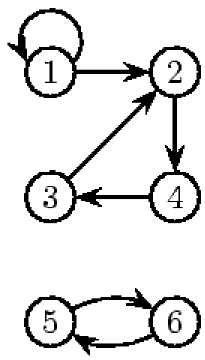
Introduction aux graphes

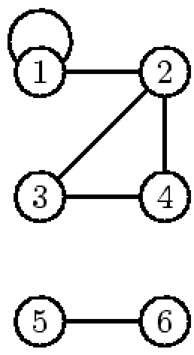
1 Premières définitions

Définition 1 (Graphe)**.** On appelle graphe la donnée d'un ensemble **fini** V de points (ou **sommets** ou vertices en anglais) et d'un ensemble E de liens entre ces points.

Définition 2 (Graphe orienté et non orienté)**.** L'ensemble E de liens peut être vu comme une relation $\mathcal{R}$ sur V $\times$ V. Lorsque cette relation est symétrique (c'est à dire lorsque l'existence d'un lien entre un sommet s_1 et un sommet s_2 équivaut à l'existence d'un lien entre le sommet s_2 et le sommet s_1) le graphe est dit non orienté. Un lien est alors appelé une **arête** (ou edge en anglais). Lorsque cette relation n'est pas symétrique, le graphe est dit orienté. On parle alors d'**arc** entre deux sommets.

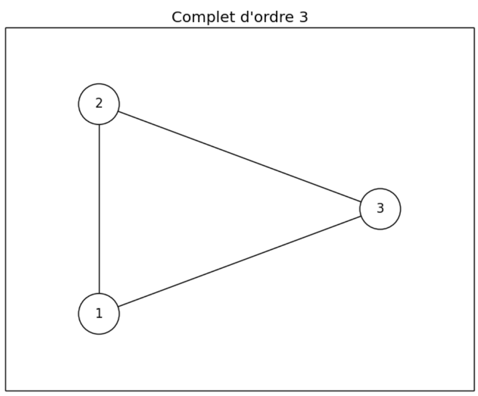
Nous noterons généralement G=(V,E) un graphe.

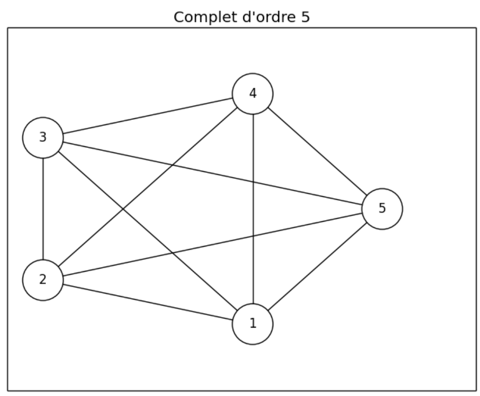




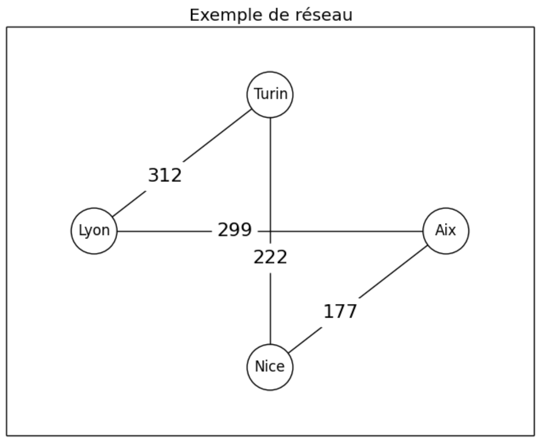
Définition 3 (ordre)**.** On appelle ordre d'un graphe le cardinal de son ensemble de sommets.

Définition 4 (graphe complet)**.** On appelle graphe complet d'ordre n et on note K_n l'unique (au nom des sommets près) graphe non orienté d'ordre n tel que toute paire de sommets (distincts) soit reliée.





Définition 5 (graphe pondéré)**.** On appelle graphe pondéré (ou valué) un graphe où les arêtes sont affectées d'un poids qui est un nombre réel. Il peut être orienté ou non. On considérera pour certains algorithmes le seul cas où le poids affecté est strictement positif. Cela représentera par exemple des situations de distances dans un réseau routier.



2 Connexité d'un graphe

Définition 6 (chemin ; chaîne)**.** Soit G=(V,E) un graphe. Un chemin P=(S,A) est défini par :
S=$\{s_1,s_2,...,s_k\}$, A=$\{s_0s_1,s_1s_2,...,s_{k-1}s_k\}$ avec $S \subset V$ et $A \subset E$.

Autrement dit, un chemin est une suite consécutive d'arcs dans un graphe orienté. Dans le cas d'un graphe non orienté on parle de chaîne.

Définition 7 (cycle)**.** Un cycle est un chemin ou une chaîne pour lequel $s_0 = s_k$ (le sommet de départ et le sommet d'arrivé sont identiques).

Définition 8 (longueur d'un chemin)**.** La longueur d'une chaîne (resp. d'un chemin) dans un graphe non pondéré est son nombre d'arêtes (resp. d'arcs) dans un graphe non orienté (resp. orienté).

