Chapitre 2

Généralités sur les graphes

Introduction

De manière générale, un graphe permet de représenter les connexions d'un ensemble en exprimant les relations entre ses éléments : réseau de communication, réseau routier, circuit électronique, ... , mais aussi relations sociales ou interactions entre espèces animales.

Le vocabulaire de la théorie des graphes est utilisé dans de nombreux domaines : chimie, biologie, sciences sociales, etc., mais c'est avant tout une branche à part entière et déjà ancienne des mathématiques (le fameux problème des ponts de Königsberg d'Euler date de 1736). Néanmoins, l'importance accrue que revêt l'aspect algorithmique dans ses applications pratiques en fait aussi un domaine incontournable de l'informatique.

1 Graphes non orientés

1.1 Définitions et première propriété

Définition.

Un **graphe non orienté** G est un couple (S, A) où :

- S est un ensemble fini dont les éléments sont appelés les **sommets** de G :

$$S = \{s_1, s_2, \ldots, s_n\},$$

- A est un ensemble fini de paires non ordonnées d'éléments de S :

$$A = \{\{s_i, s_j\}, \ldots\}.$$

Les éléments de A sont appelés les **arêtes** de G.

Définition.

Soit $G = (S, A)$ un graphe non orienté.

- Si $a = \{x, y\}$ est une arête de G, alors les sommets x et y sont dits **adjacents** et ce sont les **extrémités** de l'arête a.
- Si x est un sommet de G, alors on appelle **voisin** de x tout sommet y de G tel que x et y soient adjacents.

Définition.

Soit $G = (S, A)$ un graphe non orienté.

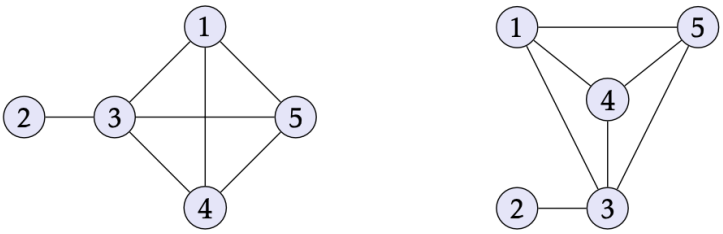
- On appelle **ordre** du graphe G le nombre de ses sommets, c'est-à-dire $n = |S|$.
- On appelle **degré** d'un sommet x de G et on note $\deg(x)$ le nombre de voisins de x :

$$\deg(x) = |\{y \in S \mid \{x, y\} \in A\}|.$$

Remarques.

1. Les graphes tirent leur nom du fait qu'on peut les représenter graphiquement : à chaque sommet de G, on fait correspondre un point du plan et on relie les points correspondant aux extrémités de chaque arête. Il existe donc une infinité de représentations possibles.

   Par exemple, les deux dessins qui suivent représentent le même graphe non orienté $G = (S, A)$, avec $S = \{1, 2, 3, 4, 5\}$ et $A = \{\{1, 3\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\}$:



   G est un graphe non orienté d'ordre 5 ; il possède un sommet de degré 1 (le sommet 2), trois sommets de degré 3 (les sommets 1, 4 et 5) et un sommet de degré 4 (le sommet 3).

2. On peut observer que ce graphe a la particularité de posséder une représentation (celle de droite) pour laquelle les arêtes ne se coupent pas. Un tel graphe est dit **planaire**. Cette notion ne sera pas abordée dans la suite de ce cours.

3. Un graphe non orienté est dit **simple** lorsqu'aucun sommet n'est adjacent à lui-même. Par la suite, nous nous restreindrons à l'étude des graphes simples.

Propriété 1 (Formule d'Euler)

Si $G = (S, A)$ est un graphe non orienté, alors

$$\sum_{s \in S} \deg(s) = 2\,|A|.$$

Preuve. Une arête est comptée deux fois dans le calcul de la somme des degrés des sommets : une fois pour chacun des sommets qu'elle relie. □

1.2 Chemins et connexité

Définition.

Soit $G = (S, A)$ un graphe non orienté et x et y deux sommets de G.
On appelle **chemin** reliant les sommets x et y toute suite de sommets :

$$x = s_0, s_1, s_2, \ldots, s_{k-1}, s_k = y$$

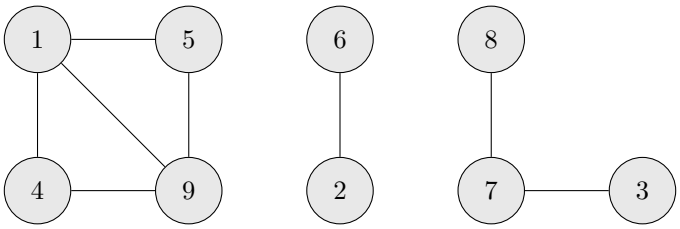
où k est un entier naturel non nul et, pour tout $i \in [\![1, k]\!]$, $\{s_{i-1}, s_i\}$ est une arête de G.
L'entier k est appelé la **longueur** de ce chemin, égale au nombre d'arêtes traversées pour aller de x à y.

