MOALIGOU Elise
MALESTROIT Lilwen
Groupe 1D

SAE 02.02

Le problème du postier chinois

Introduction

Le problème du postier chinois modélise le trajet d'un facteur qui aurait à déposer des lettres dans plusieurs maisons d'une ville, et à revenir finalement à son point de départ. Lors de sa tournée, le facteur doit choisir le plus court chemin et ne passer qu'une seule fois dans chaque rue afin de prendre le moins de temps possible et d'optimiser son passage. Pour modéliser ce problème nous allons utiliser un graphe non orienté dans lequel les arêtes représenteront des rues de la ville et les sommets les intersections entre les rues. Nous avons choisi ce problème, car nous avons très vite réussi à nous approprier le sujet et à le mettre en relation avec nos cours de graphes. Aussi, le fait que ce soit un problème très concret ayant des applications pratiques dans de nombreux domaines nous a beaucoup plu.

Description des Algorithmes

Approche générale

Le problème du postier chinois se résout sur un graphe eulérien. Ce type de graphe permet d'effectuer un cycle eulérien, concrètement, ce cycle permet – à partir d'un point X – de parcourir une seule et unique fois chaque arête du graphe, tout en revenant au point X à la fin du trajet.

Le souci, hors implémentation de l'algorithme du cycle eulérien, est donc de déterminer si un graphe donné est eulérien ou non. Et, sinon, de modifier ce graphe pour qu'il le devienne. C'est sur ce point que les deux algorithmes fonctionnent différemment.

Les deux algorithmes ont été réalisés en développement objet en python :

- Premier nommé LePostierChinois1.py
- Second nommé LePostierChinois2.py

Description de la base commune

Méthode `algorithme(_pc1/_pc2)` :

Cette méthode permet de résoudre le problème du postier chinois et de **retourner un chemin que le postier pourra emprunter**.

La méthode teste en premier lieu le nombre de sommets du graphe. Si ce nombre est supérieur à deux, il sera possible de travailler avec un parcours eulérien afin de retourner le chemin que le postier devra parcourir.

Elle teste d'abord si le graphe utilisé est eulérien et se charge de le transformer comme tel si ce n'est pas le cas. Et fait ensuite le nécessaire pour faire en sorte qu'un cycle eulérien puisse se faire. Elle lance enfin la méthode retournant un cycle eulérien (c'est-à-dire le chemin que doit emprunter le postier), lui-même retourné par la méthode en cours.

Méthode `cycle_eulerien` :

Cette méthode permet de **créer et de renvoyer un cycle eulérien**. Ce dernier est construit en passant de sommet en sommet, son but est de ne pas repasser par les mêmes arêtes, c'est pourquoi ces dernières sont supprimées dès qu'elles sont empruntées par le postier (autrement dit, dès qu'elles rejoignent la liste des arêtes parcourues).

Méthode `sommet_suivant_cycle_eulerien` :

Cette méthode détermine le sommet devant être le prochain dans la création du cycle et le retourne. Le sommet suivant est déterminé grâce à son degré. Sauf cas exceptionnel, c'est **le voisin ayant le degré le plus petit qui sera retourné**.

Cas 1 : Le sommet sur lequel la méthode travaille à plusieurs voisins :

Dans ce cas, la méthode **exclut naturellement tout voisins de degré 1**, permettant ainsi qu'il ne soit pas le sommet suivant.

Un voisin de degré 1 ne peut correspondre qu'au point de départ du facteur. Or, dans ce cas, il n'a pas encore fini sa tournée, il ne peut donc pas rentrer.

Cas 2 : Le sommet sur lequel la méthode travaille n'a qu'un seul voisin :

L'unique voisin est donc déterminé comme étant le sommet suivant.

Description de l'algorithme 1 : LePostierChinois1

Méthode `transforme_toEulerien` :

Cette méthode permet de **transformer un graphe non eulérien en eulérien**. Un graphe eulérien est un graphe connexe – c'est-à-dire qu'il n'y a pas de sommet isolé – et dont tous les sommets sont de degrés pairs. Elle commence donc par s'occuper des sommets isolés, puis des sommets de degré impairs.

Sommet isolé :

La méthode **crée une arête entre un sommet isolé et le suivant**, formant une sorte de chenille. Puis **relie** le **dernier sommet isolé trouvé** à un sommet de degré impair si possible et pair autrement.

Sommet impair :

La méthode **crée des arêtes aux sommets de degré impair** afin de les rendre pairs.
Elle priorise le fait de créer des arêtes entre deux sommets impairs lorsque c'est possible, elle les relit à un sommet pair s'il n'y a pas d'autres choix.
Dans cet algorithme, le **redoublement d'arête est interdit**, c'est pourquoi dans un cas où un sommet est déjà relié à tous les autres sommets du graphe mais est de degré impair, la **possibilité de supprimer** une des arêtes est nécessaire.

Description de l'algorithme 2 : `LePostierChinois2`

Méthode `transforme_toEulerien` :

Cette méthode permet **de transformer un graphe non eulérien en eulérien**. Un graphe eulérien est un graphe connexe – c'est-à-dire qu'il n'y a pas de sommet isolé – et dont tous les sommets sont de degrés pairs. Elle commence ainsi par s'occuper des sommets isolés, puis des sommets de degré impairs.

