Algorithmique des graphes

5 — Parcours des graphes non-orientés et applications

Marie-Pierre Béal (Cours d'Anthony Labarre)

Université
Gustave Eiffel

Rappels : graphe non-orienté

Définition 1

Un **graphe (non-orienté)** est un couple $G = (V, E)$, où :

- V est un ensemble de **sommets** ;
- E un ensemble de paires d'éléments de V appelées **arêtes**.

On utilisera aussi les notations $V(G)$ et $E(G)$ pour bien distinguer le graphe G d'un autre graphe.

Rappels : graphe non-orienté

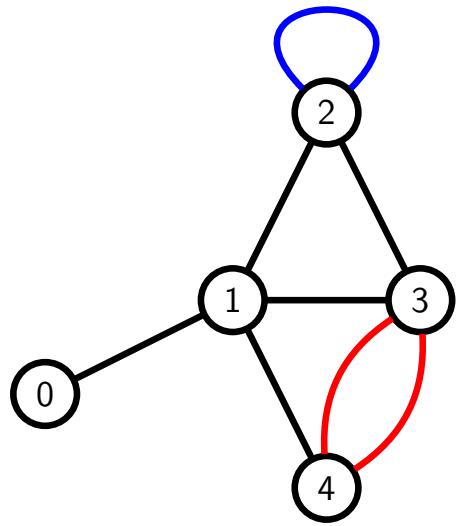
Définition 1

Un **graphe (non-orienté)** est un couple $G = (V, E)$, où :

- V est un ensemble de **sommets** ;
- E un ensemble de paires d'éléments de V appelées **arêtes**.

On utilisera aussi les notations $V(G)$ et $E(G)$ pour bien distinguer le graphe G d'un autre graphe.

Exemple 1



Rappels : graphe non-orienté

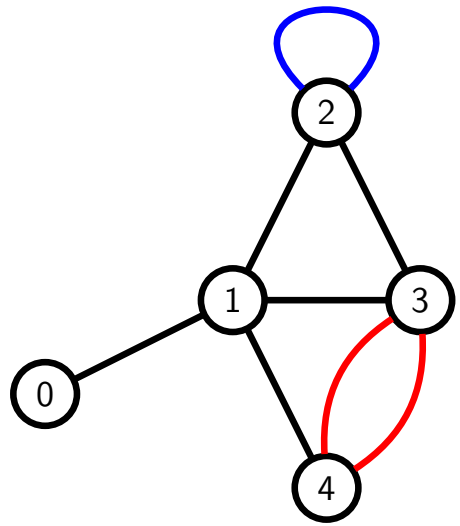
Définition 1

Un **graphe (non-orienté)** est un couple $G = (V, E)$, où :

- V est un ensemble de **sommets** ;
- E un ensemble de paires d'éléments de V appelées **arêtes**.

On utilisera aussi les notations $V(G)$ et $E(G)$ pour bien distinguer le graphe G d'un autre graphe.
G est **simple** s'il ne contient ni **arêtes parallèles** ni **boucles**.

Exemple 1



Rappels : graphe non-orienté

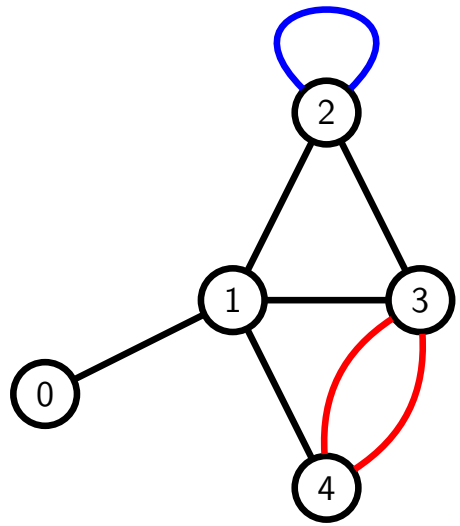
Définition 1

Un **graphe (non-orienté)** est un couple $G = (V, E)$, où :

- V est un ensemble de **sommets** ;
- E un ensemble de paires d'éléments de V appelées **arêtes**.

On utilisera aussi les notations $V(G)$ et $E(G)$ pour bien distinguer le graphe G d'un autre graphe.
G est **simple** s'il ne contient ni **arêtes parallèles** ni **boucles**.

Exemple 1



- Une arête $\{u, v\}$ relie deux sommets **adjacents** ou **voisins** ; elle est **incidente** à u et v, qui sont ses **extrémités** ;

Rappels : graphe non-orienté

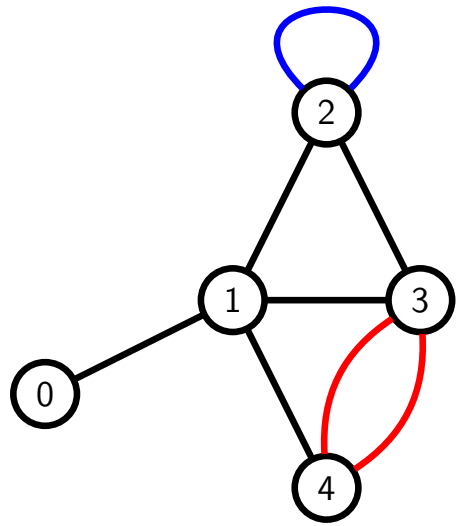
Définition 1

Un **graphe (non-orienté)** est un couple $G = (V, E)$, où :

- V est un ensemble de **sommets** ;
- E un ensemble de paires d'éléments de V appelées **arêtes**.

On utilisera aussi les notations $V(G)$ et $E(G)$ pour bien distinguer le graphe G d'un autre graphe.
G est **simple** s'il ne contient ni **arêtes parallèles** ni **boucles**.

Exemple 1



- Une arête $\{u, v\}$ relie deux sommets **adjacents** ou **voisins** ; elle est **incidente** à u et v, qui sont ses **extrémités** ;
- Le **voisinage** d'un sommet v est l'ensemble de ses voisins :

$$N_G(v) = \{u \mid \{u, v\} \in E(G)\}$$

Rappels : graphe non-orienté

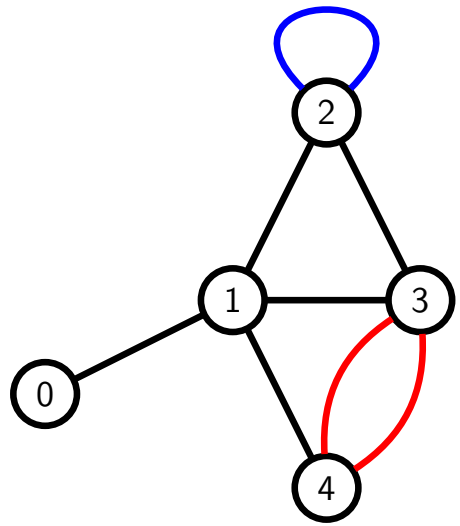
Définition 1

Un **graphe (non-orienté)** est un couple $G = (V, E)$, où :

- V est un ensemble de **sommets** ;
- E un ensemble de paires d'éléments de V appelées **arêtes**.

On utilisera aussi les notations $V(G)$ et $E(G)$ pour bien distinguer le graphe G d'un autre graphe.
G est **simple** s'il ne contient ni **arêtes parallèles** ni **boucles**.

Exemple 1



- Une arête $\{u, v\}$ relie deux sommets **adjacents** ou **voisins** ; elle est **incidente** à u et v, qui sont ses **extrémités** ;
- Le **voisinage** d'un sommet v est l'ensemble de ses voisins :

$$N_G(v) = \{u \mid \{u, v\} \in E(G)\}$$

- Le **degré** du sommet v est la taille de son voisinage, noté $\deg_G(v)$;

Matrice d'adjacence

Définition 2

La **matrice d'adjacence** $A(G)$ du graphe $G = (V, E)$ contient un 1 en ligne i et en colonne j si l'arête $\{i, j\}$ existe, et un 0 sinon :

$$A_{ij}(G) = \begin{cases} 1 & \text{si } \{i,j\} \in E, \\ 0 & \text{sinon.} \end{cases}$$

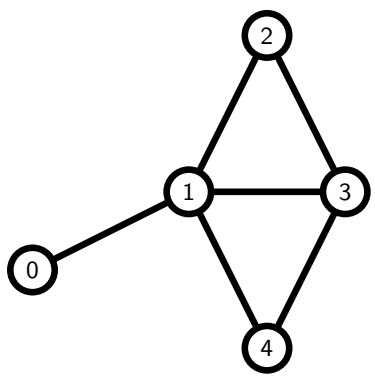
Matrice d'adjacence

Définition 2

La **matrice d'adjacence** $A(G)$ du graphe $G = (V, E)$ contient un 1 en ligne i et en colonne j si l'arête $\{i, j\}$ existe, et un 0 sinon :

$$A_{ij}(G) = \begin{cases} 1 & \text{si } \{i,j\} \in E, \\ 0 & \text{sinon.} \end{cases}$$

Exemple 2



	0	1	2	3	4
0	0	1	0	0	0
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	0	1	0	1	0

	0	1	2	3	4
0	0				
1	1	0			
2	0	1	0		
3	0	1	1	0	
4	0	1	0	1	0

010010011001010

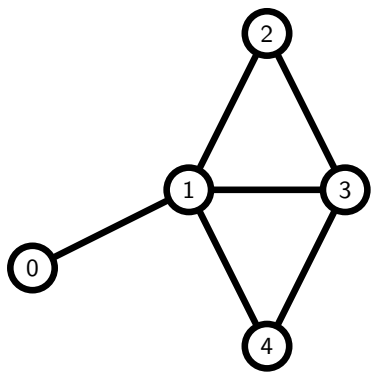
Matrice d'adjacence

Définition 2

La **matrice d'adjacence** $A(G)$ du graphe $G = (V, E)$ contient un 1 en ligne i et en colonne j si l'arête $\{i, j\}$ existe, et un 0 sinon :

$$A_{ij}(G) = \begin{cases} 1 & \text{si } \{i,j\} \in E, \\ 0 & \text{sinon.} \end{cases}$$

Exemple 2



	0	1	2	3	4
0	0	1	0	0	0
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	0	1	0	1	0

	0	1	2	3	4
0	0				
1	1	0			
2	0	1	0		
3	0	1	1	0	
4	0	1	0	1	0

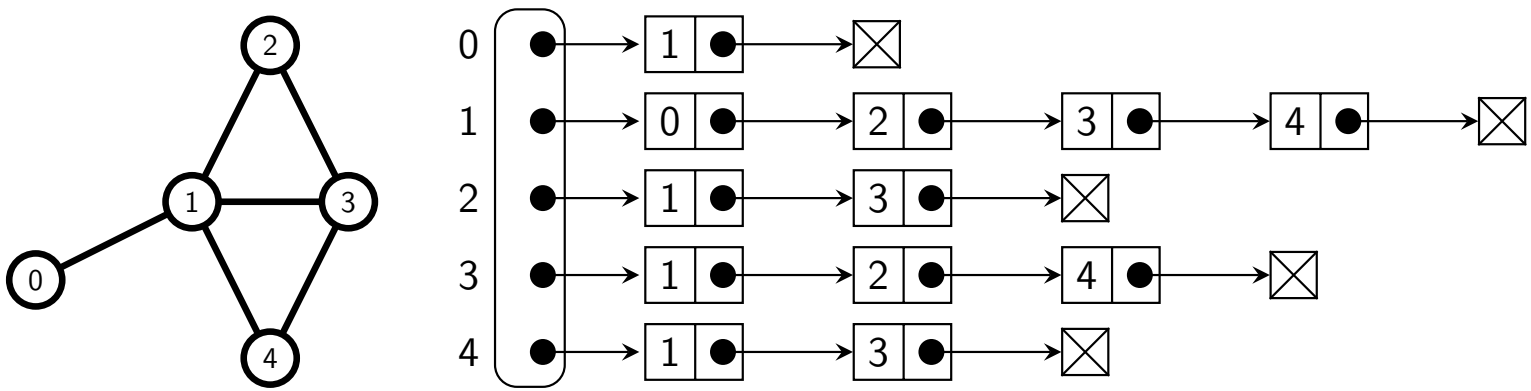
010010011001010

Les connexions sont symétriques, donc $A(G)$ aussi, et on pourrait donc n'en garder que la moitié. S'il n'y a pas de poids, une chaîne binaire suffit.

Listes d'adjacence

Les **listes d'adjacence** encodent, pour chaque sommet du graphe $G = (V, E)$, la liste des voisins de ce sommet.

Exemple 3

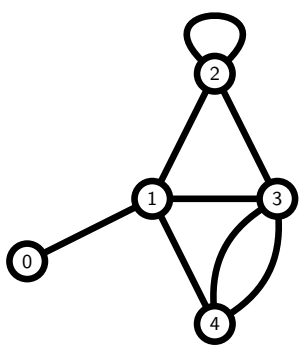


Sous-graphes

On s'intéressera régulièrement à un "morceau" particulier d'un graphe G.

- Un **sous-graphe** d'un graphe $G = (V, E)$ est un graphe $H = (V', E')$ avec $V' \subseteq V$ et $E' \subseteq E$.

Exemple 4



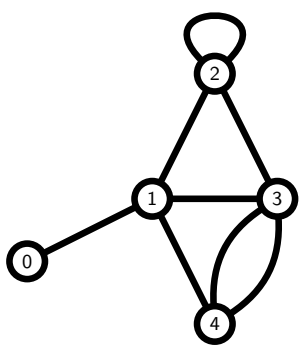
G

Sous-graphes

On s'intéressera régulièrement à un "morceau" particulier d'un graphe G.

- Un **sous-graphe** d'un graphe $G = (V, E)$ est un graphe $H = (V', E')$ avec $V' \subseteq V$ et $E' \subseteq E$.
- H est **induit** par V' si E' contient toutes les arêtes de E reliant deux sommets de V' dans G ; on écrit alors $H = G[V']$.

Exemple 4



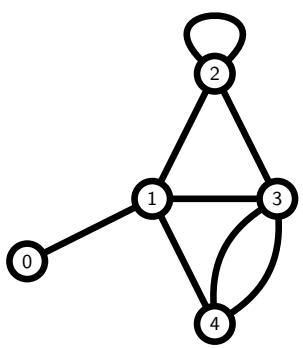
G

Sous-graphes

On s'intéressera régulièrement à un "morceau" particulier d'un graphe G.

- Un **sous-graphe** d'un graphe $G = (V, E)$ est un graphe $H = (V', E')$ avec $V' \subseteq V$ et $E' \subseteq E$.
- H est **induit** par V' si E' contient toutes les arêtes de E reliant deux sommets de V' dans G ; on écrit alors $H = G[V']$.
- H est **induit** par F' si V' contient seulement les sommets de V reliés par F' dans G ; on écrit alors $H = G[F']$.

Exemple 4



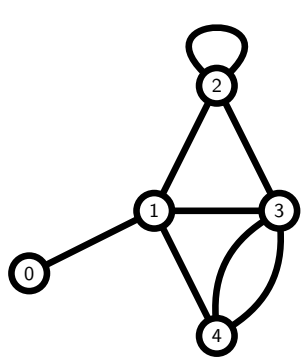
G

Sous-graphes

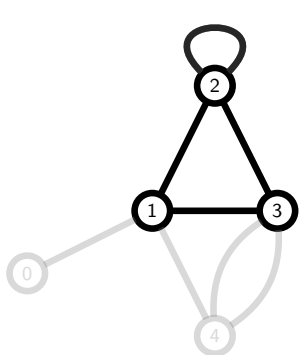
On s'intéressera régulièrement à un "morceau" particulier d'un graphe G.

- Un **sous-graphe** d'un graphe $G = (V, E)$ est un graphe $H = (V', E')$ avec $V' \subseteq V$ et $E' \subseteq E$.
- H est **induit** par V' si E' contient toutes les arêtes de E reliant deux sommets de V' dans G ; on écrit alors $H = G[V']$.
- H est **induit** par F' si V' contient seulement les sommets de V reliés par F' dans G ; on écrit alors $H = G[F']$.

Exemple 4



G



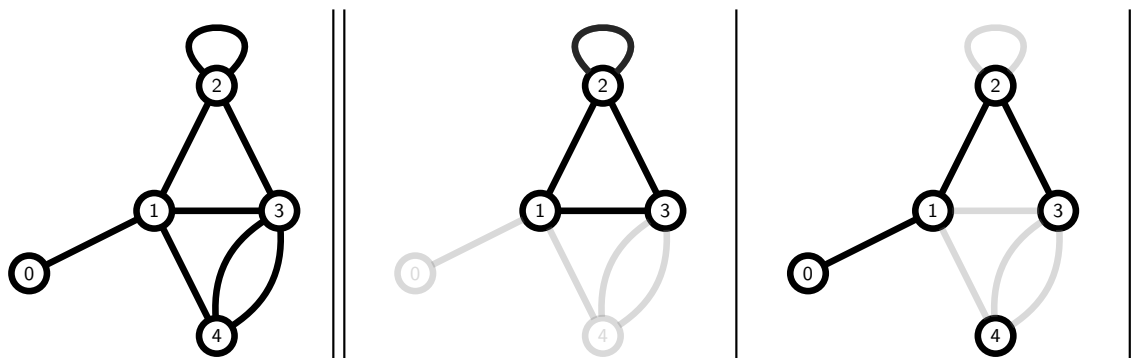
sous-graphe induit

Sous-graphes

On s'intéressera régulièrement à un "morceau" particulier d'un graphe G.

- Un **sous-graphe** d'un graphe $G = (V, E)$ est un graphe $H = (V', E')$ avec $V' \subseteq V$ et $E' \subseteq E$.
- H est **induit** par V' si E' contient toutes les arêtes de E reliant deux sommets de V' dans G ; on écrit alors $H = G[V']$.
- H est **induit** par F' si V' contient seulement les sommets de V reliés par F' dans G ; on écrit alors $H = G[F']$.

Exemple 4



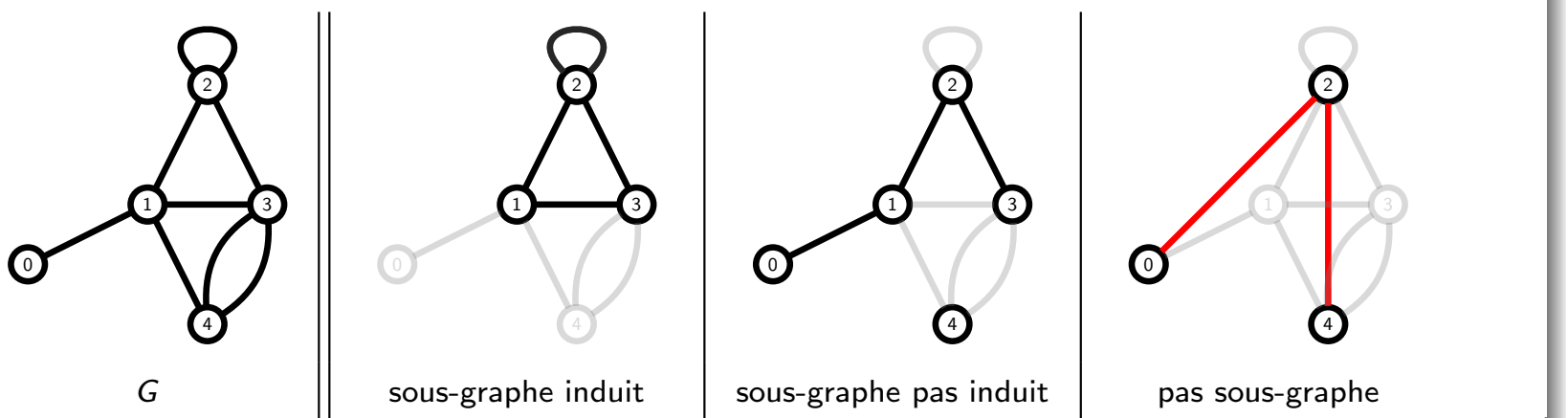
G | sous-graphe induit | sous-graphe pas induit

Sous-graphes

On s'intéressera régulièrement à un "morceau" particulier d'un graphe G.

- Un **sous-graphe** d'un graphe $G = (V, E)$ est un graphe $H = (V', E')$ avec $V' \subseteq V$ et $E' \subseteq E$.
- H est **induit** par V' si E' contient toutes les arêtes de E reliant deux sommets de V' dans G ; on écrit alors $H = G[V']$.
- H est **induit** par F' si V' contient seulement les sommets de V reliés par F' dans G ; on écrit alors $H = G[F']$.

Exemple 4



Graphes fréquents

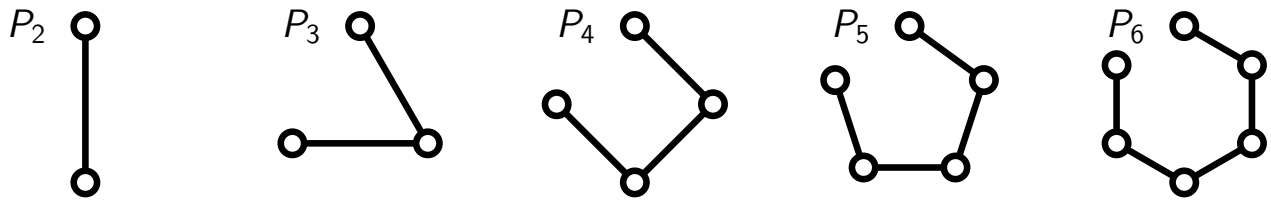
Certains graphes reviennent souvent dans les applications :

- un **chemin ou chaîne** dans un graphe G est une séquence $P = (u_0, u_1, u_2, \ldots, u_{p-1})$ de sommets où $\{u_i, u_{i+1}\} \in E(G)$ pour $0 \leq i \leq p-2$;

Graphes fréquents

Certains graphes reviennent souvent dans les applications :

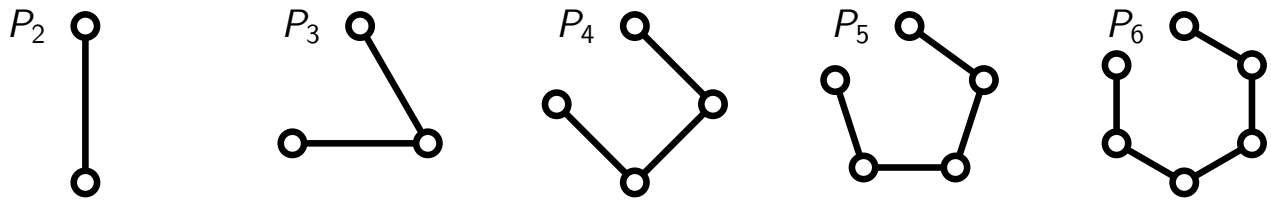
- un **chemin ou chaîne** dans un graphe G est une séquence $P = (u_0, u_1, u_2, \ldots, u_{p-1})$ de sommets où $\{u_i, u_{i+1}\} \in E(G)$ pour $0 \leq i \leq p-2$;



Graphes fréquents

Certains graphes reviennent souvent dans les applications :

- un **chemin ou chaîne** dans un graphe G est une séquence $P = (u_0, u_1, u_2, \ldots, u_{p-1})$ de sommets où $\{u_i, u_{i+1}\} \in E(G)$ pour $0 \leq i \leq p-2$;

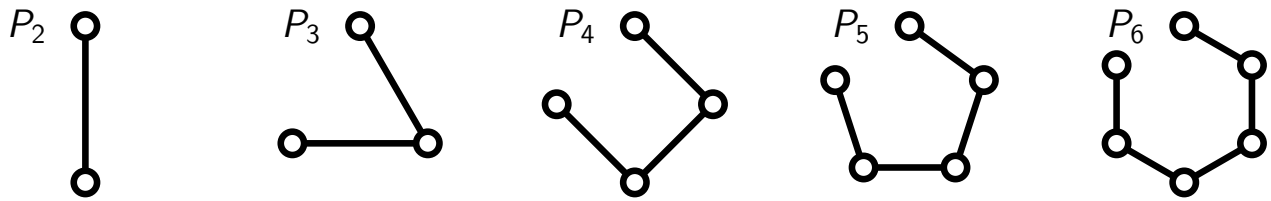


- un **cycle élémentaire** dans un graphe G est un chemin dont toutes les arêtes sont distinctes et dont les deux extrémités sont identiques.

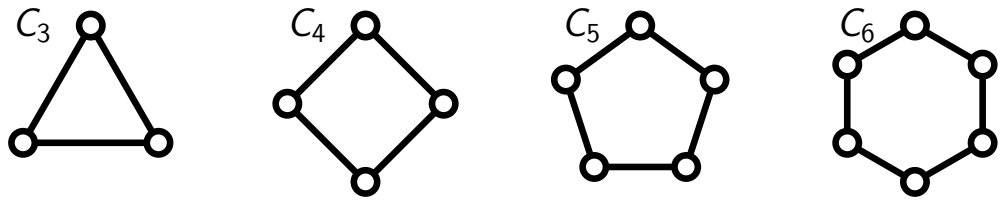
Graphes fréquents

Certains graphes reviennent souvent dans les applications :

- un **chemin ou chaîne** dans un graphe G est une séquence $P = (u_0, u_1, u_2, \ldots, u_{p-1})$ de sommets où $\{u_i, u_{i+1}\} \in E(G)$ pour $0 \leq i \leq p-2$;



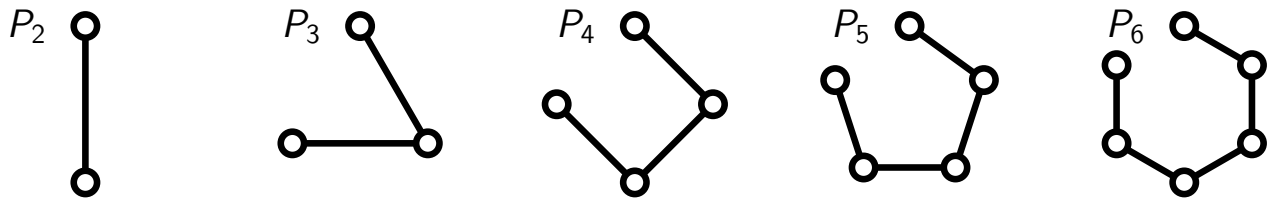
- un **cycle élémentaire** dans un graphe G est un chemin dont toutes les arêtes sont distinctes et dont les deux extrémités sont identiques.



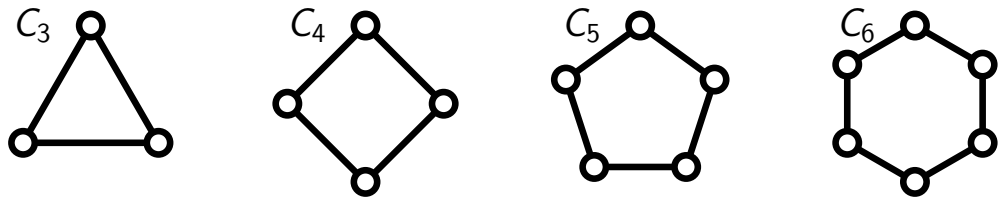
Graphes fréquents

Certains graphes reviennent souvent dans les applications :

- un **chemin ou chaîne** dans un graphe G est une séquence $P = (u_0, u_1, u_2, \ldots, u_{p-1})$ de sommets où $\{u_i, u_{i+1}\} \in E(G)$ pour $0 \leq i \leq p-2$;



- un **cycle élémentaire** dans un graphe G est un chemin dont toutes les arêtes sont distinctes et dont les deux extrémités sont identiques.

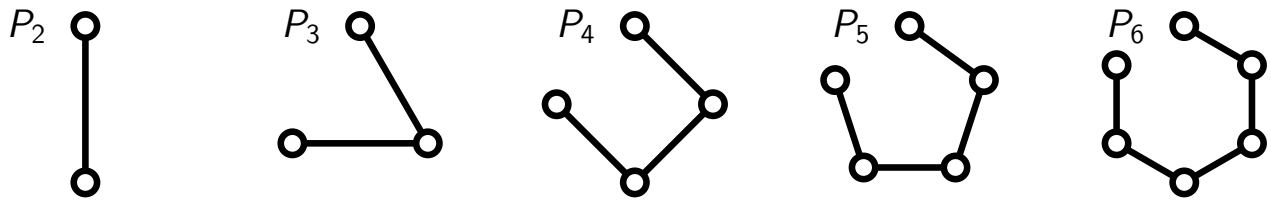


- enfin, on dit d'un graphe $G = (V, E)$ qu'il est **complet** si $E = V \times V$, c'est-à-dire que chaque sommet est adjacent à tous les autres.

