PROGRAMMATION

PROJET DE FIN D'ANNÉE

Plus court chemin sur un réseau de transports

Pierre Cheylus - Hugo Curtillet

I - Introduction

Contexte et objectifs du projet :
Ce projet de programmation a pour objectif de réaliser un ensemble de fonctions et programmes dans l'objectif de trouver le plus court chemin entre 2 points correspondant à une station d'un réseau de transports, en tenant compte des routes présentes sur le réseau étudié et du temps nécessaire pour les emprunter. Plusieurs algorithmes de recherche de chemin seront utilisés sur différents réseaux de transport tels que le réseau Parisien ou le réseau Ouest Américain par exemple. Ces algorithmes doivent pouvoir être utilisés et être fonctionnels quel que soit le réseau étudié, et quelle que soit sa taille. Nous verrons dans ce rapport tout d'abord quelles fonctionnalités ont été implantées dans notre projet et comment nous avons vérifié leur bon fonctionnement, puis nous nous pencherons sur le détail de la réalisation de ce projet, notamment la répartition du travail et les difficultés rencontrés lors du développement, avant de conclure sur ce que ce projet nous a apporté et quelles améliorations il serait possible d'ajouter pour le rendre plus complet.

II - Implantation du projet

II - A) Fonctionnalités du projet

La réalisation de ce projet a nécessité l'utilisation de structures de données variées permettant de modéliser les différents éléments nécessaires au projet.

Le premier choix pour ce projet a été de modéliser le réseau de transports par un graphe. Ainsi, chaque station sur une ligne du réseau étudié est considérée comme un nœud du graphe, et les arcs du graphe représentent les routes ou lignes présentes sur le réseau. Il a donc été nécessaire de créer une structure de données adapté à notre problématique permettant de modéliser numériquement un graphe grâce au langage C.

Pour ce faire, on utilise des fichiers texte détaillant le nom et la position de chaque station du réseau ainsi que l'ensemble des routes permettant de se déplacer d'une station à l'autre et le temps de trajet nécessaire pour les emprunter.

Dans un second temps, plusieurs algorithmes de parcours de graphe tels que le parcours en profondeur, le parcours en largeur et le parcours en largeur amélioré proche de l'algorithme de Dijkstra ont été réalisés afin de calculer le plus court chemin entre deux points du graphe, ce qui correspond à résoudre la problématique principale de notre projet.

Enfin, le programme a été retravaillé afin de prendre en compte la présence de correspondances au sein d'un réseau de transports afin de pouvoir permettre à l'utilisateur de trouver l'itinéraire à emprunter lorsque celui-ci entre le nom des stations de départ et d'arrivée de son trajet.

Les fonctionnalités suivantes sont opérationnelles dans notre projet :

1. Fonctions essentielles pour la manipulation d'une structure de données de type liste.
2. Fonctions essentielles pour la manipulation d'une structure de données de type file.
3. Fonctions essentielles pour la manipulation d'une structure de données de type tas (pour un tas minimier).

4. Création de graphe : remplissage d'une structure de données graphe à partir d'un fichier texte contenant les informations utiles sur le réseau étudié.
5. Fonction d'affichage de graphe.
6. Recherche du plus court chemin sur un réseau routier par la méthode du parcours en profondeur.
7. Recherche du plus court chemin sur un réseau routier par la méthode du parcours en largeur.
8. Recherche du plus court chemin sur un réseau routier par la méthode du parcours en largeur amélioré : Algorithme de Dijkstra
9. Reconstruction du plus court chemin à partir des informations écrites dans le graphe après utilisation d'un algorithme de recherche du plus court chemin.
10. Fonction d'affichage textuel d'itinéraire.
11. Implantation des algorithmes de recherche de plus court chemin pour un réseau présentant des correspondances.
12. Recherche du plus court chemin sur un réseau routier en entrant les noms des stations de départ et d'arrivée.

Les fonctionnalités suivantes sont manquantes dans notre projet :

1. Tracé graphique du réseau routier et des itinéraires à partir du graphe.

II - B) Tests effectués pour les fonctionnalités du projet

Pour atteindre le bon fonctionnement du projet dans son ensemble, nous avons utilisé au fur et à mesure du développement plusieurs tests. Tout d'abord, à chaque fois qu'une nouvelle fonction (ou plusieurs fonctions fonctionnant ensemble) était implantée, nous avons réalisé de simples tests de routine sur des graphes simples après compilation pour vérifier que la fonction en question était bien fonctionnelle par rapport à ce qu'elle est censée réaliser.

