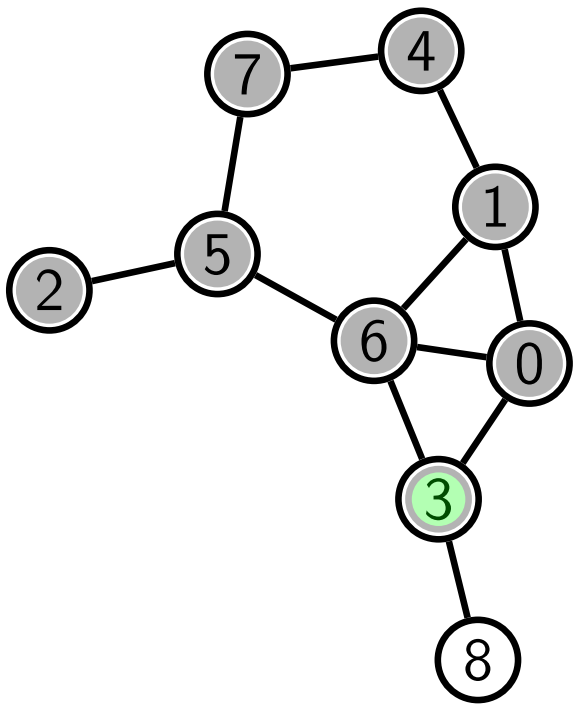
3
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

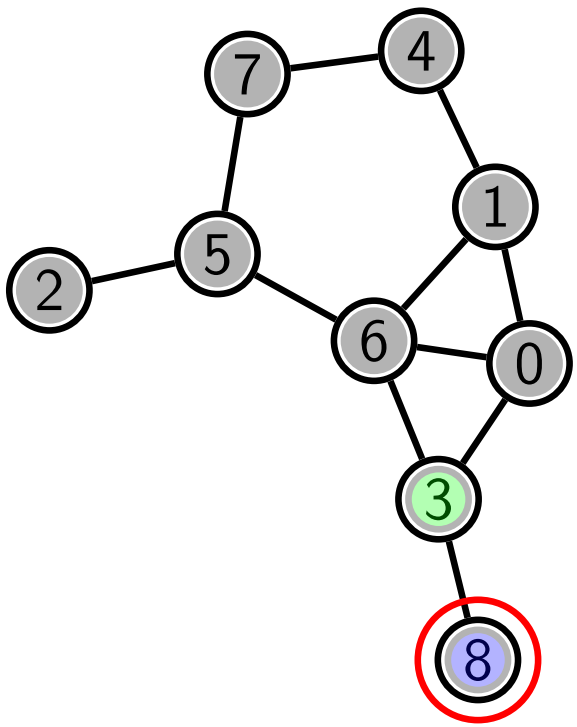
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

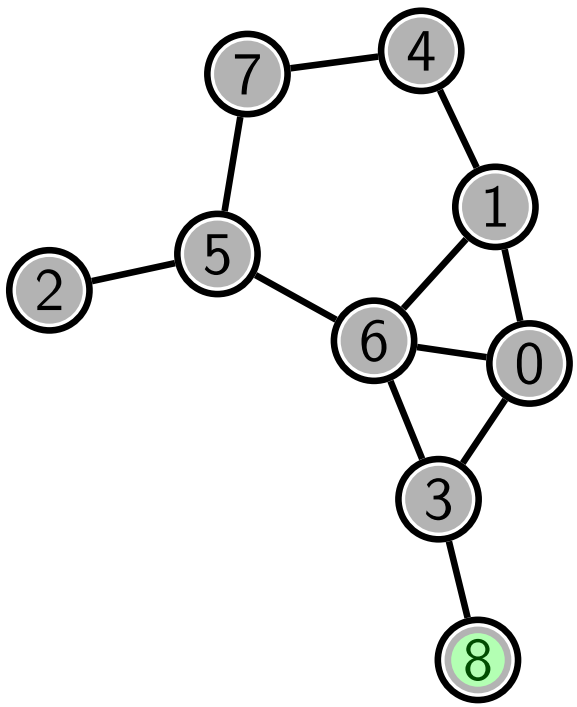
8
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

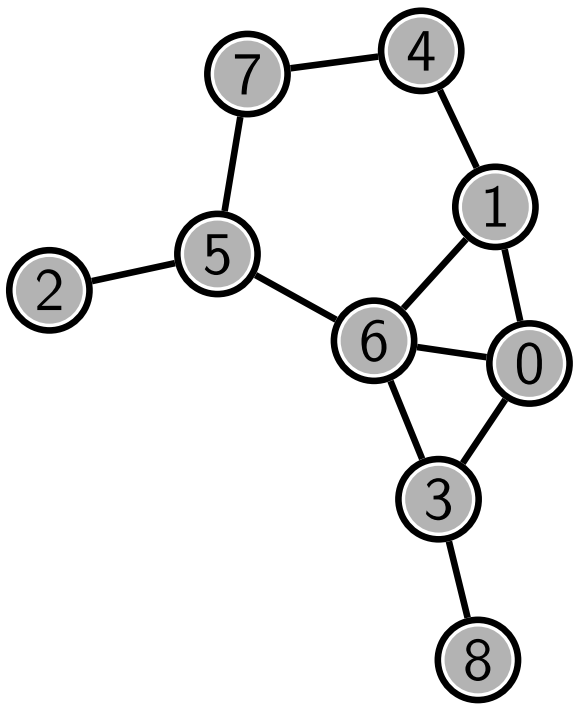
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3 8

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

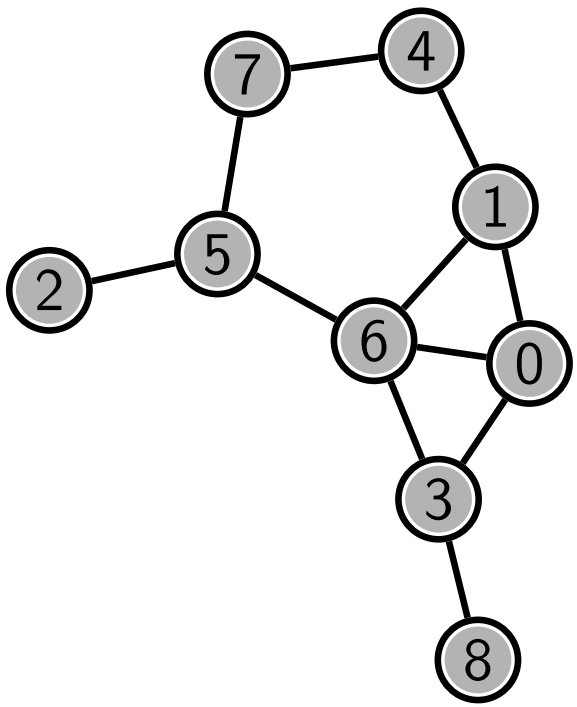
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3 8

Parcours en profondeur : exemple

Exemple 2 (départ = 0)



Les coulisses

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

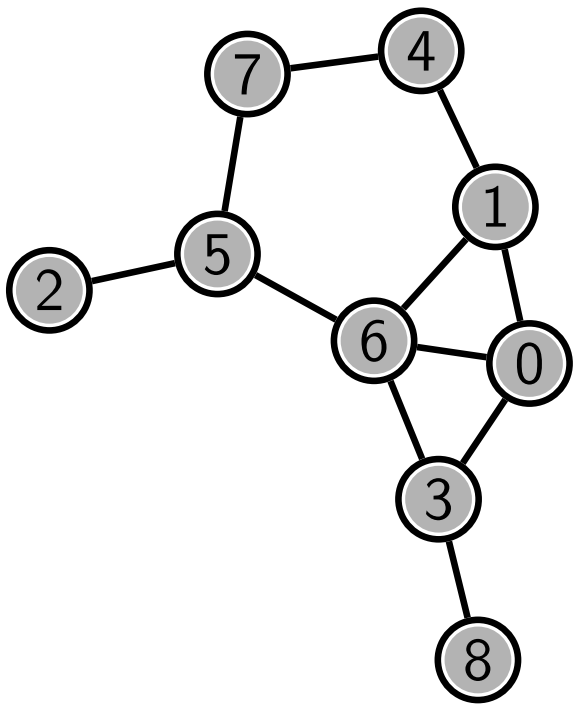
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3 8

Parcours en profondeur : exemple

Exemple 2 (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

Algorithme 3 : PROFONDEUR(G, départ, visités=NIL)

