Projet Final : Graph Theory (40%)

8GIF128, UQAC.

Enseignant : Edmond La Chance

Introduction

Dans ce projet final, vous devez concevoir 4 programmes qui communiqueront entre eux avec le réseau. Il y aura 3 services et 1 portail (site web) permettant d'utiliser les trois services.

Ces 4 programmes peuvent être faits dans les langages de votre choix (C++, PHP, Ruby, Python etc) car l'intérêt de l'architecture par services est de pouvoir avoir une flexibilité au niveau des technologies utilisées.

Il y a quatre programmes et trois services. Les trois services sont un générateur de graphe, un algorithme pour le Single Source Shortest Path problem et un algorithme pour le Minimum Spanning Tree problem. Les deux derniers services sont des problèmes classiques de la théorie des graphes. Vous irez vous documenter sur chacun et vous choisirez un algorithme. Le portail est un site web permettant d'interagir facilement avec les trois services pour les utiliser.

Le projet se fait en équipe de 4. Vous devez présenter votre travail devant la classe avec une présentation d'une durée d'environ 20 minutes. Il y aura un horaire des présentations plus tard dans le semestre.

Le projet et la présentation valent 40 % de la note finale.

Afin que le soucis soit sur la programmation et l'apprentissage et non sur des détails d'hébergement, le projet doit pouvoir fonctionner au complet sur une seule machine. Le projet peut être fait sous Linux ou Windows. Si le projet est fait sur Linux, inclure des petites instructions, si nécessaires, pour la mise en marche.

La remise du projet se fait avec une archive au format .ZIP
Peut-être en attaché email (attention aux .exe, gmail détruit les emails dont les archives contiennent des exécutables), un lien sur skydrive, google drive, dropbox ou alors un lien vers un répertoire github. Je rappelle mon email : edmond.lachance@gmail.com.

Service 1 : Générateur de graphe

Le premier service est un programme permettant de générer un graphe aléatoire. Un graphe est généralement décrit à l'aide d'une liste d'arêtes. Un algorithme peut ensuite lire la liste des arêtes et créer le graphe dans ses structures de données internes.

Pour les problèmes de graphes de ce projet, les graphes ne sont pas des graphes dirigés, ce qui veut dire que les arêtes vont dans les deux directions. Le terme utilisé est undirected graph[1]

Ce premier service est donc essentiel au fonctionnement des autres services.

Quelques demandes :

- Lorsque l'usager utilise le service de générateur de graphes, il doit entrer le nombre de sommets du graphe et la densité de ce graphe. La densité est un nombre entier entre 0 et 100. Un densité de 50 signifie tout simplement qu'il y a une chance de 50 pourcent qu'il y ait une arête entre le sommet a et b
- Le générateur de graphe doit s'assurer que le graphe est connecté et qu'il n'y ait qu'une seule arête par paire de sommets.
- Un graphe connecté signifie que tous les sommets ont au moins un chemin vers les autres sommets.

Voici un exemple de format de description de graphe simple. La première ligne contient deux valeurs entières : le nombre de sommets (n) et le nombre d'arêtes (m). Les m lignes suivantes décrivent les arêtes de la façon suivante : sommet1 sommet2 poids. Vous pouvez néanmoins employer un format différent ou alors utiliser JSON ou XML.

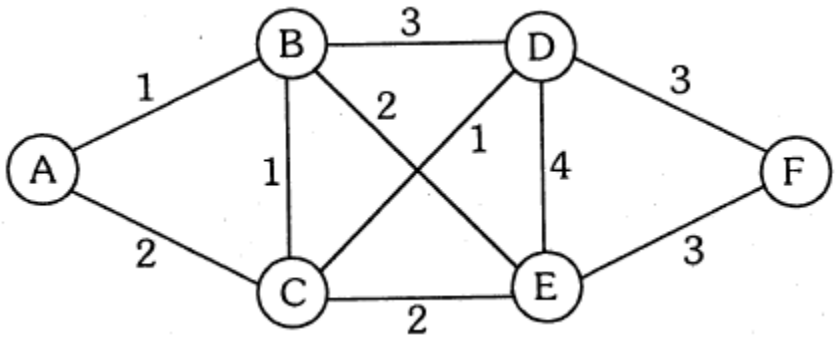
```
6 9
0 4 3
0 1 1
0 3 10
1 3 3
1 2 2
2 3 4
2 5 5
```

[1] http://en.wikipedia.org/wiki/Graph_%28mathematics%29#Undirected_graph

```
3 5 1
3 4 4
```

Service 2 : Single Source Shortest Path problem

Le problème du SSSP consiste à trouver les meilleurs chemins entre une source et tous les autres chemins. C'est un problème classique de la théorie des graphes avec beaucoup d'applications, notamment en jeu vidéos.