Dans le cas d'un graphe pondéré, la longueur d'une chaîne (resp. d'un chemin) est la somme du poids de ses arêtes (resp. arcs) dans un graphe non orienté (resp. orienté).

Définition 9 (graphe connexe)**.** Un graphe est connexe s'il existe un chemin entre tout couple de sommets. Quand on parle de connexité pour un graphe orienté, on considère non pas ce graphe mais le graphe non-orienté correspondant.

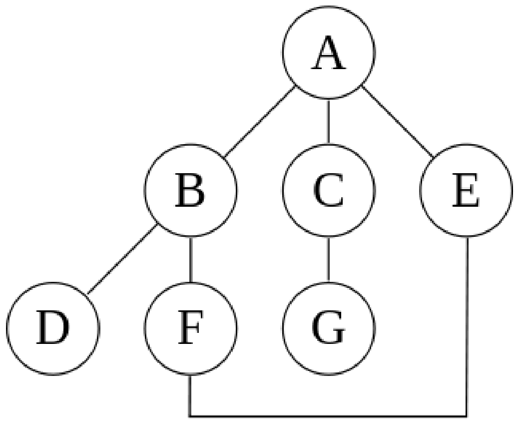
3 Parcours d'un graphe en profondeur

L'algorithme de parcours en profondeur (ou DFS, pour Depth First Search) est un algorithme de parcours de graphe qui se décrit naturellement de manière récursive. Son application la plus simple consiste à déterminer s'il existe un chemin d'un sommet à un autre et donc tester s'il est connexe ou non.

Le principe du parcours en profondeur est de partir d'un sommet s et de le marquer. Puis d'appeler récursivement la fonction de parcours en profondeur sur tous les sommets non marqués qui sont reliés à s. Nous utiliserons une liste L, initialement vide, qui correspond aux sommets marqués (déjà visités)

```
def DFS(G,s,L):
      L.append(s)
      for sommet in G.voisins(s):
            if not sommet in L:
                  DFS(G,sommet,L)
      return L
```

Exercice 1. Tester cet algorithme avec le graphe suivant :



Exercice 2. Écrire une fonction Python qui teste si un graphe est connexe ou non. On supposera que l'on dispose de toutes les structures adéquates et que l'on peut récupérer la liste des sommets de G à l'aide de G.sommets()

4 Degrés dans un graphe et parcours

Définition 10 (degré d'un sommet)**.** Dans le cas d'un graphe non orienté, on appelle degré d'un sommet s, et on note d(s), le nombre d'arêtes reliées à s.

Dans le cas d'un graphe orienté, on appelle degré sortant d'un sommet s et on note $d^+(s)$, le nombre d'arcs sortants de s. On appelle degré entrant d'un sommet s et on note $d^-(s)$, le nombre d'arcs entrants en s.

Proposition 1. *dans un graphe non orienté* $\sum d^+(s) = \sum d^-(x)$= *nombre d'arcs.*

Proposition 2. *dans un graphe non orienté* $\sum d(s)$ = *2 nombre d'arêtes.*

Exercice 3. Une ligue de foot comporte 5 équipes. Il est décidé par le bureau de la ligue que lors d'un week-end d'entraînement, chaque équipe jouera que 3 matchs. Comment organiser ce tournoi ?

5 Graphes eulériens

Définition 11 (chaîne et cycle eulérien ; graphe eulérien)**.** Une chaîne eulérienne sur un graphe G=(V,E) non orienté est une chaîne P=(S,A) pour laquelle E=A et card(E)=card(A). Cela revient à dire qu'une chaîne eulérienne est une chaîne pour laquelle toute arête est parcourue une fois et une seule.

Un cycle eulérien est un chaîne eulérienne qui est un cycle. Un graphe est dit eulérien si il contient un cycle eulérien.

Théorème 3 (Euler)**.** *Un graphe connexe est eulérien si et seulement si chacun de ses sommets est incident à un nombre pair d'arêtes.*

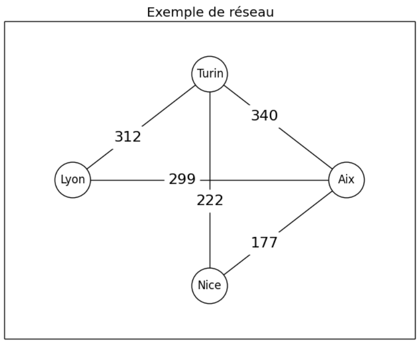
Exercice 4. Quels sont les graphes complets eulériens ?

Remarque : si seuls deux sommets s et t sont incidents à un nombre impair d'arêtes, l'ajout de l'arête st rend le graphe eulérien, en d'autres termes, on peut parcourir le graphe depuis s jusqu'à t en empruntant chaque arête exactement une fois

6 Graphes hamiltoniens

Définition 12 (graphe ; cycle hamiltonien)**.** Un graphe est hamiltonien s'il a au moins un cycle passant par tous les sommets exactement une fois, et ce cycle est appelé cycle hamiltonien.

