DANAN Emma
SAKOUN Noa
PréIng 2 MEF1

Projet CY-Trucks

Répartition des tâches

Nom	Date de début	Date de fin	Responsable
Réunion découverte sujet	27/10/2023	27/10/2023	Emma/Noa
Réunion répartition des tâches	01/12/2023	01/12/2023	Emma/Noa
Partie Shell	06/12/2023	27/01/2024	Emma/Noa
Partie C	13/12/2023	31/01/2024	Emma/Noa
Gnuplot	06/01/2024	27/01/2024	Noa
Rédaction pdf	13/01/2024	31/01/2024	Noa

Détail des tâches :

- Partie Shell :

Vérification présence de l'exécutable : Noa
Vérification/Création/Vidage dossiers temp/images : Noa
Écriture help.txt : Emma
Codage traitement -d1 : Emma/Noa
Codage traitement -d2 : Emma/Noa
Codage traitement -l : Emma/Noa
Temps d'exécution pour chaque traitement : Noa

- Partie C :

Récupération des données du fichier data.csv dans le main : Noa
Reste du main : Noa
Écriture des fonctions de base nécessaires à l'AVL (rotation, équilibre…) = fichier fonctionsAVL : Emma
Écriture des fonctions spécifiques au tri du traitement T + récupération des données dans un fichier de sortie = fichier fonctionsT : Noa
Écriture des fonctions spécifiques au tri du traitement S + récupération des données dans un fichier de sortie = fichier fonctionsS : Noa

- Partie Gnuplot :

Graphique -d1 : Noa
Graphique -d2 : Noa
Graphique -l : Noa
Graphique -t : Noa
Graphique -s : Noa

- Écriture du makefile : Emma/Noa

- Écriture du readme : Noa

- Rendu du projet sur Github : Noa

Descriptif du code

- Traitement -d1

Donne les 10 conducteurs avec le plus de trajets trié par ordre décroissant.
Après avoir testé sur les ordinateurs les plus puissants que nous avons, le temps d'exécution de ce traitement est de 11 secondes.

```
Choisir un traitement (-d1,-d2,-l,-t,-s,-h) : -d1
L'exécutable projet existe déjà.
Le dossier temporaire existe. Vidage en cours...
Temps d'exécution du traitement d1 : 11 secondes
```

Pour le graphique, on a d'abord un histogramme vertical avec les axes tournés (traitement_d1.png) puis nous avons dû faire une rotation de l'image à l'aide du logiciel ImageMagick, on obtient donc le graphique traitement_d1_rotated.png.
(Voir annexe pour avoir un exemple du graphique obtenu).

- Traitement -d2

Donne les 10 conducteurs ayant parcouru la plus longue distance trié par ordre décroissant.
Après avoir testé sur les ordinateurs les plus puissants que nous avons, le temps d'exécution de ce traitement est de 9 secondes.

```
Choisir un traitement (-d1,-d2,-l,-t,-s,-h) : -d2
L'exécutable projet existe déjà.
Le dossier temporaire existe. Vidage en cours...
Temps d'exécution du traitement d2 : 9 secondes
```

Pour le graphique, on a d'abord un histogramme vertical avec les axes tournés (traitement_d2.png) puis nous avons dû faire une rotation de l'image à l'aide du logiciel ImageMagick, on obtient donc le graphique traitement_d2_rotated.png.
(Voir annexe pour avoir un exemple du graphique obtenu).

- Traitement -l

Donne les 10 numéros d'identifiant des trajets qui ont la plus longue distance trié par ordre décroissant.
Après avoir testé sur les ordinateurs les plus puissants que nous avons, le temps d'exécution de ce traitement est de 10 secondes.

```
Choisir un traitement (-d1,-d2,-l,-t,-s,-h) : -l
L'exécutable projet existe déjà.
Le dossier temporaire existe. Vidage en cours...
Temps d'exécution du traitement l : 10 secondes
```

Pour le graphique, on a un histogramme vertical traitement_l.png.
(Voir annexe pour avoir un exemple du graphique obtenu).

