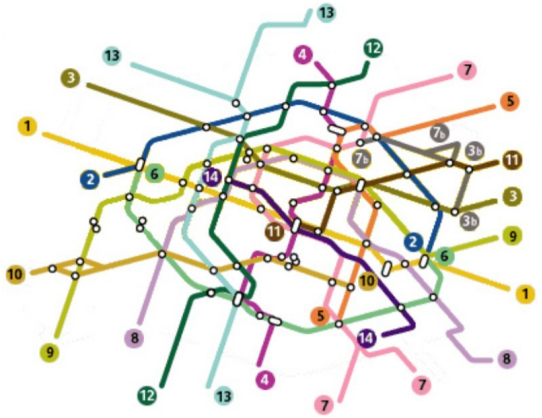
CH10 : LES GRAPHES

Dans ce chapitre, nous allons étudier un type de structure relationnelle : les graphes.

La structure de graphe permet de modéliser de nombreuses situations : un ensemble de pages HTML, un réseau social, un labyrinthe, un réseau de bus …



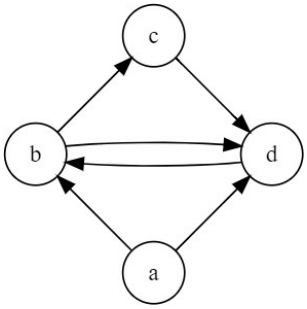
I. DEFINITIONS

Un ***graphe*** est un ensemble de sommets reliés entre eux par des arcs. Dans l'exercice d'introduction, les sommets sont les pages html, les arcs sont les hyperliens.

Graphe orienté : un graphe orienté est un graphe pour lequel on distingue le sens des arcs.

Le graphe ci-contre est orienté : il y a un arc de b vers c, mais pas de c vers b.

Ce graphe comprend 6 arcs.



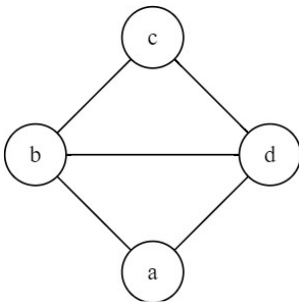
Graphe non orienté : un graphe non orienté est un graphe pour lequel on s'intéresse aux liaisons entre deux sommets, sans s'occuper du sens.

Le graphe ci-contre est graphe non orienté : il y a un arc reliant b et c.

Ce graphe comprend 5 arcs.

Le graphe de l'exercice d'introduction, et en général un ensemble de pages html reliées est un graphe orienté.

Dans l'introduction, on peut remarquer que chaque page a un lien permettant de revenir à la page d'origine. On pourrait donc le représenter sous la forme d'un graphe non orienté.



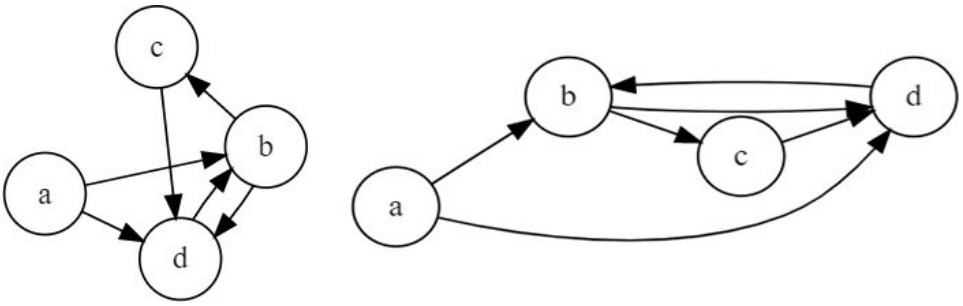
Voisinage : Lorsqu'il y a un arc d'un sommet a vers un sommet b, on dit que b est adjacent à a, ou que b est un voisin de a.

Dans les exemples ci-dessus, citer les voisins de b :

- Dans le graphe orienté :
- Dans le graphe non orienté :

Représentation d'un graphe : on peut représenter un graphe comme vu ci-dessus. La représentation n'est pas unique : un graphe est défini par l'ensemble de ses sommets et de ses arcs et non de la façon dont il est dessiné.

Les deux graphes ci-dessous sont les mêmes (le vérifier) :



2. CHEMIN DANS UN GRAPHE

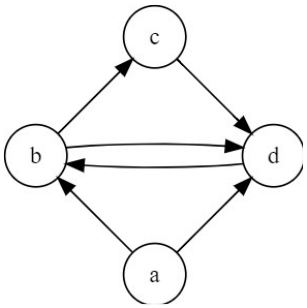
Dans l'exercice d'introduction, il faut parfois plusieurs clics pour aller d'une page donnée à une autre : on appelle cela un chemin

Chemin : un chemin est une suite finie de sommets reliés deux à deux par des arcs.

Dans le graphe orienté ci-contre, un chemin de a à c est a – b – c.

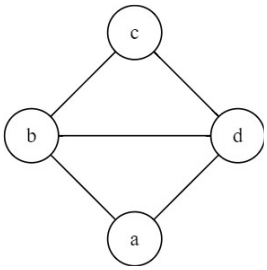
Citez deux autres chemins reliant a à c.

Il n'y a parfois pas de chemin reliant deux sommets donnés : ici on ne peut pas aller du sommet c au sommet a.



