Projet Gogol car

A réaliser en binôme [1]

1 Le sujet

Nous considérons une ville dont les rues sont toutes à double sens. La fin d'une rue est toujours une place, quel que soit le nombre de rues qui y débouchent. Puisque c'est une ville, nous supposerons que les rues permettent de joindre n'importe quelle paire d'endroits dans la ville. Une *Gogol car* a pour mission de passer dans toutes les rues de la ville et de photographier les deux cotés de la rue. Nous supposerons que :

- lorsque la Gogol car s'est engagée dans une rue, elle la parcourt jusqu'au bout (pas de demi-tour dans la rue)
- les places n'ont pas besoin d'être photographiées
- la Gogol car peut passer dans une rue sans rien photographier, juste pour se déplacer.

On souhaite écrire des algorithmes pour définir le trajet de la Gogol car, afin qu'elle puisse photographier toutes les rues, selon plusieurs variantes en revenant au point de départ. Ce qu'on se donne au départ est une carte de la ville, avec les noms des places (du genre "place de la République") et des rues (du genre "Grande rue"). La Gogol car peut commencer son trajet dans n'importe quelle place de la ville.

Variante 1. Algorithme GogolS.

Dans cette variante, la Gogol car ne peut photographier que du côté droit de la voiture. On demande que le programme fournisse un trajet le plus court possible permettant à la Gogol car de photographier les deux côtés de chaque rue de la ville.

Variante 2. Algorithmes GogolL et GogolXL.

Dans cette variante, la Gogol car peut photographier les deux côtés de la rue en un seul passage. Le but est de faire cela en minimisant la longueur du trajet, définie comme le nombre de rues parcourues entre le départ de la Gogol car de sa position initiale (que vous choisirez) et son retour.

L'**algorithme GogolL** doit fonctionner dans le cas particulier où le graphe défini par la ville contient un cycle eulerien. Pour construire le trajet de la Gogol car, l'algorithme suivant est proposé, en supposant que le graphe non-orienté est représenté comme un graphe orienté avec des arcs opposés à la place de chaque arête :

(1) Choisir un sommet quelconque r dans le graphe et construire une anti-arborescence dans G d'anti-racine r. Une anti-arborescence d'anti-racine r est un sous-graphe qui devient une arborescence de racine r une fois que tous ses arcs ont été inversés.

(2) Pour chaque sommet x de G faire
Numéroter de 1 à $d_G^+(x)$ (degré sortant de x) les arcs de G sortant de x, en mettant le plus grand numéro sur l'arc sortant appartenant à l'anti-arborescence (pour l'anti-racine, la numérotation est quelconque).

(3) Partir de r, et tant qu'il existe un arc non encore utilisé permettant de quitter le sommet où l'on se trouve, choisir celui de plus petit numéro et le parcourir.

1. Un unique monôme est accepté, et cela seulement si le nombre d'étudiants effectuant le projet est impair

L'**algorithme GogolXL** doit fonctionner même dans le cas où le graphe défini par les rues de la ville n'est pas eulérien. Il n'y a pas d'algorithme proposé dans ce cas, c'est à vous de le proposer, chercher, adapter etc. par exemple en vous renseignant sur le problème dit du Postier Chinois. Dans ce cas, on ne vous demande pas un algorithme exact (bien qu'il en existe) : les algorithmes du cours permettant l'obtention de votre solution seront implémentés de manière exacte ; ceux qui n'ont pas été vus en cours seront remplacés par des heuristiques. L'algorithme retournera non seulement la solution (le trajet proposé) mais également les étapes de sa construction.

2 Travail à fournir

Le programme sera implémenté en Java. Il sera intégralement écrit par vos soins. Il doit permettre, à l'aide d'un menu, d'entrer la ville sous la forme standardisée indiquée ci-dessous, de choisir la variante GogolS, GogolL ou GogolXL même si la variante ne s'applique pas au réseau (c'est au programme d'indiquer si tel est le cas) et de récupérer le trajet à la fois à l'écran et dans un fichier. On ne demande pas d'affichage graphique.

La ville sera fournie au programme dans un fichier de la forme suivante (sans aucun espace en début de ligne, et avec un . pour finir la ligne) :

— la première ligne contient le nombre n de places
— la deuxième ligne contient le nombre m de rues
— les n lignes suivantes contiennent les noms des places
— les m lignes suivantes contiennent les noms des rues et les deux places qui délimitent chaque rue, sous la forme
  Grande Rue ;Place de l'Eglise ;Place du Marché.

Pour les algorithmes du cours comme pour ceux que vous écrirez vous-mêmes, un lecteur ne doit jamais être perdu, il doit comprendre ce que vous faites et comment vous le faites. La notation prendra fortement en compte cet aspect de votre programme.

3 Rendu de TP

Les programmes doivent fonctionner parfaitement sur les machines du CIE, sous Linux. Un rapport d'au maximum 10 pages sera fourni, indiquant (entre autres) :

— les commandes de compilation et exécution en mode console
— les algorithmes utilisés (avec leurs descriptions en pseudo-code, pas en Java)
— les justifications des algorithmes utilisés (pour quelle raison affirmez-vous qu'ils fournissent la solution cherchée ?)
— les complexités de ces algorithmes et la complexité globale de votre programme
— des jeux de données

Les fichiers du programme, ainsi que les jeux de données seront mis dans une archive de nom Nom1Nom2.tar.gz (où Nom1 et Nom2 sont les noms des deux étudiants du binôme). L'archive sera déposée sur Madoc, dans l'espace dédié à votre groupe, au plus tard le 30 novembre 2016, jour du dernier TP, à 23h59.

Attention, la toute dernière séance d'1h20 est dédiée à une démo de vos programmes (10min par programme).

Aucun dépôt en retard ne sera autorisé par Madoc.