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

Sortie : les 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 voisin dans renverser(G.voisins(sommet)) faire
11 |   |   |   si ¬ visités[voisin] alors a_traiter.empiler(voisin) ;
12 renvoyer résultat;
```

Parcours de graphe en largeur

- Le principe du parcours en largeur se généralise comme suit aux graphes :

Parcours de graphe en largeur

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

Parcours de graphe en largeur

- Le principe du parcours en largeur se généralise comme suit aux graphes :
  1. si le sommet actuel u n'a pas encore été visité, l'afficher ;
  2. pour chaque voisin v non encore visité de u : placer v dans la file d'attente ;

Parcours de graphe en largeur

- Le principe du parcours en largeur se généralise comme suit aux graphes :
  1. si le sommet actuel u n'a pas encore été visité, l'afficher ;
  2. pour chaque voisin v non encore visité de u : placer v dans la file d'attente ;
  3. appliquer le même traitement aux éléments de la file jusqu'à ce qu'elle soit vide.

Parcours de graphe en largeur

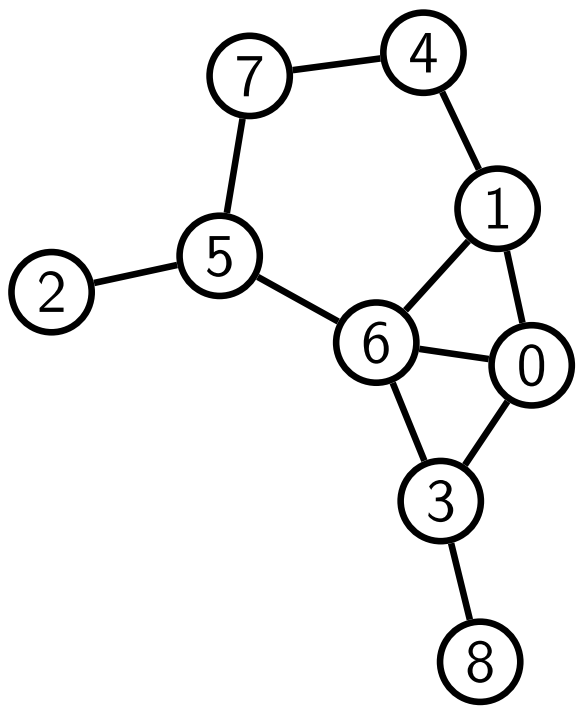
- Le principe du parcours en largeur se généralise comme suit aux graphes :
  1. si le sommet actuel u n'a pas encore été visité, l'afficher ;
  2. pour chaque voisin v non encore visité de u : placer v dans la file d'attente ;
  3. appliquer le même traitement aux éléments de la file jusqu'à ce qu'elle soit vide.
- Ce parcours partitionne les sommets de G en fonction de leur **distance** au sommet de départ ;

Parcours de graphe en largeur

- Le principe du parcours en largeur se généralise comme suit aux graphes :
  1. si le sommet actuel u n'a pas encore été visité, l'afficher ;
  2. pour chaque voisin v non encore visité de u : placer v dans la file d'attente ;
  3. 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 ;
- Examinons les étapes de ce parcours sur un exemple.

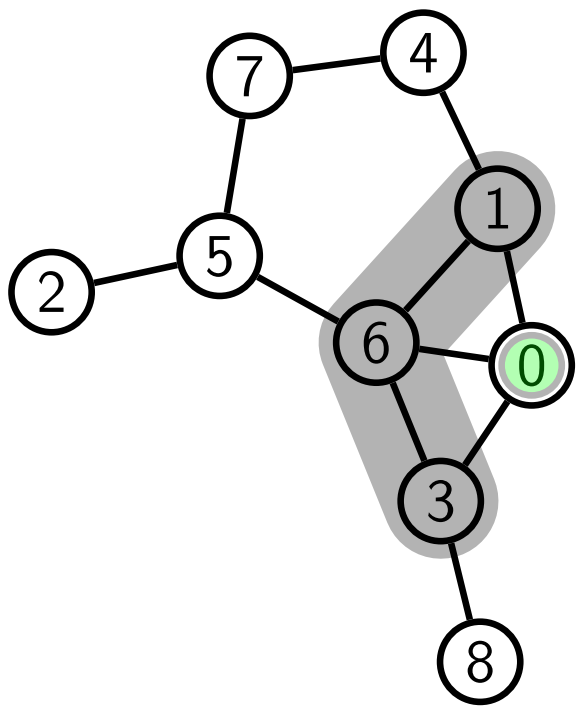
Parcours en largeur : exemple

Exemple 3 (départ = 0)



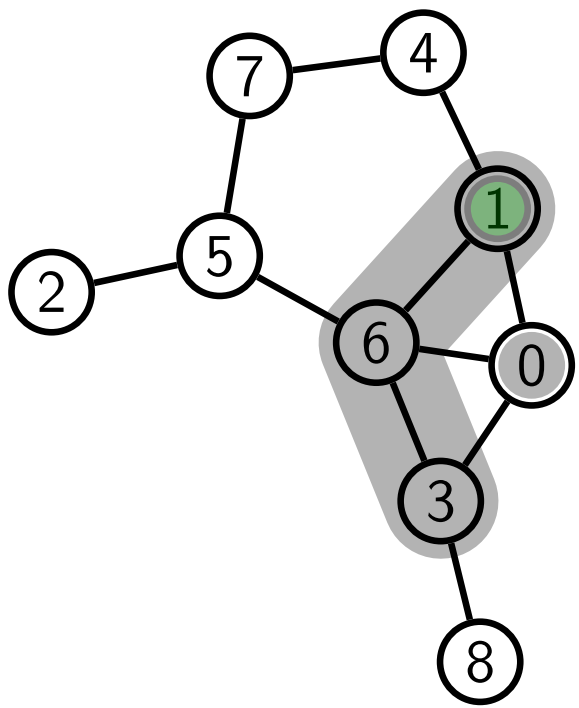
Parcours en largeur : exemple

Exemple 3 (départ = 0)



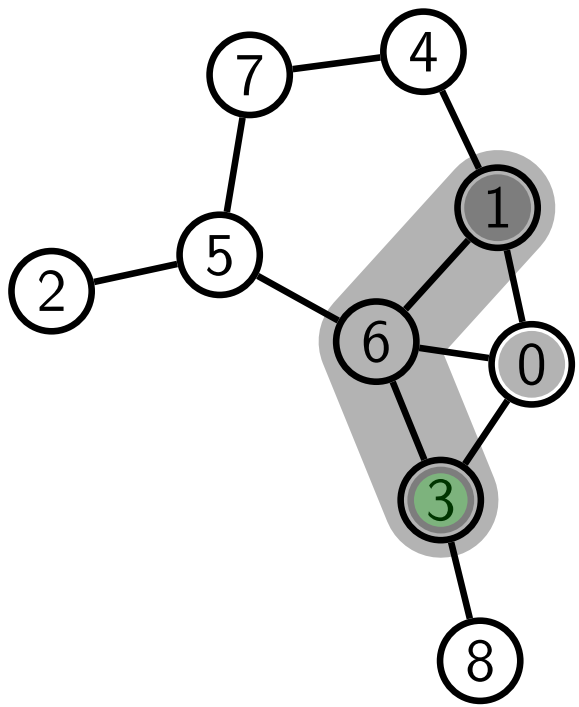
Parcours en largeur : exemple

Exemple 3 (départ = 0)



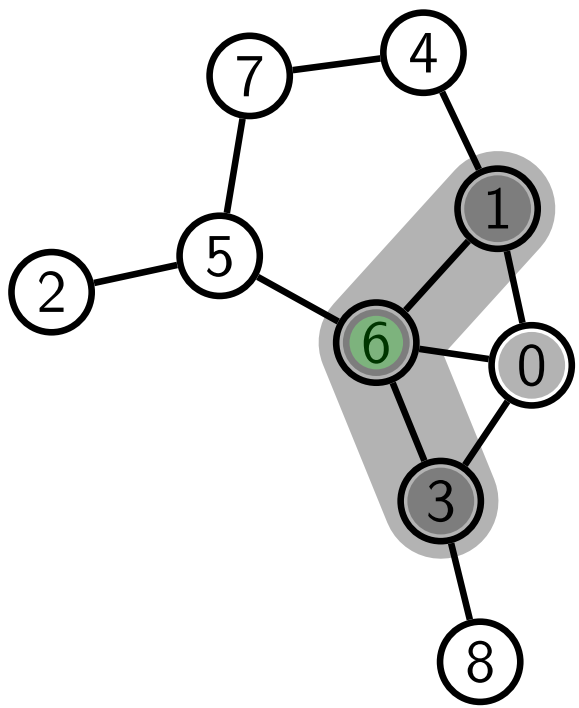
Parcours en largeur : exemple

Exemple 3 (départ = 0)



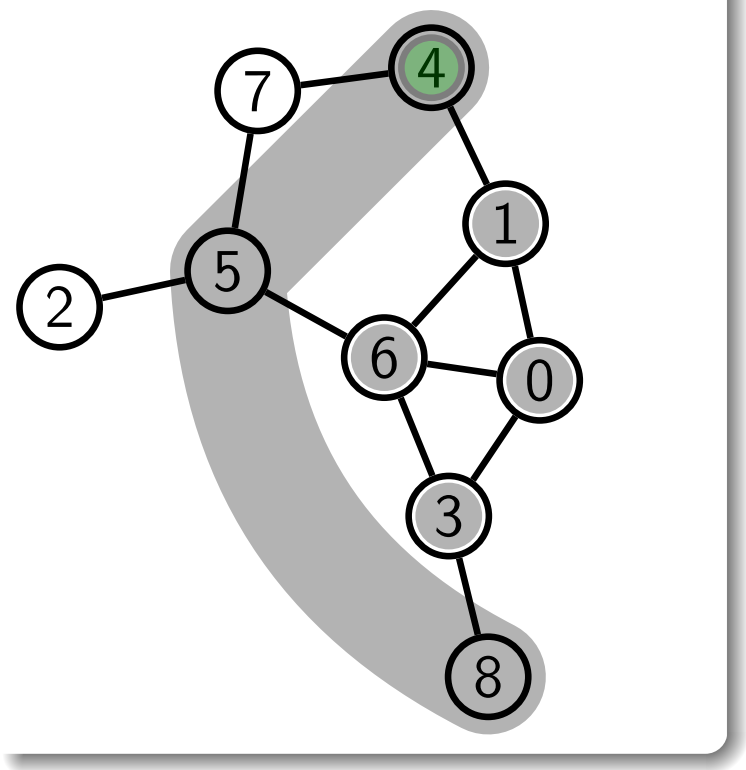
Parcours en largeur : exemple

Exemple 3 (départ = 0)



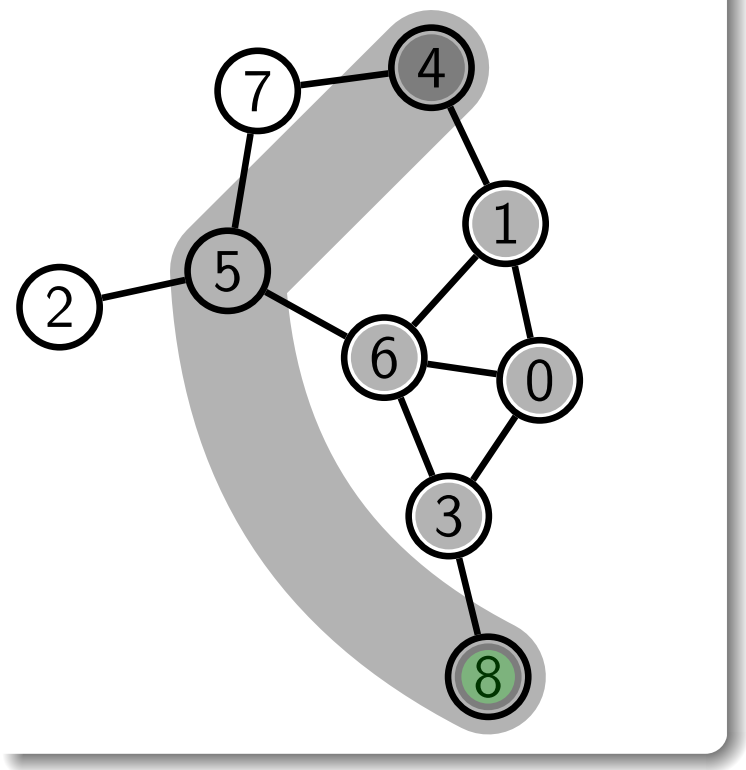
Parcours en largeur : exemple

Exemple 3 (départ = 0)



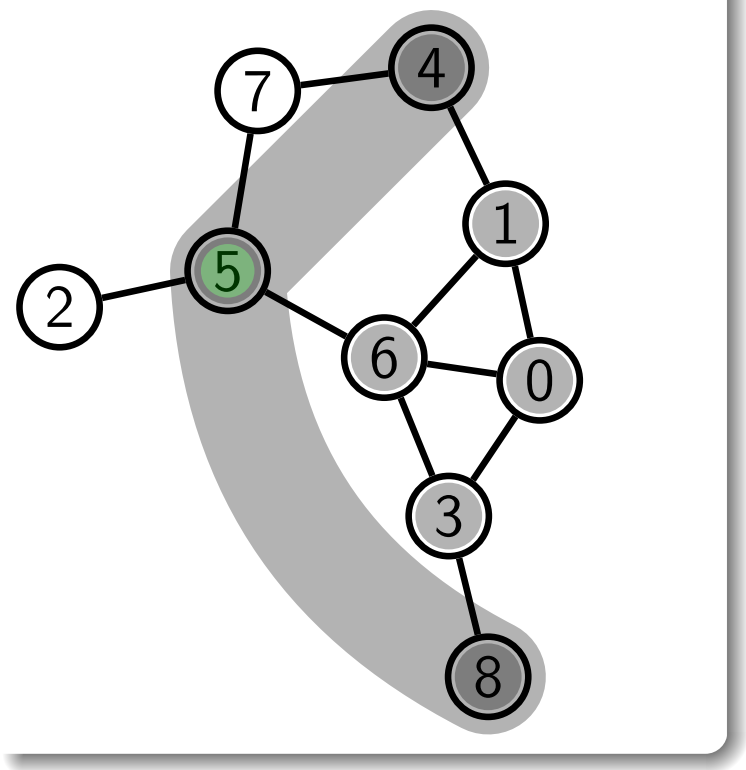
Parcours en largeur : exemple

Exemple 3 (départ = 0)



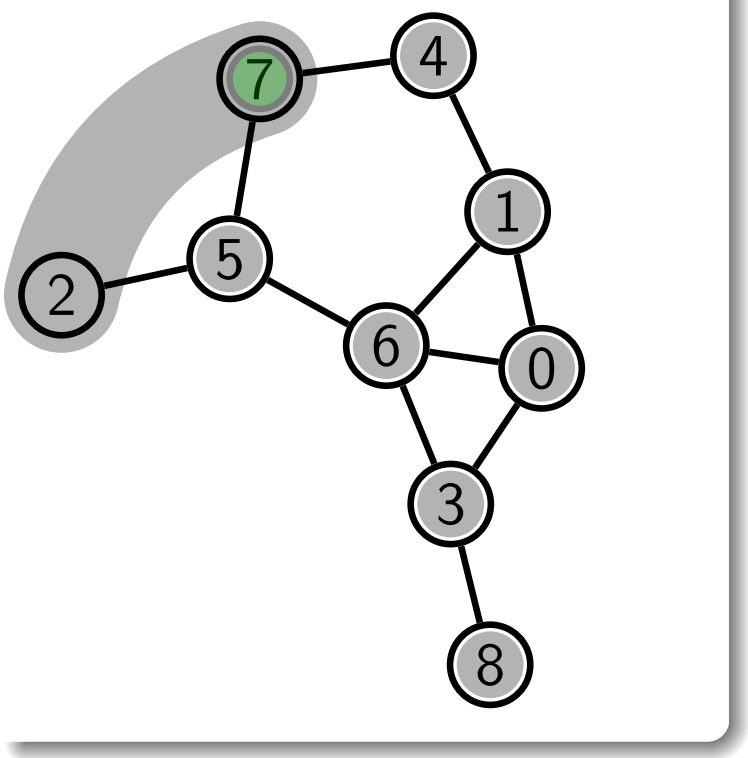
Parcours en largeur : exemple

Exemple 3 (départ = 0)



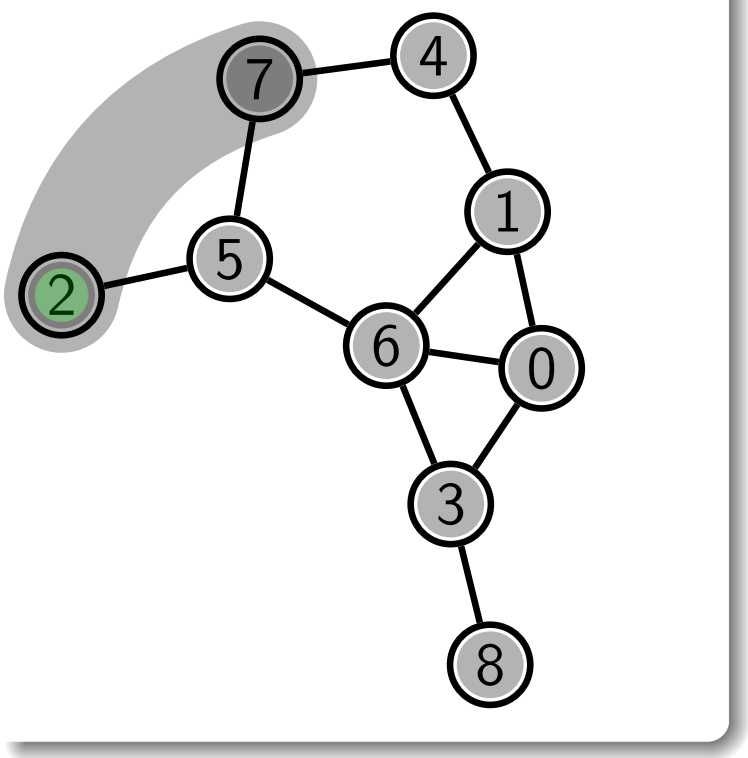
Parcours en largeur : exemple

Exemple 3 (départ = 0)



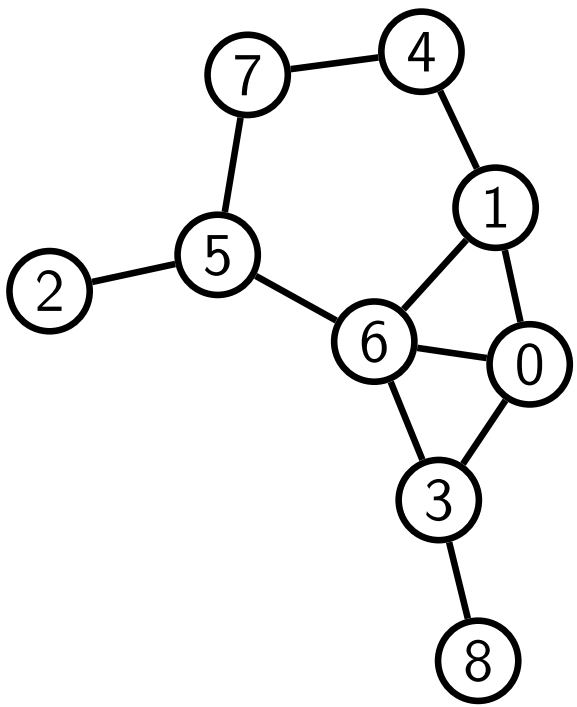
Parcours en largeur : exemple

Exemple 3 (départ = 0)

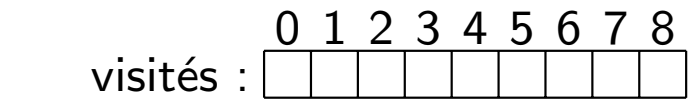


Parcours en largeur : exemple

Exemple 3 (départ = 0)



Les coulisses



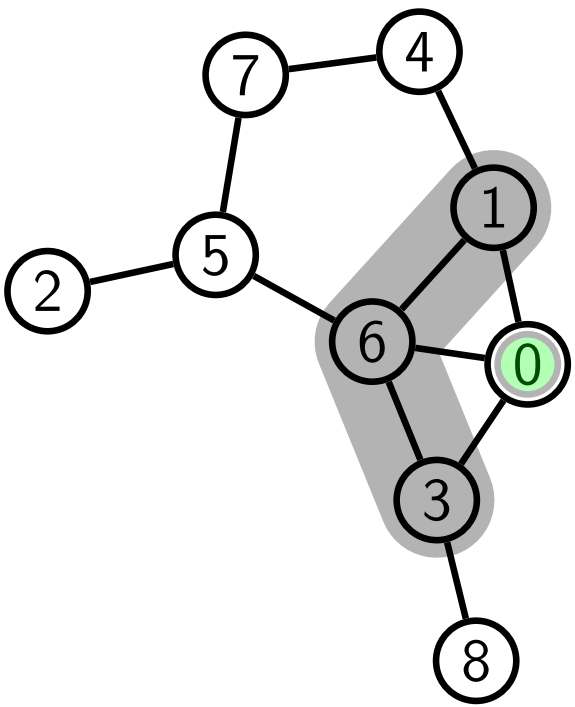
à traiter (file) : 0

résultat :

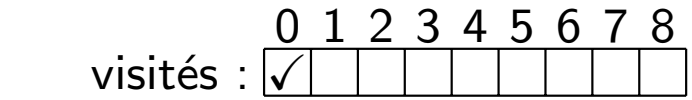
(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 3 (départ = 0)



Les coulisses



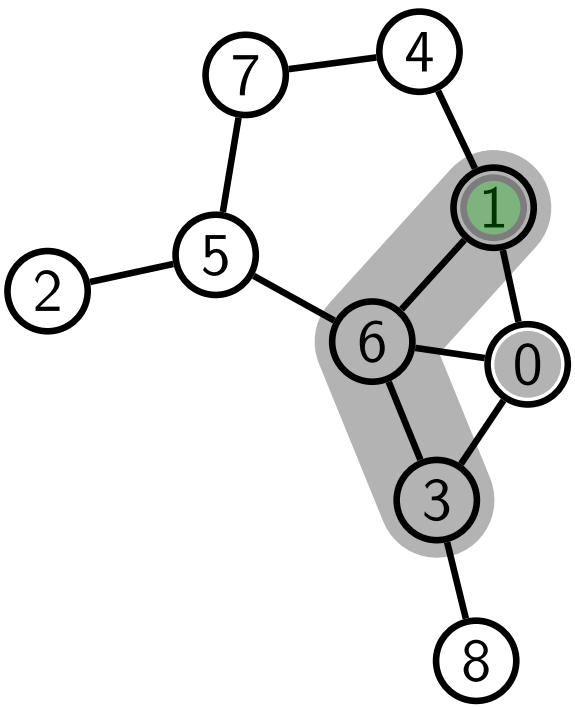
à traiter (file) : 6 | 3 | 1

résultat : 0

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 3 (départ = 0)



Les coulisses

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

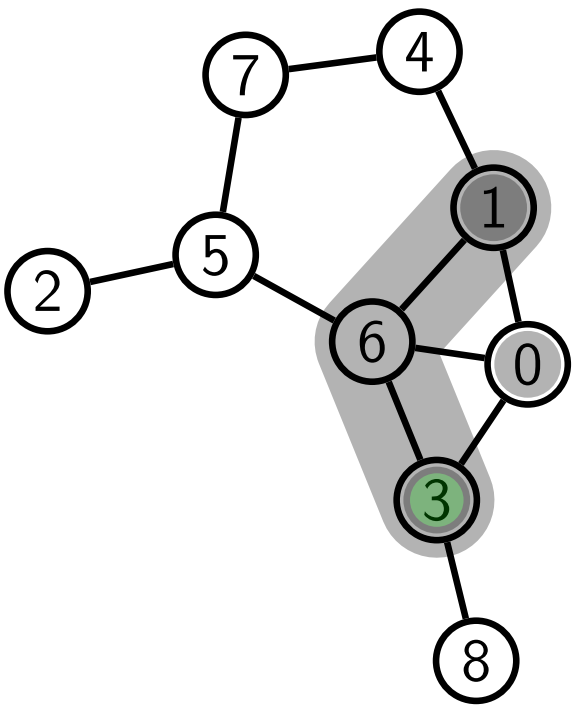
à traiter (file) : | 6 | 4 | 6 | 3 |

résultat : 0 1

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 3 (départ = 0)



Les coulisses

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

à traiter (file) :

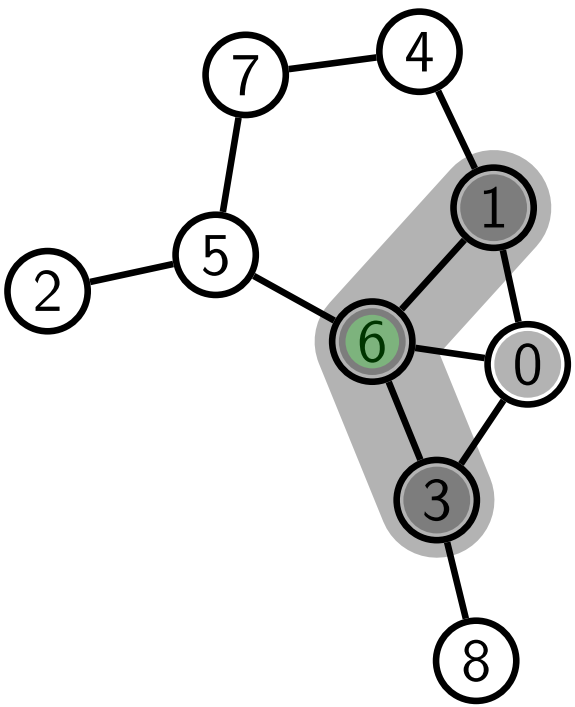
8	6	6	4	6

résultat : 0 1 3

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 3 (départ = 0)



Les coulisses

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

à traiter (file) :

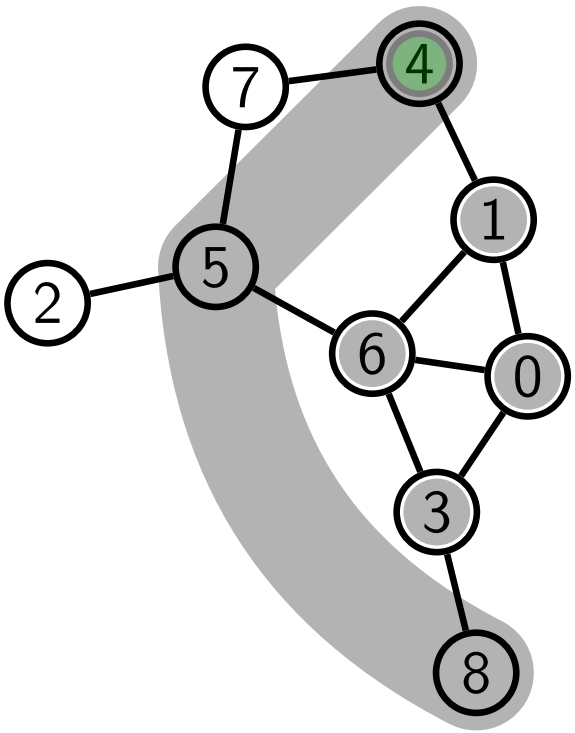
5	8	6	6	4

résultat : 0 1 3 6

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 3 (départ = 0)



Les coulisses

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

à traiter (file) :

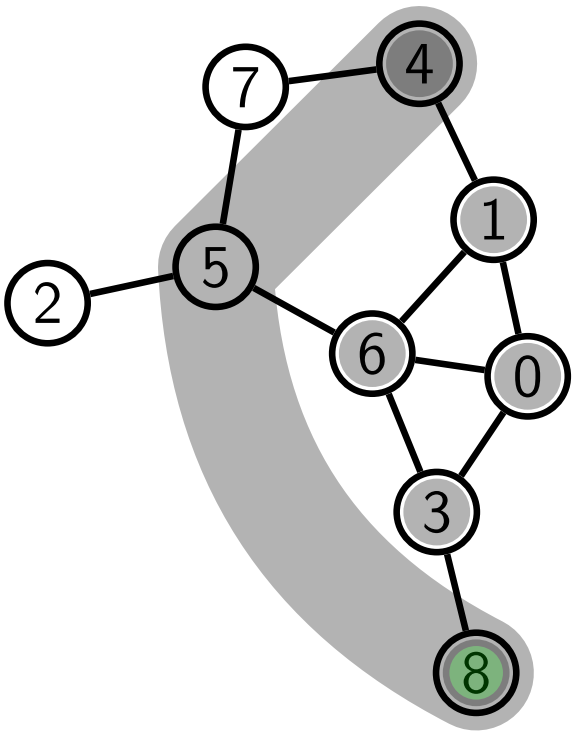
7	5	8	6	6

résultat : 0 1 3 6 4

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 3 (départ = 0)



Les coulisses

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

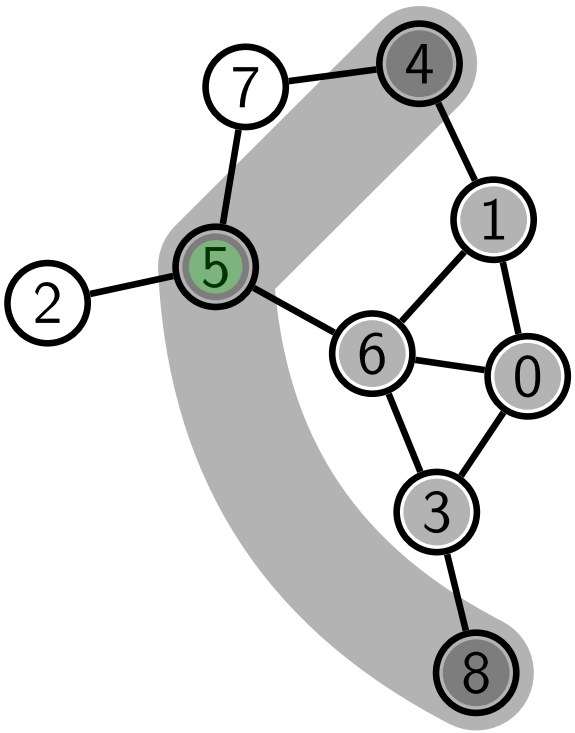
à traiter (file) : | 7 | 5 |

résultat : 0 1 3 6 4 8

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 3 (départ = 0)



Les coulisses

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

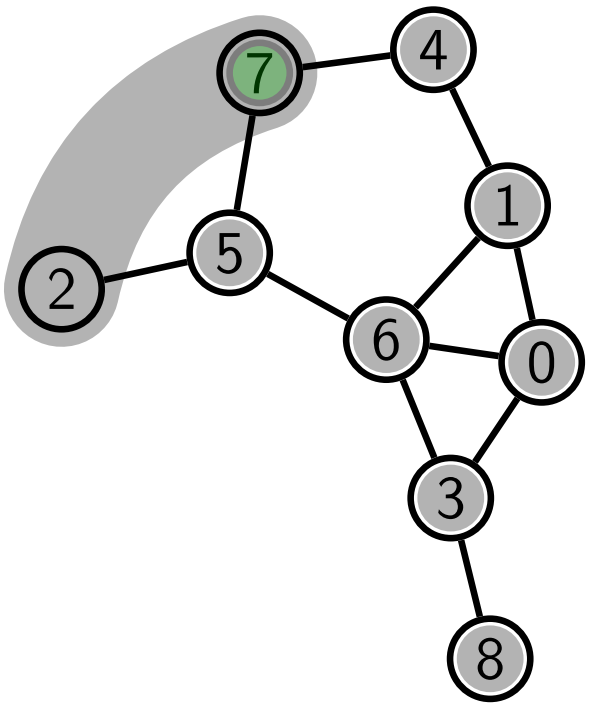
à traiter (file) : | 7 | 2 | 7 |

résultat : 0 1 3 6 4 8 5

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 3 (départ = 0)



Les coulisses

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

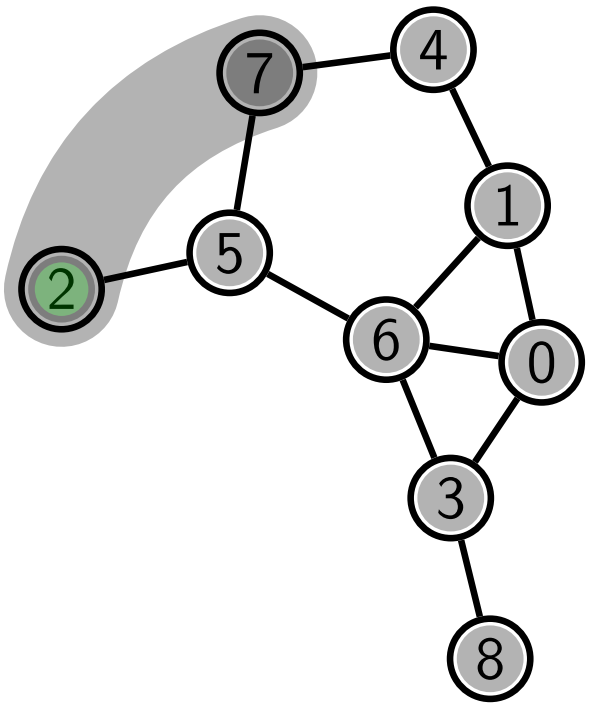
à traiter (file) : | 7 | 2 |

résultat : 0 1 3 6 4 8 5 7

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 3 (départ = 0)



Les coulisses

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

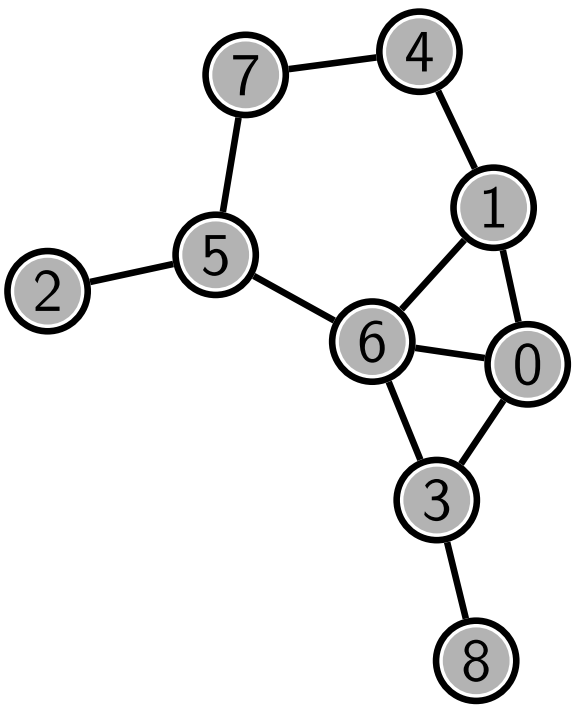
à traiter (file) : [7]

résultat : 0 1 3 6 4 8 5 7 2

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 3 (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 4 : LARGEUR(G, départ, visités=NIL)

Entrées : un graphe non-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 voisin dans G.voisins(sommet) faire
11 |   |   |   si ¬ visités[voisin] alors a_traiter.enfiler(voisin) ;
12 renvoyer résultat;
```