Exercice 5. Trouver un cycle hamiltonien sur le graphe suivant :



Remarque : Comme vu sur cet exemple des cycles hamiltoniens peuvent avoir des longueurs différentes. La recherche d'un cycle (ou d'une chaîne) hamiltonien de longueur minimale sur un graphe pondéré est appelé le problème du voyageur de commerce. C'est un problème particulièrement difficile qui est dans la classe des problèmes NP-complets.

7 Graphes et matrices

Définition 13 (matrice d'adjacence)**.** Soit G=(V,E) un graphe non pondéré. On appelle matrice d'adjacence A, la matrice carrée d'ordre card(V), dont chaque élément A_{ij} est égal à 0 si il n'y a pas de d'arête liant i à j et 1 si il existe une arête reliant i à j. Si G est non orienté, sa matrice d'adjacence est symétrique.

Dans le cas d'un graphe pondéré, A_{ij} est égal au poids de l'arête liant i à j.

Proposition 4. *Soit A la matrice d'adjacence d'un graphe G non pondéré. Il y a $(A^n)_{ij}$ chemins de longueur n de s_i à s_j.*

Exercice 6. Dessiner un graphe dont la matrice d'adjacence est : $A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$

Combien y-a-t-il de chemins de longueur 100 entre le premier et le troisième sommet dans ce graphe ?

8 Colorations de graphes

Définition 14 (coloration)**.** Une coloration est une fonction associant à tout sommet une couleur, tels que deux sommets adjacents aient une couleur différente (c'est-à-dire partitionne les sommets en ensembles indépendants).

Définition 15 (nombre chromatique)**.** Le nombre chromatique d'un graphe G désigne le nombre minimum de couleurs pour colorer un graphe G. Ile est noté $\chi(\mathrm{G})$

Exercice 7. Une chaîne de 5 magasins décide d'ouvrir le maximum de magasins en nocturne avec les contraintes suivantes : les deux premiers magasins ne peuvent être ouverts ensemble en nocturne ; il en est de même pour les deux derniers ; au plus un magasin peut être ouvert en nocturne parmi les magasins 1, 3 et 4. Trouver un état qui maximise le nombre de magasins ouverts en nocturne tout en respectant les contraintes.

Définition 16 (clique)**.** Une clique est un sous-graphe induit complet, c'est-à-dire un sous-ensemble des sommets tels que chacun est connecté à tous les autres.

Théorème 5. *$\chi(G)$ est minoré par l'ordre de la plus grande clique de G et est majoré par le max$\{d(s)+1, s \in V\}$*

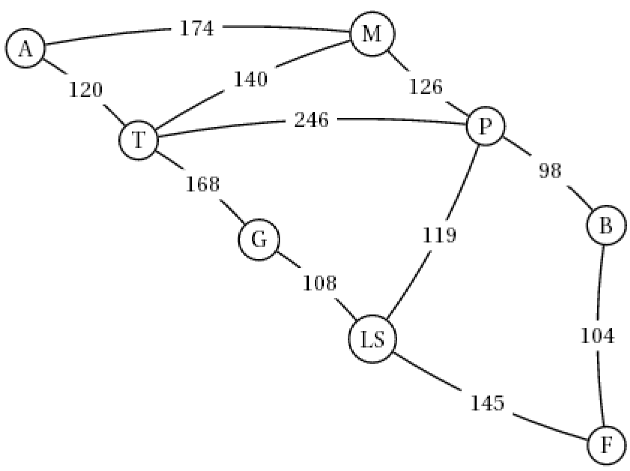
9 Problèmes de plus court chemin

On s'intéresse ici au problème de la recherche du plus court chemin sur un graphe pondéré positivement. L'algorithme de Dijsktra fournit une réponse

```
Fonction Dijkstra (nœuds, fils, distance, début, fin)
    Pour n parcourant nœuds
        n.parcouru = +infini
        n.précédent = 0
    début.parcouru = 0
    pasEncoreVu = nœuds
     Tant que pasEncoreVu != liste vide
        n1 = minimum(pasEncoreVu)
        pasEncoreVu.enlever(n1)
        Pour n2 parcourant fils(n1) // Les nœuds reliés à n1 par une arête
            Si n2.parcouru > n1.parcouru + distance(n1, n2)
                n2.parcouru = n1.parcouru + distance(n1, n2)
                n2.précédent = n1
    chemin = liste vide
    n = fin
    Tant que n != début
        chemin.ajouterAvant(n)
        n = n.précédent
    chemin.ajouterAvant(début)
    Retourner chemin
```

Exercice 8. Le graphe suivant représente les distances entre plusieurs villes d'Italie : Aoste, Milan, Parme, Turin, Gènes, La Spezia, Bologne et Florence. Chaque ville est désignée par son initiale. Appliquer à la main l'algorithme de Dijkstra pour trouver le plus court chemin entre Aoste et Florence.

Exercice 9. Montrez que la complexité de l'algorithme de Dijkstra est en $O(n^2)$ où n est le nombre de sommets.


A
M
T
P
G
B
LS
F
174
120
140
126
246
168
98
119
108
104
145