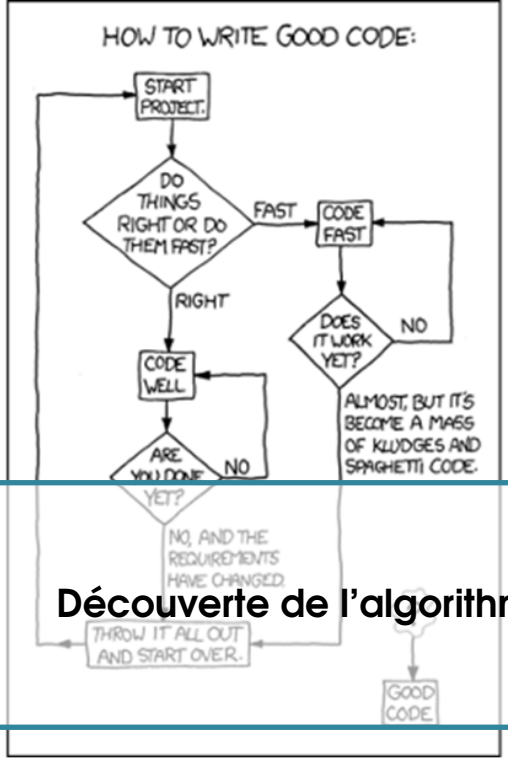
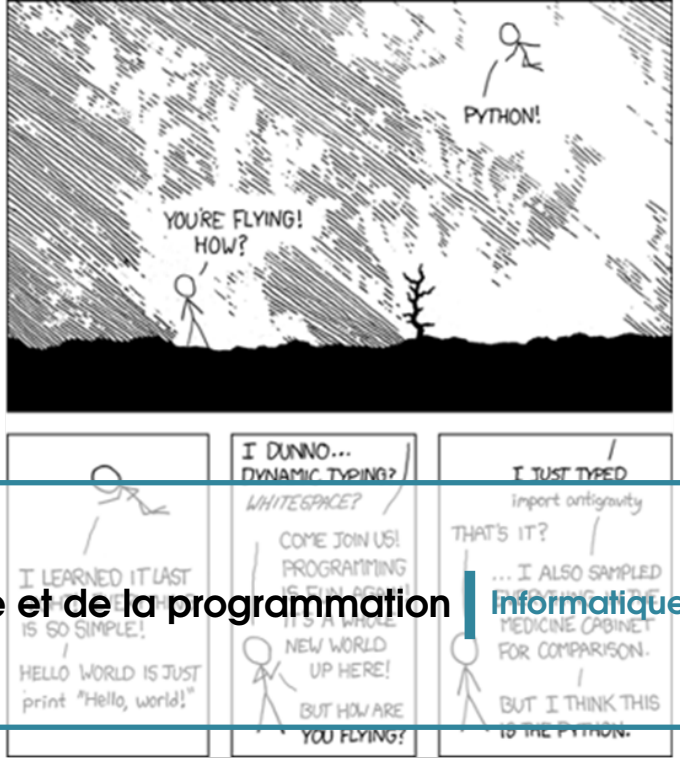







Cours

# Chapitre 11

# Introduction aux graphes

***Savoirs et compétences :***

- ❒ *Dictionnaires.*





*P. Bessonnat – Xavier Pessoles*
*MPSI – PTSI*

# 1 Vocabulaire des graphes

**Définition Graphe** Un graphe est un ensemble de **sommets** et **relations** entre ces sommets.
Lorsque deux sommets sont en relation, on dit qu'il existe une **arête** entre ces sommets.

**Définition Graphe non orienté – Arêtes** Un graphe non orienté $G$ est un couple $G=(S,A)$, où $S$ est un ensemble fini de sommets (appelés aussi nœuds) et où $A$ est un ensemble fini de paires ordonnées de sommets, appelées arêtes.
On note $x-y$ l'arête $\{x,y\}$. $x$ et $y$ sont les deux extrémités de l'arête.

**Définition Graphe orienté – Arcs** [**ref_01**] Un graphe orienté $G$ est un couple $G=(S,A)$, où $S$ est un ensemble fini de sommets et où $A$ est un ensemble fini de paires ordonnées de sommets, appelées arcs.
On note $x\to y$ l'arc $(x,y)$. $x$ est l'extrémité initiale de l'arc, $y$ est son extrémité terminale. On dit que $y$ est successeur de $x$ et que $x$ est prédécesseur de $y$.