Comparaison des algorithmes

Comparons les deux algorithmes de parcours :

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 voisin dans
   |    |      renverser(G.voisins(sommet)) faire
11 |    |    |    si ¬ visités[voisin] alors
   |    |    |      a_traiter.empiler(voisin) ;
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 voisin dans
   |    |      G.voisins(sommet) faire
11 |    |    |    si ¬ visités[voisin] alors
   |    |    |      a_traiter.enfiler(voisin) ;
12 renvoyer résultat;
```

Seul le type de la structure de données a_traiter change !

Correction des algorithmes de parcours

Nos algorithmes de parcours explorent-ils bien tout le graphe ?

Correction des algorithmes de parcours

Nos algorithmes de parcours explorent-ils bien tout le graphe ? Oui, si le graphe est "en un seul morceau".

Correction des algorithmes de parcours

Nos algorithmes de parcours explorent-ils bien tout le graphe ? Oui, si le graphe est "en un seul morceau". Plus formellement :

Définition 1

Un graphe $G = (V, E)$ est **connexe** si pour toute paire de sommets $u, v \in V$, il existe un chemin dans G dont les extrémités sont u et v.

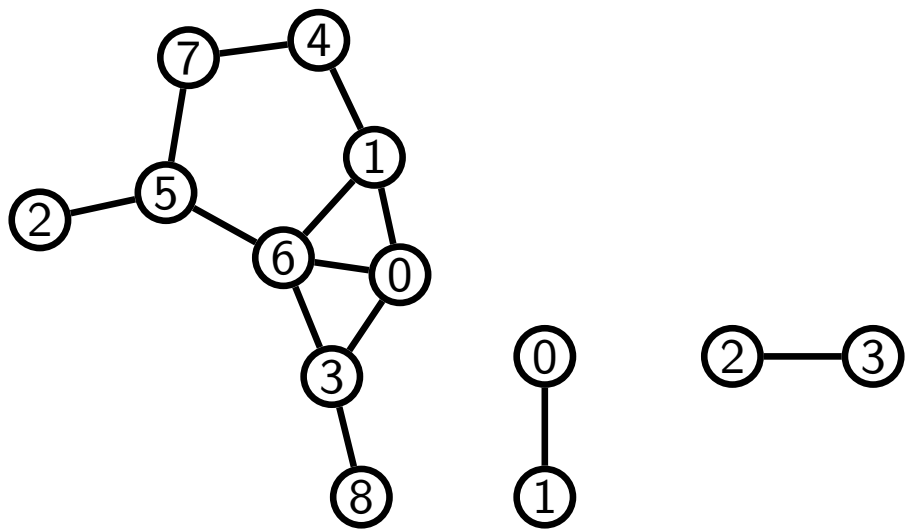
Correction des algorithmes de parcours

Nos algorithmes de parcours explorent-ils bien tout le graphe ? Oui, si le graphe est "en un seul morceau". Plus formellement :

Définition 1

Un graphe $G = (V, E)$ est **connexe** si pour toute paire de sommets $u, v \in V$, il existe un chemin dans G dont les extrémités sont u et v.

Exemple 4



graphe connexe graphe non connexe

Complexité des algorithmes de parcours

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

Complexité des algorithmes de parcours

- 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 des algorithmes de parcours

- 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 voisins ;

Complexité des algorithmes de parcours

- 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 voisins ;
  - matrice d'adjacence : G.voisins(v) est en $O(|V|)$;

Complexité des algorithmes de parcours

- 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 voisins ;
  - matrice d'adjacence : G.voisins(v) est en $O(|V|)$;
  - listes d'adjacence : G.voisins(v) est en $O(\deg(v))$;

Complexité des algorithmes de parcours

- 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 voisins ;
  - matrice d'adjacence : G.voisins(v) est en $O(|V|)$;
  - listes d'adjacence : G.voisins(v) est en $O(\deg(v))$;
- Nos parcours ont donc une complexité de :

Complexité des algorithmes de parcours

- 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 voisins ;
  - matrice d'adjacence : G.voisins(v) est en $O(|V|)$;
  - listes d'adjacence : G.voisins(v) est en $O(\deg(v))$;
- Nos parcours ont donc une complexité de :
  - $O(|V|^2)$ pour une matrice d'adjacence ;

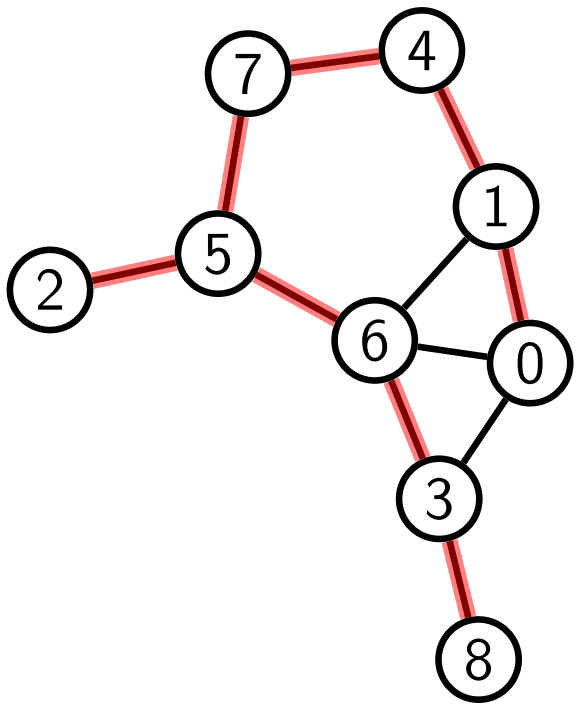
Complexité des algorithmes de parcours

- 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 voisins ;
  - matrice d'adjacence : G.voisins(v) est en $O(|V|)$;
  - listes d'adjacence : G.voisins(v) est en $O(\deg(v))$;