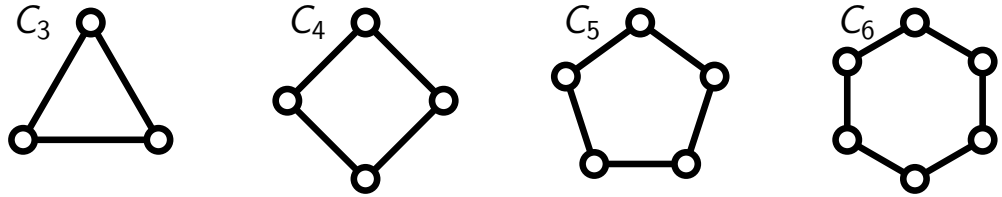
Graphes fréquents

Certains graphes reviennent souvent dans les applications :

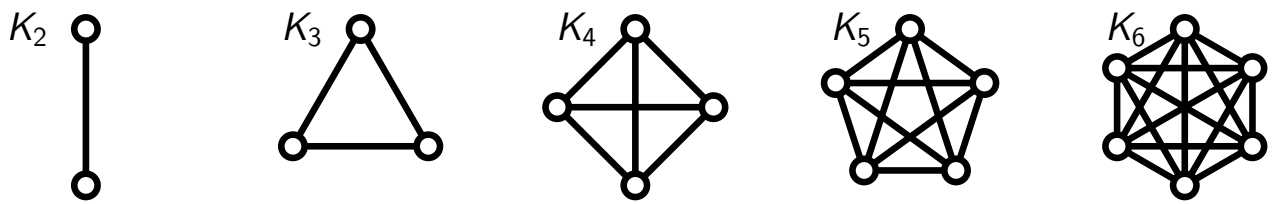
- un **chemin ou chaîne** dans un graphe G est une séquence $P = (u_0, u_1, u_2, \ldots, u_{p-1})$ de sommets où $\{u_i, u_{i+1}\} \in E(G)$ pour $0 \leq i \leq p-2$;



- un **cycle élémentaire** dans un graphe G est un chemin dont toutes les arêtes sont distinctes et dont les deux extrémités sont identiques.



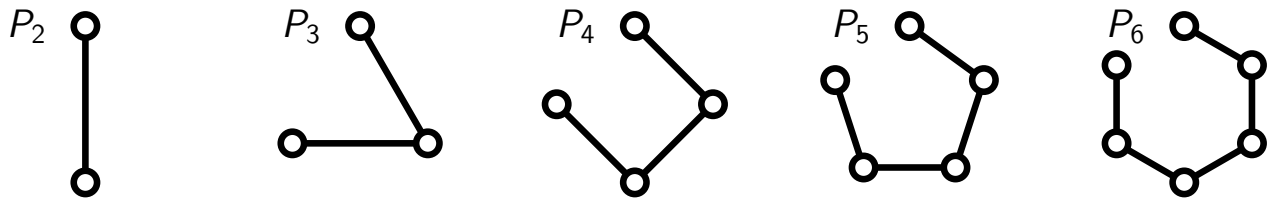
- enfin, on dit d'un graphe $G = (V, E)$ qu'il est **complet** si $E = V \times V$, c'est-à-dire que chaque sommet est adjacent à tous les autres.



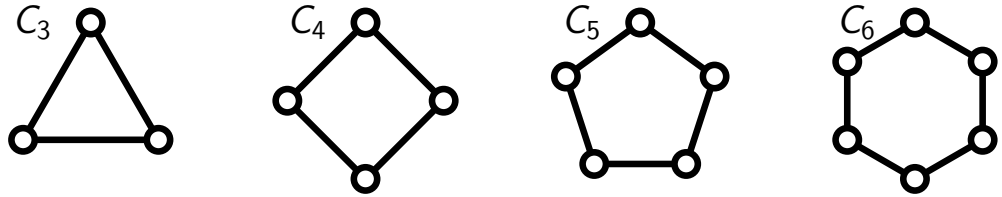
Graphes fréquents

Certains graphes reviennent souvent dans les applications :

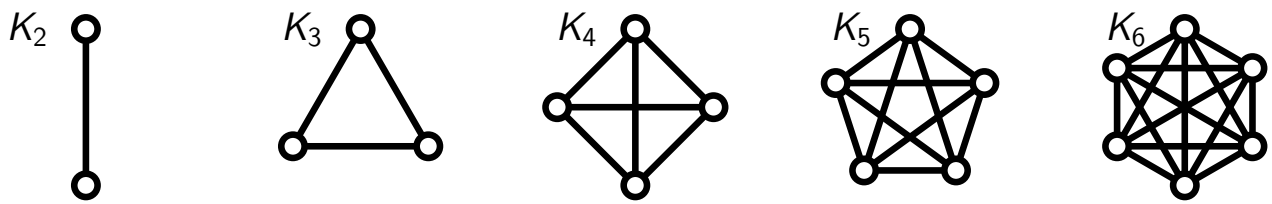
- un **chemin ou chaîne** dans un graphe G est une séquence $P = (u_0, u_1, u_2, \ldots, u_{p-1})$ de sommets où $\{u_i, u_{i+1}\} \in E(G)$ pour $0 \leq i \leq p-2$;



- un **cycle élémentaire** dans un graphe G est un chemin dont toutes les arêtes sont distinctes et dont les deux extrémités sont identiques.



- enfin, on dit d'un graphe $G = (V, E)$ qu'il est **complet** si $E = V \times V$, c'est-à-dire que chaque sommet est adjacent à tous les autres.



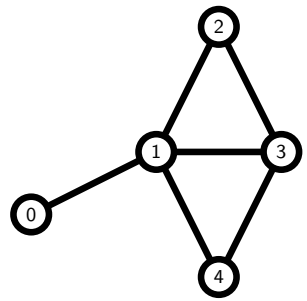
La **longueur** d'un cycle ou d'un chemin est son nombre d'arêtes.

Degrés et taille

La **taille** d'un graphe $G = (V, E)$ est son nombre d'arêtes $|E|$. On peut la calculer à l'aide de la relation suivante :

$$\sum_{v \in V} \deg(v) = 2|E|. \qquad (1)$$

Exemple 5

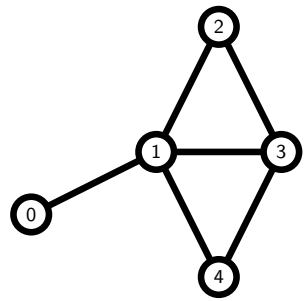


Degrés et taille

La **taille** d'un graphe $G = (V, E)$ est son nombre d'arêtes $|E|$. On peut la calculer à l'aide de la relation suivante :

$$\sum_{v \in V} \deg(v) = 2|E|. \qquad (1)$$

Exemple 5



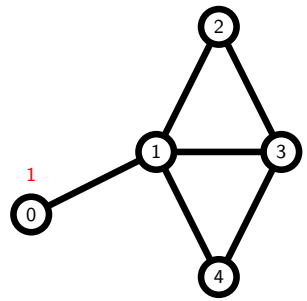
En effet, si l'on parcourt les sommets du graphe en sélectionnant chaque arête incidente à chacun des sommets, on aura sélectionné chaque arête exactement deux fois.

Degrés et taille

La **taille** d'un graphe $G = (V, E)$ est son nombre d'arêtes $|E|$. On peut la calculer à l'aide de la relation suivante :

$$\sum_{v \in V} \deg(v) = 2|E|. \qquad (1)$$

Exemple 5



En effet, si l'on parcourt les sommets du graphe en sélectionnant chaque arête incidente à chacun des sommets, on aura sélectionné chaque arête exactement deux fois.

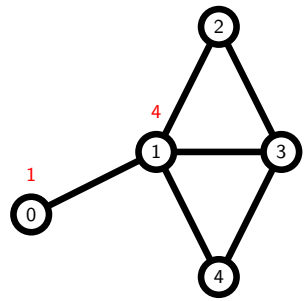
K_n contient le nombre maximum d'arêtes, à savoir $\binom{n}{2} = n(n-1)/2$.

Degrés et taille

La **taille** d'un graphe $G = (V, E)$ est son nombre d'arêtes $|E|$. On peut la calculer à l'aide de la relation suivante :

$$\sum_{v \in V} \deg(v) = 2|E|. \qquad (1)$$

Exemple 5



En effet, si l'on parcourt les sommets du graphe en sélectionnant chaque arête incidente à chacun des sommets, on aura sélectionné chaque arête exactement deux fois.

K_n contient le nombre maximum d'arêtes, à savoir $\binom{n}{2} = n(n-1)/2$.

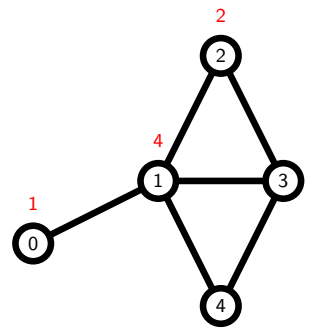
Donc pour tout graphe $G = (V, E)$: $|E| = O(|V|^2)$, et $\log |E| = O(\log |V|)$.

Degrés et taille

La **taille** d'un graphe $G = (V, E)$ est son nombre d'arêtes $|E|$. On peut la calculer à l'aide de la relation suivante :

$$\sum_{v \in V} \deg(v) = 2|E|. \qquad (1)$$

Exemple 5



En effet, si l'on parcourt les sommets du graphe en sélectionnant chaque arête incidente à chacun des sommets, on aura sélectionné chaque arête exactement deux fois.

K_n contient le nombre maximum d'arêtes, à savoir $\binom{n}{2} = n(n-1)/2$.

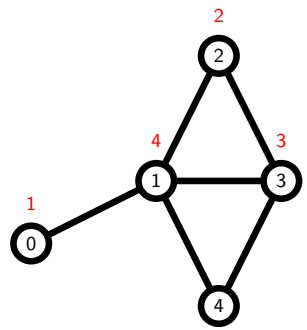
Donc pour tout graphe $G = (V, E)$: $|E| = O(|V|^2)$, et $\log |E| = O(\log |V|)$.

Degrés et taille

La **taille** d'un graphe $G = (V, E)$ est son nombre d'arêtes $|E|$. On peut la calculer à l'aide de la relation suivante :

$$\sum_{v \in V} \deg(v) = 2|E|. \qquad (1)$$

Exemple 5



En effet, si l'on parcourt les sommets du graphe en sélectionnant chaque arête incidente à chacun des sommets, on aura sélectionné chaque arête exactement deux fois.

K_n contient le nombre maximum d'arêtes, à savoir $\binom{n}{2} = n(n-1)/2$.

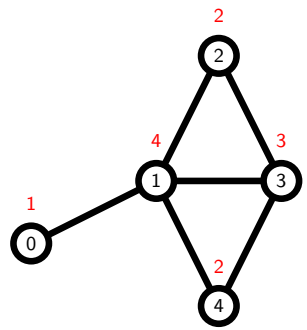
Donc pour tout graphe $G = (V, E)$: $|E| = O(|V|^2)$, et $\log |E| = O(\log |V|)$.

Degrés et taille

La **taille** d'un graphe $G = (V, E)$ est son nombre d'arêtes $|E|$. On peut la calculer à l'aide de la relation suivante :

$$\sum_{v \in V} \deg(v) = 2|E|. \qquad (1)$$

Exemple 5



En effet, si l'on parcourt les sommets du graphe en sélectionnant chaque arête incidente à chacun des sommets, on aura sélectionné chaque arête exactement deux fois.

K_n contient le nombre maximum d'arêtes, à savoir $\binom{n}{2} = n(n-1)/2$.

Donc pour tout graphe $G = (V, E)$: $|E| = O(|V|^2)$, et $\log |E| = O(\log |V|)$.

Parcours de graphes non-orientés

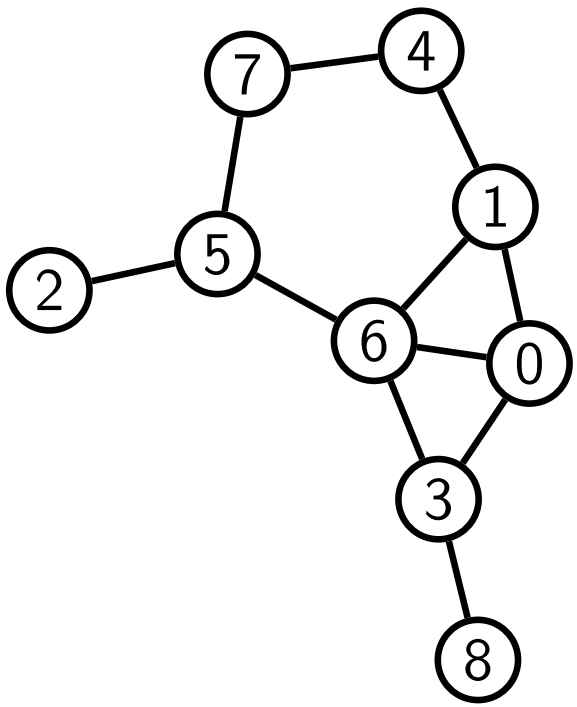
La parcours des graphes non-orientés est similaire à celui des graphes orientés.

Conventions : on suppose que :

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

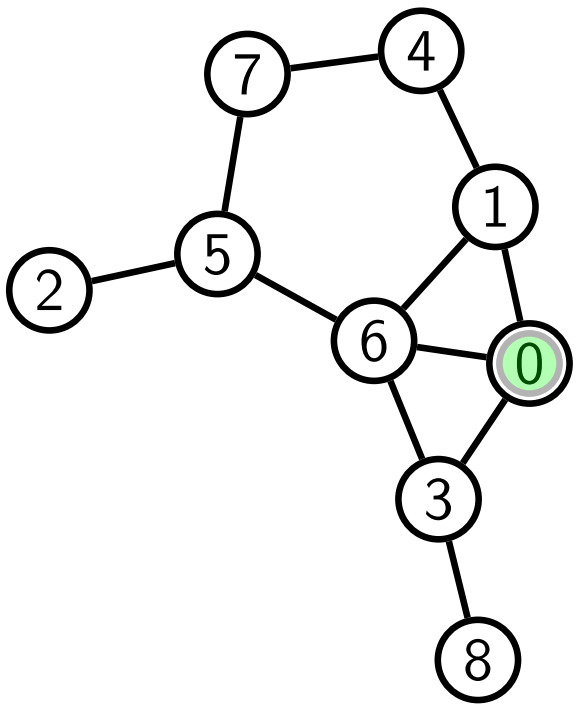
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



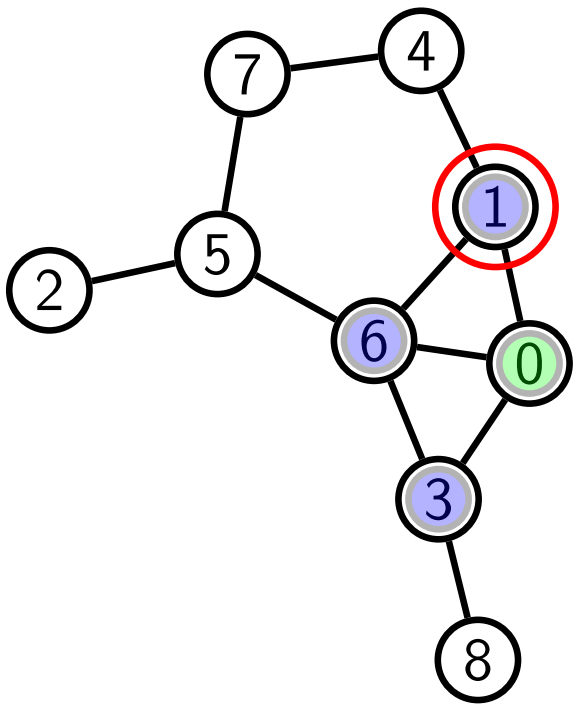
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



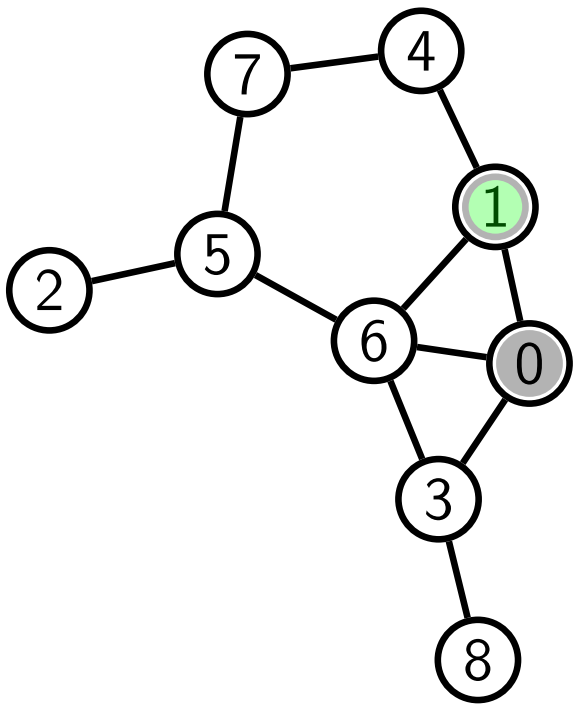
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



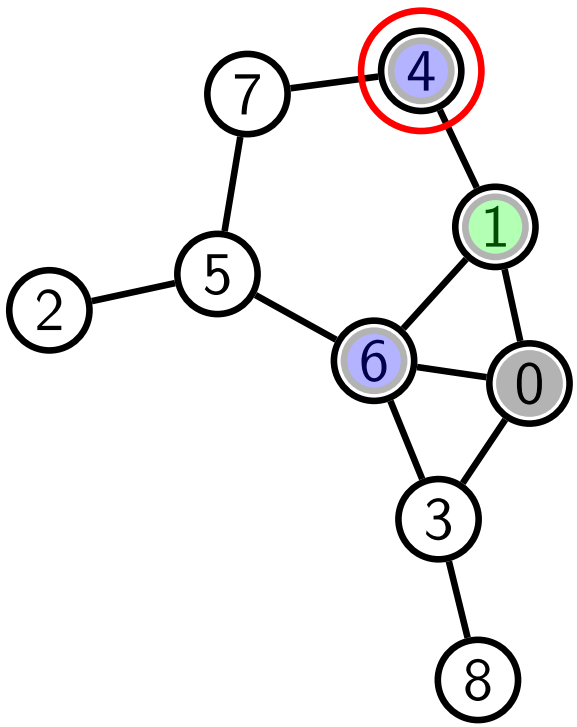
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



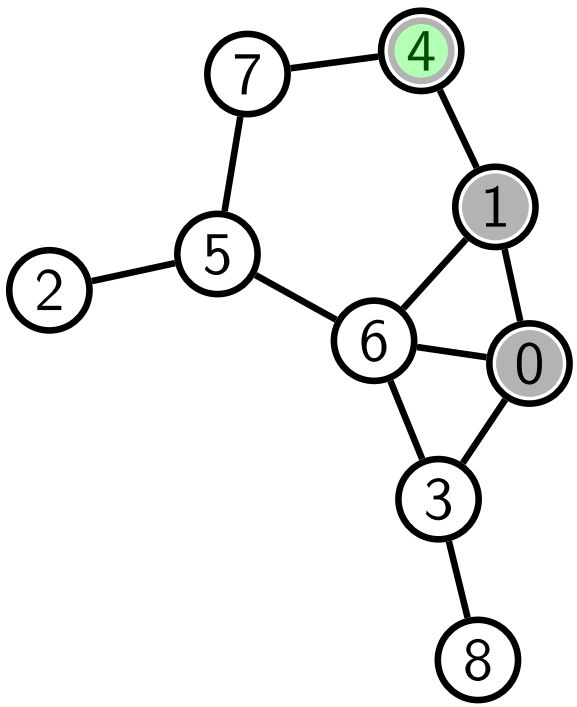
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



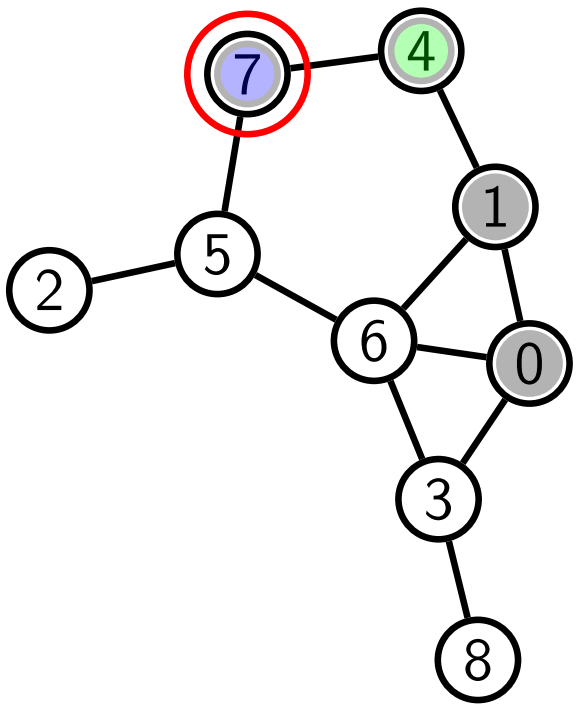
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



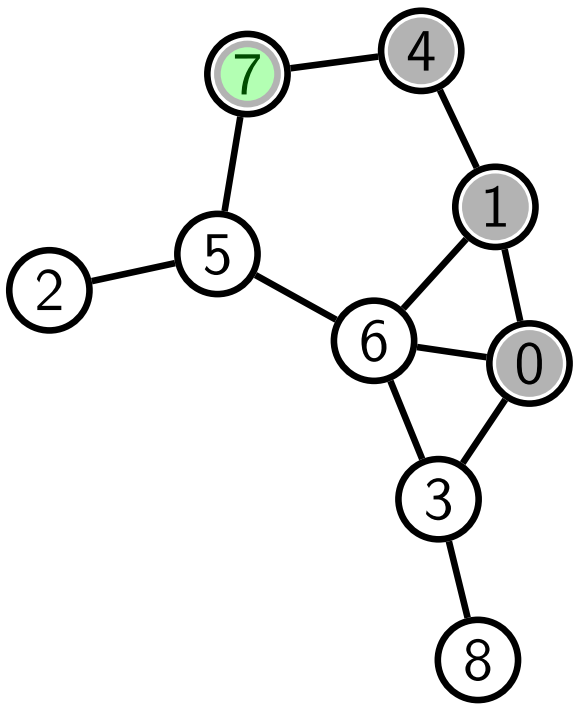
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



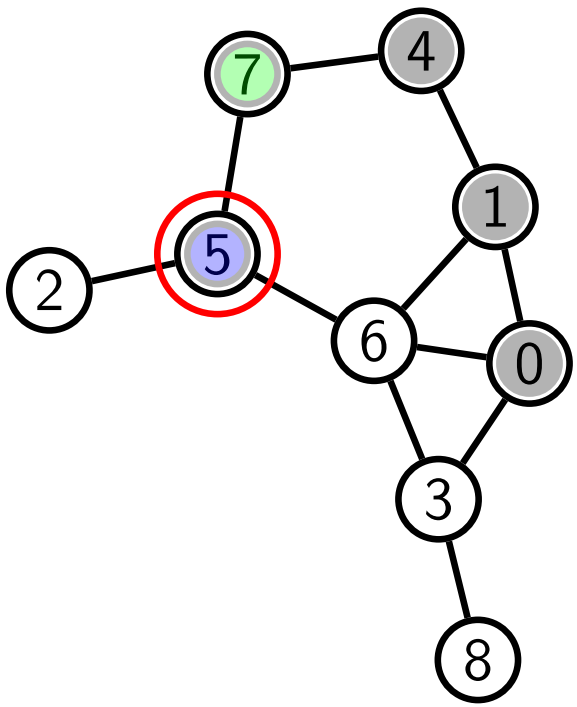
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



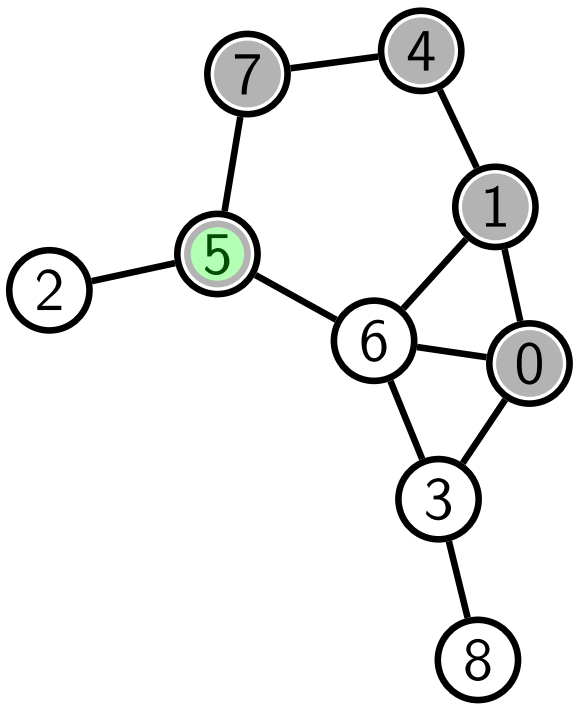
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



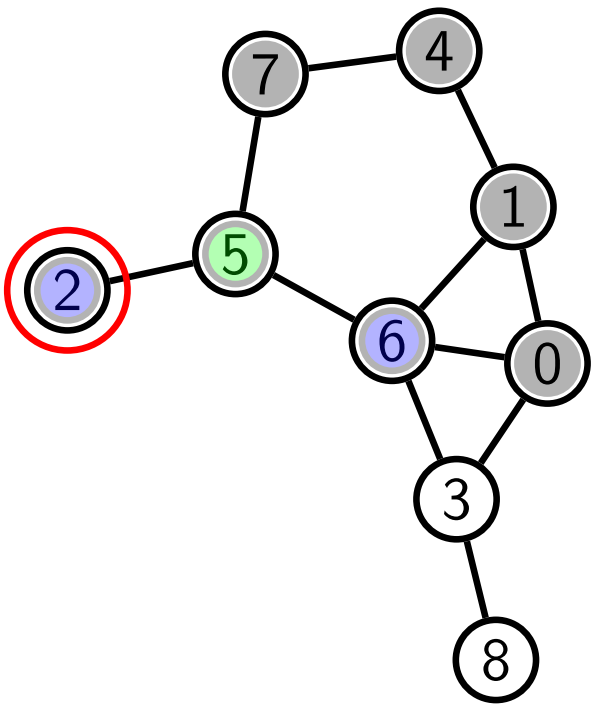
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



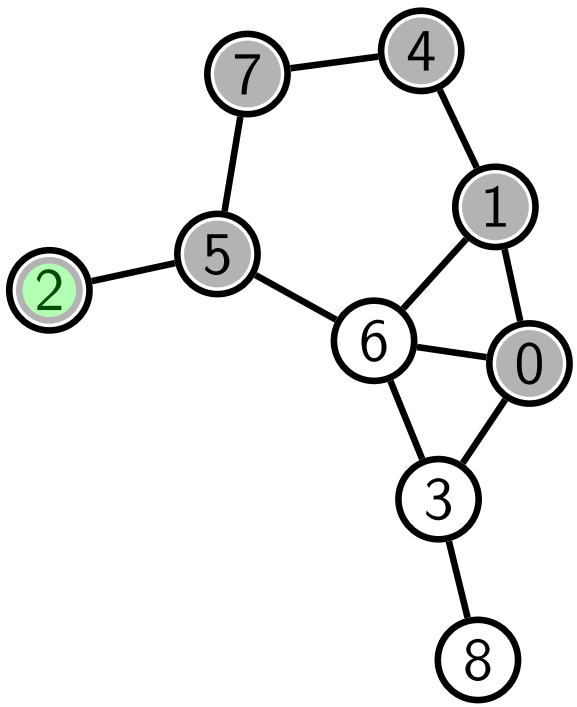
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



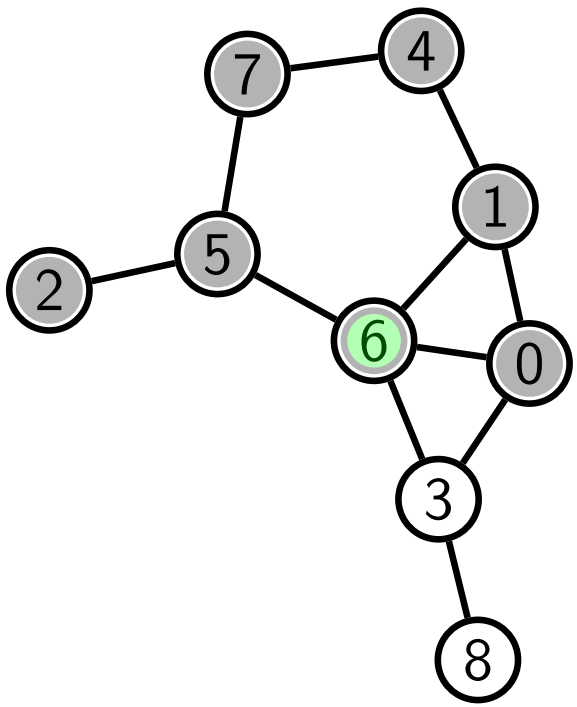
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



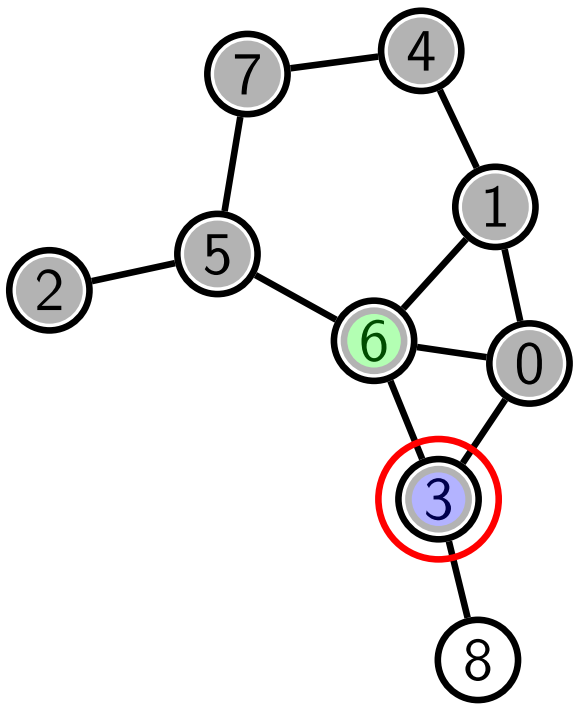
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