▪ **Exemple**

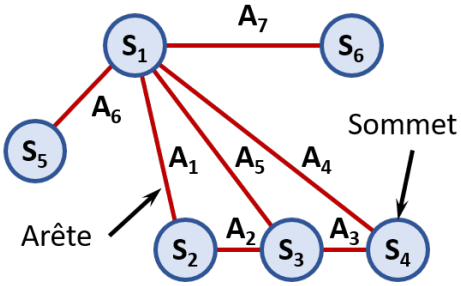



FIGURE 1 – Graphe non orienté

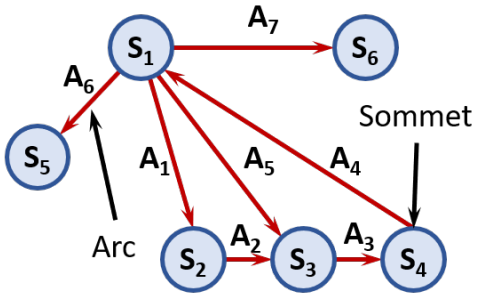



FIGURE 2 – Graphe orienté

▪

R On peut noter le graphe non orienté $G=([\![1,6]\!],E)$ où $E=(\{1,2\},\{2,3\},\{3,4\},\{1,4\},\{1,3\},\{1,5\},\{1,6\})$ désigne les arêtes.
On peut noter le graphe orienté $G=([\![1,6]\!],E)$ où $E=((1,2),(2,3),(3,4),(1,4),(1,3),(1,5),(1,6))$ désigne les arcs.

**Définition Adjacence** Deux arcs (resp. arêtes) d'un graphe orienté (resp. non orienté) sont dits adjacents s'ils ont au moins une extrémité commune.
Deux sommets d'un graphe non orienté sont dits adjacents s'il existe une arête les joignant.
Dans un graphe orienté, le sommet $y$ est dit adjacent au sommet $x$ s'il existe un arc $x\to y$.

**Définition Graphes pondérés** Étiqueter les arêtes d'un graphe $(S,A)$ (orienté ou non), c'est se donner une fonction $f:A\to V$ (où $V$ est un ensemble de valeurs). On dit qu'un graphe est pondéré si ses arêtes sont étiquetées par des nombres. On parlera alors du poids d'une arête.

# 2 Chemins

**Définition Chemin dans un graphe** On appelle chemin dans un graphe une suite finie $\{S_0,\dots,S_{n-1}\}$ de $n$ sommets tels que pour tout $i\in[\![0,n-1[\![$, une arête relie $S_i$ à $S_{i+1}$. On dit que ce chemin relie le sommet de départ $S_0$ au sommet de fin $S_{n-1}$.
Dans le cas d'un graphe non orienté, les arêtes sont notées $\{S_i,S_{i+1}\}$ pour $i\in[\![0,n-1[\![$.
Dans le cas d'un graphe orienté, les arcs sont notées $(S_i,S_{i+1})$ pour $i\in[\![0,n-1[\![$.

**Définition Chemin fermé** Chemin dont le sommet de départ et le sommet d'arrivée sont identiques.

**Définition Chemin élémentaire** Chemin n'empruntant que des arêtes distinctes.

**Définition Chemin simple** Chemin tel que les $n-2$ sommets intermédiaires si, pour $i\in[\![1,n-1[\![$ soient deux à deux distincts et tous distincts du sommet de départ $S_0$ et du sommet d'arrivée $S_{n-1}$ et tels que ce chemin n'est pas de la forme $a,b,a$ dans le cas non-orienté.

**Définition Circuit** Chemin fermé de longueur non nulle.

**Définition Cycle** Circuit élémentaire (chemin fermé de longueur non nulle dont toutes les arêtes sont distinctes).

**Définition Cycle simple** Chemin fermé et simple de longueur non nulle.

**Définition Chemin et cycle eulérien** Chemin (resp. cycle) contenant une et une seule fois toutes les arêtes du graphe.

R Pour certains auteurs, un chemin élémentaire est ce que nous avons appelé un chemin simple et réciproquement. Pour d'autres, un cycle est ce que nous avons appelé un cycle simple.