- Nos parcours ont donc une complexité de :
  - $O(|V|^2)$ pour une matrice d'adjacence ;
  - $O(|V| + |E|)$ pour des listes d'adjacence (rappel : $\sum_{v \in V} \deg(v) = 2|E|$) ;

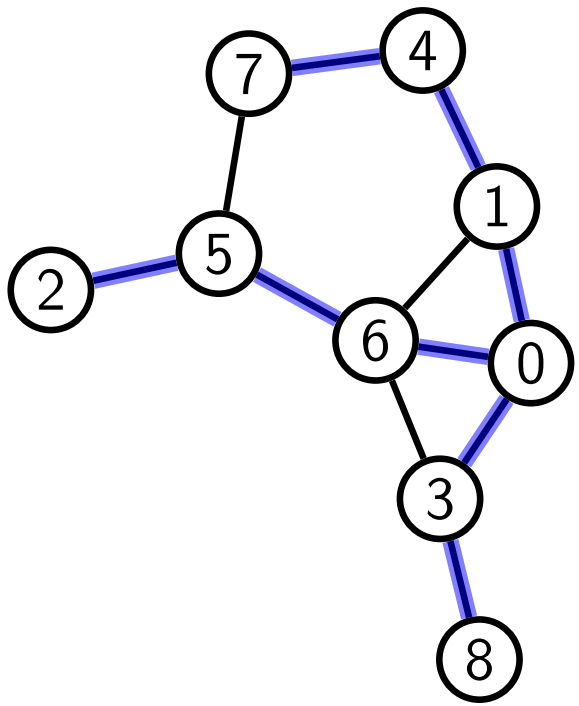
Notion d'arbre de parcours

On associe aux parcours en largeur et en profondeur des **arbres de parcours**, qui retracent l'ordre dans lequel les sommets ont été découverts.

Exemple 5 (arbres de parcours au départ de 0)



profondeur

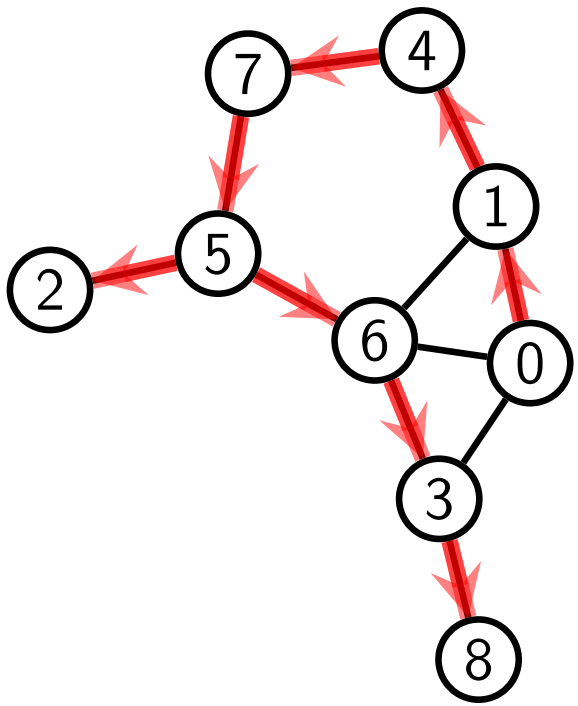


largeur

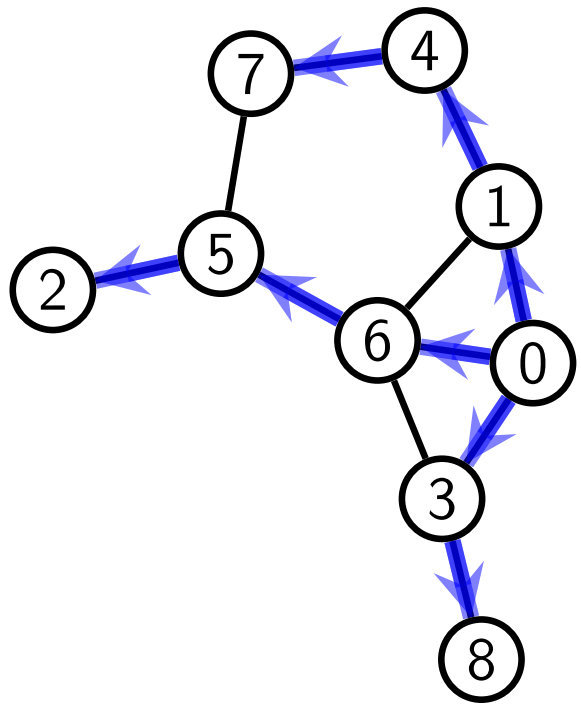
Notion d'arbre de parcours

On associe aux parcours en largeur et en profondeur des **arbres de parcours**, qui retracent l'ordre dans lequel les sommets ont été découverts.

Exemple 5 (arbres de parcours au départ de 0)



profondeur

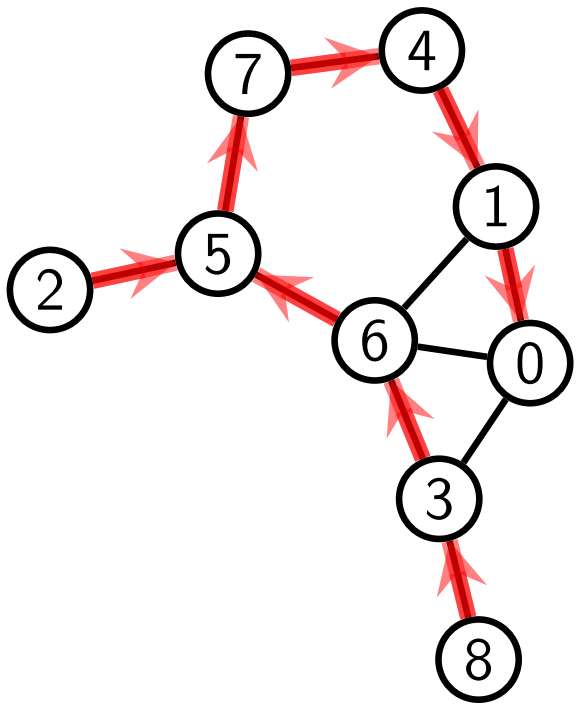


largeur

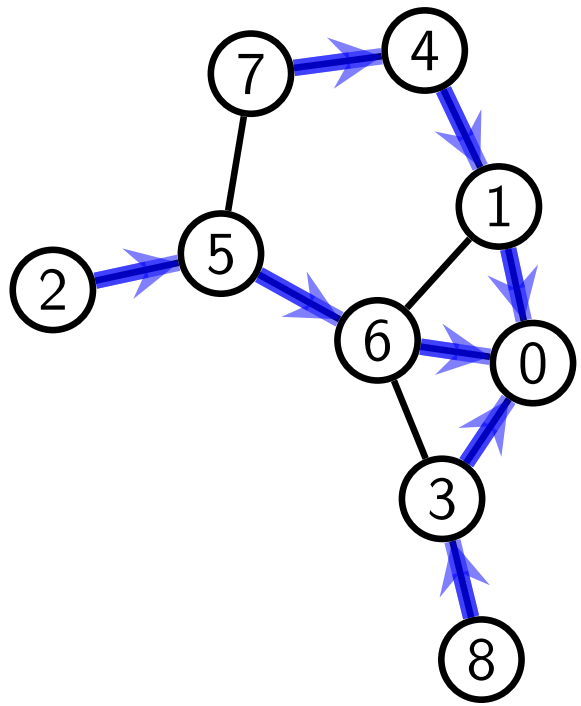
Notion d'arbre de parcours

On associe aux parcours en largeur et en profondeur des **arbres de parcours**, qui retracent l'ordre dans lequel les sommets ont été découverts.

Exemple 5 (arbres de parcours au départ de 0)



profondeur



largeur

Calcul des arbres de parcours

- Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Calcul des arbres de parcours

- Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.
- On dira qu'un sommet est **orphelin** s'il n'a pas de parent, et **adopté** sinon ;

Calcul des arbres de parcours

- Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.
- On dira qu'un sommet est **orphelin** s'il n'a pas de parent, et **adopté** sinon ;
- Les deux parcours traitent les sommets différemment :

Calcul des arbres de parcours

- Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.
- On dira qu'un sommet est **orphelin** s'il n'a pas de parent, et **adopté** sinon ;
- Les deux parcours traitent les sommets différemment :
  - profondeur : le voisin u de v est adopté par v si u n'est pas visité ;

Calcul des arbres de parcours

- Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.
- On dira qu'un sommet est **orphelin** s'il n'a pas de parent, et **adopté** sinon ;
- Les deux parcours traitent les sommets différemment :
  - profondeur : le voisin u de v est adopté par v si u n'est pas visité ;
  - largeur : le voisin u de v est adopté par v s'il est orphelin ;

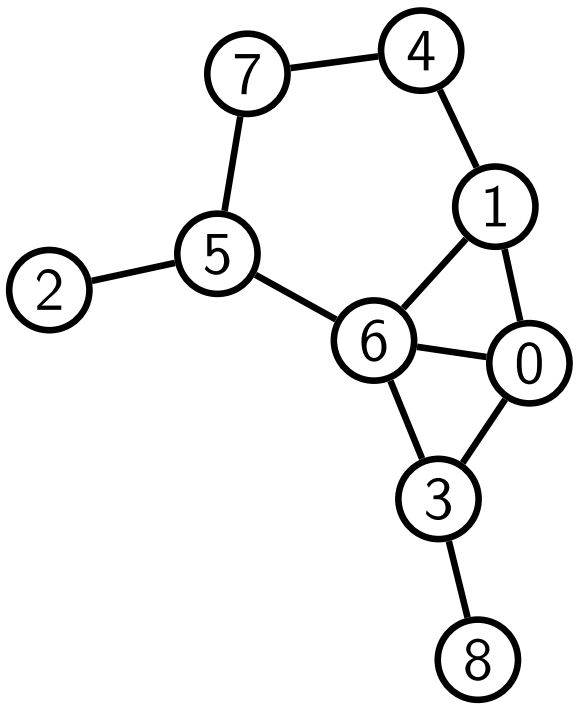
Calcul des arbres de parcours

- Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.
- On dira qu'un sommet est **orphelin** s'il n'a pas de parent, et **adopté** sinon ;
- Les deux parcours traitent les sommets différemment :
  - profondeur : le voisin u de v est adopté par v si u n'est pas visité ;
  - largeur : le voisin u de v est adopté par v s'il est orphelin ;
- Dans les deux cas, à la fin du parcours, le seul sommet orphelin est celui dont on est parti ;

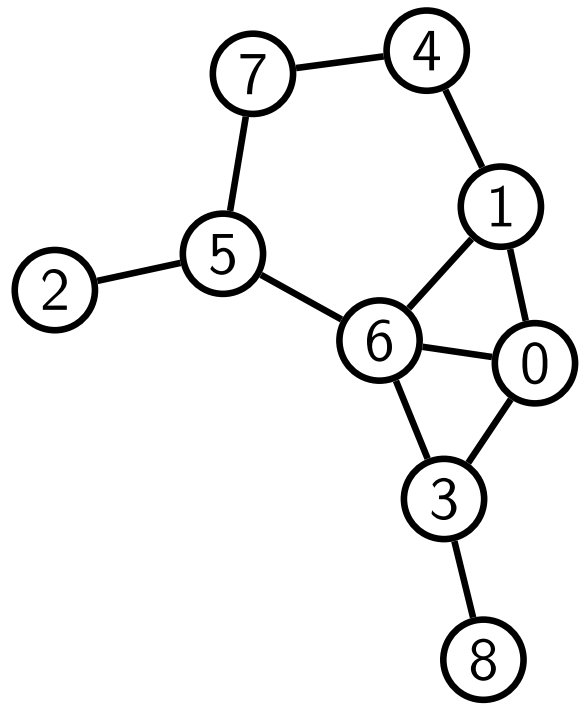
Calcul des arbres de parcours

Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



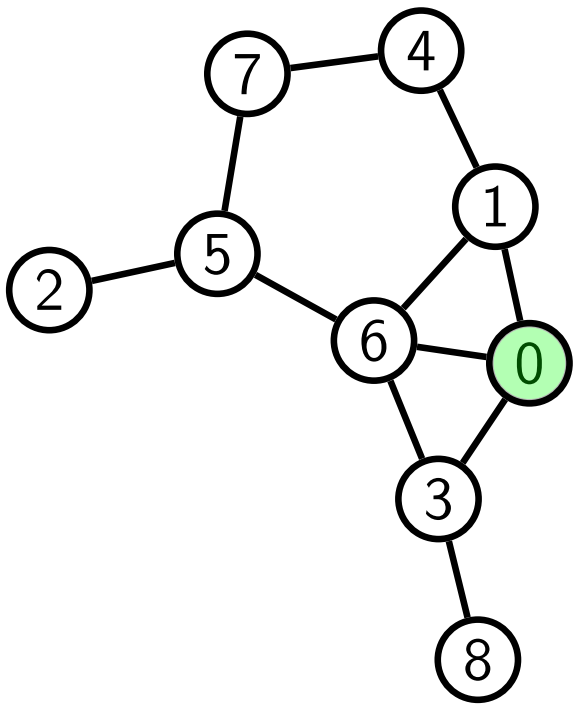
Exemple 7 (largeur)



Calcul des arbres de parcours

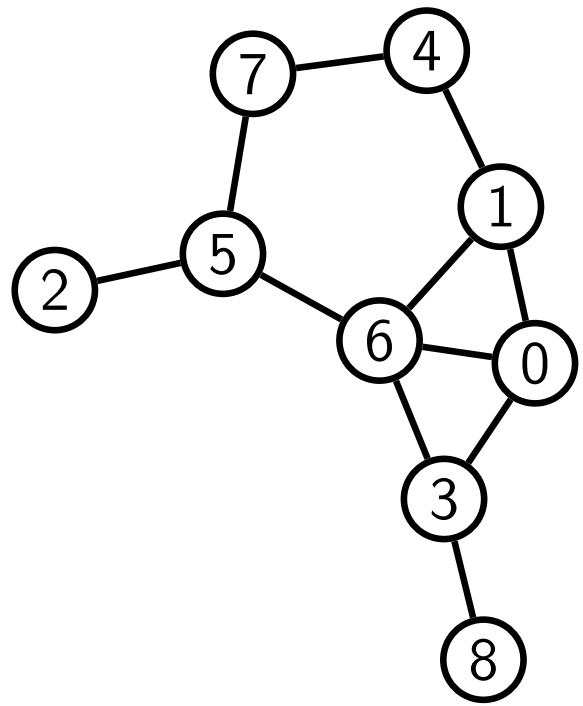
