1 Présentation

Un graphe est constitué d'un ensemble de **sommets** et d'**arêtes** reliant ces sommets.
Si les arêtes sont **orientées** (dirigées spécifiquement d'un sommet vers un autre), on les appelle plutôt des **arcs**.

Beaucoup de situations peuvent être modélisées par des graphes :
- réseau routier
- réseau social
- configurations d'un jeu
- labyrinthe
- internet (cf cours P3-3-Routage)...

2 Exemple des réseaux sociaux

On peut par exemple utiliser les graphes pour modéliser les relations entre individus dans un réseau social.

2.1 Relations symétriques : graphes non orientés

Considérons un réseau social type Facebook dans lequel les relations entre individus sont des **relations symétrique**s : si A est ami avec B, alors B est aussi ami avec A.
On peut alors construire un **graphe non orienté** dans lequel les sommets sont les individus et les arêtes représentent les relations d'amitié.

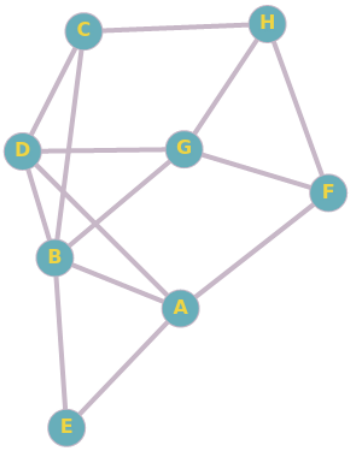
Considérons un mini réseau qui ne contient que 8 personnes : Alban, Béatrice, Charles, Déborah, Éric, Fatima, Gérald et Hélène.
Voici les relations d'amitié qui lient ces membres du réseau :
- Alban est ami avec Béatrice, Déborah, Éric et Fatima.
- Béatrice est amie avec Alban, Charles, Déborah, Éric et Gérald.
- Charles est ami avec Béatrice, Déborah et Hélène.
- Déborah est amie avec Alban, Béatrice, Charles et Gérald.
- Éric est ami avec Alban et Béatrice.
- Fatima est amie avec Alban, Gérald et Hélène.
- Gérald est ami avec Béatrice, Déborah, Fatima et Hélène.
- Hélène est amie avec Charles, Fatima et Gérald.

Cette énumération dans laquelle on fait la liste des relations, individu par individu s'appelle **liste d'adjacence**.

2.1.1 Représentation graphique

Remarque : plusieurs représentations graphiques sont possibles pour un même graphe : seule la liste des sommets et des arêtes définit réellement le graphe.

Voici par exemple une représentation de ce graphe (obtenue avec l'outil en ligne **graphonline**) :



2.1.2 Implémentation en Python

On peut représenter la **liste d'adjacence** de ce graphe à l'aide d'une liste Python avec le fonctionnement suivant :

- on numérote artificiellement les sommets (on choisira par exemple l'ordre alphabétique dans notre exemple : Alban = 0, Béatrice = 1...).
- chaque élément de la liste à l'indice i est une liste contenant les numéros des sommets en relation, appelés **successeurs**.

Exemple : Alban (i=0) a pour successeurs Béatrice (i=1), Déborah (i=3), Éric (i=4) et Fatima (i=5). Donc la liste d'adjacence commencera par :

```
liste_adj = [[1, 3, 4, 5], ...]
```

Terminez l'écriture de cette liste.
Remarque : dans cette représentation du graphe, on joindra à la liste d'adjacence une liste donnant la correspondance entre les numéros des sommets et leur étiquette ; ici, simplement :

```
etiquettes = ['A', 'B', 'C', 'D', 'E,' 'F', 'G']
```

On pourrait aussi représenter la liste d'adjacence de ce graphe à l'aide d'un **dictionnaire** dans lequel :

- les clés représentent les étiquettes des sommets (si elles sont bien uniques).
- les valeurs sont un tableau des successeurs.

Exemple : Alban est ami avec Béatrice, Déborah, Éric et Fatima.
Donc notre dictionnaire commencera par :

```
dict_adj = {'A':['B', 'D', 'E', 'F'], ...}
```

Terminez l'écriture de ce dictionnaire.

2.1.3 Matrice d'adjacence

Il existe une deuxième façon de représenter ce même graphe à l'aide d'un tableau à double entrée : la **matrice d'adjacence**.

```
0  1  0  1  1  1  0  0
1  0  1  1  1  0  1  0
0  1  0  1  0  0  0  1
1  1  1  0  0  0  1  0
1  1  0  0  0  0  0  0
1  0  0  0  0  0  1  1
0  1  0  1  0  1  0  1
0  0  1  0  0  1  1  0
```

 Expliquer comment est obtenue cette matrice.
Que remarquez-vous de particulier sur cette matrice ? A quoi est-ce dû ?

En Python, la matrice d'adjacence sera aisément implémentée par un tableau de tableaux (list of lists).

On donne ci-dessous une autre façon possible de définir la matrice d'adjacence :

```
False  True   False  True   True   True   False  False
True   False  True   True   True   False  True   False
False  True   False  True   False  False  False  True
True   True   True   False  False  False  True   False
True   True   False  False  False  False  False  False
True   False  False  False  False  False  True   True
False  True   False  True   False  True   False  True
False  False  True   False  False  True   True   False
```

La représentation en matrice d'adjacence peut **occuper une place mémoire importante** et éventuellement inutile si le nombre d'arêtes est petit. Le choix de la façon représenter un graphe dépendra du graphe et des algorithmes qu'on veut appliquer.

