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

18.10.2019

Introduction

Dans ce laboratoire nous appliquerons les algorithmes d'arbres couvrants minimaux et de plus court chemin à des graphes contenant des données réelles. Nous nous intéresserons aux grandes lignes du réseau ferroviaire de la Suisse, représentées sur la Figure 1.

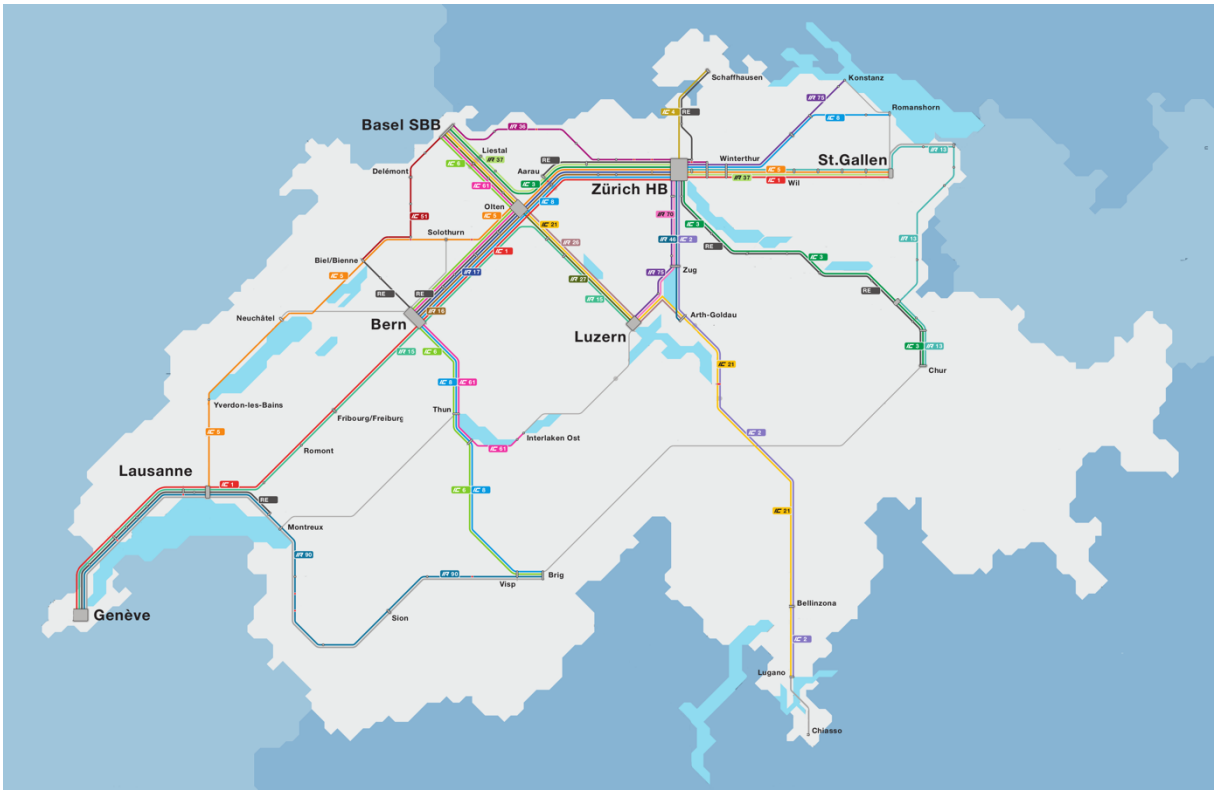

Figure 1 *- Carte des grandes lignes du réseau ferroviaire suisse. Les sommets du graphe associé correspondent aux villes et les arêtes correspondent aux lignes de chemin de fer les reliant. À chaque ligne sont associés trois poids : la distance en kilomètres séparant les deux villes, la durée moyenne du trajet en minutes, ainsi que le nombre de voies sur la ligne.*

Profs: Laura Elena Raileanu, Andrei Popescu-Belis, Jean-Claude Le Franc
Assistants: Antoine Rochat, Gabriel Luthier, Guillaume Hochet

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 :

Arbre recouvrant de poids minimum :