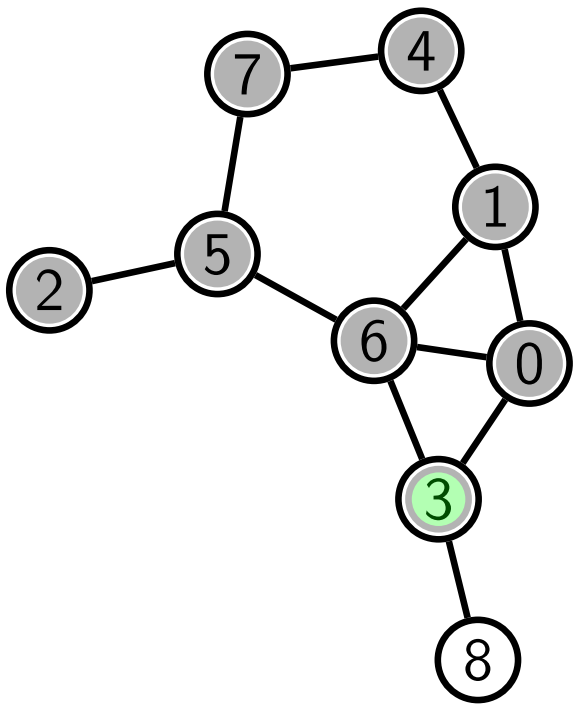
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



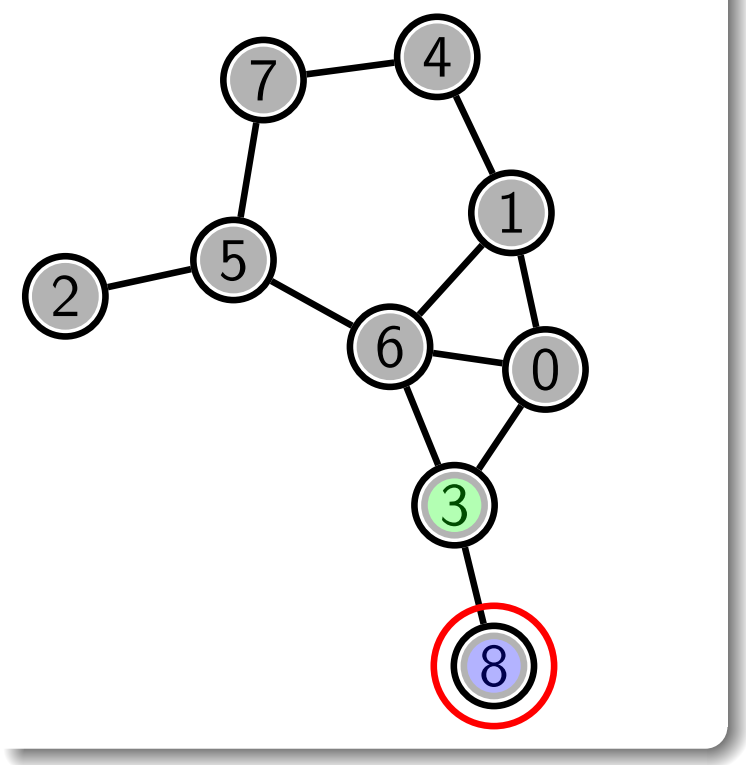
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



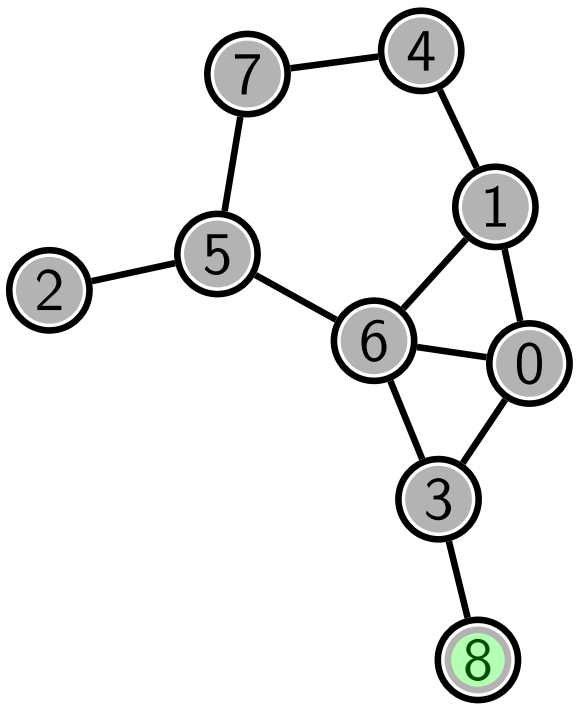
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



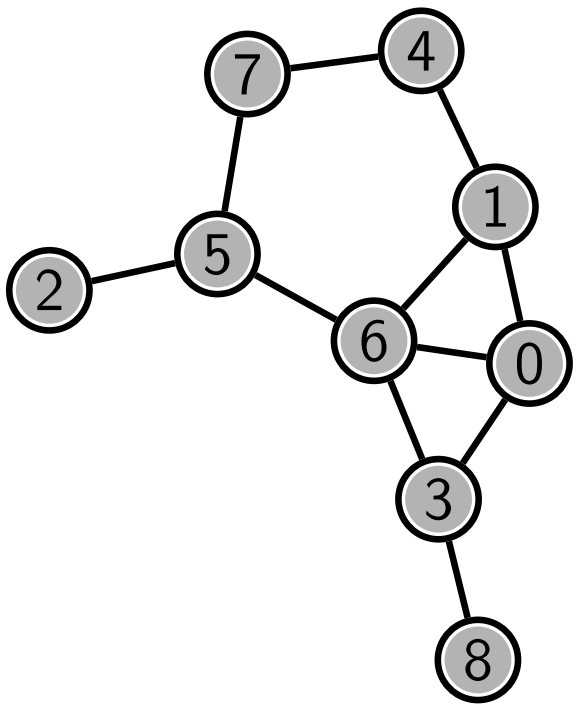
Parcours en profondeur : exemple

Exemple 6 (départ = 0)



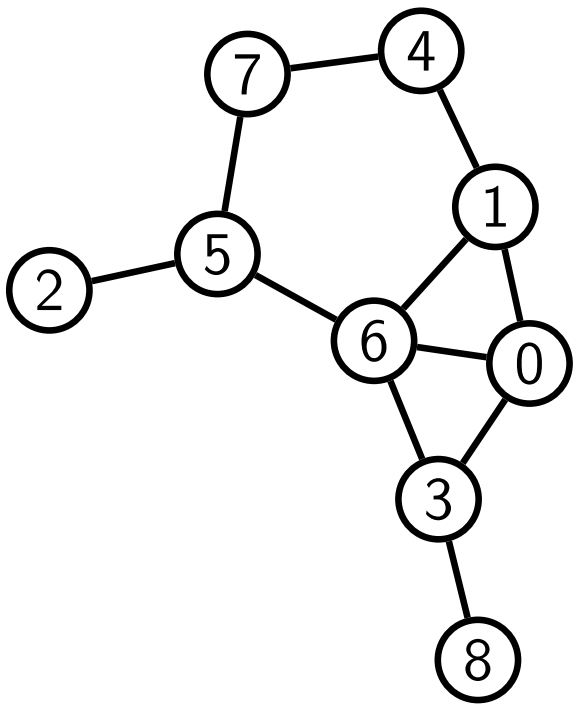
Parcours en profondeur : exemple

Exemple 6 (départ = 0)

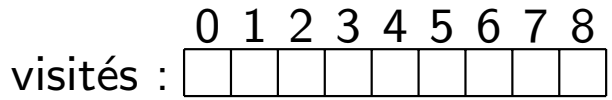


Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

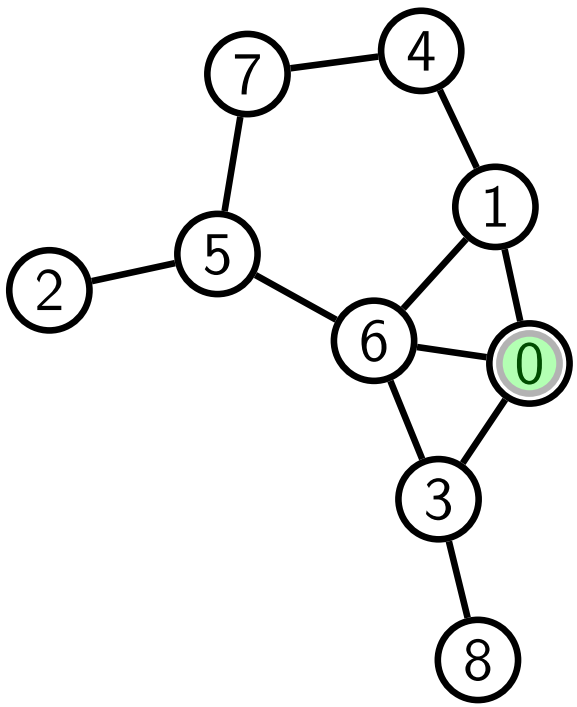


à traiter (pile)

résultat :

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

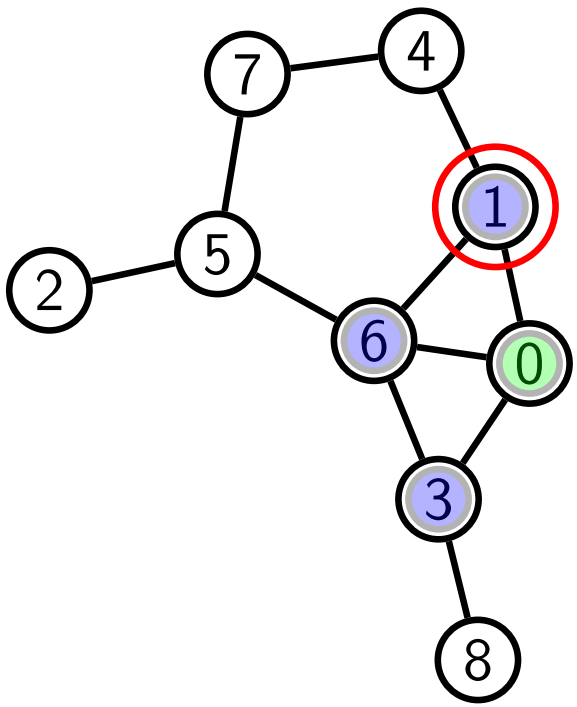
	0	1	2	3	4	5	6	7	8
visités :	✓								

à traiter (pile)

résultat : 0

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

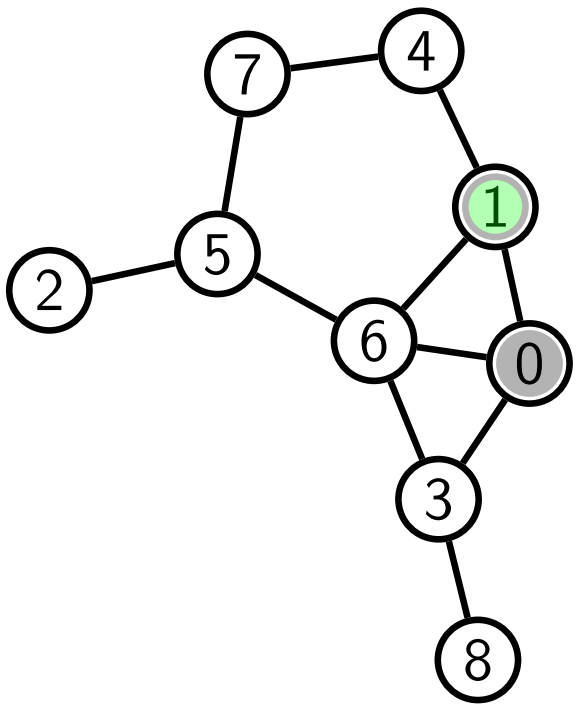
1
3
6

à traiter (pile)

résultat : 0

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

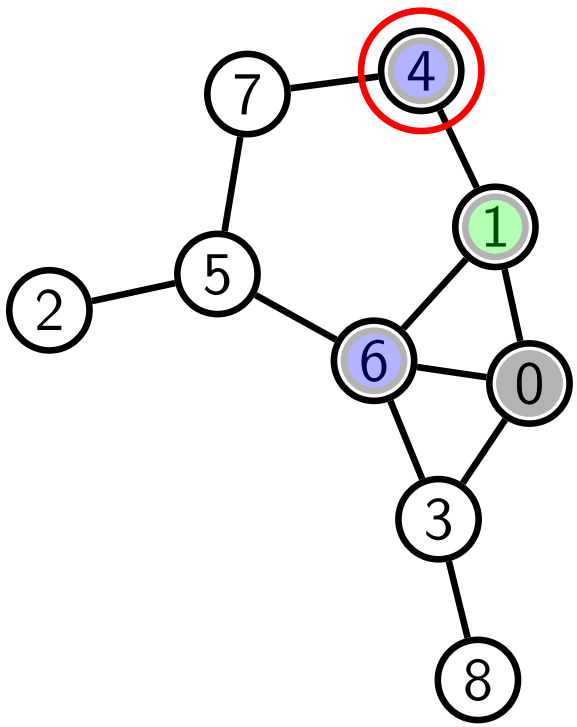
3
6

à traiter (pile)

résultat : 0 1

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

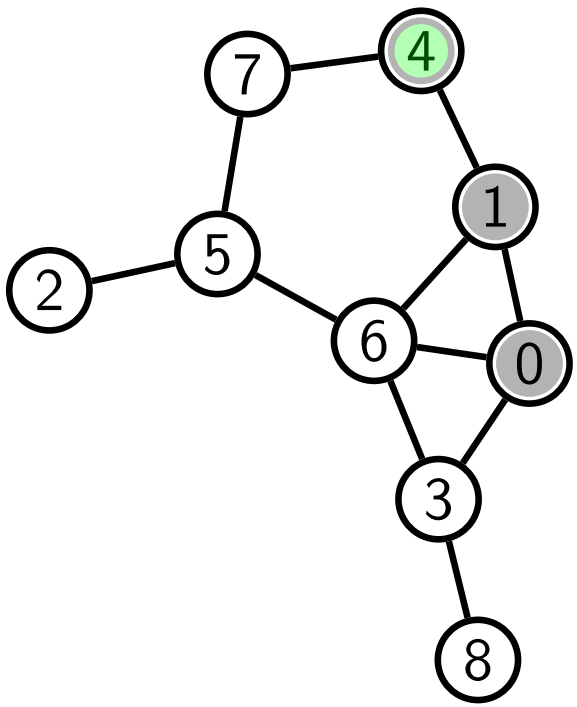
4
6
3
6

à traiter (pile)

résultat : 0 1

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

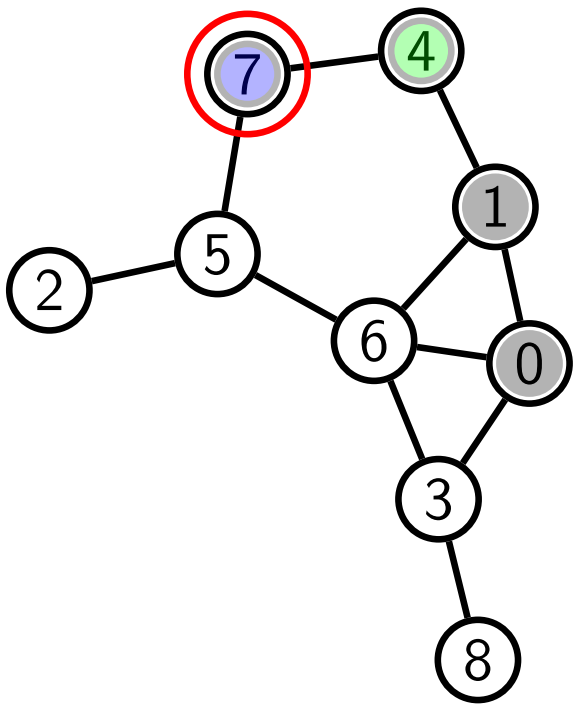
6
3
6

à traiter (pile)

résultat : 0 1 4

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

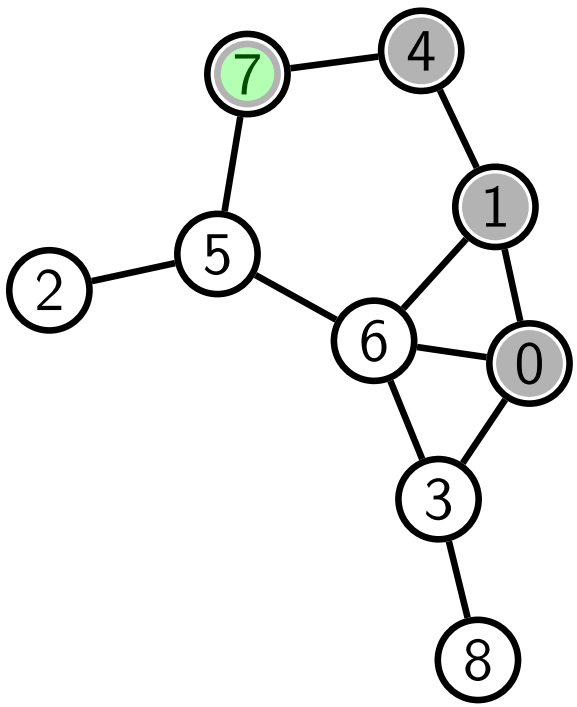
7
6
3
6

à traiter (pile)

résultat : 0 1 4

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

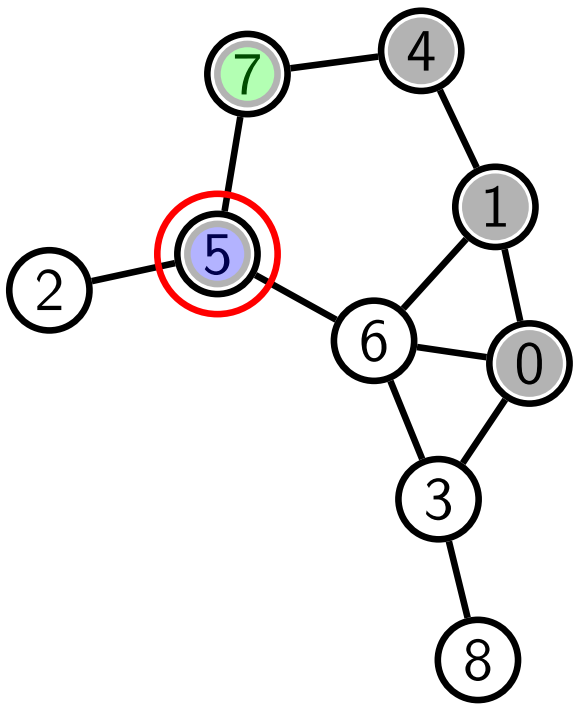
6
3
6

à traiter (pile)

résultat : 0 1 4 7

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

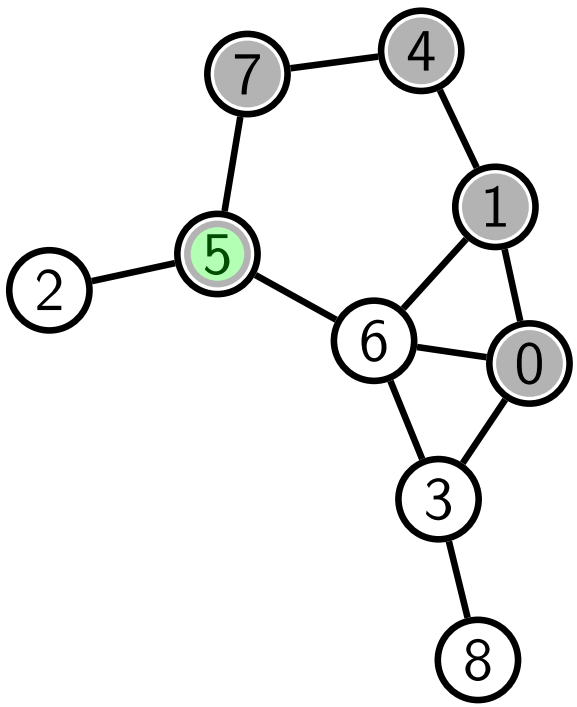
5
6
3
6

à traiter (pile)

résultat : 0 1 4 7

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

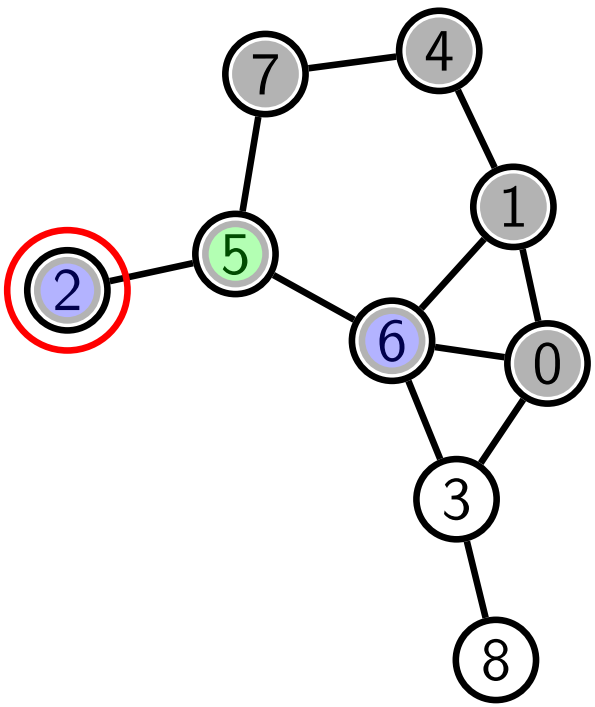
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

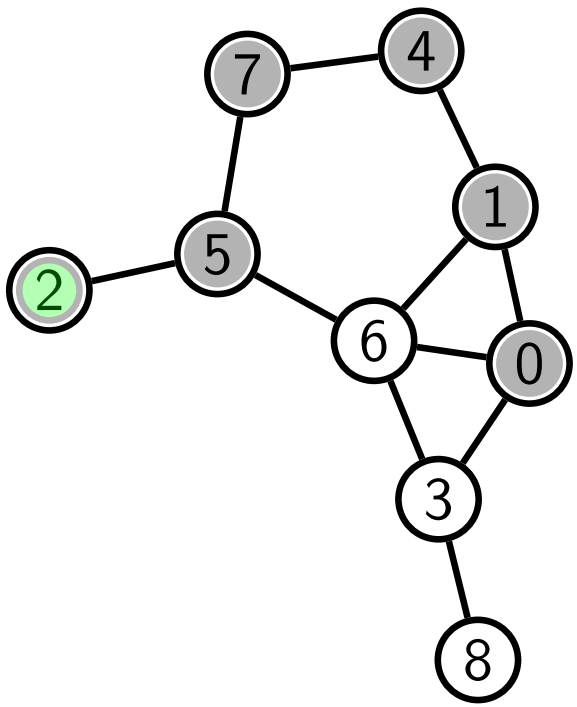
	0	1	2	3	4	5	6	7	8
visités :	✓	✓			✓	✓		✓	

à traiter (pile)
2
6
6
3
6

résultat : 0 1 4 7 5

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

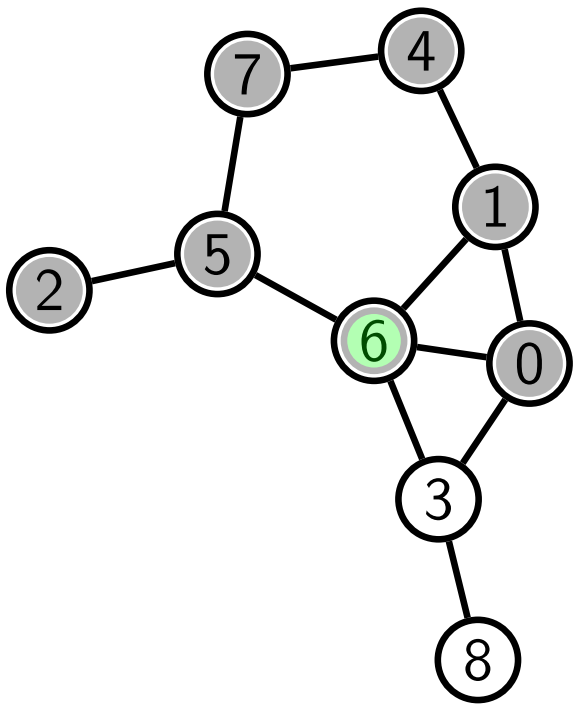
6
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

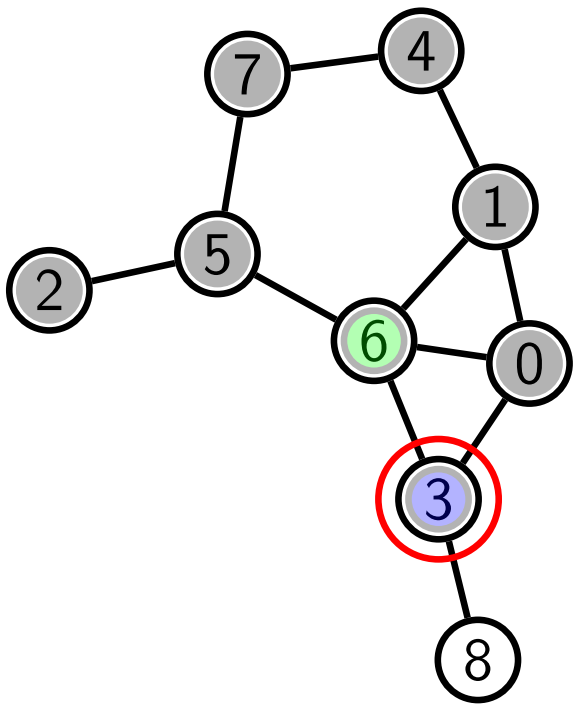
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

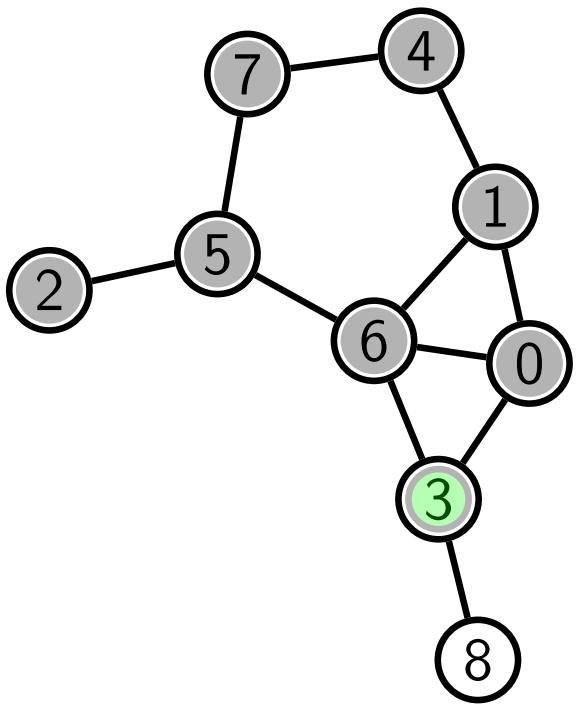
3
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

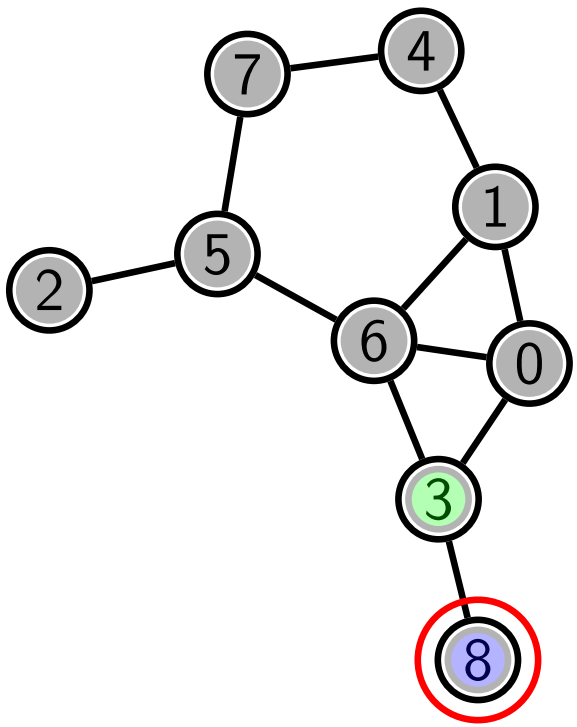
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

