TP4

Rappels de 1ère année : théorie des graphes

4.0.1 Vocabulaire

DÉFINITION 4.1 (Graphe non orienté)

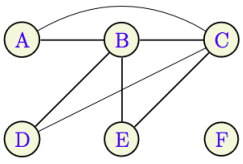
Un **graphe non orienté** est la donnée d'un couple $G = (S, A)$ où S et A sont des ensembles finis.

$S = \{s_1, ..., s_n\}$ est appelé l'ensemble des **sommets** de G.

$A = \{a_1, ..., a_p\}$ est appelé l'ensemble des **arêtes** de G.

- On appelle **ordre** du graphe le nombre de sommets.
- Une arête est un couple non ordonné de sommets : l'arête $a = \{s; s'\}$ relie les sommets s et s'. On dit que les sommets s et s' sont les **extrémités** de l'arête a.
- Un arête ayant un seul et même sommet pour extrémités est appelé une **boucle**.
- On appelle **degré** d'un sommet s noté $d(s)$ le nombre d'arêtes ayant ce sommet pour extrémité. Les boucles comptent deux fois dans ce le calcul du degré.
- Deux sommets d'un graphe sont dits **adjacents** s'il sont reliés par une arête du graphe.
- Un sommet est dit textbfisolé s'il n'est relié à aucun autre sommet du graphe ou de manière équivalente si son degré est nul.

EXERCICE 4.1. Le graphe suivant est donné par la représentation graphique suivante :



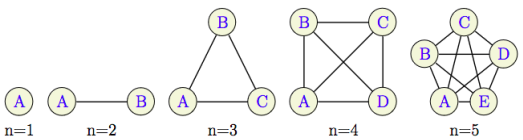
Graphe 1

Déterminer l'ensemble des sommets, l'ensemble des arêtes, l'ordre ainsi que les degrés des sommets de ce graphe.

DÉFINITION 4.2 (Graphe complet)

On dit qu'un graphe G est **complet** lorsque tous ses sommets sont deux à deux adjacents.

Voici des représentations graphiques des graphes complets d'ordre n avec $n \in [[0; 5]]$:



THÉORÈME 4.3 (Formule d'Euler)

Soit G un graphe. On note $S = \{s_1, ..., s_n\}$ les sommets de G et a le nombre d'arête de G.
Alors on a la formule

$$\sum_{i=1}^{n} d(s_i) = 2a.$$

COROLLAIRE 4.4

Le graphe complet d'ordre n possède exactement $\frac{n(n-1)}{2}$ arêtes.

4.0.2 Matrice d'adjacence

DÉFINITION 4.5 (Matrice d'adjacence)

Soit G un graphe et $S = \{s_1, ..., s_n\}$ les sommets de G.
On appelle **matrice d'adjacence** de G la matrice carrée $M \in \mathcal{M}_n(\mathbb{R})$ définie par :

$$M_{ij} = \begin{cases} 1 & \text{si } s_i \text{ et } s_j \text{ sont adjacents} \\ 0 & \text{sinon} \end{cases}$$

EXERCICE 4.2. 1. Déterminer la matrice d'adjacence du graphe 1.
2. Déterminer la matrice d'adjacence du graphe complet d'ordre 4.

REMARQUE **4.1.**

- La matrice d'adjacence d'un graphe non orienté est symétrique.
- Les diagonales des matrices d'adjacences précédemment calculées sont constituées de 0. Cela témoigne du fait que les graphes ne contiennent pas de **boucles**.

Implémentation d'un graphe en Python

La matrice d'adjacence caractérise le graphe.
En **Python**, on peut donc représenter un graphe par sa matrice d'adjacence.

Rappel sur la bibliothèque `array` de `numpy` :

• en python une matrice du type 3×3 s'écrit `np.array([[a,b,c],[d,e,f],[g,h,i]])` (déclaration ligne par ligne)
• la commande `len(M)` renvoie le nombre de lignes de la matrice
• la commande `np.zeros((n,p))` renvoie la matrice nulle de taille $n \times p$.
• la commande `np.ones((n,p))` renvoie la matrice composée uniquement de 1 de taille $n \times p$.
• la commande `np.eye((n,n))` renvoie la matrice nulle de taille $n \times n$.