▪ **Exemple**

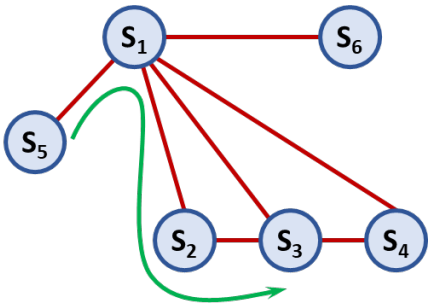



FIGURE 3 – Chemin

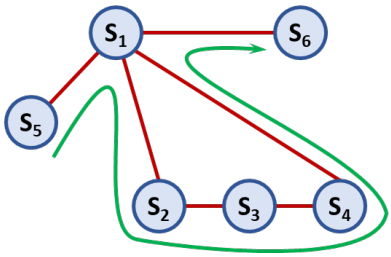



FIGURE 4 – Chemin eulérien

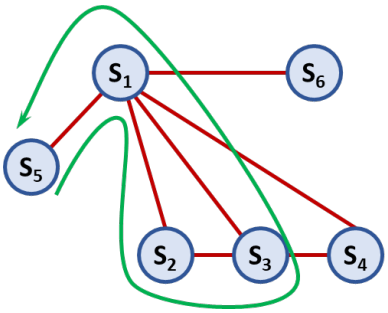



FIGURE 5 – Cycle simple

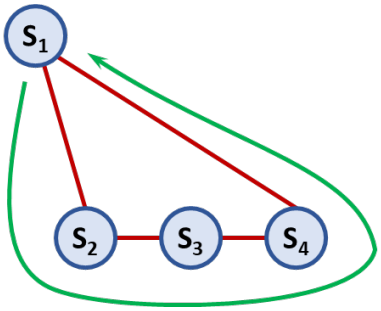



FIGURE 6 – Cycle eulérien

▪

**Définition Connexité dans les graphes non orientés** Un graphe $G = (S, A)$ est dit connexe si, pour deux sommets quelconques $S_i$ et $S_j$ de $S$, il existe un chemin de $S_i$ à $S_j$.

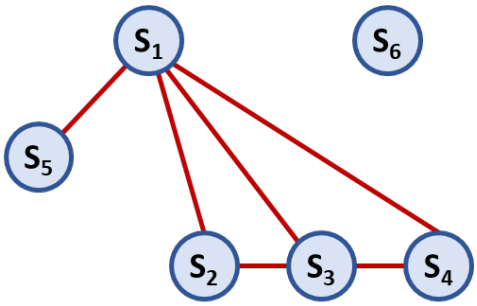



FIGURE 7 – Graphe ayant 2 composantes connexes

▪ **Exemple** ▪

## 2.1 Notations

**Définition Degré d'un sommet** On appelle degré d'un sommet $s$ et on note $d(s)$ le nombres d'arcs (ou d'arêtes) dont $s$ est une extrémité.

**Définition Degré entrant et sortant** On note $s$ le sommet d'un graphe orienté. On note :
$d_+(s)$ le demi-degré extérieur de $s$, c'est-à-dire le nombre d'arcs ayant leur extrémité initiale en $s$ (ces arcs sont dits incidents à $s$ vers l'extérieur) ;
- $d_-(s)$ le demi-degré intérieur de $s$, c'est-à-dire le nombre d'arcs ayant leur extrémité finale en $s$ (ces arcs sont dits incidents à $s$ vers l'intérieur).

Dans ce cas, on a $d^\circ(s) = d_-(s) + d_+(s)$.

▪ **Exemple**

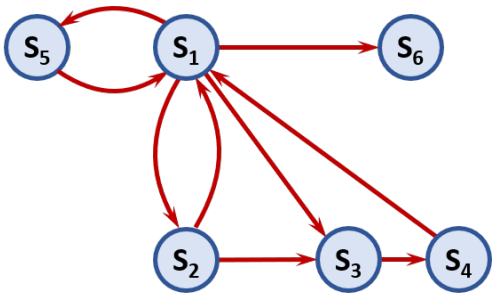



FIGURE 8 – Graphe orienté

- $d_-(S_1) = 3$.
- $d_+(S_1) = 4$.
- $d^\circ(S_1) = 7$.

▪

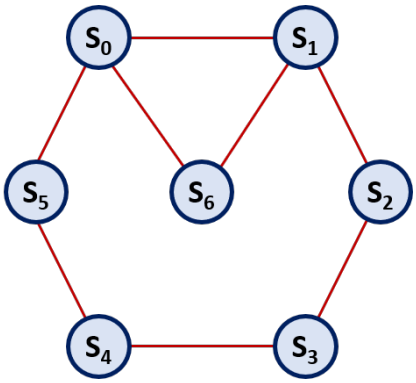
# 3 Implémentation des graphes

## 3.1 Liste d'adjacence

**Définition Liste d'adjacence** Soit un graphe de $n$ sommets d'indices $i \in [\![0, n-1]\!]$. Pour implémenter le graphe, on utilise une liste `G` de taille $n$ pour laquelle, `G[i]` est la liste des voisins de `i`.