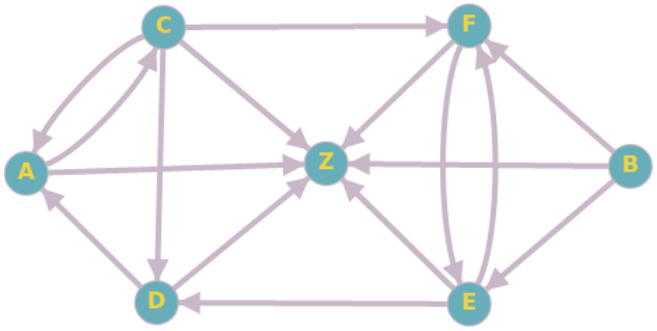
2.2 Relations non symétriques : graphes orientés

On va dans cette partie considérer un réseau social type Twitter dans lequel A peut suivre B sans que B ne suive A. On parle alors de **relation non symétrique** et de **graphe orienté**.

Voici un exemple de telles relations :

- Alice suit Chloé et Zoé.
- Bob suit Fred, Emma et Zoé.
- Chloé suit Alice, David, Zoé et Fred.
- David suit Alice et Zoé.
- Emma suit Zoé, David et Fred.
- Fred suit Zoé et Emma.

Voici une représentation graphique de ce graphe orienté :



On constate que les relations non symétriques sont représentées par des flèches : les arêtes sont des **arcs**.

Écrire la matrice d'adjacence de ce graphe.
Représenter ce graphe à l'aide d'un dictionnaire.

On retiendra qu'un graphe peut être défini au choix par :
- une **liste d'adjacence**.
- ou une **matrice d'adjacence**.

3 Modélisation d'un réseau routier

Les graphes sont abondamment utilisés par les logiciels de cartographie :

Les sommets représentent les villes et les arêtes sont les routes qui les relient.
Certaines routes peuvent être à sens unique : on utilisera alors des arcs orientés.
Les distances entre les villes peuvent être prises en compte sous forme de **poids** associé aux arêtes.

Exemple : On considère les villes A, B, C, D, E, F et G.

Voici comment est organisé le réseau routier :

- A et C sont reliées par une route à double sens ; AC = 2 km
- B est reliée à A par une route à sens unique (B → A) ; BA = 3 km
- A est reliée à D par une route à sens unique (A → D) ; DA = 4 km
- B est reliée à F par une route à sens unique (B → F) ; BF = 3 km
- E est reliée à B par une route à sens unique (E → B) ; EB = 4 km
- B et G sont reliées par une route à double sens ; BG = 6 km
- D et G sont reliées par une route à double sens ; DG = 5 km
- E et F sont reliées par une route à double sens ; EF = 7 km

Donner une représentation graphique de ce graphe.
Écrire la matrice d'adjacence de ce graphe. Les coefficients dans la matrice seront les distances entre les villes.

Remarque : une distance nulle entre 2 villes signifie conventionnellement qu'elles ne sont pas reliées. (On pourrait aussi par exemple décider d'écrire une distance infinie).

On s'intéresse à présent au temps de parcours en minutes.
Voici la matrice d'adjacence correspondante.

```
0 0 3 5 0 0 0
4 0 0 0 0 5 7
3 0 0 0 0 0 0
0 0 0 0 0 0 6
0 5 0 0 0 7 0
0 0 0 0 7 0 0
0 7 0 6 0 0 0
```

 Représenter ce nouveau graphe à l'aide d'un dictionnaire. Vous choisirez une structure de données pertinente.

4 Choix d'une représentation

Pour choisir entre matrice d'adjacence ou liste d'adjacence, on peut considérer les éléments suivants :

- la densité du graphe : c'est le rapport entre le nombre d'arêtes et le nombre de sommets. **Pour un graphe dense on utilisera plutôt une matrice d'adjacence**.
- le type d'algorithme que l'on souhaite appliquer. Certains algorithmes travaillent plutôt avec les listes d'adjacences alors que d'autres travaillent plutôt avec les matrices d'adjacences. **Le choix doit donc aussi dépendre des algorithmes utilisés**.

On utilise la matrice d'adjacence lorsqu'on a tout d'abord assez de mémoire, puis lorsqu'on a besoin d'accéder souvent et rapidement à des informations du type :

- Est-ce que le nœud A et le sommet B sont voisins ?
- Quel est le poids de l'arc entre le sommet C et D ?

La complexité en mémoire est en $O(N^2)$ (avec N le nombre de sommets du graphe), et la complexité pour accéder aux deux informations citées au-dessus est en $O(1)$ (puisqu'il s'agit d'un tableau).

La liste d'adjacence est le plus souvent utilisée lorsque :

- On n'a pas assez de mémoire pour stocker une matrice d'adjacence : la complexité en mémoire d'une liste d'adjacence est de $O(N+M)$ (avec M le nombre d'arcs du graphe).
- On cherche à parcourir le graphe plus rapidement qu'en utilisant une matrice d'adjacence. En effet, dans une liste d'adjacence il n'y a que les voisins du sommet en question, alors que dans la matrice tous les sommets sont représentés.

5 Graphonline

On consultera avec intérêt le site de **graphonline** pour représenter des graphes et y appliquer automatiquement des algorithmes : `https://graphonline.ru/fr/`