EXERCICE 4.3. Ecrire un fonction `adj_complet(n)` qui renvoie la matrice d'adjacence du graphe complet d'ordre n lorsqu'on entre la valeur de l'entier n.

PROPRIÉTÉ 4.6

Soit $M \in \mathcal{M}_n(\mathbb{R})$ la matrice d'adjacence d'un graphe de sommets $S = \{s_1, ..., s_n\}$.
Alors pour tout $j \in [\![1; n]\!]$, la somme des coefficients de la i-ème ligne (ou colonne) de M est égale au degré du sommet s_i.
En d'autres termes :

$$d(s_i) = \sum_{j=1^n} M_{ij}.$$

EXERCICE 4.4. 1. Compléter la fonction suivante pour qu'elle prenne en entrée une matrice d'un graphe et renvoie en sortie la liste des degrés des sommets :

```
def degres(M):
    L=[]
    for ..........  in range( len(M) ) :
        d=0
        for ............. in range( ......... ) :
            if M[i,j] == ........ :
                d+=1
        L.append( ...... )
    return L
```

2. Tester votre fonction avec la matrice d'adjacence du graphe complet d'ordre $n = 50$. enumerate

4.0.3 Liste d'adjacence

DÉFINITION 4.7 (Liste d'adjacence)

Soit G un graphe de sommets $S = \{s_1, ..., s_n\}$.
On appelle **liste d'adjacence** de G la liste $L = [L_1, ..., L_n]$ où chaque liste L_i est la liste de sommets adjacents au sommet s_i.

EXERCICE 4.5. Déterminer la liste d'adjacence du graphe 1.

Implémentation d'un graphe en Python

La liste d'adjacence caractérose complétement le graphe.
En Python, le graphe peut être représenté par sa liste d'adjacence, comme par sa matrice d'adjacence.
Il faut être capable de passer de l'in à l'autre.

EXERCICE 4.6. Un programme python génère des graphes. Après exécution de ce programme, on obtient :

```
[['b','e','f'],['a','c','f'],['b'],[],['a'],['a','b']]
```

1. Donner une représentation graphique de graphe correspondant.
2. Ce graphe contient-il des sommets isolés ? Si oui combien ?
3. Expliciter la matrice d'adjacence de ce graphe.

EXERCICE 4.7. Ecrire une fonction Python d'en-tête `def matrix_to_list(M):` qui prend en argument une matrice carrée M étant la matrice d'adjacence d'un graphe et qui renvoie la liste d'adjacence du même graphe.

EXERCICE 4.8. Ecrire une fonction Python d'en-tête `def nb_sommets_isoles(L):` qui prend en argument une liste L d'adjacence d'un graphe et qui renvoie nombre de sommets isolés du graphe.

4.0.4 Notion de chaîne, graphes connexes

DÉFINITION 4.8

Soit G un graphe de sommets $S = \{s_1, ..., s_n\}$.

- Une **chaîne** est une suite finie de sommets telle que chaque paire de sommets consécutifs soit une arête du graphe.
- La **longueur** de la chaîne est le nombre d'arêtes qui constituent cette chaîne.
  Un chaîne de longueur p peut être notée $s_{n_0} - ...s_{n_p}$
- Une chaîne de longueur 0 est la donnée d'un sommet.
- Une chaîne est dite simple si aucune arête n'y figure plus d'une fois.

EXERCICE 4.9. Dans le graphe 1, donner une chaîne de longueur 2, puis une de longueur 8, puis une chaîne simple de longueur 4.

THÉORÈME 4.9 (Lien avec la matrice d'adjacence)

Soit G un graphe de sommets $S = \{s_1, ..., s_n\}$ et de matrice d'adjacence $M \in \mathcal{M}_n(\mathbb{R})$.
Pour tout $p \in \mathbb{N}^*$ tout (i,j)$\in [\![1; n]\!]^2$, le coefficient d'indice (i, i) de la matrice M^p est égal au nombre de chaînes de longueur p du graphe reliant les sommets s_i et s_j.

