Table des matières

1 Définitions

Dans tout ce chapitre, $G = (S, A)$ désigne un graphe, où :

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

Pour simplifier, on supposera dans la suite que $S = [\![1, n]\!]$ où $n \in \mathbb{N}^*$. Pour les fonctions **Python**, il sera toutefois plus commode de commencer la numérotation à 0, ce que l'on fera.

Definition 1. On dit que G possède une arête *allant de i vers j* si $(i, j) \in A$. On dira aussi dans ce cas que j est un *voisin* de i.

Definition 2. On dit que le graphe G est *non-orienté* si

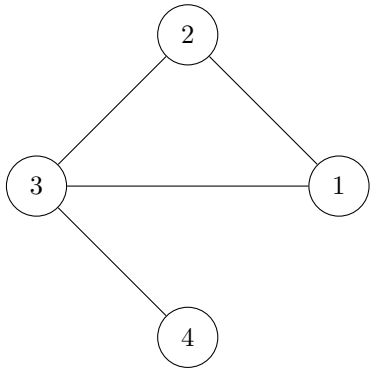
$$\forall (i, j) \in S^2, \Big((i, j) \in A \iff (j, i) \in A \Big)$$

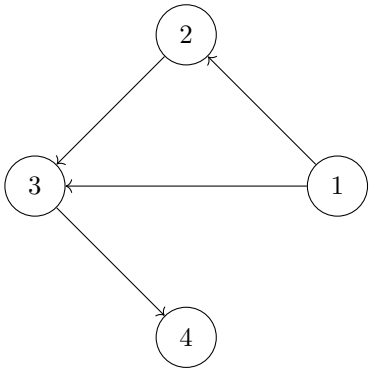
Dans le cas contraire, on dit que le graphe G est orienté.

Remarque 1.

- Dans un graphe non orienté, on représente les arêtes par des traits.
- Dans un graphe orienté, on représente les arêtes par des flèches.

Exemple 1.

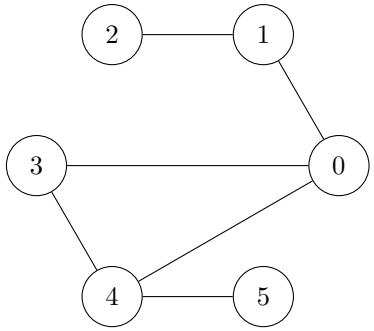




Definition 3. On appelle *matrice d'adjacence de G* la matrice $M = (m_{i,j})_{\substack{1 \leqslant i \leqslant n \\ 1 \leqslant j \leqslant n}} \in \mathscr{M}_n(\mathbb{R})$ définie par :

$$\forall (i, j) \in [\![1, n]\!]^2, \ m_{i,j} = \begin{cases} 1 & \text{si une arête va de } i \text{ vers } j \\ 0 & \text{sinon} \end{cases} = \begin{cases} 1 & \text{si } (i, j) \in A \\ 0 & \text{sinon} \end{cases}$$

Exemple 2.



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

Dans cet exemple, on a numéroté les sommets à partir de 0 en faisant un décalage d'indice. On remarque que la matrice d'adjacence est symétrique. Il s'agit d'une propriété caractéristique des graphes _____.

Exercice 1 : Décrire ce que fait la fonction **Python** suivante, sachant qu'elle prend en argument la matrice d'adjacence d'un graphe.

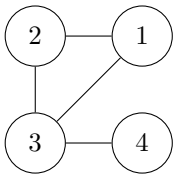
```
1 def mystere(M):
2     return not np.all(M == np.transpose(M))
```

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

Definition 4. Soit $i \in S$. On appelle *liste d'adjacence* du sommet i la liste des voisins de i.

Exemple 3.




```
L1 = [2,3]
L2 = [1,3]
L3 = [1,2,4]
L4 = [3]
```

Remarque 2. Un graphe est tout autant caractérisé par sa matrice d'adjacence que par la liste de ses listes d'adjacence. Ainsi, on peut calculer les listes d'adjacence à partir de la matrice d'adjacence et vice-versa.

Exercice 3 : Soit G un graphe. On note M sa matrice d'adjacence et L la liste de ses listes d'adjacence.

