Cours 23 – Bases des graphes – Partie 1 – Vocabulaire

A / À quoi sert la théorie des graphes ?

La théorie des graphes (vous entendrez parfois également parler de mathématiques discrètes) est une branche des mathématiques née au XX^{ème} siècle, qui permet de modéliser et de résoudre des problèmes liés notamment à :

- La cartographie : réseaux énergétiques, routiers, internet (très à la mode avec l'IOT : internet des objets) …
- La chimie et la biologie : modélisation de molécules, intéractions entre protéines, …
- Les neurosciences : modélisation neurones – synapses
- L'économie et la gestion : emplois du temps, plannings de gestion de livraisons, choix d'une succession d'étapes à suivre (ordonnancement)
- Aide à la décision : agrégation multicritère, détermination de préférences …
- Recherche d'information : PageRank de *Google*, modélisation de réseaux sociaux …

En tant que futurs ingénieurs, cette science fait partie de votre formation car elle est couramment utilisée pour résoudre certains types de problèmes dits d'**optimisation**.

« L'**Optimisation Combinatoire** consiste à trouver la meilleure solution parmi un nombre fini (mais souvent très grand) de choix. C'est une branche de la « **Programmation Mathématique** » qui recouvre les méthodes qui servent à déterminer l'optimum d'une fonction sous des contraintes données. Il s'agit donc de **minimiser** une fonction sur un **ensemble fini**, mais **éventuellement très grand**, et **dont les propriétés mathématiques ne sont pas facilement caractérisables.** » L. Esperet, laboratoire G-SCOP, Grenoble.

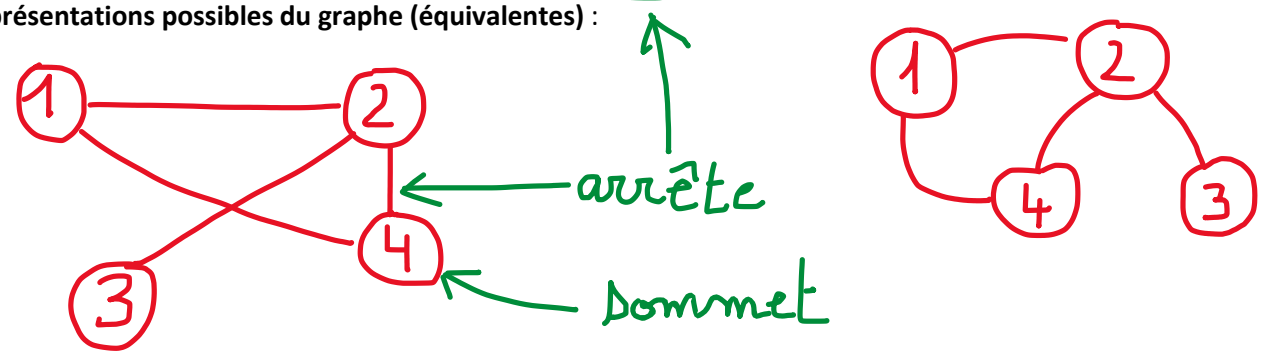
B / Graphe non orienté

B.1 / Sommets et arêtes

Soit S un ensemble fini appelé ensemble des ***sommets*** (parfois aussi appelés ***nœuds),*** et A un ensemble de couples (en toute rigueur des paires non ordonnées, cf. remarque ci-dessous) $\{S_i, S_j\}_{i\neq j}$ de $S \times S$ appelé ensemble des ***arêtes*** associé à S. Le couple $G = (S, A)$ est un ***graphe***.

Exemple : $S = \{1,2,3,4\}$ $\quad A = \{\{1,2\},\{2,3\},\{1,4\},\{2,4\}\}$

2 représentations possibles du graphe (équivalentes) :



Remarque :

• En mathématique un **couple** (ou **paire ordonnée**) est noté $(a, b) \neq (b, a)$, l'ordre est important ! Une **paire non ordonnée** est un ensemble de 2 éléments noté $\{a, b\} = \{b, a\}$, l'ordre n'a pas d'importance.

