TP5 Informatique
Accessibilité et connexité
J1 MI 1003, groupe B3, Université Bordeaux

Exercice 1 Faire l'exercice 5.3.4 du fascicule afin de programmer la fonction `accessible(G, sNom, tNom)`.

Exercice 2 Faire l'exercice 5.3.5 du fascicule afin de programmer la fonction `estConnexe(G)`.

Exercice 3 Faire l'exercice 5.3.6 qui consiste à modifier vos programmes précédents afin de faire apparaître les arêtes utilisées lors du parcours du graphe.

Exercice 4 Ajouter de l'aléatoire dans votre parcours comme indiqué dans l'exercice 5.3.7.

L'algorithme décrit dans votre fascicule est loin d'être le plus efficace, mais il a l'avantage de ne pas être compliqué. Les exercices suivants consistent à implémenter un test de connexité avec une complexité moindre.

Exercice 5 Écrire une procédure `prochaineVague(l)` qui, étant donnée une liste de sommets marqués `l`, renvoie la liste des sommets **non marqués** qui admettent un voisin dans `l` et qui les marque.

Exercice 6 Implémenter l'algorithme de connexité suivant :

1. Démarquer tous les sommets et arêtes.
2. Considérer un sommet `s` quelconque et le marquer.
3. Initialiser `l` à `[s]`.
4. Calculer la procédure `prochaineVague(l)`, et remplacer `l` par cette valeur. Répéter cette opération jusqu'à que `l` soit vide.
5. Tester si tous les sommets sont marqués.

Exercice 7 Modifier `prochaineVague(l)` de manière à ce que les arêtes utilisées soient marquées.

Exercice 8 Inspirez-vous de l'algorithme précédent afin d'écrire un programme `distance(G,s1,s2)` qui calcule le nombre minimal d'arêtes pour rejoindre un sommet `s2` à partir d'un autre sommet `s1` dans le graphe `G`. (Renvoyer `None` si on ne peut rejoindre `s2` à partir de `s1`.)