POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

LOG 2810
STRUCTURES DISCRETES

TP1 : GRAPHES

Session	Automne 2018
Pondération	10 % de la note finale
Taille des équipes	3 étudiants
Date de remise du projet	29 octobre 2018 (23h55 au plus tard)
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.
	Programmation C++, Python ou Java
Les questions sont les bienvenues et peuvent être envoyées à : Mohameth Ndiaye (mohameth-alassane.ndiaye@polymtl.ca), Marc Anhoury (marc.anhoury@polymtl.ca), Saif-eddine Sajid (saif-eddine.sajid@polymtl.ca).	

1 Connaissances requises

- Notions d'algorithmique et de programmation C++, Python ou Java.
- 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 d'optimiser le parcours de véhicules médicalisés autonomes entre les centres locaux de services communautaires (CLSC) de Montréal.

3 Mise en situation

Un étudiant du département de génie informatique et génie logiciel vient de trouver un stage auprès d'une start-up de véhicules médicalisés autonomes. Un véhicule médicalisé autonome connaitra à l'avance la destination et optimisera le temps de trajet. Ce nouveau concept permettra de faciliter la vie des patients lors des transports médicaux. De plus, un véhicule sans chauffeur réduirait la facture des usagers. La start-up en question est basée à Montréal. En outre, le projet a reçu l'approbation des autorités afin d'être implémenté dans le réseau routier de Montréal. L'étudiant est responsable de développer l'algorithme qui détermine les chemins qu'emprunteront les véhicules médicalisés autonomes

puisqu'il a indiqué dans son CV avoir fait le cours de structures discrètes. Toutefois, il ne maîtrise pas très bien cette matière. Il a donc besoin de votre aide.

Il décide d'associer à chaque chemin possible entre deux CLSC un coût en temps, qui dépend bien évidemment de la distance entre les deux centres (le coût est proportionnel à celle-ci). L'étudiant a fait beaucoup de recherches sur Internet pour déterminer les voies empruntables par les véhicules médicalisés autonomes, et voudrait que son algorithme soit capable de proposer aux véhicules intelligents un chemin optimal pour minimiser les coûts d'exploitation. Pour cela, il décide de représenter les voies entre CLSC par un graphe des distances. Ainsi, la carte des CLSC à Montréal est représentée dans le petit logiciel console qu'il élabore par un graphe dont les sommets correspondent aux points centraux des différents CLSC de Montréal, et les arêtes aux distances entre deux points centraux (sommets). L'étudiant vous fournira bien évidemment ce graphe des distances, et il a besoin de vous pour l'aider à implémenter certains composants de son algorithme.

4 Description

Lors de la séance de présentation du projet, il vous explique les concepts suivants, qui sont des concepts simplifiés de l'application qu'aimerait créer la start-up, mais suffisants pour faire une première ébauche de son travail de stage. Il vous recopie à partir de son manuel un rappel sur certaines notions de la théorie des graphes et vous donne les informations sur certaines particularités des véhicules médicalisés autonomes qui seront utilisés lors du trajet entre CLSC.