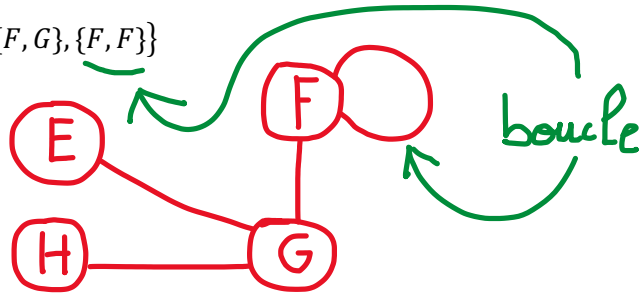
• En Python, **Les tuples** (« n-uplets » en français) sont des collections de données différentes ou identiques qui sont désignées par un index et qui ne peuvent pas être modifiées (on dit qu'elles sont « non mutables »). Seule différence notable avec les listes. On note : $c = (3, 4, 9)$. Nous ne parlerons pas de tuples dans notre cours.

B.2 / Boucle

Une arête de A peut parfois être un binôme $\{S_i, S_i\}$ de $S \times S$ (2 fois le même élément) : on parle alors de ***boucle***.

Exemple : $S = \{E, F, G, H\}$ $\quad A = \{\{E, G\}, \{G, H\}, \{F, G\}, \{F, F\}\}$

Une représentation possible du graphe :

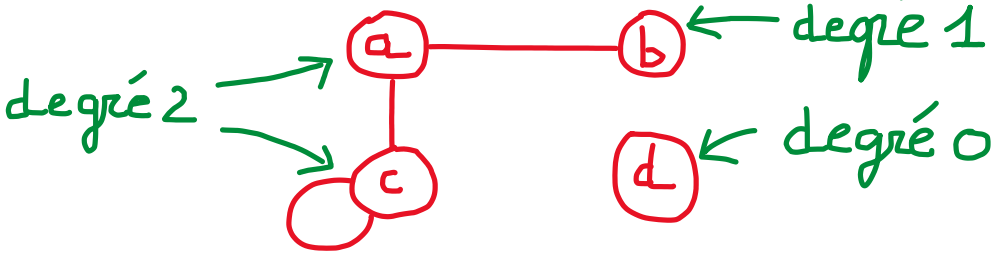


B.3 / Adjacence et degré d'adjacence

Deux sommets S_i et S_j de $S \times S$ sont dits ***voisins*** (ou ***adjacents***) s'il existe une arête $\{S_i, S_j\}$ qui les relie. Le nombre de voisins d'un sommet S_i est appelé ***degré d'adjacence*** de ce sommet.

Exemple : $S = \{a, b, c, d\}$ $\quad A = \{\{a, b\}, \{a, c\}, \{c, c\}\}$

Une représentation possible du graphe :



B.4 / Chemin et connexité

Soient deux sommets S_i et S_j $(i \neq j)$: on appelle ***chemin*** entre S_i et S_j un sous-ensemble de A permettant de relier les deux sommets. Il peut y avoir plusieurs chemins pour un même couple S_i et S_j. Un **chemin** est associé à une ***longueur***, ici (sur un graphe non pondéré) c'est le nombre des **arêtes** parcourues.

On dit qu'un graphe $G = (S, A)$ est ***connexe*** ssi tout couple de sommets S_i , S_j peut être relié par un **chemin**

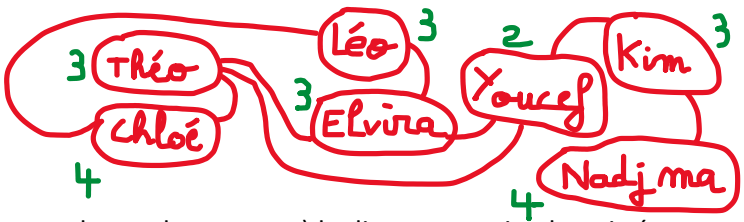
Exemple 1 : Cartographie des trajets disponibles pour une compagnie routière

Connexe ? Non

Donner 2 chemins entre Londres et Milan et leur longueur :

$\{\{L, P\}, \{P, Ma\}, \{Ma, Mi\}\}$
$\{\{L, Ma\}, \{Ma, Mi\}\}$

Exemple 2 : Représentation des liens d'«amitié» sur un réseau social type Facebook (où A est ami avec B est une relation réciproque, représentée ici par une arête).



