

up

# Les Graphes

down bruh

the quick brown fox jumps over the lazy dog 1234567890211°00100100110à

## Table des matières

1. Notion de graphe et vocabulaire .....	3
1.1. Définitions et exemples .....	3
1.2. Adjacence, degré, voisin .....	5
1.3. Chemins et connexité .....	5
2. Modélisation d'un graphe .....	5
2.1. Représentation par matrice d'adjacence .....	5
2.2. Représentation par liste d'adjacence .....	6
3. Parcours de graphes .....	6
3.1. Le parcours en largeur (BFS, Breath First Search) .....	6
3.2. Parcours en profondeur (DFS, Depth First Search) .....	8
4. Résumé .....	8

# 1. Notion de graphe et vocabulaire

Le concept de graphe permet de résoudre de nombreux problèmes en mathématiques comme en informatique. C'est un outil de représentation très courant, et nous l'avons déjà rencontré à plusieurs reprises, en particulier lors de l'étude de réseaux et d'arbres binaires.

## 1.1. Définitions et exemples

Une multitude de problèmes concrets d'origines très diverses peuvent donner lieu à des modélisations par des graphes : c'est donc une structure essentielle en sciences, qui requiert un formalisme mathématique particulier que nous allons découvrir.

L'étude de la théorie des graphes est un champ très vaste des mathématiques : nous allons surtout nous intéresser à l'implémentation en Python d'un graphe et à différents problèmes algorithmiques qui se posent dans les graphes.

### 📖 Définition 1

.....  
.....

#### 1.1.1. Graphes non orientés

### 📖 Définition 2

.....  
.....

### 💡 Remarque

- Dans une représentation graphique, les sommets sont représentés par des cercles et les arêtes par des traits.
- Une arête entre deux sommets  $u$  et  $v$  est notée indifféremment par le couple  $(u, v)$  ou par le couple  $(v, u)$ .
- Une arête reliant un sommet à lui-même est appelée une **boucle**.
- Deux sommets peuvent être reliés par plusieurs arêtes.

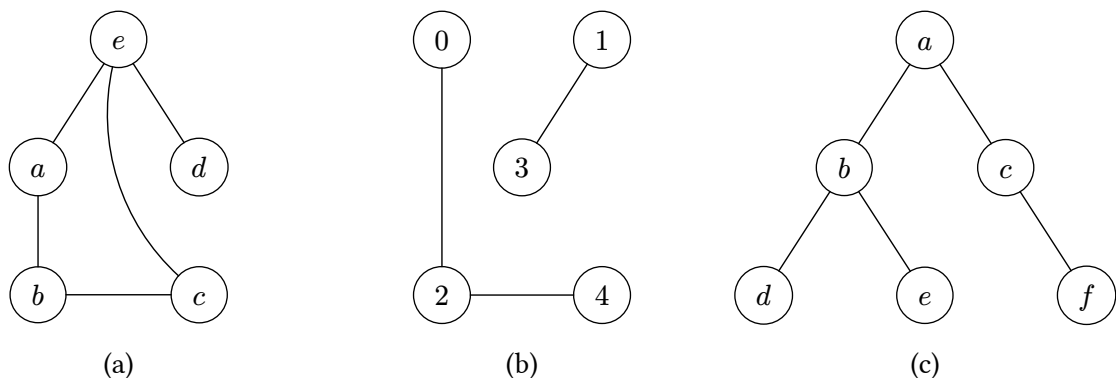


Figure 1 – Exemples de graphes non orientés

**Exemple 1:** Dans le graphe non orienté de la Figure 1, on a les sommets  $\{a, b, c, d, e\}$  et les arêtes  $\{(a, b), (a, e), (b, c), (c, e), (d, e)\}$ .

### ⚙ Exercice 1

Décrire la structure du graphe de la figure 2b comme dans l'exemple. Quelle est la particularité de ce graphe ?

.....

.....

