Rapport de projet d'algorithmique

Vivien Matter Ernest Foussard

Avril 2018

Table des matières

1 Introduction

L'objectif du projet était de modifier des graphes de manière à ce qu'il soit possible d'y trouver un cycle eulerien, puis de le calculer. Pour cela, on a été amenés à implémenter plusieurs algorithmes et à s'interroger sur la complexité de ces derniers.

2 Revue des algorithmes

L'algorithme général se décompose en plusieurs sous-algorithmes : il faut d'abord reconnecter les composantes connexes du graphe, puis ajouter des arêtes de telle sorte à ce que tous les sommets soient de degré pairs, et enfin chercher un cycle eulerien.

2.1 Reconnexion des composantes connexes

2.1.1 Algorithme quadratique

L'algorithme naïf consiste à rechercher toutes les arêtes possibles avec l'ensemble de points du graphe, de les trier par longueur, puis d'ajouter les segments qui relient deux composantes connexes distinctes entre elles jusqu'à ce qu'il n'y ait plus qu'une seule composante connexe. Soit n le nombre de segments du graphe, cet algorithme admet une complexité temporelle $O(n^2 log(n))$. On a implémenté une optimisation qui consiste à ne considerer que les arêtes qui relient deux composantes initialement distinctes, ce qui se réalise facilement avec la structure d'union-find. La complexité dans le pire des cas (cas où l'ensemble des arêtes est initialement vide) est toujours $O(n^2 log(n))$, mais dans les situations pas trop tordues, le gain en temps est considerable.

2.1.2 Algorithme avec table de hashage

L'algorithme avec table de hashage consiste à construire un mappage spatial pour repérer les points par rapport aux autres, et ensuite ajouter de manière gloutonne de manière quasiment ordonnée (on ne trouve pas nécessairement les plus petits segments, mais les segments du même ordre de grandeur que le plus petit). Le gain en complexité temporelle est considerable, a priori, et s'écrit $O(nlog(\alpha))$ où α est la plus petite distance entre deux points du graphe. Néanmoins, la complexité spatiale s'écrit $O(4^\alpha)$ ce qui est extremement problématique dès qu'on a deux points très proches, car le coût temporel pour instancier de tels objets devient alors très important. Notre implémentation de cet algorithme ne s'execute pas en temps raisonnable pour la majorité des figures (c'est certainement dû également à un manque d'optimisation ou une mécomprehension de l'algorithme).

2.2 Construction du graphe de degrés pairs

2.2.1 Algorithme quadratique

L'algorithme naïf consiste à parcourir l'ensemble des points une première fois pour compter les points de degré impairs, puis on parcourt l'ensemble des couples de points du graphe par ordre croissant jusqu'à ce qu'il n'y ait plus de sommets de degré impair, la complexité temporelle de cet algorithme est un $O(n^2 log(n))$. On a optimisé cet algorithme en ne triant et en n' iterant que sur les sommets de degré impair. La complexité dans le pire des cas reste un $O(n^2 log(n))$ mais elle est général bien moins importante (voir la partie perfomances).

2.2.2 Algorithme avec table de hashage

pas implémenté

2.3 Détection d'un cycle eulerien

g pa kompri

3 Tests de performance

hihihi.jpeg

4 Conclusion

pas fait