Connexe ? Oui

La suite de ce cadre est à titre d'exemple, les deux définitions suivantes ne sont pas à retenir.

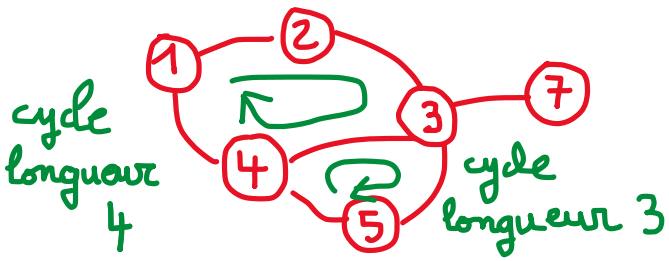
- On appelle ***écartement*** (d'un sommet S_i au reste du graphe connexe) la distance maximale qui sépare ce sommet de chacun des autres sommets S_j si l'on emprunte le chemin le plus court.
  ➔ Annoter à côté de chaque nom l'écartement du sommet.
- On appelle ***centre*** du graphe le sommet S_i ayant l'écartement le plus faible. Ici : Youcef

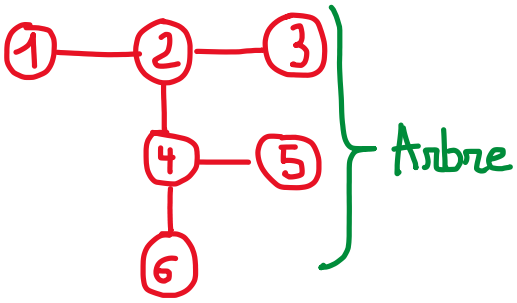
B.5 / Cycle et arbre

Les deux définitions suivantes ne sont pas officiellement au programme.

On appelle ***cycle*** un chemin permettant de relier un sommet S_i à lui-même. **Exemple :** sur le graphe de liaison d'une chaîne cinématique fermée. On appelle ***arbre*** un graphe connexe ne comportant aucun cycle.

Exemple : graphe de liaison d'une chaîne cinématique ouverte.





C / Graphe orienté

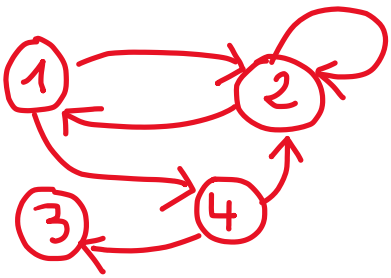
C.2 / Sommets, arcs et boucles

Pour un ensemble $S = \{S_i\}_{1 \leq i \leq N}$ de **sommets**, soit A un ensemble de tuples (ou couples ou paires ordonnées) (S_i, S_j) de $S \times S$ appelé ensemble des ***arcs*** associé à S. Le couple $G = (S, A)$ est un **graphe**.
La seule différence par rapport à une **arête** est qu'un **arc** est orienté (il va de S_i à S_j), on le représente par une flèche.

Exemple : $S = \{1,2,3,4\}$ $\quad A = \{(1,2), (2,1), (1,4), (4,2), (2,2), (4,3)\}$

Une représentation possible du graphe orienté :

Exemple : modélisation d'un réseau social type *Strava* où A « est follower » de B (relation représentée par un arc), n'est pas forcément une relation symétrique (B peut ne pas « être follower » de A).



C.3 / Adjacence et degré d'adjacence

On dit que deux sommets S_i et S_j sont ***adjacents (ou voisin)*** s'il existe un arc entre les deux, quelle que soit la direction de cet arc. Cependant, certains mathématiciens utilisent la notion d'adjacence orientée (limite programme), qui fait que S_i peut être le voisin orienté de S_j alors que S_j n'est pas le voisin orienté de S_i.

