École Polytechnique de Montréal
Département de Génie Informatique et Génie Logiciel

LOG2810

STRUCTURES DISCRETES

Hiver 2017

TP1 : Graphes

Remis par :

Matricule	Prénom & Nom
1828776	Tamer Arar
1827022	Hon-Leung Éric, Chao
1802395	Guy Frank Essome Penda

À :

David Johannès

Le 1er mars 2017

Introduction

Dans le cadre de ce travail pratique, nous avons dû appliquer divers concepts théoriques de structures discrètes portant sur les graphes tels que les arcs, les nœuds, les graphes simples les matrices adjacentes, les sous-graphes et les graphes connexes. Nous avons aussi dû adapter l'algorithme de Dijkstra afin de pouvoir concevoir une application permettant d'optimiser le parcours d'un explorateur dans un jeu du type Pokémon Go. Ce jeu permet à l'utilisateur d'attraper des Pokémons sur sa route, de visiter des arènes et des pokéstops afin de récolter des objets de valeurs. Chacun de ses endroits représente un sommet sur un graphe. Chacun de ces sommets possède un gain correspondant à son type. La distance entre 2 objets (pokéstops, arènes ou pokémons) est déterminée par un arc. Ces sommets et ces arcs constituent alors un graphe représentant le monde de Pokémon. Aussi, chacun de ses objets peut être visité un nombre limité de fois dans la journée. Il faudra alors optimiser le code pour qu'il tienne en compte ceci. Le temps de rechargement est considéré en mètre (distance marché). L'arène est le seul objet qui ne peut être visité qu'une seule fois durant la journée.

Lors de la réalisation de l'application, il a fallu implémenter quatre composants :

- Composant 1. Une méthode "creerGraphe()" qui permettra la lecture d'un fichier fourni contenant les informations sur les routes et les sommets. Le nom du fichier est passé en paramètre.
- Composant 2. Une méthode "lireGraphe()" qui permettra d'afficher le graphe lu du fichier.
- Composant 3. Une méthode "plusCourtChemin()" qui, pour un point de départ et pour un gain voulu passé en paramètre, générera le chemin ayant la distance minimale pour y parvenir (Inspiré de Dijkstra). Permet aussi l'affichage de ce chemin.
- Composant 4. Une fonction "plusGrandGain()" qui, pour un point de départ et pour une distance voulue passée en paramètre, générera le chemin ayant le gain maximal pour y parcourir (Inspiré de Dijkstra). Permet aussi l'affichage de ce chemin.
- Composant 5. Une interface qui affichera un menu :
  a) Mettre à jour la carte.
  b) Déterminer le plus court chemin.
  c) Déterminer le plus grand gain.
  d) Quitter.

