Livrable 1 - P2I2

PPII2 - Rapport

DIONISIO, FREY, BILLARD, MEZUREUX

1ère année

TELECOM Nancy

Ingénieurs du numérique • Inspiring your digital future

24 mai 2023

Table des matières

◁ Chapitre 1 ▷
Introduction

I ▷ Contexte et objectifs du projet

Ce projet s'inscrit dans le cadre pédagogique de notre première année d'étude en école d'ingénieur au sein de l'établissement TELECOM Nancy. L'objectif y est de mettre en pratique les compétences scientifiques et techniques acquises tout au long de cette première année en se rapprochant d'une étude de cas concrète.

Il se divise en deux étapes distinctes. Il nous a été demandé dans un premier temps de construire en langage «C» un programme qui permet de planifier un parcours de charge pour un véhicule électrique et pour un trajet donné et d'enrichir cette application en y incorporant les ajouts de notre choix. Dans un second temps, il nous a fallu réaliser, toujours en langage «C», un module de simulation qui pour un ensemble d'usagers, calcule le taux de charge des bornes du territoire.

Notons également que tout au long du projet nous avons dû suivre une gestion de projets rigoureuse, ce qui nous a permis d'assurer une planification efficace et un suivi approprié des activités. Nous étions obligé d'adopter une approche méthodologique solide, afin d'assurer que notre projet soit réalisé de manière efficace, dans les délais impartis.

Toute l'équipe a pris plaisir à travailler sur le projet et nous tenons à remercier tous ceux qui nous ont aidés.

II ▷ Objectifs du rapport

Ce rapport synthétise le travail réalisé par l'équipe de PoinCarWash pour répondre à la problématique posée. À savoir travailler à l'élaboration d'outils utiles au développement d'une mobilité plus écologique.

Le présent rapport fait état de la conception et de l'implémentation de notre travail mais également des performances et des tests sur ce dernier, ainsi que d'une présentation de notre gestion de projet.

III ▷ Présentation du plan

Au cours de ce rapport, nous aborderons dans un premier temps les éléments de gestion de projet communs aux deux étapes. Cette première partie nous permettra ainsi de justifier nos choix d'approches méthodologiques pour la réalisation de ce projet.

Nous procéderons ensuite de manière similaire pour les deux étapes du projet. En premier lieu, nous décrirons les éléments de gestion de projet spécifiques à chacune d'entre elles. Nous détaillerons ensuite les phases de conception et de développement avant d'enfin présenter les performances et les tests de nos applications.

Pour conclure nous récapitulerons les résultats obtenus dans chaque partie du projet en les mettant en parallèle avec les objectifs fixés au départ, nous analyserons les avantages et les limites de notre travail, et formulerons des recommandations pour de futures améliorations et développements avant de conclure définitivement le projet.

◁ Chapitre 2 ▷
Gestion de projet

I ▷ Définition du projet

I.A Contexte et justification du projet

Avec la prise de conscience écologique grandissante dans la population, les citoyens et les Etats se tournent progressivement vers des modes de consommation et des politiques plus responsables et écologiques. La Commission européenne a dans cette optique récemment pris la décision d'interdire la vente de véhicules thermiques à l'horizon 2035. Le développement et l'utilisation des véhicules électriques va donc connaître une croissance élevée dans les années à venir. En effet, selon les premières projections[1], le marché des véhicules électirifiés pourrait représenter 24% des parts de marché d'ici 2025 et 90% d'ici 2040.

Pourtant, voyager dans ce type de véhicules peut encore parfois s'avérer difficile, notamment en ce qui concerne leur autonomie toujours relativement limitée et la répartition des bornes de recharge sur le territoire, cette dernière étant encore très inégale sur le sol français. Il est donc nécessaire pour faciliter leur adoption d'en faciliter et d'en optimiser l'utilisation.

La première étape de notre projet consiste alors en la conception d'un algorithme permettant de calculer pour un utilisateur le trajet le plus court entre deux endroits situés en France, en optimisant ses passages aux bornes de recharges. Cet outil pourra aider les conducteurs à naviguer plus efficacement et à ainsi surmonter les inquiétudes liées à l'autonomie limitée des voitures électriques.

La seconde étape du projet est basée sur l'étude du remplissage des stations en fonction d'un ensemble de trajets donnés. L'acquisition de ces statistiques pourrait par exemple permettre de planifier plus efficacement l'expansion future du réseau de recharge.

Ainsi nous espérons au travers de ce projet promouvoir une mobilité plus durable en participant à l'élaboration de solutions pratiques pour surmonter les obstacles liés à l'autonomie et à la disponibilité de stations de recharge pour véhicules électriques.

I.B Portée du projet

Objectifs du projet :

- ▷ Conception d'un algorithme de calcul d'itinéraire optimisé pour véhicule électrique
- ▷ Conception d'une application d'analyse du remplissage des stations de recharge

Ressources :

- ▷ Groupe de projet (4 personnes) : temps de travail variable pendant 9 semaines
- ▷ Equipe pédagogique de TELECOM Nancy : aide ponctuelle
- ▷ Bases de données relatives aux stations de recharge et aux véhicules électriques

1. Selon l'étude BloombergNEF

Livrables :

- Code source des deux applications, **24 Mai**
- Guide utilisateur décrivant l'installation du programme et les modalités de son utilisation, **24 Mai**
- Rapport du projet incluant les éléments relatifs à la gestion de projet et décrivant les choix motivés des structures de données et des algorithmes (sur la base d'un état de l'art détaillé), les fonctions réalisées et leur analyse (complexité mémoire, temps), les fonctions réalisées et les tests mis en œuvre, **24 Mai**

Éléments non prévus dans le projet :

- Adaptation du calcul des itinéraires en fonction du taux de remplissage des stations de recharges.

I.C Parties prenantes

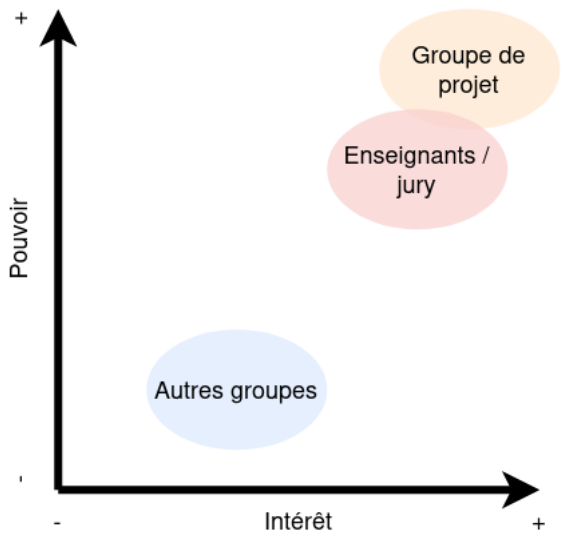
Identification :

- Interne :
  - Groupe de projet : Les participants directs au projet, responsables de la conception et du développement des outils demandés.
- Externe :
  - Enseignants et jury : Ils guident et évaluent les étudiants dans le cadre du projet.
  - Autres groupes : D'autres groupes de projet travaillant sur des sujets similaires, avec lesquels des échanges et des discussions peuvent être initié pour partager des idées et des connaissances.

Evaluation de l'influence et de l'importance :

- Groupe de projet : Influence élevée et importance élevée, car ils sont directement responsables de la réalisation du projet et de l'atteinte des objectifs fixés.
- Enseignants/jury : Influence élevée et importance élevée, car ils évalueront le projet et fourniront des conseils et des orientations pour assurer la qualité et la réussite du projet.
- Autres groupes de projet : Influence faible à moyenne et importance faible à moyenne, car ils peuvent partager des connaissances et des expériences similaires, mais leur impact direct sur le projet est limité.

On peut synthétiser cette analyse au travers de ce diagramme :



Plan d'action :

- Groupe de projet :
  - Communiquer de façon régulière et transparente au sein du groupe.
  - Organiser des réunions pour discuter des progrès réalisés, des difficultés rencontrées et des décisions à prendre.
- Enseignants/jury :
  - Communiquer de façon ponctuelle pour s'assurer que nous travaillons toujours dans la bonne direction.
  - Solliciter leur aide en cas de trop grosse difficulté.
- Autres groupes de projet :
  - Echanger de façon ponctuelle pour recueillir des idées, des connaissances ou des expériences supplémentaires.

I.D Contraintes et risques

Contraintes :

- Temporelle :
  - Le projet doit être réalisé en 9 semaines.
- Techniques :
  - Langage de programmation : le langage utilisé doit être le langage «C».

Risques :

- Techniques :
  - Incompatibilités matérielles : nous ne travaillons pas tous sous le même OS.
  - Problèmes de performance : les outils demandés peuvent demander des traitements lourds, nos machines pourraient être dépassées.
- Gestion de projet :
  - Gestion du temps et planification : nous ne travaillons pas à temps plein sur le projet et nos autres activités peuvent nous empêcher de consacrer suffisament de temps au projet.
- Liés aux données :
  - Les données utilisées pour le projet peuvent être incomplètes, inexactes ou indisponibles.

Evaluation :

- Contrainte temporelle :
  - Impact : le temps disponible pour la conception, le développement et les tests est limité.
  - Gravité : élevée, un manque de temps peut affecter la qualité globale du projet.
- Contrainte technique :
  - Impact : le langage «C» est maîtrisé par l'équipe, la contrainte n'a donc que peu d'impact.
  - Gravité : Négligeable, cette contraintes peut être gérée en planifiant un temps de formation mais cela demande du temps.