- Traitement -t

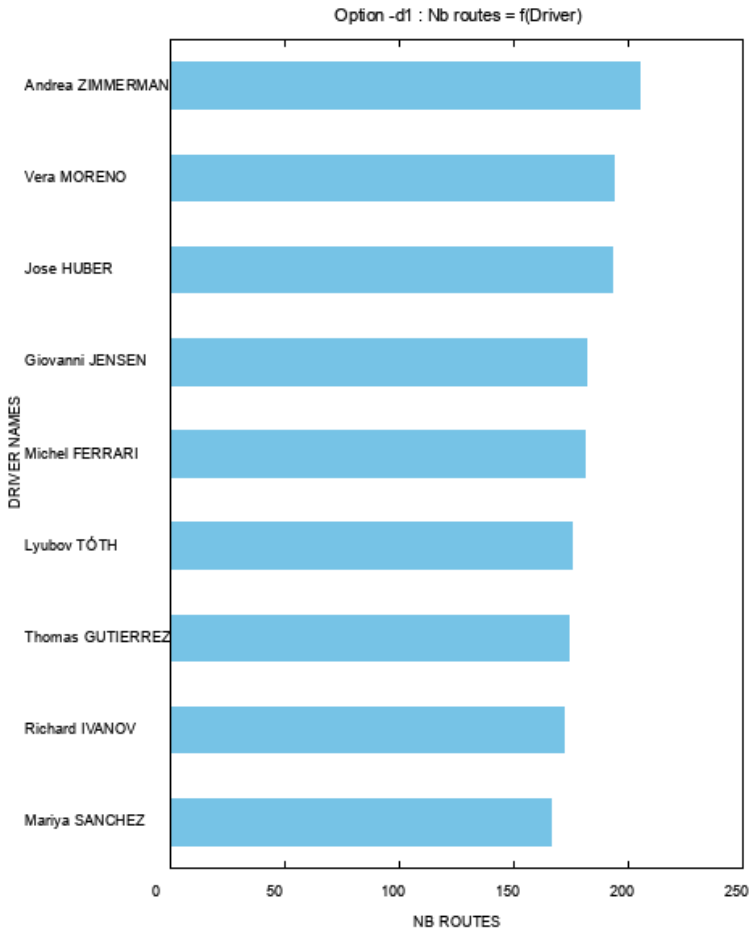
Donne les 10 villes avec le plus de trajets trié par ordre alphabétique.
L'entièreté du code pour le tri a été fait en C.
Pour le graphique, le code a été fait dans la partie script shell en utilisant le fichier de sortie créé par le programme C (traitement_t.png).
(Voir annexe pour avoir un exemple du graphique obtenu).