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0		0			0		

Exemple 7 (largeur)

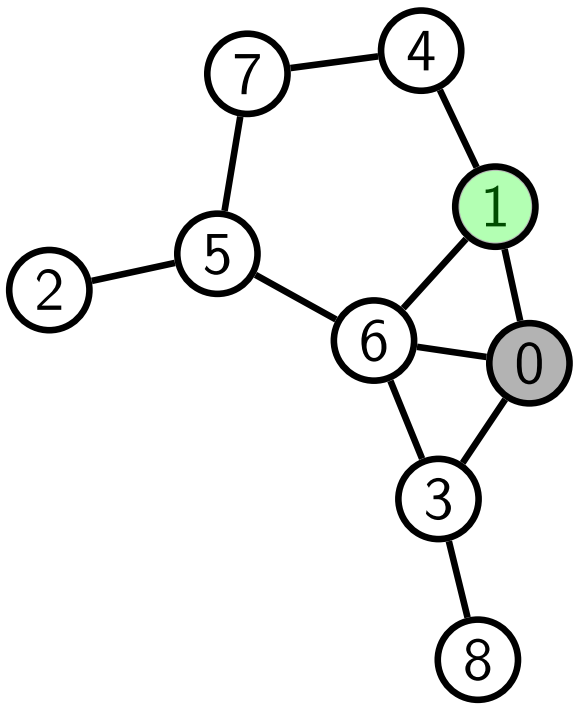


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

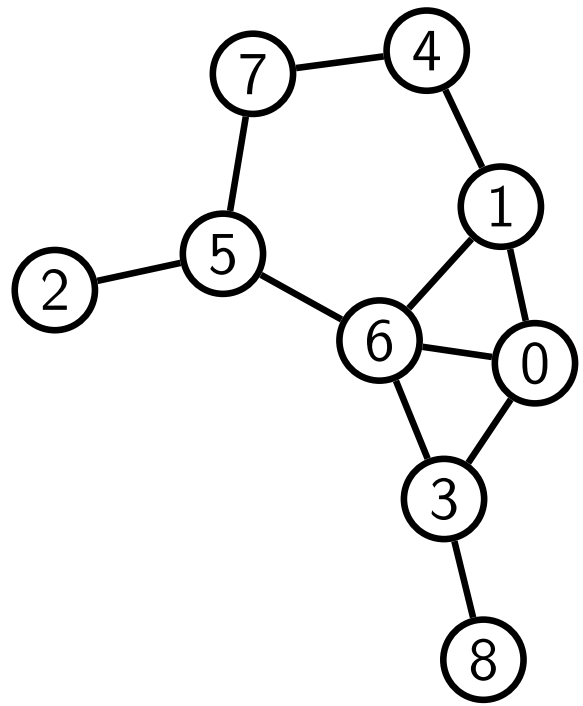
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0		0	1		1		

Exemple 7 (largeur)

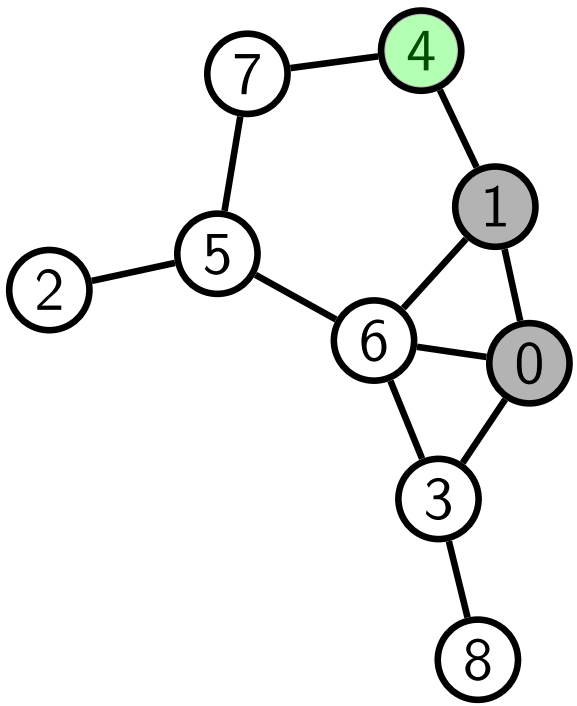


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

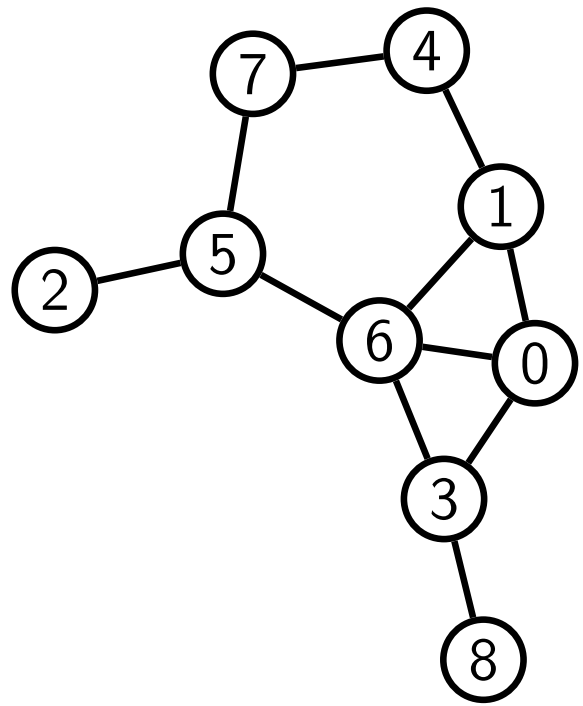
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0		0	1		1	4	

Exemple 7 (largeur)

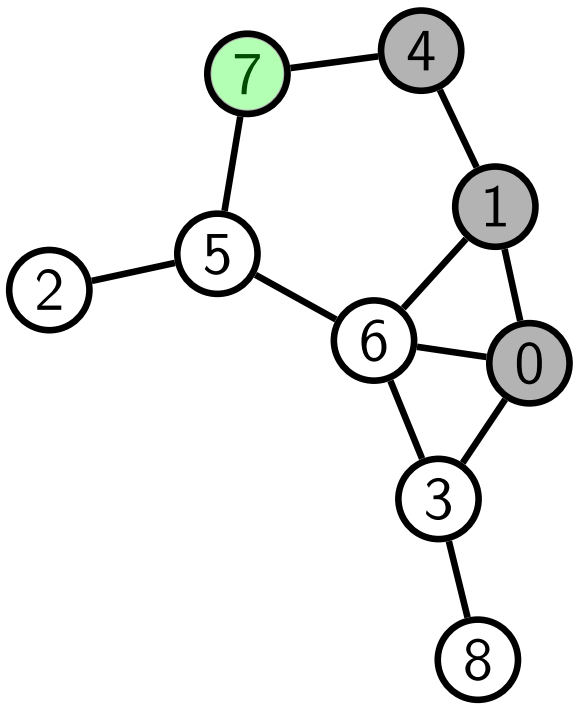


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

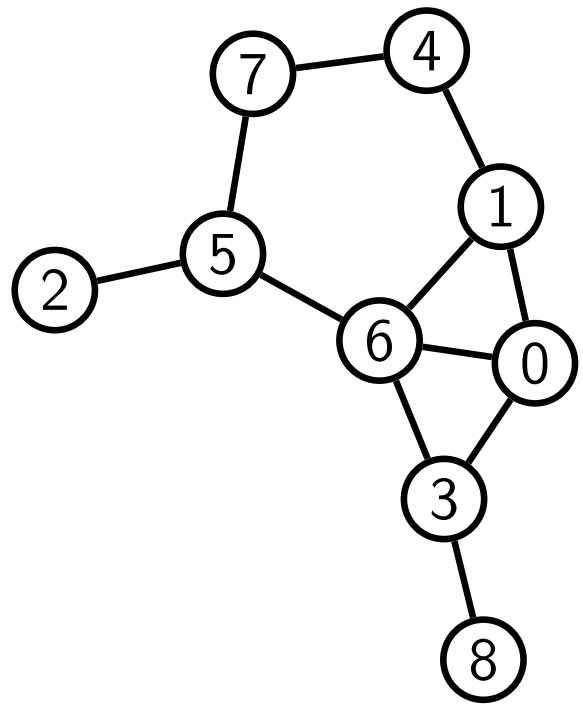
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0		0	1	7	1	4	

Exemple 7 (largeur)

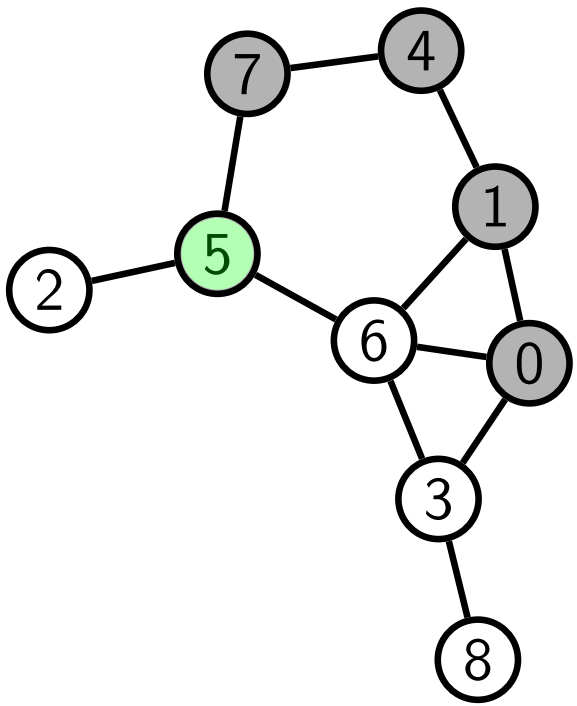


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

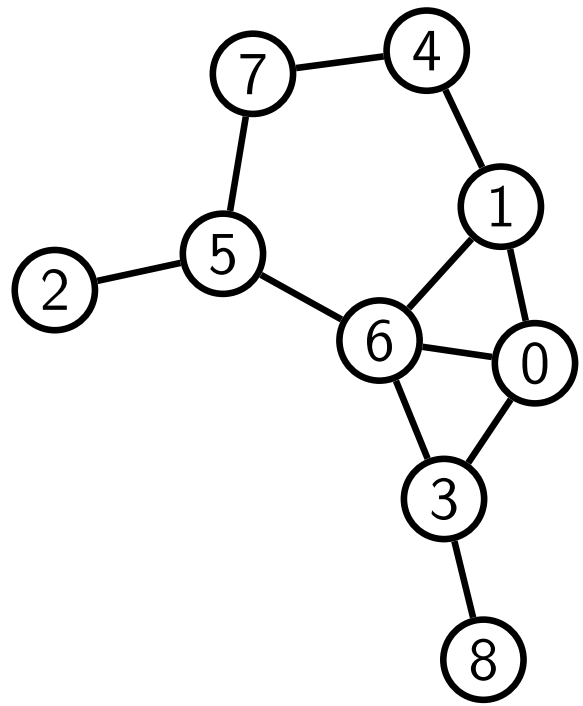
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	0	1	7	5	4	

Exemple 7 (largeur)

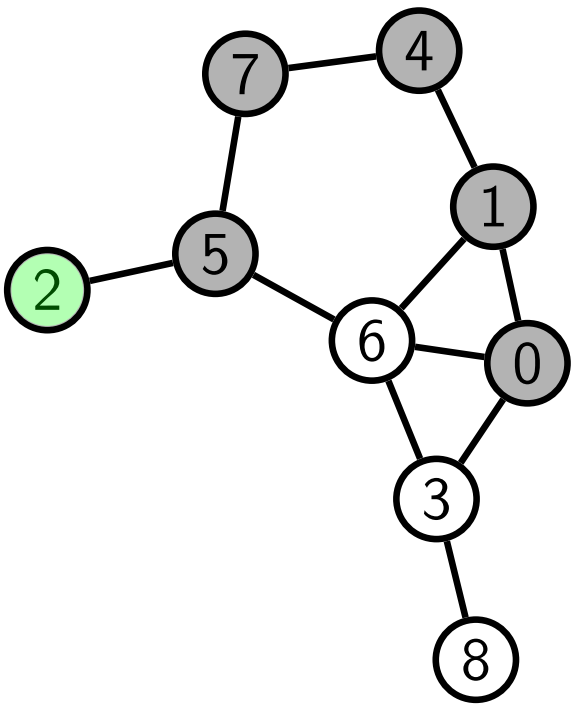


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

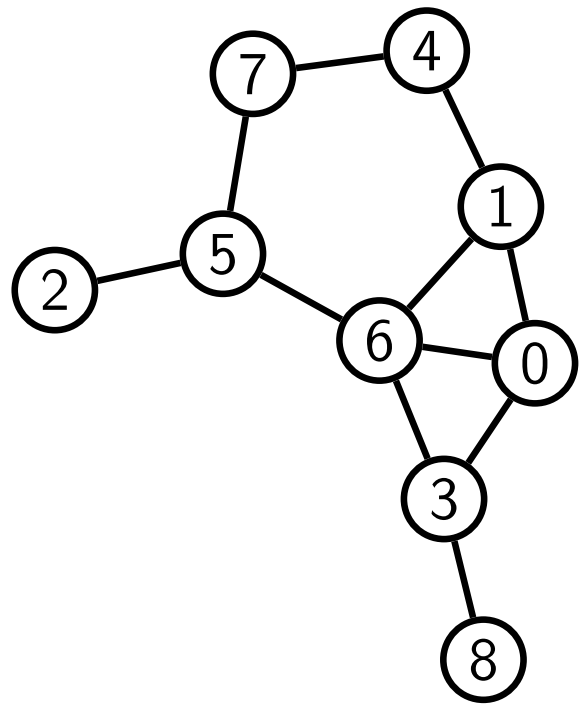
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	0	1	7	5	4	

Exemple 7 (largeur)

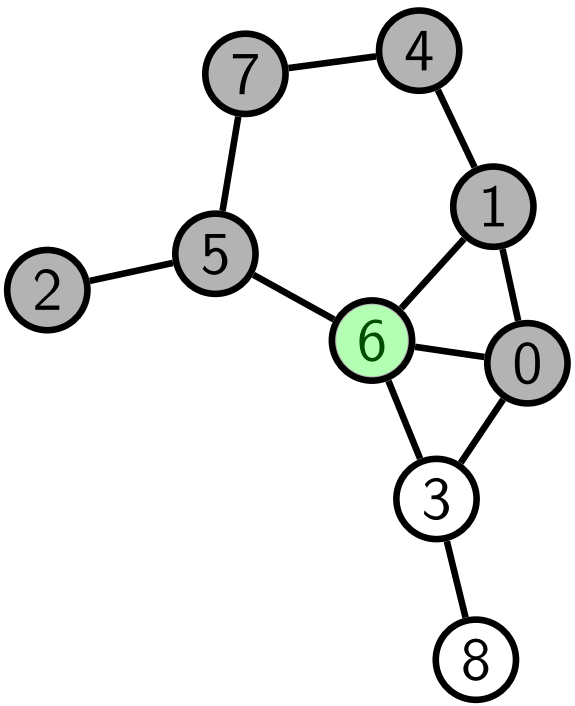


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

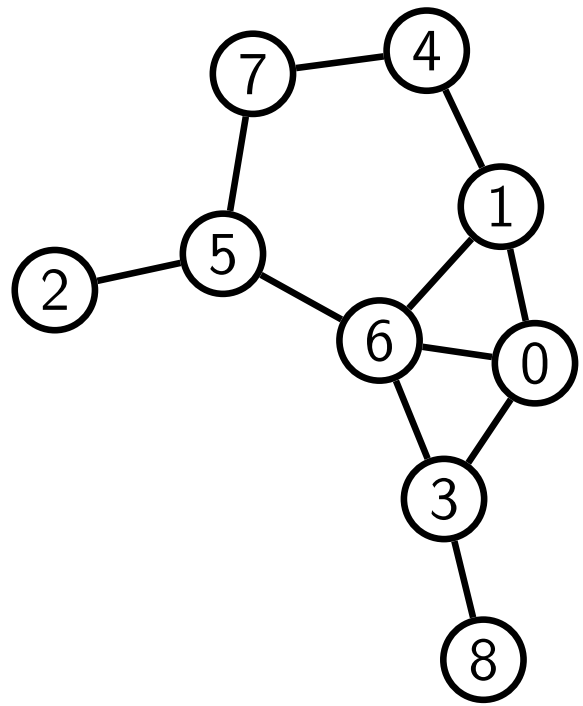
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	6	1	7	5	4	

Exemple 7 (largeur)

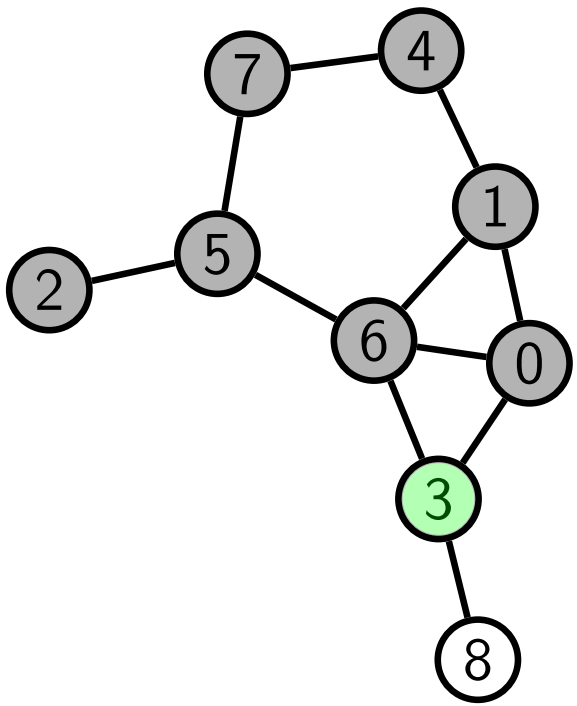


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

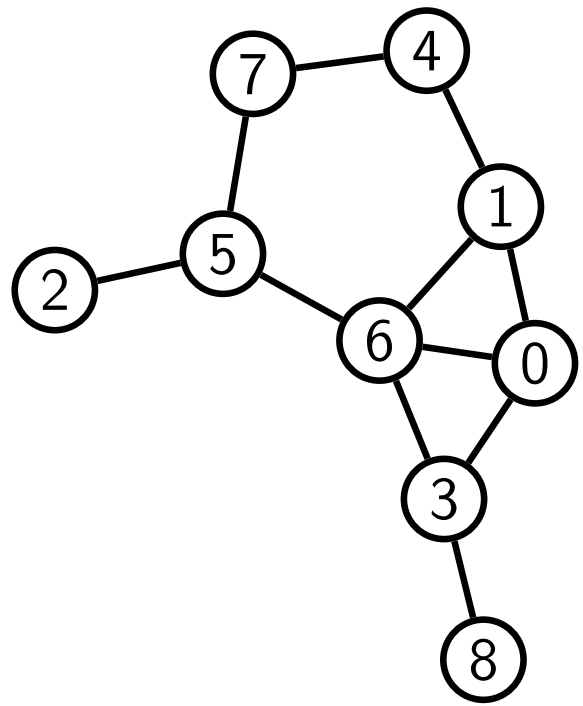
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	6	1	7	5	4	3

Exemple 7 (largeur)

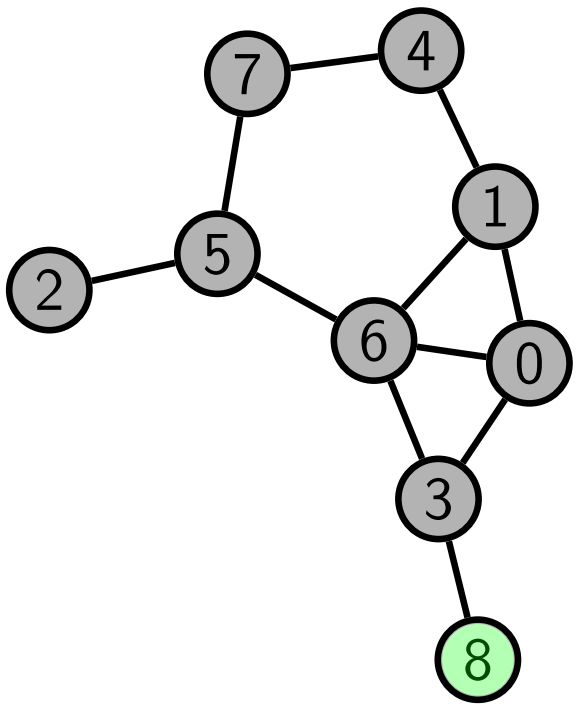


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

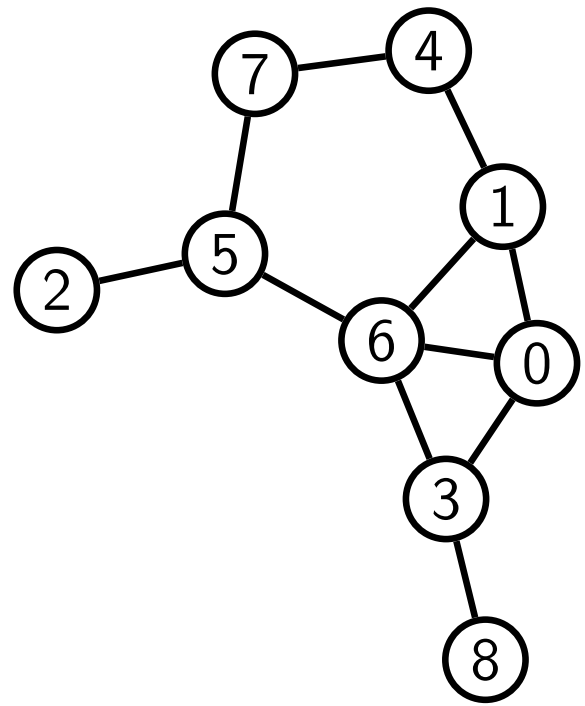
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	6	1	7	5	4	3

Exemple 7 (largeur)

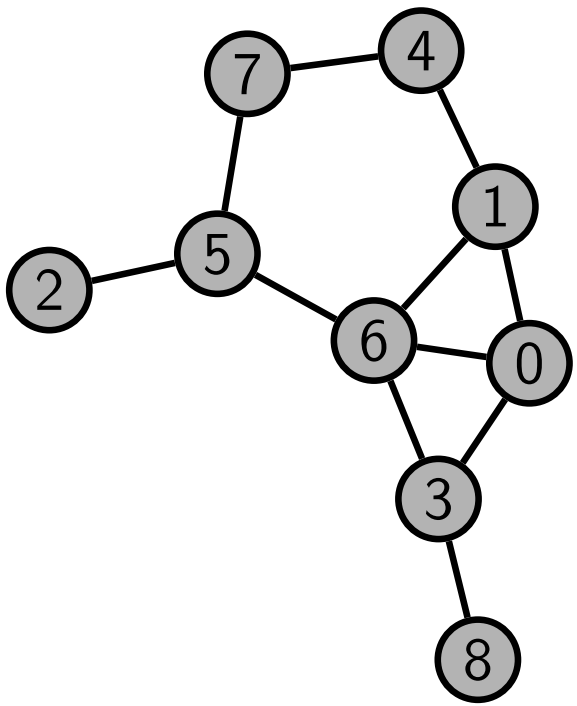


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

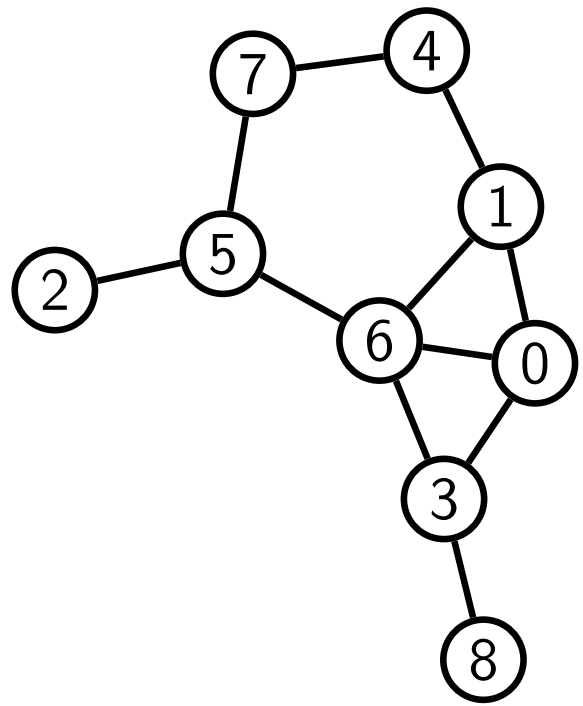
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	6	1	7	5	4	3

Exemple 7 (largeur)

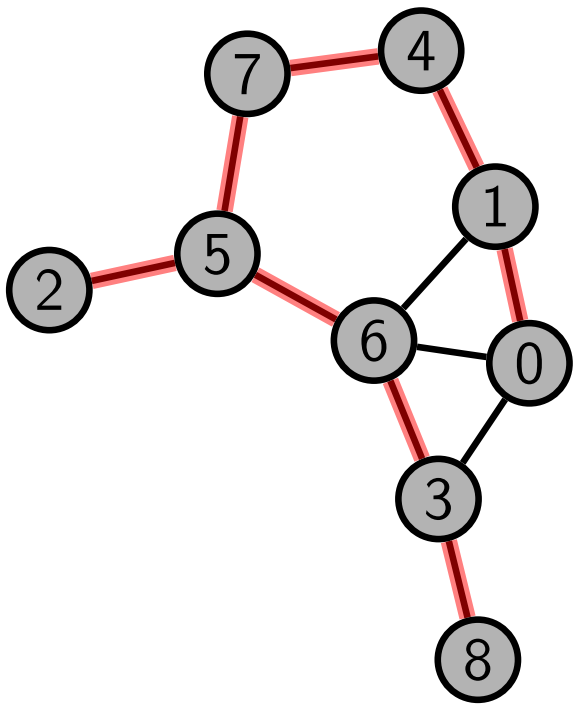


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

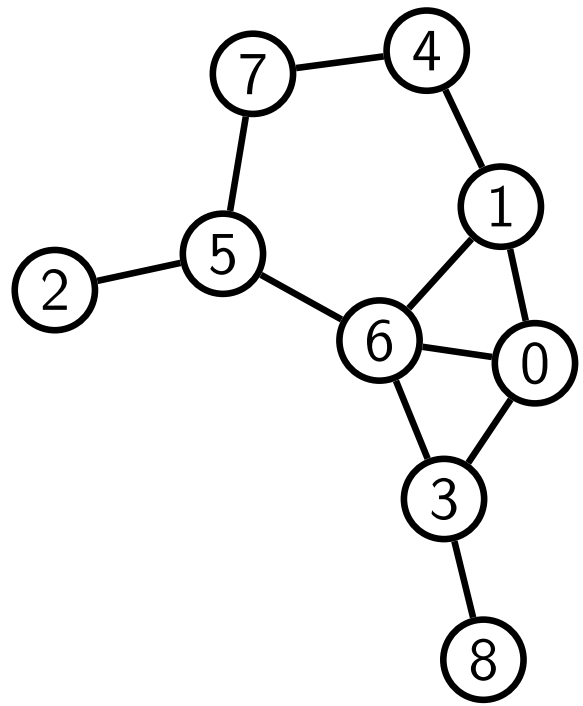
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	6	1	7	5	4	3

Exemple 7 (largeur)

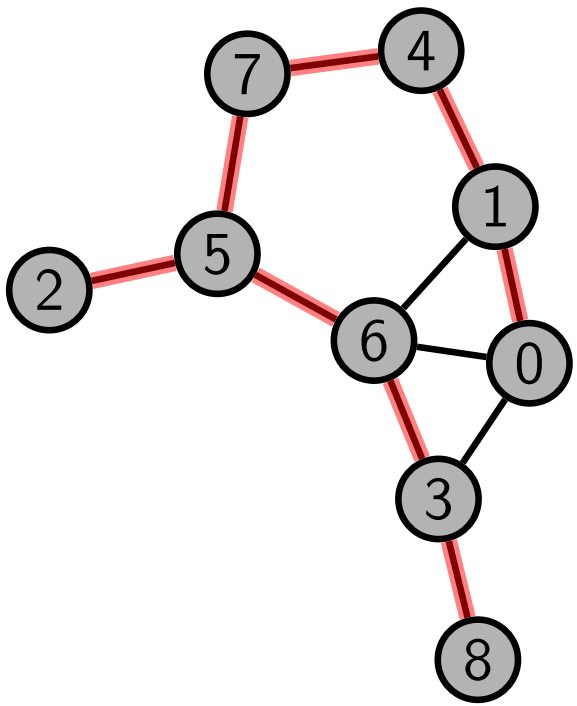


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

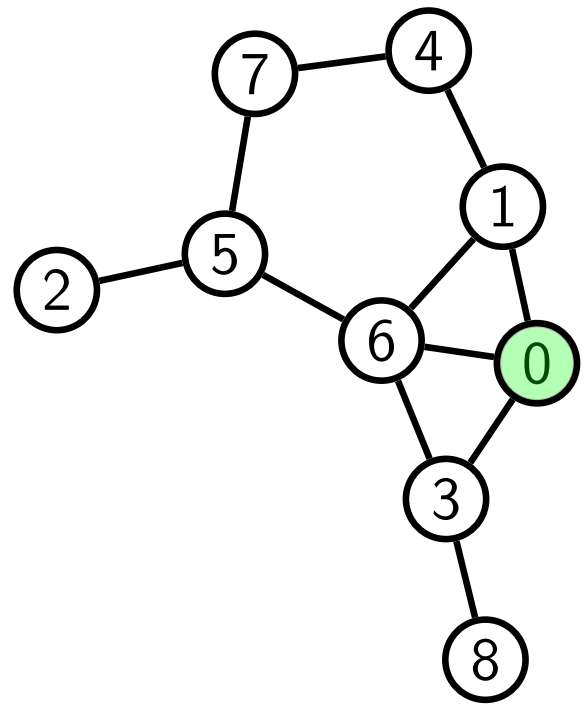
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	6	1	7	5	4	3

Exemple 7 (largeur)

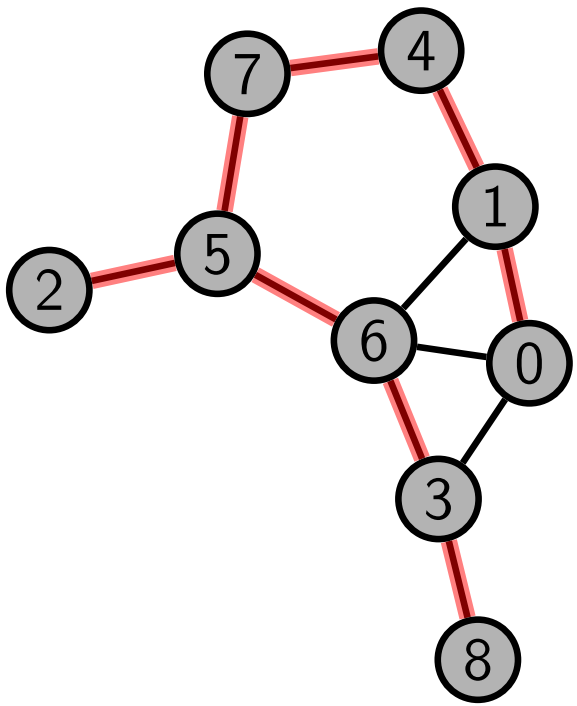


	0	1	2	3	4	5	6	7	8
parents :		0		0			0		

Calcul des arbres de parcours

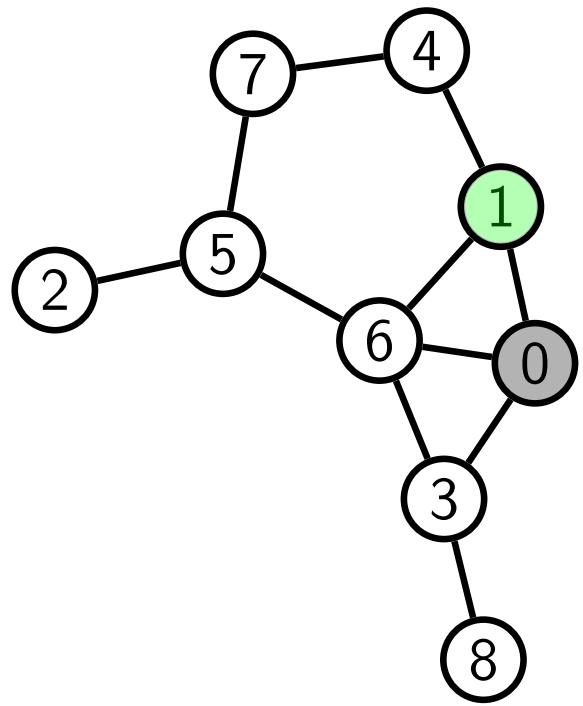
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	6	1	7	5	4	3

Exemple 7 (largeur)

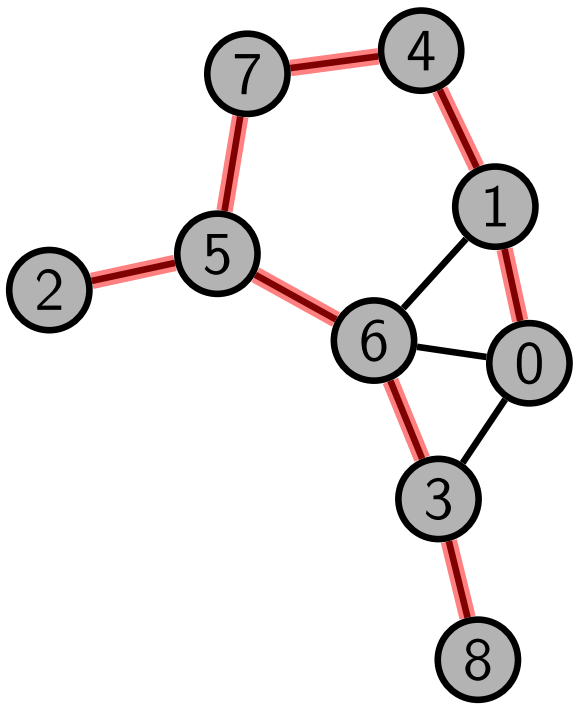


	0	1	2	3	4	5	6	7	8
parents :		0		0	1		0		

Calcul des arbres de parcours

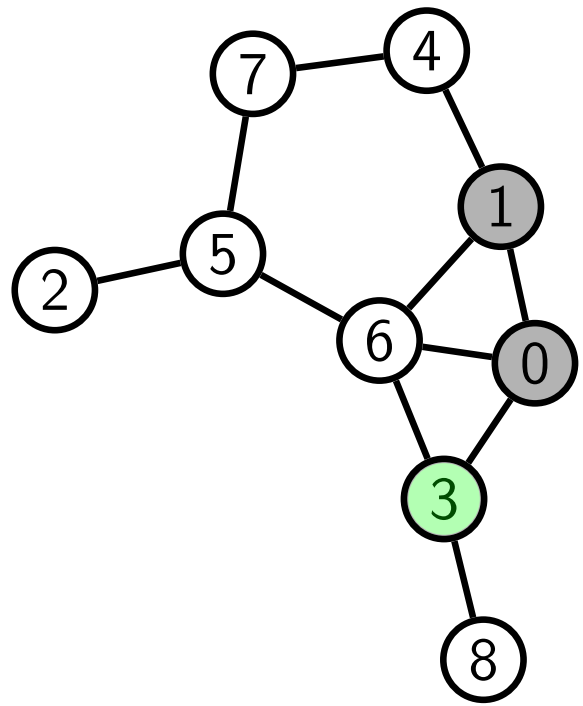
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	6	1	7	5	4	3

