ARE - SmartGrid

TP 02 – Manipulation de graphes en python.

Julien Sopena

Lors de ce TP vous allez implémenter un certain nombre de fonctionnalités de base pour l'étude des graphes. Ce travail est à réaliser dans le squelette graphProperties.py. Ainsi, chaque question correspond à un ou plusieurs commentaires notés ### **TODO : ...** ###.

Le fonctionnement du squelette est le suivant : il commence par parser ("lire et comprendre") un fichier texte contenant la matrice d'adjacence d'un graphe, puis exécute automatiquement un certain nombre de fonctions et de tests que vous allez implémenter progressivement. Avant vos modifications, le programme est capable de lire le fichier mais les fonctions retournent de fausses valeurs ("-1", "False", ...). Outre ce squelette, on vous fournit aussi les fichiers correspondants aux 3 graphes étudiés lors de la première séance : graph1.txt, graph2.txt et graph3.txt

Les fonctionnalités implémentées dans le cadre de ce TP vont servir de base à la réalisation de votre projet. Il est donc indispensable de vous assurer de leur bon fonctionnement. Pour vous aider, un ensemble de tests vous est proposé dans le squelette. Pour les activer, utilisez l'option `-t` ou les options `-t -v` pour plus de détails. La présence de ces tests ne vous dispense pas de réaliser manuellement (en générant vous même des fichiers graphes) d'autres tests par vous même.

Exercice 1 : Propriété des graphes

Question 1

Pour commencer implémentez les fonctions **numVertices** et **numEdges** qui retournent respectivement le nombre de sommets et le nombre d'arcs du graphe passé en paramètre.

Question 2

Implémentez la fonction **allVertices** qui retourne une liste (tableau) contenant tous les identifiants. Cette fonction vous permettra de simplifier vos boucles `for` pour parcourir le graphe.

Question 3

Implémentez la fonction **isReflexive** qui retourne vrai si le graphe est réflexif, faux sinon.

Question 4

Implémentez la fonction **isIreflexive** qui retourne vrai si le graphe est irréflexif, faux sinon.

Question 5

Implémentez la fonction **isSymetric** qui retourne vrai si le graphe est symétrique, faux sinon.

Question 6

Implémentez la fonction **isAsymetric** qui retourne vrai si le graphe est asymétrique, faux sinon.

Question 7

Implémentez la fonction **isTransitive** qui retourne vrai si le graphe est transitif, faux sinon.

Exercice 2 : Parcours en profondeur et connexité

Question 1

Implémentez la fonction **DFS** qui réalise un parcours en profondeur du graphe et retourne la liste des sommets dans l'ordre de leur première rencontre. En cas de choix, vous vous assurerez de choisir le sommet ayant le plus petit identifiant. Si vous ne respectez pas cette dernière règle vous ne passerez pas le test de validité même si votre parcours est correct.

Question 2

Avant d'utiliser votre parcours en largeur pour vérifier la connexité du graphe, commencez par implémenter la fonction **getUndirected** qui retourne une matrice correspondant à la version non-orientée du graphe.

Question 3

Implémentez la fonction **isConnected** qui retourne vrai si le graphe est connexe, faux sinon.

Question 4

Implémentez la fonction **isStronglyConnected** qui retourne vrai si le graphe est fortement connexe, faux sinon.

Exercice 3 : Parcours en largeur et calcul du diamètre

Question 1

Implémentez la fonction **BFS** qui réalise un parcours en largeur du graphe et retourne la liste des sommets dans l'ordre de leur rencontre. Comme pour le *DFS*, vous considérez les sommets par ordre croissant.

Question 2

Implémentez la fonction **diameter** qui retourne le diamètre du graphe, ou -1 si le graphe n'est pas fortement connexe.