TME5. Plus courts chemins et arbre de Steiner.

BM Bui-Xuan

Pendant cette séance de TME, les algorithmes sont à implanter dans le canevas `TME5_cpa_20160217.jar`.

Graphe géométrique : Un graphe géométrique dans un plan 2D est défini par un ensemble de points dans le plan appelés sommets, et un seuil sur la distance entre les points : il existe une arête entre deux sommets si et seulement si la distance Euclidienne entre les deux sommets est inférieure à ce seuil. Dans le fichier canevas la liste des sommets est appelée `points` et le seuil est appelé `edgeThreshold`. La valeur de `edgeThreshold` peut être modifiée dans le fichier `build.xml`. Tout graphe géométrique pendant cette séance de TME est pondéré sur les arêtes. Quand une arête existe entre deux sommets, le poids de l'arête est la distance Euclidienne entre les deux sommets dans le plan. Pour nous épargner de discours inutile, on peut dire que les graphes géométriques sont actuellement dans le monde numérique (et ses réseaux 2.0) ce que le sel est dans le monde réel.

Problème des plus courts chemins dans un graphe : Etant donné un graphe $G = (V, E)$, avec $V =$`points`, le problème des plus courts chemins de type "*all-to-all*" consiste à calculer un plus court chemin entre toute paire de sommets de G. Pendant cette séance de TME, nous nous intéressons exclusivement à la représentation des plus courts chemins par une matrice d'accès : `paths[i][j]` renvoie l'indice `k` du prochain sommet dans un plus court chemin de `points.get(i)` à `points.get(j)`. Ainsi, on peut retracer ce chemin de la façon suivante :

`points.get(i), points.get(k), points.get(paths[k][j]), ..., points.get(j).`

Problème de l'arbre de Steiner dans un graphe : Etant donnés un graphe $G = (V, E)$ et un sous ensemble $S \subseteq V$ de sommets, le problème de l'arbre de Steiner couvrant S consiste à calculer un sous graphe de G qui est un arbre et qui passe par tous les points de S, tel que la longueur totale des arêtes de l'arbre est la plus petite possible. Dans le fichier canevas, V est représenté par la liste `points`, S par la liste `hitPoints`, et E est défini selon le modèle des graphes géométriques où le seuil est `edgeThreshold`.

1 Plus courts chemins

Le problème des plus courts chemins est polynomial. Il s'agit d'un sujet élémentaire en théorie des graphes. Ses applications, notamment dans le transport ferroviaire, ont permis à la génération X d'acquérir maintes richesses inoubliables (dont la génération Y parle encore, parfois).

Exercice 1 – Algorithme Floyd-Warshall

1. Implanter l'algorithme Floyd-Warshall vu en cours dans la méthode `calculShortestPaths`. La valeur de retour de cette méthode doit impérativement être la matrice d'accès dont la définition est donnée ci-dessus.

2 Heuristique pour arbre de Steiner dans un graphe

Le problème de l'arbre de Steiner est NP-complet, même lorsque l'on se restreint aux graphes géométriques. Ce sujet, très classique, en algorithmique des graphes a des applications au delà du normal, voire au delà du paranormal.

Exercice 2 – Heuristique avec des plus courts chemins

Soit $G = (V, E)$ un graphe et $S \subseteq V$ un sous ensemble de sommets de G. Nous nous intéressons à la résolution du problème de l'arbre de Steiner couvrant S dans G. Le principe de l'heuristique (également vue en cours) est le suivant :

- Construire le graphe pondéré complet $K = (S, OS, w)$ avec $OS = S \times S$ et, pour tout couple de sommets $u \in S$ et $v \in S$, le poids de l'arête entre ces deux sommets $w(uv)$ est défini par la longueur du plus court chemin entre u et v dans G.
- Dans K, construire un arbre couvrant T_0 de longeur totale des arêtes la plus petite possible.
- Dans T_0, remplacer toute arête uv par un plus court chemin entre u et v dans G. Soit H le graphe obtenu après cette étape. On remarque que si U dénote l'ensemble des sommets de H, alors $S \subseteq U \subseteq V$.
- Dans H, construire un arbre couvrant T' de longeur totale des arêtes la plus petite possible.
- Retourner T'

1. Implanter l'heuristique en question dans la méthode `calculSteiner` et analyser sa complexité.