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