- Comme dit précédemment, chaque déplacement entre CLSC par la route possède un coût en temps proportionnel à la distance parcourue.
- La carte du lieu dans lequel progressera un véhicule médicalisé autonome repose sur une matrice de distances entre les différents points centraux des CLSC, et est donc représentable par un graphe non orienté. En outre, ces véhicules intelligents respecteront leur couloir de circulation à droite sur la route. Les sommets sont les points centraux des CLSC et les arêtes les distances entre ces points. Un tel graphe sera connexe, à savoir que l'on connaîtra la distance entre un CLSC et tous les autres des CLSC de Montréal.
- Les véhicules médicalisés autonomes sont des véhicules électriques. Dès lors, ils ont en général une autonomie et on doit les recharger fréquemment. Certains points centraux des CLSC sont pourvus de bornes de recharge où les véhicules pourront faire le plein d'énergie de leurs batteries (on suppose qu'il n'existe pas d'autres bornes que celles indiquées).
- Les véhicules médicalisés utilisés sont de deux types : les véhicules médicalisés à haute autonomie, qui possèdent des batteries LI-ion et que la start-up a pu se procurer de peine et de misère, car ils sont onéreux, et les véhicules à moyenne autonomie, qui possèdent des batteries à NI-MH, mais qui sont beaucoup plus accessibles. Les deux types de véhicules médicalisés parcourent les distances en un même temps.
- La start-up offre trois types de transport. La première catégorie correspond au « Transport à faible risque », autrement dit le patient n'a pas besoin de machine électrique pour survivre lors de son transfert dans un autre CLSC. La seconde catégorie est le « Transport à moyen risque », ce qui implique une situation de danger où des précautions (masque à oxygène, défibrillateur) sont prises afin que le patient reste en vie durant le transfert. La dernière catégorie correspond au « Transport à haut risque », impliquant une situation critique où le patient a besoin de soutien et d'aide constante pour le maintenir en vie tout au long du transfert.
- La catégorie « Transport à faible risque » gaspille moins l'énergie des batteries des véhicules médicalisés que celle du « Transport à moyen risque », qui quant à elle gaspille moins l'énergie des batteries des véhicules médicalisés que celle du « Transport à haut risque ». Le coût d'énergie lorsqu'un véhicule médicalisé autonome parcourt une distance entre deux CLSC varie donc selon le type de transport.

- **Le transport avec les véhicules médicalisés LI-ion sont à éviter**, puisqu'elles sont chères et sont sujettes à la surchauffe. La start-up souhaite donc réserver ces dernières lorsque les véhicules médicalisés NI-MH ne suffiront pas à la tâche. L'algorithme doit conséquemment être capable de détecter les situations où les véhicules médicalisés NI-MH ne disposent pas d'assez d'autonomie et ne pourront pas passer par suffisamment de stations de recharge pour achever le transport.

- Dans les cas où les véhicules médicalisés NI-MH ne suffisent pas, l'algorithme devra également vérifier que les véhicules médicalisés LI-ion pourront quant à eux achever le trajet. Si ce n'est pas le cas, alors le transport par la start-up devra être refusé.

- Les sommets du graphe représentant la carte devront contenir un paramètre de recharge si ce point central CLSC est pourvu d'une station de recharge. Lorsqu'un véhicule médicalisé atteint une station, il peut attendre que ses batteries récupèrent leur plein potentiel. Le potentiel d'une batterie est exprimé sur 100. À 100% d'énergie, les batteries de véhicules médicalisés sont pleines. **À 0% d'énergie, le véhicule médicalisé tombe sur la voie publique, ce qu'il faut de toute évidence éviter à tout prix.**

- À cet effet, **l'algorithme doit éviter de donner à un véhicule médicalisé un chemin qui fait passer ses batteries au-dessous de 20%**. Si c'est le cas, on préférera un autre chemin plus coûteux en termes de temps, mais plus sécuritaire pour les citoyens. Si une telle trajectoire n'existe pas, c'est seulement à ce moment-là qu'on utilisera un véhicule médicalisé LI-ion. Si les mêmes conditions ne sont pas respectées non plus pour le véhicule médicalisé LI-ion, on refusera le transport.

- Chaque arrêt dans une station de recharge représente un coût en temps dont on doit tenir compte pour calculer le plus court chemin. Le temps nécessaire pour la recharge des batteries d'un véhicule médicalisé est de 120 minutes, quel que soit le type de véhicule ainsi que son niveau de décharge. **Un véhicule qui pourrait manquer de charge pour compléter son trajet pourrait s'arrêter devant une borne pour recharger ses batteries. Dans le cas échéant, le temps de recharge demeure 120 minutes**. Après une recharge, ses batteries sont à 100% d'énergie.

- Au départ de chaque trajet, on considère que les véhicules médicalisés ont 100% d'énergie dans leurs batteries. On considère également que les véhicules médicalisés ne peuvent effectuer qu'un type de transport à la fois.

- La perte d'autonomie d'un véhicule NI-NH va comme suit : pour un transport de catégorie **« Transport à faible risque », le véhicule perd 6% d'autonomie par heure de route**. Pour un transport de catégorie **« Transport à moyen risque », le véhicule perd 12% d'autonomie par heure de route**. Pour le transport **« Transport à haut risque », le véhicule médicalisé perd 48% d'autonomie par heure de route**.