Voici une liste non-exhaustive des tests réalisés pour vérifier le bon fonctionnement des fonctions :

- Test de création affichage du graphe à partir d'un fichier texte.
- Test de l'algorithme de recherche en profondeur entre 2 points du graphe.
- Test des fonctions sur la structure de données liste.
- Test de recréation du plus court chemin à partir des données du graphe après exécution d'un algorithme de recherche du plus court chemin.
- Test des fonctions sur la structure de données file.
- Test de l'algorithme de recherche en largeur entre 2 points du graphe.
- Test des fonctions sur la structure de données tas (pour un tas minimier).
- Test de l'algorithme de Dijkstra entre 2 points du graphe.

Une fois que le projet était assez avancé, nous avons réalisé des tests plus complexes pour vérifier la bonne fonctionnalité d'un algorithme de recherche du plus court chemin sur des graphes simples avant de les tester sur des graphes plus complexes comme celui correspondant au réseau du métro parisien. Pour tester les différents algorithmes, on a par exemple cherché à trouver les plus courts chemins entre chaque station d'un graphe simple présentant quelques dizaines de sommets au maximum. On peut ensuite vérifier les résultats produits par les algorithmes afin d'abord de voir s'ils ont les mêmes résultats (ou du moins des résultats de même coût) et ensuite si ces résultats

sont justes en parcourant à la main les graphes étudiés. Il est ainsi raisonnable d'affirmer qu'un algorithme fonctionnant sur un graphe simple devrait fonctionner sur un graphe plus imposant comme celui du réseau de transports Ouest Américain.

Pour tester nos algorithmes, nous avons utilisé les fichiers graphe1.txt et graphe2.txt. Nous avons pour chaque algorithme cherché tout les chemins possible et comparer les résultats au seins d'un fichier text. Voici une partie du fichier text obtenu avec le fichier test_algo.c :

```
Test du parcours en profondeur :
Chemin trouvé entre les sommets 0 et 1
Voici le chemin le plus efficace
--------------------------------------------------------------------------------
Départ de Aaa de la ligne M1
Arrivée à Baa par la ligne M1
Le cout est de 5.000000

--------------------------------------------------------------------------------
Chemin trouvé entre les sommets 0 et 2
Voici le chemin le plus efficace
--------------------------------------------------------------------------------
Départ de Aaa de la ligne M1
Passer par Baa de la ligne M1
Arrivée à Caa par la ligne M1
Le cout est de 15.000000

--------------------------------------------------------------------------------
Chemin trouvé entre les sommets 0 et 3
Voici le chemin le plus efficace
--------------------------------------------------------------------------------
Départ de Aaa de la ligne M1
Passer par Baa de la ligne M1
Passer par Caa de la ligne M1
Arrivée à Daa par la ligne M1
Le cout est de 25.000000
```

FIGURE 1 – Test des algorithmes

Nous avons aussi pu tester notre code sur le métro parisien, tout s'est déroulé correctement, nous n'avons pas de doublons au niveau des stations de départ et d'arrivée et les correspondances sont réalisées correctement. Notre code fonctionne aussi sur le réseau routier américain , l'exécution du code est plus longue (que dizaine de seconde).

Nous avions aussi testé les chemins que nous utilisons avec test_chemin.c où nous créons artificiellement un chemin en modifiant les father de sommets. Nous avons ainsi pu constater le bon fonctionnement de nos fonctions de traitement de chemin.

III - Suivi du projet

III - A) Difficultés rencontrées au cours du projet

La réalisation d'un projet de recherche d'itinéraire optimal soulève plusieurs points complexes : Tout d'abord, la première difficulté rencontrée fut celle de la manipulation des structures de données. En effet, le cours de ce semestre nous a enseigné comment fonctionnaient les principales structures de données utilisées en programmation et comment réaliser celles-ci dans un environnement C, mais ce projet nous a permis de mieux comprendre comment celles-ci peuvent se révéler utiles et il a été

nécessaire de revoir le fonctionnement de chacune afin de les manipuler convenablement.

Par exemple, il a été nécessaire d'implémenter la structure d'un Graphe, ou de reconstruire la structure d'un tas pour en faire un tas minimier de sommets, ordonné selon le pcc des sommets en question plutôt que la valeur même du sommet.

Une seconde difficulté pour le projet a été la réalisation des algorithmes de recherche du plus court chemin. En effet, chacun des algorithmes proposent une méthode différente d'approche du problème, en utilisant ou non une structure de données différente à chaque fois - qui peuvent déjà poser un problème, comme décrit précédemment - ce qui nécessite de comprendre l'algorithme théorique afin de l'adapter en langage C pour l'utiliser avec les structures de données introduites. Or ces algorithmes ne sont pas toujours évidents à appréhender, plus particulièrement dans le cas de l'algorithme de Dijkstra utilisant les tas minimier, dont l'implantation a nécessité plus de réflexions que pour le parcours en profondeur ou en largeur.

