POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

LOG 2810
STRUCTURES DISCRETES

TP1 : GRAPHES

Session	Automne 2017
Pondération	10 % de la note finale
Taille des équipes	3 étudiants
Date de remise du projet	24 octobre 2017 (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 Alassane Ndiaye (mohameth-alassane.ndiaye@polymtl.ca), Justine Pepin (justine.pepin@polymtl.ca), Juliette Tibayrenc (juliette.tibayrenc@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 d'un colis livré par drone dans Montréal. Ce premier aspect est abordé dans la partie 5. Il est également question d'extraire le diagramme de Hasse à partir d'un graphe. Ce second aspect est abordé dans la partie 6.

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 drones qui aimerait populariser les livraisons de colis au particulier par la voie des airs afin de désengorger le réseau routier de Montréal. On le met en charge de l'algorithme qui déterminera les chemins qu'emprunteront les drones puisqu'il a indiqué dans son C.V. avoir fait le cours de structures discrètes. Malheureusement, il ne maîtrise pas très bien cette matière. Il a donc désespérément besoin de votre aide.

Il décide d'associer à chaque chemin possible entre deux points de la ville un coût en temps, qui dépend bien évidemment de la distance entre les deux points (le coût est proportionnel à celle-ci). L'étudiant a fait beaucoup de recherches sur Internet pour déterminer les voies empruntables par les drones, et voudrait que son algorithme soit capable de proposer aux robots un chemin optimal pour minimiser les coûts d'exploitation. Pour cela, il décide de représenter les voies de livraison par un graphe des distances entre les quartiers de Montréal. Ainsi, Montréal est représentée dans le petit logiciel console qu'il élabore par un graphe dont les sommets correspondent aux points centraux des différents quartiers 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 drones qui seront utilisés pour la livraison.

- Comme dit précédemment, chaque déplacement vers un autre quartier de Montréal via la voie des airs possède un coût en temps proportionnel à la distance parcourue.
- La carte du lieu dans lequel progressera un drone de livraison repose sur une matrice de distances entre les différents points centraux des quartiers, et est donc représentable par un graphe non orienté, puisque les quadricoptères savent respecter leur couloir de circulation à droite de la rue. Les sommets sont les points centraux des quartiers et les arêtes les distances entre ces points. Un tel graphe sera connexe, à savoir que l'on connaîtra la distance entre un quartier et tous les autres de Montréal.
- Les drones de livraison peuvent faire le transport de plusieurs colis dans la journée, mais ont un besoin de rechargement à prendre en compte. En effet, les quadricoptères ont en général une piètre autonomie et on doit les recharger fréquemment. Certains points centraux des quartiers sont pourvus de stations de recharge où les drones pourront se poser et faire le plein d'énergie dans leurs batteries.
- Les drones utilisés sont de deux types : les drones à haute autonomie, qui possèdent des batteries à 5.0 Ampères et que la start-up a pu se procurer de peine et de misère car ils sont onéreux, et les drones à moyenne autonomie, qui possèdent des batteries à 3.3 Ampères mais qui sont beaucoup plus accessibles. Les deux types de robots parcourent les distances en un même temps.
- Les colis transportés doivent respecter des normes de poids et de taille. La start-up permet trois types de colis qui sont classés en trois catégories selon leur poids et taille. La première catégorie contient les « poids plumes », soit les colis petits et légers. La seconde catégorie contient les « poids moyens », soit les colis de poids et de taille raisonnables. La dernière catégorie contient les « poids lourds », soit les colis qui sont légèrement surdimensionnés ou plus pesants.
- Ainsi, la catégorie des poids plumes gaspille moins l'énergie des batteries des drones que celle des poids moyens, qui quant à elle gaspille moins l'énergie des batteries des drones que celle des poids lourds. Le coût d'énergie lorsqu'un quadricoptère parcourt une distance entre deux points centraux des quartiers varie donc selon la catégorie d'appartenance du paquet envoyé.
- Les livraisons des colis avec les drones 5.0 Ampères sont à éviter, puisqu'ils sont chers et plus difficiles à réparer. La start-up souhaite donc réserver ces derniers lorsque les drones 3.3 Ampères ne suffiront pas à la tâche. L'algorithme doit conséquemment être capable de détecter les situations où les drones 3.3 Ampères ne disposent pas d'assez d'autonomie et ne pourront pas passer par suffisamment de stations de recharge pour achever la livraison.

