L'objectif de ce TP est de construire une bibliothèque pour la manipulation de graphes.

Vous pouvez utiliser le langage de votre choix : le but est de vous familiariser avec les algorithmes de graphe, pas d'évaluer votre maîtrise d'un langage donné.

1 Représentation et propriétés de graphes

N'hésitez pas à implémenter des fonctions annexes pour vous faciliter la vie (affichage, existence d'une arête entre deux sommets ...)

1. Pour commencer, implémentez les structures de données permettant de représenter un graphe orienté pondéré. On choisira l'implémentation à base de listes d'adjacence.

2. Implémentez les fonctions `numVertices` et `numEdges` qui retournent respectivement le nombre de sommets et le nombre d'arcs du graphe passé en argument.

3. Implémentez la fonction `isReflexive` qui retourne vrai si et seulement si le graphe est réflexif (c'est-à-dire qu'il existe une boucle $x \to x$ pour tous les sommets x).

4. Implémentez la fonction `isSymmetric` qui retourne vrai si et seulement si le graphe est symétrique (c'est-à-dire qu'il existe une arête $y \to x$ pour toute arête $x \to y$).

5. Implémentez la fonction `isTransitive` qui retourne vrai si et seulement si le graphe est transitif (c'est-à-dire que si $x \to y$ et $y \to z$, alors $x \to z$).

2 Parcours de graphe

1. Implémentez un parcours en profondeur et un parcours en largeur du graphe. Souvenez-vous que le parcours en profondeur s'appuie sur une pile (LIFO) alors que le parcours en largeur s'appuie sur une file (FIFO). Ces deux parcours doivent retourner la liste des sommets visités dans l'ordre de leur rencontre.

2. Implémentez une fonction `diameter` qui retourne le diamètre du graphe. NB: le diamètre d'un graphe est *la plus grande distance entre deux sommets quelconques du graphe.*

3. Implémentez une fonction `shortest` qui retourne le plus court chemin entre deux sommets donnés en argument.