TP 3-4 : implémentation des concepts de connexité et fermeture transitive

Présentation du TP

(a) **Représentation des graphes orientés à l'aide de matrices d'adjacence**
Nous considérons uniquement des graphes orientés $G = (S, A)$ et nous supposons que les sommets sont numérotés de 0 à $n-1$ d'où $S = \{0, \ldots, n-1\}$. La représentation choisie pour les graphes est celle par matrice d'adjacence. Un graphe $G = (S, A)$ est ainsi représenté par une matrice d'entiers, $matrice$, de taille $n \times n$, définie par :

$$matrice[x][y] = \begin{cases} 1 & si\ (x, y) \in A, \\ 0 & sinon. \end{cases}$$

(b) **Implémentation en `Python`**
Les graphes seront représentés par des matrices `numpy` contenant des entiers, pour représenter les matrices sous la forme donnée ci-dessus. On suppose également que le nombre de sommets du graphe est fixé, au moment de sa création, et qu'il ne pourra pas être modifié par la suite. De plus, les sommets seront numérotés de 0 à $n-1$ (correspondant à leurs indices dans la matrice).

Objectif du TP

L'objectif du TP est d'implémenter des fonctions pour calculer les composantes fortement connexes d'un graphe orienté ainsi que le graphe correspondant à sa fermeture transitive :

(a) `cfc_sommet(g, s)` : retourne l'ensemble des sommets du graphe `g`, qui appartiennent à la composante fortement connexe du sommet `s` (on utilise le théorème vu en cours ainsi que les fonctions données pour calculer les ascendants et les descendants d'un sommet) ;

(b) `cfc_graphe(g)` : retourne la liste des composantes fortement connexes du graphe `g` (on utilise la fonction précédente, pour calculer les composantes d'un sommet donné, et on parcourt tous les sommets du graphe, jusqu'à avoir trouvé toutes les composantes fortement connexes) ;

(c) `est_fortement_connexe(g)` : retourne vrai le graphe `g` est fortement connexe (on utilise la fonction précédente pour vérifier cela) ;

(d) `graphe_fermeture_transitive(g)` : retourne un nouveau graphe qui correspond à la fermeture transitive du graphe `g` donné (pour savoir quels arcs ajouter, pour construire la fermeture transitive, on peut utiliser la fonction qui calcule les descendants d'un sommet donné).

D'autres fonctions bonus peuvent être implémentées et concernent la notion de connexité simple (sans prendre en compte l'orientation des arcs du graphe orienté).

Déroulement du TP

Pour chaque cellule de code contenant l'instruction `raise NotImplementedError`, supprimez cette instruction et remplacez-la par le code python correspondant à la fonction à

implémenter. Vérifiez la syntaxe de votre implémentation en exécutant la cellule de code autant de fois que nécessaire. Pour tester votre fonction, vous pouvez utiliser la cellule de code suivante, qui contient des petits tests sur les graphes `graphe1` et `graphe2`.

Vous pouvez ajouter des fonctions supplémentaires, si vous le souhaitez. Vous pouvez également réutiliser les fonctions déjà implémentées.