- Risques techniques :
  - Impact : les incompatibilités matérielles peuvent entraîner des problèmes d'intégration et de compatibilité lors de la mise en œuvre. Les problèmes de performance peuvent affecter la réactivité de l'application et la satisfaction de l'utilisateur.
  - Gravité : Modérée, ces risques peuvent être atténués en identifiant les incompatibilités dès le début du projet et en optimisant le code pour améliorer les performances.
- Risque de gestion de projet :
  - Impact : Une mauvaise gestion du temps et de la planification peut entraîner des retards dans l'achèvement des tâches et la réalisation à terme des objectifs du projet.
  - Gravité : Modérée, une bonne communication et une planification efficace peuvent minimiser les impacts de ce risque.
- Risque lié aux données :
  - Impact : Des difficultés pour obtenir les données peuvent allonger le temps nécessaire à l'élaboration des applications, de plus des données de mauvaise qualité peuvent affecter la précision et la validité des résultats obtenus.
  - Gravité : Modérée, des efforts de collecte et de validation des données seront nécessaires pour minimiser l'impact de ce risque.

On peut synthétiser les résultats de cette analyse dans une matrice des risques :

Matrice des risques

		Gravité				
		Négligeable	Mineure	Modérée	Majeure	Catastrophique
Probabilité	Très probable					
	Probable			- Incompatibilités entre les OS (risque technique) - Difficultés d'acquisition des données (risque lié au données)	Manque de temps / mauvaise gestion du temps (contrainte temporelle et risque de gestion de projet)	
	Possible			Capacité de calcul trop limitée (risque technique)		
	Peu probable			Données de mauvaise qualité (risque lié au données)		
	Très peu probable			Non maîtrise du langage "C" (contrainte technique)		

II ▷ Approche et méthodologie de gestion de projet

II.A Méthode de gestion de projet

Pour ce projet nous avons décidé de nous orienter vers une gestion de projet classique pour les raisons suivantes :

- **La précision des spécifications** : le sujet étant clair à ce niveau, nous savions dès le départ tout ce qui était attendu de nous. Ainsi, comme les livrables ne risquaient pas de changer en cours de route et que la participation des parties prenantes était limitée, une gestion agile n'était pas de mise.

- **La structuration claire et la gestion du temps** : La gestion de projet classique offre une structure claire et bien définie avec des étapes séquentielles. Cela permet de diviser le projet en phases distinctes, ce qui facilite la planification, l'organisation et le suivi des progrès. La gestion de la contrainte temporelle en est également plus aisé.

  Nous avons décidé de découper le projet ainsi :
  - Semaine 1 : définition, planification et répartition du projet.
  - Semaine 2 à 5 : conception, développement, tests et vérifications de la première étape.
  - Semaine 5 à 8 : conception, développement, tests et vérifications de la seconde étape.
  - Semaine 9 : Evaluation, rédaction du rapport et clôture du projet

- **La documentation** : La méthode classique encourage la documentation régulière des résultats intermédiaires du projet, cela renforce le suivi des activités, la détection des erreurs et simplifie l'étape de rédaction du rapport.

II.B Forces et faiblesses

Pour travailler efficacement il est important de connaître les forces sur lesquelles ont peut s'appuyer, ainsi que les faiblesses auxquelles il faut faire attention.

Nos forces :

- On a de l'expérience ensemble, on a déjà eu l'opportunité de travailler de concert. On connait donc les méthodes de travail de chacun et on peut alors mieux adapter la planification, la répartition et l'organisation du travail en fonction de chaque individu. On peut également noter que l'équipe fait preuve d'une forte cohésion.
- Chaque membre du groupe est doté de bonnes capacités de communication. Cela sera un avantage lors de nos réunions pour se comprendre rapidement mais aussi lors de la soutenance pour communiquer avec le jury.
- Notre expérience dans le domaine associatif nous a solidement formé dans les domaines relatifs à l'organisation. Ces points forts seront utiles dans tout ce qui est relatif à la gestion de projet.

Nos faiblesses :

- Nos connaissances en langage «C» sont encore très récentes, ainsi on aura de manière certaine besoin de plus temps pour développer ce type d'outils que des développeurs expérimentés.
- On est tous très engagés dans le mileu associatif de TELECOM et les cours nous prennent beaucoup de temps, on a donc peu de temps à consacrer au projet. Il faudra être discipliner sur le temps que l'on arrive à dégager.

On peut synthétiser cette analyse sous forme de tableau :

Forces	Faiblesses
Forte expérience en équipe	Novices en langage «C»
Bonne capacité de communication	Très pris par l'associatif et les cours
Expérience solide en organisation et en gestion de projet	

II.C Communication et collaboration

- **La communication interne** : Pour notre communication interne, nous avons décidé de continuer à utiliser Discord.
- **Les réunions** : En ce qui concerne les réunions nous avons décider dans la mesure de possible de faire une réunion d'avancement par semaine et des réunions techniques ponctuelles pour échanger sur les détails techniques. Toutes les réunions feront l'objet de compte rendu de réunion.
- **La répartition du travail** : Pour gérer nos ToDo, nous avons décidé d'utiliser le logiciel Trello.
- **Partage des ressources** : En ce qui concerne le partage des ressources et de nos avancées, nous utileserons le serveur GitLab de l'école.

PPII2

III ▷ Suivi et évaluation du projet

III.A Méthodes de suivi

Afin de mieux suivre l'avancement du projet et de ne pas se perdre suite aux ajouts des autres membres, nous avons, en plus de faire des réunions régulières pour se tenir informé de l'avancement global, choisi des normes dans l'écriture des commentaires et dans le nom des *commit*.

Pour le nom des *commit* :

feat – a new feature is introduced with the changes

fix – a bug fix has occurred

chore – changes that do not relate to a fix or feature and don't modify src or test files (for example updating dependencies)

refactor – refactored code that neither fixes a bug nor adds a feature

docs – updates to documentation such as a the README or other markdown files

style – changes that do not affect the meaning of the code, likely related to code formatting such as white-space, missing semi-colons, and so on.

test – including new or correcting previous tests

perf – performance improvements

ci – continuous integration related

build – changes that affect the build system or external dependencies

revert – reverts a previous commit

Pour les commentaires :

Nous avons utilisé les normes établit par le standard Doxygen :

Code 2.1 – Convention Doxygen

```
/**
 * @brief Brief description of the function.
 *
 * Detailed description of the function.
 *
 * @param param1 Description of parameter 1.
 * @param param2 Description of parameter 2.
 * @return Description of the return value.
 */
```

III.B Évaluation des résultats

Chaque fonction implanter dans le programme fait l'objet de tests unitaires afin que l'on s'assure de la validité des résultats renvoyés.

L'application finale devra remplir les objectifs définit dans la charte de projet (annexe A) pour que l'on puisse la valider.

III.C Évaluation de la performance

Le facteur clé de performance retenu pour l'évaluation de performance de chacune de nos applications est le temps d'exécution. Il a fait l'objet d'une étude pour chacune des partis du projet.

IV ▷ Charte de projet

La conclusion de cette première phase d'analyse du projet a été l'édition de la charte de projet, disponible dans l'annexe A.

Ce document nous a servi de référence tout au long du projet. Il nous a notamment été utile pour dans le suivi de la complétion des objectifs, et pour la gestion du temps.

◁ Chapitre 3 ▷
Première étape

I ▷ Analyse

La première consiste à construire en langage «C» un programme qui permet de proposer à tout usager souhaitant se rendre d'un point A à un point B du territoire, un parcours de charge de son véhicule électrique.

I.A Etat de l'art

I.A.1 Le problème

Le problème qui nous a été donné est un problème de recherche de plus court chemin. Il est alors fortement probable que nous soyons amenés à utiliser une structure de type graphe. Le problème du plus court chemin entre un sommet A et un sommet B revient à chercher le chemin de A à B où la somme des poids des arcs qu'il traverse est minimale [1]. Il faut donc choisir un algorithme qui trouve des solutions à ce problème.

I.A.2 Algorithme

Les potentiels algorithmes pour ce problème sont :

- ▷ Dijkstra
- ▷ A* [2]
- ▷ Bellman-Ford
- ▷ Viterbi
- ▷ Floyd-Warshall[3]
- ▷ Johnson[3]

I.A.3 Applications existantes

Ce problème ayant déjà été étudié, il existe des applications permettant de répondre aux besoins de notre sujet, notamment :

- ▷ **Chargermap :** Son fonctionnement est très simple. Il faut d'abord lancer l'application sur son smartphone Android ou Apple, puis renseigner ses points de départ et d'arrivée. Vous devez ensuite préciser le modèle de votre véhicule, les niveaux de batterie souhaités au départ comme à l'arrivée et enfin, la vitesse maximale que vous prévoyez au cours du trajet. À partir de ces informations, l'algorithme calcule en une poignée de secondes le chemin le plus adapté. L'itinéraire à suivre et les bornes où s'arrêter sont ainsi affichés sur une carte. Une feuille de route vous indique également la distance à parcourir entre chaque escale et la durée de chaque recharge. Vous verrez qu'il n'est pas systématiquement nécessaire de faire un plein complet, un appoint de quelques dizaines de minutes suffisant la plupart du temps.

- ▷ **A Better Routeplanner :** Le fonctionnement de cet outil est globalement similaire celui de Chargermap. Lieu de départ et d'arrivée, niveaux de batterie, modèle de véhicule : on y renseigne les mêmes informations. A Better Routeplanner (ABRP) se démarque cependant en proposant un mode expert, qui ouvre accès à un panneau extrêmement complet de données à préciser : poids des bagages et passagers, météo, consommation moyenne personnalisée, type de chargeurs, etc. Contrairement à Chargemap, ABRP présente aussi l'avantage d'estimer les prix des recharges

1. Si on ajoute des contraintes à ce problème comme des fenêtres de temps, le problème peut devenir NP-difficile.
2. Cet algorithme semble le plus efficace d'après les recherches.
3. Si l'on veut tous les chemins à partir d'un point

sur sa feuille de route. Les différences de temps de charge et de parcours entre Chargemap et A Better Routeplanner sont souvent le résultat d'estimations différentes de ses concepteurs. Le premier outil peut, par exemple, être plus ou moins optimiste sur la capacité d'un véhicule à recharger rapidement que le second.