Dans un **graphe orienté**, on sépare le ***degré d'adjacence*** d'un sommet S_i en

- $d^+(S_i)$: nombre d'arêtes de A qui ont pour 1er élément S_i (càd nombre de flèches quittant S_i)
- $d^-(S_i)$: nombre d'arêtes de A qui ont pour 2ème élément S_i (càd nombre de flèches arrivant sur S_i)

Exemple : en reprenant l'exemple ci-dessus (**C.2**) :

- $d^+(1) = 2$ $\quad d^-(1) = 1$
- $d^+(2) = 2$ $\quad d^-(2) = 3$ (une boucle compte dans d^+ et d^-)
- $d^+(3) = 0$ $\quad d^-(3) = 1$
- $d^+(4) = 2$ $\quad d^-(4) = 1$

C.4 / Chemin et connexité

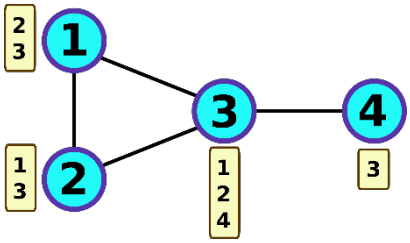
Le **chemin** doit prendre en compte l'orientation des arcs. On ne parle pas de connexité sur un graphe orienté.

D / Implémentation d'un graphe

Numériquement, un graphe peut se représenter essentiellement de deux façons. Sous la forme :

- D'une ***liste d'adjacence*** : à chaque sommet S_i de S nous allons associer une liste de ses **voisins** (éventuellement orientés). Cette représentation est pertinente dans le cas d'un graphe *creux* (ou peu *dense*) c'est-à-dire pour lequel il y a peu d'arêtes (ou d'arcs) entre les sommets.
- Ou bien d'une ***matrice d'adjacence*** : $(M_{i,j})$ carrée de taille $N \times N$ (N étant le cardinal de S) où $M_{i,j}$ vaut 0 ou 1, selon que S_i et S_j sont voisins ou non. Cette représentation est pertinente, à l'inverse, dans le cas d'un graphe dense.

D.1 / Liste d'adjacence

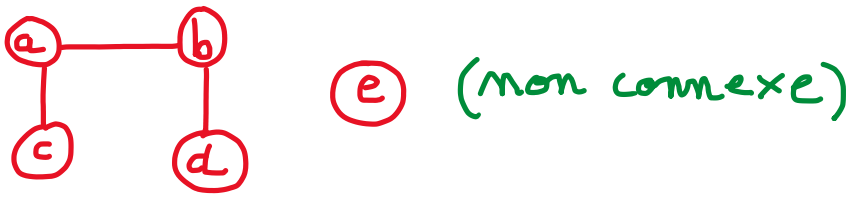


Source : *Wikipedia*

L'implémentation d'un graphe sous forme de liste d'adjacence peut être réalisée, en Python, au choix par l'intermédiaire :

- D'une liste de liste
  Exemple : Avec l'exemple ci-contre
  `[ [1,[2,3]], [2,[1,3]], [3,[1,2,4]], [4,[3]] ]`
- D'un dictionnaire
  Exemple : Avec l'exemple ci-contre
  `{ 1: [2, 3] , 2: [1, 3], 3: [1, 2, 4], 4: [3] }`

Exemple : Dessinons le graphe (non orienté) auquel correspond
`{ "a": ["b", "c"] , "b": ["a", "d"], "c": ["a"], "d": ["b"] , "e": [] }`