Présentation de Travaux

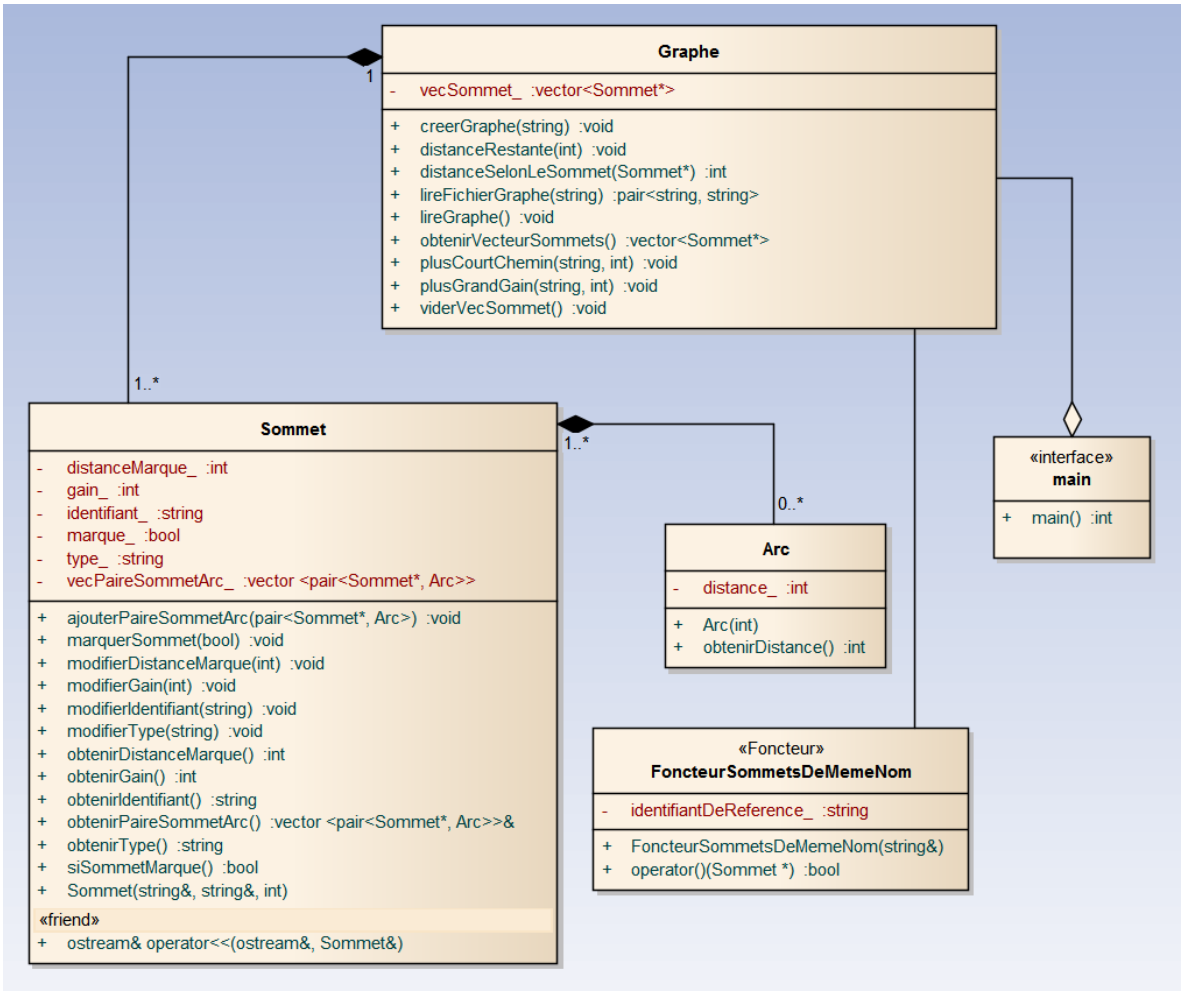

Figure 1 : diagramme de classes de la solution pour le parcours du graphe.

Figure 2 : interface de la solution pour le parcours du graphe.

La classe graphe possède un vecteur de pointeurs de sommets. Chacun de ces sommets possède un vecteur dont ses éléments sont un voisin et l'arc (distance) le reliant.
De plus, la classe foncteur permet d'itérer sur le vecteur de sommet pour rechercher un sommet de même nom à certain endroit dans le code source.
L'interface est effectuée en programmation procédurale dans le fichier main.cpp.
Le chemin construit est situé dans la classe Graphe, dans ses méthodes mêmes, dépendamment de l'option voulue (gain maximal ou distance minimale).

Tout d'abord, le programme doit lire les sommets et leurs voisins à partir d'un fichier déposé dans le même répertoire du code source. La méthode lireFichierGraphe s'en occupera, tandis que la méthode LireGraphe affichera le graphe lu.

Ensuite, concernant la méthode plusCourtChemin, parcourir le graphe et trouver la distance la moins élevée pour un gain voulu a été un succès. Ainsi, lorsque le sommet de départ est choisi, nous procédons à la visite de tous ses sommets voisins. Notre sélection du prochain voisin sera basée sur le rapport distance / gain de celui-ci. En effet, un voisin loin de 40 mètres et ayant un gain de 500 (ratio de 40/500 = 0.08) est tout de même mieux que 40 / 200 = 0.2. Plus le rapport est petit, mieux c'est. Après, lorsque le sommet adjacent ayant le plus petit rapport distance / gain est choisi. Nous le marquons, c'est-à-dire qu'il est désactivé du graphe et ne pourra être accédé que, lorsque le cumul de la distance future que l'on parcourra fera en sorte qu'il soit démarqué. Les méthodes distanteRestante et distanceSelonLeSommet sont chargées de ça.
Après avoir marqué le sommet, nous utilisons la méthode distanceRestante pour mettre à jour tous les sommets déjà visités et ajuster la distance qui leur est restante à parcourir avant qu'ils soient démarqués. Une fois démarqué, nous pouvons tenir compte une nouvelle fois de ce sommet lors de la sélection du candidat avec le meilleur rapport distance/gain. En nous assurant de mettre le nouveau sommet sur le graphe de chemin, nous recommençons à chercher le meilleur voisin pour chaque voisin visité en ne s'arrêtant qu'après avoir atteint le gain voulu.

De même, la méthode plusGrandGain suit exactement la même logique, sauf pour un point. On ne s'arrête qu'après avoir atteint, au plus, la distance donnée en paramètre.
Finalement, nous avons procédé à faire une interface pour permettre, à l'utilisateur, de sélectionner, de rentrer les données exigées et de trouver soit la plus petite distance pour un plus grand gain, soit le plus grand gain pour une distance maximale.

Enfin, il faut compiler et exécuter sous Windows pour que le programme marche correctement, car on fait appel à la librairie de *windows.h* pour l'interface.

