Rapport Algorithme Test de Planarité

Par Remi Herve, Ludovic Marquet et Valentin Jubert

# Analyses

En espace, l’algorithme stocke le graphe sous forme de liste d’adjacence. On stocke donc les sommets, et les voisins sont des pointeurs sur les sommets voisins. Malheureusement, par soucis de simplicité algorithmique, un doublon des sommets est créé pour le second graphe.

En temps, l’algorithme parcourt à plusieurs reprises les sommets ou les arêtes d’une partie du graphe initial. Le calcul de composante connexe s’exécute en O(n+m) (en cas de multiple composantes, le parcours ne visite le graphe qu’une seule fois).

# Problèmes et solutions

Nous avons rencontré quelques problèmes concernant le calcul du cycle. En essayant un marquage, le code s’est avéré ne pas être très clair. En optant pour une liste, il y a eu des soucis lors de l’ajout d’un chemin, donc nous avons finalement décidé d’implémenter le cycle sous la forme d’un second graphe avec une copie des sommets.

Ensuite, un problème est survenu au niveau du calcul des fragments. Nous avons décidé de créé un graphe représentant la différence du graphe avec le cycle, puis d’exécuter un calcul de composantes connexes sur ce graphe.

Enfin, le dernier problème a été la mise à jour des faces à partir d’un chemin donné. En effet, il a fallu créer deux faces à partir d’une face, et d’un chemin. Pour cela, on exécute un parcours sur la face à la manière d’un cycle, et on divise l’ensemble des sommets de la face en 2 parties, puis on ajoute le chemin dans les deux parties.

# Explication méthodes complexes

Mise à jour Face

Entrée : Un chemin C, une face F (une liste de sommet)

Sortie : 2 Faces

Début

F1 <- Vide (Une liste sans doublon)

F2 <- Vide (Une liste sans doublon)

Si le dernier sommet du chemin est le premier à apparaitre dans la Face (par rapport au premier

sommet de C)

Inverser C

Pour chaque sommet v de la face {

Si v est avant le premier sommet du chemin ou après le dernier sommet du chemin

F1 <- F1 U V

Si v est après le premier sommet du chemin et avant le dernier

F2 <- F2 U V

Si v est le premier sommet du chemin {

Mettre v dans F2

Mettre les sommets du chemin dans F1

}

Si v est le dernier sommet du chemin

Mettre v dans F2

}

Inverser C

Ajouter C \ {1er Sommet U Dernier sommet} dans F2

Retourner F1 et F2

Fin

# Exécution

Pour pouvoir exécuter le projet il est nécessaire d’avoir Java 8. Le programme demande un unique argument, qui est l’adresse du fichier représentant le graphe.