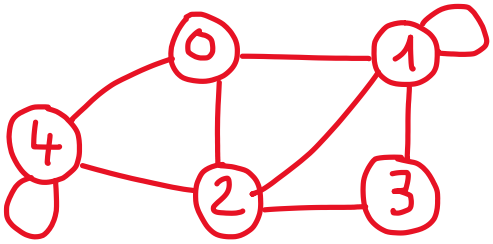
D.2 / Matrice d'adjacence

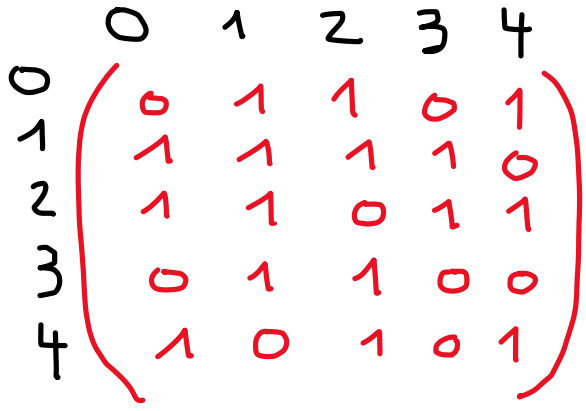
Soit un graphe (orienté ou non) $G = (S, A)$ avec $N = \text{Card}(S)$, alors en *Python* une matrice d'adjacence sera codée par une table *Numpy* M matrice carrée de taille $N \times N$, où

$$\forall (i,j) \in [\![0, N-1]\!]^2, \quad M_{i,j} = \begin{cases} 1 \text{ ou } True & \text{si } S_i \text{ et } S_j \text{ voisins} \\ 0 \text{ ou } False & \text{sinon} \end{cases}$$

Remarque : on pourra aussi utiliser une liste (de taille *N*) de listes (chacune de taille *N*) pour stocker cette matrice.

Exemple : donnons la matrice d'adjacence représentant le graphe illustré ci-dessous :





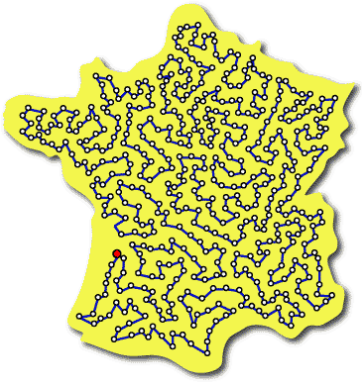
E / Graphe pondéré

On dit qu'un graphe $G = (S, A)$ est ***étiqueté*** si à ***chaque arête / arc*** de A est rattachée une étiquette, qui peut être un mot, une valeur, un symbole, …

En SI, vous allez utiliser deux graphes pondérés :
- Non orienté : Le graphe de liaisons (étiquettes = désignation des liaisons)
- Orienté : Le graphe d'états (étiquettes = conditions de franchissement, une par arc)

La plupart des problèmes d'optimisation combinatoire que vous rencontrerez en tant qu'ingénieurs se ramèneront à des ***graphes pondérés***, c'est-à-dire des graphes où les ***étiquettes sont des valeurs numériques, généralement positives***. Les valeurs associées à un arc ou une arête correspondront au coût du passage d'un sommet à un autre. Auquel cas, le problème pourra généralement se ramener à trouver un chemin entre deux sommets ayant le coût minimal, c'est-à-dire un problème de minimisation.

L'exemple le plus classique est le problème du voyageur de commerce :

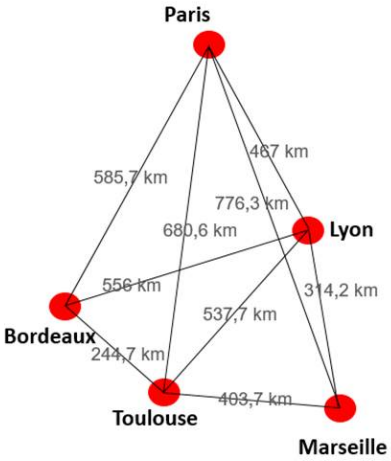


Source : *S. VEREL*
Université Côte d'Opale

Un voyageur de commerce doit passer dans une liste pré-établie de villes, mais l'ordre lui importe peu. Il souhaite déterminer l'ordre des villes à visiter (peu importe la ville dont il part, et celle par laquelle il termine) pour parcourir le moins de kilomètres en fin de tournée.

On représente alors le réseau routier sous la forme d'un graphe (en l'occurrence non orienté), où les sommets sont les villes à parcourir, les arêtes représentent les liaisons par autoroute, et les pondérations sont les km à parcourir.

→ Auquel cas le problème se ramène à une recherche du plus court chemin (couvrant, c'est-à-dire parcourant tous les sommets) d'un graphe non orienté.



Source : *Wavestone*