École Polytechnique de Montréal
Département de génie informatique et génie logiciel

LOG2810
Structures Discrètes

TP1 : Graphes

Soumis par :
Alexandre Hua (1795724)
Ayoub El Asry (1800595)
Gabriel Hélie (1791764)

Groupe 01

28 octobre 2016

Table de matière

Introduction

L'objectif de ce TP est d'utiliser les notions de graphes appris et de les appliquer dans un contexte concret pour résoudre un problème informatique.

La situation que nous sommes confrontés est la suivante : nous avons une automobile qui parte d'un certain point de départ et qui veut se rendre à un point de destination dans un temps minimal. Cependant, il ne suffit pas de trouver le chemin le plus court. En effet, Il faut aussi considérer le type de carburant que l'automobile utilise afin de trouver un chemin qu'il puisse s'arrêter pour faire le plein.

Pour répondre aux exigences du contexte, notre application doit être capable de suggérer le chemin le plus court à prendre tout en incluant des stations de plein correspondant au type de carburant de l'automobile dans le trajet. La carte routière, qui doit être fournie, peut être modélisée à l'aide d'un graphe. Les routes sont représentées par des arcs et les points d'intersection sont représentés par des sommets. Toutes les routes sont de sens unique et chacun des points d'intersection peuvent avoir des stations de plein ou pas. Ces dernières peuvent subvenir les automobiles en essence, en électricité ou les deux, ce qu'on appelle une station hybride.