EXERCICE 4.10. Dans l'exemple du graphe 1, déterminer M^3 puis en déduire le nombre de chaînes de longueurs 2 reliant les sommets D et E.

DÉFINITION 4.10 (Graphe connexe)

On dit qu'un graphe est connexe si tout couple de sommets peut être relié par une chaîne du graphe.

REMARQUE **4.2.**
Le graphe 1 n'est clairement pas connexe en revanche le *sous-graphe* ne contenant pas le sommet F l'est.

Les graphes complets sont clairement connexes.

4.0.5 Graphes orientés

DÉFINITION 4.11 (Graphe orienté)

Un **graphe orienté** est la donnée d'un couple $G = (S, A)$ où S et A sont des ensembles finis.

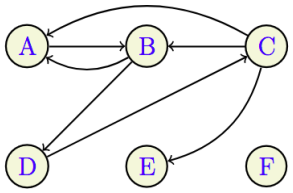
S est appelé l'ensemble des **sommets** de G.

$A \subset S \times S$ est appelé l'ensemble des **arcs** de G.

- On appelle **ordre** du graphe le nombre de sommets.
- Un arc est un couple ordonné de sommets : l'arête $a = (s; s')$ relie les sommets s et s'. On dit que les sommets s est l'**extrémité initiale** et s' l'**l'extrémité finale** de l'arc a.
- Un arc ayant un seul et même sommet pour extrémités est appelé une **boucle**.
- On appelle **degré** d'un sommet s noté $d(s)$ le nombre d'arcs ayant ce sommet pour extrémité (initiale ou finale). Les boucles comptent deux fois dans le calcul du degré.

  On peut distinguer le **degré entrant** d'un somme s, noté $d^-(s)$ qui est le nombre d'arcs ayant s pour extrémité finale et le **degré sortant**, noté $d^+(s)$ qui est le nombre d'arcs dont s est l'extrémité initiale.
- Deux sommets d'un graphe sont dits **adjacents** s'il sont reliés par un arc du graphe.
- Un graphe est dit **connexe** s'il existe une chaîne entre tout couple de sommets.

EXERCICE 4.11. Voici une représentation graphique du graphe 2 :



1. Déterminer les degrés sortant et entrant du sommet A puis en déduire son degré.
2. Le graphe est-il connexe ?
3. Le sous-graphe obtenu en enlevant le sommet F est-il connexe ?
4. Le sous-graphe obtenu en enlevant les sommets E et F est-il connexe ?

DÉFINITION 4.12 (Matrice et liste d'adjacence)

Soit G un graphe et $S = \{s_1, ..., s_n\}$ les sommets de G.
• On appelle **matrice d'adjacence** de G la matrice carrée $M \in \mathcal{M}_n(\mathbb{R})$ définie par :

$$M_{ij} = \begin{cases} 1 & \text{si } s_i \text{ et } s_j \text{ sont adjacents} \\ 0 & \text{sinon} \end{cases}$$

On appelle **liste d'adjacence** de G la liste $L = [L_1, ..., L_n]$ où chaque liste L_i est la liste de sommets adjacents au sommet s_i.

EXERCICE 4.12. Déterminer la matrice et la liste d'adjacence du graphe 2.

REMARQUE **4.3.**
• La matrice d'adjacence d'un graphe orienté n'est **pas toujours symétrique**.
• La somme des lignes ou des colonnes de la matrice d'adjacence ne sont plus égales et on a :

$$\sum_{j=1}^{n} M_{ij} = d^+(s_i), \quad \sum_{i=1}^{n} M_{ij} = d^-(s_j).$$

EXERCICE 4.13. Ecrire une fonction Pyhon d'en-tête `def test_oriente(M):` qui prend en argument la matrice d'adjacence d'un graphe t qui renvoie 1 ou 0 selon que le graphe est orienté ou non.

REMARQUE **4.4.**
Le théorème concernant la matrice M^p s'adapte au cas des graphes orienté.

THÉORÈME 4.13 (Caractérisation des graphes orientés connexes)

Soit M la matrice d'adjacence d'un graphe orienté.
G est connexe si et seulement si la matrice $I_n + M + ... + M^{n-1}$ a **tous** ses coefficients strictement positifs.

