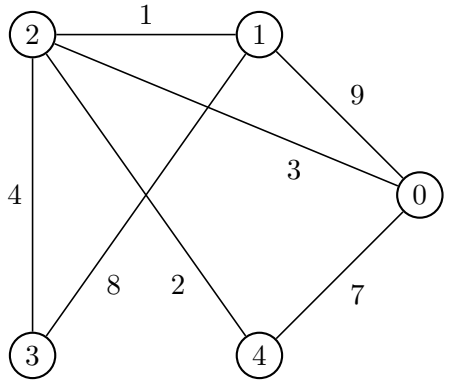
TP : Graphes

Toutes les fonctions doivent etre testées. On testera l'algorithme de Dijkstra sur le graphe du premier exercice puis sur un graphe orienté (librement choisi).

Exercice 1 (Exercices types)
On considère le graphe pondéré G suivant, où le nombre situé sur l'arête joignant deux sommets est leur distance, supposée entière :



1) Construire la matrice $M = (m_{ij})_{0 \leqslant i,j \leqslant 4}$ matrice de distances du graphe G, définie par : pour tous les indices i, j, m_{ij} représente le poids de l'arête reliant les sommets i et j.
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.

2) Écrire une suite d'instructions permettant de dresser à partir de la matrice M la liste des voisins du sommet 4.

3) Écrire une fonction `voisins`, d'arguments un sommet s et une matrice M, renvoyant la liste des voisins du sommet s.

4) Écrire une fonction `longueur`, d'argument une liste `L` de sommets de G et une matrice M, renvoyant la longueur du trajet décrit par cette liste `L`, c'est-à-dire la somme des poids des arêtes empruntées. Si le trajet n'est pas possible, la fonction renverra -1.

Exercice 2 (Algorithme de Dijkstra)
Soit G un graphe pondéré à poids positifs. On suppose que le graphe est connexe, c'est-à-dire que pour tout couple de sommet du graphe, il existe un chemin qui les relient.

1) Construire une fonction qui prend en arguments une matrice d'adjacence M (comme à l'exercice précédent) représentant un graphe G et un sommet s représenté par un entier, puis retourne la liste des couples (poids, sommet) obtenue par l'algorithme de Dijkstra. On pourra, en cas de besoin, utiliser des fonctions annexes.

2) Écrire une fonction `itineraire` qui prends deux sommets i et j, et un matrice M en arguments et retourne le plus court chemin de i à j.

En fait, il n'est pas nécessaire de supposer le graphe connexe, l'algorithme s'arrête de lui-même si le graphe n'est pas connexe en retournant les sommets inaccessible avec une distance infinie

Exercice 3 (Graphe aléatoire et complexité de l'algorithme de Dijkstra)
Le but de cet exercice est de mesurer statistiquement la complexité de l'algorithme de Dijkstra. Dans un premier temps, on cherche à créer un graphe connexe raisonnablement aléatoire.

1) Pourquoi ne pas simplement tire au hasard une matrice d'adjacence d'un graphe (pondéré à poids positifs) ?

2) Créer une fonction qui rajoute un sommet à graphe déjà construit, et rajoute (ou pas), avec une probabilité p, une arête (pondérée à poids positif) vers chacun des sommets déjà créé.
À la fin de la fonction, on testera qu'il y a bien eu au moins une arête de créée, et donc que le graphe est connexe. Sinon, on retourne le graphe de départ.

3) Créer une fonction `graphe_alea` qui retourne un graphe aléatoire ayant v sommets, de poids au plus N. Elle prendra comme arguments le nombre v de sommets, la probabilité p de créer une arête et le poids maximal N (par défaut $N = 100$).

4) Écrire une fonction `test_Dijkstra`, dont on choisira habilement les arguments, qui teste le temps que prend l'algorithme de Dijkstra en fonction du nombre de sommets et du nombre d'arêtes. On fera des moyennes pour lisser les statistiques (cf TP précédent). Cette fonction retournera un tableau à deux dimensions.

5) Comme le stockage des sommets en cours de traitement n'a pas été optimisé, la recherche de celui de poids minimal est coûteuse, et la complexité n'est pas optimal. Montrer (graphiquement) qu'elle est en $O(v^2)$.