LOG 2810
STRUCTURES DISCRETES

TP1 : GRAPHES

Session	Automne 2019
Pondération	10 % de la note finale
Lieu de réalisation	L-4708
Taille des équipes	3 étudiants
Date de remise du projet	**5 novembre 2019**
Directives particulières	Soumission du livrable par moodle uniquement (https://moodle.polymtl.ca).
	Toute soumission du livrable en retard est pénalisée à raison de 10% par jour de retard.
Les questions sont les bienvenues et peuvent être envoyées à: Stéphane Burwash (stephane.burwash@polymtl.ca), Jean-Philippe Anctil (Jean-Philippe.Anctil@polymtl.ca), Paulina Stevia Nouwou Mindom (Paulina_stevia.nouwou_mindom@polymtl.ca),	

1 Connaissances requises

- Notions d'algorithmique et de programmation C++, Python, Java ou autre (voir chargé).
- Notions de théorie des graphes.

2 Objectif

L'objectif de ce travail pratique est de vous permettre d'appliquer les notions théoriques sur les graphes que nous avons vues en cours, sur des cas concrets mais hypothétiques, tirés de votre quotidien. À cet effet, il est question dans ce travail de permettre à une flotte de drones de pouvoir ramasser des paquets le plus efficacement possible.

3 Mise en situation

Une compagnie cherche à diminuer leur coût de main d'œuvre en utilisant des robots au lieu d'employés pour préparer les commandes des clients. Elle vous engage afin que vous optimisiez le trajet de leurs robots. Vous contrôlez un seul robot à la fois. Il existe plusieurs modèles de robot, certains se déplacent plus rapidement alors que d'autres peuvent supporter un plus grand poids. Le robot que vous allez choisir aura une liste d'objets à aller chercher, lesquels sont éparpiller dans l'entrepôt. Vous décidez de représenter l'entrepôt comme un graphe, et d'utiliser l'algorithme bien connu de Dijkstra pour optimiser le trajet.

4 Description

Lors de votre rencontre avec votre employeur, on vous explique les contraintes qui doivent être respectées lors de l'implémentation de votre solution.

- Les différentes sections de l'entrepôt, représentées par des nœuds numérotés de 0 à n, sont reliées entre elles par des arcs, chacun ayant un coût en distance. Le robot commence toujours son trajet sur le nœud 0 et doit terminer sur celui-ci.
- La carte de l'entrepôt repose sur une matrice de distances entre les différentes sections de l'entrepôt et est donc représentable par un graph non orienté. De plus, les robots respecteront leur couloir de circulation. Les sommets représentent les différentes sections de l'entrepôt, chacun ayant une liste d'objets. Les arcs représentent la distance entre ces sommets.
- Il n'existe que 3 type d'objets que le robot peut ramasser (A,B,C). Par exemple le client peut commander 3 objets A, 4 objets B et 1 objet C.
- Les objets de type A ont un poids de 1Kg.
- Les objets de type B ont un poids de 3Kg.
- Les objets de type C ont un poids de 6Kg.
- Le coût en temps pour passer d'un sommet à l'autre est caractérisé par la fonction suivante : $T = k_i D$ où T est le temps en secondes, D la distance entre deux nœuds et k_i est caractéristique à chaque type de robot.
- Il y a un coût en temps constant de 10 secondes pour prendre un objet pour tous les types de robot et pour tous les types d'objet.
- Il existe 3 types de robot. Le robot X se déplace rapidement, mais ne soulève pas de charges lourde. Le robot Y se déplace un peu moins vite, mais à plus de facilité à soulever des charges. Finalement le robot Z se déplace lentement, mais peut soulever de lourdes charges.
- Le robot X a une constante $k_x = 1+m$ où m est la masse totale des objets que le robot transporte. Autrement dit, plus le robot transporte une charge lourde, plus le temps pour voyager d'un nœud à l'autre est long. Si la masse totale est 0, le temps de transport entre deux nœuds est directement proportionnel à la distance entre ces nœuds.
- Le robot Y a une constante $k_y = 1.5 + 0.6m$ où m est la masse totale des objets que le robot transporte. Le robot Y est donc plus lent lorsqu'il ne transporte rien que le robot X, mais il est moins impacté par la masse des objets qu'il transporte.
- Le robot Z a une constante $k_z = 2.5 + 0.2m$ où m est la masse totale des objets que le robot transporte. Le robot Z se déplace beaucoup moins vite que les robots X et Y, mais est beaucoup moins impacter par le poids des objets.
- De plus, chaque modèle de robot a une masse total maximale qu'il peut transporter. Les robots de type X peuvent transporter jusqu'à 5Kg, les robots de type Y jusqu'à 10Kg, et les robots de type Z jusqu'à 25Kg.
- Un fichier texte "entrepot.txt" vous est fournis et est séparé en 2 parties. Les premières lignes représentent chaque nœud, ainsi que le nombre d'objet de chaque type que ce nœud contient de la façon suivante : N, A, B, C où N est le numéro du nœud, A le nombre d'objets A qu'il contient, B le nombre d'objets B et C le nombre d'objets C.
- L'autre partie du fichier texte, séparé par une ligne vide, représente les arcs. Nous avons donc chaque arc représenté de la façon suivante : N1, N2, D où N1 est le numéro du premier nœud, N2 le numéro du deuxième nœud, et D la distance entre ces nœuds. À souligner que le graphe est non orienté.