- Dans les cas où les drones 3.3 Ampères ne suffisent pas, l'algorithme devra également vérifier que les drones 5.0 Ampères pourront quant à eux achever le trajet. Si ce n'est pas le cas, alors l'envoi du colis par la start-up devra être refusé.
- Les sommets du graphe représentant la carte devront contenir un paramètre de recharge si ce point central de quartier est pourvu d'une station de recharge. Lorsqu'un drone se pose sur une station, il attend que ses batteries récupèrent leur plein potentiel. Le potentiel d'une batterie est exprimé sur 100. À 100% d'énergie, les batteries du drone sont pleine. À 0% d'énergie, le drone 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 drone une chemin qui fait passer ses batteries au-dessous de 20%. Si c'est le cas, on préférera une 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 drone 5.0 Ampères. Si les mêmes conditions ne sont pas respectées non plus pour le drone 5.0 Ampères, on refusera la livraison.
- 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 drone est de 20 minutes, quelque soit le type de drone ainsi que son niveau de décharge. Le drone s'arrête pour recharger ses batteries dès qu'il passe par une station de recharge, peu importe le niveau de ses batteries. En quittant la station, ses batteries sont à 100% d'énergie.
- Au départ de chaque trajet, on considère que les drones ont 100% d'énergie dans leurs batteries. On considère également que les drones ne peuvent transporter qu'un seul colis à la fois.
- La perte d'autonomie d'un drone 3.3 Ampères va comme suit : pour un colis de catégorie poids plumes, le quadricoptère perd 10% d'autonomie par 10 minutes de vol. Pour un colis de catégorie poids moyens, le quadricoptère perd 20% d'autonomie par 10 minutes de vol. Pour les colis de catégorie poids lourds, le quadricoptère perd 40% d'autonomie par 10 minutes de vol.
- La perte d'autonomie d'un drone 5.0 Ampères va comme suit : pour un colis de catégorie poids plumes, le quadricoptère perd 10% d'autonomie par 10 minutes de vol. Pour un colis de catégorie poids moyens, le quadricoptère perd 15% d'autonomie par 10 minutes de vol. Pour les colis de catégorie poids lourds, le quadricoptère perd 25% d'autonomie par 10 minutes de vol.
- 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 chacunes 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 arrondissements de Montréal entre lesquels doivent se déplacer les drones :

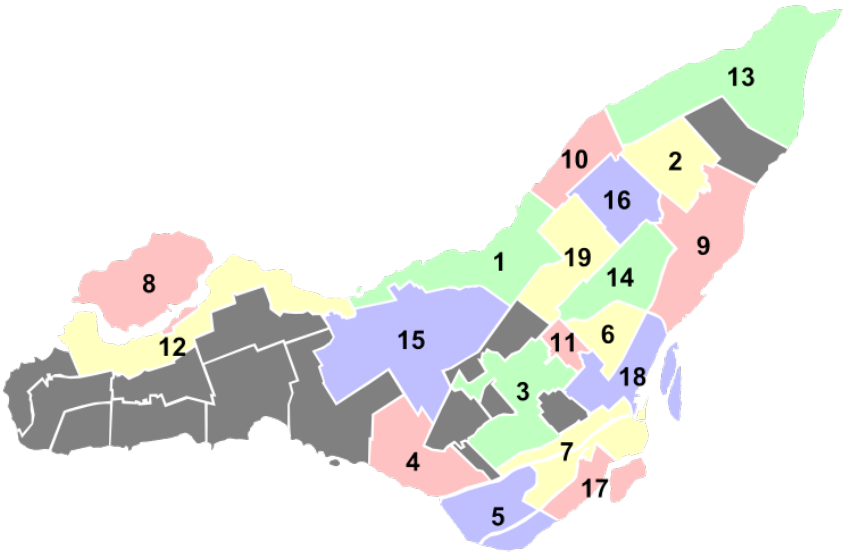

FIG. 1: Arrondissements de Montréal[1]

1	Ahuntsic-Cartierville	11	Outremont
2	Anjou	12	Pierrefonds-Roxboro
3	Côte-des-Neiges–Notre-Dame-de-Grâce	13	Rivière-des-Prairies–Pointe-aux-Trembles
4	Lachine	14	Rosemont–La Petite-Patrie
5	LaSalle	15	Saint-Laurent
6	Le Plateau-Mont-Royal	16	Saint-Léonard
7	Le Sud-Ouest	17	Verdun
8	L'Île-Bizard–Sainte-Geneviève	18	Ville-Marie
9	Mercier–Hochelaga-Maisonneuve	19	Villeray–Saint-Michel–Parc-Extension
10	Montréal-Nord		

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

- Les premières lignes contiennent la liste des sommets. Chaque ligne contient le numéro d'un arrondissement et si cet arrondissement 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.