PPII2

I.B Planification

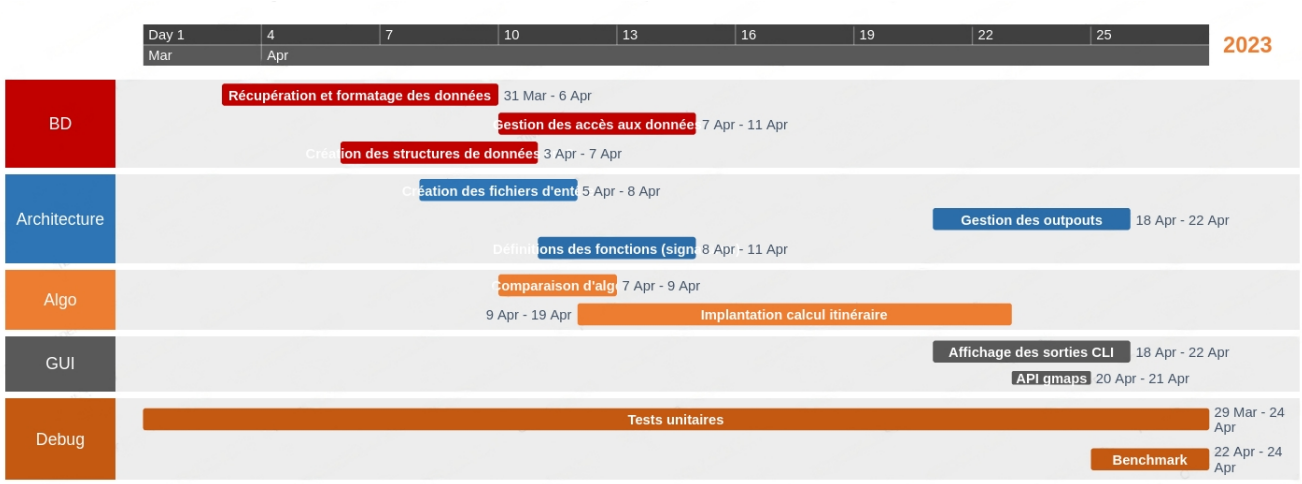
I.B.1 Work Breakdown Structure et Gantt

La première étape de la planification est de diviser le travail en sous lots. Le travail a été divisé en 5 parties :

- Architecture
- Algorithmie
- Base de données
- GUI
- Test et debug

Ces sous-lots sont eux-mêmes divisé (voir le diagramme de Gantt et la matrice RACI ci-dessous)

La seconde étape a été de répartir le travail dans le temps (notons que les éléments de la partie debug sont fait en parallèle de toutes les autres étapes) :



I.B.2 RACI

Enfin il a fallu répartir les responsabilités de chacun pour chacune des tâches :

Tâche	Stanislas	Corentin	Antonin	Yann
Récupération et formatage des données	R	I	I	I
Création des structures de données	R	I	I	A
Gestion des accès aux données	R	I	I	A
Création des fichiers d'entête	I	C	R	I
Définitions des fonctions (signatures)	CI	A	R	CI
Comparaison / choix d'algo plus court chemin	I	R	I	I
Implantation calcul d'itinéraire	I	R	I	I
Affichage des sorties en CLI	A	A	R	A
Lien avec gmaps	I	I	R	I
Benchmark	A	I	R	I
Gestion des outpouts	R	I	A	A
Tests unitaires	R	A	A	A

II ▷ Conception

II.A Manipulation des données

L'énoncé fournit deux ressources. La première est une basse de données du gouvernement répertoriant les stations de recharges pour véhicules électriques du territoire français ainsi que diverses informations sur ces stations (nombre de bornes, puissance, ...). La seconde est une base de données tirée du site https://ev-database.org/ qui contient une kyrielle de données sur les véhicules électriques du marché (autonomie, vitesse de recharge, ...).

La première étape de cette partie consiste à extraire les données mises à notre disposition de manière à pouvoir être exploitées facilement en C par la suite. Pour les stations de recharge, le site du gouvernement (serveur Etalab) propose nativement un export au format CSV qui est un format qui nous être le plus simple pour arriver à nos fins. Cependant, le site qui fournit les données sur les véhicules, lui, ne propose aucune option d'export. Nous avons donc commencé par écrire un script de *parsing* en Python qui va récupérer les données sur le site et les enregistrer dans un fichier CSV. Nous n'avons récupéré que les données qui nous semblaient utiles parmis toutes celles proposées sur le site internet. C'est-à-dire le nom du véhicule, son autonomie ainsi que sa vitesse de recharge

Code 3.1 – Script de récupération des données sur les véhicules électriques

```python
import requests
from bs4 import BeautifulSoup
import json

url = """https://ev-database.org/#sort:path~type~order=.erange_real~number~desc|
    range-slider-range:prev~next=0~1200|range-slider-acceleration:prev~next=2~23|
    range-slider-topspeed:prev~next=110~350|range-slider-battery:prev~next=10~200|
    range-slider-towweight:prev~next=0~2500|range-slider-fastcharge:prev~next=0~1500|
    paging:currentPage=0|paging:number=all"""

response = requests.get(url)
assert (response.ok)

soup = BeautifulSoup(response.text, 'html.parser')

vehicle_info = []
vehicles = soup.find_all('div', class_='list-item')[1:]  # skip header row
for vehicle in vehicles:
    vehicle_tmp = {}
    vehicle_tmp['name'] = vehicle.find('h2').text
    specs = vehicle.find('div', class_='specs').find_all('p')
    vehicle_tmp['range'] = specs[2].find_all('span')[1].text[:-3]
    vehicle_tmp['fast_charge'] = specs[4].find_all('span')[2].text

    vehicle_info.append(vehicle_tmp)

with open('vehicle_info.json', 'w') as f:
    json.dump(vehicle_info, f, indent=4)
```

PPII2

II.B Théorie des graphes

Le problème posé dans cette première partie nous a tout de suite fait penser à la théorie des graphes et plus particulièrement à un calcul de plus court chemin. L'algorithme de Dijkstra[3] ayant été étudié en MSED, nous avons été sceptiques dès le début quant à son utilisation. En effet, cet algorithme nécessite de parcourir au moins une fois chaque borne de recharge pour calculer le chemin le plus court, ce qui ne semble pas très efficace compte tenu des ~ 50000 stations. Nous nous sommes donc tournés vers un autre algorithme, A^*[4] qui considère l'éloignement au point d'arrivée et donc ne parcours pas toutes les stations.

Une fois l'algorithme choisi, une zone d'ombre subsiste. Comment prendre en compte l'autonomie de la voiture dans notre calcul de plus court chemin ? La solution est venue assez naturellement, il suffit à chaque itération de ne pas considérer les voisins dont la distance est supérieure à l'autonomie du véhicule considéré.

II.C Paramètres d'entrée et de sortie

En plus des deux fichiers `CSV` contenant les données sur les stations de recharge et les véhicules électriques contenus dans le répertoire `/data/raw/`, l'exécutable prendra en paramètres directement sur la ligne de commandes les arguments suivants :

Entrée :

- Adresse de départ ou coordonnées géographiques
- Véhicule utilisé
- *(optionnel)* Pourcentage de batterie minimum à ne pas dépasser
- *(optionnel)* Temps de recharge maximum

Sortie :

- Liste des stations à visiter
- Pourcentage de batterie et distance restante à chaque étape
- Lien de visualisation sut Google Maps

3. https://fr.wikipedia.org/wiki/Algorithme_de_Dijkstra
4. https://fr.wikipedia.org/wiki/Algorithme_A*

III ▷ Développement

PPII2

III.A Structures de données

La première structure de données à choisir et sans aucun doute la plus importante est celle qui va contenir les informations sur les stations importées depuis le fichier `CSV`. Étant données que ces informations vont être consultées plusieurs fois à chaque étape du calcul de plus court chemin, il est important de choisir une structure de données qui profère un accès rapide. Ainsi, nous nous sommes tournés vers une table de hachage appelée **`Table_t`** qui sera chargée une seule fois à partir du fichier `CSV` puis qui permettra de consulter les informations avec une complexité temporelle en $\mathcal{O}(1)$.

En ce qui concerne les informations sur les véhicules électriques, nul besoin d'une table de hachage, lors de l'appel de la fonction, le fichier `CSV` sera parcouru une seule fois de manière linéaire et les informations associées au véhicule considérées seront placées dans une structure de données appelée **`Vehicle_t`**.

Le résultat de l'algorithme *i.e.* la liste des stations à visiter, sera stocké dans une liste contiguë (déjà implantée pour la table de hachage) appelée **`List_t`**.

III.B Algorithmes

III.B.1 Calcul de plus court chemin : `A-star`

III.B.2 Conversion d'adresse en coordonnées géographiques

Afin de faciliter l'utilisation de l'application, et d'être plus productifs lors des tests de nos programmes, nous avons implémenté l'utilisation d'une API de nominatims pour rechercher les coordonnées géographiques d'un endroit à partir d'une description variable de son adresse. Par exemple : «TELECOM Nancy», «193 avenue paul muller», «LORRAINE», «Place Stanislas», etc.

Pour ce faire, nous avons utilisé la librairie `libcurl`[5] du projet *open source* cURL. Cette librairie permet de faire des requêtes HTTP et HTTPS, et donc de communiquer avec des API. Il nous a donc fallu créer un certain nombre de structures de données, car la requête à l'API génère un *buffer* en temps direct, et gérer tous les éventuels cas d'erreurs : réponse nulle, connexion impossible, etc.

L'API utilisée est celle d'**OpenStreetMap**[6]. Elle permet de faire des recherches d'adresses, de lieux, de villes, etc. à partir d'une description textuelle. Elle renvoie un fichier JSON contenant les informations demandées, dont les coordonnées géographiques.

Notre programme exécute donc une requête HTTPS à cette API et récupère la réponse sous forme de JSON, qu'une autre fonction parse pour en extraire les informations utiles.

