Table des matières

1 Définitions **1**

2 Représentation d'un graphe **2**

2.1 Représentation via la matrice d'adjacence 2

2.2 Représentation via les listes d'adjacence 3

3 Propriétés des graphes **6**

4 Graphes pondérés **8**

1 Définitions

Definition 1. On appelle *graphe (fini)* la donnée d'un couple $G = (S, A)$ où

- S est un ensemble fini, appelé ensemble des *sommets (ou noeuds)*
- A est un sous-ensemble de S^2, appelé ensemble des *arêtes* et vérifiant

$$\forall x \in S,\ (x, x) \notin A$$

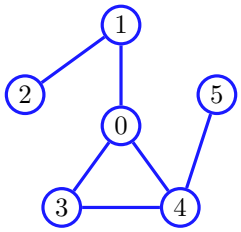
Remarque 1. Il est important de noter qu'un sommet ne peut pas être relié à lui même via une arête.

Definition 2. Soit $G = (S, A)$ un graphe. On dit que le graphe G est *non-orienté* si

$$\forall (x, y) \in S^2, (x, y) \in A \iff (y, x) \in A$$

Dans le cas, contraire, on dit que le graphe G est orienté.

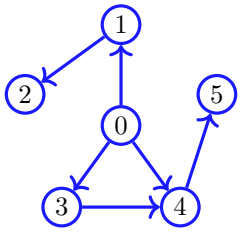
Exemple 1. Considérons le graphe non orienté



Il est mathématiquement modélisé par

$$S = \{0, 1, 2, 3, 4, 5\}$$
$$A = \{(0, 1), (0, 3), (0, 4), (1, 0), (1, 2), (2, 1), (3, 0), (3, 4), (4, 0), (4, 3), (4, 5), (5, 4)\}$$

Exemple 2. Considérons le graphe orienté



Il est mathématiquement modélisé par

$$S = \{0, 1, 2, 3, 4, 5\}$$
$$A = \{(0, 1), (0, 3), (0, 4), (1, 2), (3, 4), (4, 5)\}$$

Definition 3. Soit $G = (S, A)$ un graphe. Soit $x \in S$ un sommet.

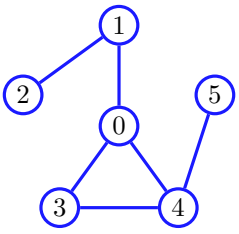
- Soit $y \in S$. On dit que y est un *voisin* de x si $(x, y) \in A$. Autrement dit, y est un voisin de x si x et y sont reliés par une arête partant de x.
- On appelle *voisinage* de x l'ensemble $V(x) \subset S$ des voisins de x, *i.e.*

$$V(x) \overset{\text{def}}{=} \{y \in S \mid (x, y) \in A\}$$

- On appelle *degré de* x le nombre de voisins de x. On note ce nombre $\deg(x)$, *i.e.*

$$\deg(x) \overset{\text{def}}{=} \text{Card}(V(x))$$

Exemple 3. Reprenons l'exemple du graphe



On a

- $V(0) = \{1, 3, 4\}$ et $\deg(0) = 3$
- $V(1) = \{0, 2\}$ et $\deg(1) = 2$
- $V(2) = \{1\}$ et $\deg(2) = 1$
- $V(3) = \{0, 4\}$ et $\deg(3) = 2$
- $V(4) = \{0, 3, 5\}$ et $\deg(4) = 3$
- $V(5) = \{4\}$ et $\deg(5) = 1$

2 Représentation d'un graphe

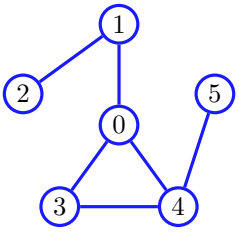
2.1 Représentation via la matrice d'adjacence