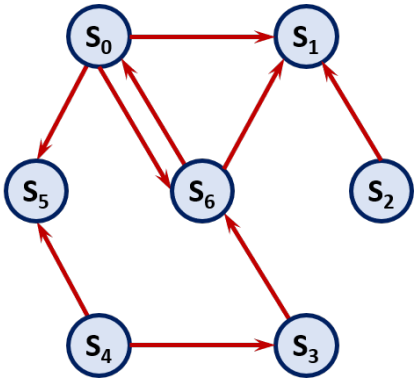
R Cette implémentation est plutôt réservée au graphes « creux », c'est-à-dire ayant peu d'arêtes.

▪ **Exemple**




Dans ce cas $S_0$ est voisin de $S_1$, $S_5$ et $S_6$; donc `G[0]=[1,5,6]`. $S_3$ est voisin de $S_2$ et $S_4$; donc `G[3]=[2,4]`.

```
G = [[1,5,6],[0,2,6],[1,3],[2,4],[3,5],
     [4,0],[1,0]]
```




Dans ca cas, le graphe est orienté. La liste d'adjacence contient la liste des successeurs. Ainsi, les successeurs de $S_0$ sont $S_1$, $S_5$ et $S_6$; donc `G[0]=[1,5,6]`. $S_1$ n'a pas de successeur donc `G[1]=[]`.

```
G = [[1,5,6],[],[1],[6],[3,5],[],[0,1]]
```

Dans la même idée, il est aussi possible d'utiliser des dictionnaires d'adjacence dans lequel les clés sont les sommets, et les valeurs sont des listes de voisins ou de successeurs.

```
# Graphe non orienté
G = {"S0":["S1","S5","S6"],"S1":["S0","S2","S6"],"S2":["S1","S3"],"S3":["S2","S4"],"S4":["↙
    S3","S5"],"S5":["S4","S0"],"S6":["S1","S0"]}
# Graphe orienté
G = {"S0":["S1","S5","S6"],"S1":[],"S2":["S1"],"S3":["S6"],"S4":["S3","S5"],"S5":[],"S6":["↙
    S1","S0"]}
```

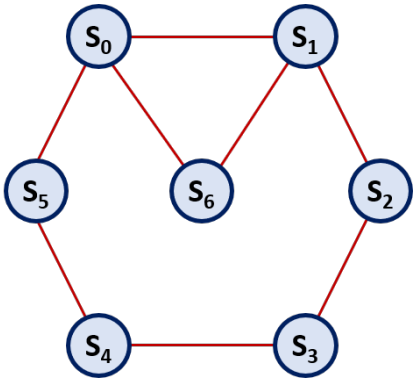
▪

## 3.2 Matrice d'adjacence

**Définition Matrice d'adjacence** Soit un graphe de $n$ sommets d'indices $i \in [\![0, n-1]\!]$ et $E$ l'ensemble des arêtes (on notera $G = ([\![0, n-1]\!], E)$. Pour implémenter le graphe, on utilise la matrice d'adjacence carrée de taille $n$, $\mathscr{M}_n$ `G` de taille $n$ pour laquelle, $m_{i,j} = \begin{cases} \texttt{True} & \text{si } \{i,j\} \in E \\ \texttt{False} & \text{sinon} \end{cases}$ avec $i, j \in [\![0, n-1]\!]$.

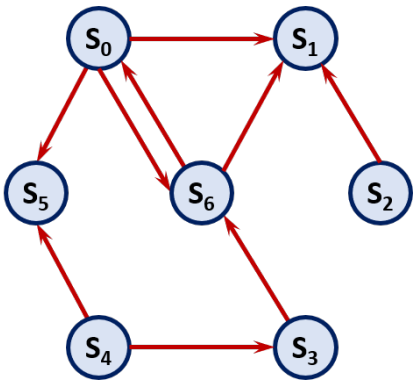
R Cette implémentation est plutôt réservée au graphes « denses » ayant « beaucoup » d'arêtes.