### ⚙ Exercice 2

Décrire la structure du graphe de la figure 2c comme dans l'exemple. Quelle est la particularité de ce graphe ?

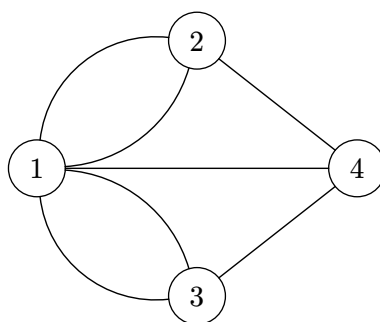
.....

.....

.....

### ⚙ Exercice 3

Un graphe possible permettant de modéliser le problème des sept ponts de Königsberg (voir problème) est le suivant :



Décrire la structure de ce graphe comme dans l'exemple. Que représentent les sommets et les arêtes de ce graphe ?

.....

.....

.....

.....

Que recherche-t-on sur ce graphe pour répondre au problème posé ?

### 1.1.2. Graphes orientés

#### Définition 3

.....

.....

#### Remarque

- Le nombre de sommets est appelé l'**ordre** du graphe et le nombre d'arcs est appelé la **taille** du graphe.
- Dans une représentation graphique, les sommets sont représentés par des cercles et les arcs par des flèches.
- Un arc partant d'un sommet  $u$  et arrivant à un sommet  $v$  est noté par le couple  $(u, v)$ .
- Un arc reliant un sommet à lui-même est appelé une **boucle**.
- Deux sommets peuvent être reliés par plusieurs arcs.

-FIGURE3-

**Exemple 2:** Dans le graphe orienté de la figure 3a, on a les sommets  $\{2, 3, 5, 7, 8, 9, 10, 11\}$  et les arcs  $\{(3, 8), (3, 10), (5, 11), (7, 8), (7, 11), (8, 9), (11, 2), (11, 9), (11, 10)\}$

### 1.1.3. Graphes pondérés

## 1.2. Adjacence, degré, voisin

### 1.2.1. Adjacence de sommets

### 1.2.2. Voisins d'un sommet

## 1.3. Chemins et connexité

### 1.3.1. Chemins entre deux sommets

### 1.3.2. Connexité

## 2. Modélisation d'un graphe

### 2.1. Représentation par matrice d'adjacence

#### 2.1.1. Graphe orienté

#### 2.1.2. Graphe non orienté

#### 2.1.3. Graphe pondéré

#### 2.1.4. Exercices

#### 2.1.5. Implémentation

## 2.2. Représentation par liste d'adjacence


### 2.2.1. Exercices

### 2.2.2. Implémentation

## 3. Parcours de graphes

### Définition 4

Un parcours de graphe est un algorithme consistant à explorer tous les sommets d'un graphe de proche en proche à partir d'un sommet initial. Ces parcours sont notamment utilisés pour rechercher un plus court chemin (et donc dans les GPS) ou pour trouver la sortie d'un labyrinthe.

 **Attention:** parcourir simplement le dictionnaire ou la matrice d'un graphe n'est pas considéré comme un parcours de graphe.

Tout les parcours suivent plus ou moins le même algorithme de base:

- On visite un sommet A . On crée une structure S qui contiendra au départ l'ensemble des voisins de A.
- Tant que S n'est pas vide:
  1. on choisit un sommet s dans S
  2. on visite s
  3. on ajoute à S tout les voisins de s pas encore visités

### Remarque

Contrairement à un parcours d'arbre, où les fils d'un nœud ne peuvent pas avoir été visités avant le nœud, un voisin d'un sommet peut avoir déjà été visité en tant que voisin d'un sommet précédent.

Il est donc nécessaire de mémoriser les sommets déjà visités ou découverts (on dira qu'un sommet est découvert lorsqu'on l'ajoute à S).