Exemple 7 (largeur)

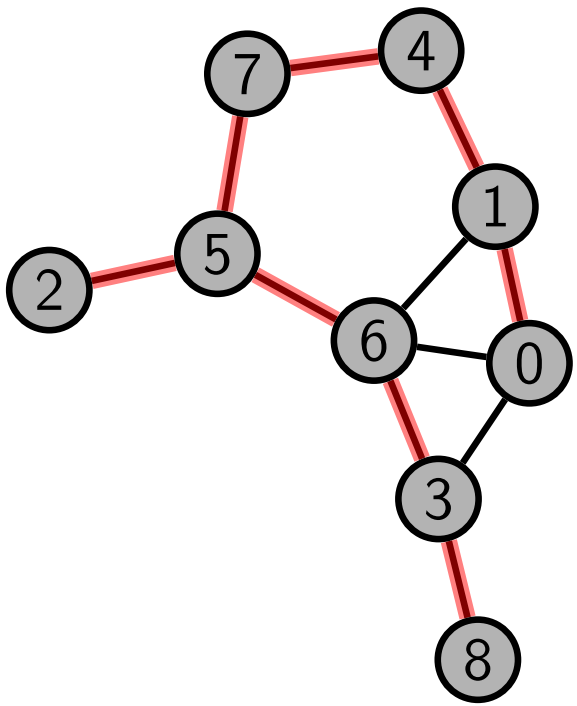


	0	1	2	3	4	5	6	7	8
parents :		0		0	1		0		3

Calcul des arbres de parcours

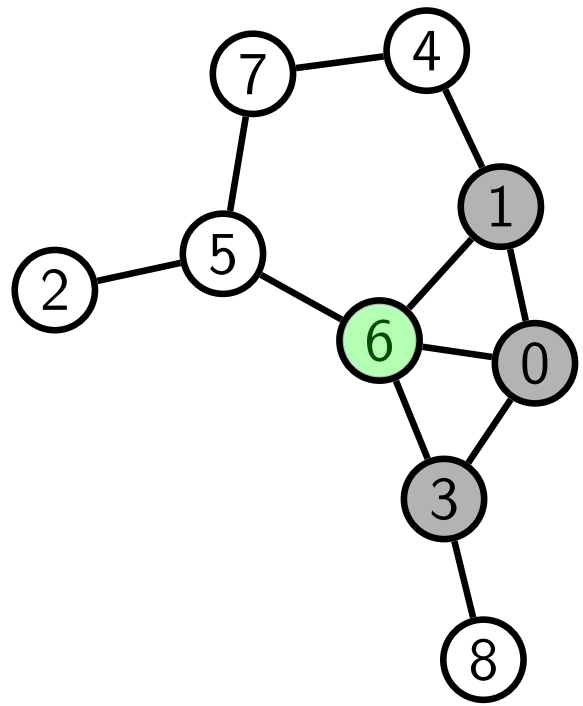
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	6	1	7	5	4	3

Exemple 7 (largeur)

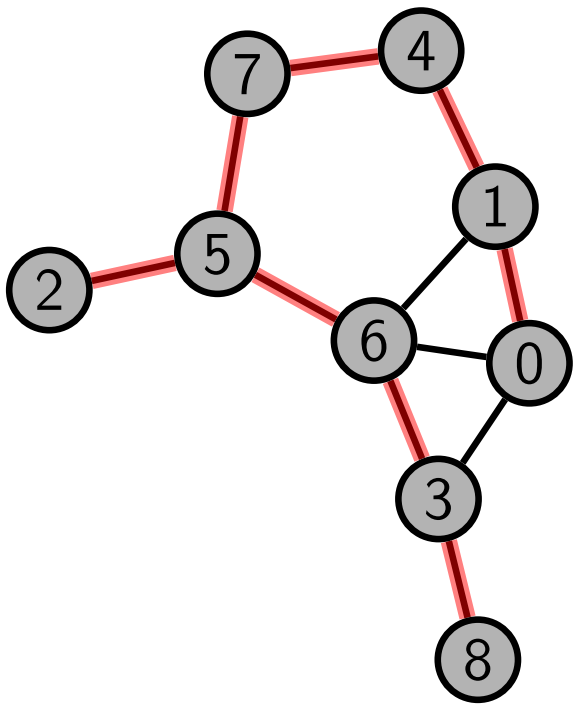


	0	1	2	3	4	5	6	7	8
parents :		0		0	1	6	0		3

Calcul des arbres de parcours

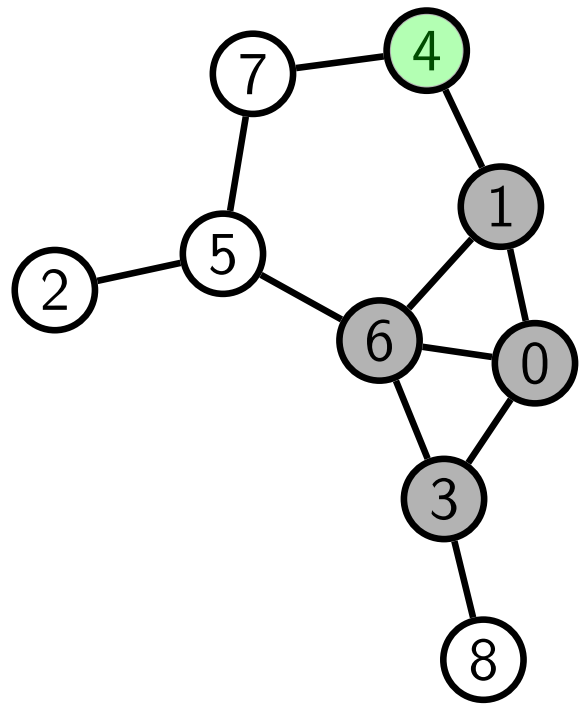
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	6	1	7	5	4	3

Exemple 7 (largeur)

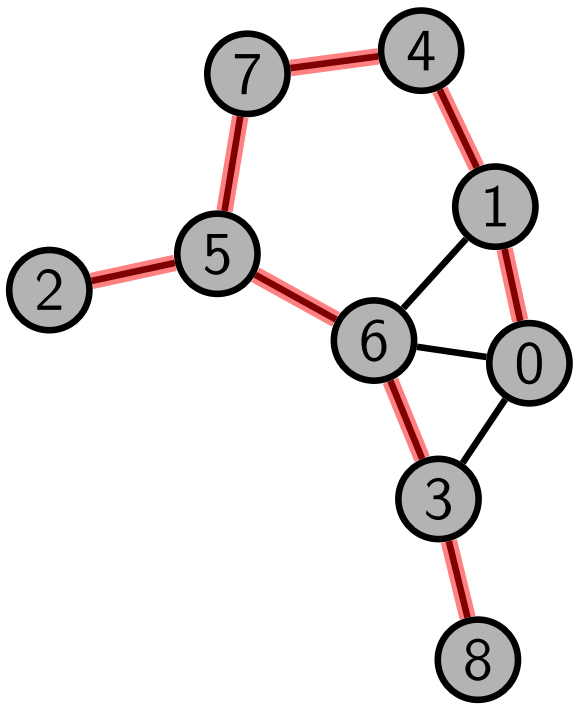


	0	1	2	3	4	5	6	7	8
parents :		0		0	1	6	0	4	3

Calcul des arbres de parcours

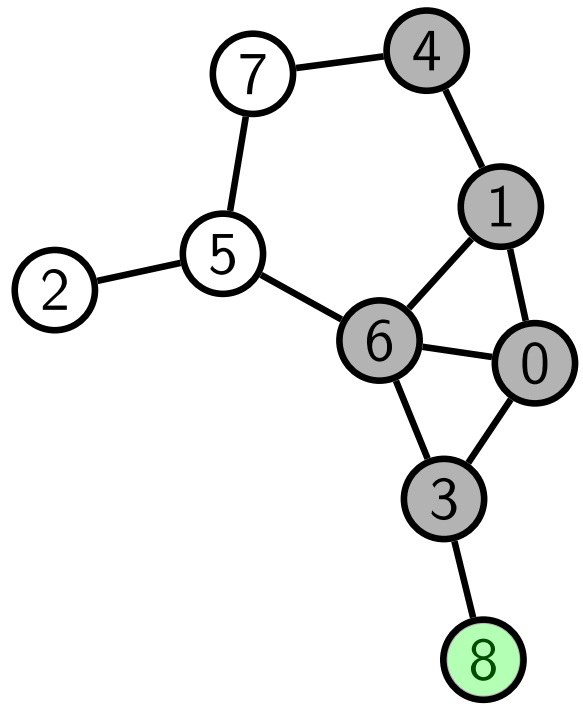
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	6	1	7	5	4	3

Exemple 7 (largeur)

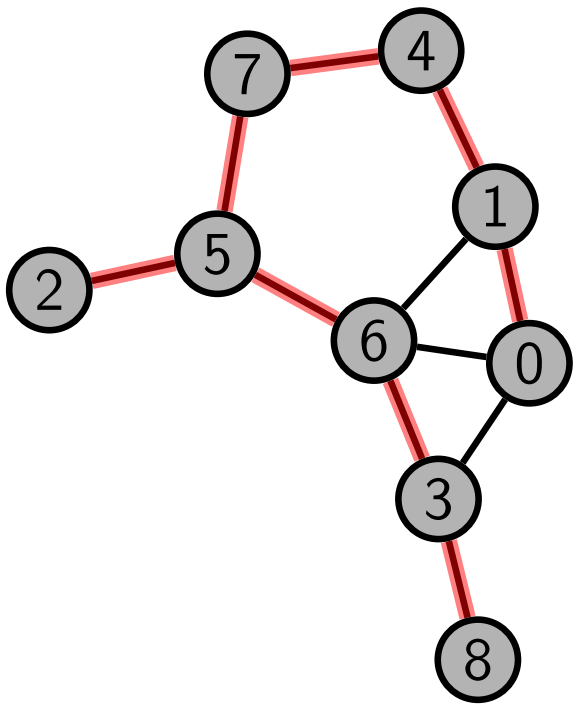


	0	1	2	3	4	5	6	7	8
parents :		0		0	1	6	0	4	3

Calcul des arbres de parcours

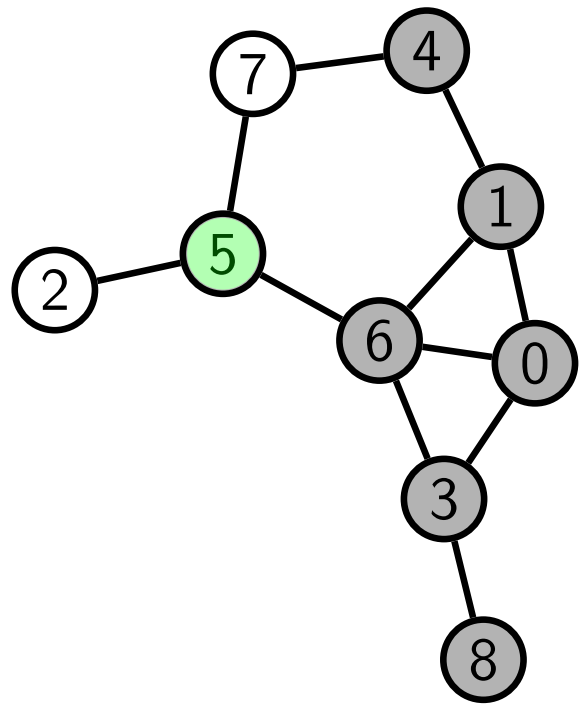
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	6	1	7	5	4	3

Exemple 7 (largeur)

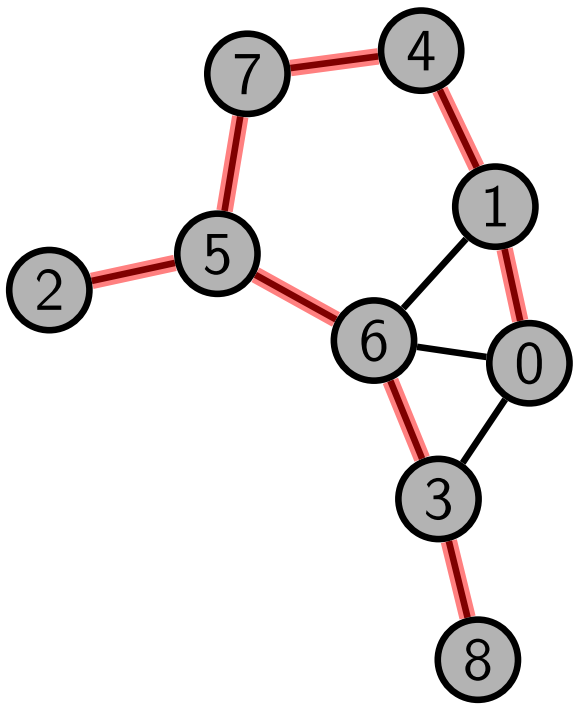


	0	1	2	3	4	5	6	7	8
parents :		0	5	0	1	6	0	4	3

Calcul des arbres de parcours

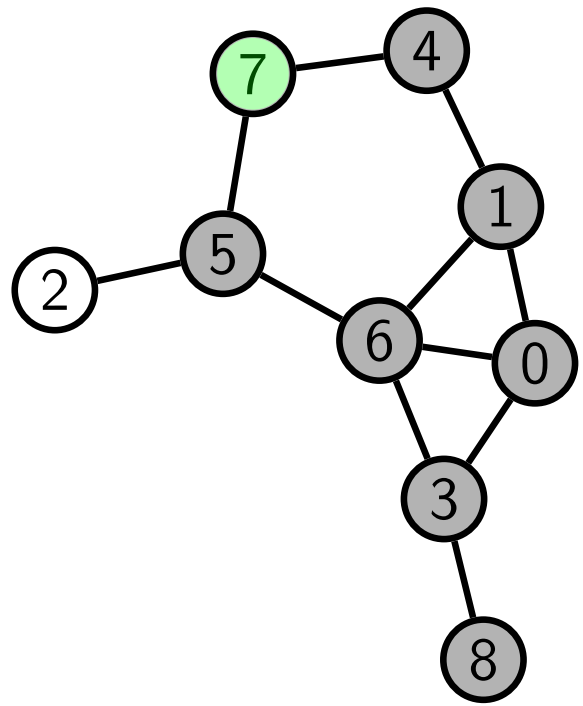
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	6	1	7	5	4	3

Exemple 7 (largeur)

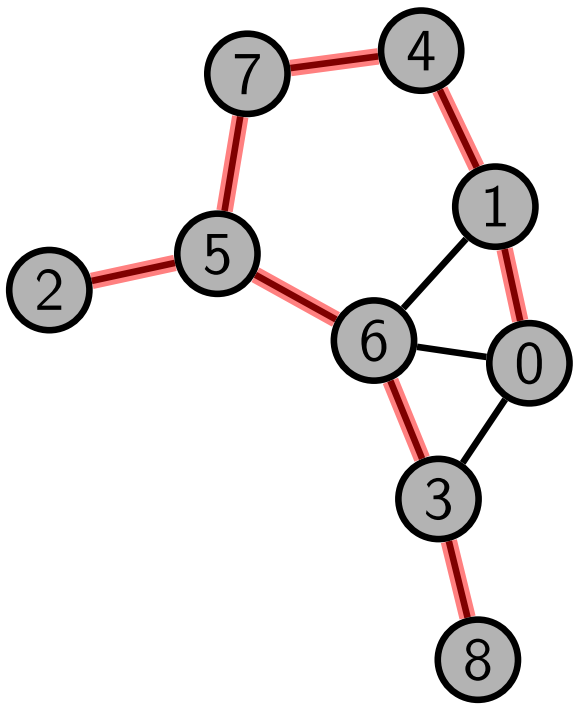


	0	1	2	3	4	5	6	7	8
parents :		0	5	0	1	6	0	4	3

Calcul des arbres de parcours

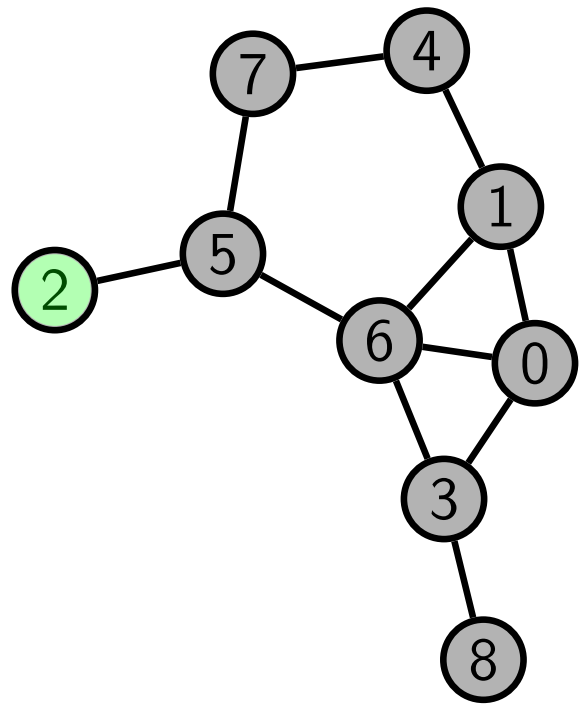
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	6	1	7	5	4	3

Exemple 7 (largeur)

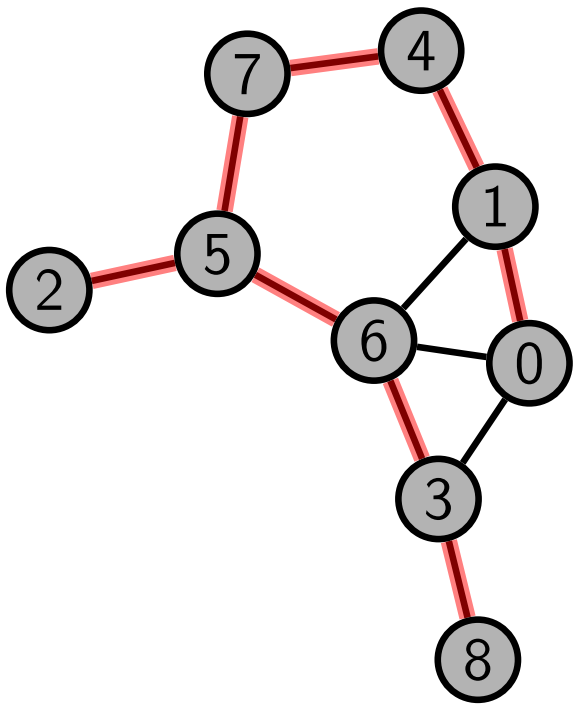


	0	1	2	3	4	5	6	7	8
parents :		0	5	0	1	6	0	4	3

Calcul des arbres de parcours

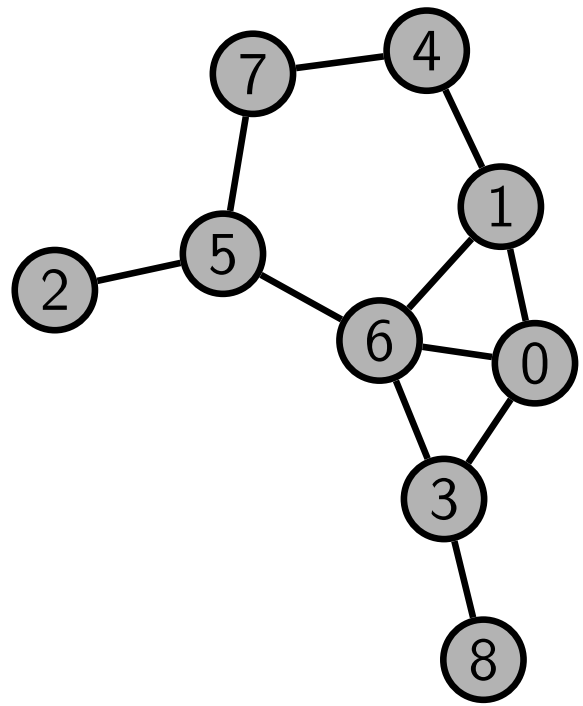
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	6	1	7	5	4	3

Exemple 7 (largeur)

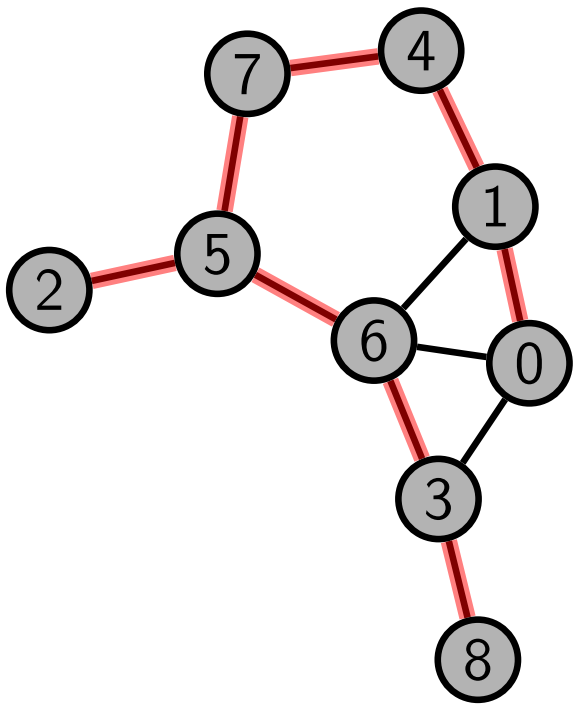


	0	1	2	3	4	5	6	7	8
parents :		0	5	0	1	6	0	4	3

Calcul des arbres de parcours

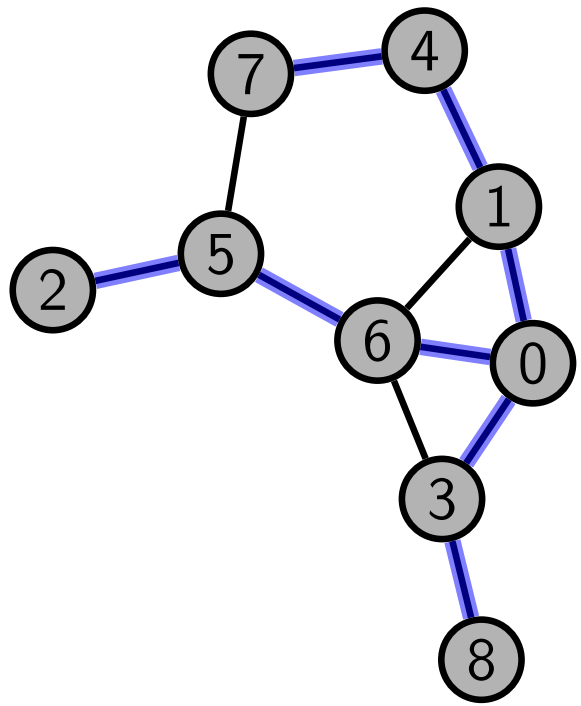
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 6 (profondeur)



	0	1	2	3	4	5	6	7	8
parents :		0	5	6	1	7	5	4	3

Exemple 7 (largeur)

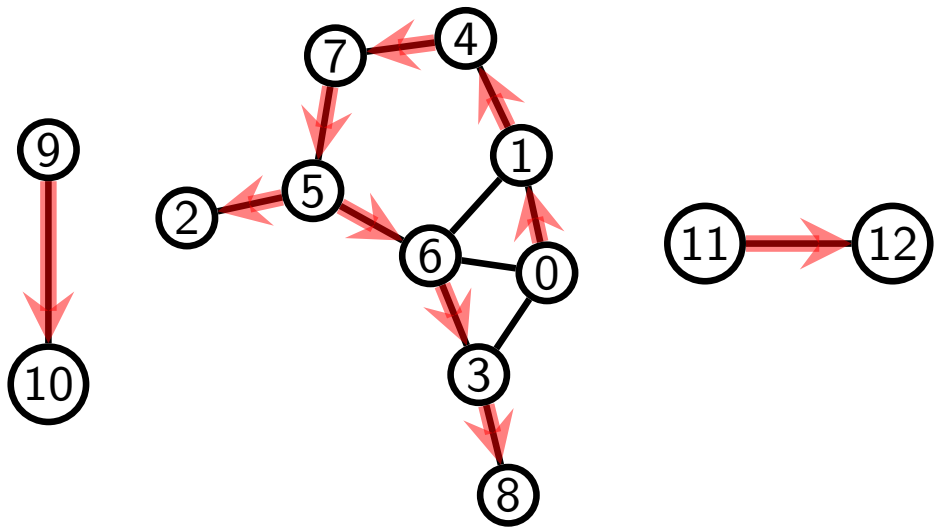


	0	1	2	3	4	5	6	7	8
parents :		0	5	0	1	6	0	4	3

Forêts de parcours

- Si le graphe est connexe, ces arbres sont dits "couvrants" car ils couvrent tous les sommets du graphe.
- Sinon, on cherchera à construire une **forêt couvrante** (un arbre par "morceau" du graphe) ;

Exemple 8 (forêt couvrante)



Connexité et composantes connexes

Comme on l'a vu, un graphe connexe est un graphe "en un seul morceau".

Définition 2

Une **composante connexe** d'un graphe G est un sous-graphe connexe H de G qui est maximal, c'est-à-dire qu'il n'existe pas de sommet de G à la fois accessible à partir d'un élément de $V(H)$ et hors de $V(H)$.

Comment identifier ces composantes connexes ?

Identification des composantes connexes

Il suffit de lancer un parcours à partir de chacun des sommets du graphe.

Algorithme 5 : ComposantesConnexes(G)

Entrées : un graphe non orienté G.
Sortie : les composantes connexes de G, identifiées par la liste de leurs sommets.