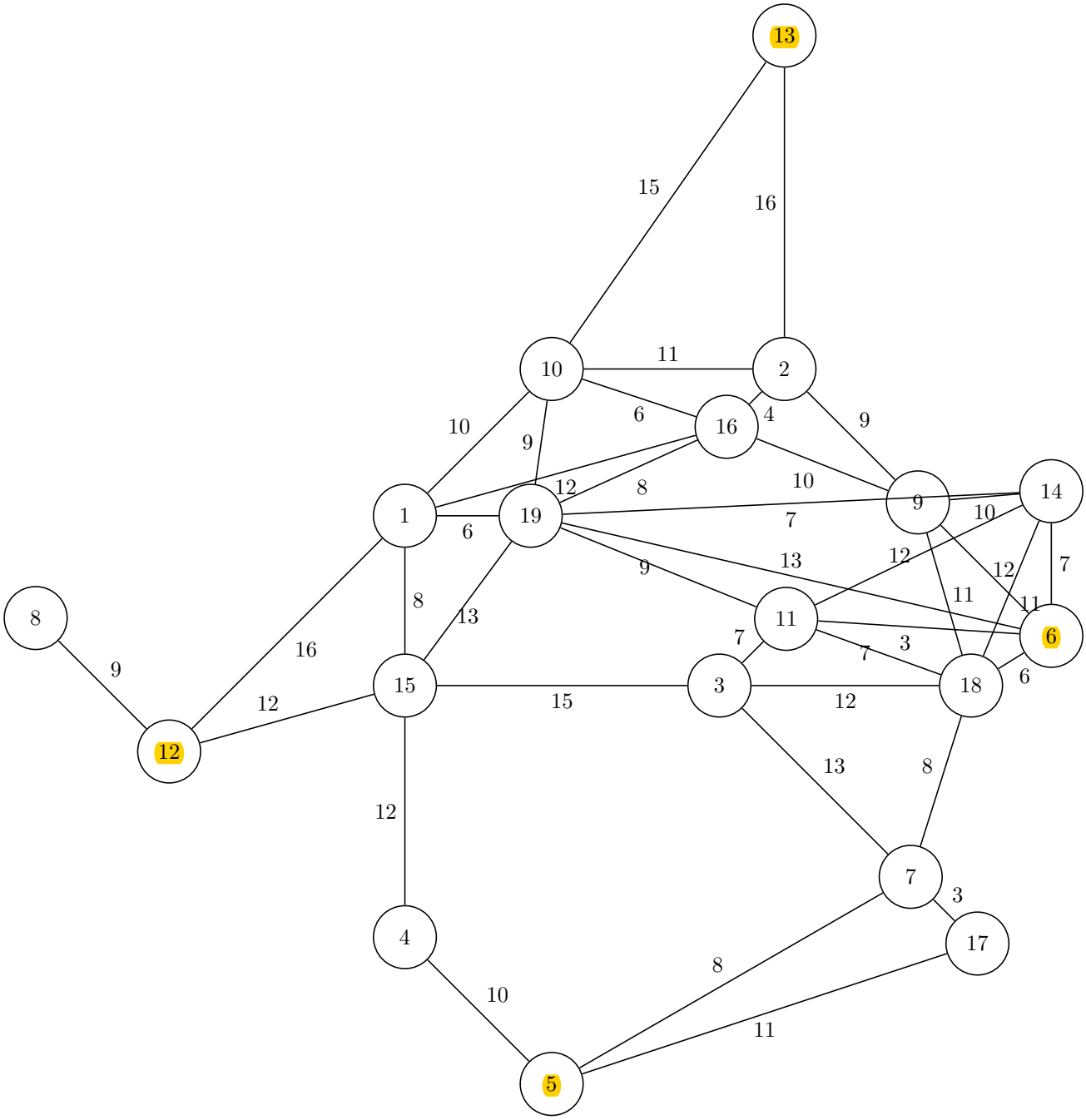

Fig. 2: Graphe des quartiers de Montréal et temps de parcours entre quartiers

5 Composants à implémenter - Drones

C1. Écrire une fonction récursive “creerGraphe()” qui permet de créer le graphe représentant les routes et les points centraux des quartiers (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 livrer un colis dont la catégorie est passée en paramètre. L'origine (point de départ) et la destination (sommet d'arrivée) doivent aussi être passés en paramètres. La fonction affiche le type de drone utilisé (5.0 Ampères ou 3.3 Ampères), le pourcentage final d'énergie dans les batteries du drone, 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 la livraison est refusée, rien de tout cela ne doit être affiché ; on doit plutôt recevoir un message d'excuses justifiant le refus de la livraison.

C4. Faire une interface qui affiche le menu suivant :

(a) Mettre à jour la carte.

(b) Déterminer le plus court chemin sécuritaire.

(c) Quitter.

Notes

- Le programme doit toujours réafficher le menu, tant que l'option (c), 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 la catégorie du colis. Pour ce faire, les paramètres doivent être demandés par la console.