Le choix de la structure de l'ensemble S est prépondérant:

- Si on choisit une file (FIFO): on visitera les sommets dans l'ordre d'arrivée, donc les plus proches du sommet précédent. On obtient donc un parcours en largeur -> BFS.
- Si on choisit une pile (LIFO): on visitera d'abord les derniers sommets arrivés, donc on parcourt le graphe en visitant à chaque étape un voisin du précédent. On obtient donc un parcours en profondeur -> DFS.

### 3.1. Le parcours en largeur (BFS, Breath First Search)

### Définition 5

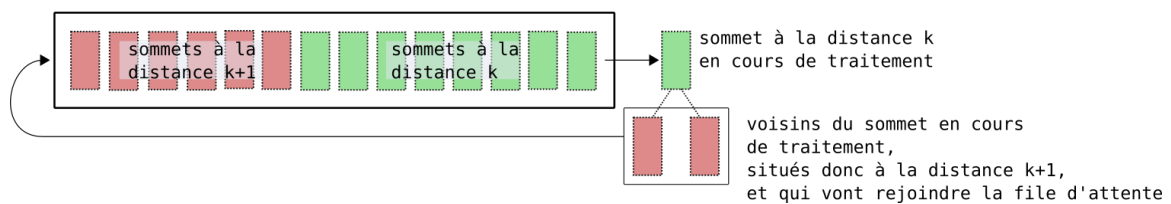
Pour le parcours en largeur, on va:

1. Commencer à un sommet du graphe.
2. Visiter tout les voisins de ce sommets.
3. Visiter tout les voisins non visités de ces voisins.
4. Répéter étape 3. jusqu'à ce que tout les sommets du graphe soient parcourus.

## Algorithme

```
entrées:  $G$  un graphe et  $s$  un sommet  
1   $f$ : une file vide  
2  enfiler  $s$  dans  $f$   
3  marquer  $s$   
4  tant que  $f$  non vide  
5      défiler  $f$  dans  $s$   
6      pour tout voisin  $t$  de  $s$  dans  $G$   
7          si  $t$  non marqué  
8              enfiler  $t$  dans  $f$   
9              marquer  $t$   
10         fin si  
11     fin pour  
12 fin tant que
```

## Remarque



## Exercice 4

Appliquer l'algorithme du BFS à ce graphe en partant de B.

⚠ – TODO – ⚠

Animation de cet algorithme sur un gros graphe: <https://youtu.be/x-VTfcmrLEQ>

### 3.1.1. Application du BFS: recherche du plus courts chemin

## Définition 6

L'algorithme BFS découvre les sommets «par cercles concentriques» autour du point de départ, chaque sommet est découvert via un sommet de distance  $k-1$  du centre. Chaque sommet de distance  $k$  peut voir le sommet de distance  $k-1$  qui l'a découvert comme parent.

Nous allons pour cela nous servir d'une structure de dictionnaire pour associer à chaque sommet son sommet-parent.

Il faudra ensuite une fonction pour recréer le chemin.

## Remarque

- Comment est-on sûr qu'un chemin **existe** entre deux sommets A et B ?

Si le graphe est connexe, tout parcours BFS au départ de A va parcourir l'intégralité du graphe, et donc passera par B à un moment. Un chemin sera donc forcément trouvé entre A et B.

- Comment est-on sûr que ce chemin trouvé est **le plus court** ?

La découverte des sommets par cercles concentriques entre A et B nous assure qu'on ne peut pas rater le point B : s'il est à la distance  $k$  de A, il sera forcément visité puisque tous les sommets à la distance  $k$  vont passer par la liste d'attente, après les sommets de distance  $k-1$  et avant les sommets de distance  $k+1$ .

Lorsqu'on remontera de B vers A en passant par les sommets parents successifs, il ne peut y avoir qu'un seul sommet par «couche» : le chemin sera donc exactement de longueur  $k$ , il sera donc minimal.

## Algorithme

### 3.2. Parcours en profondeur (DFS, Depth First Search)

#### 3.2.1. Algorithme du DFS

## 4. Résumé

hey what's up