LOG2810
STRUCTURES DISCRÈTES

TP1 : GRAPHES

Abderrahim Ammour 1924705
Abdelkader Zobiri 1891451
Ikhelef Hanane 1891934

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

REMIS LE 5 NOVEMBRE 2019

Introduction:

La théorie des graphes vue en cours représente une partie vitale des mathématiques et informatique. Les graphes sont constitués par des sommets et des arcs reliés, il existe plusieurs algorithmes élaborés afin de parcourir ces graphes pour optimiser un trajet souhaité. L'objectif de ce travail pratique est de permettre à une flotte de drones de pouvoir ramasser des paquets le plus rapidement et efficacement possible en appliquant les notions théoriques sur les graphes apprises en cours, sur des cas concrets et réels.

Pour ce faire, nous allons tout d'abord créer notre graphe à partir d'un document texte qui va contenir tous les sommets et les arcs. Nous avons comme information le coût des arcs et le nombre d'objets de type A, B ou C dans chaque sommet. Nous devons recueillir les commandes d'un utilisateur et les garder en mémoire. L'objectif est de calculer le chemin le plus court qu'un robot peut parcourir pour ramasser l'entièreté d'une commande passer par l'utilisateur. Pour ce faire nous utilisant utilisant l'algorithme Dijkstra pour déterminer le parcours le plus efficace à prendre, revenir au point de départ le plus rapidement possible. Ensuite, nous déterminons le robot le plus efficace parmi les 3 types des robots qui sera apte à réaliser la tâche plus aisément. Enfin, nous allons afficher le graphe du meilleur parcours, ainsi toutes ces étapes seront présentées dans une interface d'affichage détaillé incluant les objets pris lors du parcours.

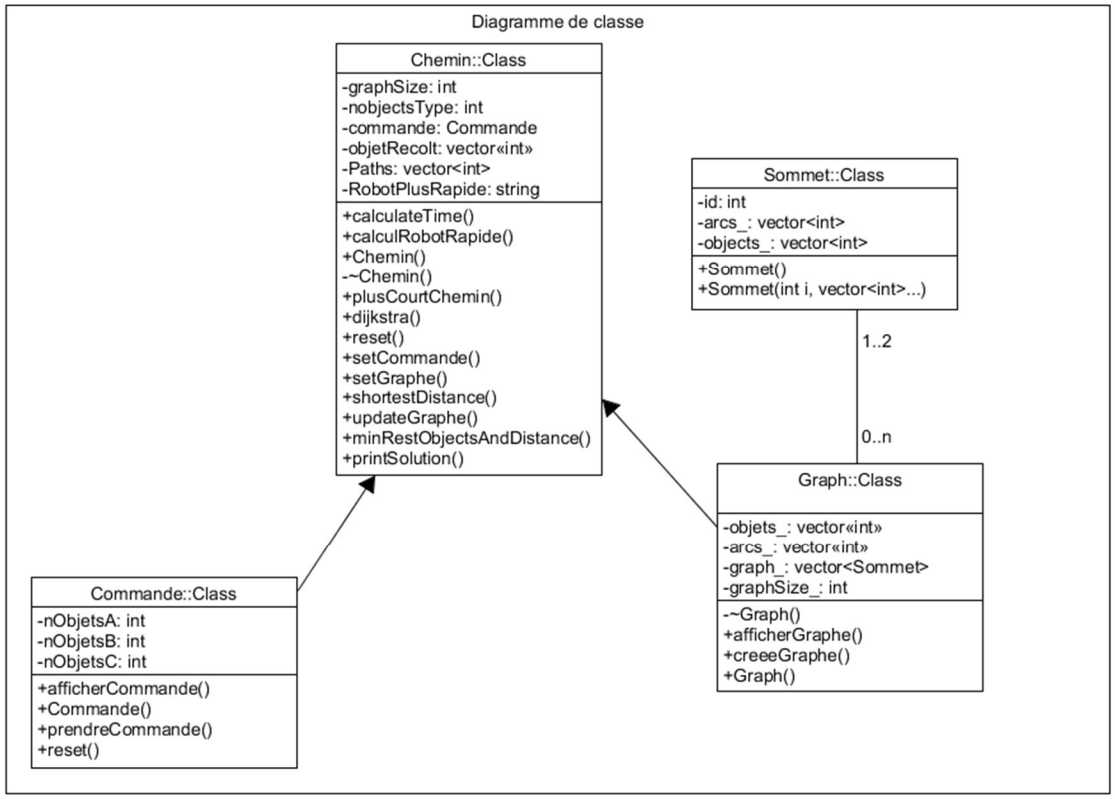
Présentation des travaux

Lors de l'élaboration de l'application, nous avons travaillé avec le langage de programmation C++ en utilisant une approche orientée objet puisque nous avions tous une bonne connaissance de ce langage utilisé auparavant dans d'autres cours. Notre solution contient donc les quatre classes suivantes : Chemin, Graph, Commande et Sommet.