5 Composantes à implémenter

C1. La première composante se sépare en deux parties. Une fonction "creerGraphe()" qui permet de créer un graphe représentant les différentes sections de l'entrepôt, ainsi que les chemins reliants ces sections entre elles à partir d'un fichier texte. Et une fonction "afficherGraphe()" qui permet de voir une représentation du graphe sauvegardé en memoire.

C2. La seconde composante se sépare également en deux parties. Une fonction "prendreCommande()" qui permet à l'utilisateur de rentrer un nombre d'objets de chaque type que le robot devra allez chercher. La seconde partie, une fonction "afficherCommande()" qui permet à l'utilisateur de voir la commande en mémoire.

C3. Il faut par la suite implémenter la fonction "plusCourtChemin()" qui détermine, en vous inspirant de l'algorithme de Dijkstra le chemin le plus rapide pour allez chercher votre commande. Le point de départ et d'arriver est toujours le noeud 0. La fonction affiche le type de robot utilisé, la liste des noeuds qu'il à traversé, specifie lorsqu'un objet est prit à un noeud et le temps total que le robot a prit pour allez chercher la commande. Si le chemin est impossible, la fonction doit en informer l'utilisateur.

C4. Faire une interface qui affiche les options suivante :

- creer le graphe
- afficher le graphe
- prendre une commande
- afficher la commande
- trouver le plus court chemin
- quitter

Le programme doit toujours afficher le menu tant que l'option "Quitter" n'a pas été selectionner et il doit fonctionner même lorsque l'utilisateur selectionne des options invalide.

6 Livrable

Le livrable attendu est constitué du code source et du rapport de laboratoire. Le livrable est une archive (ZIP ou RAR) dont le nom est formé des numéros de matricule des membres de l'équipe, séparés par un trait de soulignement (_). L'archive contiendra les fichiers suivants:

- les fichiers .cpp;
- les fichiers .h le cas échéant;
- le rapport au format PDF;
- le fichier .txt passé en argument (option (a)), s'il est différent de celui fourni par l'instructeur du laboratoire (vous pouvez en effet modifier le format des informations contenues dans le fichier fourni sur Moodle, mais vous devez à ce moment-là le remettre avec vos travaux).

L'archive ne doit pas contenir de programme exécutable, de fichier de projet ou solution de Visual Studio, de répertoire Debug ou Release, etc. Les fichiers .cpp et .h suffiront pour l'évaluation du travail.

6.1 Rapport

Un rapport de laboratoire rédigé avec soin est requis à la soumission (format .pdf, maximum 8 pages). Sinon, votre travail ne sera pas corrigé (aussi bien le code source que l'exécutable). Le rapport doit obligatoirement inclure les éléments ou sections suivantes :

1. Page présentation : elle doit contenir le libellé du cours, le numéro et l'identification du TP, la date de remise, les matricules et noms des membres de l'équipe. Vous pouvez compléter la page présentation qui vous est fournie.
2. Introduction avec vos propres mots pour mettre en évidence le contexte et les objectifs du TP.
3. Présentation de vos travaux : une explication de votre solution. Ajoutez le diagramme de classes complet, contenant tous les attributs et toutes les méthodes ajoutées.
4. Difficultés rencontrées lors de l'élaboration du TP et les éventuelles solutions apportées.
5. Conclusion : expliquez en quoi ce laboratoire vous a été utile, ce que vous avez appris, vos attentes par rapport au prochain laboratoire, etc.