III.C Visualisation

L'utilisation de bibliothèques graphiques telles que `GTK`[7] ou d'un framework web tel que `Flask`[8] aurait été une solution envisageable pour la visualisation des résultats. Cependant, nous avons choisi d'utiliser `Google Maps`[9] pour plusieurs raisons :

5. https://curl.se/libcurl/
6. https://nominatim.org/release-docs/develop/api/Search/
7. https://www.gtk.org/
8. https://flask.palletsprojects.com/en/1.1.x/
9. https://www.google.com/maps

- Beaucoup plus simple à mettre en place car il suffit de générer un lien avec les coordonnées géographiques des stations à visiter et les points de départ er d'arriver pour obtenir une carte interactive
- Le lien peut être partagé facilement
- Le temps économisé sur cette partie nous a permis d'ajouter des fonctionnalités supplémentaires

IV ▷ Tests et performances

IV.A Tests unitaires

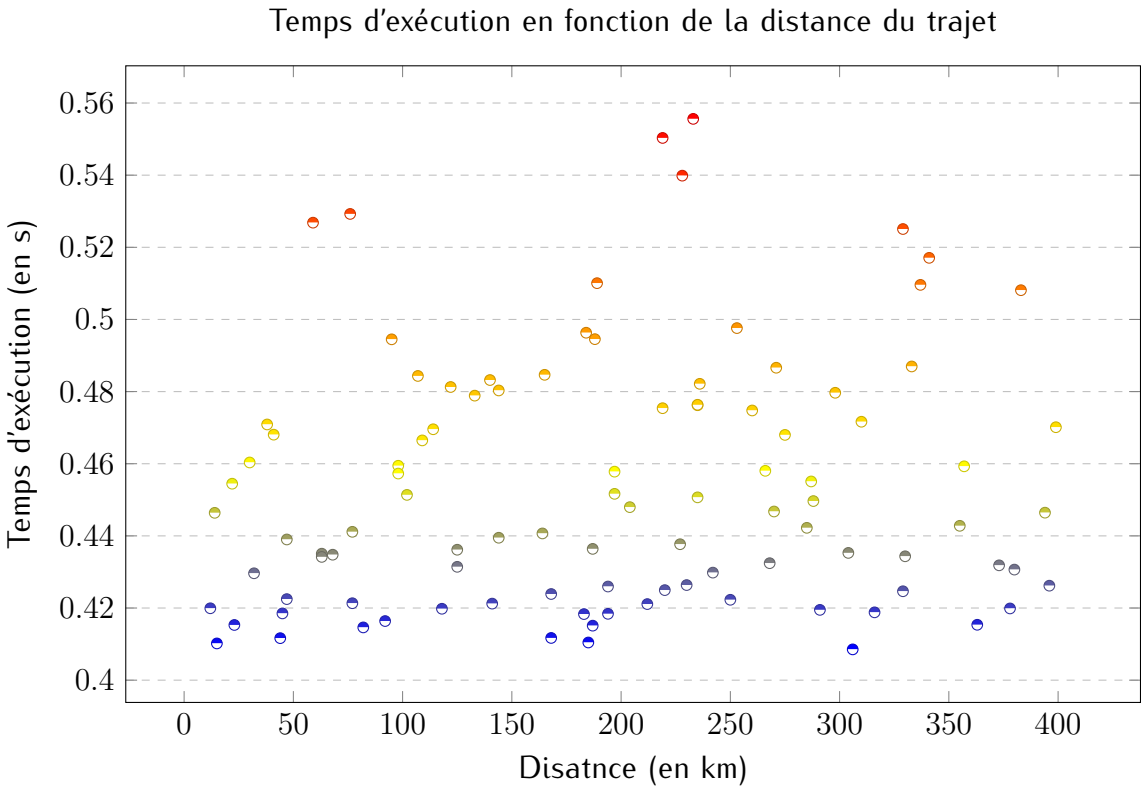
Dès le début du projet, nous avons choisi d'utiliser la bibliothèque `Snow`[10] pour les tests unitaires. En effet, cette bibliothèque est très simple d'utilisation et permet de tester les fonctions de manière isolée. De plus, l'utilisation de `Snow` a été couplée à celle du serveur d'intégration continue de `GitLab`[11]. Cela nous a permis de vérifier à chaque *commit* que le code poussé sur le serveur n'entravait pas les fonctions déjà poussées mais également de ne pas faire fusionner malencontreusement une branche dont les tests unitaires ne passaient pas.

IV.B Tests de performance

L'implantation d'un module de teste de performances nous semblé être important compte tenu de la taille des données manipulées et du fait de leur impact direct sur les performances de la partie suivante. Ainsi, le répertoire `/benchmark/` contient un programme de test pouvant être exécuté avec les arguments suivants :

- `-f` pour spécifier le nom du fichier `CSV` contenant les résultats des tests (fichier placé dans le répertoire `/data/benchmark/`)
- `-d` pour utiliser les paramètres par défaut, à savoir
  - 100 trajets
  - trajets aléatoires
  - distances entre 1 et 400km (le point de référence étant le milieu de la France)

L'exécution sur le serveur de l'école `https://gitlab.telecomnancy.univ-lorraine.fr/` avec les paramètres par défaut a permis de générer les graphiques suivants :



10. https://github.com/mortie/snow
11. https://docs.gitlab.com/ee/ci/

IV.C Interprétation des résultats

Remarquons que l'impact de la distance ne se ressent quasiment pas sur le temps d'exécution. Ce qui montre que l'utilisation d'une table de hash est pertinent. De plus, les temps d'exécutions comprennent le temps de remplissage de la table de hachage et de lecture du fichier `CSV`. Lors de la seconde partie, plusieurs appels au calcul de plus court chemin ne demanderont qu'un seul chargement de la table de hachage.

◁ Chapitre 4 ▷
Seconde étape

I ▷ Analyse

I.A Etat de l'art

PPII2

I.A.1 Le problème

Le problème qui nous a été proposé est un problème de surveillance du réseau de station de recharge pour véhicule électrique. Il s'agit en fait de repérer les stations susceptibles d'être trop souvent surchargées afin de mieux prévoir les expansions futures du réseau.

I.A.2 Algorithme

Il s'agit ici de réutiliser l'outil crée dans la partie 1 sur une base de plusieurs utilisateurs et d'en récupérer les résultats. Il faudra ensuite traiter ces résultats pour en extraire les données voulues.

I.A.3 Applications existantes

Il est assez difficile pour une application réelle de savoir si une file d'attente s'est générée aux différentes station de recharge car la plupart du temps les données sont privées. Toutefois il existe quelques alternatives :

- **Chargermap :** L'application propose aux utilisateurs de s'ajouter à une liste d'attente mais cela ne fonctionne que pour les bornes possédé par l'entreprise.

I.B Planification

I.B.1 Work Breakdown Structure

La planifiacation s'est fait de la même manière que pour la première partie. La division en sous-lot est identique :

- Architecture
- Algorithmie
- Base de données
- GUI
- Test et debug

Ces sous-lots sont eux-mêmes divisé (voir la matrice RACI ci-dessous)

I.B.2 RACI

Nous nous sommes ensuite réparti les responsabilités de chacun pour chacune des tâches :

Tâche	Stanislas	Corentin	Antonin	Yann
Récupération et formatage des données	R	I	I	I
Création des structures de données	R	I	I	A
Gestion des accès aux données	R	I	I	A
Création des fichiers d'entête	I	C	R	I
Définitions des fonctions (signatures)	CI	A	R	CI
Traitement des donnés	CI	R	A	I
Affichage des sorties en CLI	A	A	R	A
Gestion des outpouts	R	I	A	A
Affichage des données	R	I	I	A
Tests unitaires	R	A	A	A

I.C Paramètres d'entrée et de sortie

En plus des deux fichiers CSV contenant les données sur les stations de recharge et les véhicules électriques contenus dans le répertoire `/data/raw/`, l'exécutable prendra en paramètres directement sur la ligne de commandes l'arguments suivant :

PPII2

Entrée :

- Nom du fichier `.csv` d'entrée (peut être précédé d'un chemin), contenant la liste des utilisateurs et leurs trajets respectifs
- Nom du fichier `.csv` de sortie (peut être précédé d'un chemin)

Sortie :

- Nom du fichier `.csv` de sortie

I.C.1 Format des données en entrée

Pour cet exemple d'ensemble de trajets :

Départ	Arrivée	Véhicule	Tick de départ
Cannes	Paris	Peugeot e-208	2
Nancy	Toulouse	Tesla Model 3	0
Paris	Nice	Zoe	1

Le fichier `.csv` à donner en entrée devra avoir ce format :

Code 4.1 – Exemple de fichier d'entrée

I.D Structures de données

Pour cette seconde partie, nous avons bien entendu réutilisé les structures de listes de stations, de véhicules et la table de hachage de la partie 1.

I.D.1 Ticks

Comme proposé dans le sujet, nous avons découpé le temps en **ticks**. Chaque tick correspond à un instant dans le temps et est séparé du précédent par 10 minutes. Le déplacement des utilisateurs dans le réseau de stations correspond donc à un système à événements discrets.

I.D.2 Timelines

Nous avons aussi décidé d'utiliser des structures que nous avons dénommé **`Timeline_station_t`** et **`Timeline_user_t`**. Ces structures sont des listes chaînées dont chaque élément correspond à l'état d'une station ou d'un utilisateur à un instant (tick) donné. L'utilisation de pointeurs vers les stations et les véhicules a rendu très simple la gestion de ces structures.

Pour pouvoir suivre l'état global du réseau de stations, nous avons aussi créé des structures **`Timeline_all_stations_t`** et **`Timeline_all_users_t`**. Ces structures sont des tableaux de pointeurs vers les **`Timeline_station_t`** et **`Timeline_user_t`** respectivement. Chaque élément de ces tableaux correspond à une timeline pour une station ou un utilisateur donné.

I.E Algorithmes

I.E.1 Initialisation

PPII2

Tout d'abord, nous calculons tous les trajets des utilisateurs. Pour cela, nous utilisons les mêmes algorithmes que ceux de la partie 1.

Il s'agit ensuite de créer les timelines. Pour cela, nous initialisons la timeline des utilisateurs avec les états initiaux de chaque utilisateur.