Structure du code:

Nous avons décidé d'utiliser la programmation orientée objet pour réaliser ce travail, car le travail consiste à différentes fonctionnalités qui font fonctionner le code à la fin.

Figure 1 : Diagramme de classes de la solution



Détails sur les classes utilisées :

Classe Chemin:
Elle représente la classe initiale qui calcule le chemin le plus court en utilisant l'algorithme Djikstra, ainsi que calculer le temps pris par le robot pour aller chercher la commande. La classe Chemin contient également plusieurs méthodes qui fonctionnent ensemble pour atteindre notre objectif. Les méthodes clés de notre classe sont:

plusCourtChemin :
Cette méthode qui retournera et affichera également le plus court chemin pour ramasser la commande. Cette méthode se base sur l'algorithme de Dijkstra. Nous avons d'abord calculé les chemins les plus courts pour arriver à chaque sommet et au fur et à mesure on calcule les objets qu'on peut ramasser. À la fin, un tableau avec tous les noeuds accompagnés avec leurs chemins et les objets ramassés sera créé et on choisit le sommet qui a le chemin le plus court et qui satisfait à notre commande. Si la commande a été satisfaite, on arrête la recherche et on revient à notre Sommet initial. Sinon on continue à chercher les objets en appliquons le même principe jusqu'à arriver à satisfaire notre commande.

calculRobotRapide:

Cette méthode vérifie le poids total de notre commande et l'associé au robot le plus approprié. Si la commande dépasse 25 kg, aucun robot n'est associé et le chemin est impossible.

calculateTime:
Cette méthode calcule le temps pris par le robot en appliquant la bonne formule qui change en fonction du robot choisi au début.

Classe Commande:
Cette classe représente la structure de notre commande. elle contient trois attributs qui représentent les nombres de chaque type d'objet demandé et deux fonctions:
La fonction ***prendreCommande***() consiste à prendre la commande saisie par l'utilisateur cela dit les nombres d'objets rentrés de chaque type que le robot devra aller chercher ainsi qu'afficher la commandent. La fonction ***afficherCommande***() sert à afficher la commande en mémoire de l'utilisateur.

Classe Graphe:
Cette classe joue le rôle de notre base de données. Nous avons également utilisé le modèle vu en classe pour représenter les relations entre les sommets (matrice d'adjacente) et comme il s'agit d'un graphe value. Nous avons choisi cette notation pour interpréter le stock dans notre matrice:

0: pas d'arc entre A et B
x (entier non nul): la distance entre A et B.

Tout d'abord on fait la lecture du fichier qui nous permet ensuite à **créer le graphe** selon la fonction **creeGraphe(),** de plus que la fonction **afficherGraphe**() existe dans la classe graphe aussi qui permet d'afficher le graphe selon le format de l'énoncé cela dit l'affichage de chaque noeud avec ces voisins ainsi que les distances entres ces les voisins et le noeud.

Classe Sommet:
La classe Sommet contient notre vecteur des sommets, qui contient les arcs et les objets correspondants ainsi que la fonction d'affichage.

Difficultés rencontrées

L'organisation du travail:
Avant de commencer le travail, nous voulions nous assurer que le travail est bien clair et compréhensible pour tous les membres de l'équipe ainsi que définir clairement les objectifs du travail, car cela nous facilitera les divisions des tâches entre les membres. Ensuite, nous avons divisé les tâches équitablement entre les membres de l'équipe en nous assurant que chaque personne comprend bel et bien le travail à effectuer ainsi que l'objectif à atteindre.

Le manque d'information:

Après avoir lu le travail et la consigne, nous avions plein de questions concernant le sujet adopté ainsi que les travaux à effectuer. On a donc posé ces questions au chargé de laboratoire afin de clarifier certaines parties.

La communication dans le groupe:
Pour ce faire, on a utilisé slack pour communiquer et planifier nos rencontres selon nos disponibilités.

Difficulté de fusion du code et d'intégration:
Afin de fusionner notre bout de code tous ensemble, on a dû déboguer tous ensemble le travail final et le tester pour s'assurer que le travail est fonctionnel.

Conclusion

À la fin de ce laboratoire, nous avons réussi à créer un graphe de zéro et faire parcourir les robots dans ce dernier en prenant le chemin le plus court possible dans un temps très optimal et également prendre n'importe quelle commande saisie et l'appliquer sur notre algorithme sans avoir de difficultés ainsi qu'afficher le graphe et gérer tous les cas de bord. Ce travail nous a permet de mettre en pratique toutes les notions théoriques sur les graphes que nous avons vus en cours. Nous apprécier faire ce laboratoire puisque travaille qui nous a demandé de réfléchir sur la conception et la réalisation d'une solution à un problème courant.