M1 - Structures de Données
Projet : Réseau d'Aqueducs

La France a décidé de repenser son réseau d'aqueducs pour rendre l'approvisionnement des communes en eau plus efficace. Le chef du projet vous charge de calculer le réseau le plus court reliant toutes les villes, ainsi que sa taille.

Pour ce faire, vous calculerez l'arbre couvrant de poids minimal sur un graphe dont les nœuds sont les villes ; les arêtes sont les connexions entre les villes, et le poids associé à chaque arête est la distance entre les villes en kilomètre.

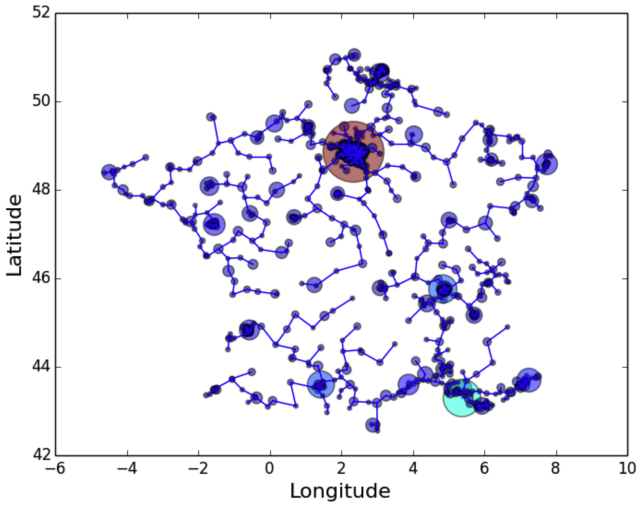

FIGURE 1 – Exemple du réseau le plus court reliant les villes de France. La taille de chaque bulle correspond à la population.

1 Prise en main

Télécharger l'archive `2020-m1-projet.zip` disponible sur la page du cours. Ce fichier contient :

- Une base de données [1] avec les villes françaises de métropole et leurs caractéristiques. (`citiesList.csv`).
- Une fonction C pour lire les latitudes/longitudes (en degrés) des villes françaises (de n habitants ou plus) à partir la base de données, et pour écrire ces coordonnées dans un autre fichier pour la visualisation. (`citiesReader.h` et `citiesReader.c`).
- Un programme de test en C (`main.c`) qui :
  - demande à l'utilisateur un seuil de population n ;
  - utilise la fonction `citiesReader(n);` pour filtre charger les villes dont la population est supérieur à n et stocke le résultat dans un fichier. Chaque ligne du fichier représente une ville (population, longitude, latitude) ; et

1. Le projet est adapté de A. Loseille et A. Modave. Les données sont librement disponibles à l'adresse `http://sql.sh/736-base-donnees-villes-francaises`

— génère un graphe *complet* entre ces villes et le stocke dans un fichier.

— Un script Python pour générer des images avec la position des villes et le graphe obtenu à partir des fichiers générés par le programme précédent. (`visualisation.py`)

— Les graphes optimaux (les réponses) pour les villes de tailles supérieurs à 250,000, 100,000 et 50,000 habitants. (dans le répertoire `solutions`). Les fichier `length*.dat` contiennent les villes et dont la dernière ligne contient la longueur du réseau optimal ; alors que chaque linge du fichier `graphe*.dat` correspondant représente une connexion entre deux villes dans le réseau optimal. Comparez vos résultats avec ces fichiers pour tester votre implémentation.

Afin de pourvoir compiler le programme de test il vous faut un compilateur du langage C comme `gcc`. Pour pouvoir visualiser les graphes, il vous faut Python avec la libraire `matplotlib` :

```
sudo apt install python-pip python-tk
pip install matplotlib
pip install networkx
pip install --upgrade networkx
```

Une fois les les pré-requis sont installés. Examiner les fichiers fournis pour comprendre leur fonctionnement. Ensuite compiler et lancer les programmes :

```
cmd> gcc -o graphe main.c citiesReader.c

cmd> ./graphe
Minimal population? 250000
== Reading cities with population >= 250000 from 'citiesList.csv' ==
== Writing cities with population >= 250000 in 'resuCities.dat' ==
NICE 343304 7.250000 43.700001
MARSEILLE 850726 5.376390 43.296700
TOULOUSE 441802 1.433330 43.599998
MONTPELLIER 257351 3.883330 43.599998
NANTES 284970 -1.550000 47.216702
STRASBOURG 271782 7.750000 48.583302
LYON 484344 4.841390 45.758900
PARIS 2243833 2.344450 48.860001

cmd> python visualisation.py
```