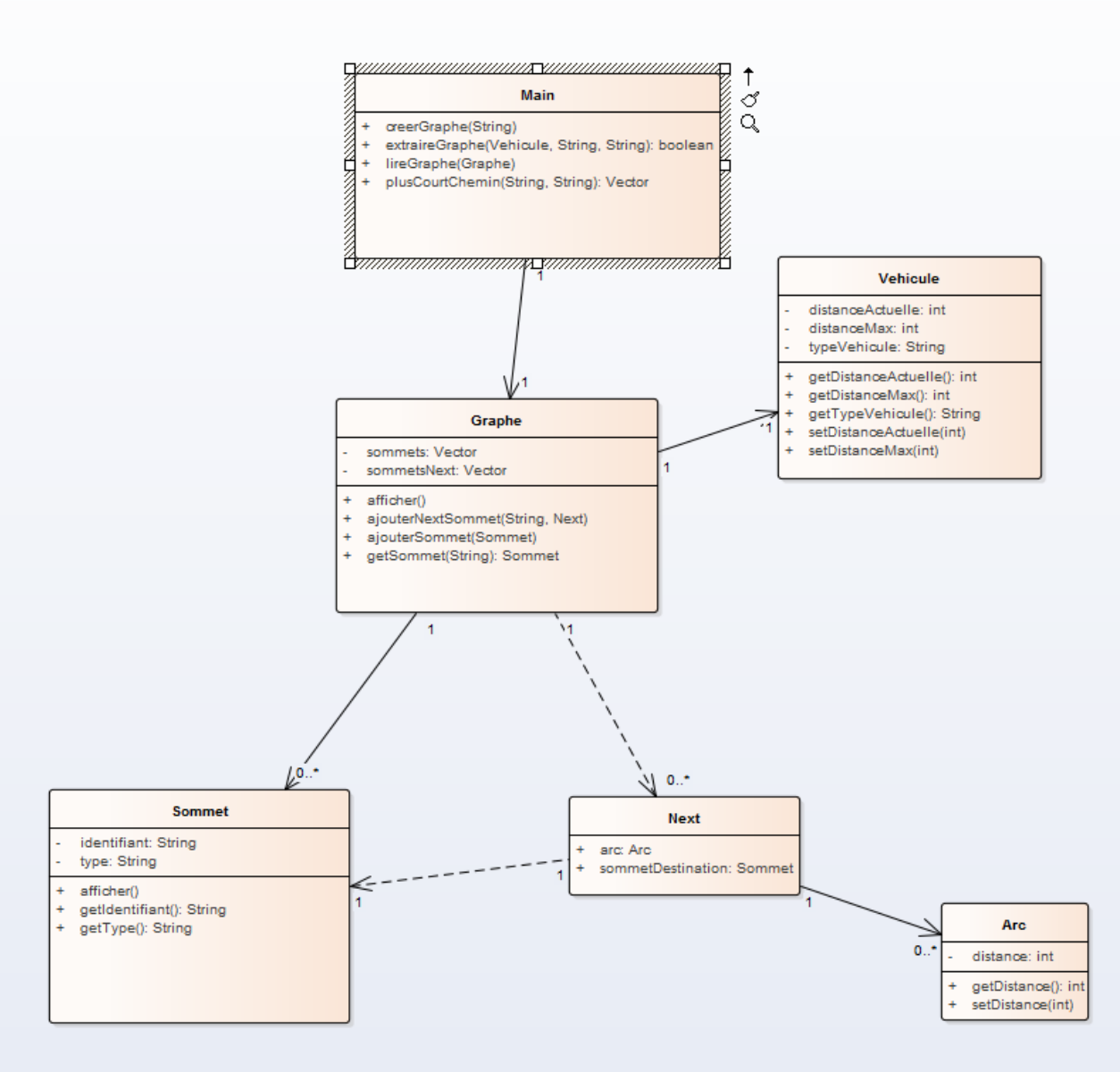
Notre solution

L'approche de notre solution est de sous diviser les classes de façon à obtenir des classes concrètes afin qu'on puisse mieux visualiser les différents éléments de notre graphe. Les trois classes principales sont les suivantes : un graphe qui représente la carte routière, un sommet qui représente les stations dans la carte et un arc qui représente la distance entre deux stations. D'après nos recherches, nous avons trouvé que la façon la plus optimale de représenter un graphe est d'utiliser la structure de liste adjacente, c'est-à-dire que chaque sommet contient une liste de sommet vers où il pointe.

Une des façons de résoudre notre problématique est de subdiviser la tâche en deux parties. La première partie consiste à créer une fonction extraireGraphe qui permet d'extraire un sous-graphe où notre véhicule peut rouler. La deuxième partie consiste à appliquer l'algorithme de Dijkstra sur le graphe qu'on vient d'extraire et de trouver le plus court chemin par la suite.

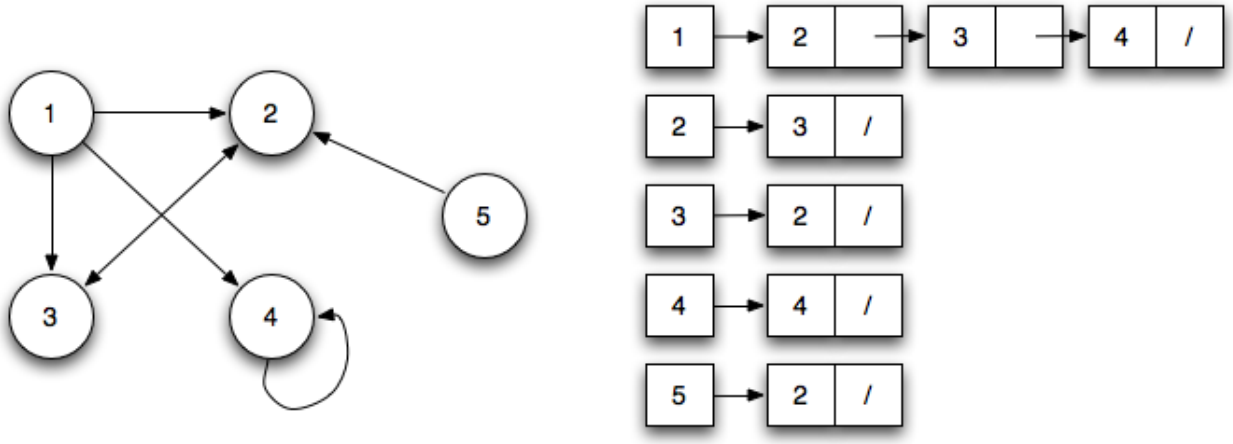
La fonction extraireGraphe est une fonction récursive qui retourne un graphe complet des chemins possible que le véhicule peut emprunter. La récursivité permet de parcourir les nœuds voisins de chaque nœud et d'effectuer un test de validation. Le test de validation consiste à vérifier si un chemin est empruntable entre deux sommets.