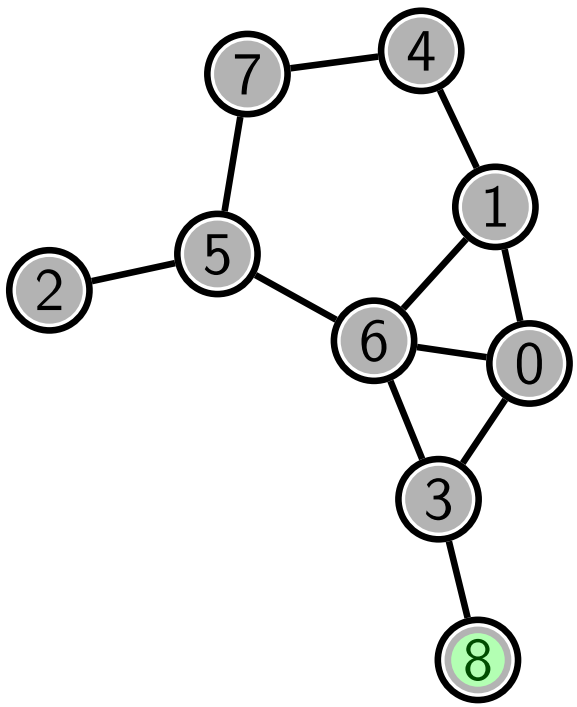
8
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

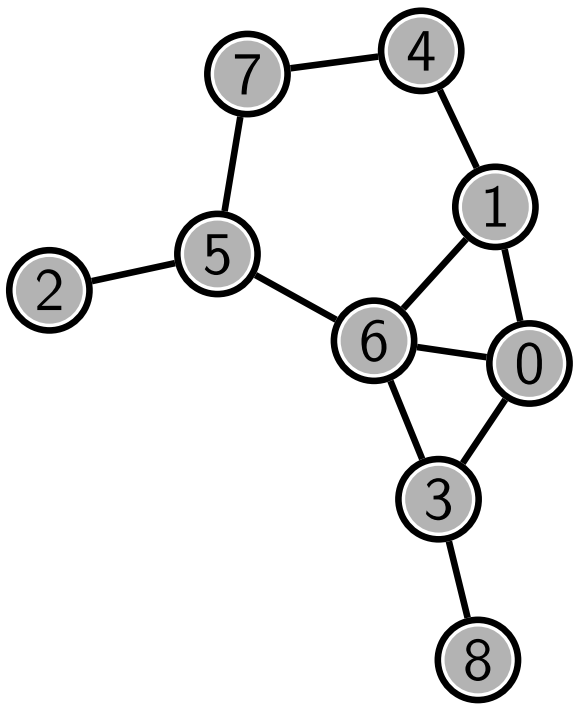
6
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3 8

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

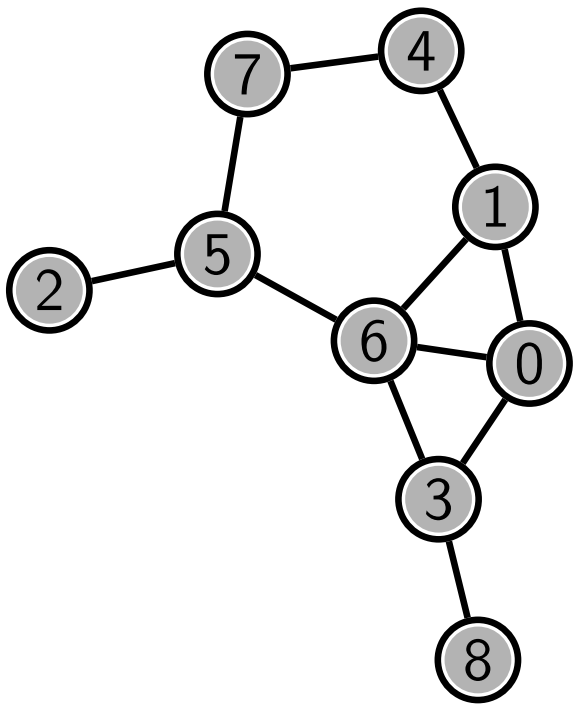
3
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3 8

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

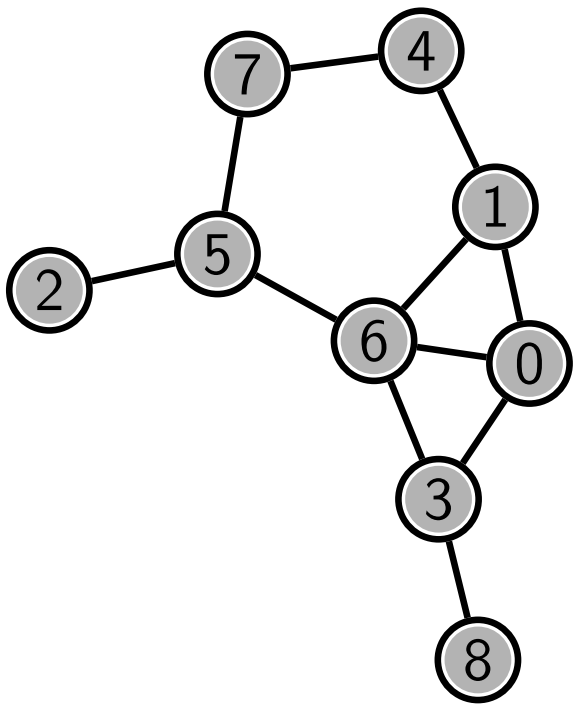
6

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3 8

Parcours en profondeur : exemple

Exemple 6 (départ = 0)



Les coulisses

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

à traiter (pile)

résultat : 0 1 4 7 5 2 6 3 8

Algorithme 1 : PROFONDEUR(G, départ, visités=NIL)

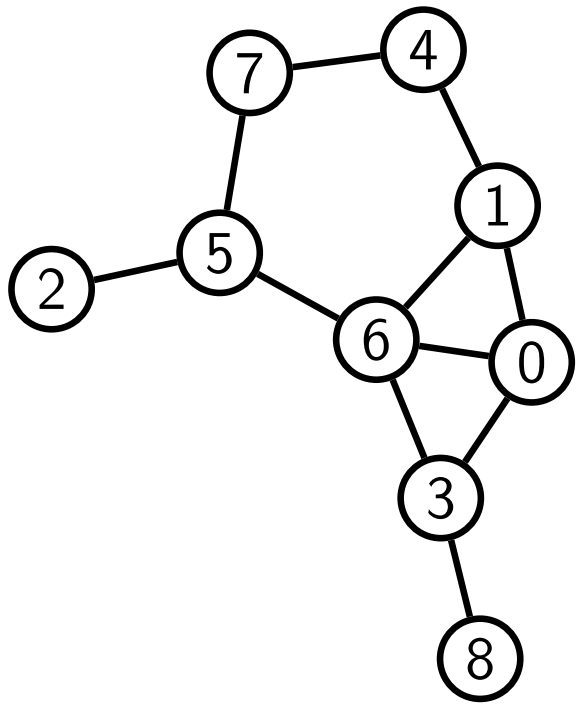
Entrées : un graphe non-orienté G, un sommet de départ, un tableau (facultatif) visités de taille $|V|$.

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;
```

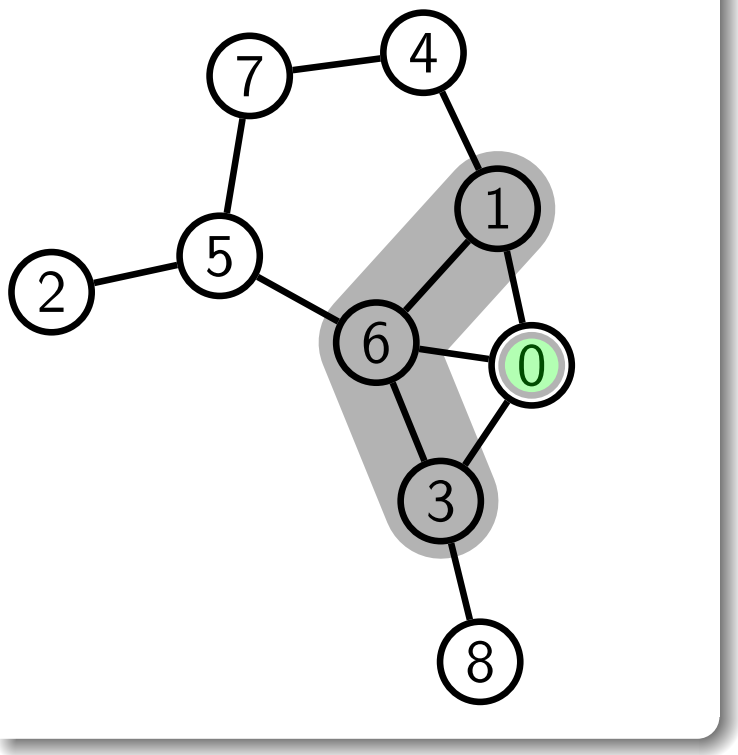
Parcours en largeur : exemple

Exemple 7 (départ = 0)



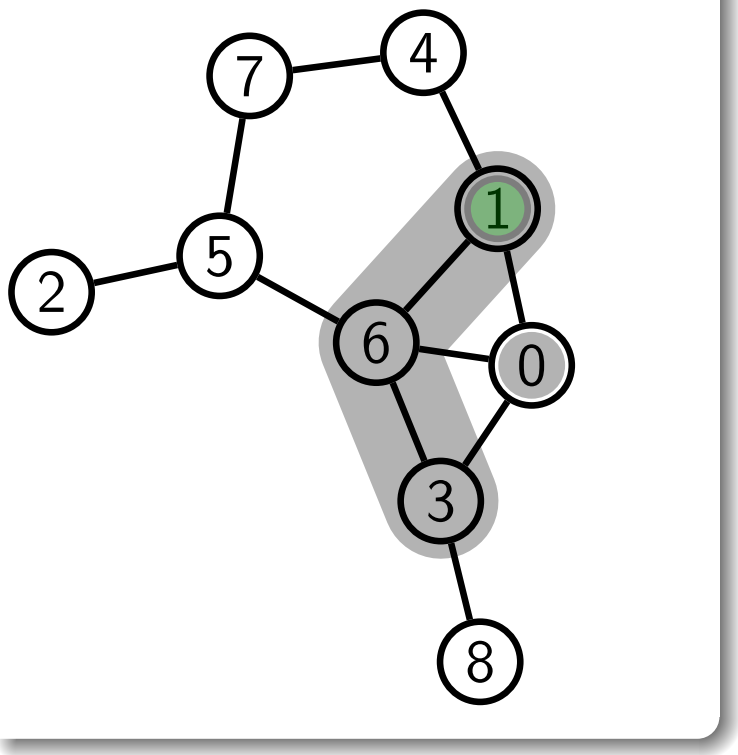
Parcours en largeur : exemple

Exemple 7 (départ = 0)



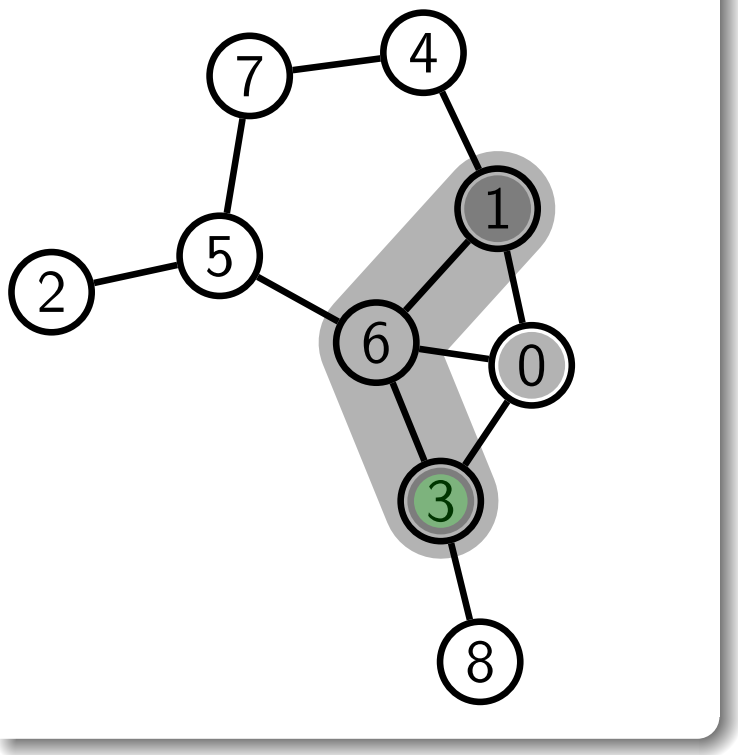
Parcours en largeur : exemple

Exemple 7 (départ = 0)



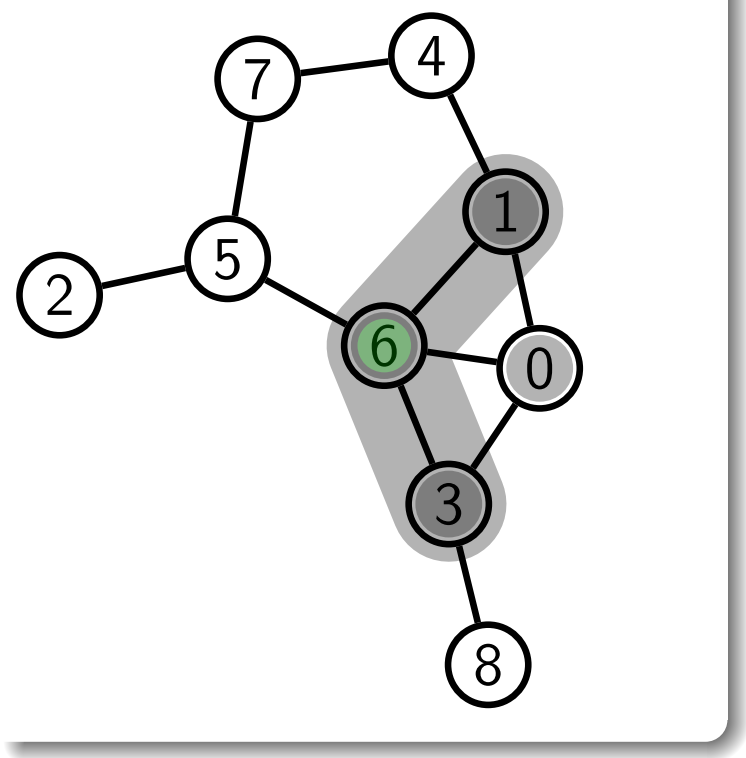
Parcours en largeur : exemple

Exemple 7 (départ = 0)



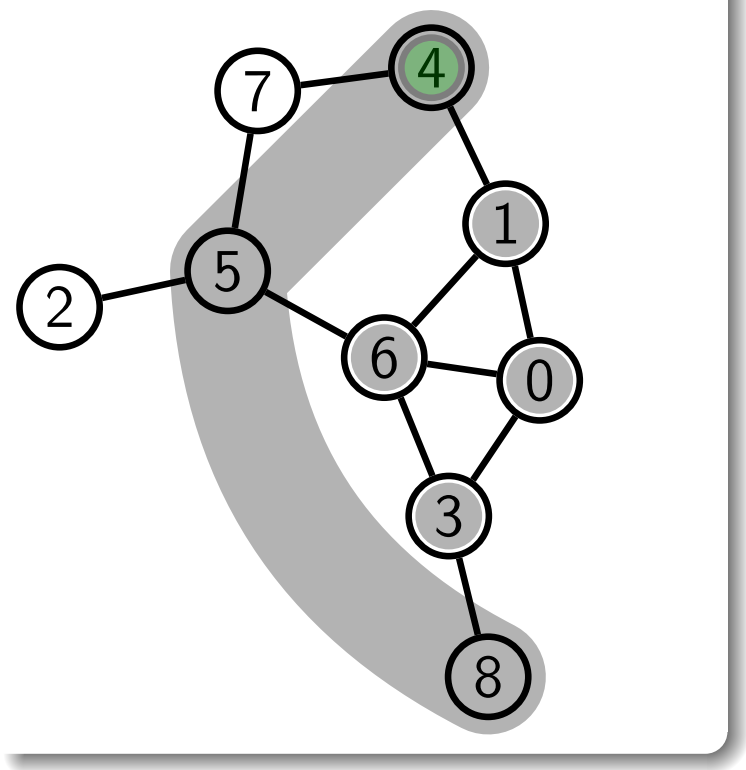
Parcours en largeur : exemple

Exemple 7 (départ = 0)



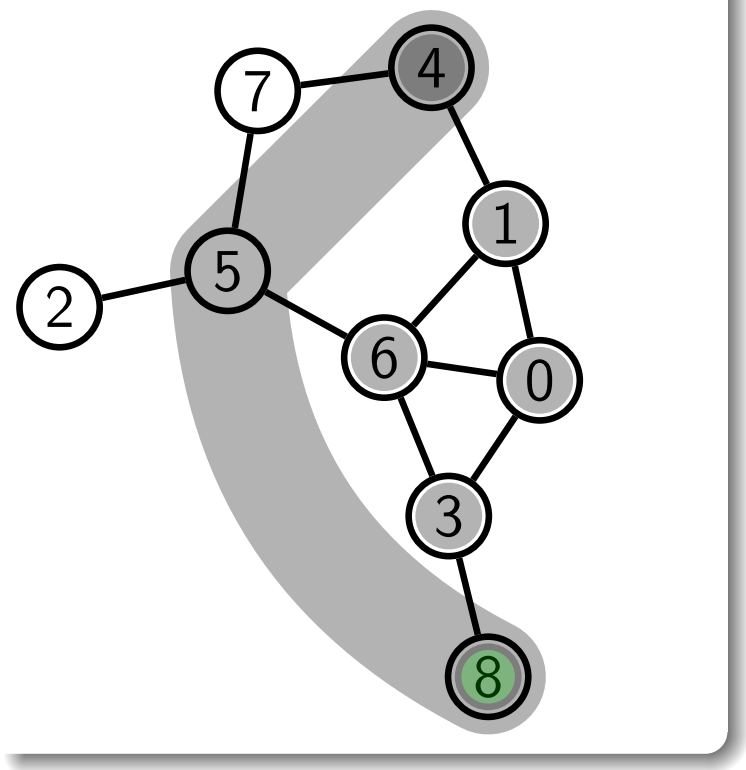
Parcours en largeur : exemple

Exemple 7 (départ = 0)



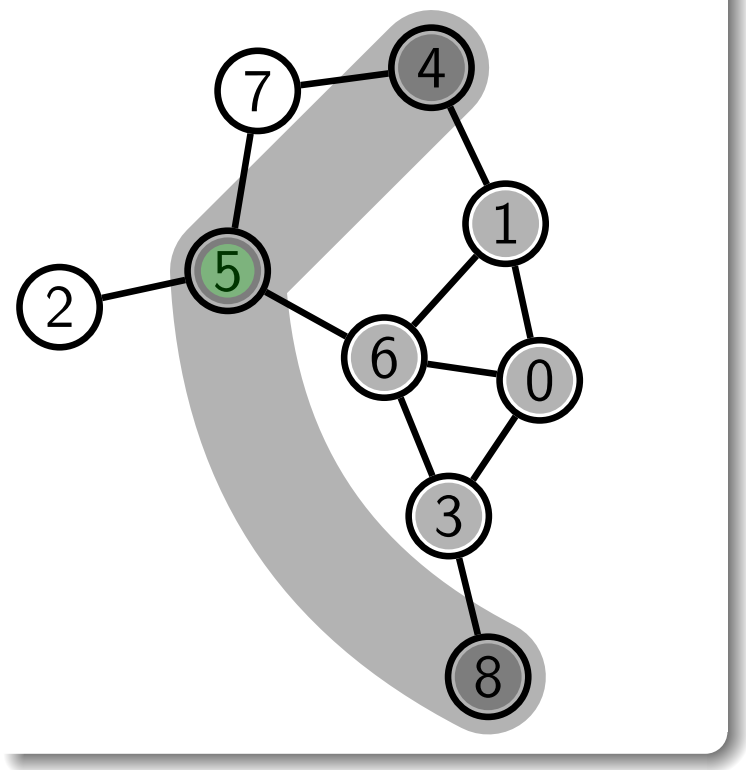
Parcours en largeur : exemple

Exemple 7 (départ = 0)



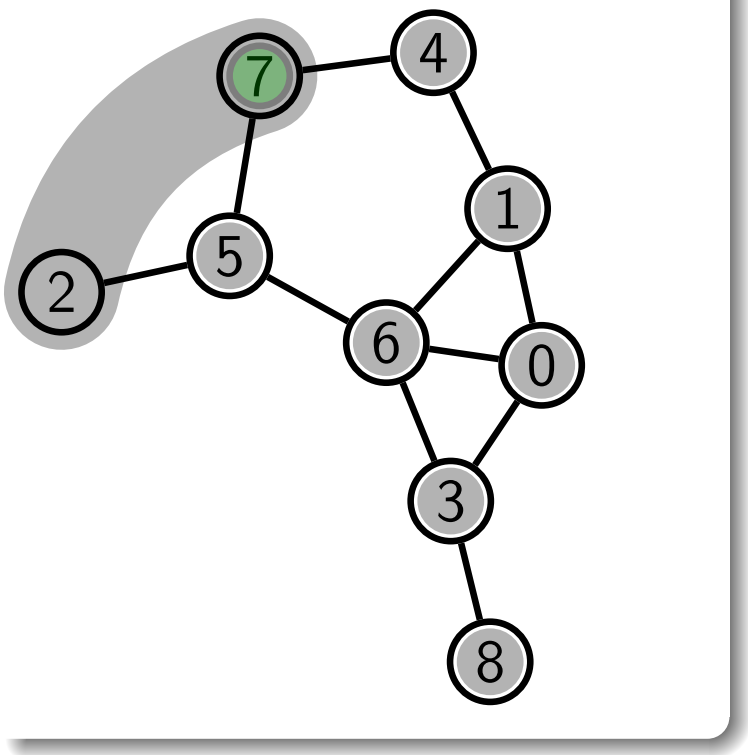
Parcours en largeur : exemple

Exemple 7 (départ = 0)



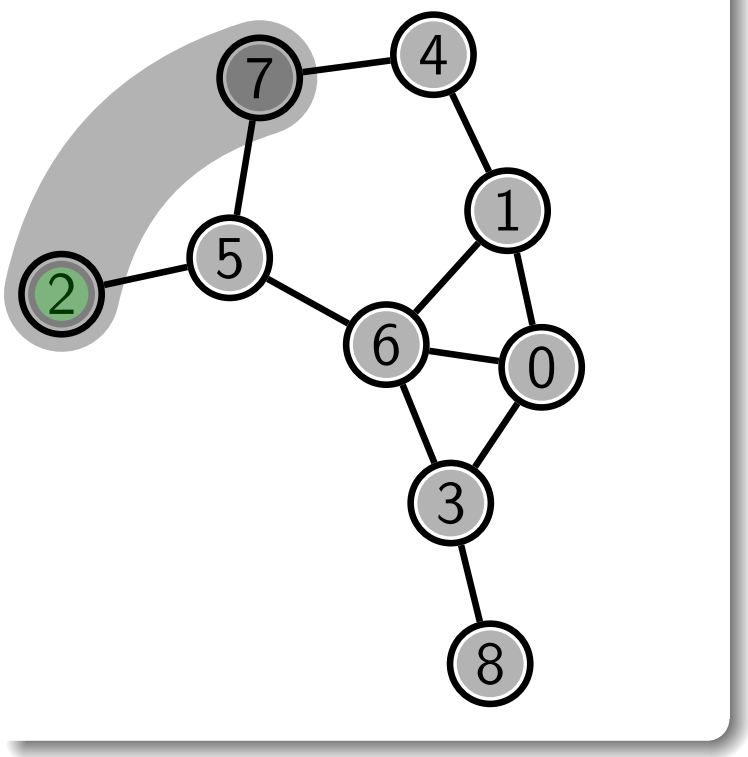
Parcours en largeur : exemple

Exemple 7 (départ = 0)



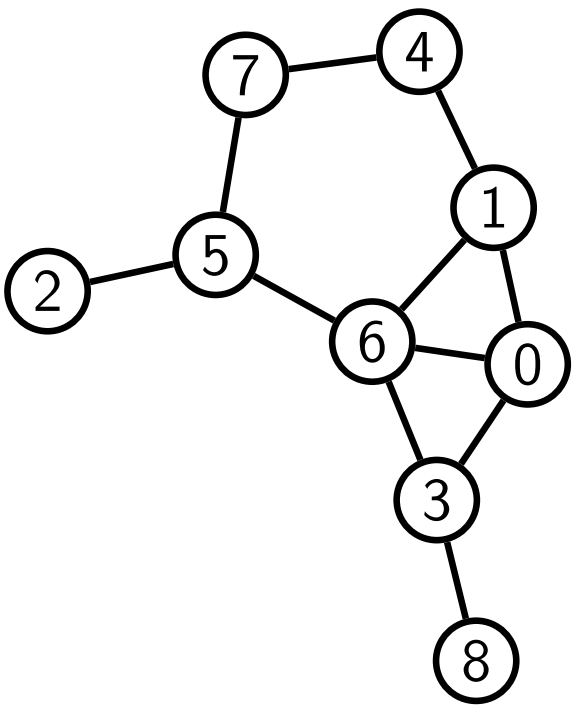
Parcours en largeur : exemple

Exemple 7 (départ = 0)

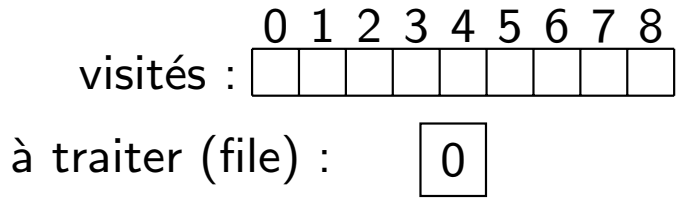


Parcours en largeur : exemple

Exemple 7 (départ = 0)



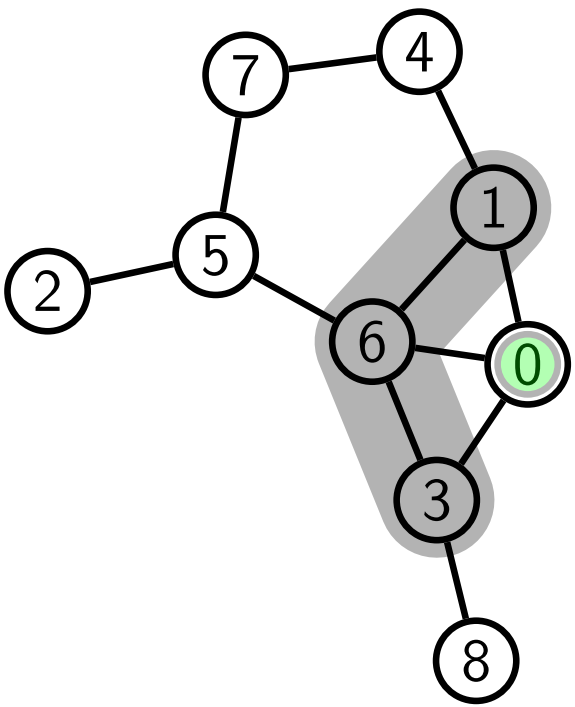
Les coulisses



(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 7 (départ = 0)



Les coulisses

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

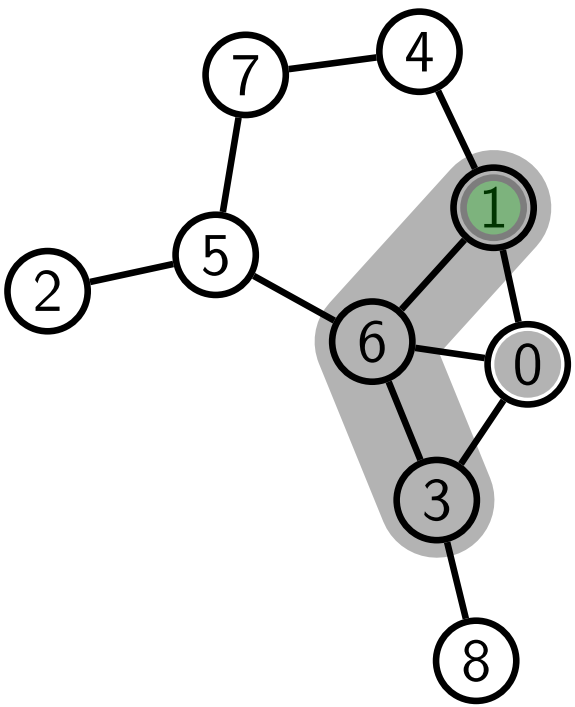
à traiter (file) : | 6 | 3 | 1 |

résultat : 0

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 7 (départ = 0)



Les coulisses

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

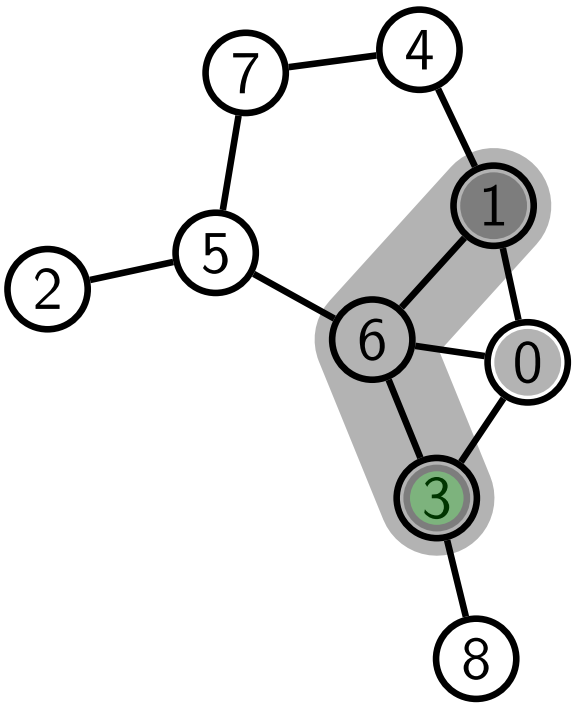
à traiter (file) : | 6 | 4 | 6 | 3 |

résultat : 0 1

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 7 (départ = 0)



Les coulisses

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

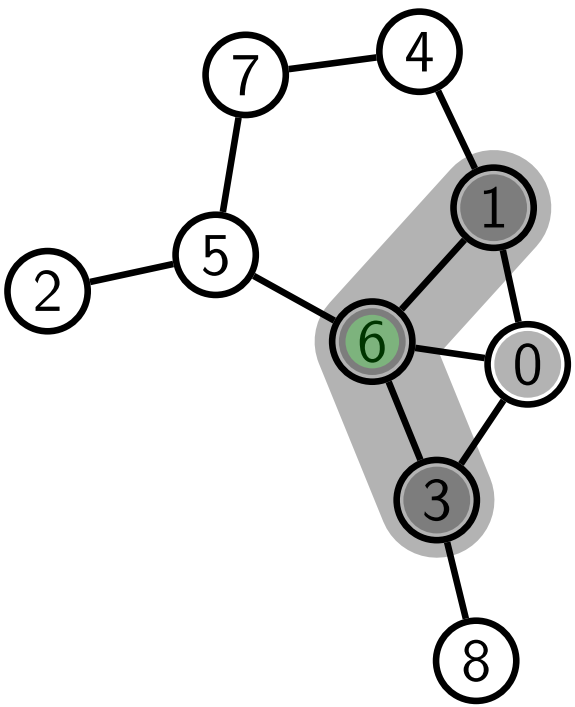
à traiter (file) :	8	6	6	4	6

résultat : 0 1 3

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 7 (départ = 0)



Les coulisses

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

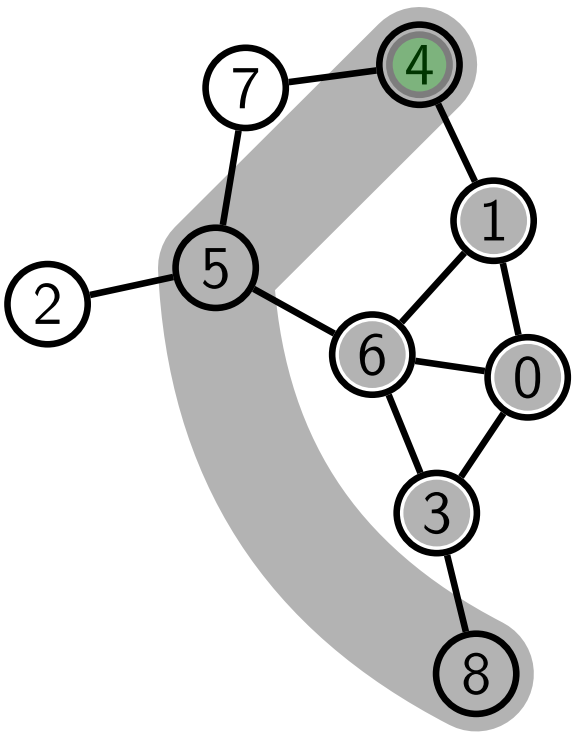
à traiter (file) : | 5 | 8 | 6 | 6 | 4 |

résultat : 0 1 3 6

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 7 (départ = 0)



Les coulisses

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

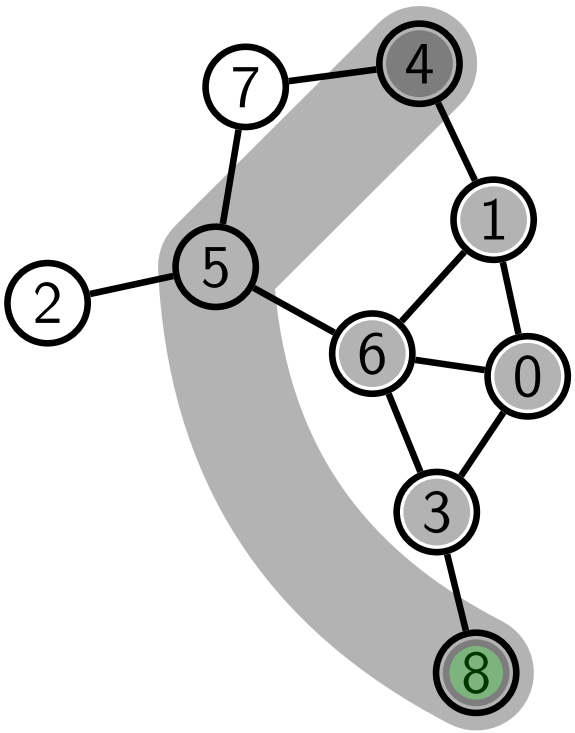
à traiter (file) :	7	5	8	6	6

résultat : 0 1 3 6 4

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 7 (départ = 0)



Les coulisses

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

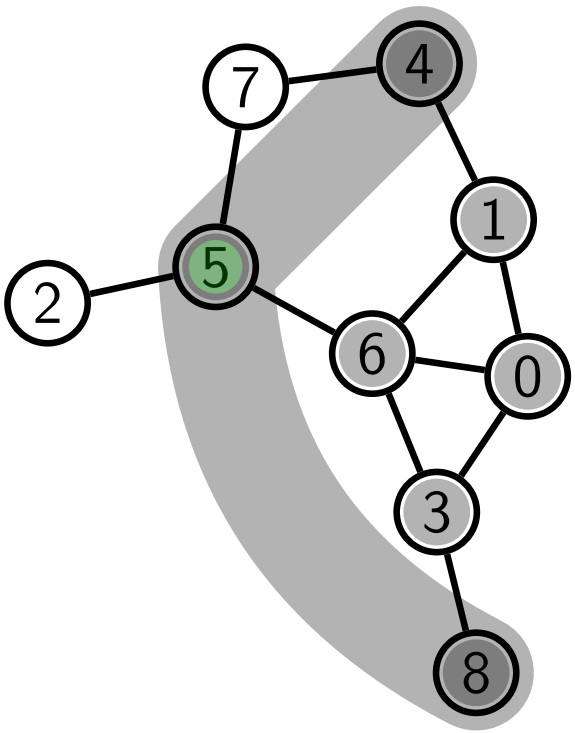
à traiter (file) : | 5 | 5 | 7 | 5 |

résultat : 0 1 3 6 4 8

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 7 (départ = 0)



Les coulisses

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

à traiter (file) :

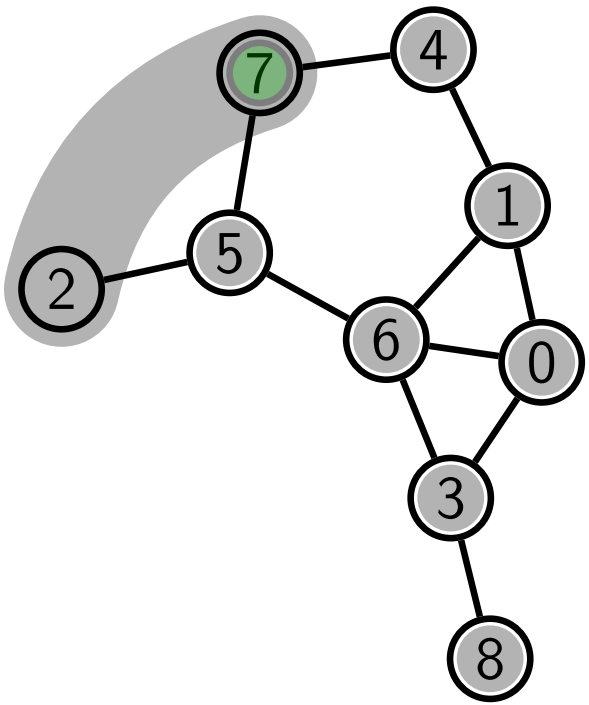
7	2	5	5	7

résultat : 0 1 3 6 4 8 5

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 7 (départ = 0)



Les coulisses

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

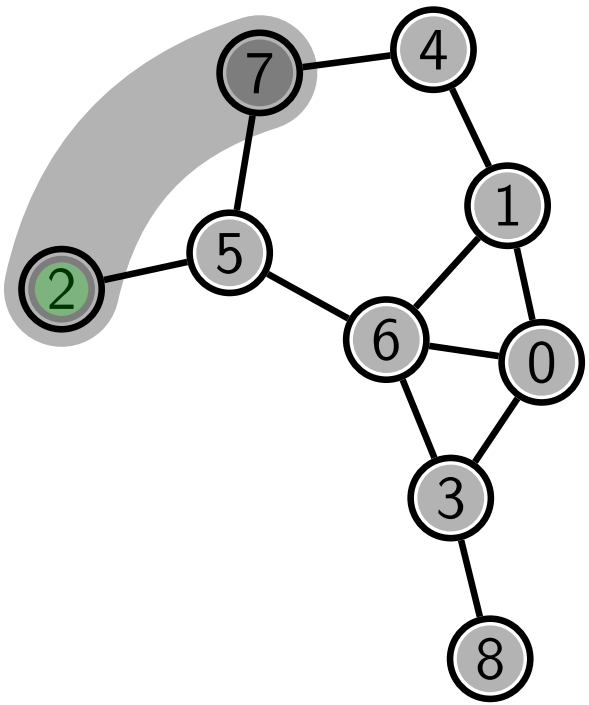
à traiter (file) : | 2 | 2 | 7 | 2 |

résultat : 0 1 3 6 4 8 5 7

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 7 (départ = 0)



Les coulisses

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

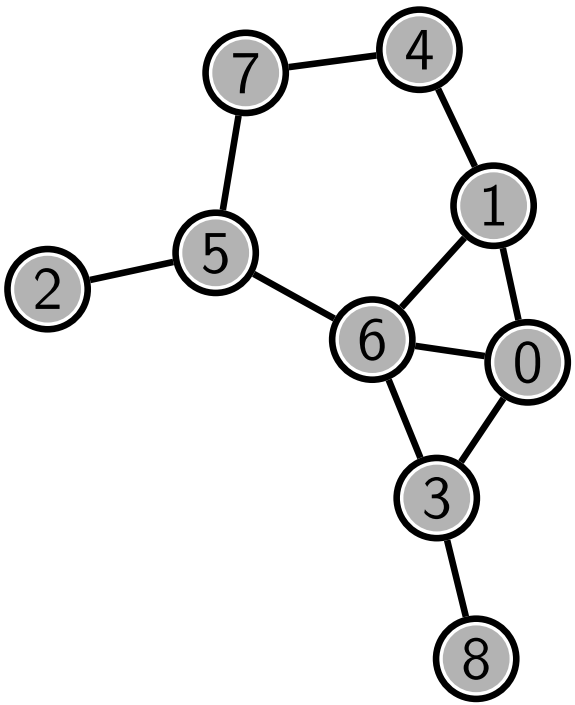
à traiter (file) : | 2 | 2 | 7 |

résultat : 0 1 3 6 4 8 5 7 2

(file : fin à gauche, début à droite)

Parcours en largeur : exemple

Exemple 7 (départ = 0)



Les coulisses

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

à traiter (file) :

résultat : 0 1 3 6 4 8 5 7 2

(file : fin à gauche, début à droite)

Algorithme 2 : 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;
```