Nous faisons ensuite un inventaire des stations au moins parcourues une fois par un utilisateur, afin d'économiser de la mémoire lors de l'exécution du programme. Nous initialisons ensuite la timeline des stations avec les états initiaux de chaque station utile.

I.E.2 Calcul des états

Pour chaque tick, nous calculons de manière procédurale le nouvel état (si différent du précédent) de chaque station et de chaque utilisateur. Nous ajoutons ensuite ces états à la timeline correspondante.

I.E.3 Lecture des timelines

Une fois les timelines construites, nous pouvons les lire pour en faire une analyse. Les taux de remplissage des stations sont calculés à partir des états des stations à chaque tick. Il sont ensuite exportés dans un fichier `.csv`.

I.F Visualisation

Le fichier `.csv` contenant les taux de remplissage des stations peut être facilement importé dans un tableur pour être visualisé sous forme de graphiques.

Voici un exemple de fichier en sortie :

tick	0	1	2
ID_Station_1	0.000	0.500	0.250
ID_Station_2	0.000	0.000	0.333
ID_Station_3	0.166	1.125	0.625

II ▷ Tests

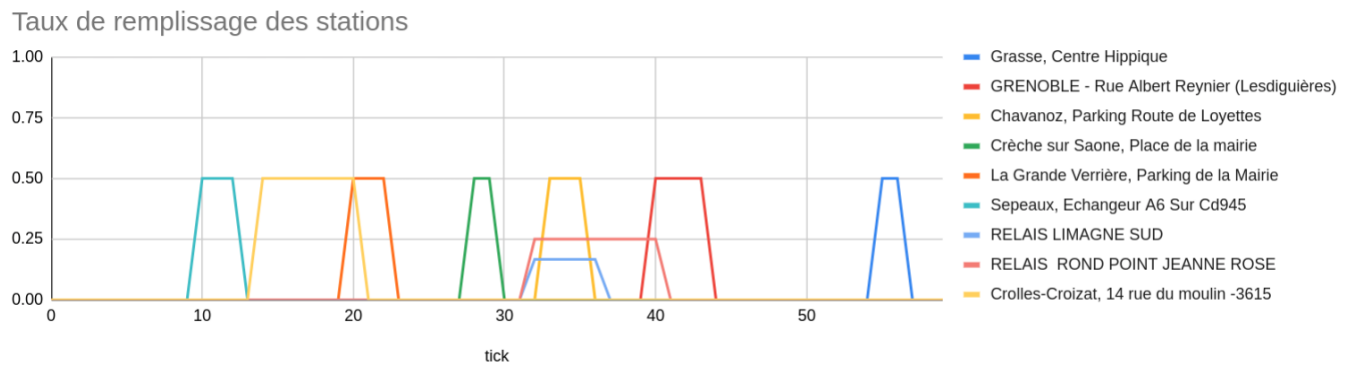
II.A Tests unitaires

Pour cette partie aussi, nous avons utilisé la bibliothèque `Snow` pour les tests unitaires. Nous avons testé les fonctions de manière isolée. Nous avons également utilisé le serveur d'intégration continue de `GitLab` pour vérifier à chaque *commit* que le code poussé sur le serveur ne contenait pas d'erreurs.

II.B Tests de simulations

Nous avons aussi effectué des tests de simulations et vérifié que le remplissage des stations, ainsi que les files d'attentes des utilisateurs étaient cohérents. Nous avons également vérifié que les trajets étaient bien effectués dans le temps imparti.

PPII2



◁ Chapitre 5 ▷
Conclusion

I ▷ Complétion des objectifs

En ce qui concerne l'étape 1, notre application a été développée de manière à renvoyer le trajet le plus court possible tout en respectant les capacités de la voiture. Nous avons également réussi à générer automatiquement le dit trajet dans l'application *Google Maps*, offrant ainsi à l'utilisateur un confort supplémentaire. Enfin, plusieurs fonctions ont été ajoutées comme la possibilité de spécifier des paramètres supplémentaires tels que le seuil de batterie minimal ou le temps de recharge maximal, permettant ainsi une personnalisation avancée des trajets.

Quant à l'étape 2, notre application a été étendue pour renvoyer l'évolution de la charge des stations en fonction d'un nombre de trajets d'utilisateurs donné. Cette fonctionnalité permet de mieux gérer les ressources des stations de recharge en anticipant les demandes et en évitant les congestions. De plus, les données collectées sont affichables, ce qui permet aux gestionnaires du système de visualiser facilement l'état des stations de recharge.

Notons également que dans les étapes les indicateurs de performances sont satisfaisant.

Nous pouvons donc affirmer que le projet a été mené à bien et a atteint tous les objectifs fixés pour chaque étape. L'ensemble des critères de succès ont été validés avec des indicateurs de performances satisfaisant.

II ▷ Améliorations possibles

- ▷ Gestion de projet
  - Mieux prévoir l'influence des évènements extérieurs sur le projet que ce soit les évènements relatifs à la vie associative de l'école (organisation d'évènements et donc disponibilité limitée sur de relativement longue période) ou à la vie académique (il est par exemple difficile de consacrer beaucoup de temps au projet en période de partiel)

- ▷ Etape 1
  - L'application calcul un trajet optimisé en distance, il pourrait être intéressant de pouvoir également calculer un trajet optimisé en temps.

- ▷ Etape 2
  - Ne prend pas en compte la vitesse de recharge des stations car les unités dans la base de données ne sont pas toujours les mêmes.

III ▷ Clôture

Ce projet a été passionnant pour toute notre équipe, et nous avons eu plaisir à travailler sur un sujet aussi intéressant. L'optimisation des trajets pour les véhicules électriques est une question pertinente et cruciale à l'heure actuelle, car elle contribue à la transition vers une mobilité plus durable.

Nous avons abordé ce projet avec enthousiasme et détermination, en mettant à profit nos compétences et notre expertise pour atteindre les objectifs fixés. Nous sommes parvenus à créer un environnement de travail productif grâce à une collaboration étroite au sein de l'équipe, essentielle pour notre réussite.

En résumé, ce projet a été un succès technique et humain. Nous sommes fiers d'avoir développé une application d'optimisation des trajets pour les véhicules électriques, contribuant ainsi à une mobilité plus durable. Travailler sur ce projet stimulant a été une expérience enrichissante, et nous sommes inspirés pour continuer à explorer de nouvelles opportunités dans le domaine de la technologie verte. Nous espérons que notre travail encouragera l'adoption des véhicules électriques et favorisera la transition vers un avenir plus respectueux de l'environnement.

◁ Annexe A ▷
Charte de projet

CP1 | PPII2 ▷ Charte de Projet

Table 1 – Auteurs

Nom / mail	Qualité / rôle
Stanislas MEZUREUX / stanislas.mezureux@telecomnancy.eu	Chef de projet
Yann DIONISIO / yann.dionisio@telecomnancy.eu	Membre de l'équipe projet

Table 2 – Historique des modifications et révisions de ce document

nº de version	Date	Description et circonstances de la modification
V1	23/05/2023	Première version à la suite de la deuxième réunion

Table 3 – Validation / autorisations

nº de version	Nom / qualité	Date / signature	Commentaires er réserves éventuelles
V1	Commanditaires		

I Résumé

- ▷ Il nous est dans un premier temps demandé de créer une application de calcul d'itinéraire optimisé pour véhicule électrique.
- ▷ Dans un second temps, il nous faudra concevoir une application de monitoring du réseau de stations de recharge.
- ▷ Ces deux applications seront principalement conçu en langage "C".

II Cadrage

II.1 Finalités et importance du projet

Ce projet s'inscrit dans le cadre du programme d'étude de première année de Telecom Nancy. Il répond à l'objectif proposé pour le Projet Interdisciplinaire d'Informatique Intégrative 2 (PPII2).

L'augmentation de l'utilisation des voitures électriques induite par la prise de conscience écologique apporte également son lot de contraintes parmi lesquelles l'autonomie encore limitée des batteries et la répartition inégale des stations de recharges sur le territoire.

Pour répondre à ces problématiques qui peuvent ralentir la transition écologique, il nous est proposé de concevoir des outils utiles à l'utilisateur (système d'optimisation d'itinéraire) mais aussi les entreprises afin de mieux planifier l'expansion du réseau de bornes de recharge.

II.2 Objectifs et résultats opérationnels

Liste des livrables :

- **Application de planification d'itinéraire fonctionnelle** : code source
- **Application de monitoring des stations de recharge fonctionnelle** : code source
- **Rapport de projet** : document de présentation de développement du projet
- **Documents relatifs à la gestion de projet**

Critères de succès et indicateurs mesurables :

- Étape 1 :
  - Critères de succès :
    - L'application renvoie le trajet le plus court (qui respecte les capacités de la voiture).
    - Le trajet est affichable dans une application tierce.
    - Il est possible d'indiquer des paramètres supplémentaires (seuil de batterie minimal, temps de recharge maximal, ...)
  - Indicateurs clés de performance :
    - Temps d'exécution.
- Étape 2 :
  - Critères de succés :
    - L'application renvoie l'évolution de la charge des stations en fonction d'un nombre de trajets d'utilisateurs donnés.
    - Les données sont affichables
  - Indicateurs clés de performances :
    - Temps d'exécution.

III Déroulement du projet

III.1 Organisation / ressources, budget

Ce travail s'effectue par groupe de quatre, toutes les ressources produites transitent via le serveur GitLab de l'école et les ressources dont nous disposons sont les locaux de l'école, le soutient du corps enseignant ainsi que les bases de données relatives aux stations de recharge et aux véhicules électriques.

Table 4 – Parties prenantes

Membres de l'équipe	Autres parties prenantes
Stanislas MEZUREUX : chef de projet	Olivier FESTOR : commanditaire
Corentin BILLARD	Gérald OSTER
Antonin FREY	Autres groupes
Yann DIONISIO	Utilisateurs finaux

Moyens à mobiliser ordinateurs (personnels et de l'école), C, Latex, ...