```
1 résultat ← liste();
2 visités ← tableau(G.nombre_sommets(), FAUX);
3 pour chaque sommet dans G.sommets() faire
4 |   si ¬ visités[sommet] alors
5 |   |   résultat.ajouter_en_fin(Largeur(G, sommet, visités)) ;
6 renvoyer résultat;
```

Graphes bipartis

Définition 3

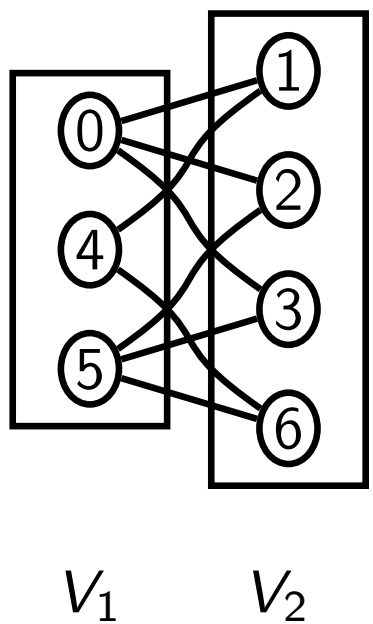
Un graphe $G = (V, E)$ est **biparti** s'il existe une bipartition $V = V_1 \cup V_2$ telle que deux sommets de la même partie ne sont jamais adjacents.

Graphes bipartis

Définition 3

Un graphe $G = (V, E)$ est **biparti** s'il existe une bipartition $V = V_1 \cup V_2$ telle que deux sommets de la même partie ne sont jamais adjacents.

Exemple 9

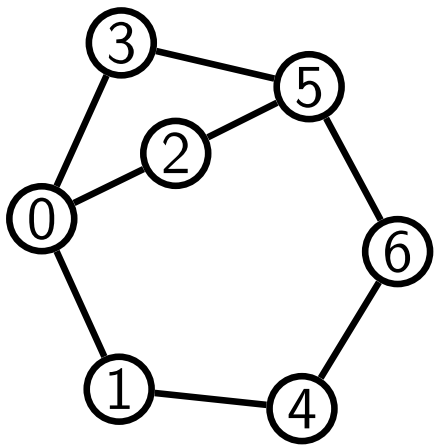


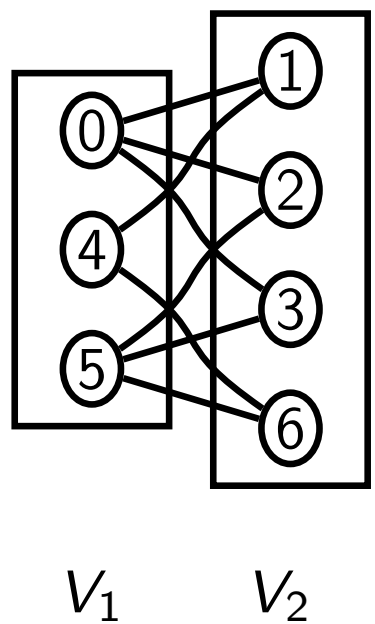
Graphes bipartis

Définition 3

Un graphe $G = (V, E)$ est **biparti** s'il existe une bipartition $V = V_1 \cup V_2$ telle que deux sommets de la même partie ne sont jamais adjacents.

Exemple 9



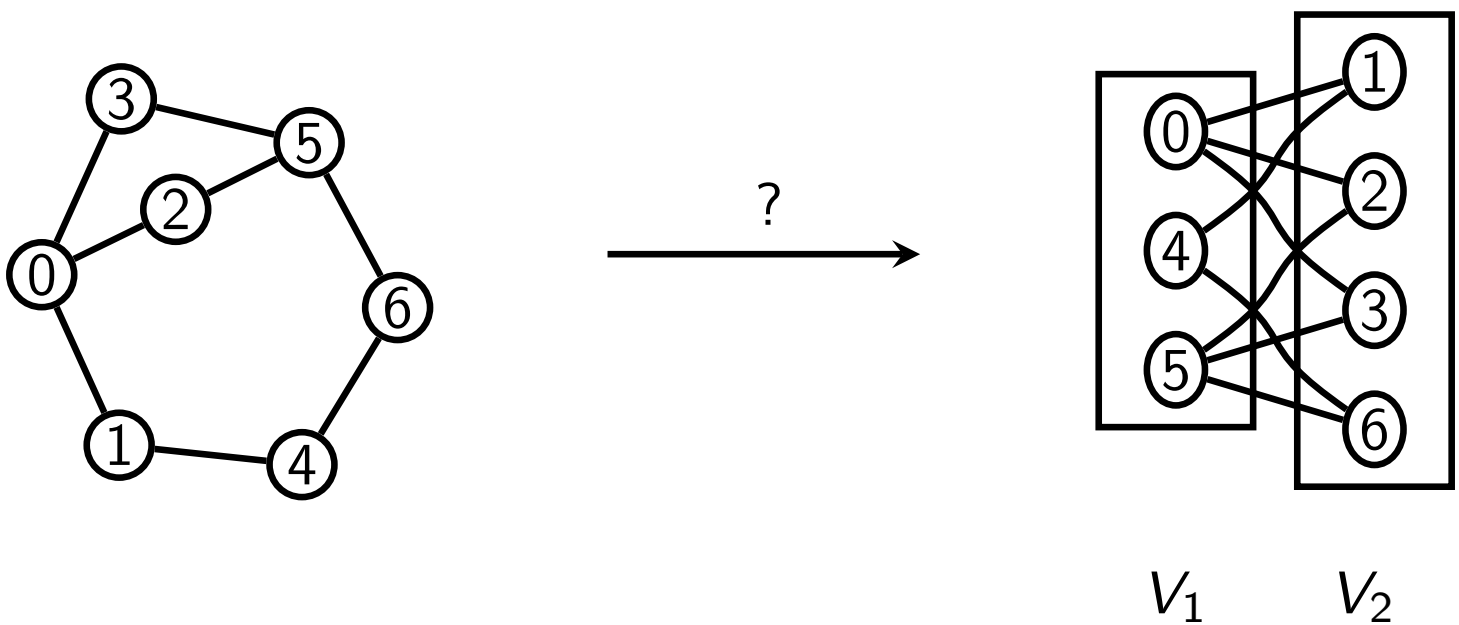


Graphes bipartis

Définition 3

Un graphe $G = (V, E)$ est **biparti** s'il existe une bipartition $V = V_1 \cup V_2$ telle que deux sommets de la même partie ne sont jamais adjacents.

Exemple 9



Reconnaissance de graphes bipartis

- De nombreux problèmes "difficiles" sur les graphes deviennent "faciles" sur des graphes bipartis ;

Reconnaissance de graphes bipartis

- De nombreux problèmes "difficiles" sur les graphes deviennent "faciles" sur des graphes bipartis ;
- Il est donc important de pouvoir les reconnaître.

Reconnaissance de graphes bipartis

- De nombreux problèmes "difficiles" sur les graphes deviennent "faciles" sur des graphes bipartis ;
- Il est donc important de pouvoir les reconnaître.
- Comment procéder ? Approche naïve : examiner toutes les bipartitions jusqu'à ce qu'on en trouve une satisfaisante ou qu'on puisse conclure qu'il n'en existe pas ;

Reconnaissance de graphes bipartis

- De nombreux problèmes "difficiles" sur les graphes deviennent "faciles" sur des graphes bipartis ;
- Il est donc important de pouvoir les reconnaître.
- Comment procéder ? Approche naïve : examiner toutes les bipartitions jusqu'à ce qu'on en trouve une satisfaisante ou qu'on puisse conclure qu'il n'en existe pas ;
- Mauvaise idée : il y a $O(2^{|V|})$ bipartitions.

Reconnaissance de graphes bipartis

- De nombreux problèmes "difficiles" sur les graphes deviennent "faciles" sur des graphes bipartis ;
- Il est donc important de pouvoir les reconnaître.
- Comment procéder ? Approche naïve : examiner toutes les bipartitions jusqu'à ce qu'on en trouve une satisfaisante ou qu'on puisse conclure qu'il n'en existe pas ;
- Mauvaise idée : il y a $O(2^{|V|})$ bipartitions.
- La caractérisation suivante va nous donner un algorithme efficace.

Caractérisation des graphes bipartis

Théorème 4

Un graphe est biparti si et seulement s'il ne contient pas de cycle de longueur impaire.

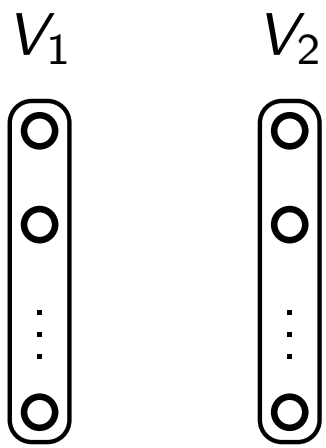
Caractérisation des graphes bipartis

Théorème 4

Un graphe est biparti si et seulement s'il ne contient pas de cycle de longueur impaire.

Démonstration.

$\Rightarrow$: tout cycle partant de v y revient par des aller-retours :



. . . et donc tout cycle de G est pair.

□

Caractérisation des graphes bipartis

Théorème 4

Un graphe est biparti si et seulement s'il ne contient pas de cycle de longueur impaire.

Démonstration.

$\Leftarrow$: $\equiv$ "si G n'est pas biparti, alors il contient un cycle impair".

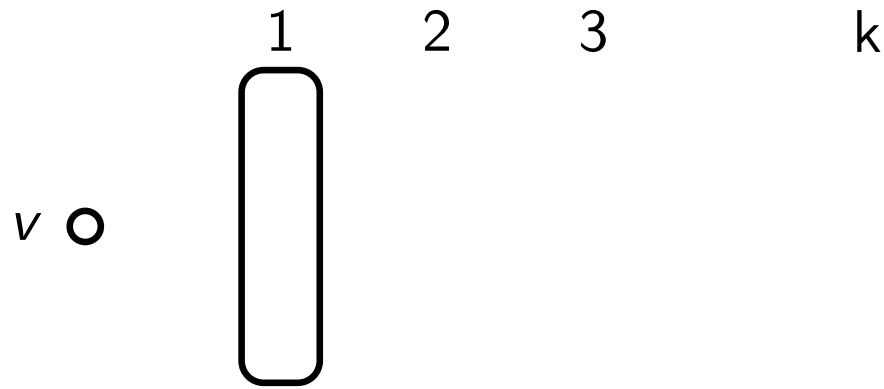
Caractérisation des graphes bipartis

Théorème 4

Un graphe est biparti si et seulement s'il ne contient pas de cycle de longueur impaire.

Démonstration.

$\Leftarrow$: $\equiv$ "si G n'est pas biparti, alors il contient un cycle impair".



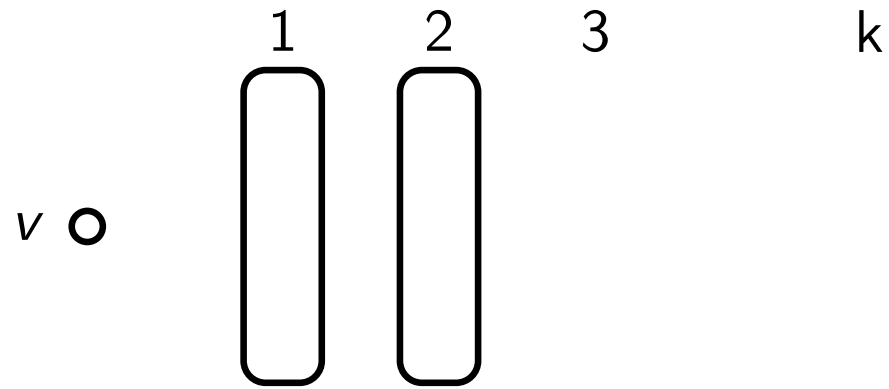
Caractérisation des graphes bipartis

Théorème 4

Un graphe est biparti si et seulement s'il ne contient pas de cycle de longueur impaire.

Démonstration.

$\Leftarrow$: $\equiv$ "si G n'est pas biparti, alors il contient un cycle impair".



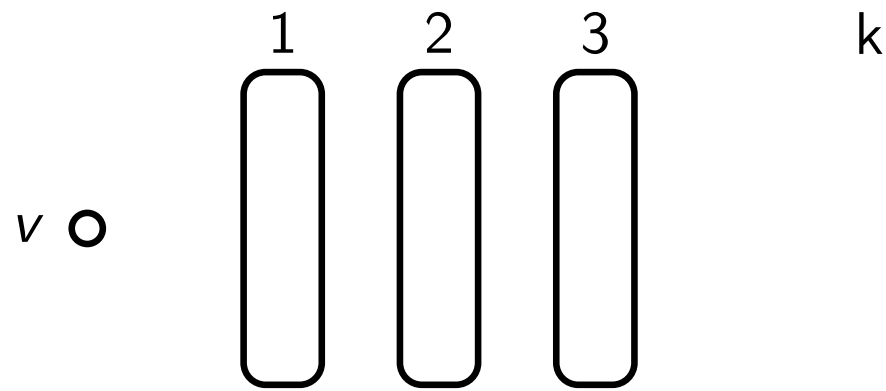
Caractérisation des graphes bipartis

Théorème 4

Un graphe est biparti si et seulement s'il ne contient pas de cycle de longueur impaire.

Démonstration.

$\Leftarrow$: $\equiv$ "si G n'est pas biparti, alors il contient un cycle impair".



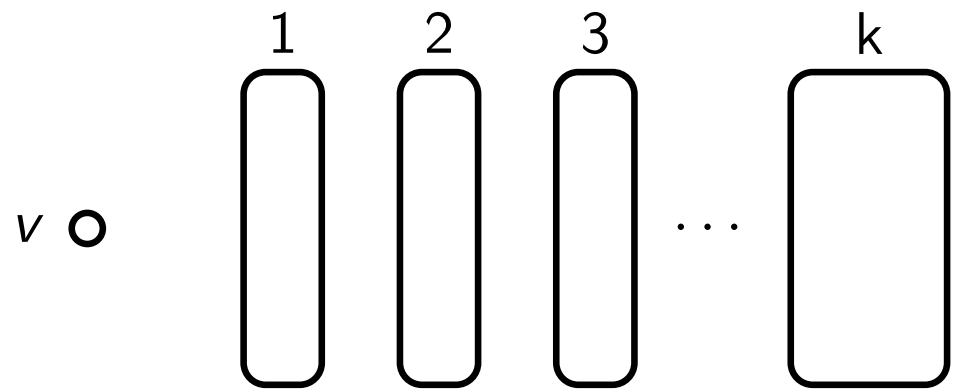
Caractérisation des graphes bipartis

Théorème 4

Un graphe est biparti si et seulement s'il ne contient pas de cycle de longueur impaire.

Démonstration.

$\Leftarrow$: $\equiv$ "si G n'est pas biparti, alors il contient un cycle impair".



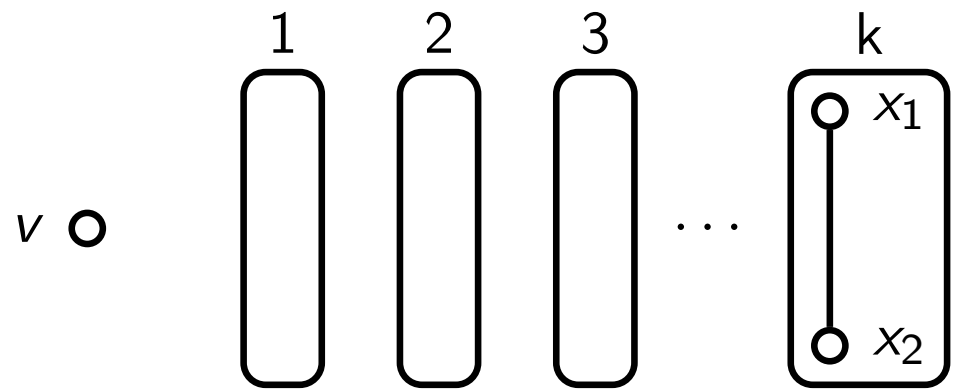
Caractérisation des graphes bipartis

Théorème 4

Un graphe est biparti si et seulement s'il ne contient pas de cycle de longueur impaire.

Démonstration.

$\Leftarrow$: $\equiv$ "si G n'est pas biparti, alors il contient un cycle impair".



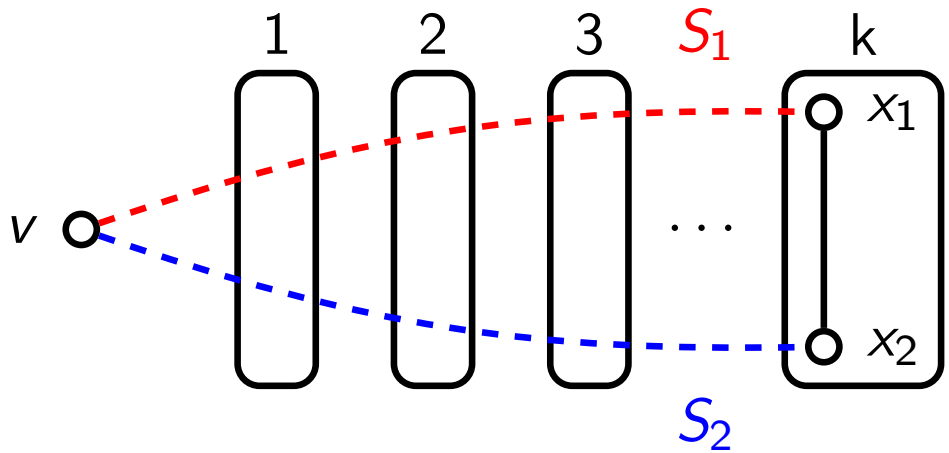
Caractérisation des graphes bipartis

Théorème 4

Un graphe est biparti si et seulement s'il ne contient pas de cycle de longueur impaire.

Démonstration.

$\Leftarrow$: $\equiv$ "si G n'est pas biparti, alors il contient un cycle impair".



□

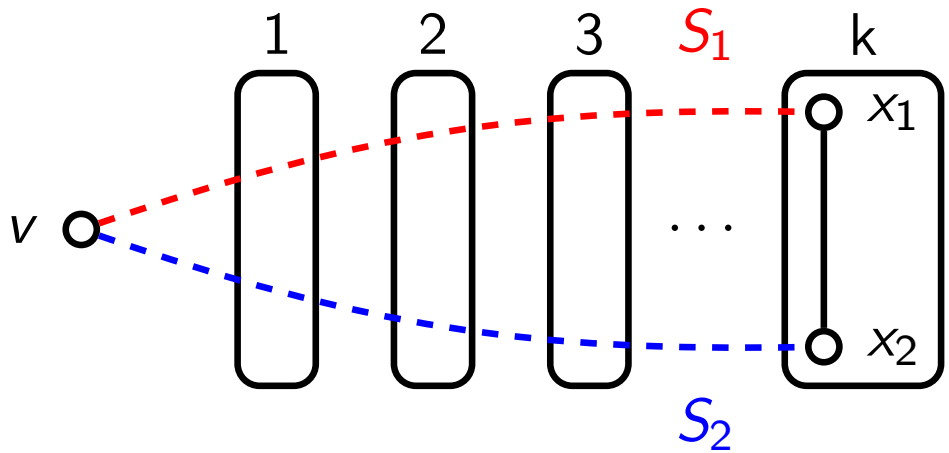
Caractérisation des graphes bipartis

Théorème 4

Un graphe est biparti si et seulement s'il ne contient pas de cycle de longueur impaire.

Démonstration.

$\Leftarrow$: $\equiv$ "si G n'est pas biparti, alors il contient un cycle impair".



$S_1 + \{x_1, x_2\} + S_2$ est un cycle de longueur impaire.

Algorithme de reconnaissance des graphes bipartis

- On en déduit l'algorithme de reconnaissance suivant :

Algorithme de reconnaissance des graphes bipartis

- On en déduit l'algorithme de reconnaissance suivant :
  - choisir un sommet arbitraire et le colorier (rouge) ;

Algorithme de reconnaissance des graphes bipartis

- On en déduit l'algorithme de reconnaissance suivant :
  - choisir un sommet arbitraire et le colorier (rouge) ;
  - colorier ses voisins en bleu ;

Algorithme de reconnaissance des graphes bipartis

- On en déduit l'algorithme de reconnaissance suivant :
  - choisir un sommet arbitraire et le colorier (rouge) ;
  - colorier ses voisins en bleu ;
  - colorier leurs voisins pas encore coloriés en rouge ;

Algorithme de reconnaissance des graphes bipartis

- On en déduit l'algorithme de reconnaissance suivant :
  - choisir un sommet arbitraire et le colorier (rouge) ;
  - colorier ses voisins en bleu ;
  - colorier leurs voisins pas encore coloriés en rouge ;
  - ...

Algorithme de reconnaissance des graphes bipartis

- On en déduit l'algorithme de reconnaissance suivant :
  - choisir un sommet arbitraire et le colorier (rouge) ;
  - colorier ses voisins en bleu ;
  - colorier leurs voisins pas encore coloriés en rouge ;
  - ...
- Si l'on essaie d'attribuer deux couleurs différentes à un même sommet, on a un cycle impair et le graphe n'est pas biparti ;

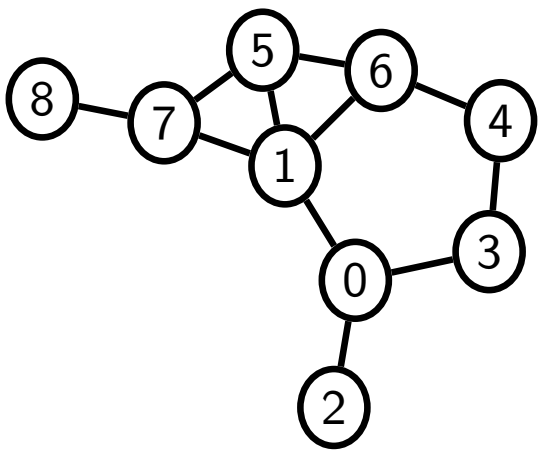
Algorithme de reconnaissance des graphes bipartis

- On en déduit l'algorithme de reconnaissance suivant :
  - choisir un sommet arbitraire et le colorier (rouge) ;
  - colorier ses voisins en bleu ;
  - colorier leurs voisins pas encore coloriés en rouge ;
  - ...
- Si l'on essaie d'attribuer deux couleurs différentes à un même sommet, on a un cycle impair et le graphe n'est pas biparti ;
- Sinon, le graphe est biparti **et** on obtient une bipartition ($V = V_1 \cup V_2$) ;

Algorithme de reconnaissance des graphes bipartis

- On en déduit l'algorithme de reconnaissance suivant :
  - choisir un sommet arbitraire et le colorier (rouge) ;
  - colorier ses voisins en bleu ;
  - colorier leurs voisins pas encore coloriés en rouge ;
  - ...
- Si l'on essaie d'attribuer deux couleurs différentes à un même sommet, on a un cycle impair et le graphe n'est pas biparti ;
- Sinon, le graphe est biparti **et** on obtient une bipartition ($V = V_1 \cup V_2$) ;

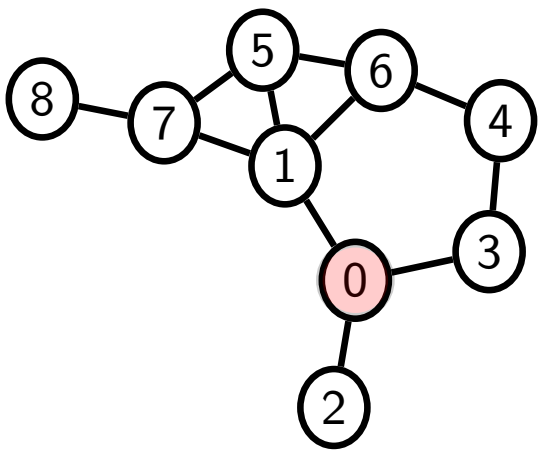
Exemple 10 (pas biparti)



Algorithme de reconnaissance des graphes bipartis

- On en déduit l'algorithme de reconnaissance suivant :
  - choisir un sommet arbitraire et le colorier (rouge) ;
  - colorier ses voisins en bleu ;
  - colorier leurs voisins pas encore coloriés en rouge ;
  - ...
- Si l'on essaie d'attribuer deux couleurs différentes à un même sommet, on a un cycle impair et le graphe n'est pas biparti ;
- Sinon, le graphe est biparti **et** on obtient une bipartition ($V = V_1 \cup V_2$) ;

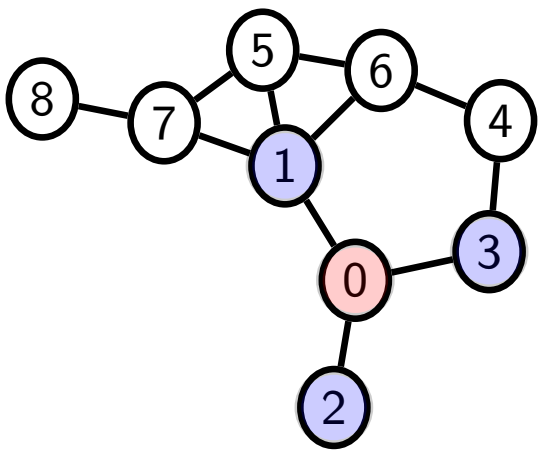
Exemple 10 (pas biparti)



Algorithme de reconnaissance des graphes bipartis

- On en déduit l'algorithme de reconnaissance suivant :
  - choisir un sommet arbitraire et le colorier (rouge) ;
  - colorier ses voisins en bleu ;
  - colorier leurs voisins pas encore coloriés en rouge ;
  - ...
- Si l'on essaie d'attribuer deux couleurs différentes à un même sommet, on a un cycle impair et le graphe n'est pas biparti ;
- Sinon, le graphe est biparti **et** on obtient une bipartition ($V = V_1 \cup V_2$) ;

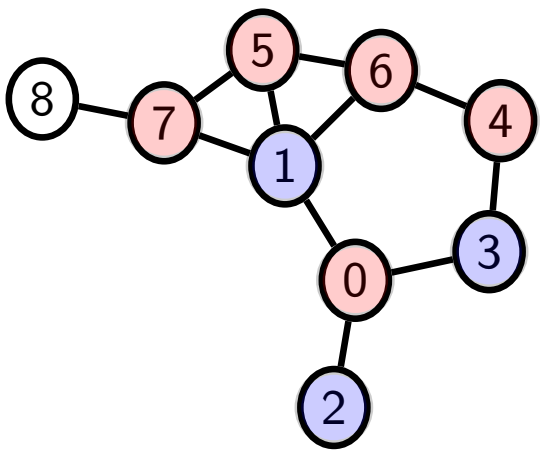
Exemple 10 (pas biparti)



Algorithme de reconnaissance des graphes bipartis

- On en déduit l'algorithme de reconnaissance suivant :
  - choisir un sommet arbitraire et le colorier (rouge) ;
  - colorier ses voisins en bleu ;
  - colorier leurs voisins pas encore coloriés en rouge ;
  - ...
- Si l'on essaie d'attribuer deux couleurs différentes à un même sommet, on a un cycle impair et le graphe n'est pas biparti ;
- Sinon, le graphe est biparti **et** on obtient une bipartition ($V = V_1 \cup V_2$) ;

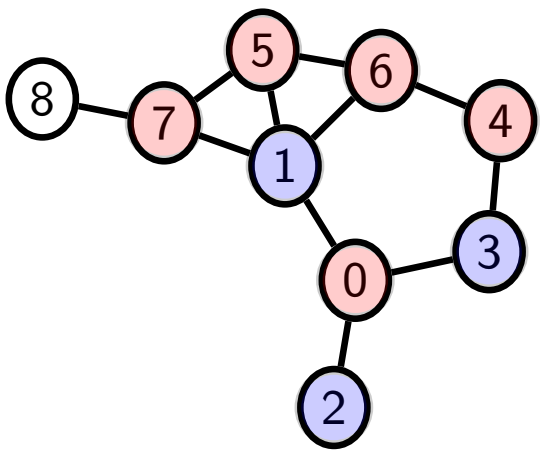
Exemple 10 (pas biparti)



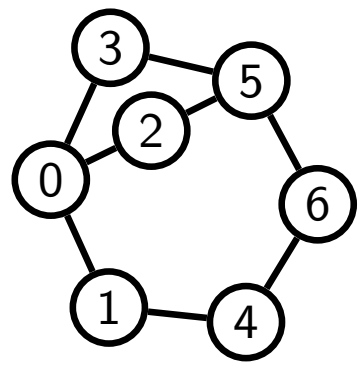
Algorithme de reconnaissance des graphes bipartis

- On en déduit l'algorithme de reconnaissance suivant :
  - choisir un sommet arbitraire et le colorier (rouge) ;
  - colorier ses voisins en bleu ;
  - colorier leurs voisins pas encore coloriés en rouge ;
  - ...
- Si l'on essaie d'attribuer deux couleurs différentes à un même sommet, on a un cycle impair et le graphe n'est pas biparti ;
- Sinon, le graphe est biparti **et** on obtient une bipartition ($V = V_1 \cup V_2$) ;

Exemple 10 (pas biparti)



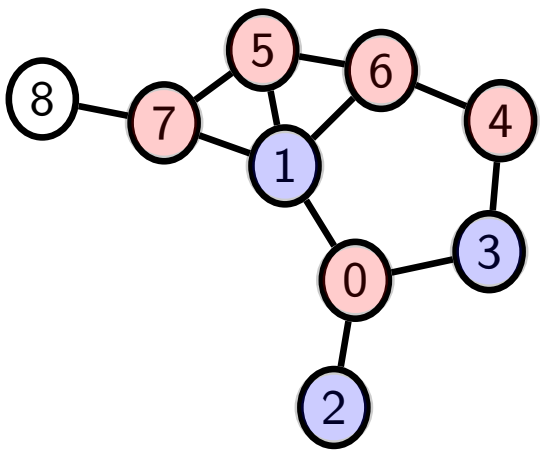
Exemple 11 (biparti)



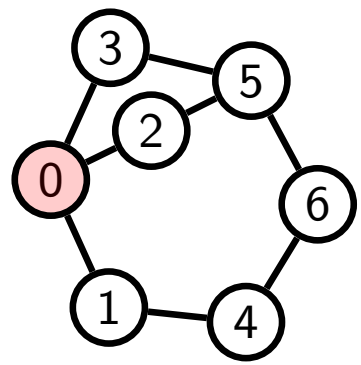
Algorithme de reconnaissance des graphes bipartis

- On en déduit l'algorithme de reconnaissance suivant :
  - choisir un sommet arbitraire et le colorier (rouge) ;
  - colorier ses voisins en bleu ;
  - colorier leurs voisins pas encore coloriés en rouge ;
  - ...
- Si l'on essaie d'attribuer deux couleurs différentes à un même sommet, on a un cycle impair et le graphe n'est pas biparti ;
- Sinon, le graphe est biparti **et** on obtient une bipartition ($V = V_1 \cup V_2$) ;

Exemple 10 (pas biparti)



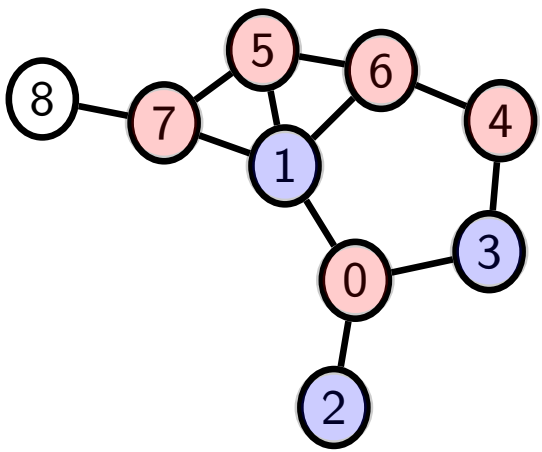
Exemple 11 (biparti)



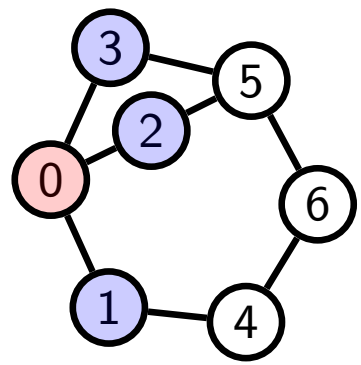
Algorithme de reconnaissance des graphes bipartis

- On en déduit l'algorithme de reconnaissance suivant :
  - choisir un sommet arbitraire et le colorier (rouge) ;
  - colorier ses voisins en bleu ;
  - colorier leurs voisins pas encore coloriés en rouge ;
  - ...
- Si l'on essaie d'attribuer deux couleurs différentes à un même sommet, on a un cycle impair et le graphe n'est pas biparti ;
- Sinon, le graphe est biparti **et** on obtient une bipartition ($V = V_1 \cup V_2$) ;

Exemple 10 (pas biparti)



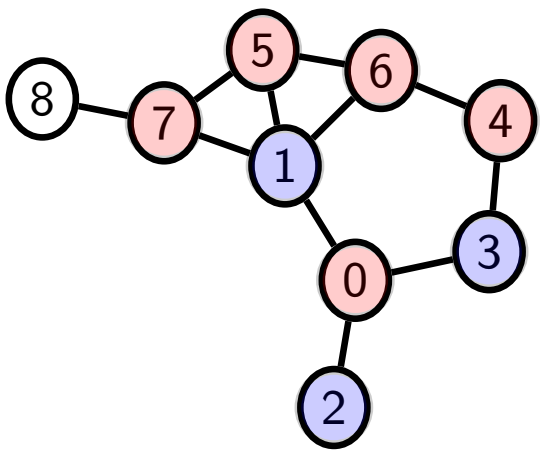
Exemple 11 (biparti)



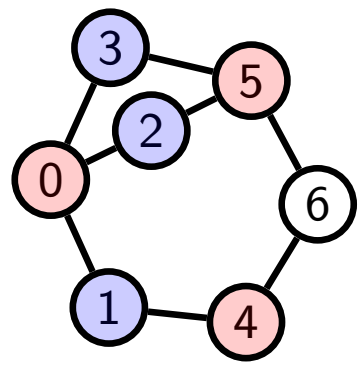
Algorithme de reconnaissance des graphes bipartis

- On en déduit l'algorithme de reconnaissance suivant :
  - choisir un sommet arbitraire et le colorier (rouge) ;
  - colorier ses voisins en bleu ;
  - colorier leurs voisins pas encore coloriés en rouge ;
  - ...
- Si l'on essaie d'attribuer deux couleurs différentes à un même sommet, on a un cycle impair et le graphe n'est pas biparti ;
- Sinon, le graphe est biparti **et** on obtient une bipartition ($V = V_1 \cup V_2$) ;

Exemple 10 (pas biparti)



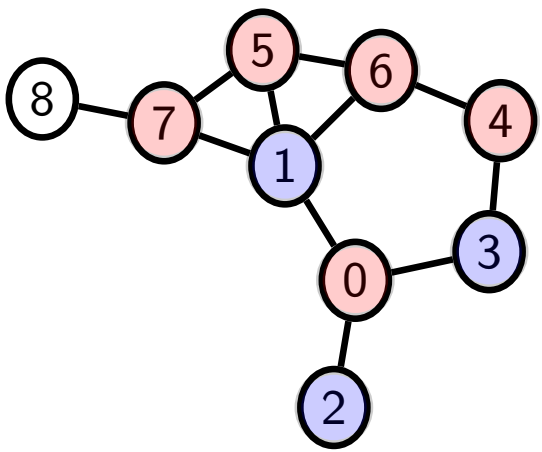
Exemple 11 (biparti)



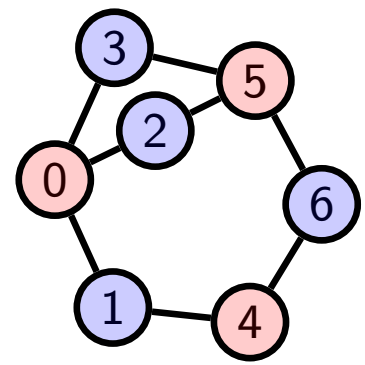
Algorithme de reconnaissance des graphes bipartis

- On en déduit l'algorithme de reconnaissance suivant :
  - choisir un sommet arbitraire et le colorier (rouge) ;
  - colorier ses voisins en bleu ;
  - colorier leurs voisins pas encore coloriés en rouge ;
  - ...
- Si l'on essaie d'attribuer deux couleurs différentes à un même sommet, on a un cycle impair et le graphe n'est pas biparti ;
- Sinon, le graphe est biparti **et** on obtient une bipartition ($V = V_1 \cup V_2$) ;

Exemple 10 (pas biparti)



Exemple 11 (biparti)



L'algorithme

Algorithme 6 : EstBiparti(G)

Entrées : un graphe connexe G.
Sortie : VRAI si G est biparti, FAUX sinon.