Complexité des algorithmes de parcours

- $O(|V|^2)$ pour une matrice d'adjacence ;

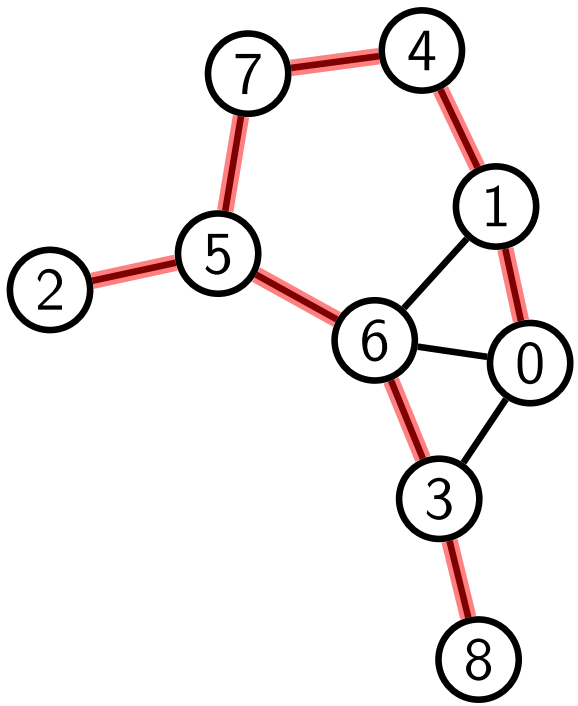
Complexité des algorithmes de parcours

- $O(|V|^2)$ pour une matrice d'adjacence ;
- $O(|V| + |E|)$ pour des listes d'adjacence (rappel : $\sum_{v \in V} \deg(v) = 2|E|$) ;

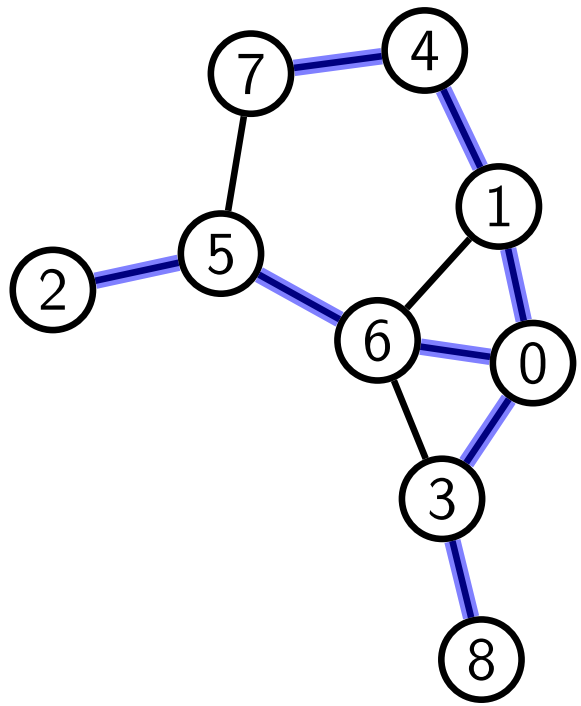
Notion d'arbre de parcours

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 8 (arbres de parcours au départ de 0)



profondeur

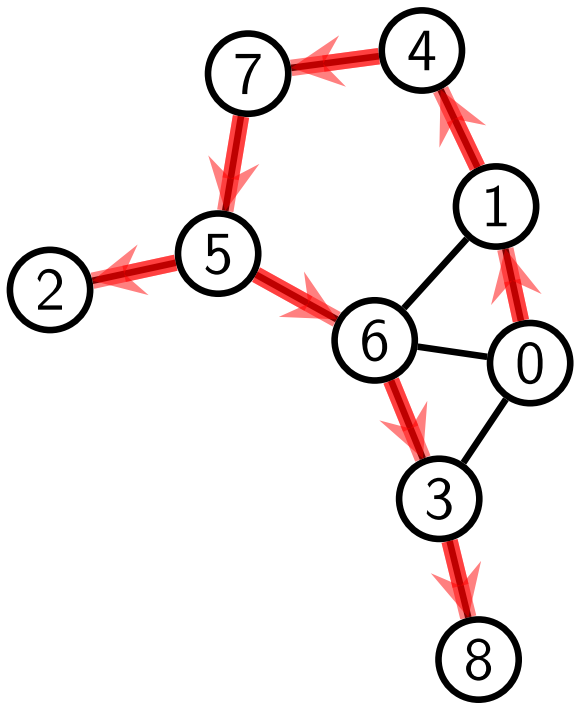


largeur

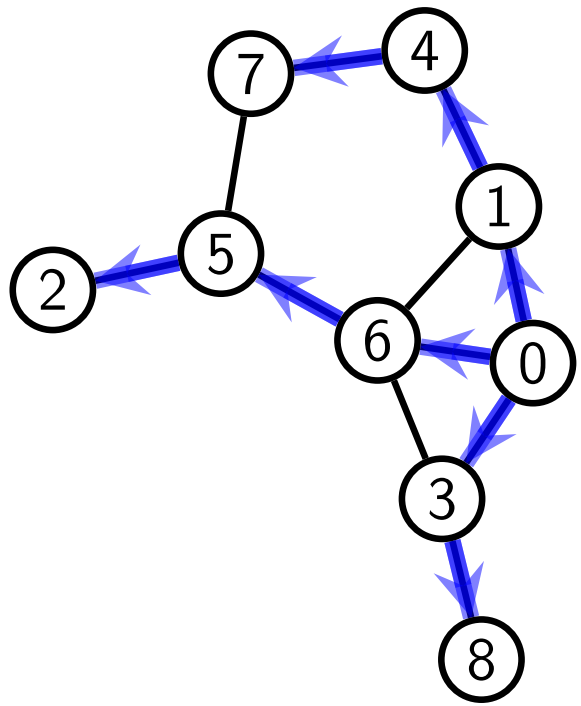
Notion d'arbre de parcours

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 8 (arbres de parcours au départ de 0)



profondeur

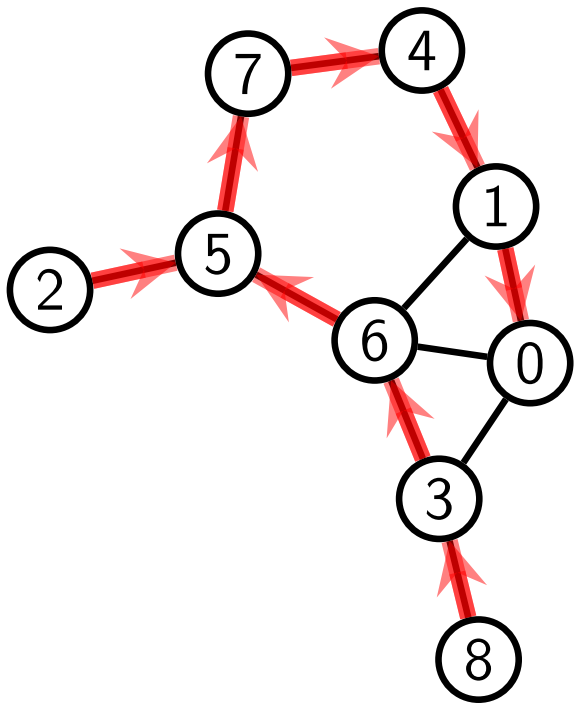


largeur

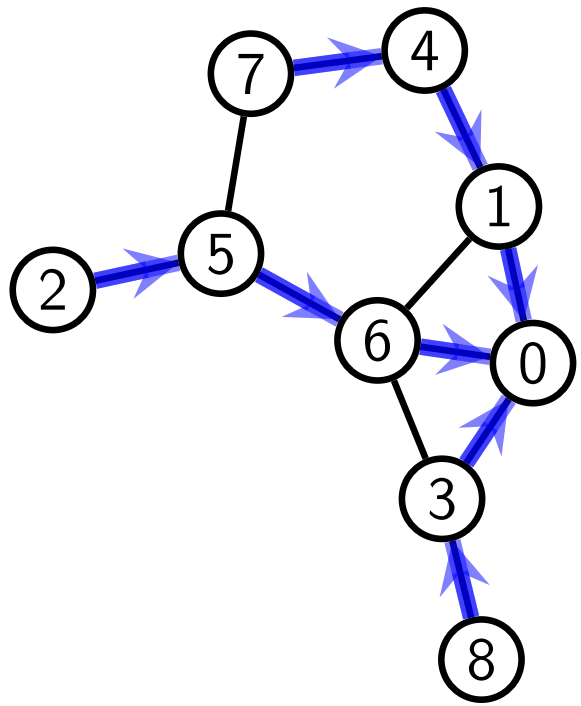
Notion d'arbre de parcours

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 8 (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 ;

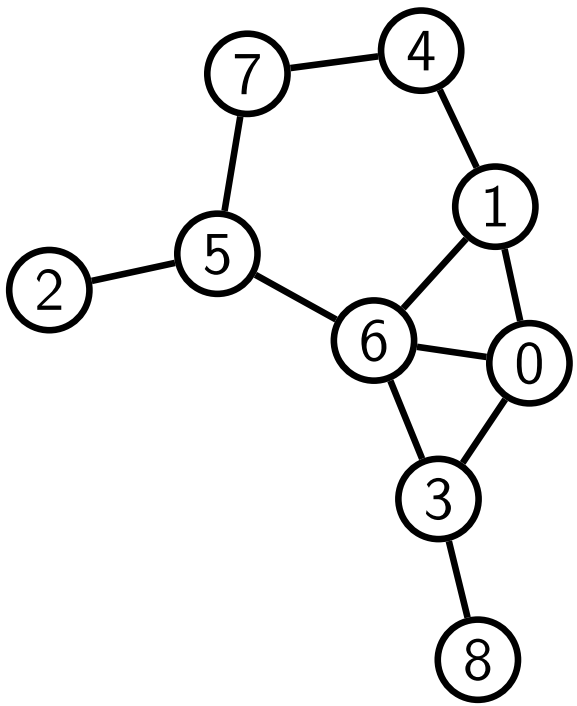
Calcul des arbres de parcours

- 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 ;

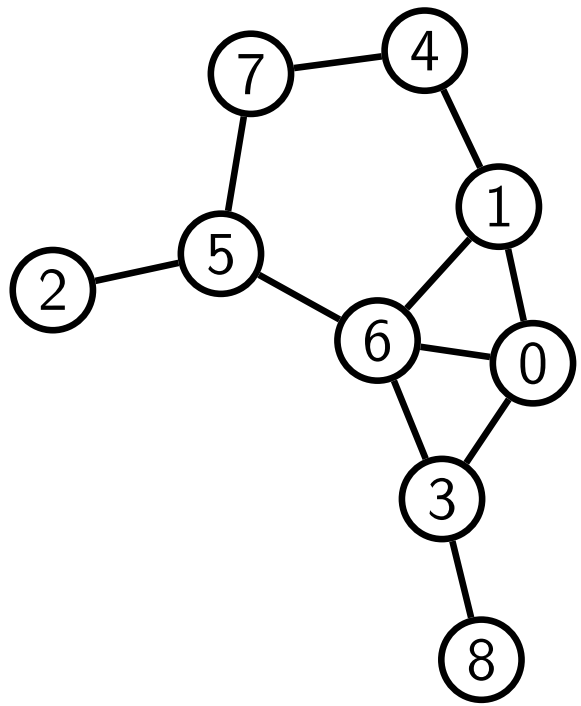
Calcul des arbres de parcours

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

Exemple 9 (profondeur)



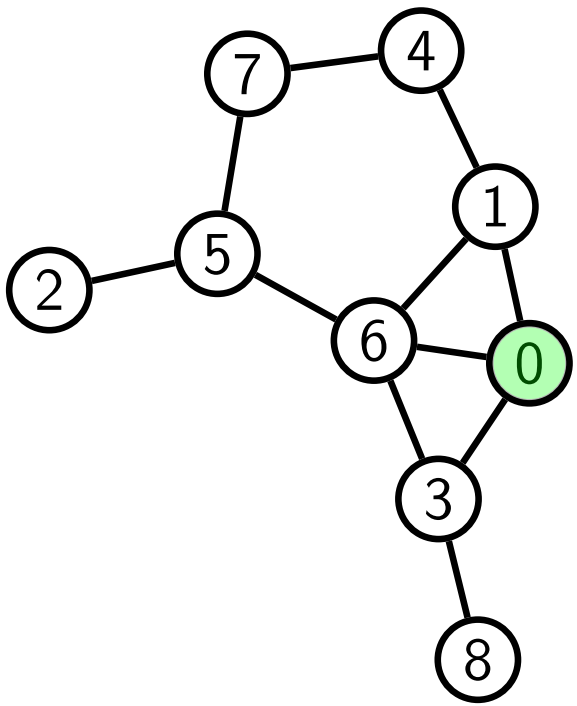
Exemple 10 (largeur)



Calcul des arbres de parcours

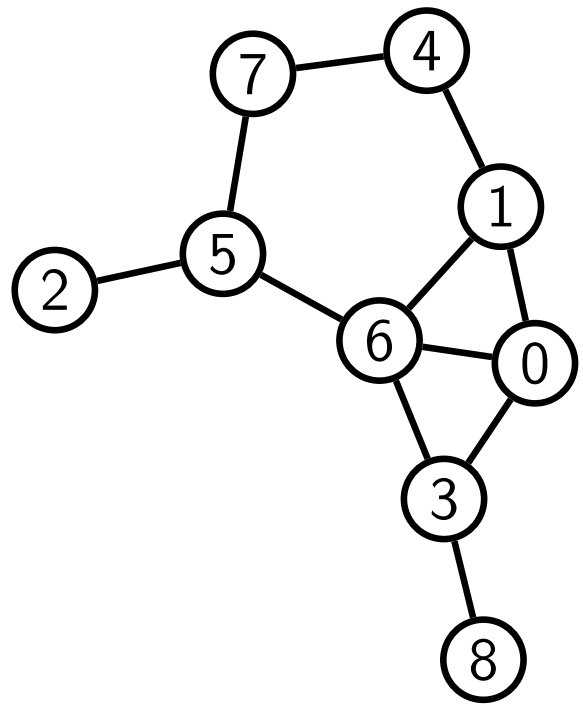
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

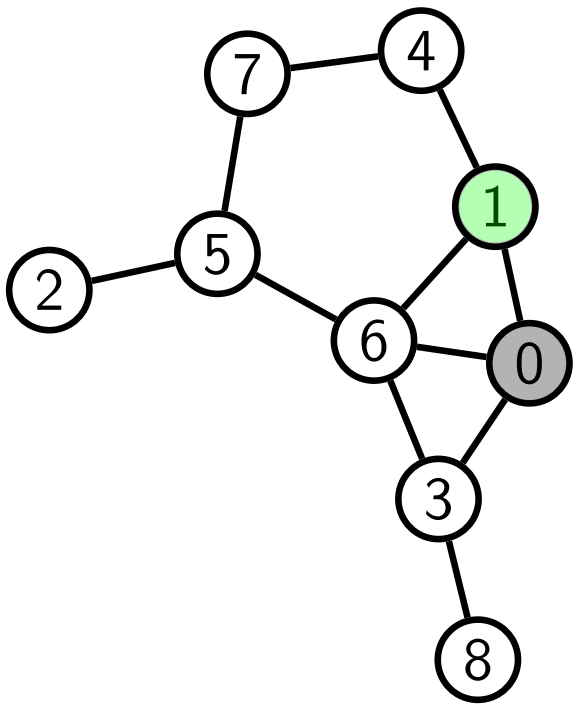


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

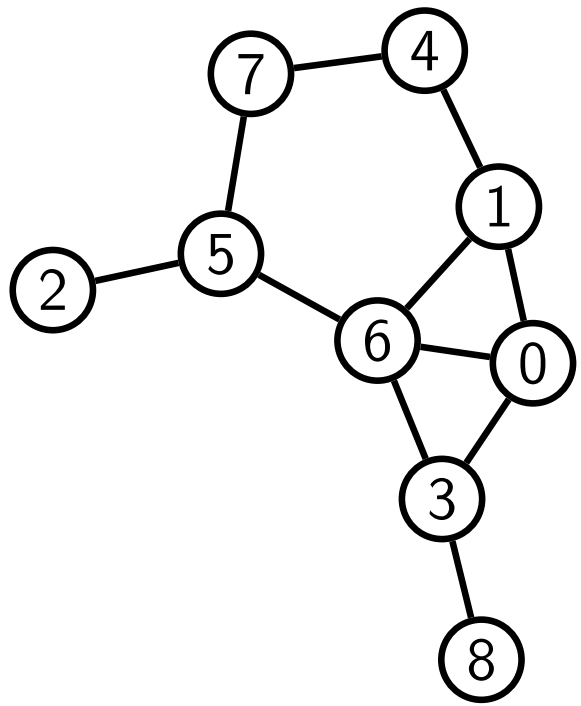
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

