Structures de données relationnelles : les graphes

8.1 Introduction et notions de base

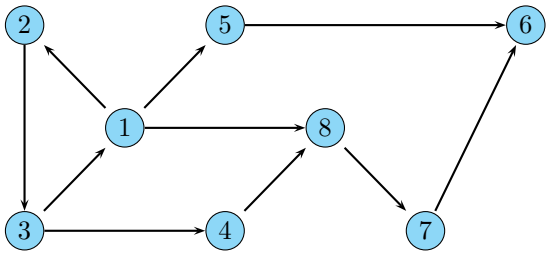
Définition 1
Un graphe est une structure constituée de deux ensembles :

un ensemble fini V de sommets (*vertex* en anglais), les élements du graphe ;

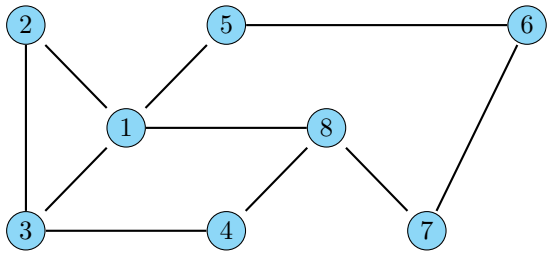
un ensemble E d'arcs (edge en anglais), qui sont des relations entre certains des sommets du graphe.

Définition 2
Chaque arc possède une extrémité initiale et une extrémité terminale. Lorsque cette distinction n'est pas pertinente, le graphe est dit non-orienté et les arcs sont alors appelés des arêtes. Lorsque cette distinction est pertinente, le graphe est dit orienté.

On a l'habitude de représenter les graphes sous la forme de diagrammes sagittaux. Les sommets sont représentés sous forme de points ou de cercles. Les arcs sous formes de flêches s'il s'agit d'un graphe orienté et les arêtes pour un graphe non-orienté par des segments.