- La perte d'autonomie d'un véhicule médicalisé LI-ion va comme suit : pour un transport de type **« Transport à faible risque », le véhicule médicalisé perd 5% d'autonomie par heure de route**. Pour un transport de catégorie **« Transport à moyen risque », le véhicule médicalisé perd 10% d'autonomie par heure de route**. Pour le transport **« Transport à haut risque », le véhicule médicalisé perd 30% d'autonomie par heure.**

- Un sous-graphe est un graphe contenu dans un autre. Plus formellement, H est un sous-graphe d'un graphe G si c'est un graphe, si l'ensemble des sommets de H est un sous-ensemble de l'ensemble des sommets de G, et si l'ensemble des arêtes de H est un sous-ensemble de l'ensemble des arêtes de G.

- Un chemin reliant un sommet a à un sommet b est l'ensemble des arêtes consécutives reliant a à b. La séquence des arêtes parcourues définit le chemin entre a et b.

- Un graphe est connexe s'il existe toujours un chemin entre chaque paire de sommets distincts du graphe. Dans le cas contraire, un graphe est constitué de l'union de plusieurs sous-graphes disjoints, lesquels représentent les composantes connexes du graphe.
- Un graphe valué (ou pondéré) est un graphe dans lequel chacune des arêtes présente une valeur. Cette valeur peut symboliser une distance, un coût, une durée, etc.
- La longueur d'un chemin, dans un graphe pondéré, est la somme des poids des arêtes parcourues.

Voici une carte des centres locaux de services communautaires de Montréal entre lesquels doivent se déplacer les véhicules médicalisés autonomes :

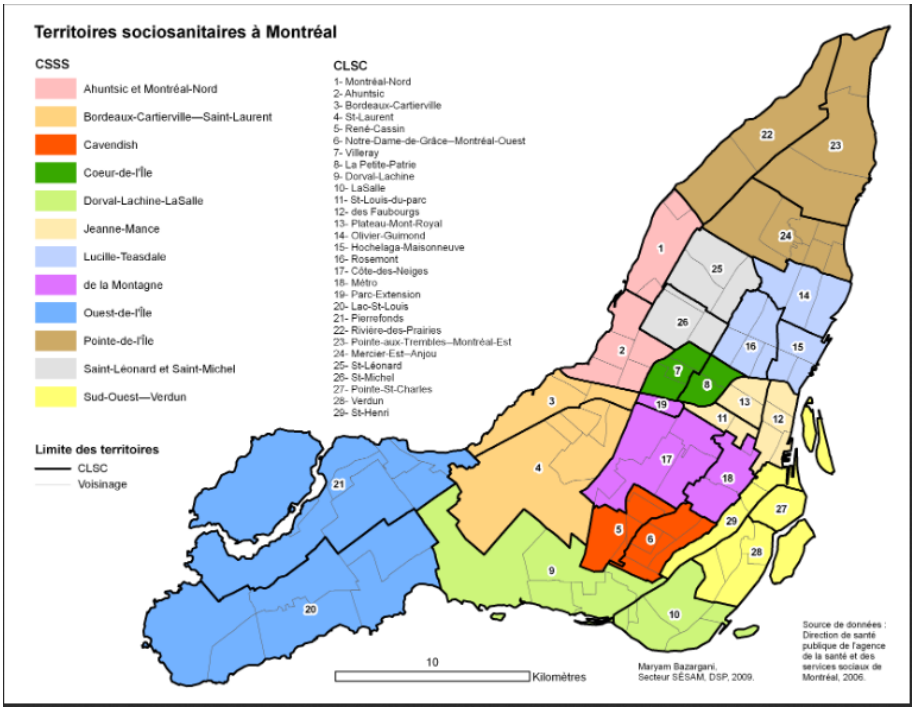

FIG. 1 : Centres locaux de services communautaires de Montréal[1]

