Compte Rendu SAE S2.02 - Exploration algorithmique d'un problème

Jeanne Lebeau et Manon Chaperon Groupe 1C1

Introduction :

Nous avons choisi le problème du postier chinois, ce casse-tête consiste à trouver le chemin le plus court dans un graphe connexe non orienté qui passe au moins une fois par chaque arête et revient à son point de départ.

A l'origine le problème vient d'un mathématicien qui a étudié la tournée d'un facteur qui devait effectuer le plus efficacement possible sa tournée en passant au moins une fois par chaque rue de son secteur.

Il a pour particularité de pouvoir être résolu si le graphe est eulérien, ce qui rend alors le cycle le plus court eurélien. Mais l'une des difficultés est que chaque sommet du graphe doit être de degré pair pour qu'un tel cycle existe.

Description de la solution :

Pour obtenir une solution à ce problème il faut premièrement rendre eulérien le graphe, en essayant de minimiser la longueur totale des arêtes qu'on ajoute.

Ensuite on doit chercher les isthmes ou ponts du graphe et chercher un circuit eulérien dans le graphe complété en utilisant l'algorithme de Fleury.

Le type d'algorithme que nous avons mis en place est celui du plus court chemin entre deux points.

Notre programme fait appel à plusieurs algorithmes, tout d'abord "pile" qui nous permet d'utiliser le système de pile , "test" qui nous permet de tester l'ensemble, c'est le fichier qu'on doit compiler et pour finir le programme principal.

Il contient plusieurs méthodes :

__init__(self, graphe_dict=None) : La méthode initialise un objet graphe. Si aucun dictionnaire n'est créé ou ne lui est donné, on en utilisera un vide.

aretes(self, sommet) : La méthode retourne la liste de toutes les arêtes d'un sommet.

all_sommets(self) : La méthode retourne tous les sommets du graphe.

all_aretes(self) : La méthode retourne toutes les arêtes du graphe.

add_sommet(self, sommet) : Si le “sommet” n’est pas déjà présent dans le graphe, la méthode ajoute au dictionnaire une clé “sommet” avec une liste vide pour valeur. Sinon elle ne fait rien.

add_arete(self, arete) : L'arête est de type set, tuple ou list. Entre deux sommets il peut y avoir plus d’une arête (multi-graphe).

nb_aretes(self) : La méthode retourne le nombre d’arêtes.

__list_aretes(self) : La méthode est privée et récupère les arêtes. Une arête est un ensemble (set) avec une boucle ou deux sommets

trouve_chaine(self, sommet_dep, somet_arr, chain=None) : La méthode trouve un chemin élémentaire de sommet_dep à sommet_arr dans le graphe.

trouve_tous_chaines(self, sommet_dep, sommet_arr, chain=[]) : La méthode trouve les chemins élémentaires de sommet_dep à sommet_arr dans le graphe.

__iter__(self) et __next__(self) : Les méthodes servent à itérer sur les sommets du graphe.

sommet_degre(self, sommet) : La méthode renvoie le degré du sommet.

trouve_sommet_isole(self): La méthode renvoie la liste des sommets isolés.

Delta(self) : La méthode retourne le degré maximum.

list_degres(self) : La méthode calcule tous les degrés et renvoie un tuple de degré décroissant.

eulerien(self) : La méthode teste si le graphe est eulérien.

<u>Comparaison :</u>

Par exemple, un des algorithmes qui résout ce problème est celui de qui traverse les arêtes au hasard jusqu'à trouver un itinéraire qui soit sans issue, ou soit qui aboutit à un circuit

Tout d'abord on peut comparer les similitudes, les deux programmes testent d’abord si le graphe est eulérien, le rendent eulérien si il ne l’est pas et ensuite cherchent un chemin eulérien.

Ensuite on peut développer les différences, le nôtre trouve le plus court chemin d’un point à un autre. Celui-ci trouve le parcours le plus court dans tout le graphe. Une

des différences principales est que notre programme ne passe pas par toutes les arêtes du graphe.

Conclusion :

Pour conclure, notre programme parcours le graphe, si il est eurélien l'algorithme cherche le chemin le plus court. Nous avons eu des difficultés pour trouver le chemin, notre programme trouve le plus court chemin d'un point à un point et non pas de tout le graphe. Nous aurions pu l'améliorer en faisant un programme qui résout entièrement le problème et qui passe par toutes les arêtes du graphe. Une des solutions alternatives aurait été d'utiliser l'algorithme de dijkstra.