Definition 4. Soit $n \in \mathbb{N}^*$. Soit $G = (S, A)$ un graphe à n sommets. On note $S = \{s_1, s_2, \dots, s_n\}$. On appelle *matrice d'adjacence de* G la matrice $M = (m_{i,j})_{\substack{1 \leqslant i \leqslant n \\ 1 \leqslant j \leqslant n}} \in \mathscr{M}_n(\mathbb{R})$ définie par

$$\forall (i, j) \in [\![1, n]\!]^2,\ m_{i,j} = \mathbb{1}_A((s_i, s_j)) = \begin{cases} 1 & \text{si } (s_i, s_j) \in A \\ 0 & \text{sinon} \end{cases}$$

Remarque 2. Si l'ensemble des sommets est fixé, alors la matrice d'adjacence caractérise le graphe. Ainsi, donner le graphe ou la matrice d'adjacence revient au même.

Exemple 4. Reprenons l'exemple du graphe



Sa matrice d'adjacence est

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

On remarque que la diagonale contient des 0. C'est toujours le cas, cf définition d'un graphe.

On définit cette matrice en **Python** via les commandes

```
import numpy as np
M = np.array([[0,1,0,1,1,0],[1,0,1,0,0,0],[0,1,0,0,0,0],
              [1,0,0,0,1,0],[1,0,0,1,0,1],[0,0,0,0,1,0]])
```

On remarque que ${}^tM = M$. C'est en fait une propriété générale des graphes non orientés.

Comment calculer le degré d'un sommet à l'aide de la matrice d'adjacence ?

- Mathématiquement : soit $k \in S$.
$$\deg(k) = \sum_{j=1}^{6} m_{k+1,j}$$
Attention au décalage de numérotation, les lignes et les colonnes des matrices sont numérotées à partir de 1.
- En **Python** :

```
for k in range(6):
    deg = np.sum(M[k])
    print(f'Le degré du sommet {k} est {deg}')
```

On prend en compte le fait que **Python** numérote à partir de 0.

Proposition 1. *Soit G un graphe et soit M sa matrice d'adjacence. Alors*

G est non orienté ssi M est symétrique

Exercice 1 : Ecrire une fonction **Python**, nommée `Graph_orient`, qui :

- prend en argument la matrice d'adjacence `M` d'un graphe
- renvoie le booléen `True` si ce graphe est orienté et `False` sinon

On importera la bibliothèque `numpy` sous l'alias `np`.

```
import numpy as np
def graph_orient(M):
    return not np.all(M == np.transpose(M))
```

Exercice 2 : Tracer les graphes pour chacune des matrices d'adjacence suivantes. On numérotera les sommets à partir de 1.

1. $M = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
2. $M = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$
3. $M = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$
4. $M = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$

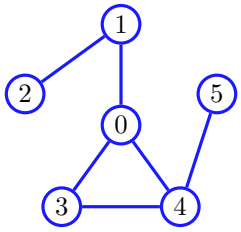
2.2 Représentation via les listes d'adjacence

Definition 5 (Informatique)**.** Soit $G = (S, A)$ un graphe. Soit $x \in S$. On appelle *liste d'adjacence* du sommet x la liste des voisins de x. Autrement dit, la liste d'adjacence du sommet x contient les éléments du voisinage de x.

Remarque 3. L'ordre des éléments dans la liste d'adjacence de x n'a pas d'importance.

Remarque 4. Si l'ensemble des sommets est fixé, alors la liste des listes d'adjacence caractérise le graphe. Ainsi, donner le graphe ou la liste des listes d'adjacence revient au même.

Exemple 5. Reprenons l'exemple du graphe



En **Python**, la liste des listes d'adjacence peut être obtenue de plusieurs manières :

- De manière directe en lisant le graphe :

```
L = [[1,3,4],[0,2],[1],[0,4],[0,3,5],[4]]
```

- Via la matrice d'adjacence :

```
import numpy as np
def list_ad_from_Sym_matrix(M):
    nbS = len(M)
    L = [[] for k in range(nbS)]
    for i in range(nbS):
        for j in range(i+1,nbS):
            if M[i,j] == 1:
                L[i].append(j)
                L[j].append(i)
    return L
```

Remarque 5. On a utilisé dans le programme précédent le fait que la matrice d'adjacence est symétrique. Ceci a permis de ne visiter que la partie triangulaire supérieure de la matrice.