(a) Réprésentation sagittale d'un graphe orienté

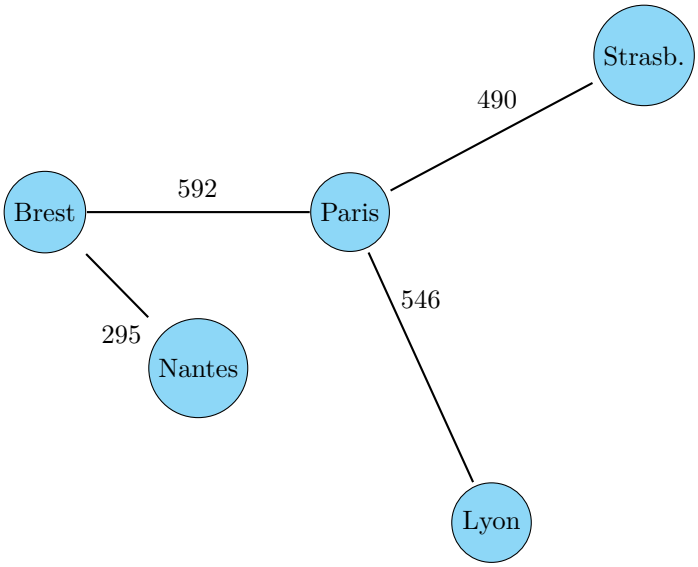


(b) Représentation sagittale d'un graphe non-orienté

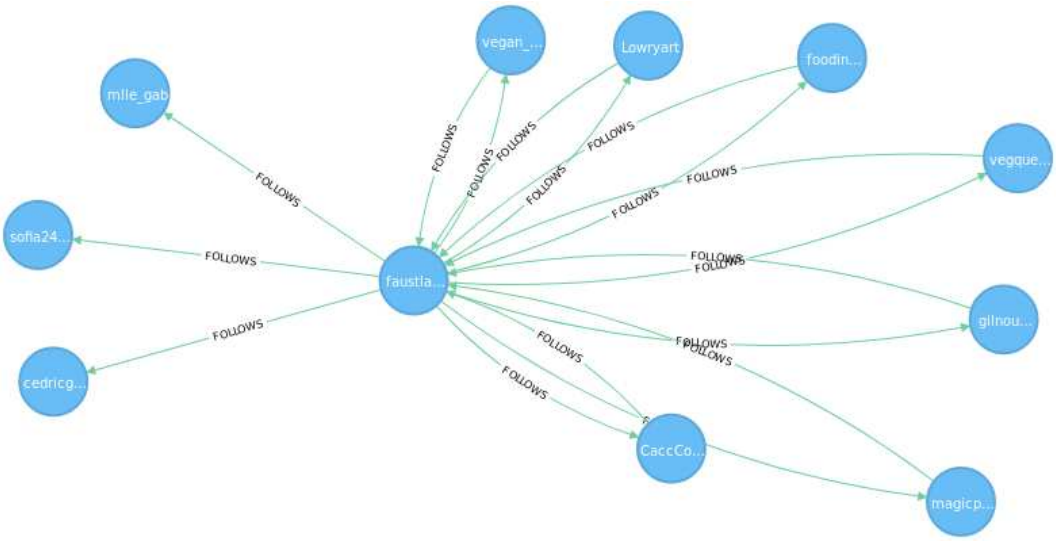
Par exemple pour le graphe non-orienté b) ci-dessus l'ensemble V des sommets est $\{1, 2, 3, 4, 5, 6, 7, 8\}$ tandis que celui des arêtes est $\{\{1, 2\}\{1, 3\}, \{1, 5\}, \{1, 8\}\{2, 3\}, \{3, 4\}, \{4, 8\}, \{5, 6\}, \{6, 7\}, \{7, 8\}\}$.

a. À quoi sert un graphe ?

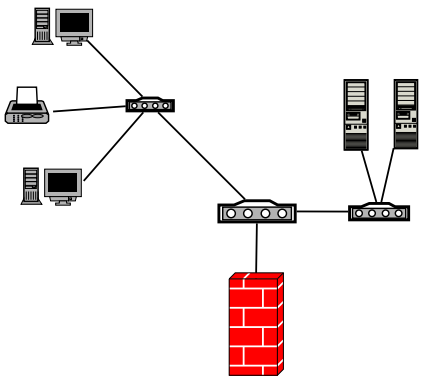
Principalement à représenter des relations entre différents objets, comme par exemple un réseau de transport. Sur ce graphe on a indiqué la distance séparant deux villes. Il s'agit d'un graphe pondéré, à chaque arête on attribue un poids ou coefficient.



Un réseau social. Ci-dessous un extrait des relations « follower - following » d'un compte d'un réseau social, il s'agit d'un graphe orienté puisque la relation n'est pas forcément réciproque (A peut suivre B sans que B ne suive A) :



un réseau informatique :



En informatique, un grand nombre de problèmes peuvent être modélisés sous forme de graphes, par conséquent les algorithmes opérant sur les graphes sont primordiaux.
Avant de traiter les algorithmes sur les graphes, il convient de réaliser qu'un graphe est une structure de données au même titre que les piles, les files, les listes et les arbres. À ce titre il est important de signaler que les arbres sont des graphes particuliers. Les arêtes du graphe symbolisent les relations entre ces données.

Les données : élements de l'ensemble V
Les opérations primitives :

`creer_graphe()` : crée un graphe vide.

`est_dans(G, v)` : retourne vrai si `v` est un sommet du graphe G et sinon faux.

`ajouter_sommet(G, v)` : ajoute un sommet `v` au graphe G.

`supprimer_sommet(G, v)` : supprimer un sommet `v` au graphe G.

`ajouter_arc(G, v1, v2)` : ajoute un arc entre `v1` et `v2`, sommets du graphe G.

`supprimer_arc(G, v1, v2)` : supprime un arc entre `v1` et `v2` éléments du graphe G.

`voisins(G,v)` : retourne les sommets dits adjacents ou voisins du sommet `v` du graphe `G`.

Il conviendra aussi de préciser la sémantique des opérations pour l'implémentation choisie. En effet en fonction de l'implémentation les opérations n'auront pas tout à fait le même sens. Par exemple pour certaines implémentation ajouter un sommet impliquera nécessairement de préciser sa relation avec les autres c'est-à-dire d'ajouter des arcs au graphe alors que pour d'autres implémentation on peut ajouter un sommet puis dans un deuxième ajouter les arcs au graphe.

