Projet Structures de données : avancé

1 Objectif

Les fichiers cities.txt et roads.txt contiennent des données sur les principales routes en Europe. Ces données constituent un graphe non dirigé où les sommets représentent des villes (par exemple : Brussels), et il existe un arc entre deux villes s'il y a une route les reliant. L'objectif global du projet est d'implémenter un programme Java capable de calculer des itinéraires entre deux villes. Voici un exemple d'itinéraire :

```
Itinéraire de Berlin à Madrid: 10 routes et 2435.89502691291 km
Berlin -> Hamburg (258,32 km)
Hamburg -> Bremen (93,08 km)
Bremen -> Groningen (150,17 km)
Groningen -> Amsterdam (146,76 km)
Amsterdam -> Brussels (172,89 km)
Brussels -> Paris (264,83 km)
Paris -> Orleans (110,60 km)
Orleans -> Toulouse (479,30 km)
Toulouse -> Barcelona (254,23 km)
Barcelona -> Madrid (505,72 km)
```

Cet exemple représente un itinéraire de Berlin à Madrid, composé de 10 routes pour une longueur totale d'environ 2436 km.

2 Sur Moodle

Pour mener à bien votre projet, nous vous fournissons plusieurs fichiers :

- `cities.txt` contient des informations sur différentes villes, avec une ligne par ville. Pour chaque ville, les informations suivantes sont séparées par des virgules dans cet ordre : un identifiant unique, son nom, sa longitude et sa latitude.
- `roads.txt` contient des informations à propos des routes. Il contient une ligne par route. Pour chaque route, l'identifiant des deux extrémités est séparé par une virgule. **Attention, ce graphe est non dirigé. Les routes peuvent être parcourues dans les deux sens.**
- la classe `Util.java` possédant une méthode statique permettant de calculer la distance entre deux coordonnées géographiques. Une coordonnée géographique est composée d'une longitude et d'une latitude.
- une classe `Main.java` que vous ne pouvez pas modifier. Le code de cette classe est présenté ci-dessous. La classe à construire `Graph` devra avoir un constructeur qui prendra les deux fichiers contenant les informations des villes et des routes. Elle doit également contenir deux méthodes pour calculer un itinéraire. Ces deux méthodes prennent deux paramètres : les noms des villes de départ et d'arrivée.

```java
import java.io.File;

public class Main {
	public static void main(String[] args) {
		File cities = new File("cities.txt");
		File roads = new File("roads.txt");
		Graph graph = new Graph(cities, roads);
		graph.calculerItineraireMinimisantNombreRoutes("Berlin", "Madrid");
		System.out.println("--------------------------");
		graph.calculerItineraireMinimisantKm("Berlin", "Madrid");
	}
}
```

3 Tâches à effectuer

Nous vous demandons de rendre la classe `Main` fonctionnelle en réalisant les deux tâches suivantes :

- Implémenter la méthode `calculerItineraireMinimisantNombreRoutes` qui calcule l'itinéraire entre deux villes avec le moins de routes possibles (voir point 4). S'il est impossible d'aller d'une ville à une autre, votre programme lancera une exception.
- Implémenter la méthode `calculerItineraireMinimisantKm` qui calcule l'itinéraire entre deux villes avec le moins de kilomètres (voir point 5). S'il est impossible d'aller d'une ville à une autre, votre programme lancera une exception.

4 Itinéraire avec le moins de tronçons

La méthode `calculerItineraireMinimisantNombreRoutes` calculera et affichera à la console les itinéraires qui minimisent le nombre de routes.
Par exemple, pour aller de Berlin à Madrid, l'itinéraire le plus court contient 10 routes.

```
Trajet de Berlin à Madrid: 10 routes et 2435.89502691291 kms
Berlin -> Hamburg (258,32 km)
Hamburg -> Bremen (93,08 km)
Bremen -> Groningen (150,17 km)
Groningen -> Amsterdam (146,76 km)
Amsterdam -> Brussels (172,89 km)
Brussels -> Paris (264,83 km)
Paris -> Orleans (110,60 km)
Orleans -> Toulouse (479,30 km)
Toulouse -> Barcelona (254,23 km)
Barcelona -> Madrid (505,72 km)
```

Il est peut-être possible de trouver d'autres itinéraires avec 10 routes.

5 Itinéraire les plus courts en kilomètre

La méthode `calculerItineraireMinimisantKm` calculera et affichera à la console les itinéraires les plus courts en kilomètres.
Pour aller de Berlin à Madrid, il est possible de trouver un itinéraire de 2071,79 km.

```
Trajet de Berlin à Madrid: 18 routes et 2071.789315586917 kms
Berlin -> Leipzig (148,14 km)
Leipzig -> Gera (55,49 km)
Gera -> Erfurt (74,50 km)
Erfurt -> Eisenach (49,67 km)
Eisenach -> Bad Hersfeld (44,54 km)
```

```
Bad Hersfeld -> GieBen (79,16 km)
GieBen -> Wetzlar (12,36 km)
Wetzlar -> Koblenz (68,33 km)
Koblenz -> Trier (94,66 km)
Trier -> Luxembourg (40,52 km)
Luxembourg -> Charleville-Mezieres (102,95 km)
Charleville-Mezieres -> Reims (75,91 km)
Reims -> Paris (130,65 km)
Paris -> Orleans (110,60 km)
Orleans -> Bordeaux (390,26 km)
Bordeaux -> San Sebastian (202,71 km)
San Sebastian -> Burgos (177,06 km)
Burgos -> Madrid (214,30 km)
```

6 Organisation et livrables

Ce projet se déroule du 26 février au 24 mars par groupe de trois étudiants maximums. Les étudiants d'un même groupe doivent être dans la même série. La constitution des groupes se fera lors de la première séance. Les absents à la première séance ne pourront pas participer au projet (sauf certificat médical)

Le projet est à remettre via « Moodle » pour le dimanche 24 mars 2022 à 20h00. Nous ne demandons pas de rapport. Si certaines choses méritent des explications, vous pouvez les fournir en commentaire dans les différents fichiers.

Le plagiat sera lourdement sanctionné (au minimum 0 à l'UE). Tous les projets seront automatiquement testés par un logiciel de détection de plagiat.

Le but du projet est d'approfondir la compréhension des concepts relatifs aux graphes ainsi que des algorithmes de parcours associés. Il est important de souligner que l'utilisation de logiciels de type ChatGPT peut fournir des solutions, mais cela contourne l'objectif d'apprentissage visé. De plus, les implémentations proposées par ChatGPT peuvent manquer d'optimalité en termes de complexité algorithmique et de gestion de la redondance de code.