Si le graphe est orienté, il faut tester tous les coefficients de la matrice :

```
import numpy as np
def list_ad_from_matrix(M):
    nbS = len(M)
    L = [[] for k in range(nbS)]
    for i in range(nbS):
        for j in range(nbS):
            if M[i,j] == 1:
                L[i].append(j)
    return L
```

Exercice 3 : Compléter la fonction **Python** suivante qui

- prend en argument la liste `L` des listes d'adjacence d'un graphe
- renvoie la matrice d'adjacence de ce graphe

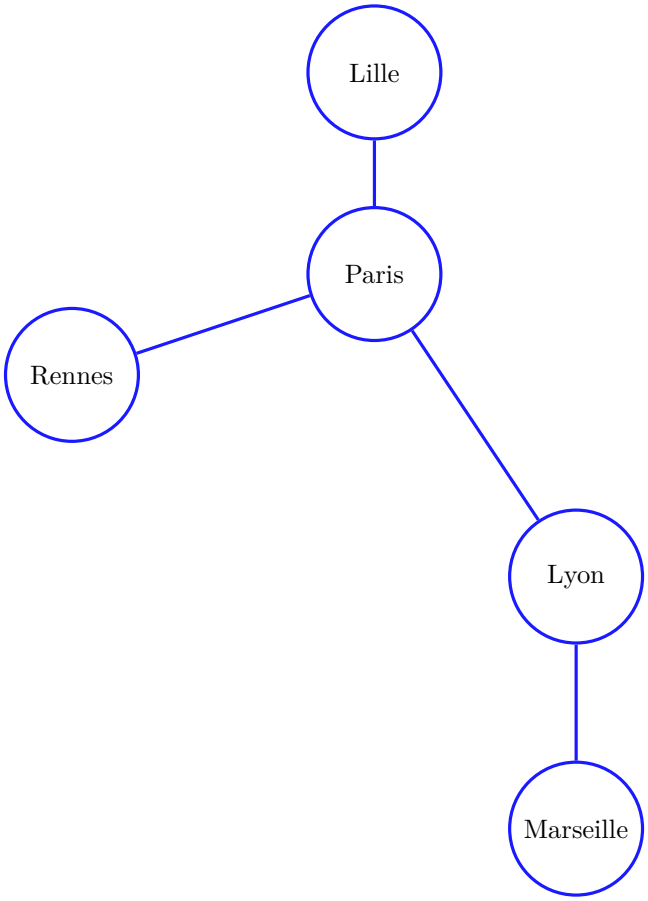
```
import numpy as np
def ad_matrix_from_list(L):
    nbS = __________
    M = np.zeros([nbS,nbS])
    for i in range(nbS):
        nbV = _______________
        for j in range(nbV):
            M[i,L[i][j]] = _____
    return M
```

Exercice 4 : Ecrire une fonction **Python**, nommée `List_deg`, qui :

- prend en argument la liste `L` des listes d'adjacence d'un graphe
- renvoie la liste des degrés des sommets de ce graphe

```
1  import numpy as np
2  def list_deg(L):
3      D = []
4      for k in range(len(L)):
5          D.append(len(L[k]))
6      return D
```

Exemple 6. On peut aussi rassembler les listes d'adjacence dans un dictionnaire. Exemples d'une partie du réseau ferré français :




```
1  Dictionnaire = {'Paris': ['Rennes','Lyon','Lille'], 'Rennes': ['Paris'],
2                  'Lyon' : ['Paris', 'Marseille'], 'Marseille': ['Lyon'], 'Lille': ['Paris']}
```

On accède aux voisins de `'Paris'` en rentrant dans la console la commande `Dictionnaire['Paris']`. La console renvoie `['Rennes', 'Lyon', 'Lille']`.