1	Montréal-Nord	16	Rosemont
2	Ahuntsic	17	Côte-des-Neiges
3	Bordeaux-Cartierville	18	Métro
4	Saint-Laurent	19	Parc-extension
5	René-Cassin	20	Lac-Saint-Louis
6	Notre-Dame-de-Grâce—Montréal-Ouest	21	Pierrefonds
7	Villeray	22	Rivière-des-Prairies
8	La Petite-Patrie	23	Pointe-aux-Trembles—Montréal-Est
9	Dorval-Lachine	24	Mercier-est-anjou
10	LaSalle	25	Saint-Léonard
11	St-Louis-du-parc	26	Saint-Michel
12	Faubourgs	27	Pointe-saint-charles
13	Plateau-Mont-Royal	28	Verdun
14	Olivier-Guimond	29	Saint-Henri
15	Hochelaga-Maisonneuve		

Le fichier `centresLocaux.txt` contient les informations nécessaires pour l'algorithme.

- Les premières lignes contiennent la liste des sommets. Chaque ligne contient le numéro d'un CLSC et si ce dernier contient une borne de recharge (1) ou non (0). Ces deux nombres sont séparés par une virgule.
- Il y a une ligne vide entre les lignes sur les sommets et celles sur les arêtes.
- Les lignes suivantes contiennent la liste des arêtes. Chaque ligne contient le numéro du premier sommet, le numéro du deuxième sommet, et le temps de parcours entre les deux (en minutes). Les arêtes ne sont pas orientées.

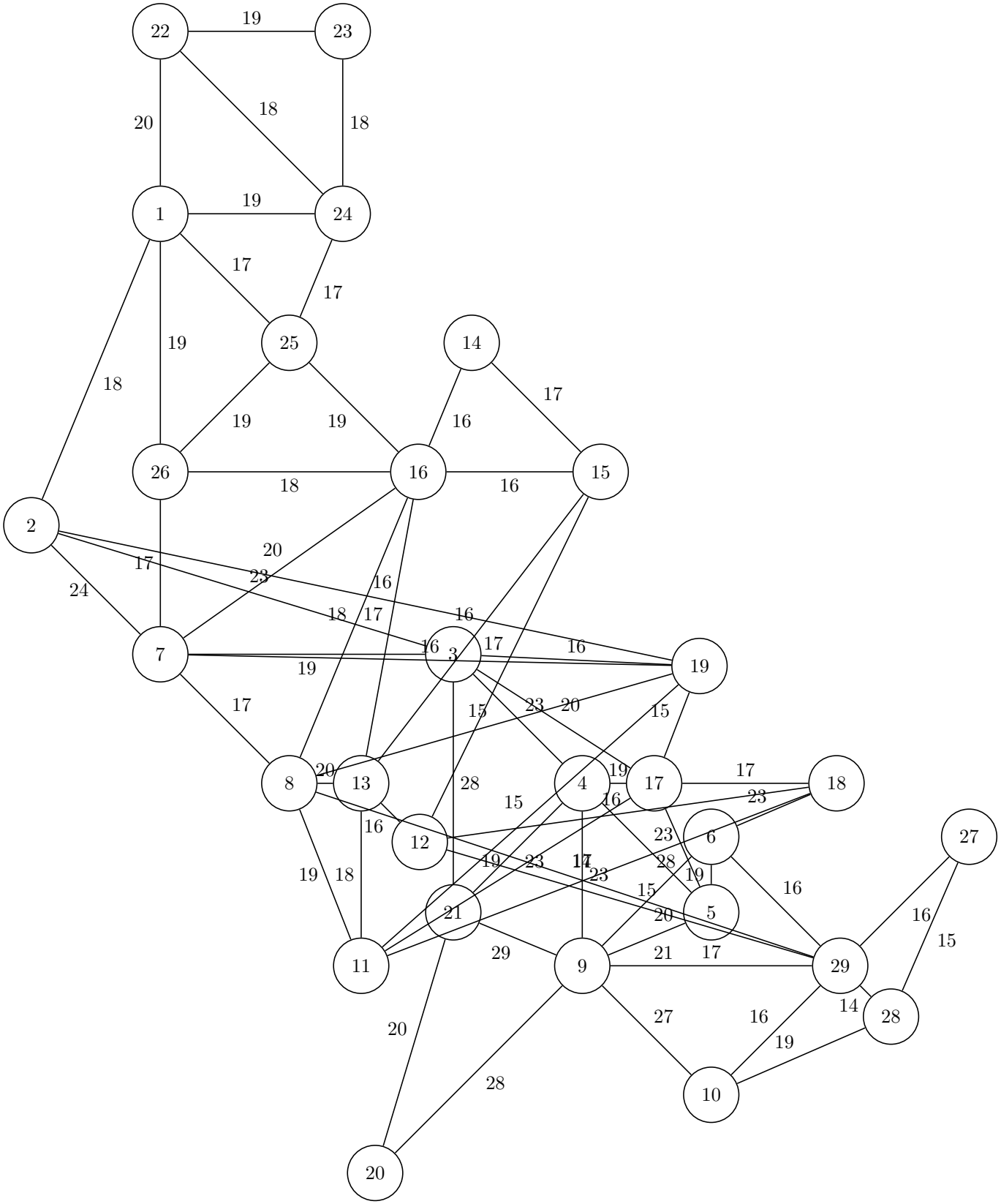
22
23
1
24
25
14
26
16
15
2
7
3
19
8
13
4
17
18
12
6
27
21
5
11
9
29
28
10
20