Définition.

Soit $G = (S, A)$ un graphe non orienté et x et y deux sommets de G.
Nous définissons la distance $d(x, y)$ de x et y par :

- $+\infty$ dans le cas où il n'existe pas de chemin reliant x et y ;
- la longueur du plus court chemin reliant x et y dans le cas où il existe au moins un chemin reliant x et y.

Exemple. Considérons le graphe suivant :



Dans ce graphe,

- $5 - 1 - 4 - 9$ est une chemin de longueur 3 reliant les sommets 5 et 9 ;
- $d(5, 9) = 1$, $d(8, 3) = 2$ et $d(1, 2) = +\infty$.

Définition.

Un graphe non orienté est dit **connexe** si tous les sommets sont à une distance finie les uns des autres.

Remarque. Dans un graphe non orienté et connexe, tous les sommets sont de degré supérieur ou égal à 1.

Attention.

La réciproque est fausse : un graphe peut être non connexe et avoir tous ses sommets de degré supérieur ou égal à 1 (par exemple le graphe de l'exemple ci-dessus).

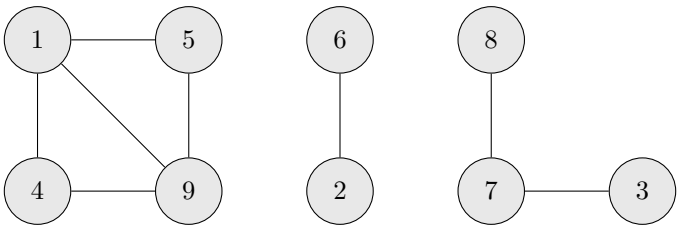
Définition.

Soit $G = (S, A)$ un graphe non orienté.

- Un **sous-graphe** de G est un graphe $G' = (S', A')$ tel que $S' \subset S$ et $A' \subset A \cap \mathscr{P}(S')$.
- Une **composante connexe** de G est un sous-graphe connexe de G qui est maximal au sens de l'inclusion pour cette propriété.

Remarque. Un graphe est connexe si et seulement si il possède une unique composante connexe.

Exemple. Le graphe



possède trois composantes connexes.

Propriété 2 (Nombre d'arêtes et connexité)

Un graphe non orienté, connexe et d'ordre n admet au moins $n-1$ arêtes.

Preuve.

□

1.3 Acyclicité

Définition.

Soit $G = (S, A)$ un graphe non orienté.

- Un **cycle** de G est un chemin reliant un sommet à lui-même et ne comportant pas deux fois la même arête.
- Le graphe G est dit **acyclique** s'il n'existe pas de cycle de G.

Propriété 3 (Condition suffisante d'existence d'un cycle)

Soit $G = (S, A)$ un graphe non orienté.
Si tous les sommets de G sont de degré supérieur ou égal à 2, alors G possède au moins un cycle.

Preuve.

□

Propriété 4 (Nombre d'arêtes et acyclicité)

Un graphe non orienté, acyclique et d'ordre n admet au plus $n - 1$ arêtes.

Preuve.

□

1.4 Arbres

Définition.

Soit $G = (S, A)$ un graphe non orienté.
On dit que G est un **arbre** si G est connexe et acyclique.

Remarques.

1. Un graphe acyclique mais non connexe est appelé une **forêt**, chacune de ses composantes connexes étant un arbre.
2. D'après les propriétés précédentes, un arbre d'ordre n possède exactement $n-1$ arêtes. Plus précisément, on démontre le résultat suivant :

Théorème 5 (Caractérisations des arbres)

Soit $G = (S, A)$ un graphe non orienté d'ordre n. Les assertions suivantes sont équivalentes :

(1) G est un arbre ;

(2) G est un graphe connexe à $n-1$ arêtes ;

(3) G est un graphe acyclique à $n-1$ arêtes.

Remarque. Un arbre est donc un graphe connexe minimal et un graphe acyclique maximal.

Preuve.

□

2 Graphes orientés

2.1 Définitions

Définition.