- 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.

1. Source : `https ://commons.wikimedia.org/wiki/File :Carte_Montr%C3%A9al_Arrondissements.svg ?uselang=fr`, carte réalisée par Chicoutimi, licence CC-BY-SA 3.0

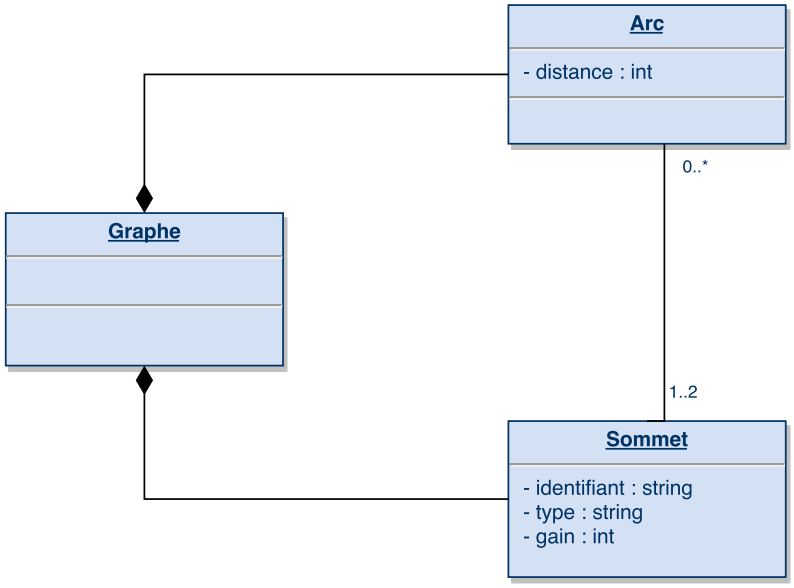

FIG. 3: Exemple du diagramme de classes de la solution

6 Composants à implémenter - Déjeuner et desserts

Notre jeune stagiaire hyperactif a déjà son prochain stage en vue. Cette fois-ci, il s'agit d'un stage dans une grosse compagnie et on lui demande en entrevue d'écrire un court programme à remettre dans quelques semaines. Le court programme est sur le sujet de son choix, mais doit permettre de déterminer un ordre (total ou partiel) et de tracer un diagramme de Hasse valide d'après une liste de sommets et d'arêtes orientées entre ces sommets dont il aura généré le graphe. Comme il s'agit encore de notions de structures discrètes, l'étudiant vient vous voir de nouveau avec la description du graphe qu'il veut utiliser (qu'il a tiré du livre de cuisine de ses parents).