Sommet isolé :

La méthode **crée une arête entre un sommet isolé et le suivant**, formant une sorte de chenille. Puis **relie** le **premier et dernier sommets isolés** trouvés à un sommet de degré impair si possible et pair sinon.

Sommet impair :

La méthode **crée des arêtes aux sommets de degré impair** afin de les rendre pairs.
Elle priorise le fait de créer des arêtes entre deux sommets impairs lorsque c'est possible, dans le cas ou ça ne l'est pas, elle **double une arêtes déjà existante** du sommet impair.

MOALIGOU Elise
MALESTROIT Lilwen
Groupe 1D

Comparaison des Algorithmes

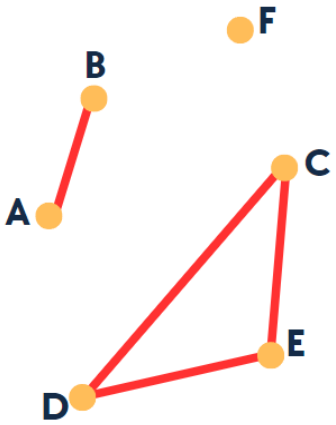
Pour le premier algorithme, nous avons déterminé que nous pourrions ajouter et supprimer des arêtes au graphe afin de la rendre eulérien. Donc qu'il était possible de construire et de détruire des rues. Cette suppression d'arête est obligatoire, car nous avions la volonté que le doublement de rues soit interdit.

Pour le second algorithme, seul l'ajout d'arête est possible pour rendre le graphe eulérien. Dans ce cas-là le doublement d'arête est possible, on peut d'ailleurs l'illustrer par une grande avenue ou le facteur est obligé de passer dans les deux sens pour distribuer ses lettres des deux côtés de la route.

La comparaison des deux algorithmes se fait donc sur le même graphe (excepté le fait que l'un a des sets pour valeur tandis que l'autre des listes)

```
graphe_algo1 = {
                "A" : {"B"},
                "B" : {"A"},
                "C" : {"D", "E"},
                "D" : {"C", "E"},
                "E" : {"C", "D"},
                "F" : set()
            }
```

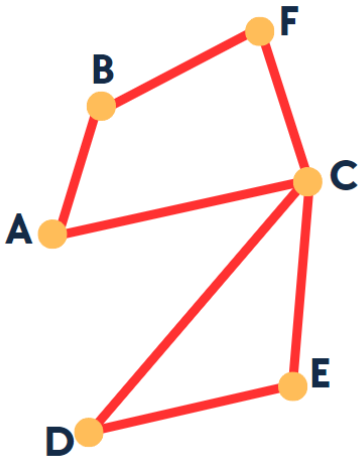
```
graphe_algo2 = {
                "A" : ["B"],
                "B" : ["A"],
                "C" : ["D", "E"],
                "D" : ["C", "E"],
                "E" : ["C", "D"],
                "F" : []
            }
```

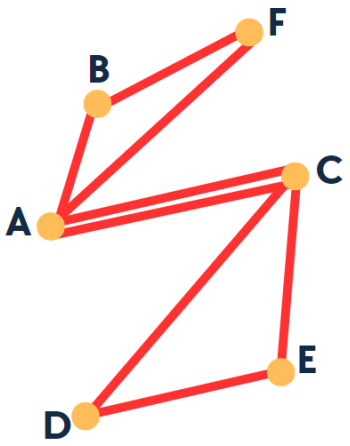


Après le lancement des deux algorithmes, les graphes ont été modifiés :

```
graphe_algo1 = {
                "A" : {"B", "C"},
                "B" : {"A", "F"},
                "C" : {"D", "E", "A", "F"},
                "D" : {"C", "E"},
                "E" : {"C", "D"},
                "F" : {"C", "B"}
              }
```

```
graphe_algo2 = {
                "A" : ["B", "F", "C", "C"],
                "B" : ["A", "F"],
                "C" : ["D", "E", "A", "A"],
                "D" : ["C", "E"],
                "E" : ["C", "D"],
                "F" : ["A", "B"]
              }
```





Pour l'algorithme 1, 3 arêtes ont été créées, tandis que pour le deuxième, il y en a eu 4.

Le temps d'exécution diffère également, 0.456 secondes pour le premier et 0.503 secondes pour le second.

Enfin, les chemins parcourus ne sont pas les mêmes.

```
['A', 'B', 'F', 'C', 'E', 'D', 'C', 'A']
```

```
['A', 'B', 'F', 'A', 'C', 'D', 'E', 'C', 'A']
```

Avec 8 étapes pour le premier et 9 pour le second.

C'est donc l'algorithme 1 qui semble le plus efficace pour résoudre le problème du postier chinois.

Conclusion

Solution alternative :
Étant donné que le graphe est une modélisation d'une ville, alors si l'on modifie le graphe en ajoutant des arêtes c'est comme si l'on créait de nouvelles rues, or cela semble irréaliste. C'est pourquoi, on aurait pu, au lieu de modifier le graphe pour le rendre eulérien, tenter de trouver un chemin que le postier aurait pu emprunter quitte à repasser plusieurs fois par les mêmes arêtes.