1. Compléter la fonction **Python** suivante pour qu'elle prenne en entrée la matrice d'adjacence de G et renvoie la liste de ses listes d'adjacence.

```
1  def matAdjtolistAdj(M):
2      n = len(M)
3      L = [[] for k in range(n)]
4      for i in range(n):
5          for j in range(n):
6              if M[i,j] == _____:
7                  ____________________
8      return L
```

2. Compléter la fonction **Python** suivante pour qu'elle prenne en entrée la liste des listes d'adjacence de G et renvoie sa matrice d'adjacence.

```
1  def listAdjtomatAdj(L):
2      n = len(L)
3      M = np.zeros([n,n])
4      for i in range(n):
5          for _____________:
6              M[i,j] = _____
7      return M
```

3. On suppose dans cette question que le graphe G est non-orienté. On souhaite alors améliorer la fonction de la première question, en faisant moins de calculs. Compléter la fonction **Python** suivante pour qu'elle prenne en entrée la matrice d'adjacence de G et renvoie la liste de ses listes d'adjacence.

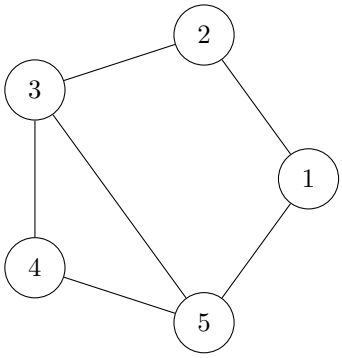
```
1  def matAdjtolistAdjNotOriented(M):
2      n = len(M)
3      L = [[] for k in range(n)]
4      for i in range(n):
5          for j in range(_________):
6              if M[i,j] == _____:
7                  ____________________
8                  ____________________
9      return L
```

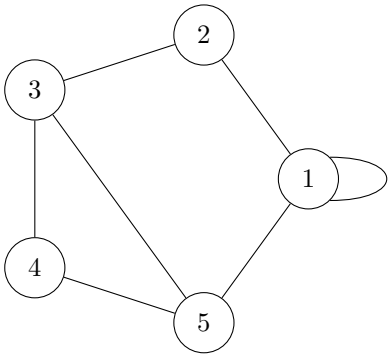
Definition 5. On dit que le graphe G est *sans boucle* si :

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

Autrement dit, G est sans boucle si aucune arête ne relie un sommet à lui-même.

Exemple 4.





Exercice 4 : Écrire une fonction **Python** `sansBoucle(M)` prenant en entrée la matrice d'adjacence d'un graphe, renvoyant `True` si ce graphe est sans boucle et `False` sinon.

Definition 6. Soit $i \in S$. On appelle degré de i et on note $\deg(i)$ le nombre de voisins que possède le sommet i. Autrement dit :

$$\deg(i) = \text{Card}\Big(\{j \in S \mid (i,j) \in A\}\Big)$$

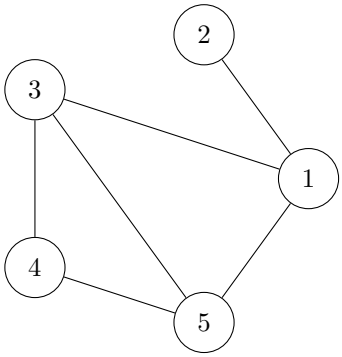
On dit que i est un sommet *isolé* si $\deg(i) = 0$ (autrement dit, si i n'a pas de voisins).

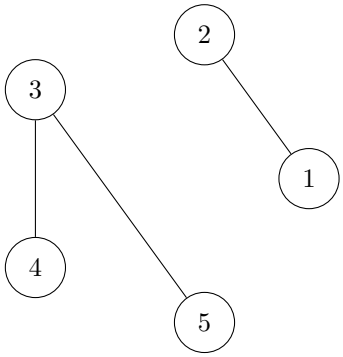
Exercice 5 :

1. Écrire une fonction **Python** `degfromMat(i,M)` prenant en entrée un sommet i et la matrice d'adjacence M d'un graphe et renvoyant le degré de i.
2. Écrire une fonction **Python** `degfromList(i,L)` prenant en entrée un sommet i et la liste des listes d'adjacence L d'un graphe et renvoyant le degré de i.
3. Écrire une fonction **Python** `Listdeg(L)` prenant en entrée la liste des listes d'adjacence L d'un graphe et renvoyant la liste des degrés des sommets du graphe.
4. Écrire une fonction **Python** `degMax(M)` prenant en entrée la matrice d'adjacence d'un graphe M et renvoyant le maximum des degrés des sommets du graphe.