Difficultés rencontrées lors du TP

Lors de la réalisation du travail pratique, nous avons rencontré divers types de problèmes. Dans l'aspect organisationnel, nous avons dû nous trouver une méthode pour répartir les tâches et échanger nos travaux. L'entretien du code fut fructueux, car il a fallu inventer un système de commentaire afin d'avertir les coéquipiers des modifications apportées par un membre afin qu'il n'ajoute pas un concept logiciel qui pourrait entrer en conflit avec d'autres éléments du code.
Il fallait ainsi se trouver un moyen de se comprendre après l'implémentation de plusieurs de ligne de code. De plus, le code grossissant de plus en plus, il a fallu trouver un moyen de se retrouver dans cet amas afin de pouvoir réajuster notre concentration en écrivant des commentaires et en encapsulent des blocs de codes en méthodes.

Dans l'aspect technique de la programmation, nous avons eu de la difficulté à apercevoir la meilleure approche pour trouver notre solution. En effet, nous avons tout d'abord essayé d'appliquer l'algorithme de parcours de profondeur (depth-first search) pour trouver le chemin le plus court, mais celui-ci n'était pas compatible avec l'ajout du système de gain. De plus, il cherchait tous les chemins possibles et cela pouvait prendre des heures avant qu'il s'arrête. Nous avons alors tenté d'appliquer un autre algorithme. Ce dernier était l'algorithme de Floyd-Warshall, bien qu'il pût nous apporter une solution convenable, il prenait trop de temps à implémenter, car il fallait, une fois la table des plus courts chemins entre chaque paire de sommets établis, trouver petit à petit le meilleur en rapport distance / gain. Ainsi, nous avons dû nous rapatrier sur un algorithme s'inspirant de celui de Dijkstra, qui, bien qu'étant très difficile à imaginer, reste dur à implémenter en c++. On a perdu beaucoup de temps pour le comprendre, alors que la solution était plus simple que prévu. Aussi, l'algorithme achevé s'exécute plus rapidement que les deux précédents. De plus, nous avons aussi eu des problèmes à tester la validité de nos fonctions, car l'ensemble de données, étant très vaste, était difficile à appréhender et ne nous donnait pas une certitude en la validité de notre fonction. Ainsi, nous avons dû utiliser une base de données plus simple afin de pouvoir mieux appréhender nos méthodes et ainsi améliorer le prototype pour finalement l'appliquer à plus grande échelle. Les algorithmes étant équivoques à implémenter, nous avons dû nous pousser à la limite de nos connaissances logicielles et nous avons appris différentes approches conceptuelles afin de résoudre certains défis tels que l'utilisation de méthode de parcours de graphe pour générer la liste des chemins possibles.

Ensuite, le temps était un enjeu important. La nécessité de plus de temps a vite été ressentie. La pression rencontrée de la part de ce facteur temps nous a permis de travailler davantage et plus sérieusement.

Aussi, il y avait le défi limitant l'application direct d'algorithme. Il a fallu arranger l'algorithme de Dijkstra afin de pouvoir l'adapter à nos problèmes exigeant un système de gain et de retour sur les sommets. Ainsi, nous avons dû trouver un moyen efficace pour générer les bons chemins.

Finalement, un des problèmes était de gérer les entrées de l'utilisateur. Il fallait s'imaginer à la place de l'utilisateur et bloquer toute tentative de mauvaises entrées qui pouvaient faire planter notre programme.

Conclusion

Ce laboratoire nous a permis de percevoir l'utilité de concepts mathématiques abstraits appliqués au quotidien, nous permettant en autre temps de prendre conscience des mathématiques dans le travail d'un ingénieur en informatique/logiciel. Ce travail touche à l'essence de l'ingénierie, car il nous donne l'occasion de trouver un moyen d'implémenter des notions théoriques en pratique. Il nous permet de travailler en équipe et d'appliquer les notions vues dans les cours d'habiletés personnelles et relationnelles. Il nous a aussi permis d'appréhender la complexité des différentes applications que nous utilisons au quotidien, nous donnons une meilleure impression du travail d'ingénieur logiciel. De plus, comme chaque projet de programmation, il nous a poussés aux limites de nos connaissances en programmation, nous forçons à apprendre de nouveaux concepts, nous permettant de voir de nouvelle manière d'implémenter les choses. Nous avons ainsi, de nouveaux patrons de conception à appliquer, nous donnons une meilleure approche pour les futurs problèmes à résoudre.

Pour le projet travail pratique, nous nous attendons à un travail stimulant qui nous permettra à nouveau d'appliquer des notions en mathématiques à des exemples plus concrets à la réalité afin d'approfondir nos connaissances sur les structures discrètes et d'y voir une plus grande utilité afin de nos permettre de trouver un rapprochement futur pour un travail dans le futur. Nous pensons que le travail pratique doit aussi nous pousser aux limites de nos connaissances afin de pouvoir nous aider à progresser avant la rencontre d'un stage qui exploitera une démarche d'apprentissage et de travail semblable.

Pour conclure, le travail était un défi de taille qui nous a permis de nous transcender sous tous nos aspects académiques nous permettant d'utiliser notre créativité afin d'aboutir à une solution.