15.Séoul

Gérer un réseau de transports en commun, par exemple le métro de Séoul, via un graphe (Lis et Mco). Résoudre dans ce cas le problème du plus court chemin, algorithme de Dijkstra.

Mustapha Ilyès DAHMANI

Numéro d'étudiant : 16 70 27 85

Décembre 2018

Plan

1. Les différentes étapes à faire pour ce projet.

2. Explication de certaines parties du code

3. Erreurs et améliorations qu'on pourrait apporter.

1. Les différentes étapes à faire pour ce projet.

1. Carte, une origine et une destination.
2. Création de la base de données
3. Traduction des informations de la base de données dans une matrice stations_finales de structure station.
4. Création d'une matrice compacte et d'une liste de successeurs à partir de la matrice stations_finales.
5. Application de l'algorithme de Dijkstra sur la matrice compacte et sur la liste de successeurs.
6. Correction apportée et correspondances.

2. Explication de certaines parties du code

-voir stations_finales (la récupération d'information)

-voir création Mco et Lis

-voir application Dijkstra et la correction apportée

3. Erreurs et améliorations qu'on pourrait apporter.

Erreur : on observe parfois des différences de trajets quand on veut faire des trajets aller-retour, alors qu'il devraient passer par le même chemin (mais inversé). (sauf pour le cas des one-way station => cas particulier).

Problème : l'ajout de correspondances fausse l'algorithme de Dijkstra, malgré la création d'une fonction CorrectionChemin().

On aurait pu faire une variante de la procédure "CorrectionChemin()" en amont, c'est-à-dire l'utiliser dans la procédure "evaluationDijkstra()" En recalculant, lorsqu'il y a une correspondance, le poids des successeurs du pivot.

On aurait pu "éclater" les stations en autant de stations que de lignes la traversant. On aurait alors appliqué l'algorithme de Dijkstra sans contraintes de correspondances, mais on aurait utilisé plus de place mémoire.