Definition 7. On dit que G est *connexe* si deux sommets quelconques sont toujours connectés par au moins un chemin.

Exemple 5.





2 Résultats de cours

2.1 Formule d'Euler

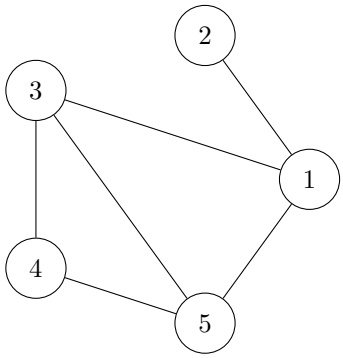
Théoreme 1 (Formule d'Euler)**.** *Soit $G = (S, A)$ un graphe fini non orienté et sans boucle. Alors*

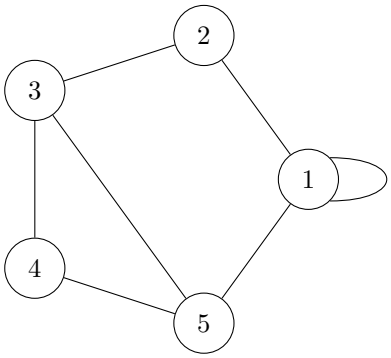
$$2\,\mathrm{Card}(A) = \sum_{x \in S} \deg(x)$$

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

Remarque 3. 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 la formule d'Euler, il y a un nombre pair de personnes qui serreront la main à un nombre impair de personnes. Pour cette raison, cette formule est également appelée « Lemme des poignées de main ».

Exemple 6.





Exercice 6 : Soit $G = (S, A)$ un graphe fini non orienté et sans boucle. On note n ($n \in \mathbb{N}^*$) le nombre de sommets de G et on note p le nombre de sommets isolés de G. Montrer qu'il y a au plus $\dfrac{(n-p)(n-p-1)}{2}$ arêtes dans G. Proposer un graphe pour lequel c'est un cas d'égalité.

2.2 Nombre de chemins

Théoreme 2. *On note M la matrice d'adjacence du graphe G.*

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

Exercice 7 : Compléter la fonction **Python** suivante pour qu'elle prenne en entrée la matrice d'adjacence M d'un graphe et pour qu'elle renvoie le booléen `True` si ce graphe est connexe et `False` sinon.