De plus une difficulté supplémentaire qu'il a fallu surmonter est dans l'extension de recherche d'itinéraire à partir des noms de stations. Tout d'abord, puisque le graphe organise les stations par numéro et non par nom, il a fallu trouver une solution afin d'identifier le numéro correspondant aux stations de départ et d'arrivée recherchées. Pour ce faire, nous avons d'abord simplement fait une recherche dans la structure du graphe en parcourant les sommets un par un jusqu'à tomber sur un possédant le nom choisi par l'utilisateur. Cependant, une nouvelle question s'est alors posée : celle des correspondances. Lorsque l'utilisateur entre le nom d'une station étant présente sur plusieurs lignes, il est important de choisir la bonne ligne à emprunter parmi les lignes possibles pour obtenir le trajet minimal. Pour ce faire, le travail en amont d'identification des correspondances nous a permis de créer un algorithme empruntant toujours la ligne optimale parmi les lignes présentes sur une même station en essayant chacune des stations possédant le même nom et en trouvant celle possédant le chemin le plus court.

Enfin, la dernière difficulté rencontrée fut la question des graphes possédant des correspondances, pour ensuite pouvoir implémenter une recherche d'itinéraire par le nom d'une station. La présence de ces stations faisant le lien entre plusieurs lignes du réseau de transport pose la question de leur représentation dans la structure de données qu'est le graphe. La solution apportée par les fichiers texte de description du graphe représentant le réseau est d'introduire des arcs entre les différentes points d'une même station, dont le coût est fixe afin de les identifier. Dans notre algorithme qui réalise une recherche en fonction du nom, nous avions décidé de prendre pour numéro de sommet de départ et d'arrivée n'importe quel sommet ayant le nom correspondant. Afin d'éviter les liaisons inutiles au sein de la station de départ ou d'arrivée, nous vérifions que le nom de la deuxième station est différent de celui de la première. Si ça n'est pas le cas, nous apportons les modifications nécessaires : avancer le chemin d'un sommet et réduire les pcc de tous les sommets suivants du cout de la correspondance inutile. Nous réalisons cette étape autant de fois que nécessaire, même si on ne peut normalement réaliser qu'une seule correspondance inutile. Nous vérifions la même condition pour la station d'arrivée et la station la précédant, ainsi si les noms sont similaires, nous retirons le dernier maillon du chemin puisque nous sommes déjà arrivé à destination.

Une difficulté que nous n'avons pas pu surmonter est l'affichage graphique. Nous avons rencontré des problèmes de linkage avec le SDL de Phelma et les solutions apportées pas le professeur ne fonctionnaient malheureusement pas sous la distributions Linux utilisée (Manjaro). Nous n'avons donc pas pu réaliser la partie graphique.

III - B) Répartition du travail et planning

Le contexte particulier du confinement pour la réalisation de ce projet nous a demandé d'organiser le travail à distance via Discord, GitHub et Atom Teletype.

Dans un premier temps (Mars/Avril), nous avons chacun travaillé de notre côté sur le début du projet durant les séances puis nous avons mis en commun notre travail en dehors du temps de projet afin d'apporter nos réflexions personnelles sur le sujet et d'identifier les points plus complexes du projet. Nous avons ainsi réalisé les parties structures de donnée Graphe et remplissage du graphe à partir d'un fichier texte décrivant les noeuds et arcs composant le graphe.

Dans un second temps (Mai), nous avons profité du déconfinement et de notre proximité géographique afin de se retrouver ensemble pour travailler 2 à 4h par jour pendant 1 semaine sur le projet afin d'avancer plus efficacement. Nous avons ainsi décidé de travailler ensemble sur les fonctionnalités portant sur les algorithmes de recherche du plus court chemin et de reconstruction du plus court chemin, tandis que nous nous sommes réparties certaines fonctionnalités comme l'identification des correspondances dans les algorithmes, la recherche d'itinéraire par nom (Hugo) ou la réalisation des tests (unitaires et complexes, Pierre).

Voici un récapitulatif des tâches effectuées par chacun (se référer au numérotage des fonctionnalités du projet dans la partie II-A) :

- Fonctionnalités déjà réalisées auparavant (Cours de structures de données) : 1) et 2).
- Fonctionnalités réalisées à deux : 3), 6), 7), 8), 9), 10).
- Hugo : Fonctionnalités 4), 5), 11) et 12), recherches sur la partie graphique et arrangement général du code.
- Pierre : Réalisation des tests sur les différentes fonctionnalités et rédaction du rapport de projet.

