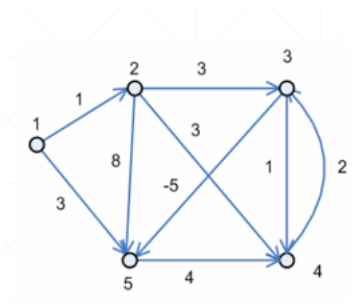
TP noté

Soit un graphe orienté G = (V, E) qui est constitué

- d'un ensemble fini V = {v_1, v_2, ..., v_n} de n éléments, appelés sommets

- d'un ensemble fini E = {e_1, e_2, ..., e_m} appelés arcs avec e_k = (v_i, v_j) est une paire ordonné de deux sommets : v_i est l'extrémité initiale, v_j est l'extrémité finale, k de 1 à m ; v_i, v_j appartiennent à V



1. Implémentez des classes nécessaires pour pouvoir instancier un graphe en demandant la saisie des sommets et des arcs auprès de l'utilisateur
   Exemple : Il y a combien de sommets ? 5
   Nom du sommet 1 : 1
   Nom du sommet 2 : 2
   …
   Il y a combien d'arcs ? 9
   Arc 1 :
   L'extrémité initiale est le sommet : 1
   L'extrémité finale est le sommet : 2
   Sa valeur est : 1
   Arc 2 :
   ….

2. Ecrivez une fonction **constructFromFile** qui permet de lire un fichier graphe.txt et de construire une instance de graphe

   Format du fichier graphe.txt est comme suit :

   {A, B, C, D, E, F}
   A-B-3
   B-D-4
   E-F-1
   A-E-3
   C-F-5

3. Le degré d'un sommet dans un graphe orienté est calculé par la somme de son degré intérieur et extérieur.
   Degré intérieur d'un sommet v = le nombre d'arcs ayant *v* comme extrémité initiale

Degré extérieur d'un sommet v = le nombre d'arcs ayant *v* comme extrémité finale
Ecrivez une ou plusieurs méthodes dans la classe Graphe pour calculer le degré d'un sommet.

4. Ecrivez une fonction **plusCourtChemin** (sommet 1, sommet 2) qui sert à calculer le plus court chemin entre deux sommets.
5. Ecrivez une fonction **plusCourtCheminAvecObstacle** (sommet 1, sommet 2, sommet 3) qui sert à calculer le plus court chemin entre le sommet 1 et 2 en évitant le sommet 3
6. Ecrivez une fonction permettant de comparer deux graphes et de retourner les éléments communs entre eux.
   Exemple : soit 2 graphes comme ci-dessous → ses éléments communs sont les parties qui sont en bleus

   Sommets : A, B, C, B
   Arcs : AB, BC, CD, DA avec les valeurs correspondantes 0.18, 0.5, 0.75, 2.6