Le script de visualisation génère l'image ci-dessous :

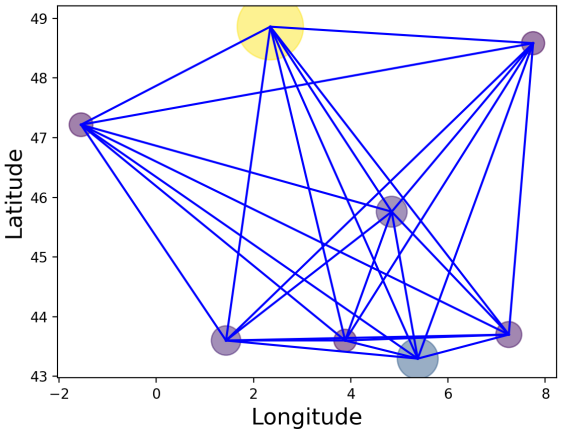

FIGURE 2 – Graphe complet entre les villes françaises de plus de 250,000 habitants.

2 Calcul du réseau optimal

Le calcul du réseau optimal s'agit de sélectionner un sous-ensemble des arêtes du graphe complet, qui feront partie du réseau. On pourrait sélectionner l'ensemble du graphe, mais cela reviendrait à installer beaucoup d'aqueducs inutiles ; or, pour optimiser le coût, il faut réduire la longueur totale du réseau au maximum. Une solution optimale est nécessairement un arbre : en effet, si un ensemble

d'arêtes contient un cycle, alors on peut supprimer l'une des arêtes tout en préservant la connexité de l'ensemble.

La distance entre deux villes (a et b) est calculée avec la formule :

$$R \cdot \mathrm{acos}\left(\sin(\mathrm{lat}_a)\ \sin(\mathrm{lat}_b) + \cos(\mathrm{lat}_a - \mathrm{lat}_b)\ \cos(\mathrm{lat}_a)\ \cos(\mathrm{lat}_b)\right)$$

avec $R \approx 6371$ km.

Q2.1 Écrire un programme à l'instar de celui du test fourni calculant un réseau optimal. Le réseau optimal n'est pas nécessairement unique. A titre d'exemple, la Figure 6 montre le réseau optimal (le plus cours) reliant les villes de la figure 6. Sa longueur est 1,850.230748 km.

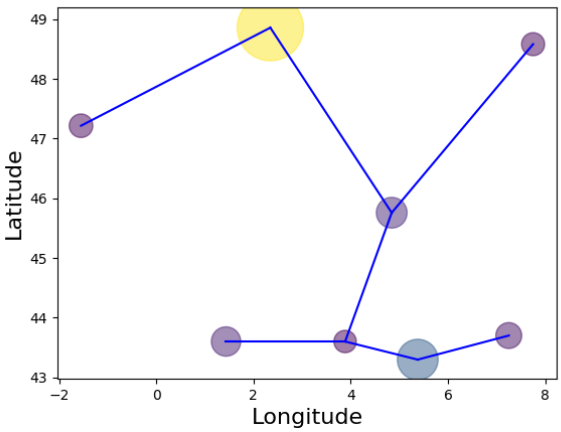

FIGURE 3 – Graphe optimal entre les villes françaises de plus de 250,000 habitants.

Q2.2 Tester votre programme sur des sous-ensembles des villes ayant plus de 250,000, 100,000, 50,000, 10,000 et 1,000 habitants, ainsi que toutes les villes. Pour chacun des test effectué, donner

- la longueur du réseau optimal (il est obtenu additionnant le poids des arêtes du graphe final ;
- le temps de calcul ;
- une figure du graphe généré par le script de visualisation.