▪ **Exemple**




On a dans ce cas

$M =$
$$\begin{pmatrix}
\texttt{False} & \texttt{True} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{True} & \texttt{True} \\
\texttt{True} & \texttt{False} & \texttt{True} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{True} \\
\texttt{False} & \texttt{True} & \texttt{False} & \texttt{True} & \texttt{False} & \texttt{False} & \texttt{False} \\
\texttt{False} & \texttt{False} & \texttt{True} & \texttt{False} & \texttt{True} & \texttt{False} & \texttt{False} \\
\texttt{False} & \texttt{False} & \texttt{False} & \texttt{True} & \texttt{False} & \texttt{True} & \texttt{False} \\
\texttt{True} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{True} & \texttt{False} & \texttt{False} \\
\texttt{True} & \texttt{True} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{False}
\end{pmatrix}$$
ou
$$M = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$




Dans ce cas, le graphe est orienté. On a On a dans ce cas

$M =$
$$\begin{pmatrix}
\texttt{False} & \texttt{True} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{True} & \texttt{True} \\
\texttt{False} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{False} \\
\texttt{False} & \texttt{True} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{False} \\
\texttt{False} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{True} \\
\texttt{False} & \texttt{False} & \texttt{False} & \texttt{True} & \texttt{False} & \texttt{True} & \texttt{False} \\
\texttt{False} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{False} \\
\texttt{True} & \texttt{True} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{False} & \texttt{False}
\end{pmatrix}$$
ou $M =$
$$\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

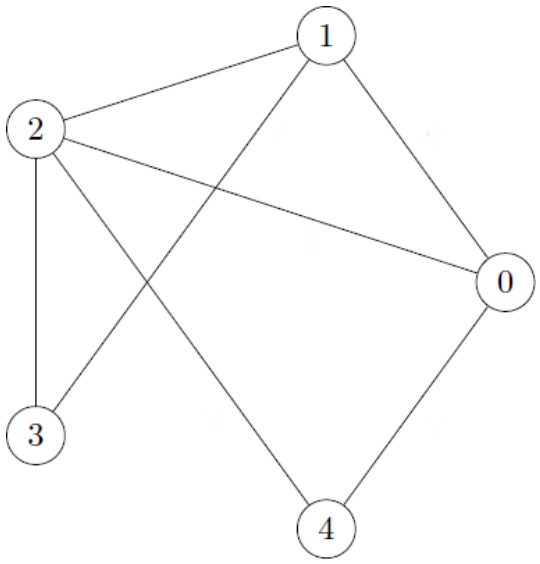
■

R

- Dans le cas d'un graphe non orienté, la matrice est symétrique.
- Si on avait un bouclage sur un sommet, il y aurait des valeurs non nulles sur la diagonale.

## Exercice 1 – Implémentation des graphes par une liste d'adjacence

On considère le graphe `G` suivant, où le nombre situé sur l'arête joignant deux sommets est leur distance, supposée entière.




Pour implémenter le graphe, on utilise une liste `Gl` qui a pour taille le nombre de sommets. Chaque élément `Gl[i]` est la liste des voisins de `i`.

Dans ce cas, `Gl[0]=[1,2,4]` car Les sommets 1, 2 et 4 sont des voisins de 0.

**Question 1** *Construire la liste d'adjacence* `Gl` *en utilisant la méthode énoncée ci-dessus.*

**Question 2** *Écrire une fonction* `voisins_l(G:list, i:int) -> list`*, d'argument la liste d'adjacence* `G` *et un sommet* $i$*, renvoyant la liste des voisins du sommet* $i$*.*

**Question 3** *Écrire une fonction* `arretes_l(G:list) -> list`*, renvoyant la liste des arêtes. Les arêtes seront constitués de couples de sommets (l'arête entre les sommets 0 et 1 sera donnée par* `(0,1)`*.*

Les instructions suivantes permettent de tracer un graphe.

```
import networkx as nx

def plot_graphe_l(G):
    Gx = nx.Graph()
    edges = arretes_l(G)
    Gx.add_edges_from(edges)
    nx.draw(Gx,with_labels = True)
    plt.show()
plot_graphe(M)
```

**Question 4** *Écrire et tester la fonction* `plot_graphe_l(G)`*.*

**Question 5** *Écrire une fonction* `degre_l(G:list, i:int) -> int`*, d'argument un sommet* $i$*, renvoyant le nombre des voisins du sommet* $i$*, c'est-à-dire le nombre d'arêtes issues de* $i$*.*

**Question 6** *Écrire la fonction* `ajout_sommet_l(G:list, L:list) -> None` *permettant d'ajouter un sommet au graphe.* `L` *désigne la liste des sommets auxquels le nouveau sommet est relié.* `ajout_sommet` *agit avec effet de bord sur* `G`*.*