Diagramme de classe



Difficultés rencontrées

Une des plus grandes difficultés que nous avons rencontrées est la modélisation du problème. En effet, nous avons pensé à plusieurs façons différentes pour structurer les classes et les relations entre elles. Il existe plusieurs façons de représenter un graphe en c++. On peut faire un vecteur qui contient les chemins : source, distance et destination, utiliser une matrice d'adjacence, faire des vecteurs d'arc qui pointent vers des sommets etc. Pour déterminer la structure idéale il faut tenir compte : de la sauvegarde des données textuelles, pouvoir le parcourir aisément et pouvoir effectuer des opérations dessus. Donc nous avons opté pour une structure de liste adjacente.



http://ycpcs.github.io/cs360-spring2015/lectures/images/lecture15/adjlistexample.png

Une autre difficulté que nous avons rencontrée est dans l'implémentation en code de notre solution. Dans l'implémentation du parcours des sommets du graphe, la partie qui a été compliqué pour nous est le fait qu'un sommet peut contenir un autre sommet. En effet, un sommet peut être relié à plusieurs autres sommets et ces derniers aussi peuvent être reliés à encore d'autres sommets, donc il était impossible de savoir en avance le nombre d'itération. La solution que nous avons trouvée est d'utiliser la récursivité qui réappelle notre méthode. Par contre, l'implémentation de la récursive a été elle-même compliquée. En effet, il y avait plusieurs choses que nous devions tenir compte, tel que l'état des variables des niveaux supérieurs. Par exemple, nous avons utilisé la récursivité dans notre méthode extraireGraphe pour parcourir les sommets reliés avec le sommet de départ et il était important de tenir compte de la variable autonomie. Chaque fois que le véhicule passe dans un prochain sommet, il faut diminuer son autonomie par la distance de l'arc qu'il a parcouru. Cependant, lorsque la méthode retourne la valeur et qui passe au prochain élément, il faut restaurer la variable de l'autonomie au bon niveau.

À part les problèmes en lien avec le contexte du travail pratique et l'implémentation de code, nous avons également éprouvé un peu de difficulté avec l'organisation du travail et les temps de

rencontre. En effet, par le fait que nos horaires de cours sont très différents, il était difficile de trouver du temps pour travailler tous les 3 ensembles. Il nous arrivait souvent de rester tard après les cours afin de pouvoir se réunir pour travailler. Quant à l'organisation, nous avons parfois eu des difficultés à gérer la version de notre projet. Étant donné que nous ne nous rencontrons pas souvent, nous travaillions chacun de notre côté et au moment de la mise en commun, il y avait parfois des conflits à résoudre. De plus, lorsque nous testons du code, par crainte que le code soit défectueux et qu'il causerait des problèmes d'incompatibilité par rapport aux versions des autres membres de l'équipe, nous gardons plusieurs versions temporaires locales, ce qui devenait de plus en plus à gérer et apporte parfois de la confusion.

Conclusion

En appliquant les notions théoriques dans un travail pratique concret, ça nous a permis de renforcer notre compréhension sur les graphes, ainsi que son domaine d'application. Bien que ce ne soit pas encore un exemple très réaliste, nous avons pu comprendre comment le fait de modaliser ce type de situation en se servant des graphes puissent nous être utile pour la résolution d'un problème, tel que de trouver le chemin le plus court.

Si nous avons l'opportunité de faire un autre travail pratique, nous s'attendons à recevoir un autre problème tiré d'une situation de tous les jours qui utilisent les notions du cours pour le résoudre, afin d'avoir une meilleur idée comment les connaissances que nous acquérons puisse nous être utile pour faire face ces problèmes.