3 Propriétés des graphes

Théoreme 2 (Lemme des poignées de main)**.** *Soit $G = (S, A)$ un graphe fini non orienté. Alors*

$$2\,\mathrm{Card}(A) = \sum_{x \in S} \deg(x)$$

En particulier, G possède un nombre pair de sommets ayant un degré impair.

Remarque 6. On considère une soirée où se rencontrent n personnes. On construit alors le graphe possédant n sommets (chaque sommet représente une personne) et où deux sommets sont reliés par une arête si les deux personnes se serrent la main. D'après le lemme des poignées de mains, il y a un nombre pair de personnes qui serreront la main à un nombre impair de personnes.

Démonstration. On remarque qu'additionner le nombre de sommets attachés à chaque arête ou le nombre d'arêtes attachées à chaque sommet donne le même résultat. En effet, notons, pour tout $x \in S$ et pour tout $a \in A$,

$$f(x,a) = \begin{cases} 1 & \text{si } x \text{ est un sommet attaché à l'arête } a \\ 0 & \text{sinon} \end{cases}$$

On a alors, par interversion de sommes finies :

$$\sum_{x \in S} \sum_{a \in A} f(x,a) = \sum_{a \in A} \sum_{x \in S} f(x,a)$$

Or,

- $\displaystyle\sum_{x \in S} \sum_{a \in A} f(x,a) = \sum_{x \in S} \deg(x)$
- $\displaystyle\sum_{a \in A} \sum_{x \in S} f(x,a) = \sum_{a \in A} 2 = 2\,\mathrm{Card}(A)$

□

Definition 6. Soit $G = (S, A)$ un graphe. Soient x et y deux sommets de G. Soit $k \in \mathbb{N}^*$.

- On dit qu'il existe un *chemin de longueur k allant de x à y* si on peut relier x à y par k arêtes successives. Plus précisément, si

$$\exists (s_0, s_1, \dots, s_k) \in S^{k+1},\ \begin{cases} x = s_0 \\ y = s_k \\ \forall i \in [\![0, k-1]\!],\ (s_i, s_{i+1}) \in A \end{cases}$$

  Si c'est le cas, on note $x \xrightarrow{k} y$.
- Par convention, on dit qu'il existe un unique chemin de longueur 0 allant de x à lui-même : $x \xrightarrow{0} x$.
- On dit qu'il existe un *chemin allant de x à y* (on dit aussi que *x mène à y*) si il existe $k \in \mathbb{N}$ et un chemin de longueur k allant de x à y. Si c'est le cas, on note $x \to y$.
- Si x mène à y, on note

$$\mathrm{d}(x,y) \overset{\text{def}}{=} \min \left\{ k \in \mathbb{N} \mid x \xrightarrow{k} y \right\}$$

  et on appelle ce nombre la *distance de x à y*. Pour le dire autrement, $\mathrm{d}(x,y)$ est la longueur minimale parmi tous les chemins reliant x à y.

Remarque 7. Soit $G = (S, A)$ un graphe. On a

$$\begin{aligned} G \text{ est non orienté} &\iff \left(\forall (x,y) \in S^2,\ x \xrightarrow{1} y \iff y \xrightarrow{1} x \right) \\ &\iff \left(\forall k \in \mathbb{N},\ \forall (x,y) \in S^2,\ x \xrightarrow{k} y \iff y \xrightarrow{k} x \right) \end{aligned}$$

Remarque 8. La relation $\to$ est transitive : si $x \to y$ et $y \to z$, alors $x \to z$. Il suffit de concaténer les chemins.