Un **graphe orienté** G est un couple (S, A) où :

- S est un ensemble fini dont les éléments sont appelés les **sommets** de G :
$$S = \{s_1, s_2, \ldots, s_n\},$$
- A est une partie de S^2 :
$$A = \{(s_i, s_j), \ldots\}.$$
Les éléments de A sont appelés les **arcs** de G.

Définition.

Soit $G = (S, A)$ un graphe orienté.

- Si $a = (x, y)$ un arc de G, alors les sommets x et y sont les **extrémités** de a, x est l'**origine** de a et y est la **destination** de a.
- Si x est un sommet de G, alors on appelle **voisin** de x tout sommet y de G tel que (x, y) soit un arc de G.

Définition.

Soit $G = (S, A)$ un graphe orienté.

- On appelle **ordre** du graphe G le nombre de ses sommets, c'est-à-dire $n = |S|$.
- On appelle **degré sortant** d'un sommet x de G et on note $\deg_+(x)$ l'entier :

$$\deg_+(x) = |\{y \in S \mid (x, y) \in A\}|.$$

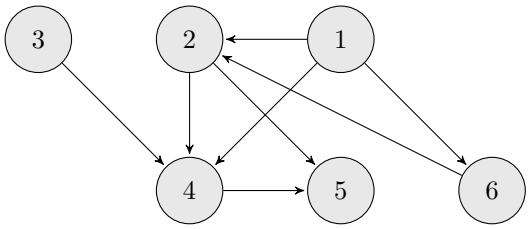
- On appelle **degré entrant** d'un sommet x de G et on note $\deg_-(x)$ l'entier :

$$\deg_-(x) = |\{y \in S \mid (y, x) \in A\}|.$$

Remarques.

1. Pour représenter un graphe orienté, nous associons à chaque sommet un point du plan et on relie par une flèche orientée les points correspondant aux extrémités de chaque arc.

   Par exemple, on peut représenter le graphe orienté $G = (S, A)$ avec $S = \{1, 2, 3, 4, 5, 6\}$ et $A = \{(1, 2), (1, 4), (1, 6), (2, 4), (2, 5), (3, 4), (4, 5), (6, 2)\}$ par :



   Le sommet 1 a un degré sortant égal à 3 et un degré entrant égal à 0, tandis que le sommet 2 a un degré entrant et un degré sortant tous deux égaux à 2.

2. Un graphe orienté est dit **simple** lorsqu'aucun sommet n'est voisin de lui-même. Comme dans le cas non orienté, nous nous restreindrons par la suite à l'étude des graphes simples.

2.2 Chemins et connexité

Définition.

Soit $G = (S, A)$ un graphe orienté et x et y deux sommets de G.

On appelle **chemin** reliant les sommets x et y toute suite de sommets :

$$x = s_0, s_1, s_2, \ldots, s_{k-1}, s_k = y$$

où k est un entier naturel non nul et, pour tout $i \in [\![1, k]\!]$, (s_{i-1}, s_i) est un arc de G.

L'entier k est appelé la **longueur** de ce chemin, égale au nombre d'arcs traversées pour aller de x à y.

Remarque. On étend sans difficulté les notions de distance et de cycle au cas des graphes orientés.

Définition.

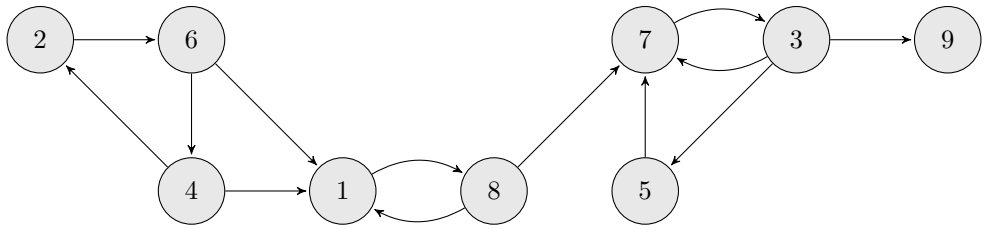
Un graphe orienté est dit **fortement connexe** lorsque pour tous sommets distincts x et y de G, il existe un chemin reliant x et y.

Définition.

Soit $G = (S, A)$ un graphe orienté.

- Un **sous-graphe** de G est un graphe $G' = (S', A')$ tel que $S' \subset S$ et $A' \subset A \cap (S' \times S')$.
- Une **composante fortement connexe** de G est un sous-graphe fortement connexe de G qui est maximal au sens de l'inclusion pour cette propriété.

Exercice. Déterminer (en les entourant) les composantes fortement connexes du graphe suivant :



2.3 Enracinement d'un arbre

Revenons au cas d'un arbre, c'est-à-dire d'un graphe non orienté, connexe et acyclique.