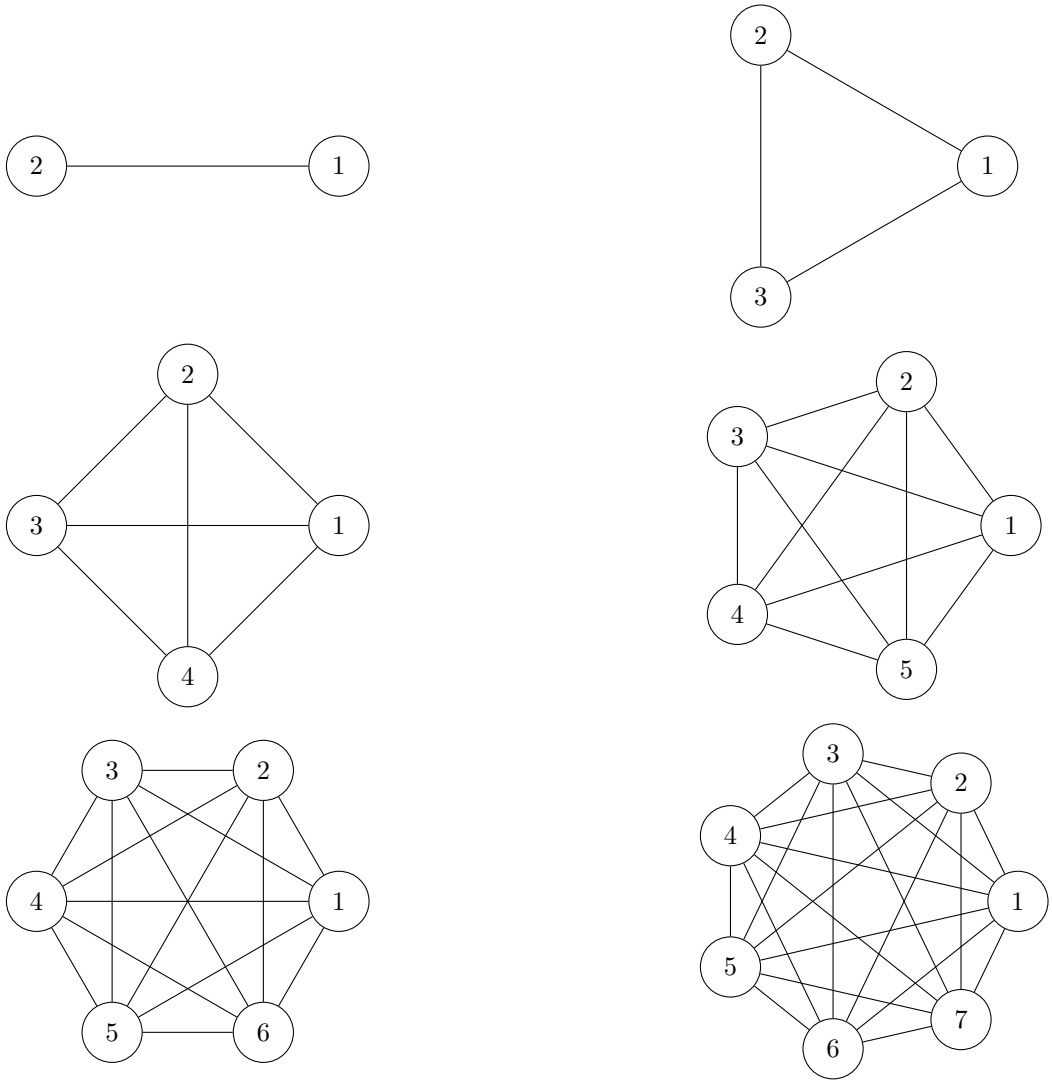
```
1  def graphConnexe(M):
2      n = len(M)
3      S = np.zeros([n,n])
4      puissance = _____________
5      for k in range(n):
6          S = S + puissance
7          ____________________________________________
8      return ___________________
```

3 Quelques classes de graphes spéciaux

3.1 Graphes complets

Definition 8. On dit que le graphe G est complet si chacun de ses sommets est relié à tous les autres sommets par une arête.

Exemple 7.



Exercice 8 : Soit G un graphe complet à n sommets. Quel est le degré commun des sommets de G ? Combien G possède-t-il d'arêtes ?

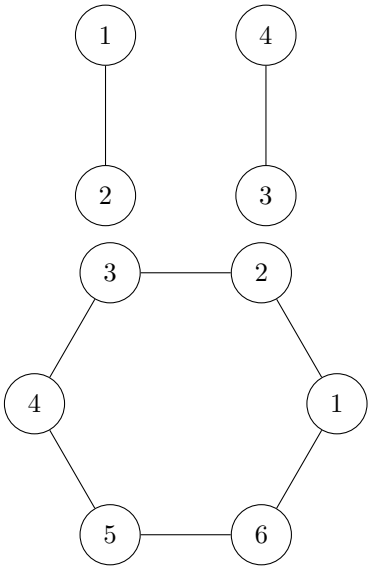
Exercice 9 : Écrire une fonction **Python** `completeGraphTest(L)` prenant en entrée la liste des listes d'adjacence d'un graphe, renvoyant `True` si celui-ci est complet et `False` sinon.

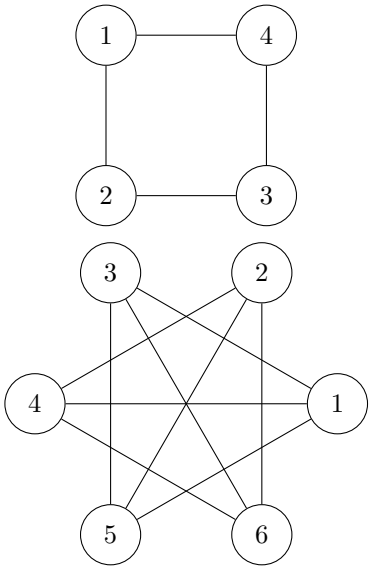
Exercice 10 : Écrire une fonction **Python** `completeGraphMat(n)` prenant en entrée un entier $n \geq 2$ et renvoyant la matrice d'adjacence du graphe complet à n sommets.