Notez que vous ne devez pas mettre le code source dans le rapport.

6.2 Soumission du livrable

La soumission doit se faire uniquement par Moodle.

7 Évaluation

Éléments évalués	Points
Qualité du rapport : respect des exigences du rapport, qualité de la présentation des solutions	2
Qualité du programme: compilation, structures de données, gestion adéquate des variables et de toute ressource (création, utilisation, libération), passage d'arguments, gestion des erreurs, documentation du code, etc.	2
Composants implémentés : respect des requis, logique de développement, etc.	
C1	4
C2	3
C3	6
C4	3
Total de points	20

8 Documentation.txt

- http://www.cplusplus.com/doc/tutorial/
- http://public.enst-bretagne.fr/∼brunet/tutcpp/Tutoriel%20de%20C++.pdf
- http://fr.openclassrooms.com/informatique/cours/programmez-avec-le-langage-c
- Algorithme de Dijkstra illustré : https://www.youtube.com/watch?v=0nVYi3o161A

Annexe

1 Plus court chemin

Soient un graphe valué (ou pondéré) G et deux sommets a et b de G. Pour calculer le plus court chemin entre a et b, utilisons l'algorithme de Dijkstra. Soit *d(x)*, la distance du sommet x par rapport au sommet de départ a, *w(u,v)*, la longueur d'une arête $\{u,v\}$ et *ch(a,x)*, le chemin (liste des arêtes traversées) du sommet a au sommet x.

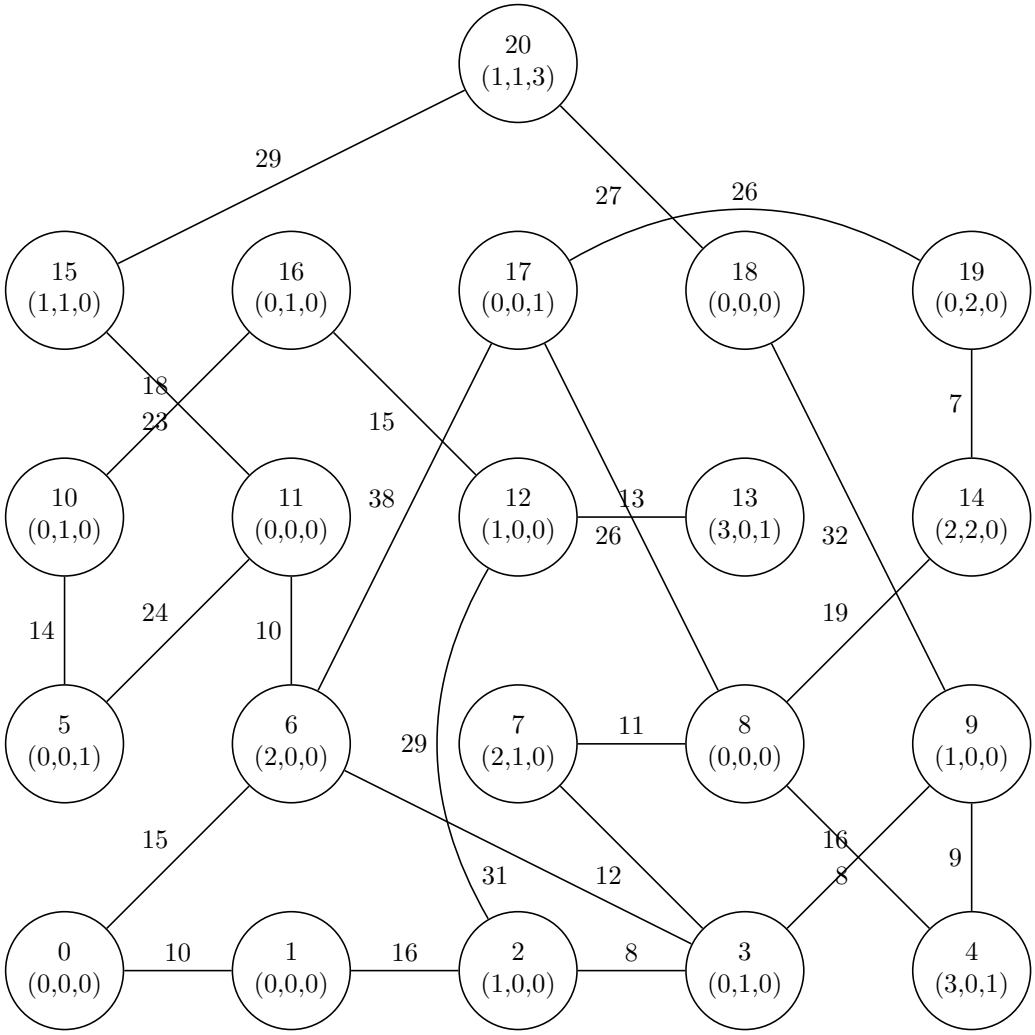
- Au début de l'algorithme, les distances de chaque sommet x au sommet de départ a sont fixées à la valeur infinie ($d(x) = \infty$), à l'exception du sommet de départ, a, dont la distance (par rapport à lui-même) est 0. Pour chaque sommet, on associe également la liste des sommets traversés (le chemin emprunté) du sommet initial a jusqu'au sommet en question. À cette étape de l'algorithme, cette liste est vide pour tous les sommets, sauf pour le sommet a, dont la liste se résume à lui-même. En d'autres termes, pour tous les autres sommets x de G, on associe une étiquette "x, ∞, ()", et pour le sommet a, on a "a, 0, (a)". On considère également un sous-graphe vide S.
- À chaque itération, on sélectionne le sommet x de G, qui n'apparaît pas dans S, de distance minimale (par rapport au sommet a). Ce sommet x est ajouté au sous-graphe S. Par la suite, si nécessaire, l'algorithme met à jour les étiquettes des voisins x_v du sommet x ajouté. Cette mise à jour s'effectue si $d(x_v) > d(x) + w(x,x_v)$. Dans ce cas, l'étiquette du sommet x_v est modifiée comme suit:
  $\{x_v, d(x) + w(x,x_v), (ch(a,x),x_v)\}$.
- On répète l'opération précédente jusqu'à la sélection du sommet d'arrivée b, ou jusqu'à épuisement des sommets. L'étiquette associée à b donne alors le plus court chemin de a à b, de même que la longueur de ce dernier.

