Algorithmes et structures de données 2
Laboratoire n°3 : Réseau routier

31.10.2018

Introduction

Dans ce laboratoire nous allons continuer à étudier les graphes en les appliquant cette fois-ci à des données réelles. Nous allons nous intéresser au réseau routier de l'ouest de la Suisse, voir la Figure 1.

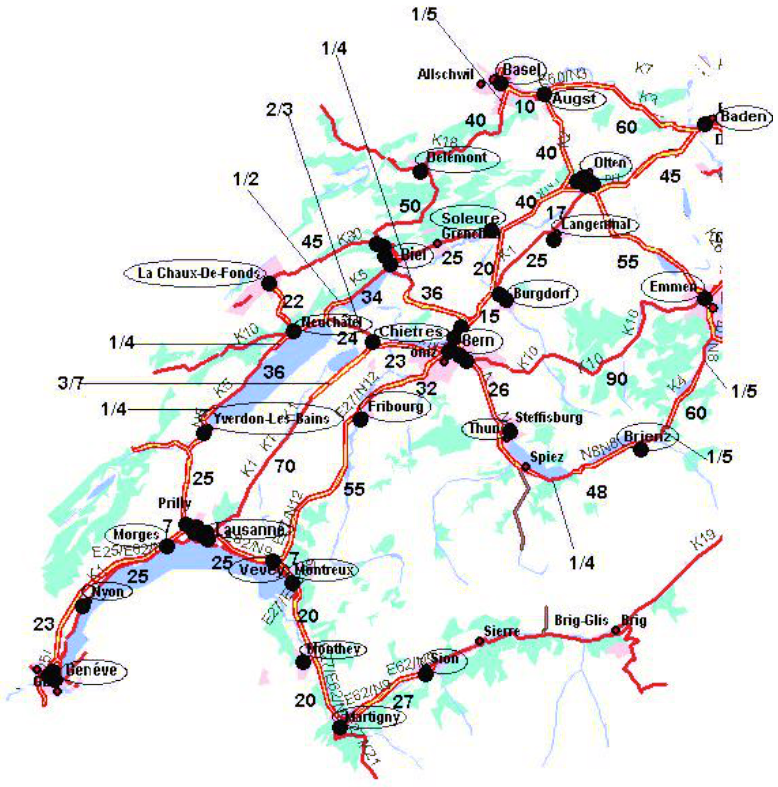

Figure 1 *- Carte du réseau routier de l'ouest de la Suisse. Les sommets sont entourés et les routes représentent les arcs. Les tronçons peuvent être autoroutier, routier ou un mélange des deux. Les fractions indiquées sur l'image correspondent à la partie autoroutière d'un arc. Vous retrouverez ces fractions dans le fichier d'import du graphe.*

Objectifs

L'objectif de ce laboratoire est d'implémenter et d'appliquer des algorithmes, vus en cours, à un graphe représentant des données réelles. L'application que vous nous remettrez devra être en mesure de répondre à certaines questions :

- Quel est le chemin le plus court entre Genève et Emmen ? Quelles sont les villes traversées ?
- Mêmes questions entre Lausanne et Bâle ?
- Quel est le chemin le plus rapide entre Genève et Emmen en passant par Yverdon ? Quelles sont les villes traversées ? On admettra une vitesse de 120 km/h sur autoroute et de 70 km/h sur route.
- Mêmes questions mais en passant par Vevey ?
- Nous voulons faire des travaux d'entretien sur le réseau routier, mais dans un souci d'économie nous souhaitons réduire le coût de travaux au maximum. Toutes les villes devront être accessibles par une route rénovée. Quelles routes doivent être rénovées ? Quel sera le coût de la rénovation de ces routes ? Sachant que le coût pour rénover 1 km d'autoroute est de 15M CHF et de 7M CHF pour les routes.

Durée

- 8 périodes
- A rendre le dimanche **25.11.2018** à **23h55** au plus tard

Donnée

Vous trouverez les structures et exemples fournis sur la page Moodle du cours :
https://cyberlearn.hes-so.ch/course/view.php?id=11757

A faire :

`main.cpp` Implémentation des méthodes *PlusCourtChemin()*, *PlusRapideChemin()* et *ReseauLeMoinsCher()*. Nous vous fournissons la méthode *testShortestPath()* qui vous permettra de tester votre implémentation de *Dijkstra* (cf. point suivant).

`ShortestPath.h`
Vous devez implémenter la méthode *PathTo()* de la classe *ShortestPath*.
Vous devez implémenter la classe *DijkstraSP*, sous-classe de *ShortestPath*, permettant d'appliquer l'algorithme de *Dijkstra* à un graphe. Vous pouvez vous inspirer du pseudo code vu en cours, de l'implémentation de *Prim* ou alors implémenter vous-même une structure *IndexMinPQ*.

Attention ! Concernant les arcs du graphe orienté représentant votre réseau routier, vous ferez en sorte que pour chaque route reliant 2 villes A et B dans le réseau routier, vous ayez deux arcs (A -> B et B -> A) de manière à ce que ça soit bidirectionnel.

Wrappers : Vous devrez définir des *Wrappers* encapsulant un *RoadNetwork* et permettant d'appeler les algorithmes de chemin le plus court ou d'arbre couvrant minimum. Vos *Wrappers* incluront au minimum une fonction de coût et implémenteront les méthodes nécessaires (par ex. : *V()*, *forEachEdge(Func f)* ou *forEachAdjacentEdge(int v, Func f)*). Ils ne devront rien stocker de plus que la référence vers le réseau routier et la fonction de coût. Vous pouvez définir autant de *Wrappers* que nécessaires, soit avec la fonction de coût entièrement hard-codée à l'intérieur, soit avec un paramètre permettant de passer une fonction de coût au constructeur.

Voici comment les Wrappers seront utilisés :

```
RoadNetwork rn("reseau.txt");
RoadGraphWrapper rgw(rn);
auto mst = MinimumSpanningTree<RoadGraphWrapper>::Kruskal(rgw);

RoadDiGraphWrapper rdgw(rn);
DijkstraSP<RoadDiGraphWrapper> sp(rdgw, v);
```

Rendu/Evaluation

En plus de votre code, vous remettrez un **rapport** comportant au minimum une introduction, les réponses aux questions posées et une conclusion. Pour la dernière question vous mettrez en évidence le réseau routier réduit sur la carte. Merci de rendre votre travail sur *CyberLearn* dans un zip unique, dont le nom respecte les consignes.

Bonne chance !