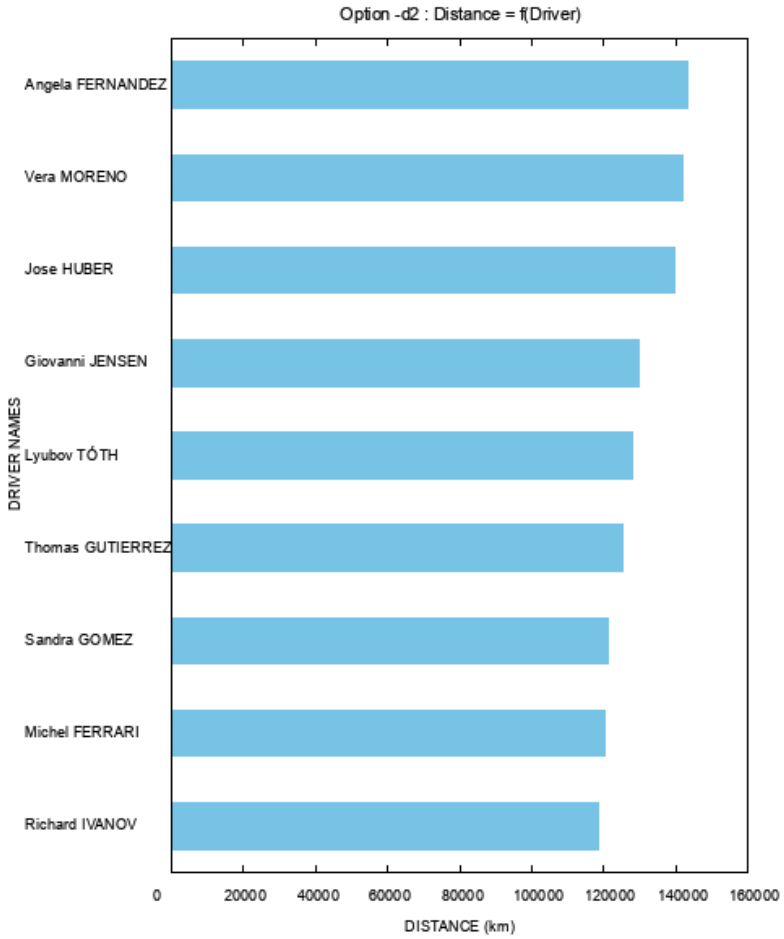
- Traitement -s

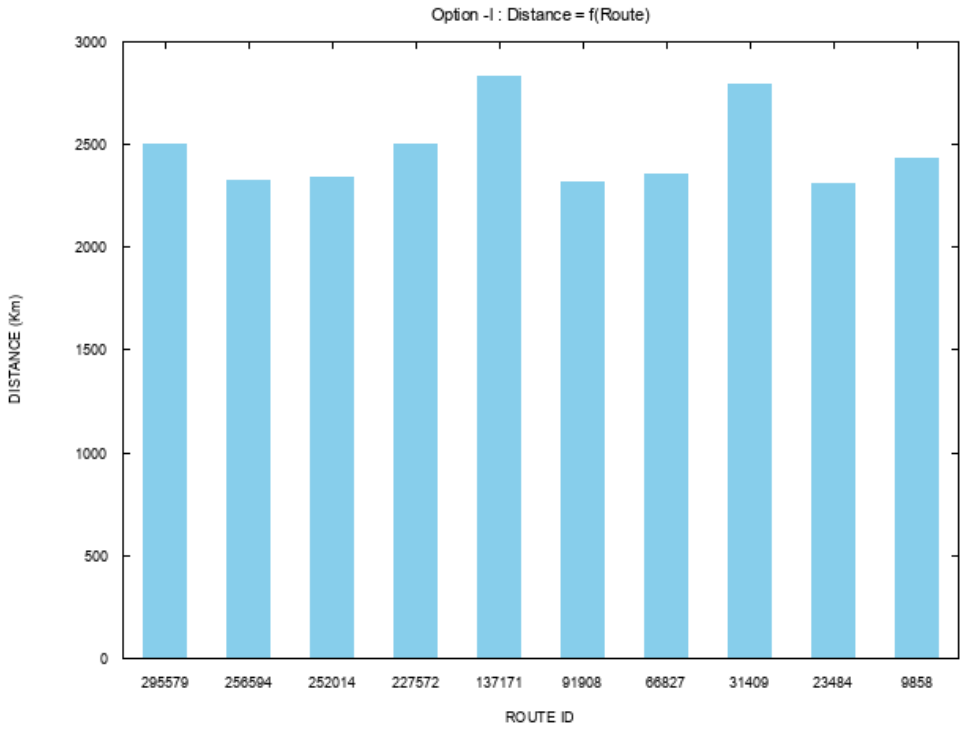
Donne les 50 numéros d'identifiant des trajets qui ont la plus grande différence de distance max-min trié par ordre décroissant.
L'entièreté du code pour le tri a été fait en C. Nous avons rencontré quelques problèmes lors de la compilation, cela dépend des logiciels utilisés. Sur les ordinateurs de l'école le code présente des erreurs de segmentation alors que sur des logiciels de type Visual Studio Code, Xcode et VirtualBox le code fonctionne parfaitement.
Pour le graphique, le code a été fait dans la partie script shell en utilisant le fichier de sortie créé par le programme C (traitement_s.png).
(Voir annexe pour avoir un exemple du graphique obtenu).

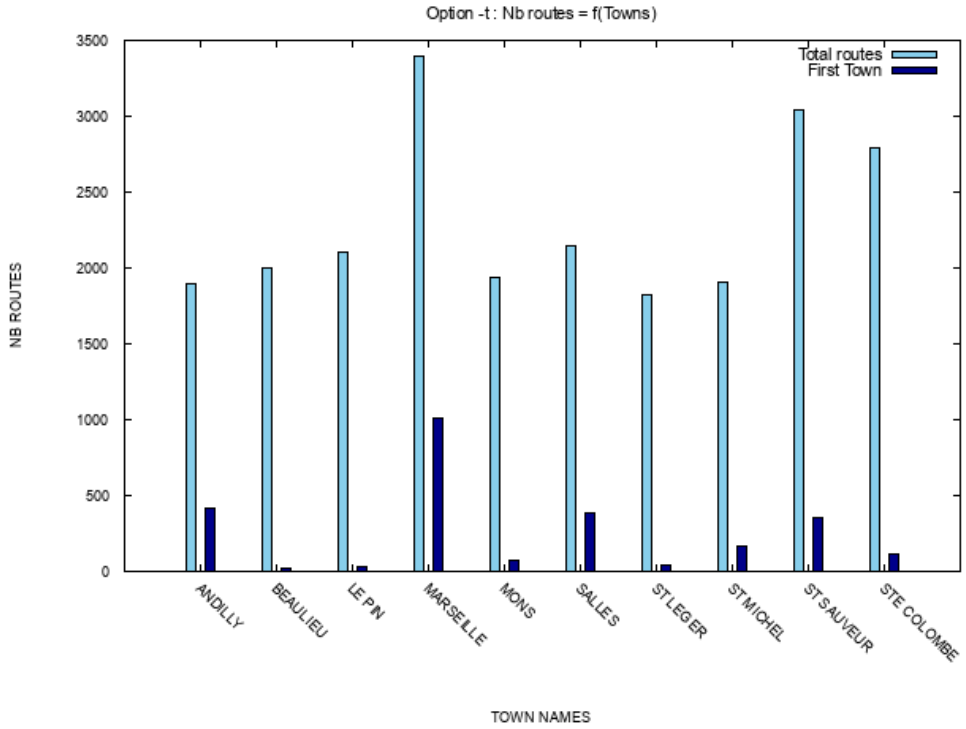
Le fichier « demo » vous permettra de voir les résultats obtenus pour chaque traitement avec le tri effectué (.txt) ainsi que le graphique généré (.png).