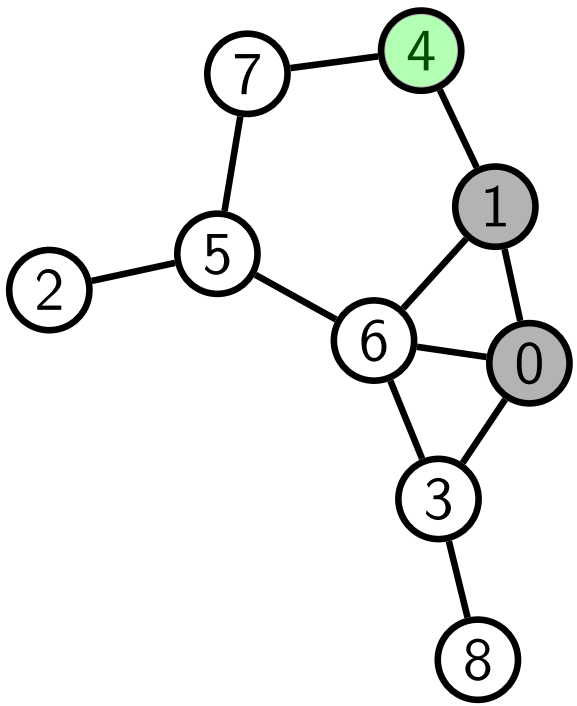


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

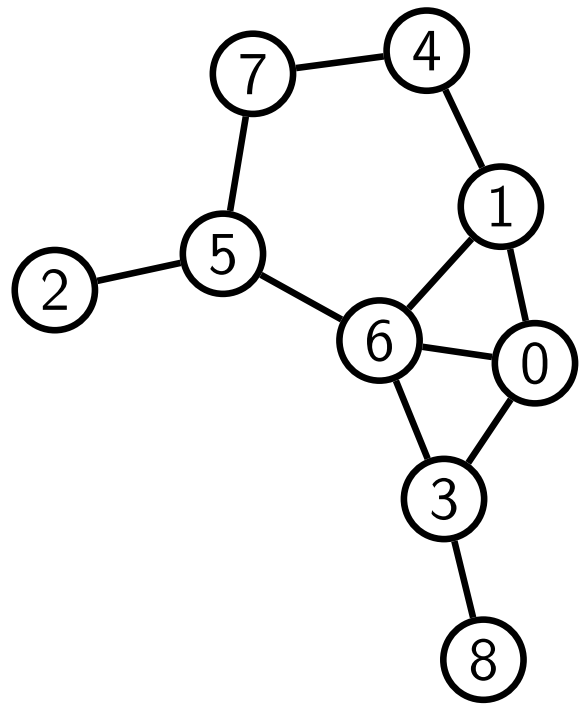
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

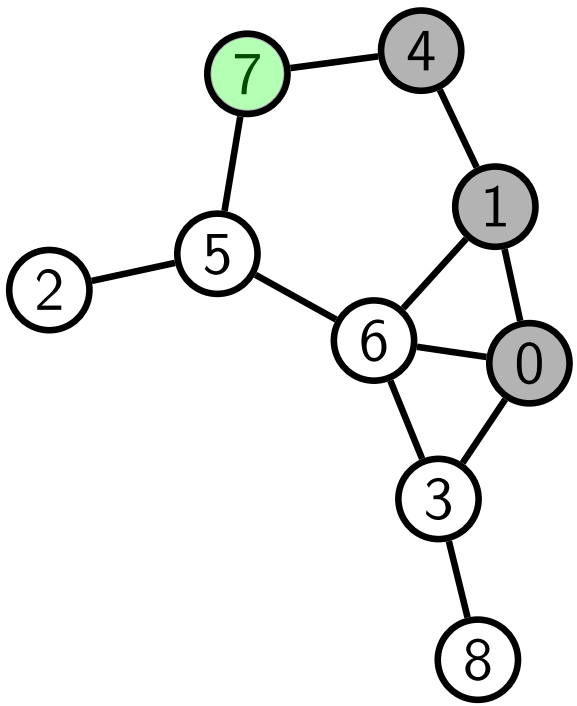


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

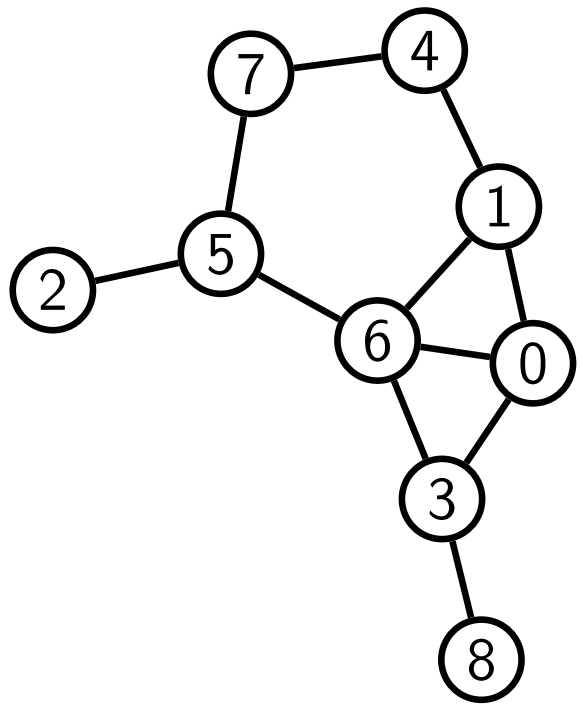
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

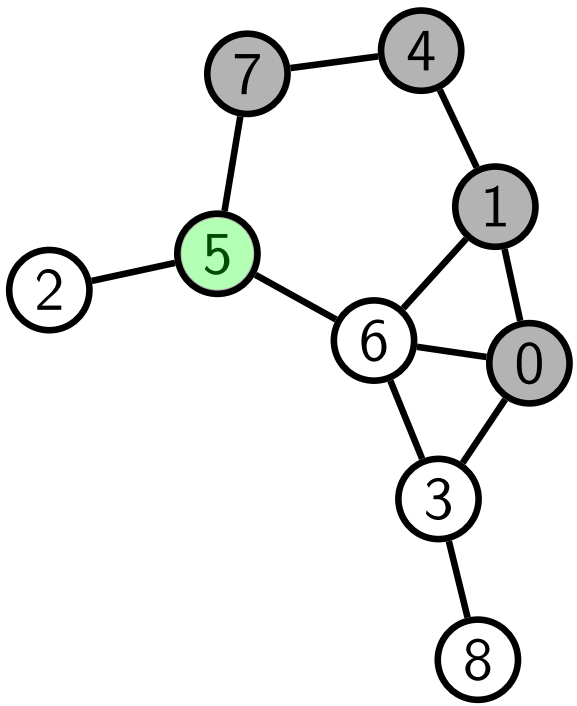


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

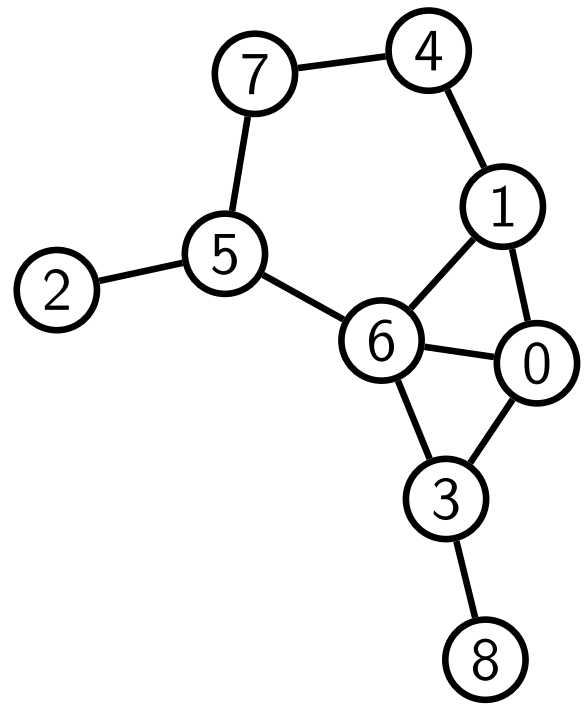
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

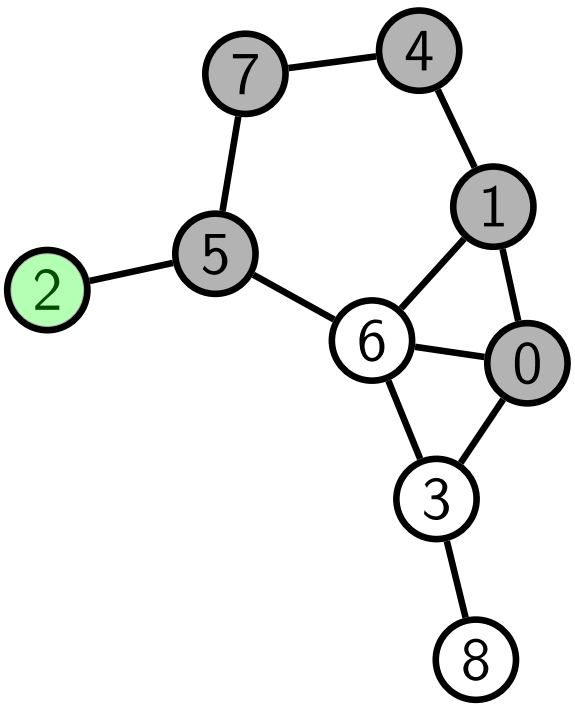


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

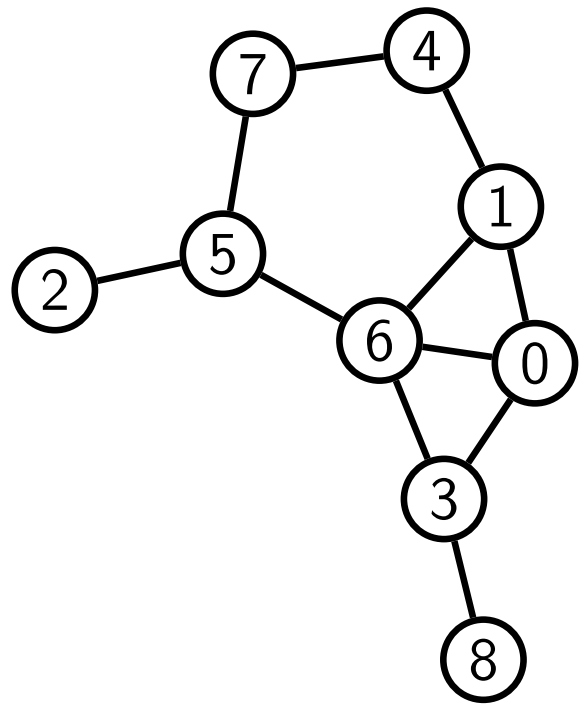
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

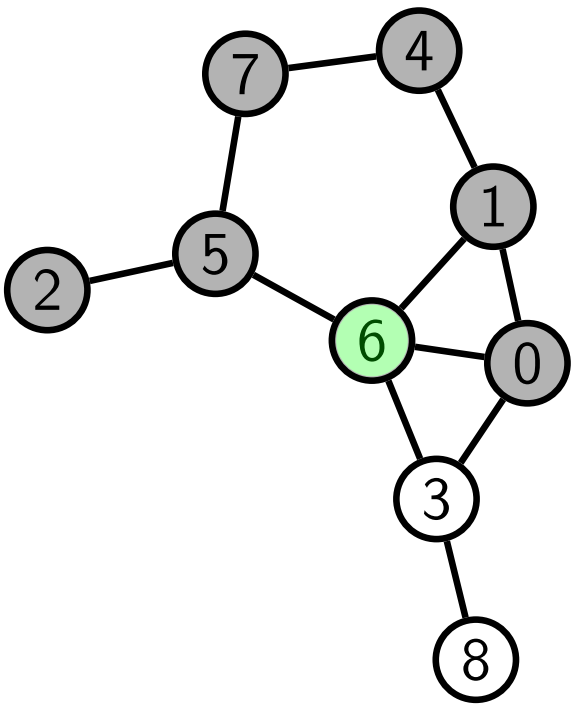


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

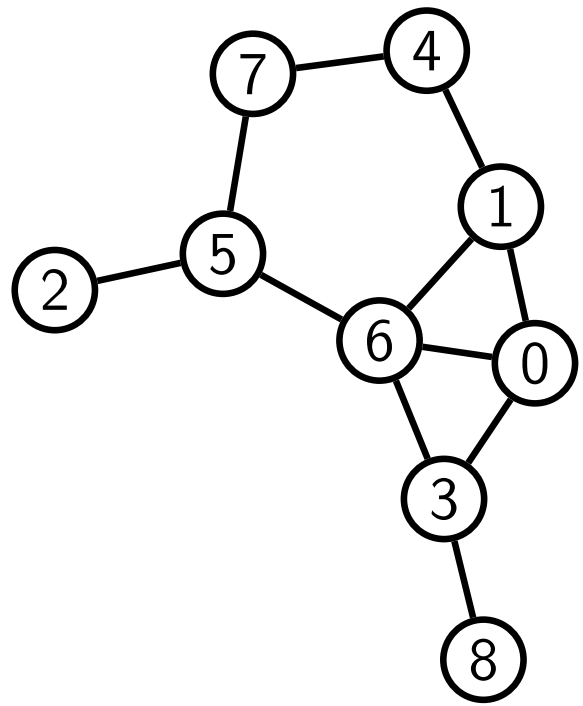
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

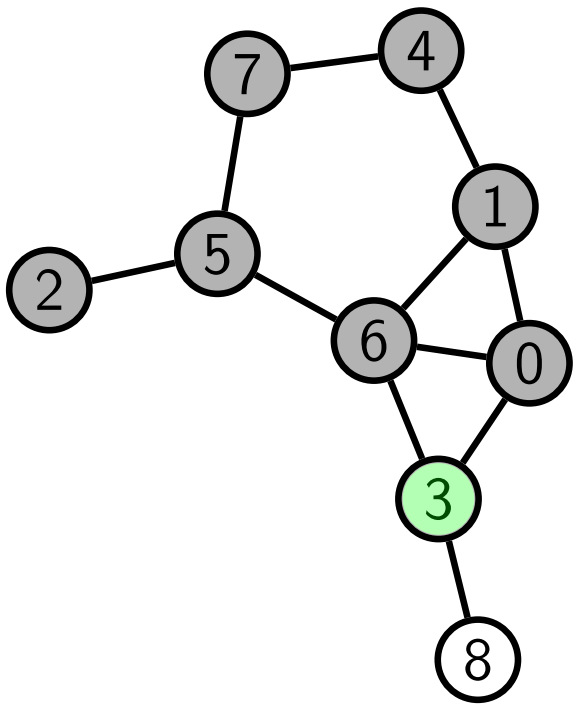


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

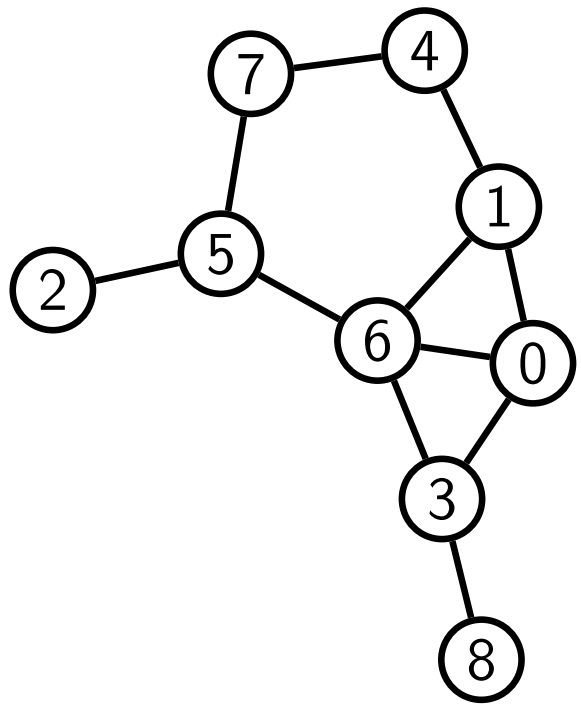
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

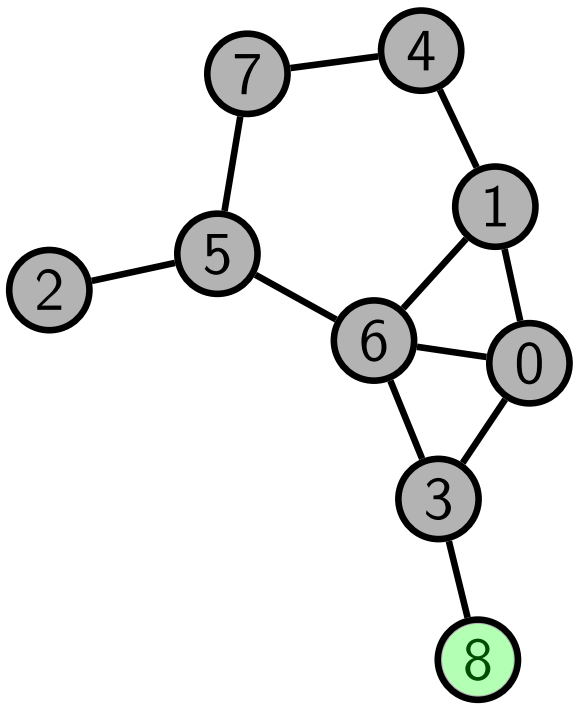


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

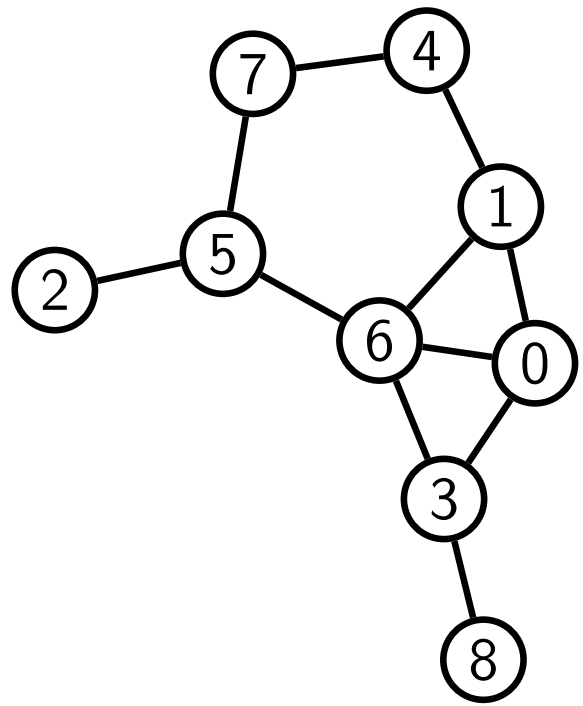
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

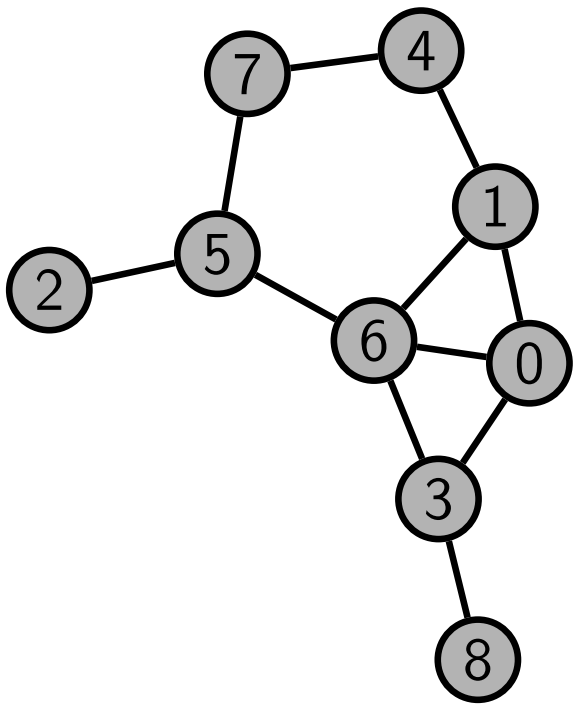


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

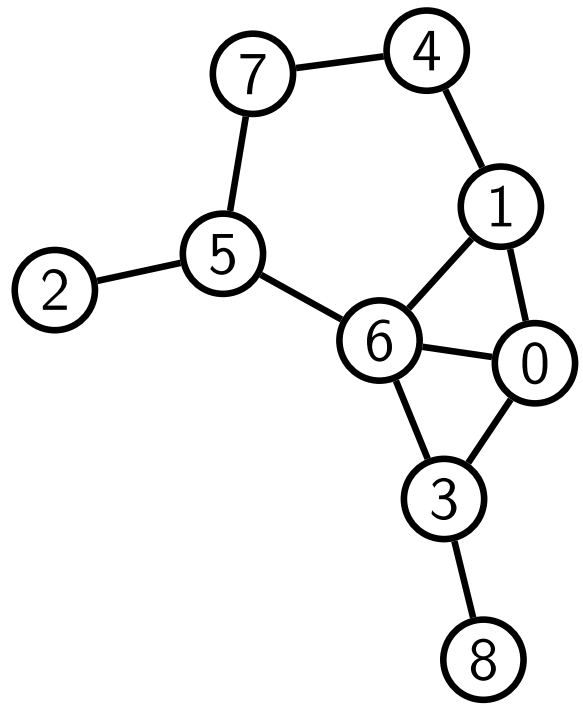
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

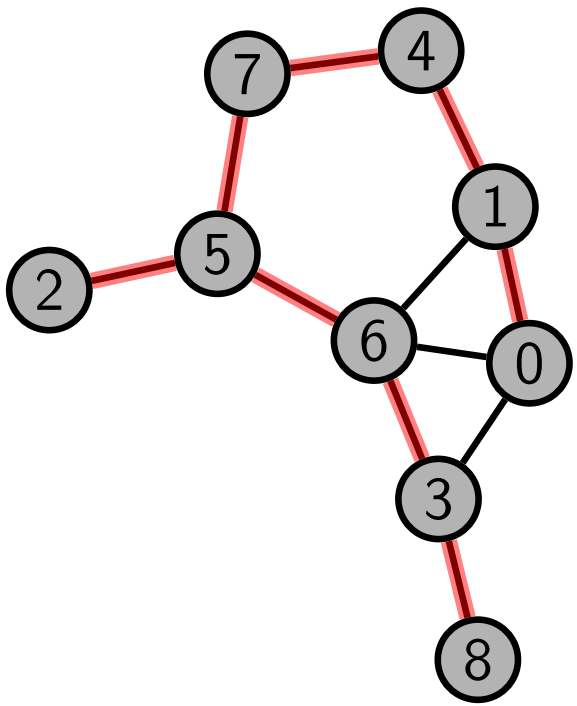


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

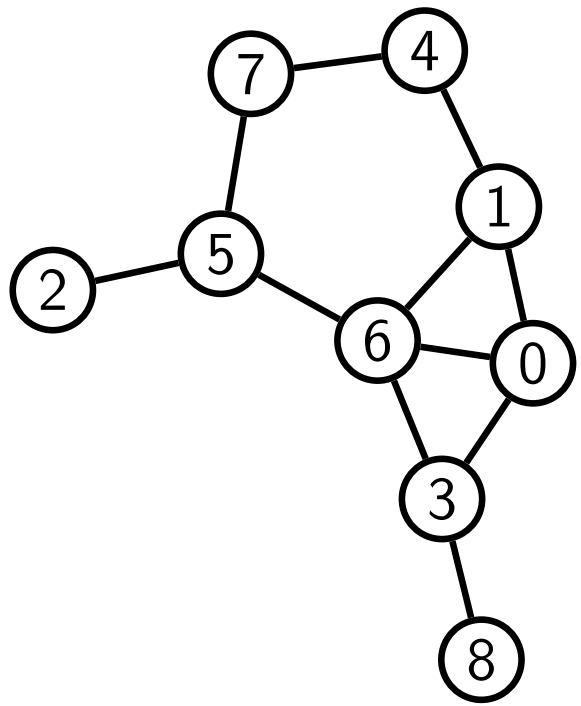
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

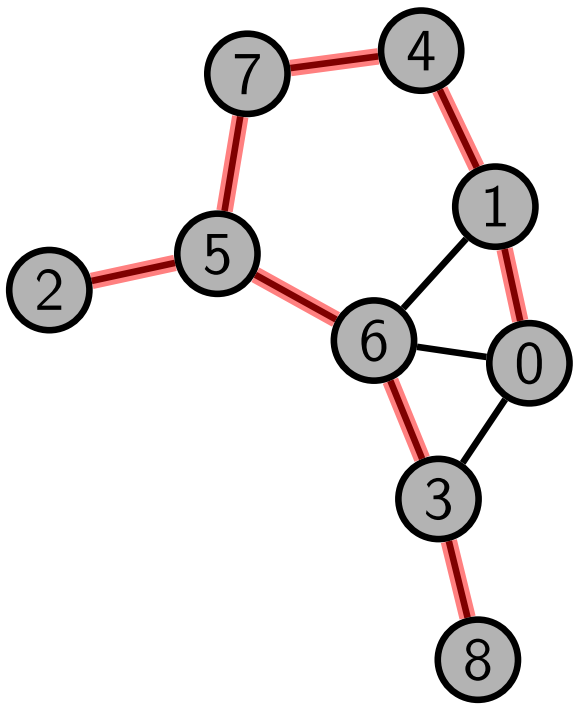


	0	1	2	3	4	5	6	7	8
parents :									

Calcul des arbres de parcours

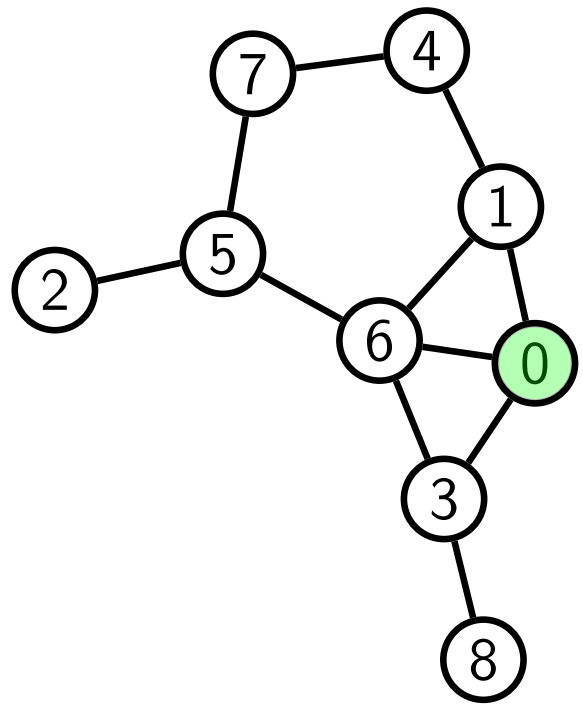
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

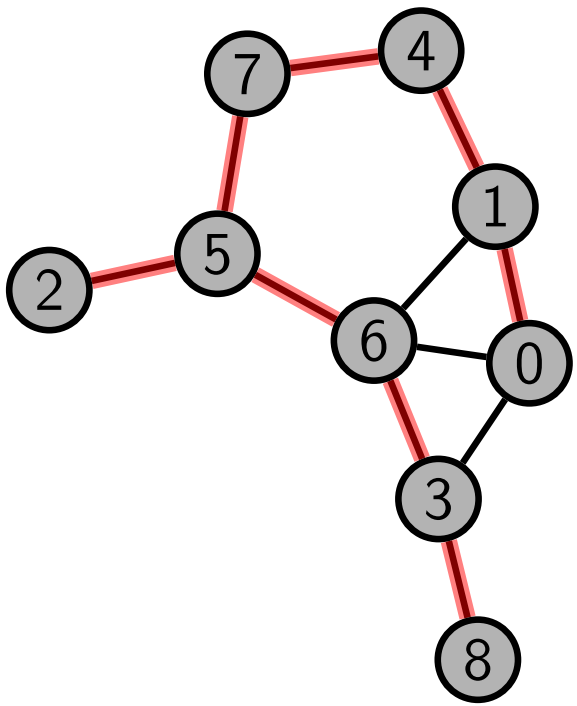


	0	1	2	3	4	5	6	7	8
parents :		0		0			0		

Calcul des arbres de parcours

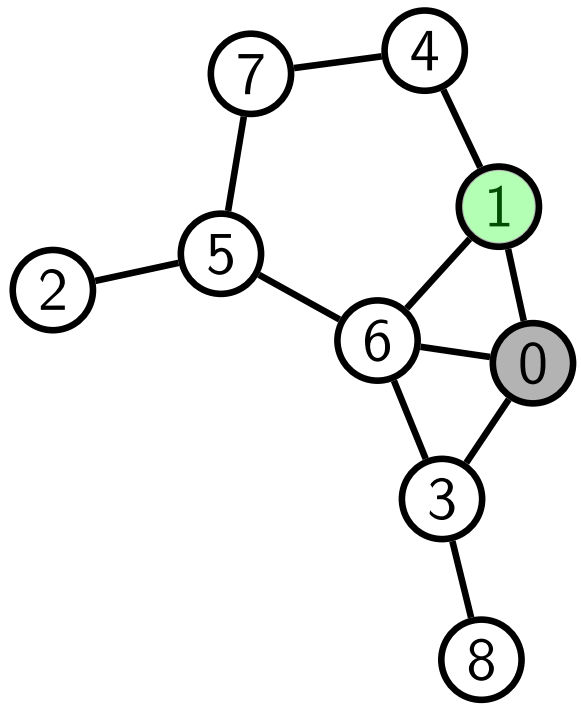
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