III.2 Jalons : échéancier / événements importants

Table 5 – Échéancier

Jalon	Description	Date
Étape 1 : Définition et cadrage	Latex : Etat de l'art et charte de projet	24/03/2023
Étape 2 : Montage partie 1	WBS, Gantt et RACI	29/03/2023
Étape 3 : Développement partie 1	Code source	19/04/2023
Étape 4 : Évaluation et test partie 1	Tests et Benchmark	24/04/2023
Étape 5 : Montage partie 2	WBS, Gantt et RACI	30/04/2023
Étape 6 : Développement partie 2	Code source	14/05/2023
Étape 7 : Évaluation partie 2	Tests et Benchmark	17/05/2023
Étape 8 : Rapport de projet	Latex : Rapport	24/05/2023
Étape 9 : Soutenance	Soutenance	31/05/2023

III.3 Risques et opportunités

Table 6 – Éléments favorables et défavorables

Favorable	Défavorable
L'équipe a déjà travaillée ensemble Pas de coûts	Peu d'expérience en C Difficile d'évaluer le temps que va prendre chaque jalon

Scénarios défavorables :

1) Le projet est trop ambitieux, nous ne parvenons pas rendre le livrable principal à temps

◁ Annexe B ▷
Compte rendu de réunion 1

CR1 | PPII2 ▷ Compte-rendu de réunion

Motif / type de réunion : Réunion technique	**Lieu :** Salle S0.8
Présent(s) : Tout le monde	**Date / heure :** 21/03/2023 | 16h15 - 17h15

I Ordre du jour

1) Analyse du sujet
2) Organisation
3) Premières difficultées
4) Remarques / questions
5) To Do List

II Informations échangées

II.1 Analyse du sujet

II.1.a Parcours rapide du sujet

Première partie :

- ▷ Le sujet demande de coder en C, il faut donc que nous nous assurions que nous sommes tous au point dans ce langage.
- ▷ L'objectif du sujet est de trouver le chemin le plus optimisé entre deux bornes de recharges pour véhicules électriques. Dans un premier temps, nous travaillerons en utilisant les coordonnées GPS et en considérant que toutes les bornes sont reliées entre elles par une route directe.
- ▷ Pour cette première partie, les données dont nous aurons besoin sont :
  Pour les voitures :
  - ○ La capacité
  - ○ La consommation

  Pour les stations :
  - ○ Les coordonnées

Deuxième Partie :

- ▷ La seconde partie consiste à mettre en application notre premier programme lorsque plusieurs usagers utilisent le réseau de station de recharge.
- ▷ La gestion du temps sera primordiale dans cette partie, nous essayerons dans un premier temps de faire partir tout le monde en même temps, puis de faire des départs différés. Il faudra également prendre en compte le temps de recharge des utilisateurs.

- La vitesse étant constante, il faudra que nous la choissisions de manière arbitraire. Il pourrait être intéressant de la choisir de manière à simplifier les calculs.

Les rendus :

- L'application et son code source
- Une documentation fournie (ReadMe, ...)
- Les documents de GdP

II.1.b Premières idées

- L'idée d'optimisation des chemins nous a fait penser aux algorithmes de recherche des plus courts chemin comme celui de Dijsktra.
- Nous nous sommes également posé la question de comment pouvions-nous nous démarquer des autres groupes lors de ce projet, de quelle valeur ajoutée avons nous à aporter au projet. Dans un premier ce sera la qualité de notre algorithme qui rentrera en compte, il faudra essayer de l'optimiser un maximum. Ensuite ce sera les différentes features que nous déciderons d'implémenter (utilisation d'adresse au lieu de coordonnées GPS, prise en compte du temps de recharge, interface graphique, ...). Pour se faire, il peut être intéressant de récupérer plus de données en prévisions. Il sera également important de faire attention aux détails comme les commentaires sur le code.
- Stanislas a proposé d'utiliser l'intégration continue sur gitlab, notamment en ce qui concerne la gestion des tests unitaires. Cette méthode nous permettrait de refaire passer tous les tests à chaque commit et ainsi de détecter directement une erreur.

II.2 Organisation

- Nous avons décidé de continuer à utiliser Discord pour notre communication interne.
- Nous avons en revanche décidé d'abandonner Notion au profit de Trello plus simple à mettre en place pour la gestion de To Do List.

II.3 Premières difficultées

- La base de données concernant les véhicules électriques n'est pas directement exportable, nous allons donc certainement devoir faire du scraping pour récupérer les données.

III Remarques / questions

- -

IV To Do List

Tache	Responsable	Deadline
CR 1	Yann	23/03/2023
Scraping	Stanislas	29/03/2023
Trello	Corentin	29/03/2023
WBS	All	29/03/2023
Etat de l'art	Corentin	29/03/2023
Idées de features	Antonin	29/03/2023
SWOT	Yann	29/03/2023

Prochaine réunion : 29/03/2023 - 12h

◁ Annexe C ▷
Compte rendu de réunion 2

Dionisio, Frey, Billard, Mezureux

CR2 | PPII2 ▷ Compte-rendu de réunion

Motif / type de réunion : Réunion d'avancement	**Lieu :** Salle de TN
Présent(s) : Les quatre membres du groupe projet	**Date / heure :** 29/03/2023 | 13h15 - 14h00

I Ordre du jour

1) SWOT
2) WBS
3) État de l'art

II Informations échangées

II.1 SWOT

II.1.a Forces

- ▷ On a de l'expérience ensemble et on se connaît.
- ▷ Par exemple, préférer les deadlines courtes parce que Corentin travaille souvent à la dernière minute.

II.1.b Faiblesses

- ▷ L'associatif à TELECOM nous demande à tous beaucoup de temps.
- ▷ C'est pourquoi, il a été difficile de trouver un créneau pour la dernière réunion.

II.1.c Opportunités

- ▷ Des locaux de Citroën ne se trouve pas très loin de TELECOM. Il pourrait être intéressant d'aller les voir pour se renseigner sur le sujet.
- ▷ Le sujet qu'on nous a donné est un sujet très important actuellement, ce qui nous permet d'avoir accès à beaucoup de ressource (sur Internet ou autre).

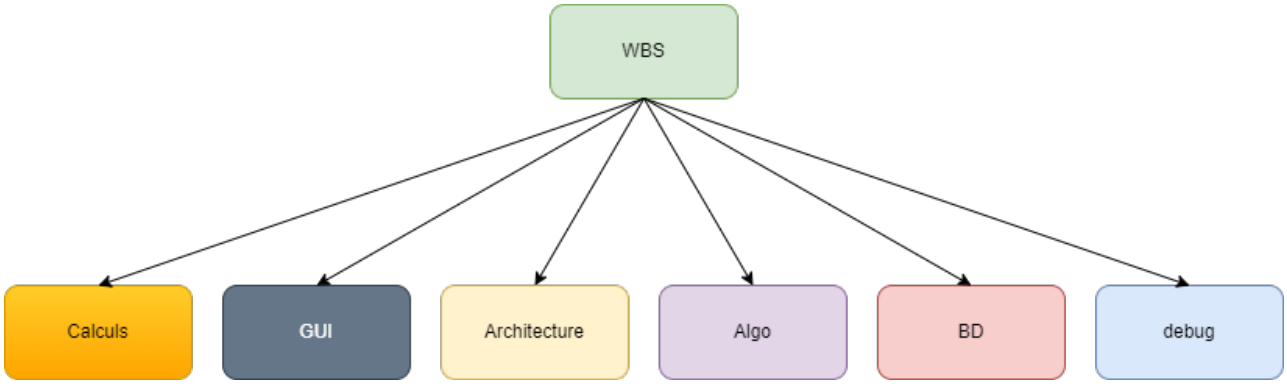
II.1.d Menaces

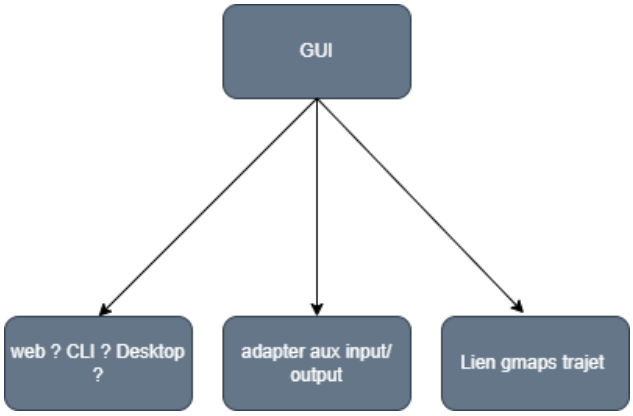
- ▷ Nous n'avons peut-être pas du matériel assez puissant pour faire tourner des algorithmes comme celui de Dijkstra sur une base de données aussi grosse que celle qui nous a été donnée. Nous devrons alors chercher à optimiser ces algorithmes, voir chercher d'autres algorithmes plus puissants. [1]
- ▷ Attention aux météorites

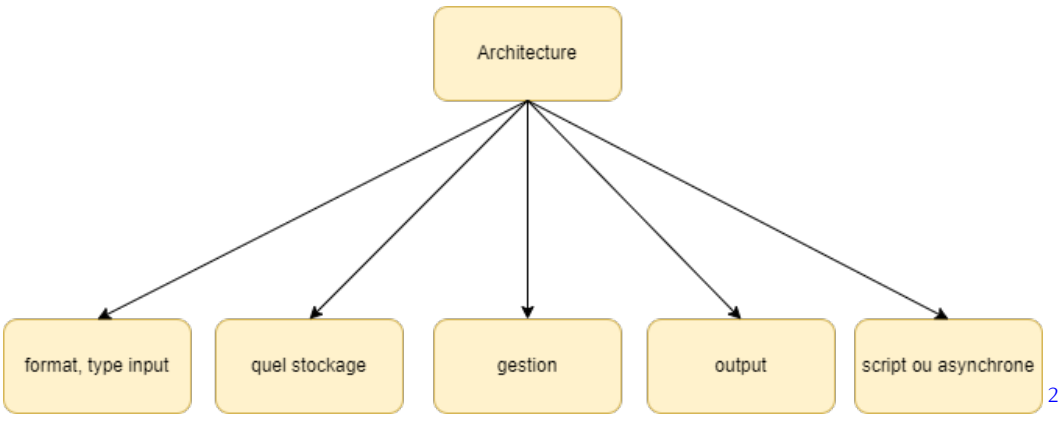
1. Voir commentaire dans la partie III. Remarques / questions