Fig. 2 : Graphe des CLSC de Montréal et temps de parcours entre CLSC

5 Composants à implémenter

C1. Écrire une fonction récursive “creerGraphe()” qui permet de créer le graphe représentant les routes et les points centraux des CLSC (sommets) à partir d’un fichier dont le nom est passé en paramètre.

C2. Écrire une fonction “lireGraphe()” qui permet d’afficher le graphe (cf. annexe a. pour un exemple d’affichage de la carte sous forme de graphe).

C3. Écrire la fonction ”plusCourtChemin()” qui permet de déterminer, en vous inspirant de l’algorithme de Dijkstra, le plus court chemin sécuritaire pour transporter un patient dont la catégorie est passée en paramètre. L’origine (point de départ) et la destination (sommet d’arrivée) doivent aussi être passées en paramètres. La fonction affiche le type de véhicule médicalisé utilisé (NI-NH ou LI-ion), le pourcentage final d’énergie dans les batteries du véhicule médicalisé, le plus court chemin utilisé (d’après la liste de ses sommets, selon le format de l’annexe) et la longueur de ce dernier en temps (minutes). Si le transport est refusé, rien de tout cela ne doit être affiché ; on doit plutôt recevoir un message d’excuse justifiant le refus du transport.

C4. Écrire une fonction “extraireSousGraphe()” qui permet d’extraire le sous-graphe qui contient le chemin le plus long qu’un véhicule NI-NH ou LI-ion pourrait parcourir à partir d’un sommet en explorant toutes les arêtes de celui-ci.

C5. Faire une interface qui affiche le menu suivant :

   (a) Mettre à jour la carte.

   (b) Déterminer le plus court chemin sécuritaire.

   (c) Extraire un sous-graphe.

   (d) Quitter.

Notes

- Le programme doit toujours réafficher le menu, tant que l’option (d), ou « Quitter », n’a pas été choisie.
- L’utilisateur doit entrer un index valide, sinon le programme le signale et affiche le menu de nouveau.
- L’option (a) permet de lire une nouvelle carte afin de créer le graphe correspondant. La génération d’une nouvelle carte pourrait être importante dans les cas où des travaux importants seraient en cours ou des changements dans les voies aériennes autorisées de la ville de Montréal seraient autorisés. Pour lire une nouvelle carte, le nom du fichier contenant les informations de la carte doit être demandé. Il est demandé d’afficher le graphe obtenu à la lecture d’un fichier. Le format du fichier est décrit en annexe.
- L’option (b) permet de déterminer le plus court chemin sécuritaire d’après les points de départ et d’arrivée ainsi que le type de transport. Pour ce faire, les paramètres doivent être demandés par la console.

1. Source : `https://emis.santemontreal.qc.ca/fileadmin/emis/Outil/Atlas/carte_pdf/decoupages/CSSS_CLSC_Voisinages_Montreal_01.pdf`, carte réalisée par Maryam Bazargani

- L'option (c) prendra le sommet (son indice) et le type de véhicule à partir de la console afin d'extraire le sous-graphe qui contient le chemin le plus long que le véhicule choisi pourrait parcourir à partir du sommet fourni en explorant toutes les arêtes de celui-ci. Le véhicule doit parcourir le plus long chemin possible avec une seule charge et sans tomber en panne.
- Si une nouvelle carte est lue, les options (b) et (c) doivent être réinitialisées.