3.2 Graphes réguliers

Definition 9. Soit G un graphe non orienté et sans boucle à n sommets ($n \geqslant 2$). On dit que G est *régulier* si tous ses sommets ont le même degré. Plus précisément, on dit que G est *d-régulier* si tous ses sommets ont un degré égal à d (où $d \in [\![0, n-1]\!]$ est appelé le *degré de régularité* de G).

Exemple 8.





Exercice 11 : Soit n un entier supérieur ou égal à 2. Soit $d \in [\![0, n-1]\!]$. Soit G un graphe à n sommets et d-régulier. On note $A = (a_{i,j})_{\substack{1 \leqslant i \leqslant n \\ 1 \leqslant j \leqslant n}}$ la matrice d'adjacence du graphe G.

1. Soit $i \in [\![1, n]\!]$. Que vaut la somme $\sum_{j=1}^{n} a_{i,j}$?

   En déduire une valeur propre de A. On explicitera un vecteur propre associé à cette valeur propre.

2. Soit λ une valeur propre de A. Soit $X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathscr{M}_{n,1}(\mathbb{R}) \setminus \{0_{\mathscr{M}_{n,1}(\mathbb{R})}\}$ tel que :

   $$AX = \lambda X \quad (*)$$

   On fixe $i_0 \in [\![1, n]\!]$ tel que :

   $$|x_{i_0}| = \max_{i \in [\![1,n]\!]} |x_i|$$

   (a) En utilisant $(*)$, montrer que : $|\lambda| \, |x_{i_0}| \leqslant d \, |x_{i_0}|$.

   (b) Conclure que : $|\lambda| \leqslant d$.

3. Quelle est la plus grande valeur propre de A ?

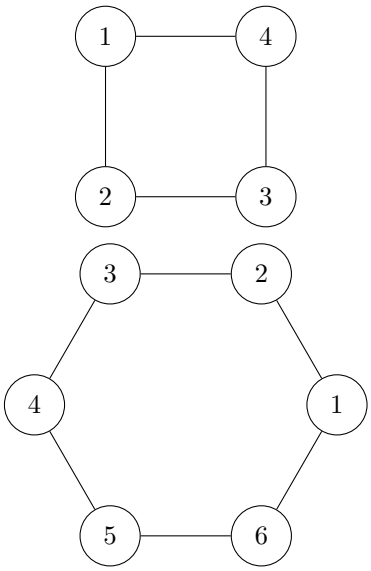
Exercice 12 : Compléter la fonction **Python** suivante pour qu'elle prenne en entrée la liste des listes d'adjacence d'un graphe et qu'elle renvoie le couple `(True, d)` si le graphe est d-régulier pour un certain entier d et `False` sinon.

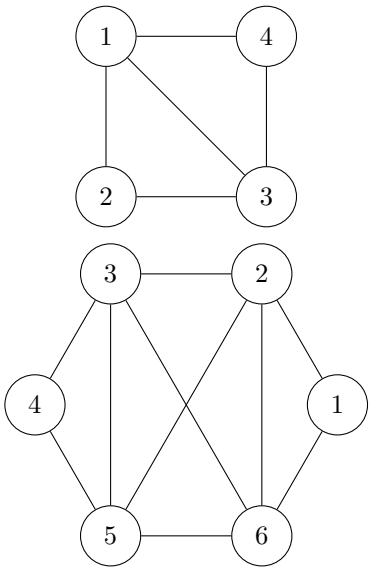
```
1  def regularGraphTest(L):
2      n = len(L)
3      deg = _____________
4      for k in _____________:
5          if deg != _____________:
6              return _________
7      return _____________
```

3.3 Graphes Eulériens

Definition 10. On dit que G est *Eulérien* si il existe un chemin cyclique dans G (pour lequel le sommet d'arrivée est égal au sommet de départ) passant une et une seule fois par chacune des arêtes de G.

Exemple 9.





Exercice 13 : Soit G un graphe non orienté et sans boucle. On admet que G est Eulérien si et seulement si tous ses sommets ont un degré pair.

En déduire une fonction **Python** `graphEulerTest(L)` prenant en entrée la liste des listes d'adjacence d'un graphe, renvoyant `True` si celui-ci est Eulérien et `False` sinon.