2 Informations utiles

(a) **Affichage de la carte**
Pour afficher la carte, il faut indiquer, pour chaque sommet, quel est son inventaire (le nombre d'objet de type A, B et C), et la liste des sommets voisins avec les distances associées, comme illustré ci-dessous (par exemple) :
(Noeud0, NbObjetA, nbObjetB, nbObjetC, ((noeud_voisin$_{1_0}$, distance$_{1_0}$), (noeud_voisin$_{2_0}$, distance$_{2_0}$), ..., (noeud_voisin$_{n_0}$, distance$_{n_0}$)))
...

(b) **Affichage du parcours**
Pour afficher le plus court chemin, vous devez afficher la liste des sommets par lesquels le robot à traversé. Vous devez egalement spécifier lorsque le robot collecte un objet. Finalement, vous devez aussi afficher le type de robot utilisé ainsi que le temps qu'a prit le trajet. Par exemple :
$point_{départ} \rightarrow point_1 \rightarrow point_2 \rightarrow collecting\ A \rightarrow ... \rightarrow point_n \rightarrow point_{arrivée}$
robot utilisé : Type X
temps : 256

3 Graphe


20
(1,1,3)
29
27
26
15
(1,1,0)
16
(0,1,0)
17
(0,0,1)
18
(0,0,0)
19
(0,2,0)
18
23
15
7
10
(0,1,0)
11
(0,0,0)
38
12
(1,0,0)
13
26
13
(3,0,1)
32
14
(2,2,0)
14
24
10
19
5
(0,0,1)
6
(2,0,0)
29
7
(2,1,0)
11
8
(0,0,0)
9
(1,0,0)
15
16
9
31
12
8
0
(0,0,0)
10
1
(0,0,0)
16
2
(1,0,0)
8
3
(0,1,0)
4
(3,0,1)