```
1  si G.nombre_sommets() = 0 alors renvoyer VRAI;
2  couleurs ← tableau(G.nombre_sommets(), −1);
3  catégorie ← VRAI;
4  départ ← sommet arbitraire de G;
5  a_traiter ← file();
6  a_traiter.enfiler(départ);
7  couleurs[départ] ← catégorie;
8  tant que a_traiter.pas_vide() faire
9      sommet ← a_traiter.défiler();
10     catégorie ← ¬ couleurs[sommet];
11     pour chaque voisin dans G.voisins(sommet) faire
12         si couleurs[voisin] = −1 alors
13             couleurs[voisin] ← catégorie;
14             a_traiter.enfiler(voisin);
15         sinon si couleurs[voisin] ≠ catégorie alors renvoyer FAUX ;
16 renvoyer VRAI;
```

Détection de cycles

- On peut utiliser l'algorithme de reconnaissance de graphes bipartis pour savoir si le graphe contient un cycle impair ;

Détection de cycles

- On peut utiliser l'algorithme de reconnaissance de graphes bipartis pour savoir si le graphe contient un cycle impair ;
- Comment savoir si un graphe contient un cycle quelconque ? Deux options :

Détection de cycles

- On peut utiliser l'algorithme de reconnaissance de graphes bipartis pour savoir si le graphe contient un cycle impair ;
- Comment savoir si un graphe contient un cycle quelconque ? Deux options :
  - si on veut juste répondre "oui" ou "non" : comparer $|E|$ à $|V|$;

Détection de cycles

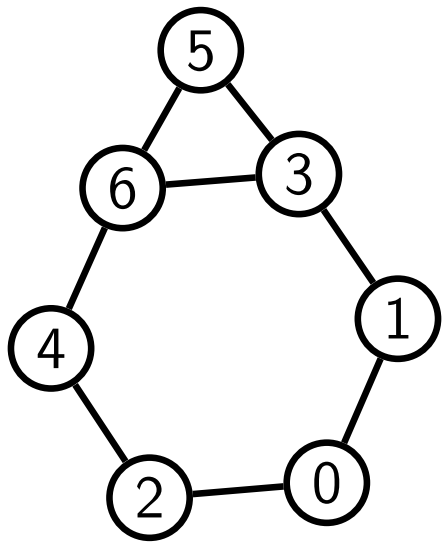
- On peut utiliser l'algorithme de reconnaissance de graphes bipartis pour savoir si le graphe contient un cycle impair ;
- Comment savoir si un graphe contient un cycle quelconque ? Deux options :
  - si on veut juste répondre "oui" ou "non" : comparer $|E|$ à $|V|$;
  - si on veut renvoyer un cycle explicite : parcourir le graphe, et repérer si l'on retombe sur un sommet . . . déjà visité ?

Détection de cycles

- On peut utiliser l'algorithme de reconnaissance de graphes bipartis pour savoir si le graphe contient un cycle impair ;
- Comment savoir si un graphe contient un cycle quelconque ? Deux options :
  - si on veut juste répondre "oui" ou "non" : comparer $|E|$ à $|V|$;
  - si on veut renvoyer un cycle explicite : parcourir le graphe, et repérer si l'on retombe sur un sommet . . . déjà visité ?
  - non : sur un sommet **adopté** ;

Détection de cycles

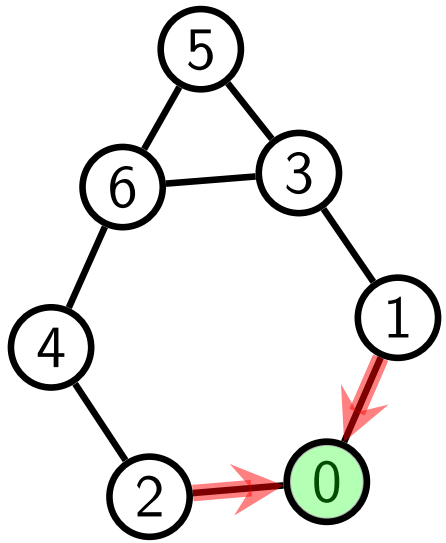
Exemple 12



Pour obtenir le cycle, on combine l'arête "fautive" et celles du chemin alternatif.

Détection de cycles

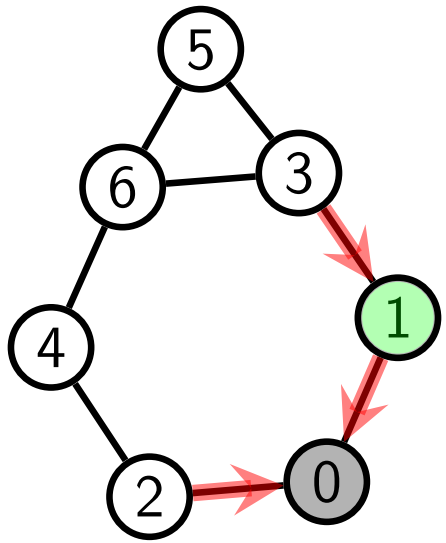
Exemple 12



Pour obtenir le cycle, on combine l'arête "fautive" et celles du chemin alternatif.

Détection de cycles

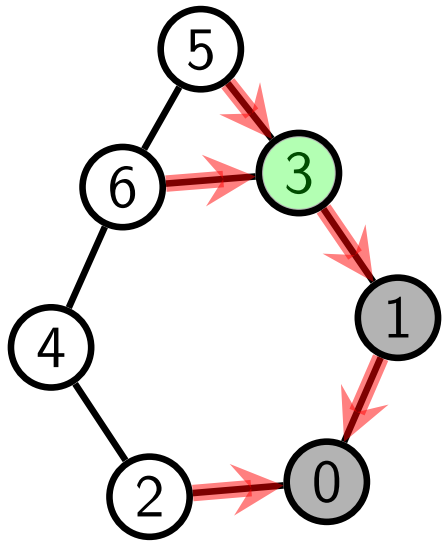
Exemple 12



Pour obtenir le cycle, on combine l'arête "fautive" et celles du chemin alternatif.

Détection de cycles

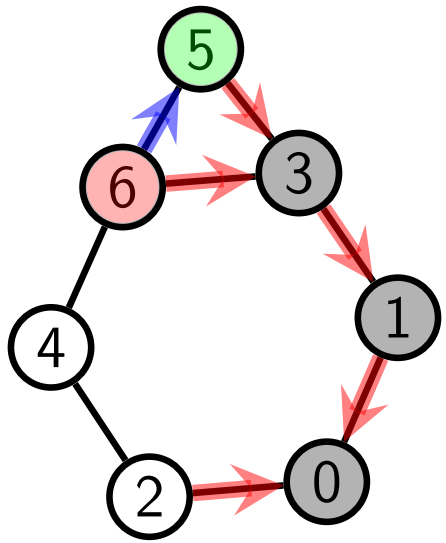
Exemple 12



Pour obtenir le cycle, on combine l'arête "fautive" et celles du chemin alternatif.

Détection de cycles

Exemple 12



Pour obtenir le cycle, on combine l'arête "fautive" et celles du chemin alternatif.

Clusters en dot

On peut demander à Graphviz d'encadrer des sous-graphes, appelés *clusters* :

Exemple 13

```
graph G {
    # ...
    subgraph cluster_moncluster {
        # définition des sommets et / ou des arêtes du cluster
        # ...
    }
}
```

Attention :

- on **doit** utiliser le préfixe `cluster_` après `subgraph` pour nommer le sous-graphe ;
- seuls les programmes `dot` et `fdp` prennent en compte les clusters ;