Propriété 6 (Unicité des chemins dans un arbre)

Soit $G = (S, A)$ un arbre.
Pour tous sommets distincts x et y de G, il existe un unique chemin reliant x et y.

Preuve.

□

Une conséquence de ce résultat est le :

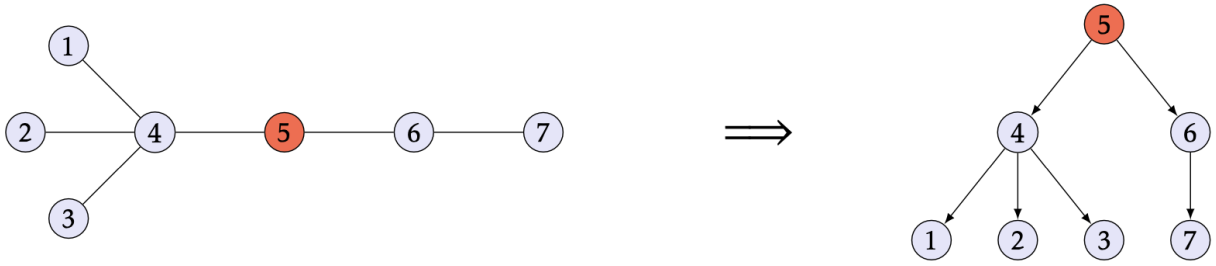
Théorème 7 (d'enracinement)

Soit $G = (S, A)$ un arbre et soit r un sommet arbitraire de G.
Il est possible d'orienter les arêtes de G de telle sorte qu'il existe un unique chemin reliant r à tous les sommets de G.

Preuve.

□

Exemple. Un exemple d'enracinement :



Remarque. On retrouve la notion d'arbre au sens qu'on lui accorde usuellement en informatique.

3 Implémentation en OCaml

Deux méthodes s'offrent à nous pour représenter un graphe en machine :

- à l'aide de listes d'adjacence (à chaque sommet on associe la liste de ses voisins) ;
- à l'aide de matrices d'adjacence (le coefficient d'indice (i, j) traduit l'existence ou non d'une liaison entre deux sommets).

3.1 Par listes d'adjacence (première version)

Nous pouvons définir un sommet comme un enregistrement formé d'un identifiant et de la liste des identifiants de ses voisins, et un graphe comme une liste de sommets :

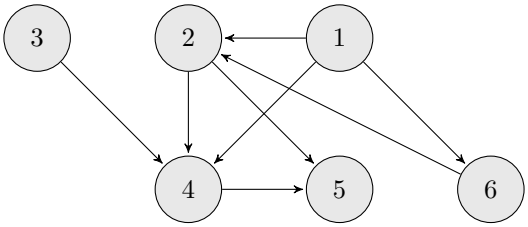
```
type 'a sommet = {id : 'a ; voisins : 'a list} ;;
type 'a graphe1 = 'a sommet list ;;
```

Cette représentation est la plus souple : elle présente l'avantage de permettre d'ajouter ou de supprimer facilement des sommets et des arêtes, mais l'inconvénient de ne pas permettre un accès rapide à un sommet ou une arête particulière.

En général, les sommets de chaque liste d'adjacence sont rangés selon un ordre arbitraire.

Définition d'un graphe par sa liste d'adjacence.

On considère le graphe :



Définir ce graphe orienté en OCaml avec le type `'a graphe1`.

Ajout et suppression d'une arête/arc.

Écrire en OCaml des fonctions d'ajout et de suppression d'une arête/arc dans un graphe de type `'a graphe1`.

Ajout et suppression d'un sommet.

Écrire en OCaml des fonctions d'ajout et de suppression d'un sommet dans un graphe de type `'a graphe1`.

3.2 Par listes d'adjacence (seconde version)

La représentation que nous venons d'étudier possède la propriété avantageuse de ne demander qu'une quantité minimale de mémoire ($O(n+p)$ si n désigne le nombre de sommets et p le nombre d'arêtes) et de permettre l'ajout et la suppression des sommets comme des arêtes. En contrepartie, elle présente l'inconvénient de ne pas permettre un accès rapide, ni aux sommets, ni aux arêtes.