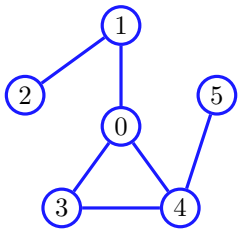
Definition 7. Soit $G = (S, A)$ un graphe. On dit que G est *connexe* si

$$\forall (x, y) \in S^2,\ x \to y$$

Autrement dit, G est connexe si tous les sommets sont reliés entre eux par un chemin.

Remarque 9. Le mot connexe est de la même famille que le mot connecté : G est connexe si tous ses sommets sont connectés.

Exemple 7. Reprenons l'exemple du graphe



Ce graphe est connexe et on a, par exemple,

- $\mathrm{d}(0,1) = 1$
- $\mathrm{d}(0,2) = 2$
- $\mathrm{d}(2,5) = 4$

Proposition 3. *Soit $G = (S, A)$ un graphe à n sommets ($n \geqslant 1$). On note $S = \{s_1, \dots, s_n\}$ et on note M la matrice d'adjacence du graphe G.*

- *Soient x et y deux sommets. Si il existe un chemin allant de x à y, alors on peut trouver en particulier un chemin de longueur inférieure ou égale à $n-1$,* i.e. $\mathrm{d}(x, y) \leqslant n-1$.
- *Soit $(i, j) \in [\![1, n]\!]^2$ et soit $k \in \mathbb{N}$. Alors $[M^k]_{i,j}$ est le nombre de chemins de longueur k allant de s_i à s_j.*
- *G est connexe si et seulement si tous les coefficients de la matrice $\sum_{k=0}^{n-1} M^k$ sont strictement positifs.*

Démonstration.

- Soient x et y deux sommets. On suppose qu'il existe un chemin allant de x à y. Fixons un chemin de longueur minimale, noté $(s_0, s_1, \dots, s_k)$ et supposons que $k > n-1$. On a alors $k+1 \geqslant n+1$ et donc il y a nécessairement deux sommets égaux parmi les sommets $s_0, \dots, s_k$. Si il y a deux sommets égaux, c'est donc qu'il y a une boucle, que l'on peut retirer du trajet pour aller plus vite. Il ne s'agit donc pas d'un trajet minimal, c'est absurde.
- Notons, pour tout $(i, j) \in [\![1, n]\!]^2$ et pour tout $k \in \mathbb{N}$, $c_{i,j}(k)$ le nombre de chemins de longueur k reliant s_i à s_k. Montrons par récurrence : $\forall k \in \mathbb{N}$, $\mathcal{P}(k)$
  où $\mathcal{P}(k)$: « pour tout $(i, j) \in [\![1, n]\!]^2$, $[M^k]_{i,j}$ est le nombre de chemins de longueur k allant de s_i à s_j »
  Initialisation :
  D'une part, $M^0 = I_n$.
  D'autre part, on a bien un unique chemin de longueur 0 reliant x à lui même pour tout sommet x. Et si x et y sont deux sommets distincts, alors x et y ne sont pas reliés par un chemin de longueur 0. D'où $\mathcal{P}(0)$.
  Hérédité : soit $k \in \mathbb{N}$. Supposons $\mathcal{P}(k)$. Montrons $\mathcal{P}(k+1)$.
  Soit $(i, j) \in [\![1, n]\!]^2$. Soit $(x_0, \dots, x_{k+1})$ un chemin de longueur $k+1$ reliant s_i à s_j. Ce chemin est la concaténation d'un chemin de longueur k reliant s_i à s_l et d'un chemin de longueur 1 reliant s_l à s_j, où s_l est un sommet tel que (s_l, s_j) est une arête, *i.e.* tel que $[M]_{l,j} = 1$.
  On a alors

$$\begin{aligned} c_{i,j}(k+1) &= \sum_{\substack{l \in [\![1,n]\!] \\ (s_l, s_j) \in A}} c_{i,l}(k) \\ &= \sum_{l=1}^{n} c_{i,l}(k)[M]_{l,j} && \text{(par définition de la matrice d'adjacence)} \\ &= \sum_{l=1}^{n} [M^k]_{i,l}[M]_{l,j} && \text{(par hypothèse de récurrence)} \\ &= [M^{k+1}]_{i,j} \end{aligned}$$

  D'où $\mathcal{P}(k+1)$.