II.2 WBS

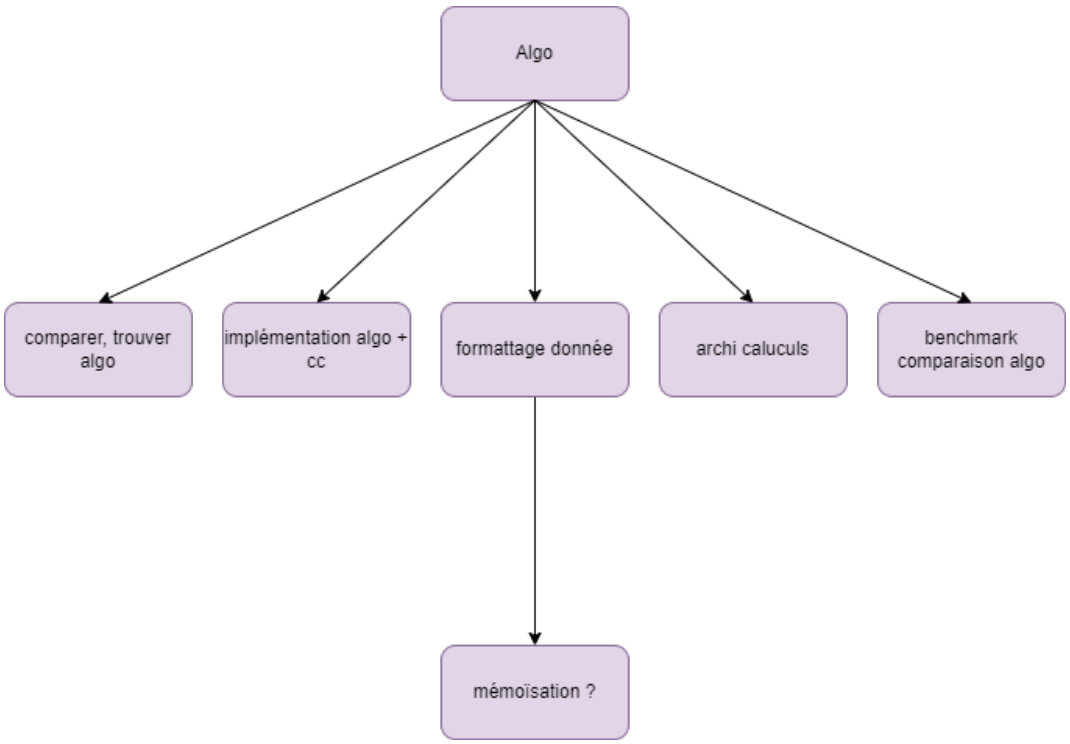
Le travail a été divisé en 6 parties :



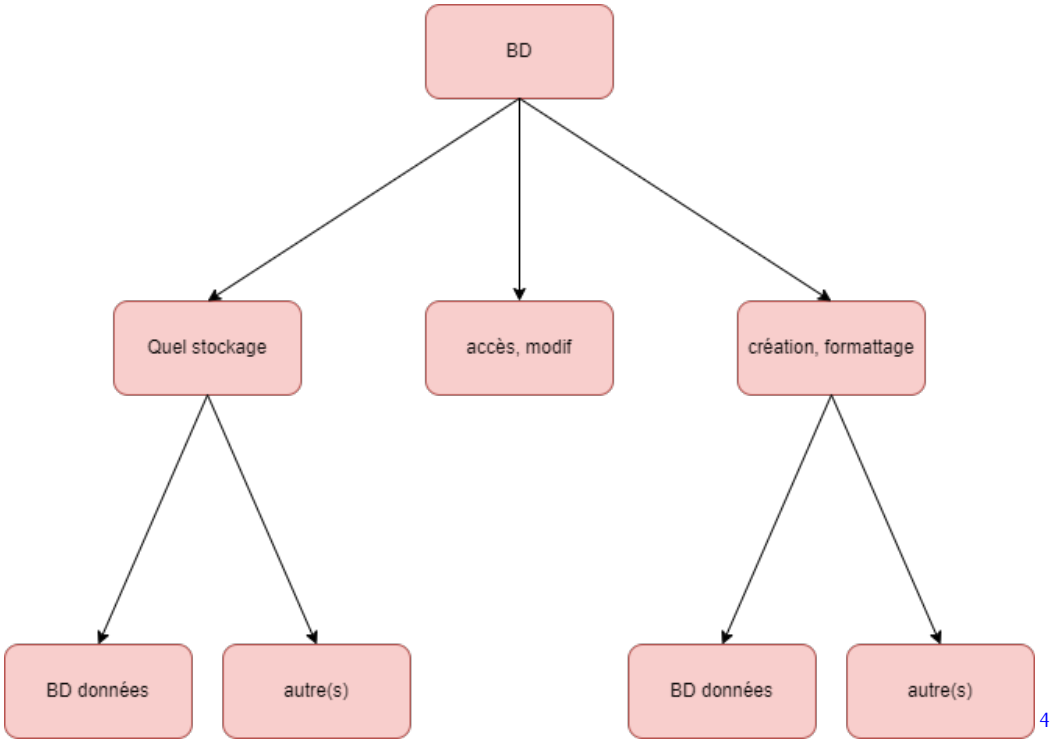




2. Voir commentaire dans la partie III. Remarques / questions

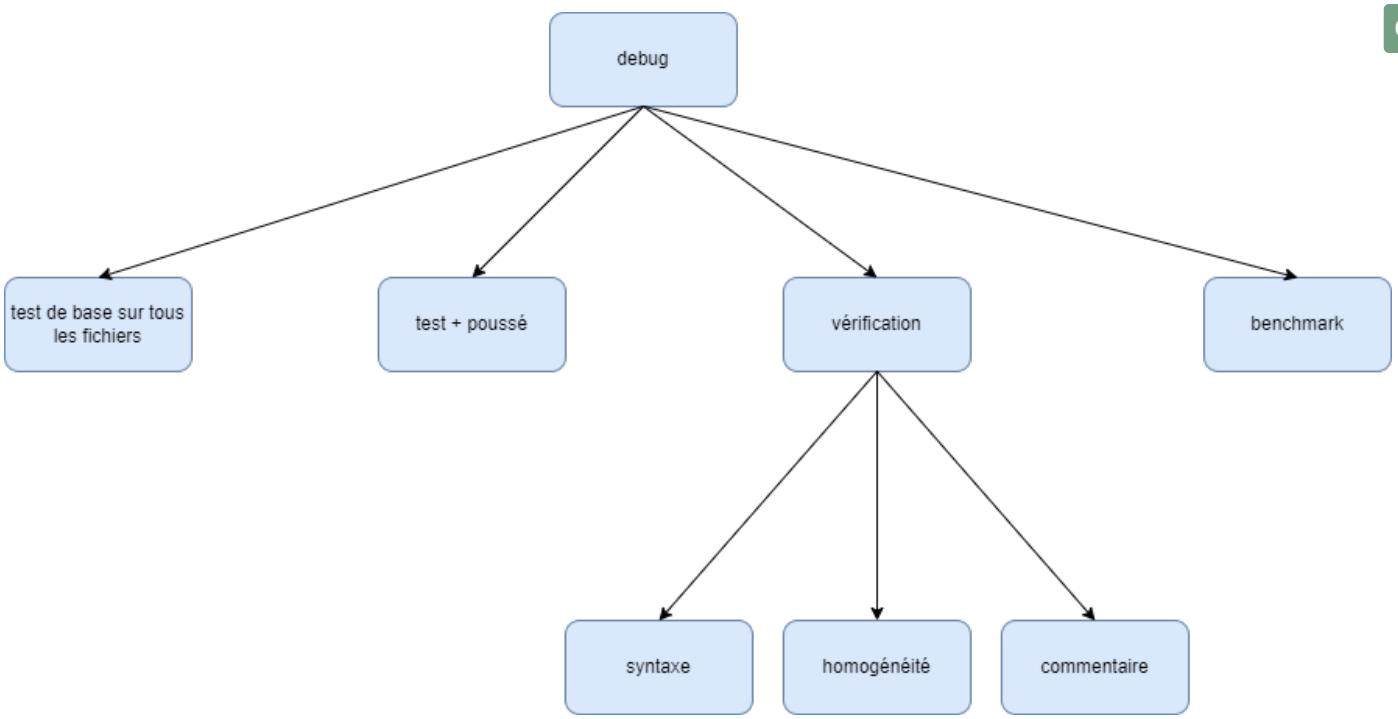


[3]



[4]

3. Mémoïsation ou autre technique
4. *Autre(s)* base(s) de donnée ajoutée(s) plus tard



II.3 État de l'art

II.3.a Le problème

- Le problème qui nous a été donné est un problème de recherche de plus court chemin.
- Problème du plus court chemin : soit un graphe orienté. Le problème du plus court chemin entre un sommet A et un sommet B revient à chercher le chemin de A à B où la somme des poids des arcs qu'il traverse est minimale. [5]
- Il faut donc choisir un algorithme qui trouve des solutions à ce problème.

