TP 5-6 : implémentation de la recherche de circuits euleriens

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 :

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

(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 rechercher des circuits eulériens dans des graphes orientés. La première fonction vérifie si le graphe est ou non eulérien. Les 3 fonctions suivantes sont des fonctions à utiliser pour implémenter la cinquième fonction qui correspond à l'implémentation de l'algorithme de recherche de circuit eulérien.

(a) `est_graphe_eulerien(graphe)` : retourne vrai si `graphe` est un graphe eulérien et faux sinon (on utilise le théorème d'Euler pour vérifier cela) ;
(b) `nombre_arcs(graphe)` : retourne le nombre d'arcs du graphe `graphe` ;
(c) `indice_premier_sommet_avec_successeur(graphe, circuit)` : retourne l'indice du premier sommet de `circuit` qui a au moins un successeur dans `graphe` ;
(d) `construit_circuit_simple_algo_eulerien(graphe, s)` : construit un circuit simple, issu du graphe `graphe`, à partir du sommet `s`, pour l'algorithme de recherche d'un circuit eulérien (c'est-à-dire qu'on utilisera cette fonction après s'être assuré qu'un circuit eulérien existait dans le graphe) et retourne ce circuit. On supprime également du graphe les arcs correspondant à ce circuit, au fur et à mesure de leur sélection ;
(e) `construit_circuit_eulerien(graphe)` : construit un circuit eulérien, pour le graphe `graphe`, en commençant par s'assurer que ce graphe est bien eulérien.

D'autres fonctions bonus peuvent être implémentées et concernent les graphes semi-eulériens.

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 chacune de vos fonctions, vous pouvez utiliser la cellule de code suivant chaque fonction, qui contient des petits tests sur les graphes `graphe1eulerien`, `graphe2eulerien` et `graphe3noneulerien`. Les réponses attendues, pour les 5 fonctions, sont données en commentaire des appels de fonction.

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