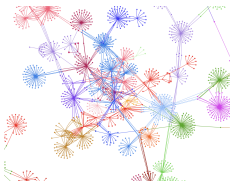
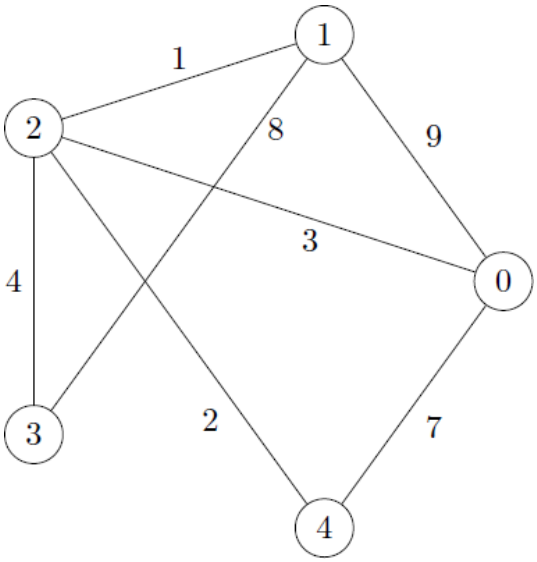
TP01



Découverte de la représentation des graphes

Exercice 1 – 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 1 *Construire la matrice* $\left(G_{ij}\right)_{0\leq i,j\leq 4}$, *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 2 *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 3 *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 4 *Écrire et tester la fonction* `plot_graphe(G)`.

Question 5 *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`.

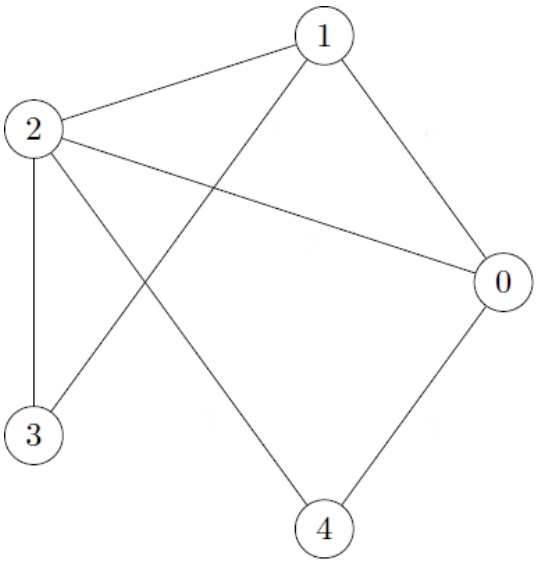
Question 6 *Écrire une fonction* `longueur(G:list,L:list) -> int`, *d'argument une liste* `L` *de sommets de* `G`, *renvoyant la longueur du trajet d'écrit par cette liste* `L`, *c'est-à-dire la somme des longueurs des arêtes empruntées. Si le trajet n'est pas possible, la fonction renverra* -1.

Question 7 *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 8 *Écrire la fonction* `supprime_sommet(G:list, i: int) -> None` *permettant de supprimer le sommet* `i` *du graphe.*

Exercice 2 – 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 9 *Construire la liste d'adjacence* `Gl` *en utilisant la méthode énoncée ci-dessus.*

Question 10 *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 11 *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 12 *Écrire et tester la fonction* `plot_graphe_l(G)`.

Question 13 *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 14 *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 15 *Écrire la fonction* `supprime_sommet_l(G:list, i: int) -> None` *permettant de supprimer le sommet* i *du graphe.*

Question 16 *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 17 *Écrire la fonction* `from_matrix_to_listmatrix(G:list) -> list` *permettant de convertir un graphe implémenté sous forme de matrice d'adjacence en liste d'adjacence.*