II.3.b Algorithme

Les potentiels algorithmes pour ce problème sont :

- Dijkstra
- A* [6]
- Bellman-Ford
- Viterbi
- Floyd-Warshall [6]
- Johnson [6]

5. Si on ajoute des contraintes à ce problème comme des fenêtres de temps, le problème peut devenir NP-difficile.
6. Cet algorithme semble le plus efficace d'après les recherches
6. Si l'on veut tous les chemins à partir d'un point

II.3.c Application existante

Comme expliqué plus haut (cf. II.1.c page 1), ce problème a déjà été étudié. Il existe alors des applications permettant de répondre aux besoins de notre sujet, notamment :

- **Chargermap** : Son fonctionnement est très simple. Il faut d'abord lancer l'application sur son smartphone Android ou Apple, puis renseigner ses points de départ et d'arrivée. Vous devez ensuite préciser le modèle de votre véhicule, les niveaux de batterie souhaités au départ comme à l'arrivée et enfin, la vitesse maximale que vous prévoyez au cours du trajet. À partir de ces informations, l'algorithme calcule en une poignée de secondes le chemin le plus adapté. L'itinéraire à suivre et les bornes où s'arrêter sont ainsi affichés sur une carte. Une feuille de route vous indique également la distance à parcourir entre chaque escale et la durée de chaque recharge. Vous verrez qu'il n'est pas systématiquement nécessaire de faire un plein complet, un appoint de quelques dizaines de minutes suffisant la plupart du temps.
- **A Better Routeplanner** : Le fonctionnement de cet outil est globalement similaire celui de Chargemap. Lieu de départ et d'arrivée, niveaux de batterie, modèle de véhicule : on y renseigne les mêmes informations. A Better Routeplanner (ABRP) se démarque cependant en proposant un mode expert, qui ouvre accès à un panneau extrêmement complet de données à préciser : poids des bagages et passagers, météo, consommation moyenne personnalisée, type de chargeurs, etc. Contrairement à Chargemap, ABRP présente aussi l'avantage d'estimer les prix des recharges sur sa feuille de route. Les différences de temps de charge et de parcours entre Chargemap et A Better Routeplanner sont souvent le résultat d'estimations différentes de ses concepteurs. Le premier outil peut, par exemple, être plus ou moins optimiste sur la capacité d'un véhicule à recharger rapidement que le second.

III Remarques / questions

- Pour ce qui est de la menace de la performance de nos machines (cf. II.1.d page 1), il ne faut pas oublier que nous utilisons du C qui est un langage bas niveau. Ainsi, les calculs sont plus rapide avec ce langage qu'avec un langage haut niveau comme Python.
- **script ou asynchrone** (cf. II.2 page 2) pose la question : veut-on un code qui s'exécute comme un script, à savoir une fois pour un trajet, ou de manière asynchrone ; une fois lancer, il suffit de lui demander de faire un calcul pour qu'il le fasse même si un résultat précédent n'est pas encore calculé.
- Il faut qu'on définisse des conventions que tout le monde suivront.
- Rappel de l'importance de commenter le code.

Prochaine réunion : 07/04/23

◁ Annexe D ▷
Compte rendu de réunion 3

Dionisio, Frey, Billard, Mezureux

CR3 | PPII2 ▷ Compte-rendu de réunion

Motif / type de réunion : Réunion d'avancement	**Lieu :** Discord (salon vocal)
Présent(s) : Stanislas, Corentin, Antonin	**Date / heure :** 09/04/2023 | 14h00 - 15h50

I Ordre du jour

1) Algorithme de plus court chemin
2) Représentation des données

II Choix d'un algorithme de plus court chemin

Corentin nous présente les différents algorithmes de calcul de plus court chemin que nous pourrions utiliser afin d'évaluer le meilleur parcours d'un utilisateur. Les diapositives utilisées sont disponibles ici.

II.1 Algorithme de Dijkstra

- ▷ Cet algorithme est trop exhaustif : on calcule la distance de parcours de chemins dont on sait déjà qu'ils ne sont pas les meilleurs.
- ▷ Pour une matrice d'adjacente de plusieurs milliards de valeurs, la complexité serait trop mauvaise.

II.2 Algorithme de Bellman-Ford

- ▷ Similaire à celui de Dijkstra, cet algorithme est très exhaustif et a une méthode similaire, qui cherche à explorer tous les chemins possibles.

II.3 Algorithme A*

- ▷ Cet algorithme utilise une file d'attente pour prioriser les meilleurs chemins et mettre en retrait ceux qui ont le moins de chance d'être les meilleurs chemins.
- ▷ Il faut concevoir et fournir une fonction heuristique qui majore le calcul de la distance. Corentin précise que nous n'aurons qu'à utiliser la fonction heuristique de Manhattan (norme 1 dans le plan) car elle majore forcément un chemin (ce d'après l'inégalité triangulaire).
- ▷ Corentin nous décrit l'exécution de l'algorithme A* sur un exemple, et nous montre la comparaison avec l'algorithme de Dijkstra. Comme prévu, ce dernier est bien moins efficace.
- ▷ Dans un cas où l'algorithme aurait plusieurs sommets du graphe desquels il devrait explorer à nouveau, nous nous accordons à dire qu'il vaudrait mieux explorer ces sommets dans un ordre aléatoire plutôt que dans l'ordre de leurs indices. L'aléatoire n'augmenterait pas gravement la complexité et pourrait même la diminuer sur une moyenne d'essais.

II.4 Algorithme de Floyd-Warshall

- D'autres algorithmes comme celui de Floyd-Warshall existent aussi, mais servent à calculer tous les meilleurs chemins pour un graphe. Ces algorithmes ne sont donc pas ce que nous recherchons

II.5 Choix définitif

- Nous nous accordons donc à choisir l'algorithme A*, qui semble être celui qui convient le mieux à notre projet.
- Cet algorithme sera par ailleurs l'occasion de faire du *backtracking* et de la programmation dynamique (notamment de la mémoïsation).

III Représentation des données

Afin de pouvoir entièrement lancer la phase de développement, nous discutons de certaines conventions et de formats de données pour que le projet soit homogène.

III.1 Conventions

- Il faudra décrire les fonctions avec des commentaires mentionnant les paramètres et valeurs de retour (le cas échéant) afin que les intégrations dans les IDE puissent en donner des aperçus.
- Les noms de structures et de pointeurs devront être écrits en *Pascal case* (`CommeCeci`), les noms de fonctions devront être écrits en *snake case* (`comme_ceci()`) et les noms de fonctions n'étant pas censées être appellées à l'extérieur du fichier devront être écrits en *snake case* précédés d'un *underscore* (`_comme_ceci()`).
- Des structures pourront notamment être créées pour faciliter les opérations avec les points sur la carte ou la file d'attente de l'algorithme A*.

III.2 Base de données

- Il faudra formatter les données (informations sur les modèles de voitures, localisations des stations de recharge) d'une manière à ce qu'elles soient facilement accessibles et adaptées aux opérations.
- Nous évoquons la création d'un matrice d'adjacence, qui pose le problème du nombre de données stockées. Nous pourrions aussi stocker seulement les adjacences inférieures à une distance maximale, dans une structure semblable à un dictionnaire par exemple. Stanislas mentionne aussi l'utilisation d'une table de hachage pour accéder en temps constant à la valeur associée à une clé.
- Nous décidons d'y refléchir chacun de notre côté et de remettre nos idées en commun lors d'un stand-up meeting ou de la prochaine réunion d'avancement.

III.3 Visualisation

- Antonin propose d'implémenter une façon de visualiser brièvement un résultat de parcours donné par le programme, à des fins de débogage pour vérifier la bon fonctionnement de l'algorithme. Stanislas précise que cela pourrait être implémenté en C avec GTK.
- Cela pourrait aussi faire partie de l'interface utilisateur finale si les délais le permettent.

IV To-do list

Tâche	Responsable(s)	Deadline
Réfléxion format des données	Tous	14/04/23
Implémentation algorithme A*	Corentin	14/04/23
Implémentation inputs du programme	Non attribué	14/04/23
Conversion données (GeoJSON, etc.) en format décidé	Non attribué	21/04/23
Implémentation pré- et post-traitements données	Non attribué	21/04/23

Prochaine réunion : 14/04/23 - 13h

◁ Annexe E ▷
Compte rendu de réunion 4

Dionisio, Frey, Billard, Mezureux

CR4 | PPII2 ▷ Compte-rendu de réunion

Motif / type de réunion : Réunion technique	**Lieu :** Salle Amphi nord
Présent(s) : Tout le monde	**Date / heure :** 13/04/2023 | 17h00 - 18h00

I Ordre du jour

1) Retour sur le fonctionnement de l'algorithme A*
2) Choix d'une heuristique
3) Choix d'une structure de données

II Informations échangées

II.1 Retour sur le fonctionnement de l'algorithme A*

- ▷ Similaire à l'algorithme de Dijsktra mais prend en plus compte de la distance entre la station visitée et la station cible. Ainsi le premier chemin trouvé est directement le meilleur et l'on ne visite pas de stations incohérentes.
- ▷ Pour choisir la prochaine station à visiter il faut utiliser une file de priorité.
- ▷ Il faudra également sauvegarder dans la distance du départ à la station visitée dans la file.

II.2 Choix d'une heuristique

- ▷ Pour prendre en compte la distance dans l'algorithme A* il faut choisir une heuristique adaptée.
- ▷ La distance de Manhattan a été évoquée mais elle n'est pas adaptée car utile seuleument si on travaille sur une grille.
- ▷ La fonction que l'on a choisit est simplement la norme 2.

II.3 Choix d'une structure de données

- ▷ La solution que l'on a choisit est d'implémenter deux tables de hach. Une faisant l'association entre l'ID d'une station et ses données utiles et l'autre faisant l'association entre l'ID d'une station et les stations accessibles depuis cette dernière.
- ▷ La seconde devra être réadaptée pour chaque voiture, deux solutions sont alors envisageables, la stocker en dure pour la voiture avec la plus grande capacité ou la recalculer à chaque fois.
- ▷ Ainsi on évite de stocker une matrice d'adjacence.

III Remarques / questions

- ▷ Le format csv est plus simple à utiliser que le format json pour le stockage des données.
- ▷ Il faudra faire des tests de performances pour statuer sur le stockage de la seconde table de hach.

IV To Do List

Tache	Responsable	Deadline
Coder A*	Corentin	19/04
Coder la table de hash pour les stations	Stan	19/04
Coder la table de hash pour les voisins	Yann	19/04

Prochaine réunion : 19/04/2023 - 20h