SÉANCE 4

RAPPELS SUR LES GRAPHES

I - Terminologie

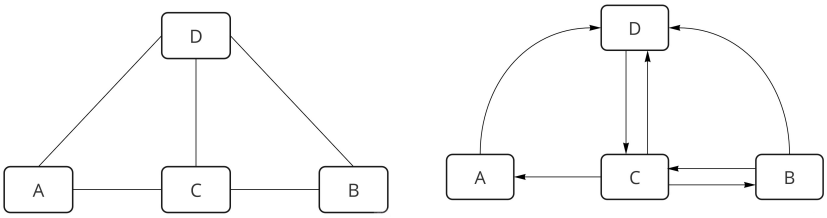
Un ***graphe*** G est un couple (S, A) où :

- S est un ensemble *fini et non vide* d'éléments appelés ***sommets*** ou ***nœuds***.
- A est un ensemble de couples ou de paires d'éléments de S appelées ***arêtes***.

Lorsque les arêtes sont des couples, on dit que le graphe est ***orienté*** et les arêtes sont appelées des ***arcs*** (et un arc ayant mêmes origine et arrivée est appelé une *boucle*).
Dans le cas contraire, on dit que le graphe est ***non orienté***.

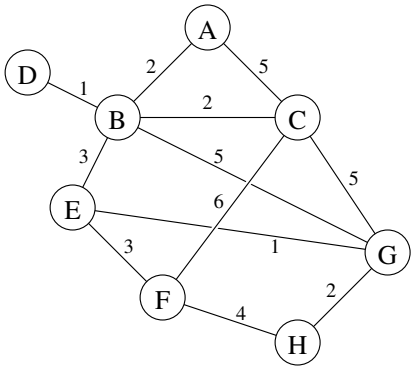
Exemple

Voici à gauche un exemple de graphe non orienté et à droite un exemple de graphe orienté.



Il peut être également utile d'ajouter des poids ou des étiquettes aux arêtes : on parle alors de graphe ***pondéré*** ou *étiqueté*.

Exemple



Il convient de maîtriser certains éléments de vocabulaire.

- Un graphe est dit *simple* lorsque, entre deux sommets, il n'y a pas plus d'une arête.
- Le nombre de sommets du graphe G s'appelle l'*ordre du graphe* et le nombre d'arêtes ou d'arcs s'appelle la *taille* du graphe G.
- Deux sommets reliées par une arête ou un arc sont dits *adjacents* ou *voisins*; ces sommets sont les *extrémités* ou les *sommets de l'arête* (dans le cas orienté, on parle d'extrémité initiale ou d'origine et d'extrémité finale ou d'arrivée de l'arc).

  Le *degré* d'un sommet x est le nombre de ses voisins. Un sommet de degré 0 est dit *isolé*.
- Un *chemin* de s_0 à s_n est une séquence $(s_0, s_1, \ldots, s_n)$ de sommets tels que deux consécutifs soient adjacents; ce chemin permet de relier s_0 à s_n, s_0 est le sommet de départ, s_n est le sommet d'arrivée; on emploie aussi le terme de *chaîne* mais plutôt dans le contexte des graphes orientés.

  La *longueur* d'un tel chemin $(s_0, \ldots, s_n)$ est le nombre d'arêtes utilisées pour aller du sommet s_0 au sommet s_n, il y en a donc n.

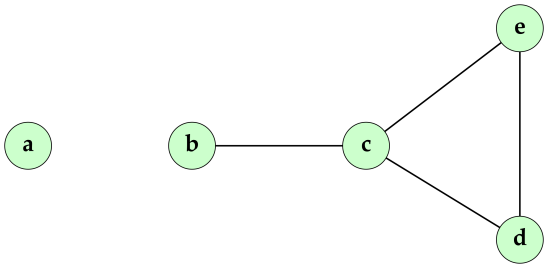
  La *distance* entre deux sommets est la longueur minimale d'un chemin reliant ces deux sommets.

  Un chemin ayant même origine que l'arrivée est appelé un *cycle*.
- On dit que le graphe est *connexe* si, pour tout couple de sommets distincts, il existe un chemin reliant ces deux sommets.

  Les sous-graphes connexes maximaux d'un graphe sont appelés ses *composantes connexes*.

Exemple

Considérons le graphe représenté ci-dessous :



Le sommet **a** est isolé et il y a deux composantes connexes.

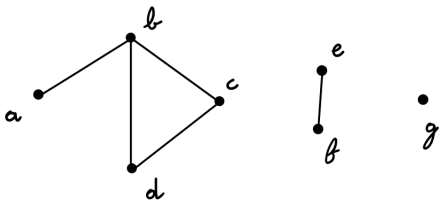
II - Représentation d'un graphe avec Python

Un graphe peut être représenté de deux façons : à l'aide d'une *liste d'adjacence* ou à l'aide d'une *matrice d'adjacence*.