Prenez note que selon votre convenance, le mot "fonction" peut être remplacé par "méthode" et vice-versa, et que plusieurs autres fonctions/méthodes peuvent être ajoutées pour vous faciliter la tâche. La figure 3 présente un exemple de diagramme de classes à partir duquel vous pouvez travailler.

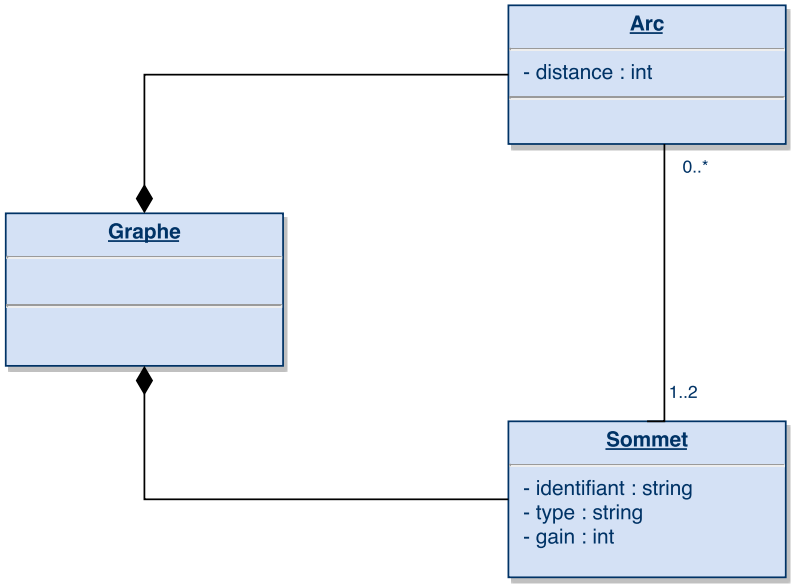

FIG. 3 : Exemple du diagramme de classes de la solution

6 Livrable

Le livrable attendu est constitué du code source et du rapport de laboratoire. Le livrable est une archive (ZIP ou RAR ou tar.gz) 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 ou .py ou .java ;
- 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, ou .py, ou. java 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. Le rapport doit obligatoirement inclure les éléments ou sections suivantes :

1. Page de 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. Vous pouvez également indiquer le temps passé sur ce TP à des fins d'ajustement pour le prochain qui aura lieu vers la fin de la session.

Notez que vous ne devez pas mettre de 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	3
C2	3
C3 Divisé comme suit	5
C3.1 Implémentation correcte de Dijkstra	2
C3.2 Choix de chemin selon le colis	2
C3.3 Choix du véhicule médicalisé/du refus selon le cas	1
C4	3
C5	2
Total de points	20

8 Documentation.txt

- `http://public.enst-bretagne.fr/$\sim$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`
- `learnxinyminutes.com`

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 du graphe**
Pour afficher le graphe, il faut indiquer, pour chaque sommet, quel est son numéro, et la liste des sommets voisins avec les temps associés, comme illustré ci-dessous :
(objet1, numéro1, ((objet_voisin$_{1_1}$, durée$_{1_1}$), (objet_voisin$_{2_1}$, durée$_{2_1}$), ..., (objet_voisin$_{n_1}$, durée$_{n_1}$)))
(objet2, numéro2, ((objet_voisin$_{1_2}$, durée$_{1_2}$), (objet_voisin$_{2_2}$, durée$_{2_2}$), ..., (objet_voisin$_{n_2}$, durée$_{n_2}$)))
...

(b) **Affichage du parcours**
Pour afficher le plus court chemin, la liste des sommets (ou objets) définissant le trajet doit être présentée comme suit :
$point_{départ} \rightarrow point_1 \rightarrow point_2 \rightarrow ... \rightarrow point_n \rightarrow point_{arrivée}$

(c) **Structure des fichiers**
Les cartes sont présentées sous forme de fichier textes (cf. le fichier texte fourni sur Moodle). Un tel fichier contient deux listes.