3.4 Graphes aléatoires

Soit n un entier supérieur ou égal à 2 et p un réel appartenant à $]0,1[$.

Pour générer des graphes non orientés de manière aléatoire, on se donne :

- $S = [\![0, n-1]\!]$, les sommets du graphe ;
- pour toute paire de sommets $\{u, v\}$ avec $u < v$, $T_{u,v}$ une variable aléatoire de Bernoulli de paramètre p.
  Les variables aléatoires $T_{u,v}$ pour $\{u, v\}$ décrivant les paires de sommets avec $u < v$, sont supposées indépendantes ;
- les arêtes d'un graphe G ainsi généré sont les paires $\{u, v\}$ telles que $T_{u,v} = 1$ si $u < v$ ou $T_{v,u} = 1$ si $v < u$.

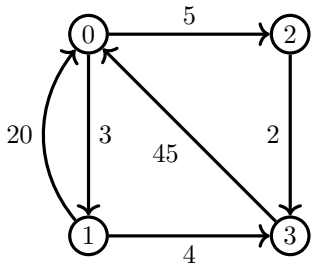
Exercice 14 : Écrire une fonction **Python** `graphe(n,p)` prenant en entrée un entier $n \geq 2$ et un réel $p \in]0,1[$ et renvoyant la liste des listes d'adjacence d'un graphe généré aléatoirement selon le procédé décrit en préambule.

Exercice 15 : Calculer le nombre moyen de sommets isolés dans un graphe aléatoire à n sommets de paramètre p.

4 Graphes pondérés et algorithme de Dijkstra

Definition 11. On dit que G est un graphe *pondéré* si à toute arête de G est associée un nombre réel strictement positif (appelé *poids* de l'arête).

Exemple 10.



Exemple 11. Exemple de graphe pondéré et non orienté. Implémentation de l'algorithme de Dijkstra sur ce graphe.

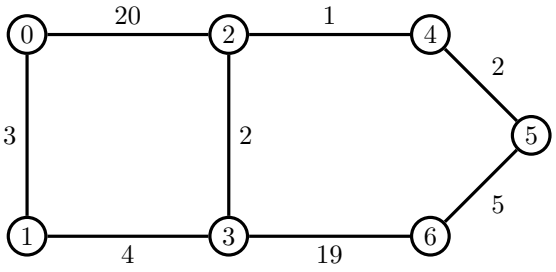


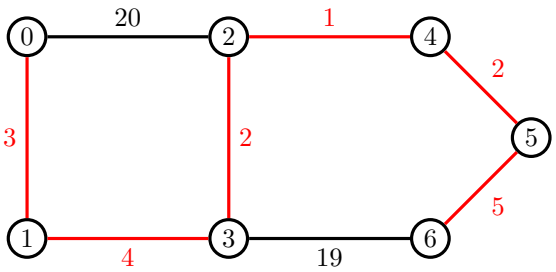
Tableau :

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
—	—	—	—	—	—	$\mathbf{17_5}$

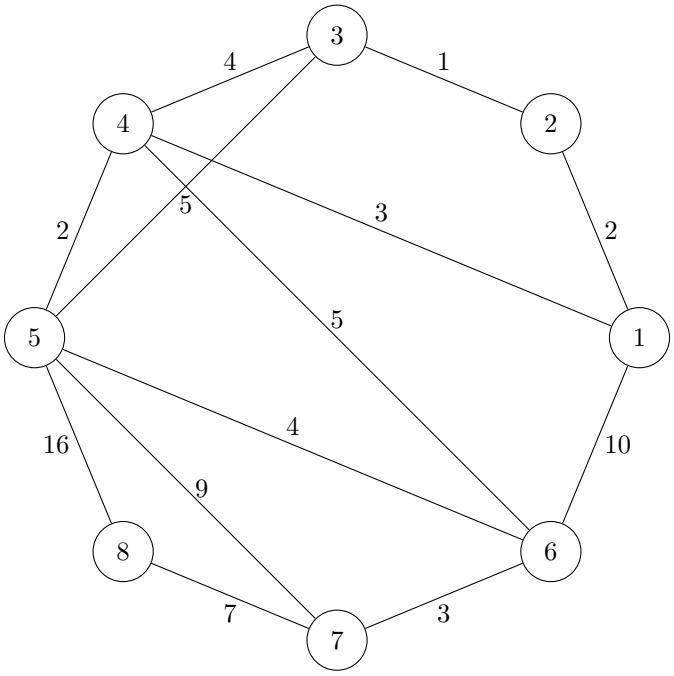
Distance 0-6 = 17

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

Visualisation du chemin le plus court :



Exemple 12.