Une représentation par **liste d'adjacence** d'un graphe associe à chaque sommet du graphe la collection de ses voisins.

Exemple

Voici un graphe G ainsi que sa représentation par liste d'adjacence.



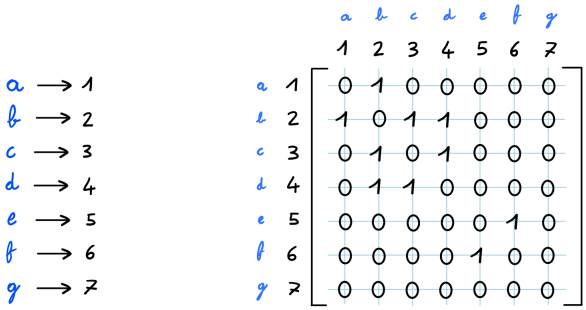
sommets	voisins
a	$\{b\}$
b	$\{a, c, d\}$
c	$\{b, d\}$
d	$\{b, c\}$
e	$\{f\}$
f	$\{e\}$
g	$\{\}$

La **matrice d'adjacence** d'un graphe d'ordre n est la matrice :

$$(a_{ij})_{(i,j)\in[\![1,n]\!]^2} \text{ avec } a_{ij} = \begin{cases} 1 & \text{si } i \text{ et } j \text{ sont voisins} \\ 0 & \text{si } i \text{ et } j \text{ ne sont pas voisins} \end{cases}$$

Exemple

En reprenant le graphe précédent, et en renommant ses sommets par les entiers de 1 à 7, on obtient la matrice d'adjacence suivante :



Notons que pour un graphe non orienté la matrice d'adjacence est toujours symétrique et la diagonale est toujours constituée de zéros.

La matrice d'adjacence M d'un graphe d'ordre n peut être représentée en Python par une liste de listes de la manière suivante :

- `M[i]` est la ligne d'indice `i` avec `i` $\in [\![0, n-1]\!]$;
- `M[i][j]` contient la valeur de $a_{\texttt{ij}}$ avec `(i,j)` $\in [\![0, n-1]\!]^2$.

Exemple

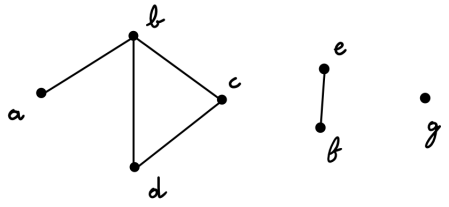
Toujours avec le même graphe, on a :

```
M = [[0, 1, 0, 0, 0, 0, 0],
     [1, 0, 1, 1, 0, 0, 0],
     [0, 1, 0, 1, 0, 0, 0],
     [0, 1, 1, 0, 0, 0, 0],
     [0, 0, 0, 0, 0, 1, 0],
     [0, 0, 0, 0, 1, 0, 0],
     [0, 0, 0, 0, 0, 0, 0]]
```

Pour mettre en œuvre en langage Python la représentation par liste d'adjacence, il est pertinent d'utiliser un dictionnaire dont les clés sont les sommets et les valeurs sont les listes de voisins.

Exemple

Considérons à nouveau le graphe :



Avec un dictionnaire, on définit un tel graphe par :

```
G = {'a' : ['b'], 'b' : ['a', 'c', 'd'], 'c' : ['b', 'd'], 'd' : ['b', 'c'], 'e' : ['f'],
    'f' : ['e'], 'g' : []}
```

Dans le cas d'un graphe pondéré, on peut remplacer les sommets (dans les listes de voisins) par un couple constitué du nom du sommet et du points de l'arête correspondante (ou de l'arc).

Exemple

Quel est le graphe représenté par le dictionnaire suivant ?

```
G = {'a': [('b', 8), ('c', 2), ('d', 1)],
     'b': [('a', 8), ('c', 4), ('d', 6), ('e', 1)],
     'c': [('a', 2), ('b', 4), ('e', 2)],
     'd': [('a', 1), ('b', 6)],
     'e': [('b', 1), ('c', 2)]
       }
```

Exemple / Exercice

Écrire une fonction python prenant un dictionnaire (représentant la liste d'adjacence d'un graphe) et renvoyant le nombre de sommets et le nombre d'arêtes.

Exemple / Exercice

Écrire une fonction python prenant un dictionnaire (représentant la liste d'adjacence d'un graphe) et le nom d'un sommet et renvoyant son degré.

Exemple / Exercice

Écrire une fonction python prenant un dictionnaire (représentant la liste d'adjacence d'un graphe non orienté) et une liste `[a, b]` correspondant à deux sommets, et ajoutant une arête entre ces deux sommets.