- Nous voulons faire des travaux d'entretien sur le réseau ferroviaire, mais dans un souci d'économie nous souhaitons réduire le coût des travaux au maximum. Toutes les villes devront être accessibles par une ligne rénovée. Chaque ligne possède un nombre de voies allant de 1 à 4. Le coût pour rénover 1 km de ligne de chemin de fer varie selon le nombre de voies :

    - 15M CHF par km pour les lignes ayant 4 voies
    - 10M CHF par km pour les lignes ayant 3 voies
    - 6M CHF par km pour les lignes ayant 2 voies
    - 3M CHF par km pour les lignes ayant 1 voie

  Quelles lignes doivent être rénovées ? Quel sera le coût total de la rénovation de ces lignes ?

Plus court chemin :

- Quel est le chemin le plus court entre Genève et Coire ? Quelles sont les villes traversées ?
- Mêmes questions mais en supposant que la gare de Sion est en travaux et donc qu'aucun train ne peut transiter par cette gare.
- Quel est le chemin le plus rapide entre Genève et Coire en passant par Brigue ? Quelles sont les villes traversées ?
- Mêmes questions mais entre Lausanne et Zurich, en passant par Bâle.

Durée

- 8 périodes
- A rendre le dimanche **17.11.2019** à **23h59** au plus tard

Donnée

Vous trouverez les structures et exemples fournis sur la page Moodle de votre classe d'ASD2. :

A faire :

`main.cpp` Implémentation des méthodes *ReseauLeMoinsCher(), PlusCourtChemin(), PlusCourtCheminAvecTravaux() et PlusRapideChemin()*. Il vous faudra utiliser un des

algorithmes de la classe *MinimumSpanningTree* pour trouver le réseau le moins cher et un des algorithmes de la classe *ShortestPath* pour les plus courts chemins.
Nous vous fournissons également la méthode *testShortestPath()* qui vous permettra de tester votre implémentation de *Dijkstra* (voir le 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 ferroviaire, vous ferez en sorte que pour chaque ligne reliant 2 villes A et B dans le réseau ferroviaire, vous ayez deux arcs (A -> B et B -> A) de manière à ce que le parcours soit toujours bidirectionnel.

Wrappers : Vous devrez définir des *Wrappers* encapsulant un *TrainNetwork* et permettant d'appeler les algorithmes de plus court chemin 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 ferroviaire et la fonction de coût. Vous pouvez définir autant de *Wrappers* que nécessaires, soit avec une fonction de coût définie explicitement dans la classe, soit avec un paramètre permettant de passer une fonction de coût au constructeur.

Voici comment les Wrappers seront utilisés :

```
TrainNetwork tn("reseau.txt");
TrainGraphWrapper tgw(tn);
auto mst = MinimumSpanningTree<TrainGraphWrapper>::Kruskal(tgw);

TrainDiGraphWrapper tdgw(tn);
DijkstraSP<TrainDiGraphWrapper> sp(tdgw, v);
```

Sortie

Nous vous fournissons un exemple de la sortie attendue afin de répondre aux différentes questions :

```
1. Quelles lignes doivent etre renovees ? Quel sera le cout de la renovation de ces
lignes ?

     ...

Lausanne - Montreux : 150 MF
Lausanne - Yverdon-les-Bains : 234 MF
Bale - Zurich : 264 MF

     ...

Coût Total: 7959 MF

2. Chemin le plus court entre Lausanne et Berne
  longueur = 97 km
  via Lausanne -> Romont -> Fribourg -> Berne

3. Chemin le plus court entre Lausanne et Berne, avec la gare de Romont en travaux
  longueur = 129 km
  via Lausanne -> Yverdon-les-Bains -> Neuchatel -> Berne

4. Chemin le plus rapide entre Lausanne et Berne en passant par Viege
  temps = 133 minutes
  via Lausanne -> Montreux -> Sion -> Viege -> Thoune -> Berne
```

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 ferroviaire réduit sur la carte (surligner dans un éditeur d'images). Merci de rendre votre travail sur *CyberLearn* dans une archive zip unique, dont le nom respecte les consignes habituelles.

Bonne chance !