- Supposons G connexe. Alors pour tout $(i,j) \in [\![1,n]\!]^2$, il existe un chemin reliant s_i à s_j de longueur inférieure ou égale à $n-1$. Ainsi,
$$\left[\sum_{k=0}^{n-1} M^k\right]_{i,j} = \sum_{k=0}^{n-1} [M^k]_{i,j} > 0$$
car tous les coefficients sont positifs et l'un au moins d'entre eux est strictement positif.

  Réciproquement, supposons que tous les coefficients de la matrice $\sum_{k=0}^{n-1} M^k$ sont strictement positifs. Alors pour tout $(i,j) \in [\![1,n]\!]^2$, il existe $k \in [\![0,n-1]\!]$ tel que $[M^k]_{i,j} > 0$ et donc il existe un chemin reliant s_i à s_j. Ceci prouve que G est connexe.

□

Exercice 5 : Ecrire une fonction **Python**, nommée `graph_connexe`, qui

- prend en argument la matrice d'adjacence `M` d'un graphe
- renvoie le booléen `True` si ce graphe est connexe et `False` sinon

On importera la bibliothèque `numpy` sous l'alias `np` et la bibliothèque `numpy.linalg` sous l'alias `al`. On rappelle que la commande `al.matrix_power(M,k)` calcule la puissance `k`$^{\text{e}}$ de la matrice `M`.

```
1  import numpy as np
2  import numpy.linalg as al
3  def graph_connexe(M):
4      nbS = len(M)
5      T = np.zeros([nbS,nbS])
6      for k in range(nbS):
7          T += al.matrix_power(M,k)
8      return np.all(T > 0)
```

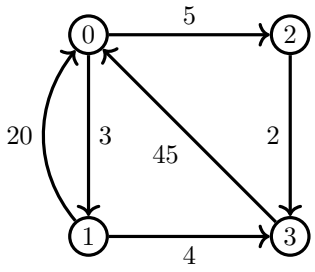
4 Graphes pondérés

Definition 8. On appelle *graphe (fini) pondéré* la donnée d'un triplet $G = (S, A, \nu)$ où

- (S, A) est un graphe
- $\nu : A \to]0, +\infty[$

Pour toute arête $a \in A$, le réel $\nu(a)$ est appelé *poids* associé à cette arête.

Exemple 8. Exemple d'un graphe pondéré et orienté.



Exemple 9. Exemple de graphe pondéré et non orienté. Implémentation de l'algorithme de Dijkstra sur ce graphe.

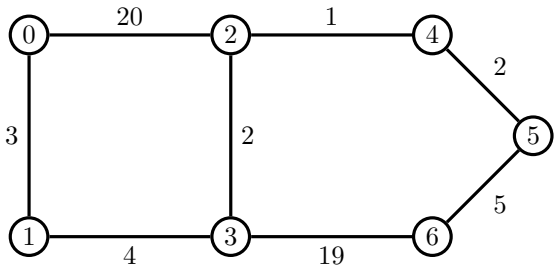


Tableau :

0	1	2	3	4	5	6
0	∞	∞	∞	∞	∞	∞
—	$\mathbf{3_0}$	20_0	∞	∞	∞	∞
—	—	20_0	$\mathbf{7_1}$	∞	∞	∞
—	—	$\mathbf{9_3}$	—	∞	∞	26_3
—	—	—	—	$\mathbf{10_2}$	∞	26_3
—	—	—	—	—	$\mathbf{12_4}$	26_3
—	—	—	—	—	—	$\mathbf{17_5}$

Distance 0-6 = 17

Chemin = 0-1-3-2-4-5-6

Visualisation du chemin le plus court :