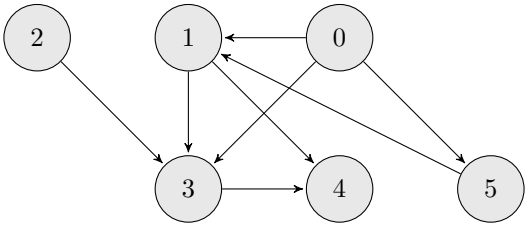
Sachant que par la suite les algorithmes que nous allons étudier vont opérer sur des graphes statiques (c'est-à-dire sans ajout ni suppression de sommet ou d'arête), nous allons modifier cette première représentation en utilisant désormais un tableau pour stocker les listes d'adjacence. Ainsi, nous pourront accéder en temps constant à la liste d'adjacence de chacun des sommets. Notons enfin que dans ce cas, il peut être intéressant de faire coïncider indices du tableau et identifiants des sommets. En OCaml, on numérotera les sommets de 0 à $n-1$ et on utilisera le type :

```
type graphe2 = int list array ;;
```

Cette nouvelle représentation des listes d'adjacence est toujours aussi économique en espace ($O(n+p)$) et permet encore l'ajout ou la suppression d'une arête, mais plus l'ajout ou la suppression d'un sommet.

Définition d'un graphe par sa liste d'adjacence.

On considère toujours le graphe :



Définir ce graphe orienté sur OCaml avec le type `graphe2`.

Ajout et suppression d'une arête/arc.

Écrire en OCaml des fonctions d'ajout et de suppression d'une arête/arc dans un graphe de type `graphe2`.

Désorientation d'un graphe.

Désorienter un graphe G consiste à calculer le plus petit graphe non orienté G' contenant G. Pour toute arête reliant les sommets a et b il faut donc ajouter l'arête reliant b à a, si celle-ci ne figure pas déjà dans le graphe.

Écrire en OCaml une fonction pour désorienter un graphe de type `graphe2`.

Remarque. Un inconvénient potentiel de la représentation par listes d'adjacence est que, pour déterminer si un arc (a, b) est présent dans un graphe, il n'existe pas de moyen plus rapide que de rechercher b dans la liste d'adjacence de a. En particulier, il est difficile de déterminer rapidement si un graphe est non orienté : il faut pour chacune des arêtes vérifier que l'arête réciproque est aussi présente.

3.3 Par la matrice d'adjacence.

Si on veut en plus accéder en coût constant à chacune des arêtes, il devient nécessaire d'ordonner les sommets $S = \{s_1, s_2, \ldots, s_n\}$ et utiliser une matrice $M \in \mathscr{M}_n(\{0,1\})$ pour représenter les arêtes :

$$m_{i,j} = \begin{cases} 1 & \text{si } (s_i, s_j) \in A, \\ 0 & \text{sinon.} \end{cases}$$

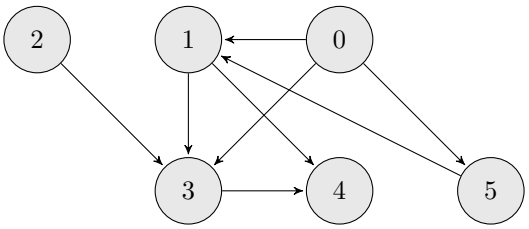
Cette représentation a des avantages : l'ajout et la suppression d'une arête a un coût constant ; déterminer si un graphe est orienté est chose aisée (il suffit de vérifier que la matrice est symétrique), mais elle présente l'inconvénient d'occuper beaucoup plus d'espace mémoire, en particulier lorsque le nombre d'arêtes est réduit vis-à-vis du nombre de sommets : si $n = |S|$ et $p = |A|$, le coût spatial de la représentation par matrice d'adjacence est un $O(n^2)$, contre un $O(n+p)$ pour une liste d'adjacence.

On utilisera le type suivant :

```
type graphe3 = int array array ;;
```

Définition d'un graphe par sa matrice d'adjacence.

On considère toujours le graphe :



Définir ce graphe orienté sur OCaml avec le type `graphe3`.

Liste des voisins d'un sommet d'un graphe.

Écrire en OCaml une fonction pour obtenir la liste des voisins d'un sommet d'un graphe de type `graphe3`.

Désorientation d'un graphe.

Écrire en OCaml une fonction pour désorienter un graphe de type `graphe3`.

Ajout et suppression d'une arête/arc.

Écrire en OCaml des fonctions d'ajout et de suppression d'une arête/arc dans un graphe de type `graphe3`.

Passage d'une liste d'adjacence à la matrice d'adjacence associée.

Écrire une fonction qui détermine la matrice d'adjacence (de type `graphe3`) associée à une liste d'adjacence donnée (de type `graphe2`).

Passage d'une matrice d'adjacence à la liste d'adjacence associée.

A l'inverse, écrire une fonction qui détermine la liste d'adjacence (de type `graphe2`) associée à une matrice d'adjacence donnée (de type `graphe3`).