Vous pouvez comparer votre solution avec l'un des exemple fourni :

```
cmd> ./graphe 200000
cmd> bash -c 'diff <(sort resuGraph.dat) <(sort ../solutions/graphe200000.dat)'
```

Comparez également les figures obtenues :

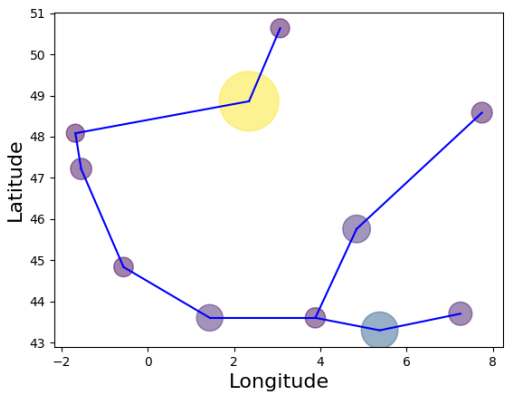

FIGURE 4 – Taille des villes $\geq 200,000$ habitants.

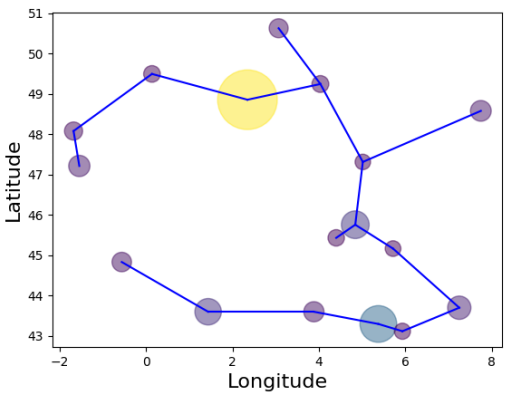

FIGURE 5 – Taille des villes $\geq 150,000$ habitants.

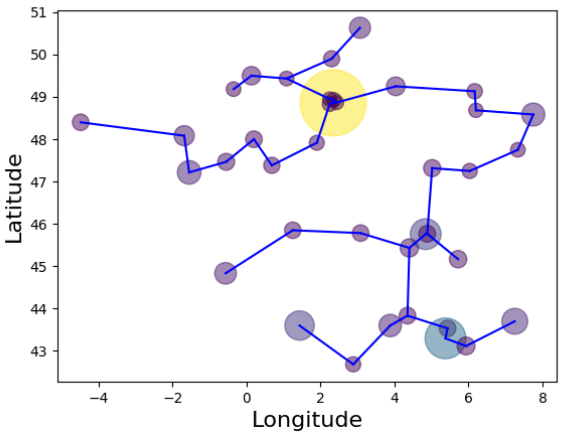

FIGURE 6 – Taille des villes $\geq 100,000$ habitants.

Q2.3 Quelles est la complexité de votre algorithme ? Expérimentalement, votre algorithme semble t'il souvent être dans le pire des cas ?

Q2.4 Quelles structures de données avez-vous utilisé ? Justifiez vos choix par une analyse théorique et expérimentale des différentes solutions possibles.

3 Consignes

- Vous travaillerez avec le langage de programmation de votre choix ;
- Le projet est à faire en groupes de 3 personnes maximum ;
- La date limite pour rendre votre projet est le **13 décembre 2020**, la soutenance aura lieu pendant la dernière séance de TP ;
- Les rendu est composé de :
  - Les fichiers sources de votre implémentation
  - Un fichier `Readme` contenant des explications pour compiler et tester votre programme
  - Un rapport de maximum 5 pages expliquant vos choix d'implémentation ainsi qu'une analyse théorique et empirique de la complexité de vos algorithmes.
- Chaque groupe présentera son projet lors d'une soutenance de 10min

Avoir un code fonctionnel calculant une solution optimale n'est pas l'objectif final de ce projet. C'est un pré-requis, mais c'est insuffisant. Nous attendons de vous d'être capable de justifier vos choix : le ou les algorithmes implémentés, les structures de données utilisées. Votre argumentation devra impérativement s'appuyer sur des résultats expérimentaux ET théoriques. Vous devez donc concevoir des expériences permettant de valider vos propos et vous renseigner sur la complexité des algorithmes que vous choisirez. Posez vous également des questions sur les propriétés des graphes que nous vous avons fourni. Ces propriétés ont elles une influence sur l'efficacité de l'algorithme ? Dans cette perspective, utiliser un code open source que vous aurez trouvé sur internet n'est pas interdit. Il faut en revanche impérativement citer vos sources (sous peine d'être considéré comme de la fraude). Un code que vous aurez fait vous même vaudra bien sûr plus de points qu'un code trouvé sur internet.