III - C) Bilan du projet et améliorations possibles

Ce projet nous aura permis d'en apprendre plus sur la manipulation de structures de données et de comprendre l'étendue des possibilités qui nous sont offertes par celles-ci. Les variables internes et la définition de sous-structures permettent de stocker une très quantité d'informations dans une seule et même structure, comme c'est le cas pour la structure de graphe ici.

De plus, nous avons pu observer la versatilité de structures de données plus simples comme les listes, les tas et les files, permettant de simplifier grandement les algorithmes complexes parcourant une structure de donnée comme un graphe, plutôt que de faire des appels récursifs d'une même fonction comme pour le parcours en profondeur, qui peut engendrer des problèmes lors d'une recherche dans un graphe de taille importante.

Un autre point sur lequel le projet nous a apporté est l'organisation du travail en équipe dans un projet de programmation. Nous avions pu par le passé réaliser un projet à distance mais les conditions de travail étaient moins complexes que celles imposées par le confinement, l'avancée du projet a pu être plus continue, tandis que nous avons ici opté pour un travail en commun à partir du déconfinement, ce qui nous a forcé à concentrer notre travail sur une plus courte période de temps et ainsi devoir mieux s'organiser.
Enfin, ce projet nous a permis d'en apprendre plus sur les algorithmes de parcours de données complexes comme pour le parcours de graphe par exemple, et ainsi d'en apprendre plus sur la nature des graphes.

Cependant, nous avons pu voir qu'il existe des limites à ce projet. En effet, la base de ce projet est d'utiliser les graphes des réseaux de transport afin de pouvoir déterminer le plus court chemin entre deux points, mais celui-ci ne prend en compte dans le coût des arcs que le temps moyen de parcours entre les deux points sans se soucier de l'heure de départ des différentes lignes présentes sur le réseau. En effet, si le chemin proposé par le programme est le plus court sur le papier mais que la ligne que l'utilisateur doit emprunter est indisponible à l'heure d'utilisation du programme, l'utilisateur ne pourra pas l'utiliser à bon escient. Il serait ainsi intéressant d'intégrer au programme un second fichier décrivant les horaires de chaque ligne composant le réseau de transports afin de pouvoir intégrer l'heure actuelle et les horaires du réseau au calcul du plus court chemin.

De la même façon, le programme tel que présenté ne prend pas en compte la réalité du trafic sur le réseau de transports. Un réseau de transports complexe tel que celui de la ville de Paris peut avoir des problèmes multiples : bouchons sur les voies empruntées par les bus, ralentissement d'un RER, travaux entre deux stations de métro, etc. Le programme actuel prends un problème simplifié où tous les arcs composant sont empruntables à tout moment de la journée, avec un coût constant. On pourrait imaginer un système remédiant à ce problème en passant par des requêtes HTML : la compagnie gérant le réseau de transports étudié aurait un serveur Web sur lequel serait hébergé une page contenant le fichier texte utilisé pour construire le graphe. Le fichier texte pourrait ainsi être actualisé par un système externe mesurant le temps de parcours entre deux stations pour mettre à jour le coût de l'arc correspondant à la liaison entre ces deux stations dans le fichier, ou supprimer des arcs lorsque ceux-ci sont momentanément inutilisables comme pour l'exemple des travaux par exemple. Les itinéraires calculés par le programme pourraient ainsi correspondre à la réalité du réseau de transports étudié.

Enfin, une version graphique du programme proposant à l'utilisateur de choisir directement son départ et sa destination via une interface graphique ou via des zones de texte et affichant l'itinéraire ainsi calculé permettrait une utilisation plus ergonomique d'un outil de calcul de plus court chemin afin de lui trouver une application pratique.

L'ensemble de ces améliorations permettrait ainsi à ce programme de constituer une base solide pour une application mobile donnant des indications d'itinéraire à un usager d'un réseau de transports en commun.

IV - Conclusion générale

Pour conclure, ce projet de recherche du plus court chemin sur un réseau de transports nous a permis de comprendre la logique mathématique derrière les solutions utilisées par les sociétés gérant les réseaux de transports en communs pour leurs applications mobiles mais aussi de comprendre comment celles-ci les implémentent réellement en programmant les algorithmes et fonctions essentielles à leur fonctionnement en C. Nous avons pu atteindre nos objectifs pour ce projet tout en nous organisant en deux temps, d'abord à distance puis en travail en commun, en utilisant trois algorithmes de recherche de plus court chemin différents pour des fichiers décrivant des réseaux de transports complexes tels que le réseau Parisien ou le réseau Ouest Américain. Nous avons ainsi pu trouver des solutions faces aux difficultés rencontrées au cours du projet avant de vérifier le bon fonctionnement de ces différentes fonctionnalités en les testant. Enfin nous avons pu étendre notre réflexion afin de trouver quelles améliorations seraient intéressantes pour ce projet.