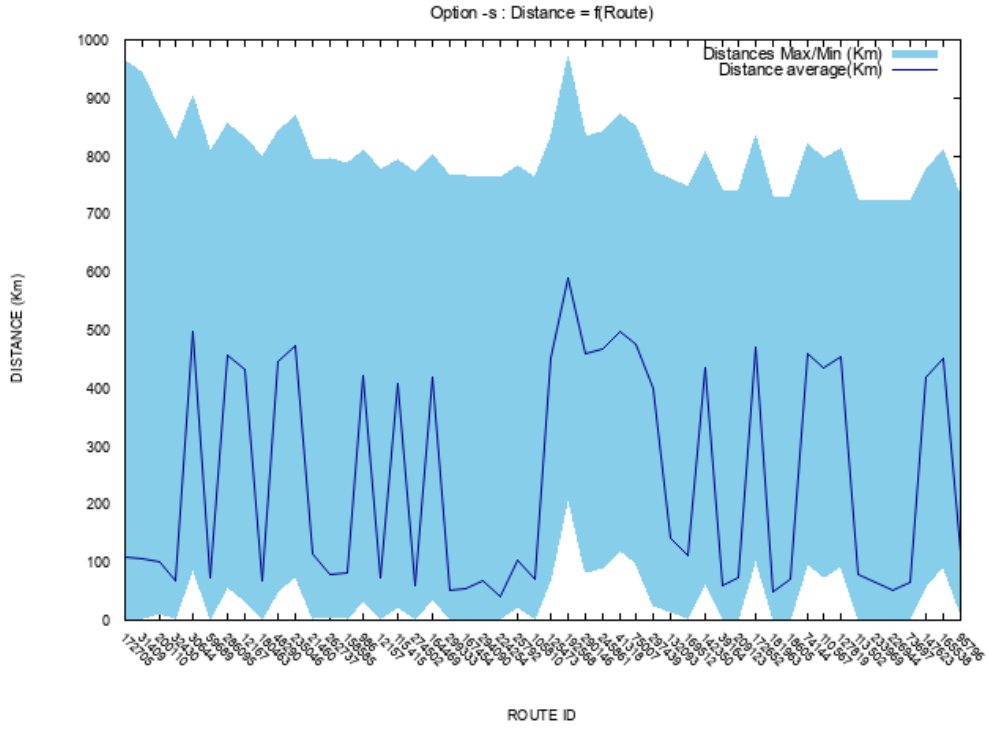
Annexes


Option -d1 : Nb routes = f(Driver)
Andrea ZIMMERMAN
Vera MORENO
Jose HUBER
Giovanni JENSEN
Michel FERRARI
Lyubov TÓTH
Thomas GUTIERREZ
Richard IVANOV
Mariya SANCHEZ
DRIVER NAMES
0
50
100
150
200
250
NB ROUTES


Option -d2 : Distance = f(Driver)
Angela FERNANDEZ
Vera MORENO
Jose HUBER
Giovanni JENSEN
Lyubov TÓTH
Thomas GUTIERREZ
Sandra GOMEZ
Michel FERRARI
Richard IVANOV
DRIVER NAMES
0
20000
40000
60000
80000
100000
120000
140000
160000
DISTANCE (km)


Option -l : Distance = f(Route)
DISTANCE (Km)
ROUTE ID
295579
256594
252014
227572
137171
91908
66827
31409
23484
9858


Option -t : Nb routes = f(Towns)
Total routes
First Town
NB ROUTES
TOWN NAMES
ANDILLY
BEAULIEU
LE PIN
MARSELLE
MONS
SALLES
ST LEGER
ST MICHEL
ST SAUVEUR
STE COLOMBE


Option -s : Distance = f(Route)
Distances Max/Min (Km)
Distance average(Km)
DISTANCE (Km)
ROUTE ID