EXERCICE 4.14. On considère le sous-graphe du graphe 2 suivant :

1. Déterminer la matrice M d'adjacence du graphe.
2. Calculer M^2 puis M^3.
3. Montrer que G est connexe.

Extrait d'Ecricome 2023 :

Soient p un entier naturel non nul et G un graphe non pondéré orienté à p sommets. On note $s_0, s_1, \ldots, s_{p-1}$ les sommets de G.

1. Dans cette question uniquement, on suppose que $p = 4$ et que la matrice d'adjacence du graphe G est la matrice $M = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$.
   (a) Représenter les sommets et les arêtes du graphe G sous forme d'un diagramme.
   (b) Le graphe G est-il connexe? Justifier votre réponse.
2. Dans cette question et les suivantes, on revient au cas général..
   Soit s un sommet de G. On dit que le sommet t est un voisin de s quand $s \neq t$ et (s, t) est une arête du graphe.
   Comme le graphe est orienté, si t est un voisin de s, alors s n'est pas forcément un voisin de t.
   On appelle liste d'adjacence du graphe G, une liste de p sous-listes telle que, pour tout entier k de $[\![0; p-1]\!]$, la sous-liste située à la position k contient tous les numéros des sommets voisins de s_k.
   Par exemple, la liste d'adjacence du graphe étudié à la question précédente est :
   ```
   L = [[0,3], [0,1,2], [0,1,2], [0,3]]
   ```
   Écrire une fonction en langage Python, nommée `matrice_vers_ligne`, prenant en entrée la matrice d'adjacence `M` d'un graphe G (définie sous forme de listes de listes) et renvoyant la liste d'adjacence de G.
3. On cherche à écrire une fonction en langage Python permettant d'obtenir la longueur du plus court chemin menant d'un sommet de départ s_i à chaque sommet du graphe G.
   On souhaite pour cela appliquer un algorithme faisant intervenir les variables suivantes :
   - Une liste `distances` à p éléments, où l'élément situé à la position k sera égal, à la fin de l'algorithme, à la longueur du plus court chemin menant du sommet de départ s_i au sommet s_k.
   - Une liste `a_explorer` contenant tous les sommets restant à traiter.
   - Une liste `marques` contenant tous les sommets déjà traités.

Nous donnons ci-dessous la description de l'algorithme :

- Initialisation des trois listes décrites ci-dessus :
  - Initialement, chaque élément de la liste `distances` est égal à p, à l'exception du sommet s_i, auquel on affecte la distance 0.
  - La liste `marques` ne contient initialement que le numéro du sommet de départ s_i.
  - La liste `a_explorer` ne contient initialement que le numéro du sommet de départ s_i.
- Tant que la liste `a_explorer` n'est pas vide, on répète les opérations suivantes :
  - Nommer `s` le premier sommet de la liste `a_explorer`, et le retirer de cette liste.
  - Pour chaque voisin `v` du sommet `s` : si `v` n'est pas dans la liste `marques`, on l'ajoute à la fin de la liste `a_explorer`, et on lui affecte une distance égale à `distances[s]+1`.

(a) On considère le graphe orienté G étudié à la question 2.
Donner la valeur de la liste `distances` à l'issue de l'exécution de l'algorithme décrit ci-dessus, lorsqu'on l'applique au graphe G en choisissant s_1 comme sommet de départ.

(b) Recopier et compléter la fonction suivante, prenant en entrée la liste d'adjcacence `L` du graphe G et le numéro `i0` du sommet de départ s_i, et renvoyant la liste `distances` après exécution de l'algorithme décrit ci-dessus.

```
def parcours(L, i0):
    p = len(L)
    distances = ...................
    distances[i0] = 0
    a_explorer  = ...................
    marques = ...................
    while .....................:
        s = ...................
        ...................
        for v in ..................  :
            if v not in marques :
                marques.append(v)
                ...................
                ...................
    return distances
```

(c) Modifier la fonction précédente pour qu'elle renvoie la liste de tous les sommets s pour lesquels il existe un chemin menant du sommet de départ s_i au sommet s.