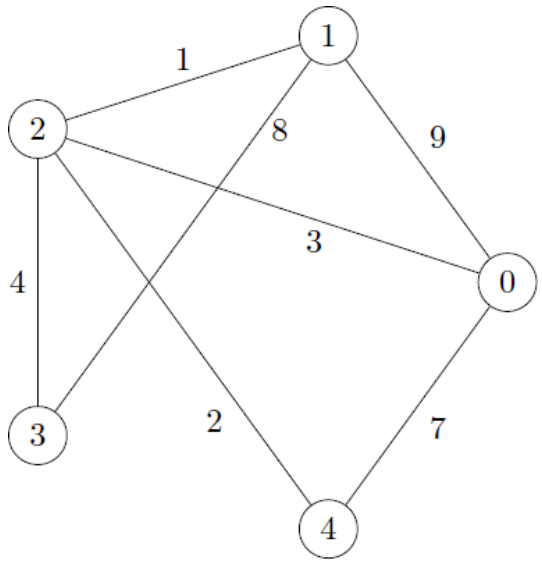
**Question 7** *Écrire la fonction* `supprime_sommet_l(G:list, i: int) -> None` *permettant de supprimer le sommet* $i$ *du graphe.*

**Question 8** *Écrire la fonction* `from_list_to_matrix(G:list -> list` *permettant de convertir un graphe implémenté sous forme de liste d'adjacence en matrice d'adjacence.*

**Question 9** *Écrire la fonction* `from_matrix_to_list -> list` *permettant de convertir un graphe implémenté sous forme de matrice d'adjacence en liste d'adjacence.*

## Exercice 2 – Implémentation des graphes par une matrice d'adjacence

On considère le graphe `G` suivant, où le nombre situé sur l'arête joignant deux sommets est leur distance, supposée entière.




**Question 10** *Construire la matrice* $(G_{ij})_{0\leq i,j\leq 4}$*, matrice de distances du graphe* `G`*, définie par : « pour tous les indices* $i$*,* $j$*,* $G_{ij}$ *représente la distance entre les sommets* $i$ *et* $j$*, ou encore la longueur de l'arête reliant les sommets* $i$ *et* $j$ *». Cette matrice sera implémentée sous forme d'une liste de listes. (Chaque « sous-liste » représentant une ligne de la matrice d'adjacence. On convient que, lorsque les sommets ne sont pas reliés, cette distance vaut* $-1$*. La distance du sommet* $i$ *à lui-même est égale à 0.*

**Question 11** *Écrire une fonction* `voisins(G:list, i:int) -> list`*, d'argument la matrice d'adjacence* `G` *et un sommet* $i$*, renvoyant la liste des voisins du sommet* $i$*.*

**Question 12** *Écrire une fonction* `aretes(G:list) -> list`, *renvoyant la liste des arêtes. Les arêtes seront constitués de couples de sommets (l'arête entre les sommets 0 et 1 sera donnée par* `(0,1)`*).*

Les instructions suivantes permettent de tracer un graphe.

```
import networkx as nx
import matplotlib.pyplot as plt

def plot_graphe(G):
    Gx = nx.Graph()
    edges = arretes(G)
    Gx.add_edges_from(edges)
    nx.draw(Gx,with_labels = True)
    plt.show()
plot_graphe(M)
```

**Question 13** *Écrire et tester la fonction* `plot_graphe(G)`.

**Question 14** *Écrire une fonction* `degre(G:list, i:int) -> int`, *d'argument un sommet* $i$, *renvoyant le nombre des voisins du sommet* $i$, *c'est-à-dire le nombre d'arêtes issues de* $i$.

**Question 15** *Écrire une fonction* `longueur(G:list,L:list) -> int`, *d'argument une liste* `L` *de sommets de* `G`, *renvoyant la longueur du trajet d'écrit par cette liste* `L`, *c'est-à-dire la somme des longueurs des arêtes empruntées. Si le trajet n'est pas possible, la fonction renverra* $-1$.

**Question 16** *Écrire la fonction* `ajout_sommet(G:list, L:list, poids : list) -> None` *permettant d'ajouter un sommet au graphe.* `L` *désigne la liste des sommets (triés dans l'ordre croissant) auxquels le nouveau sommet est relié,* `poids` *la liste des poids respectifs.* `ajout_sommet` *agit avec effet de bord sur* `G`.

**Question 17** *Écrire la fonction* `supprime_sommet(G:list, i: int) -> None` *permettant de supprimer le sommet* $i$ *du graphe.*