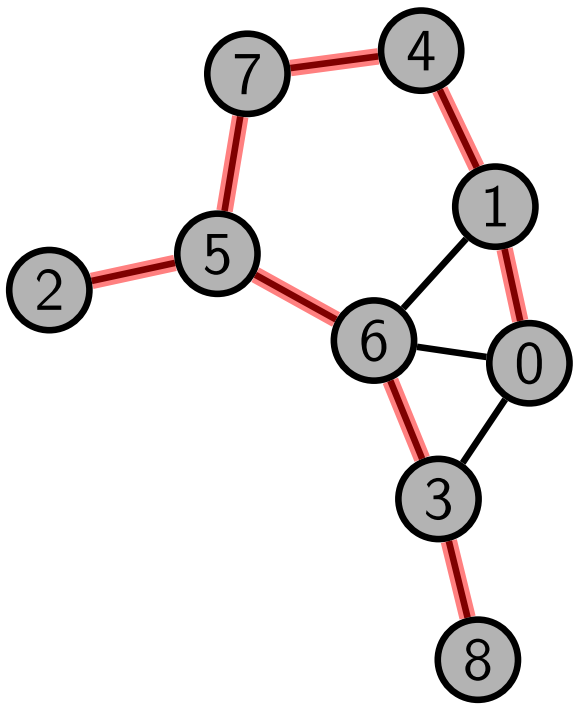


	0	1	2	3	4	5	6	7	8
parents :		0		0	1		0		

Calcul des arbres de parcours

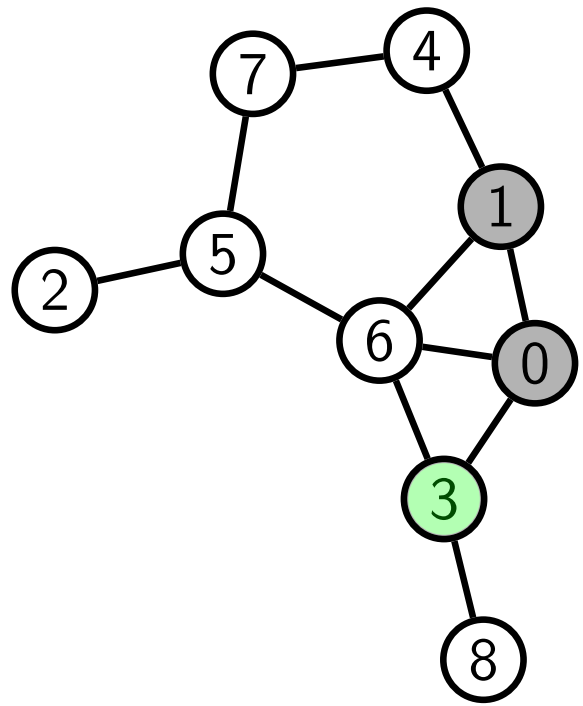
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

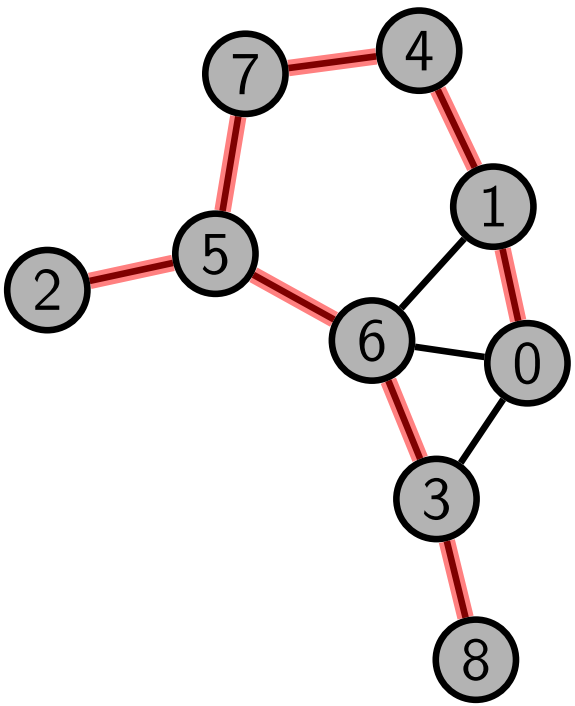


	0	1	2	3	4	5	6	7	8
parents :		0		0	1		0		3

Calcul des arbres de parcours

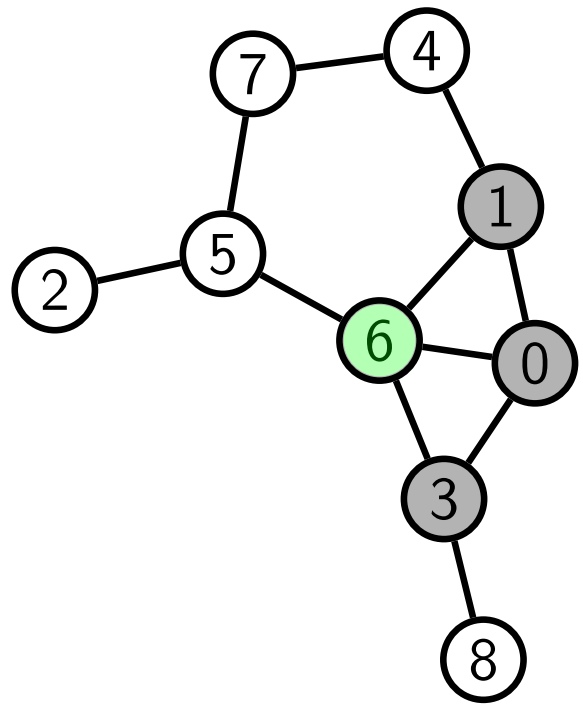
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

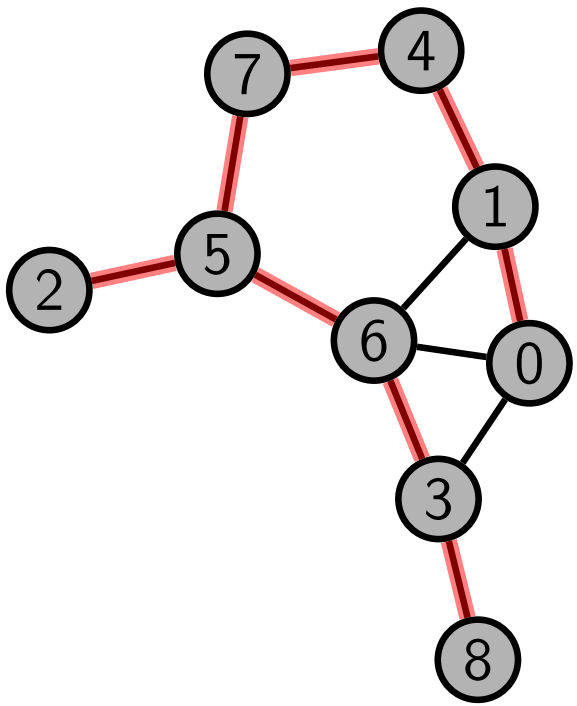


	0	1	2	3	4	5	6	7	8
parents :		0		0	1	6	0		3

Calcul des arbres de parcours

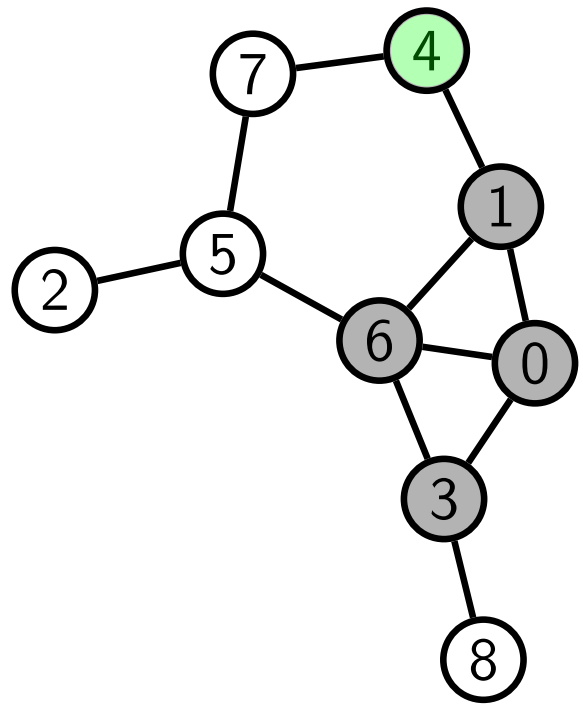
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

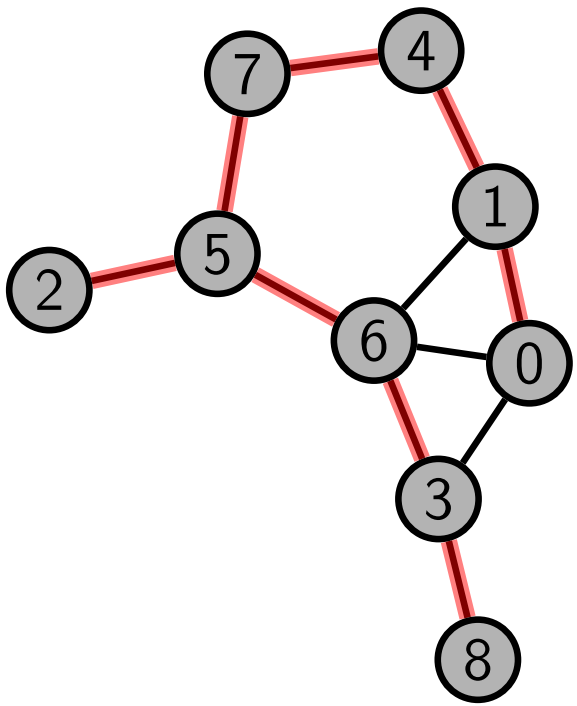


	0	1	2	3	4	5	6	7	8
parents :		0		0	1	6	0	4	3

Calcul des arbres de parcours

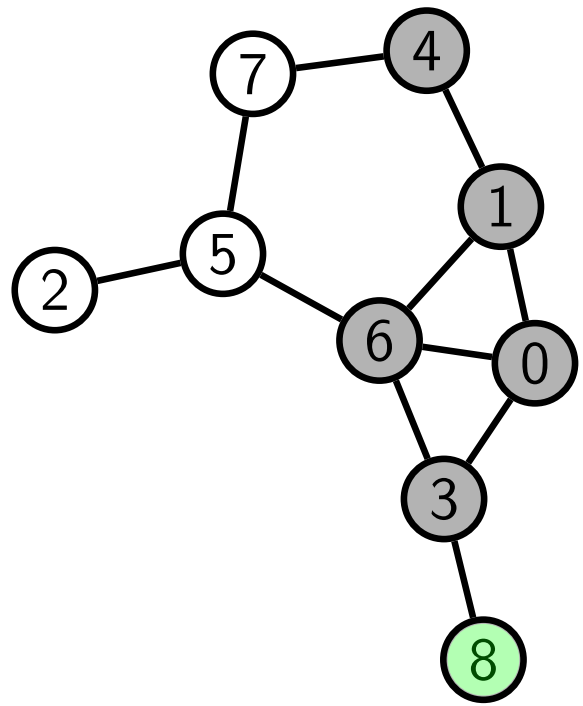
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

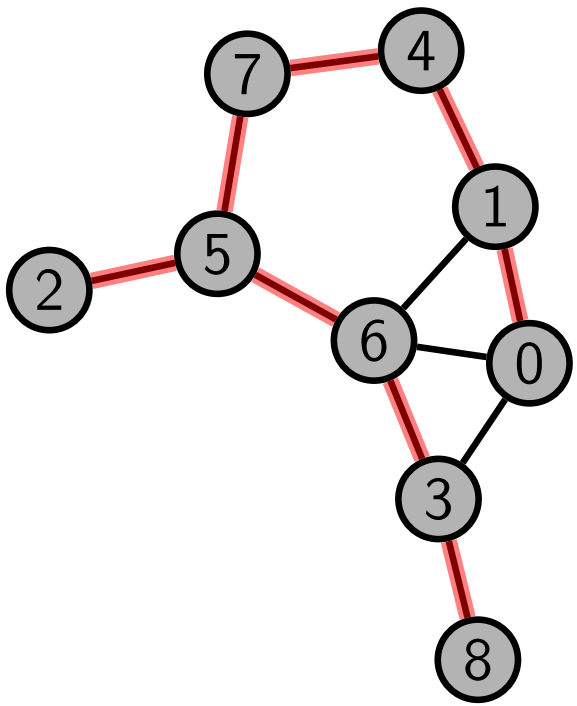


	0	1	2	3	4	5	6	7	8
parents :		0		0	1	6	0	4	3

Calcul des arbres de parcours

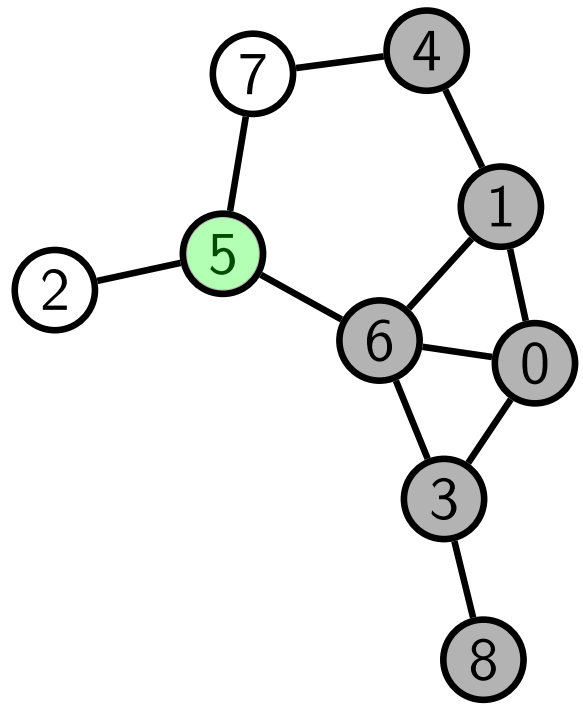
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

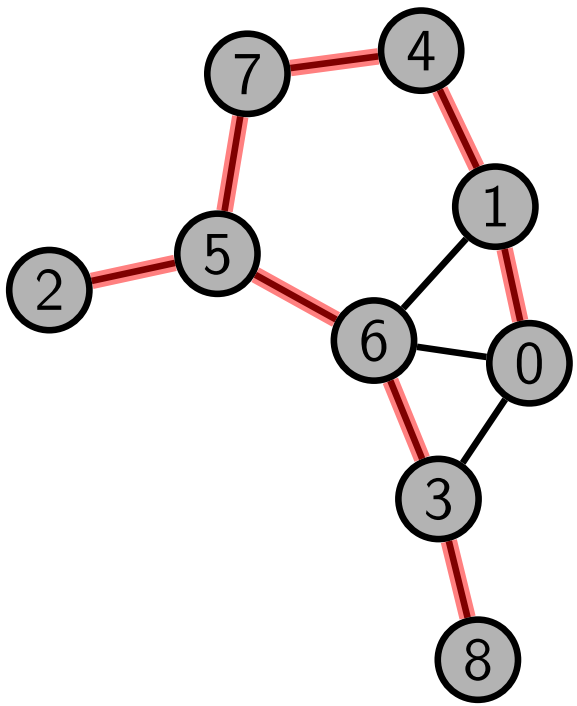


	0	1	2	3	4	5	6	7	8
parents :		0	5	0	1	6	0	4	3

Calcul des arbres de parcours

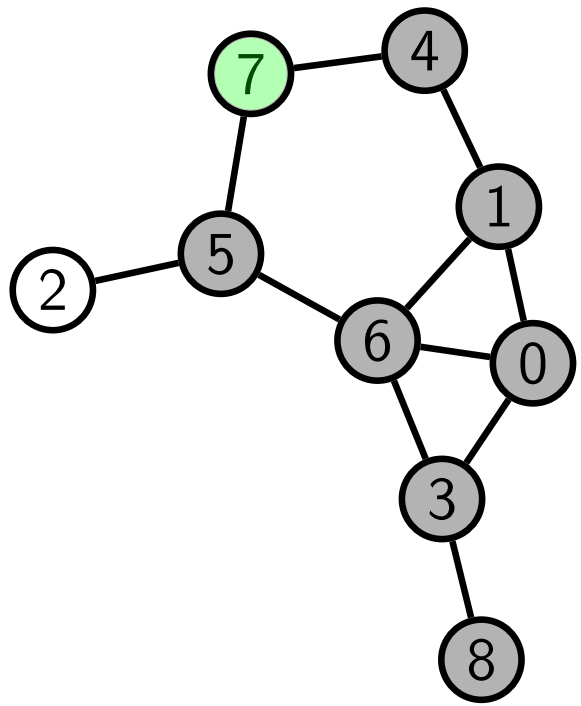
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

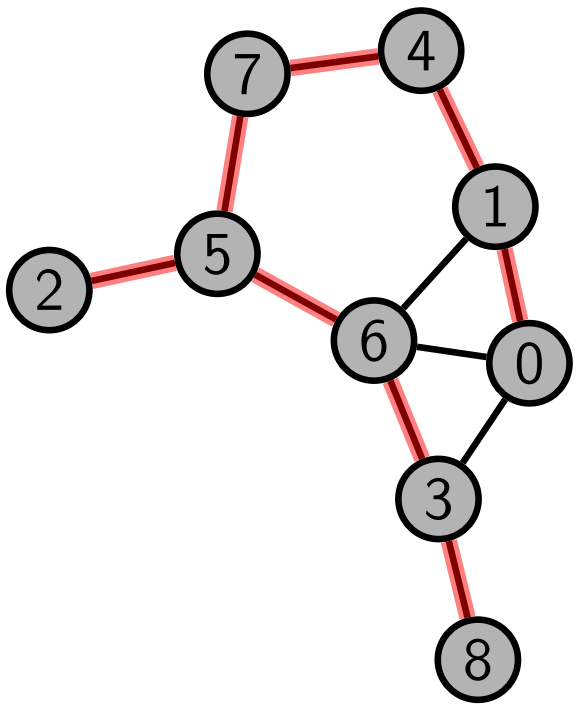


	0	1	2	3	4	5	6	7	8
parents :		0	5	0	1	6	0	4	3

Calcul des arbres de parcours

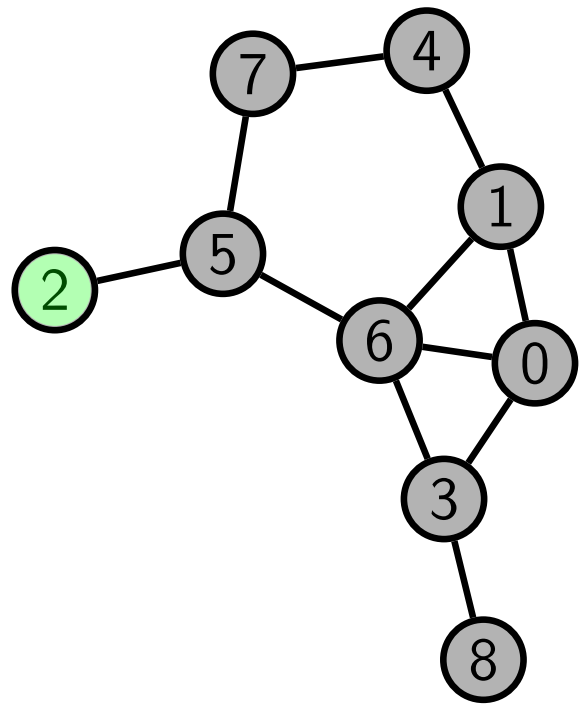
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

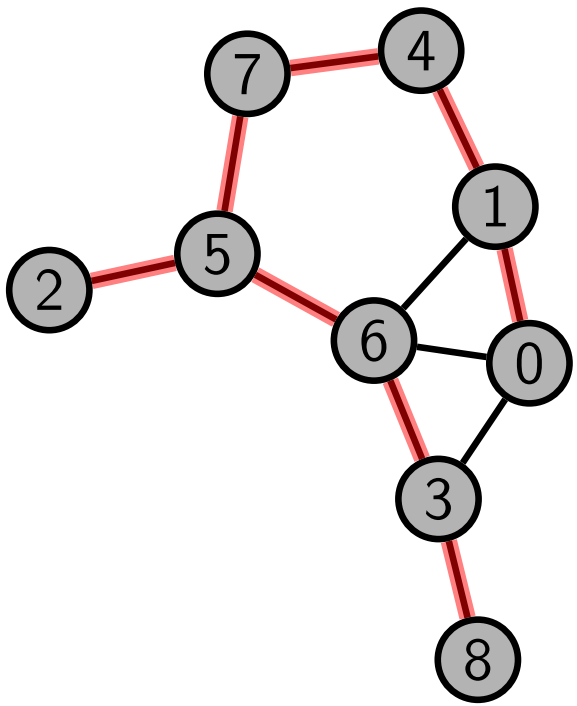


	0	1	2	3	4	5	6	7	8
parents :		0	5	0	1	6	0	4	3

Calcul des arbres de parcours

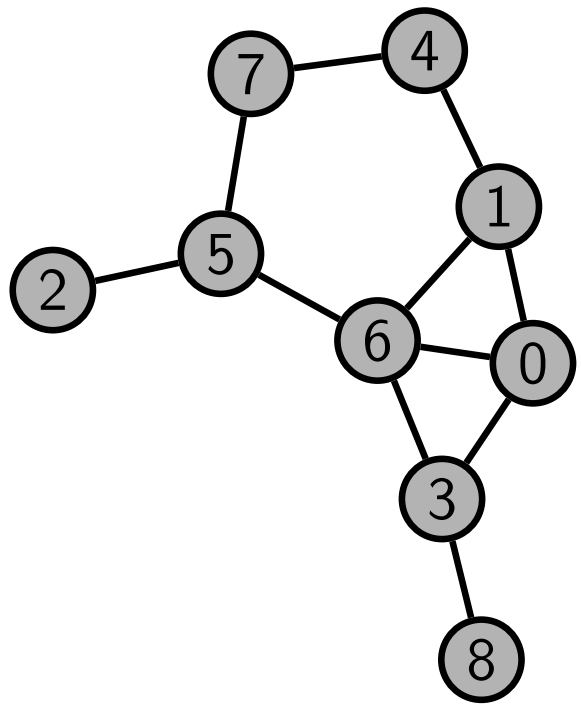
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

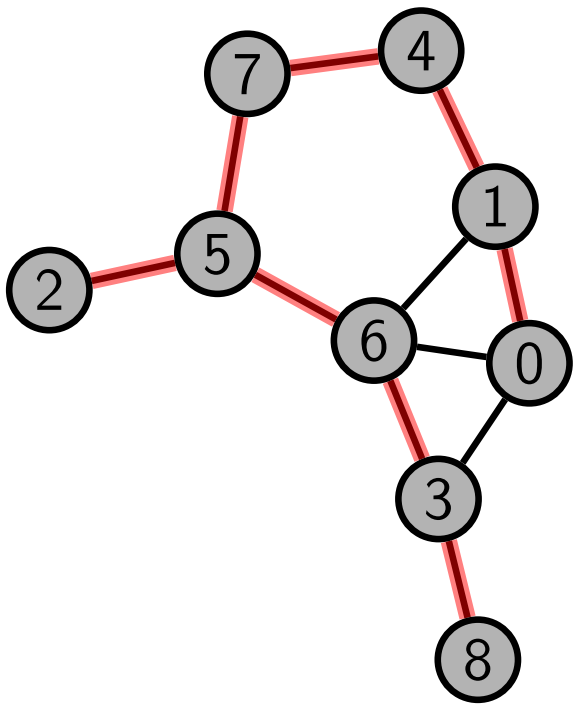


	0	1	2	3	4	5	6	7	8
parents :		0	5	0	1	6	0	4	3

Calcul des arbres de parcours

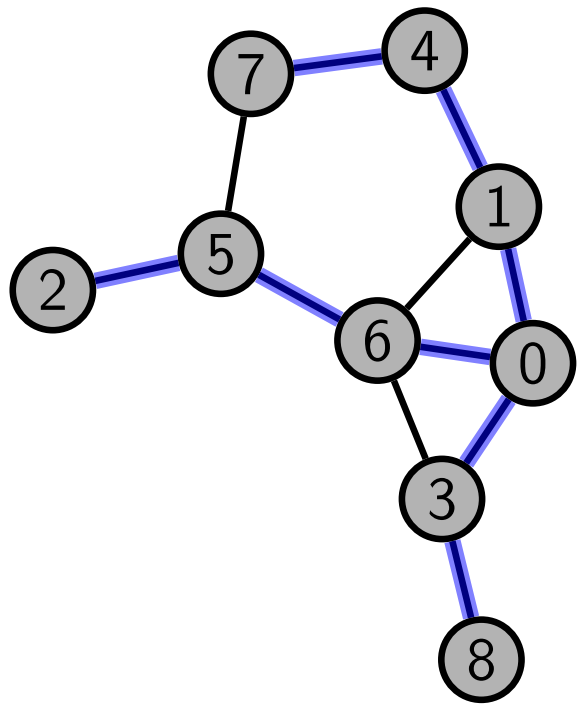
Pour calculer ces arbres, on doit stocker le parent de chaque sommet dans l'exploration.

Exemple 9 (profondeur)



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

Exemple 10 (largeur)

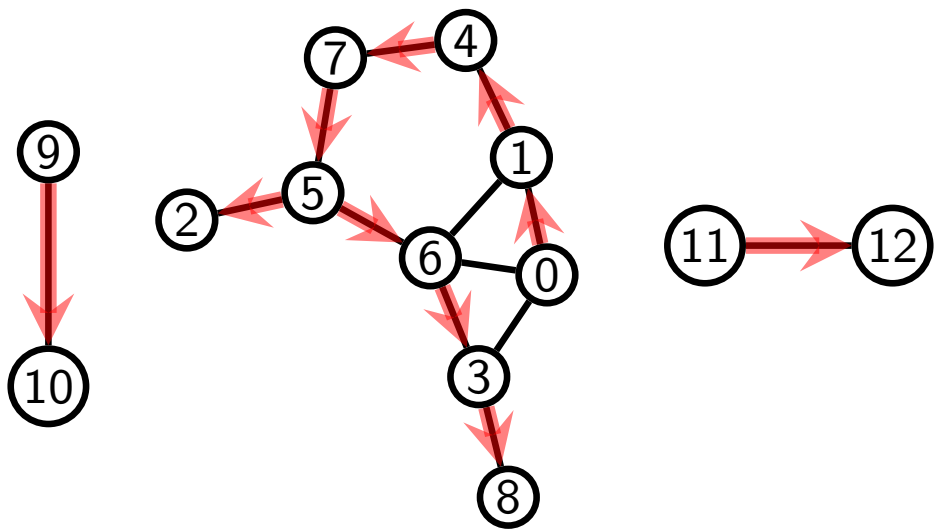


	0	1	2	3	4	5	6	7	8
parents :		0	5	0	1	6	0	4	3

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 11 (forêt couvrante)



Connexité et composantes connexes

Un graphe **connexe** est un graphe "en un seul morceau" : il existe un chemin entre deux sommets quelconques.

Définition 3

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 3 : 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 4

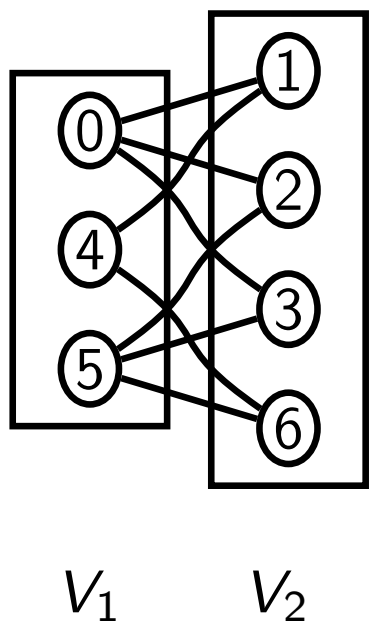
Un graphe $G = (V, E)$ est **biparti** s'il existe une bipartition $V = V_1 \cup V_2$ telle que deux sommets de la même partie ne sont jamais adjacents.