8.2 Implémentations

Il existe plusieurs façons d'implémenter un graphe, deux sont présentées ci-dessous.

a. Liste d'adjacence

Les sommets sont représentés comme des clés et à chaque sommet est associée une liste de sommets adjacents ou voisins. Cette structure permet de conserver des données supplémentaires pour les sommets comme par exemple une pondération pour chaque arc.
En Python l'utilisation de dictionnaires est pertinente, en effet les clés sont constitués par les sommets et les arcs par des listes Python (à ne pas confondre avec les listes en tant que structures de données abstraites).
Ainsi le graphe de transport pourrait être implémenté ainsi :

```
transport = {'Paris': [('Strasbourg', 490), ('Brest', 592), ('Lyon',546)],
             'Brest' : [('Paris', 592), ('Nantes', 295)],
             'Nantes' : [('Brest', 295)],
             'Lyon' : [('Paris', 546)],
             'Strasbourg' : [('Paris', 490)]}
```

Avec cette façon d'implémenter ce type de graphe voici comment pourraient s'écrire certaines opérations primitives.

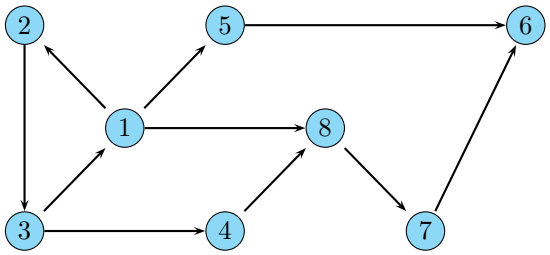
```
def est_dans(G, v):
    return v in G.keys()


def voisins(G, v):
    if est_dans(G,v):
        return [sommet for sommet, poids in G[v]]
    else :
        return []
```

La sémantique, le sens que l'on place derrière ces opérations, a son importance. Dans l'exemple ci-dessus, les voisins sont donnés au sens strict (uniquement les sommets). Une autre interprétation en fonction du contexte pourrait être plus pertinente. Dans un contexte ou l'on doit considérer les sommets voisins avec leurs pondérations, on utiliserait plutôt :

```
def voisins(G, v):
    if est_dans(G,v):
        return G[v]
    else :
        return []
```

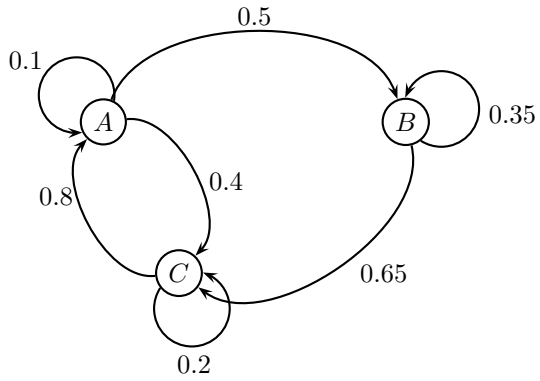
▶ Exercice 1

Écrire l'implémentation du graphe orienté ci-dessous sous forme de listes d'adjacence.



b. Matrice d'adjacence

Il s'agit d'une matrice carrée (tableau à double entrée) où les lignes représentent les sommets de départ et les colonnes les sommets d'arrivée. Une entrée nulle indique qu'il n'y a pas d'arc entre ces deux sommets, une valeur non nulle indique l'existence d'un arc avec une éventuelle pondération.
Ainsi le graphe probabiliste ci-dessous peut-être implémenté par la martice ci-contre :



On dispose les sommets dans l'ordre alphabétique en ligne et en colonne A, B et C. Puis on part d'une ligne i vers une colonne j. Si un arc existe dans ce sens, on indique son coefficient, sinon on indique 0.

$$\begin{pmatrix} 0.1 & 0.5 & 0.4 \\ 0 & 0.35 & 0.65 \\ 0.8 & 0 & 0.2 \end{pmatrix}$$

En classe de première les matrices ont été étudiées sous forme de tableaux de tableaux, d'où :

```
graph_proba = [[0.1, 0.5, 0.4],
               [0, 0.35, 0.65],
               [0.8, 0, 0.2]]
```

On remarque que dans la réprésentation à l'aide de matrice un simple coup d'oeil suffit à déterminer si un graphe est non-orienté. En effet un graphe est non-orienté si et seulement si sa matrice est symétrique (pour tout i et j le terme ligne i colonne j est égal au terme ligne j colonne i).

▶ Exercice 2

Écrire l'implémentation du graphe non-orienté à l'aide d'une matrice.