Ce programme est un service qui prend en paramètre un graphe et calcule la valeur des meilleurs chemins de chaque sommet vers une source unique. Choisir le premier sommet du graphe comme source unique.

Le résultat est un vecteur ou une string contenant la valeur du meilleur chemin entre chaque sommet et la source. Par exemple, dans ce graphe, certaines de ces distances seraient :

- A-B 1
- A-E 3
- A-D 3

Voir :

http://en.wikipedia.org/wiki/Shortest_path_problem

Service 3 : Minimum Spanning Tree

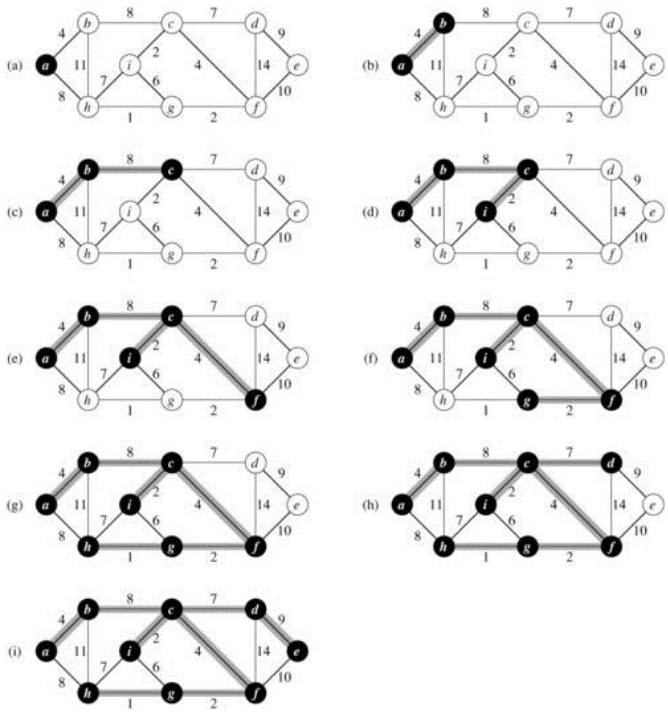
Le minimum spanning tree (Arbre couvrant minimal) est un autre problème classique de la théorie des graphes qui consiste à trouver un ensemble d'arêtes qui couvre tous les sommets avec un poids minimal. C'est un problème qui a été très étudié et les algorithmes ont été améliorés pendant de nombreuses années. Pour ce travail je vous invite à choisir un algorithme plus facile, peut-être un algorithme de Prim avec une matrice de distance ou alors l'algorithme de Kruskal.

Input : Le graphe, dans le format choisi.

Output : Poids total de l'arbre (Addition du poids de toutes les arêtes qui se trouvent dans le MST).

Vous pouvez également imprimer la liste de toutes les arêtes.

http://en.wikipedia.org/wiki/Minimum_spanning_tree



Portail

Le portail permet à l'utilisateur d'utiliser les trois services. Le site doit être fait en XHTML et CSS correct et avoir une belle interface, simple à utiliser.

Voici un croquis que quelqu'un qui est nul en interfaces ferait. Vous pouvez faire mieux ! ☺

AJAX

Essayez de faire en sorte que certains services puissent marcher directement en AJAX. Vous voulez générer un graphe? Il suffit d'appuyer sur le bouton, les résultats apparaissent directement dans le textbox après une requête http asynchrone vers le service.

Visualisation de graphe (Bonus)

Il serait vraiment super de pouvoir voir le résultat de l'algorithme de Minimum Spanning Tree en affichant le graphe résultant.
Il y a de nombreuses librairies Javascript pour cela. Ce post sur StackOverflow en regroupe beaucoup :
http://stackoverflow.com/questions/7034/graph-visualization-library-in-javascript