5 D'autres matrices associées aux graphes

Exercice 16 : Soit G un graphe connexe non orienté sans boucle à n sommets, notés $1, \dots, n$. On associe à ce graphe :

- sa *matrice des degrés* $D \in \mathscr{M}_n(\mathbb{R})$ définie par :

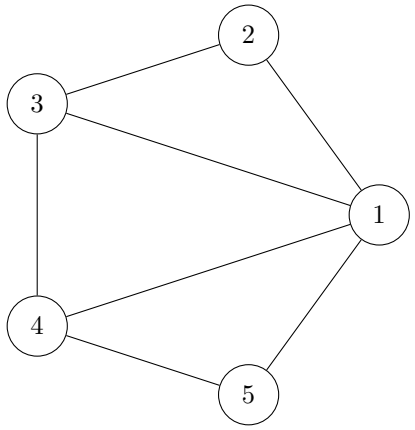
$$\forall (i,j) \in [\![1,n]\!]^2,\ [D]_{i,j} = \begin{cases} \deg(i) & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

- sa *matrice laplacienne* $L \in \mathscr{M}_n(\mathbb{R})$ définie par :

$$L = D - M$$

où M désigne la matrice d'adjacence du graphe G.

1. Donner la matrice des degrés puis la matrice laplacienne du graphe G suivant :



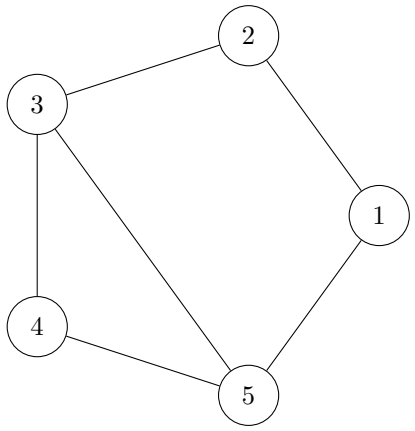
2. (a) Écrire une fonction **Python** `adj2matriceDeg(M)` prenant en entrée la matrice d'adjacence d'un graphe connexe non orienté sans boucle et renvoyant sa matrice des degrés.
   (b) En déduire une fonction **Python** `adj2matriceLaplace(M)` renvoyant cette fois-ci la matrice laplacienne de ce même graphe.
3. La matrice $D = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ est-elle la matrice des degrés d'un graphe ?
4. Écrire une fonction **Python** `laplace2Adj(L)` prenant en entrée la matrice laplacienne d'un graphe connexe non orienté sans boucle et renvoyant sa matrice d'adjacence.

Exercice 17 : Soit G un graphe connexe non orienté sans boucle à n sommets, notés $1, \dots, n$. On associe à ce graphe sa *matrice des distances* $D \in \mathscr{M}_n(\mathbb{R})$ définie par :

$$\forall (i,j) \in [\![1,n]\!]^2,\ [D]_{i,j} = d(i,j)$$

où $d(i,j)$ est la distance séparant les deux sommets i et j dans le graphe G.

1. Donner la matrice des distances du graphe G suivant :



2. Écrire une fonction **Python** `adj2matriceDistance(M)` prenant en entrée la matrice d'adjacence d'un graphe connexe non orienté sans boucle et renvoyant sa matrice des distances.
3. La matrice $D = \begin{pmatrix} 0 & 1 & 2 & 3 & 2 & 1 \\ 1 & 0 & 1 & 2 & 2 & 1 \\ 2 & 1 & 0 & 1 & 2 & 2 \\ 3 & 2 & 1 & 0 & 1 & 2 \\ 2 & 2 & 2 & 1 & 0 & 1 \\ 1 & 1 & 2 & 2 & 1 & 0 \end{pmatrix}$ est-elle la matrice des distances d'un graphe ?
   Si oui, le représenter.
4. Écrire une fonction **Python** `distance2Adj(D)` prenant en entrée la matrice des distances d'un graphe connexe non orienté sans boucle et renvoyant sa matrice d'adjacence.