Description du graphe :

V : {Déjeuner, Desserts, Confiture de rhubarbes, Pâte brisée, Rhubarbes, Fraises, Zeste de citron, Kombucha, Citron, Gélatine, Mousse au citron, Œufs, crème, Œufs brouillés, Tarte fraises-rhubarbes, Tarte au Citron, Lait, Sucre, Yoghurt, Farine, Poivre, Sel}

E : {(Confiture de rhubarbes, Déjeuner), (lait, yoghurt), (yoghurt, déjeuner), (gélatine, mousse au citron), (citron, zeste de citron), (zeste de citron, mousse au citron), (mousse au citron, tarte au citron), (tarte au citron, desserts), (farine, pâte brisée), (pâte brisée, tarte au citron), (poivre, oeufs brouillés), (sel, oeufs brouillés), (oeufs, oeufs brouillés), (sucre, confiture de rhubarbes), (sucre, mousse au citron), (sucre, kombucha), (kombucha, déjeuner), (kombucha, kombucha), (rhubarbes, kombucha), (rhubarbes, confiture de rhubarbes), (rhubarbes, tarte fraises-rhubarbes), (tarte fraises-rhubarbes, desserts), (tarte au citron, desserts), (oeufs brouillés, déjeuner), (lait, oeufs brouillés), (lait, crème), (pâte brisée, tarte fraises-rhubarbes), (crème, tarte fraises-rhubarbes), (yoghurt, déjeuner), (yoghurt, mousse au citron)}

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

- Les premières lignes contiennent la liste des sommets. Chaque ligne contient le numéro du sommet et son étiquette (nom du plat ou de l'ingrédient). Ces deux éléments 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. Ces arêtes sont orientées. Chaque ligne contient le numéro du premier sommet et le numéro du deuxième sommet.

Voici les requis pour le devoir qu'on lui a donné en entrevue.

D1. Écrire une fonction "creerGrapheOriente()" qui permet de créer le graphe représentant les recettes à partir d'un fichier dont le nom est passé en paramètre et qui permet d'afficher le graphe (cf. annexe a. pour un exemple d'affichage sous forme de graphe).

D2. Écrire la fonction "genererHasse()" qui permet de déterminer, en vous inspirant des principes de la génération des diagrammes de Hasse, un ordre valide entre les ingrédients et les recettes compris dans la description du graphe. La fonction doit afficher le diagramme de Hasse.

D3. Faire une interface qui affiche le menu suivant :

(a) Créer et afficher le graphe des recettes.
(b) Générer et afficher le diagramme de Hasse.
(c) Quitter.

Notes

- Le programme doit toujours réafficher le menu, tant que l'option (c), 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 le fichier des recettes afin de créer le graphe correspondant. Il est demandé d'afficher le graphe obtenu à la lecture du fichier. Le format du fichier est décrit en annexe. Le format d'affichage d'un graphe est en annexe.
- L'option (b) permet de déterminer un ordre et de l'afficher parmi les éléments du graphe précédent. Le format d'affichage d'un diagramme de Hasse est en annexe.

7 Main

La fonction principale de votre programme doit contenir un petit menu dirigeant l'utilisateur vers le numéro sur les drones ou le numéro sur les recettes.

Faire une interface qui affiche le menu suivant :

(a) Drones.

(b) Recettes.

(c) Quitter.

Les options « Quitter » du numéro sur les drones et du numéro sur les recettes ramènent vers ce menu. L'option »Quitter » de cette fonction principale termine l'exécution de votre programme.
Notes

- Le programme doit toujours réafficher le menu, tant que l'option (c), 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) redirige l'utilisateur vers l'interface console du numéro des drones.
- L'option (b) redirige l'utilisateur vers l'interface console du numéro des recettes.

8 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.

8.1 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.

8.2 Soumission du livrable

La soumission doit se faire uniquement par Moodle.

9 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.	3
Composants implémentés : respect des requis, logique de développement, etc.	
C1	2
C2	2
C3 Divisé comme suit	3
C3.1 Implémentation correcte de Dijkstra	1
C3.2 Choix de chemin selon le colis	1
C3.3 Choix du drone/du refus selon le cas	1
C4	2
D1	1
D2	3
D3	2
Total de points	20

10 Documentation.txt

- `http://www.cplusplus.com/doc/tutorial/`
- `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.

(d) **Affichage du diagramme de Hasse**
Pour afficher le diagramme de Hasse, les ingrédients et recettes ordonnés doivent être présentés sous forme de listes croissantes comme suit :
liste 1 : $ingrédientA_{départ} \rightarrow recetteA_1 \rightarrow recetteA_2 \rightarrow ... \rightarrow recetteA_n \rightarrow recetteA_{finale}$
liste 2 : $ingrédientB_{départ} \rightarrow recetteB_1 \rightarrow recetteB_2 \rightarrow ... \rightarrow recetteB_n \rightarrow recetteB_{finale}$
...