Dans un graphe non orienté, il existe un chemin d'un sommet x à un sommet y si et seulement si il existe un chemin du sommet y au sommet x.

Dans le graphe non orienté ci-contre, un chemin de a à c est a – b – c. Un chemin reliant c à a est alors c – b – a .



Cycle

Un chemin est dit ***simple*** s'il n'emprunte pas deux fois le même arc, et ***élémentaire*** s'il ne passe pas deux fois par le même sommet.

Dans le graphe orienté précédent :

- a – b – c est un chemin simple et élémentaire
- a -b – d – b – c est un chemin simple mais pas élémentaire.

Donner d'autres exemples du même type pour le graphe non orienté :

Un chemin simple reliant un sommet à lui-même et contenant au moins un arc est appelé un ***cycle***.

Dans le graphe orienté précédent, b – c – d – b est un cycle.

Donner un exemple de cycle, et un exemple de chemin qui n'est pas un cycle dans le graphe non orienté.

Distance

La ***longueur*** d'un chemin est le nombre d'arcs qui constituent le chemin.

La ***distance entre deux sommets*** est la longueur du plus court chemin reliant ces deux sommets.

La distance d'un sommet à lui-même en n'empruntant aucun arc est 0.

La distance entre deux sommets non reliés n'est pas définie.

Pour chacun des deux graphes précédents, déterminer plusieurs chemins reliant c à b. Donner leur longueur, puis préciser la distance entre les deux sommets.

Connexité

Un graphe non orienté est connexe si, pour tous sommets x et y, il existe un chemin reliant x et y.

Un graphe orienté est :

- connexe si, pour tous sommets x et y, il existe un chemin de x vers y OU de y vers x.
- fortement connexe si pour tous sommets x et y, il existe un chemin de x vers y ET de y vers x.

Dans les exemples précédents, les deux graphes sont connexes, mais le graphe orienté n'est pas fortement connexe.

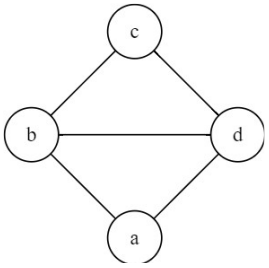
Matrice d'adjacence

On appelle matrice d'adjacence d'un graphe à n sommets la matrice (le tableau de n lignes et n colonnes) constituée des coefficients a_{ij} (à l'intersection de la ième ligne et de la jème colonne) tels que :

$$a_{ij} = \begin{cases} 1 \; s'il \; y \; a \; un \; arc \; du \; sommet \; i \; vers \leq sommet \; j \\ 0 \; sinon \end{cases}$$

Cette matrice est unique et permet de définir le graphe.

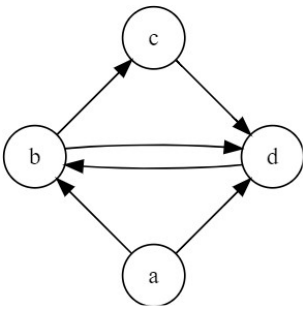
La matrice d'adjacence du graphe non orienté ci-contre, en considérant les sommets dans l'ordre alphabétique est :



$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Remarque : la matrice d'adjacence d'un graphe non orienté est symétrique.

Ecrire de même la matrice d'adjacence du graphe orienté.



Travail à faire : Activité 1

3. IMPLEMENTATION DE LA CLASSE GRAPHE

a. A l'aide d'un tableau

Travail à faire : notebook Graphes_matrice

Une première façon d'implémenter un graphe en Python est d'utiliser la matrice d'adjacence.

On peut représenter celle-ci sous la forme d'un tableau de n lignes et n colonnes (pour un graphe comportant n sommets). Cette matrice caractérise le graphe, et on peut donc travailler ensuite sur les valeurs du tableau.

Efficacité

La matrice d'adjacence est simple à mettre en œuvre, puis à utiliser pour parcourir les sommets ou les arcs d'un graphe. Mais elle comporte quelques défauts :

- L'espace mémoire utilisé est important, puisque pour n sommets, il faut n^2 éléments dans la matrice. Ainsi, un graphe de mille sommets (un réseau social en comporte des centaines de millions) nécessite une matrice avec 1 million d'éléments, *même s'il n'y a pas beaucoup d'arcs*.
- Le nombre de sommets doit être connu à la création du graphe. L'ajout postérieur d'un sommet nécessite de réécrire la matrice, ce qui est coûteux en temps.
- Les sommets sont forcément des entiers.

Nous allons voir par la suite une représentation plus optimale, à l'aide d'un dictionnaire.

b. A l’aide d’un dictionnaire

Travail à faire : Notebook Graphes_dictionnaire

Une deuxième façon d’implémenter un graphe en Python est d’utiliser un dictionnaire d’adjacence.

On crée un dictionnaire dont les clés sont les sommets, et dont les valeurs sont les voisins du sommet considéré.

Efficacité

- Utiliser un dictionnaire plutôt qu'une matrice permet de manipuler tout type de données pour les sommets.
- Il n'y a pas besoin de fixer le nombre de sommets au départ, on peut en ajouter autant que l'on veut.
- Lorsqu'il n'y a pas beaucoup d'arcs, l'espace mémoire occupé est moindre que pour la matrice.
- Le coût en temps des opérations est optimal(ajouter un arc, un sommet...).