Graphes bipartis

Définition 4

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 12

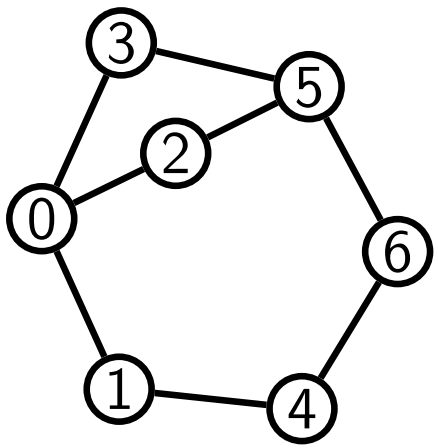


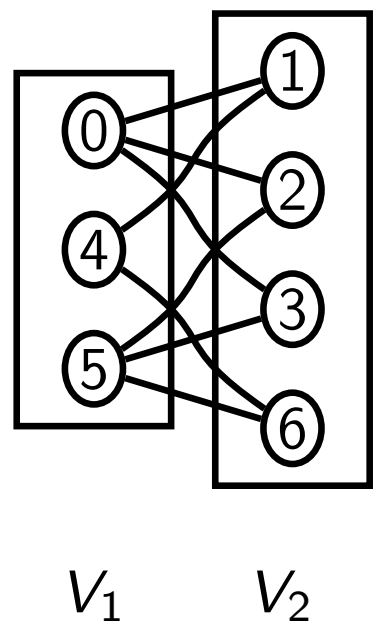
Graphes bipartis

Définition 4

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 12



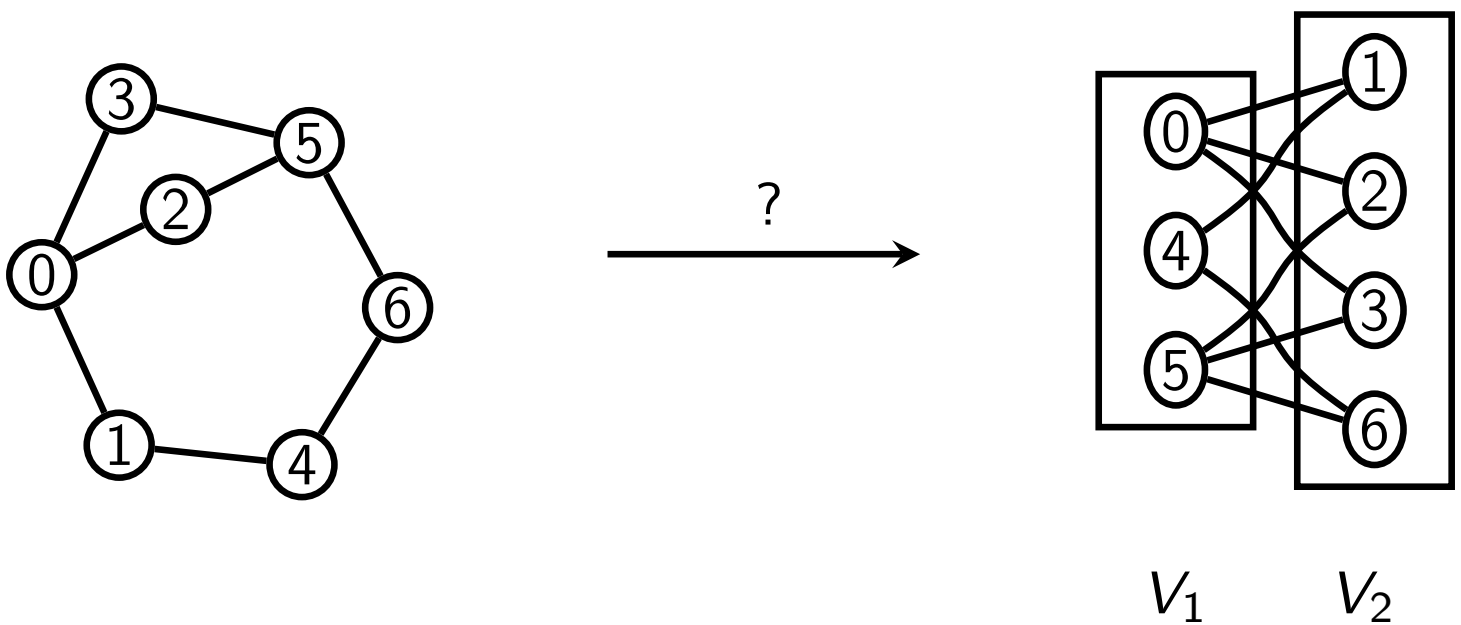


Graphes bipartis

Définition 4

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 12



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 5

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

Démonstration.

□

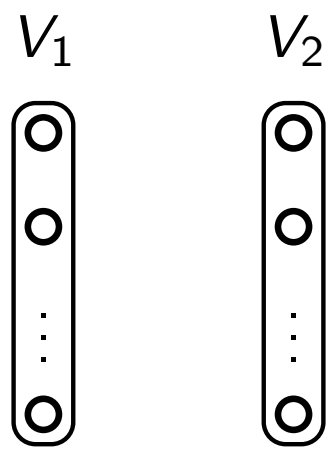
Caractérisation des graphes bipartis

Théorème 5

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. □

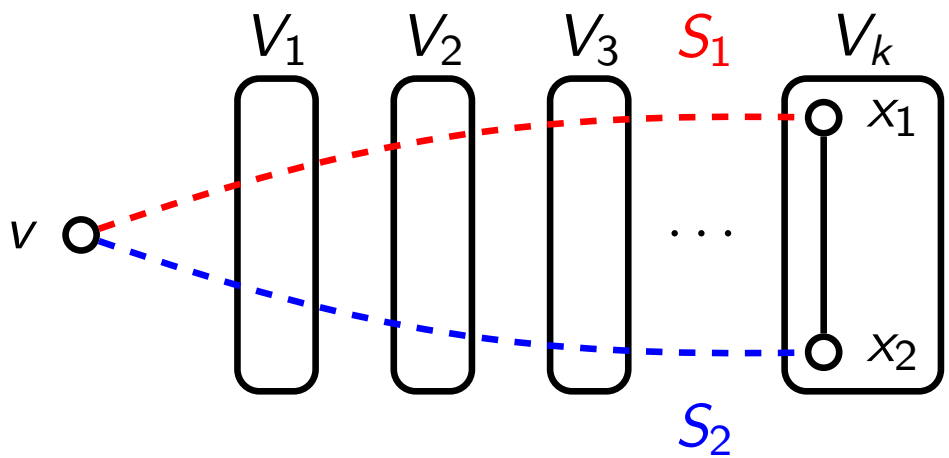
Caractérisation des graphes bipartis

Théorème 5

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

Démonstration.

$\Leftarrow$ "si G n'a pas de cycle impair il est biparti ".



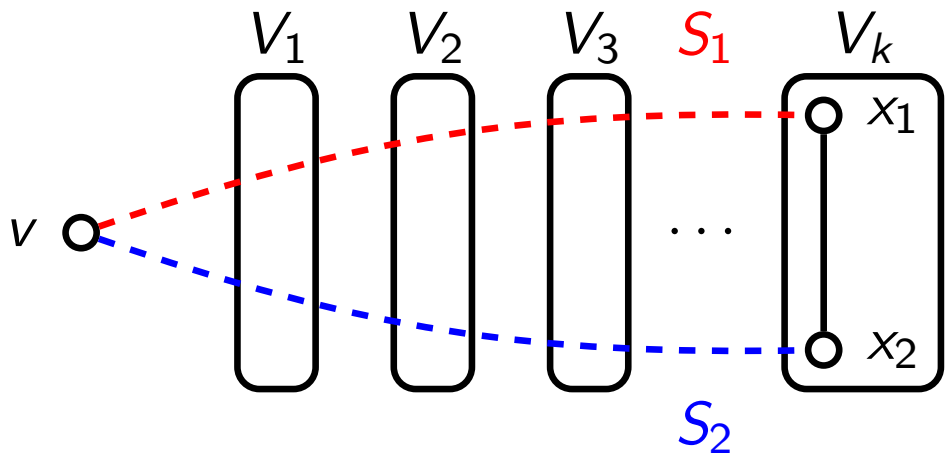
Caractérisation des graphes bipartis

Théorème 5

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

Démonstration.

$\Leftarrow$ "si G n'a pas de cycle impair il est biparti ".



On part de v et on place ses voisins dans V_1, les voisins de V_1 dans V_2 etc .... Deux sommets d'un même V_i ne sont pas reliés. Deux sommets $u \in V_i$, $v \in V_j$ avec i, j de même parité ne sont pas reliés. □

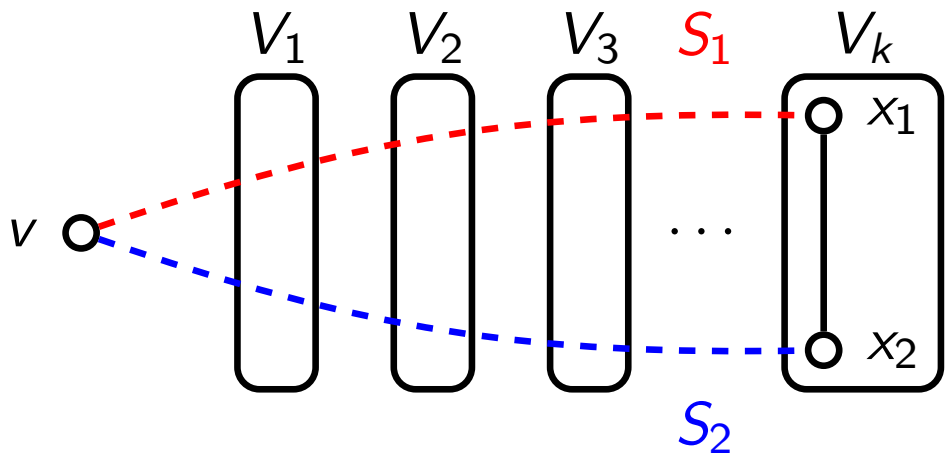
Caractérisation des graphes bipartis

Théorème 5

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

Démonstration.

$\Leftarrow$ "si G n'a pas de cycle impair il est biparti ".



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

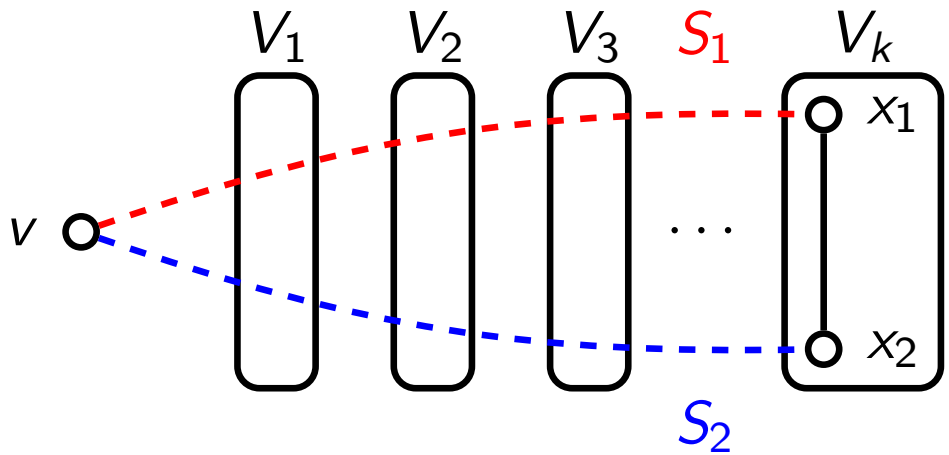
Caractérisation des graphes bipartis

Théorème 5

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

Démonstration.

$\Leftarrow$ "si G n'a pas de cycle impair il est biparti ".



Le graphe est bi-parti avec la bi-partition $S = \cup_{i\text{pair}} V_i$ et $T = \cup_{i\text{impair}} V_i$ pour la composante connexe de v. □

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 ;

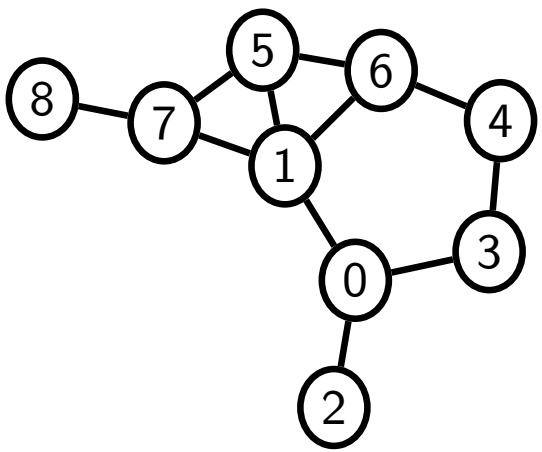
Algorithme de reconnaissance des graphes bipartis

- 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 ($V = V_1 \cup V_2$) ;

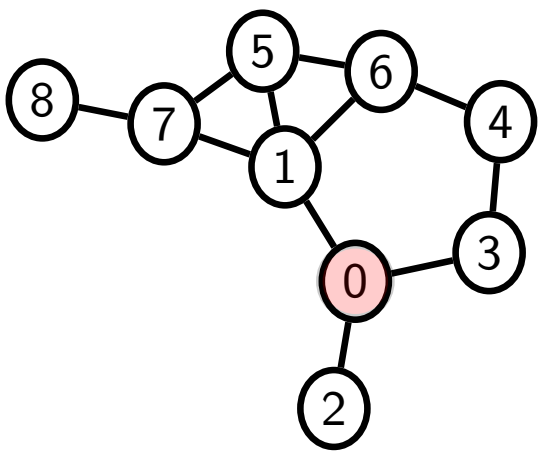
Exemple 13 (pas biparti)



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 ($V = V_1 \cup V_2$) ;

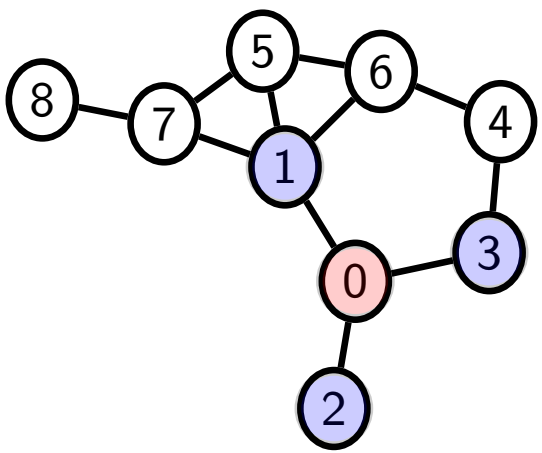
Exemple 13 (pas biparti)



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 ($V = V_1 \cup V_2$) ;

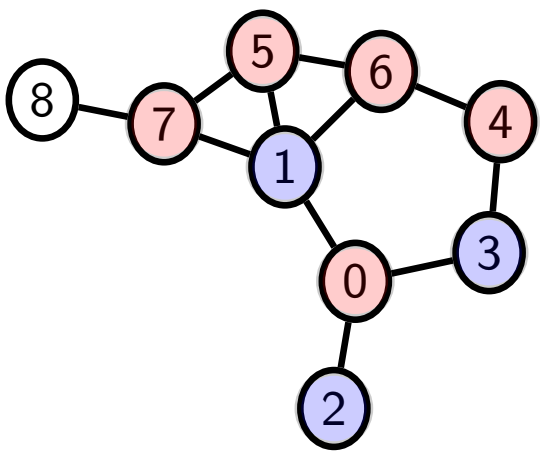
Exemple 13 (pas biparti)



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 ($V = V_1 \cup V_2$) ;

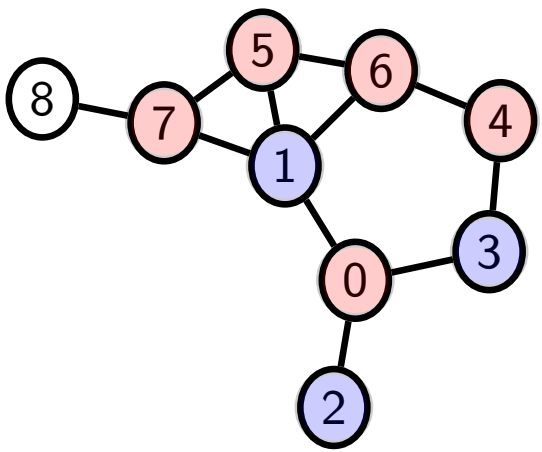
Exemple 13 (pas biparti)



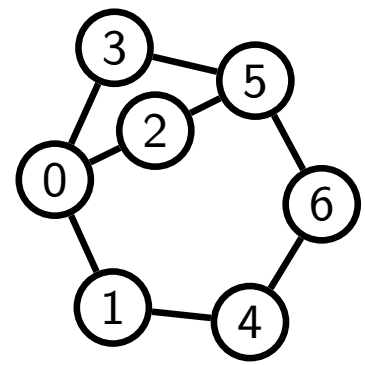
Algorithme de reconnaissance des graphes bipartis

- 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 13 (pas biparti)



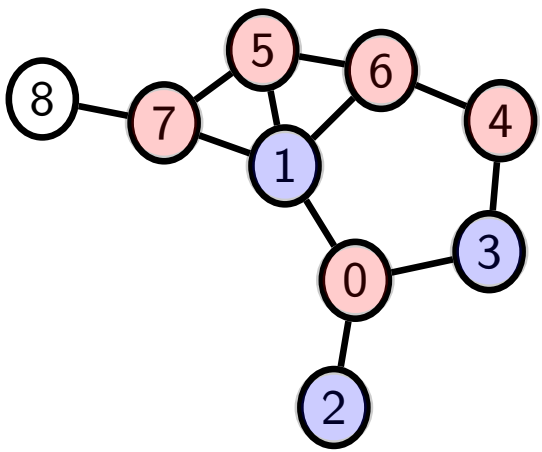
Exemple 14 (biparti)



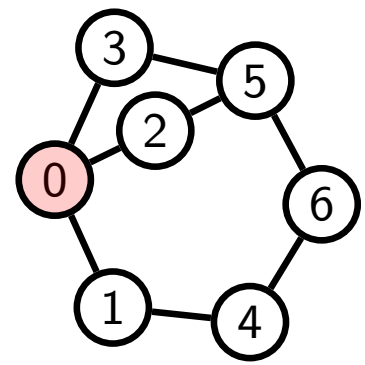
Algorithme de reconnaissance des graphes bipartis

- 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 13 (pas biparti)



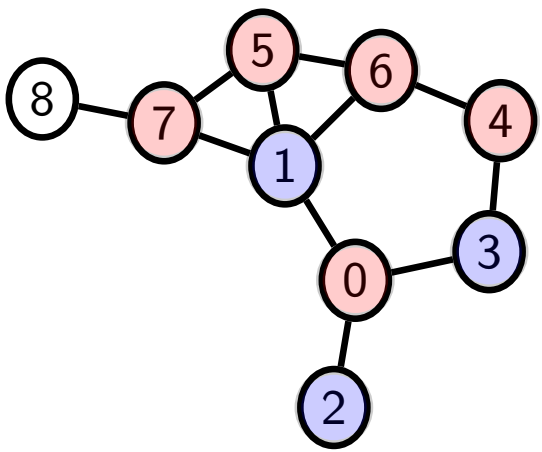
Exemple 14 (biparti)



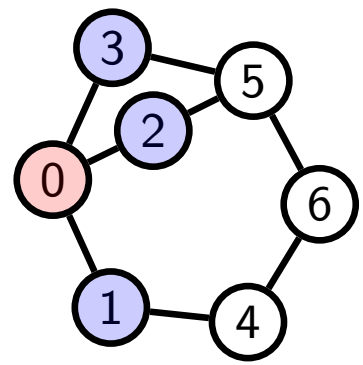
Algorithme de reconnaissance des graphes bipartis

- 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 13 (pas biparti)



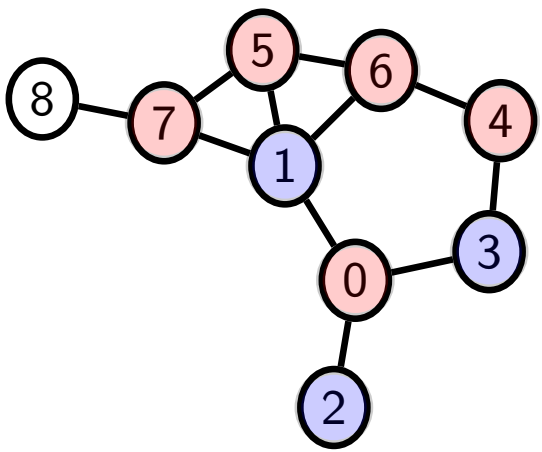
Exemple 14 (biparti)



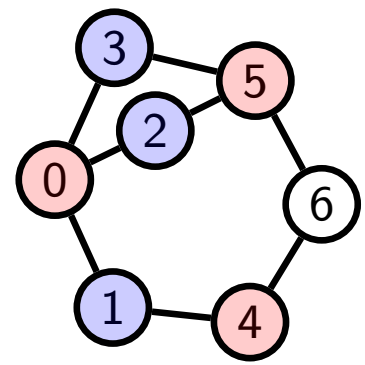
Algorithme de reconnaissance des graphes bipartis

- 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 13 (pas biparti)



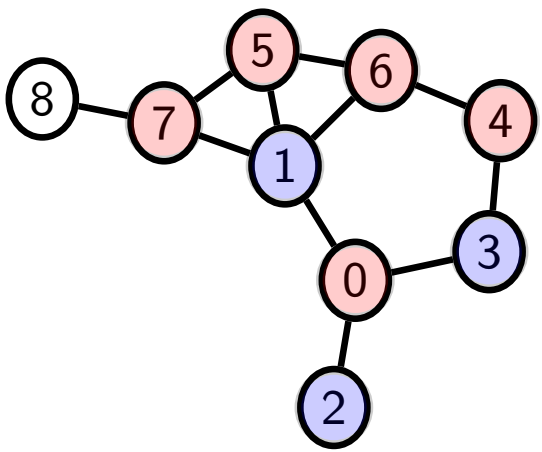
Exemple 14 (biparti)



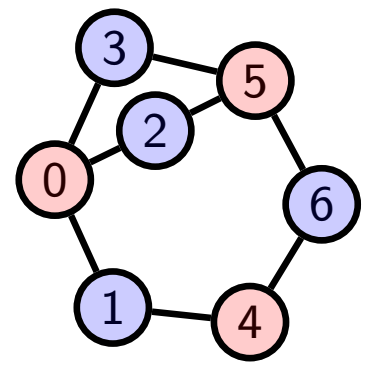
Algorithme de reconnaissance des graphes bipartis

- 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 13 (pas biparti)



Exemple 14 (biparti)



L'algorithme

Algorithme 4 : 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 le graphe est connexe et 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 le graphe est connexe et si on veut juste répondre "oui" ou "non" : comparer $|E|$ à $|V|$;
    - Si le graphe est connexe $|E| \geq |V| - 1$;

Détection de cycles

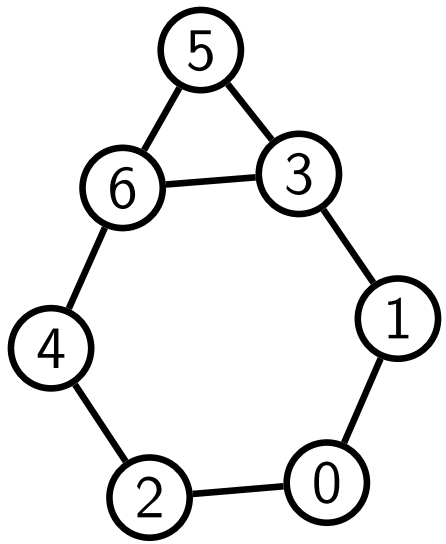
- On peut utiliser l'algorithme de reconnaissance de graphes bipartis pour savoir si le graphe contient un cycle impair ;
- Comment savoir si un graphe contient un cycle quelconque ? Deux options :
  - Si le graphe est connexe et si on veut juste répondre "oui" ou "non" : comparer $|E|$ à $|V|$;
    - Si le graphe est connexe $|E| \geq |V| - 1$;
    - Si le graphe est connexe sans cyclique (c'est-à-dire est un arbre) $|E| = |V| - 1$.

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 le graphe est connexe et si on veut juste répondre "oui" ou "non" : comparer $|E|$ à $|V|$;
    - Si le graphe est connexe $|E| \geq |V| - 1$;
    - Si le graphe est connexe sans cyclique (c'est-à-dire est un arbre) $|E| = |V| - 1$.
  - si on veut renvoyer un cycle explicite : parcourir le graphe et trouver un arc retour.

Détection de cycles

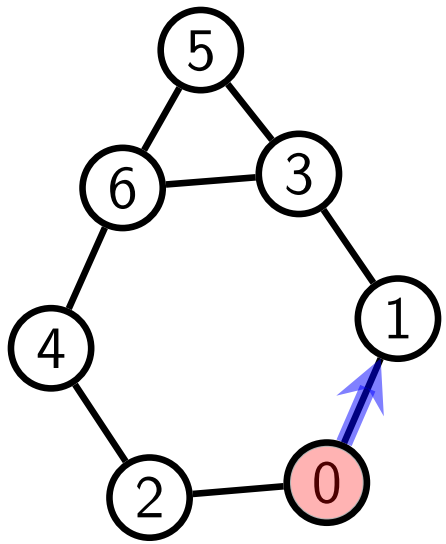
Exemple 15



On détecte un arc retour $(6, 3)$.

Détection de cycles

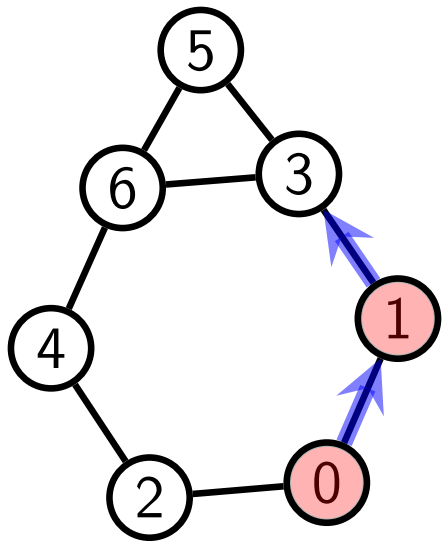
Exemple 15



On détecte un arc retour $(6, 3)$.

Détection de cycles

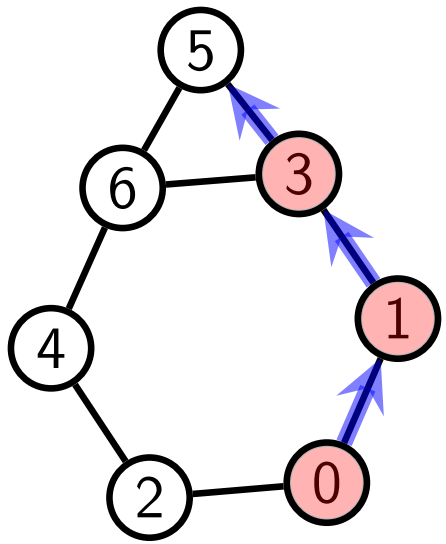
Exemple 15



On détecte un arc retour $(6, 3)$.

Détection de cycles

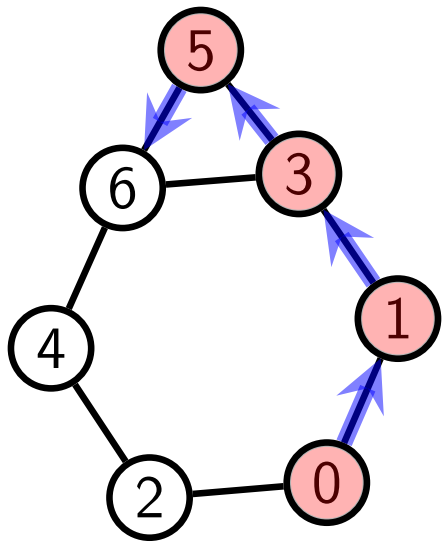
Exemple 15



On détecte un arc retour $(6, 3)$.

Détection de cycles

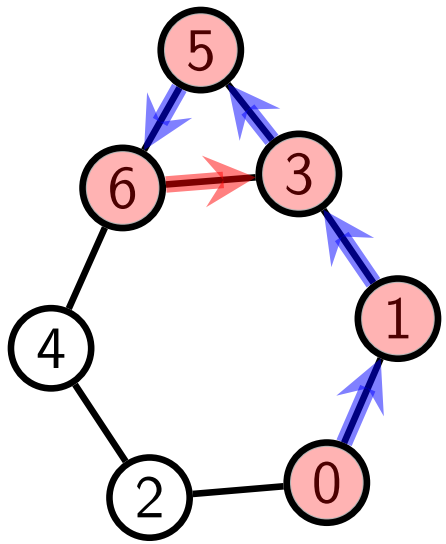
Exemple 15



On détecte un arc retour $(6, 3)$.

Détection de cycles

Exemple 15



On détecte un arc retour $(6, 3)$.

L'algorithme de Dijkstra pour les graphes non-orientés

C'est le même algorithme que pour les graphes orientés.

Les poids des arêtes doivent être positifs ou nuls.

Extraction du sommet le plus proche

L'extraction du minimum peut se faire à l'aide d'un algorithme naïf :

Algorithme 5 : ExtraireSommetLePlusProche(S, distances)

Entrées : un ensemble S de sommets, la distance de chaque sommet de S.
Résultat : le sommet de S le plus proche est extrait et renvoyé.

```
1 sommet ← NIL;
2 distance_min ← +∞;
3 pour chaque candidat ∈ S faire
4 |   si distances[candidat] < distance_min alors
5 |   |   sommet ← candidat;
6 |   |   distance_min ← distances[candidat];
7 si sommet ≠ NIL alors S ← S\ sommet ;
8 renvoyer sommet;
```

Extraction du sommet le plus proche

L'extraction du minimum peut se faire à l'aide d'un algorithme naïf :

Algorithme 5 : ExtraireSommetLePlusProche(S, distances)

Entrées : un ensemble S de sommets, la distance de chaque sommet de S.
Résultat : le sommet de S le plus proche est extrait et renvoyé.

```
1 sommet ← NIL;
2 distance_min ← +∞;
3 pour chaque candidat ∈ S faire
4 |   si distances[candidat] < distance_min alors
5 |   |   sommet ← candidat;
6 |   |   distance_min ← distances[candidat];
7 si sommet ≠ NIL alors S ← S\ sommet ;
8 renvoyer sommet;
```

Un tas comme représenter S dont clés sont les distances à s serait plus efficace.

L'algorithme de Dijkstra pour les graphes non-orientés

Algorithme 6 : DIJKSTRA(G, source)

Entrées : un graphe pondéré non orienté G, un sommet source.
Sortie : la longueur d'un plus court chemin de la source à chacun des sommets du graphe ($+\infty$ pour les sommets non accessibles).

```
1 a_traiter ← G.sommets();
2 distances ← tableau(G.nombre_sommets(), +∞);
3 distances[source] ← 0;
4 tant que a_traiter.pas_vide() faire
5 |   u ← ExtraireSommetLePlusProche(a_traiter, distances);
6 |   si u = NIL alors renvoyer distances ;
7 |   pour chaque v ∈ G.voisins(u) faire
8 |   |   distances[v] ← min(distances[v], distances[u] +
  |   |     G.poids_arête(u, v));
9 renvoyer distances;
```