POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

LOG 2810
STRUCTURES DISCRETES

TP2 : AUTOMATES

Session	Automne 2017
Pondération	10 % de la note finale
Taille des équipes	3 étudiants
Date de remise du projet	28 novembre 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 sur les automates.
- Notions sur les structures de données.

2 Objectif

L'objectif de ce travail pratique est de vous permettre d'appliquer les notions théoriques sur les automates que nous avons vues en cours, sur des cas concrets mais hypothétiques. À cet effet, il est question dans ce travail de permettre d'optimiser le parcours d'une flotte de drones de livraison dans Montréal.

3 Mise en situation

Le même étudiant que vous connaissez bien du TP précédent est toujours en train de faire son stage auprès de la start-up de drones. Il est cette fois-ci en charge de traiter les requêtes de livraison d'un point de vue plus logistique : il doit s'occuper de la portion du logiciel qui gère les requêtes et leur acheminement. Parmi les requis auxquels son logiciel doit répondre, on retrouve les tâches suivantes : il doit s'assurer que les adresses de livraison soumises par les clients figurent parmi la banque d'adresses

postales de Montréal, puis il doit attribuer des colis à un drone disponible sans dépasser les capacités physiques du drone et finalement il doit gérer la répartition des drones dans toute la ville. Il sait que les notions du cours de LOG2810 et tout spécialement le module de la théorie des langages seraient très pertinents à son travail, mais il est débordé de travail et a donc encore besoin de votre aide sans quoi il ne finira pas son projet à temps. Afin de vous guider un peu, il fait ses recherches et trouve les codes postaux valides à travers la ville de Montréal. Il vous donne aussi les spécifications des quadricoptères et un court rappel des notions du cours de LOG2810 sur les automates. Ainsi, vous pourrez 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.

- Chaque drone peut porter un poids maximum. La forme des objets transportés n'est pas prise en compte et ne présente aucun intérêt.
- Chaque centre de dépôt de colis pour les livraisons correspond à un code postal de départ qui sera adjoint à une requête faite à partir de ce point. Une erreur pourrait se produire dans l'adresse de départ (colis mal étiqueté), et il faut donc la valider.
- Chaque quartier de Montréal possède un centre de dépôt.
- Le client qui fait une requête de livraison a indiqué un code postal de destination qui n'est pas nécessairement valide (il n'existe peut-être pas dans la banque d'adresses).
- Un cycle de requêtes correspond à la lecture d'un fichier contenant des requêtes puis à l'exécution de ces dernières. Après chaque cycle de requêtes, il est nécessaire d'effectuer le rééquilibrage de la flotte de drones entre les différents centre de dépôt.
- Un cycle de requêtes permet au nombre de drones souhaité, dans la limite des drones disponibles, de se rendre à une zone de livraison. Dans une optique d'optimisation, on essaie toutefois de limiter le nombre de drones qui se déplacent à chaque cycle.
- Les drones de livraison peuvent faire le transport de plusieurs colis dans un cycle, mais il faut prendre en compte le nombre d'objets transportés et la destination de ces objets. En effet, pour chaque colis supplémentaire transporté par le drone, il restera indisponible pendant un cycle de requêtes supplémentaire. Également, les colis groupés lors d'une livraison doivent aboutir dans le même quartier.
- Une fois toutes ses livraisons effectuées, le drone réapparaît avec un statut disponible dans le centre de dépôt du quartier où se trouve le code postal de sa dernière livraison. Les drones ayant fini leurs livraisons retournent à un centre de dépôt juste avant le rééquilibrage et peuvent donc participer à ce dernier.
- Les drones se partagent en deux catégories. Les drones de catégorie 1 (faible capacité) peuvent transporter jusqu'à 1kg. Les drones de catégorie 2 (forte capacité) peuvent transporter jusqu'à 5kg.
- La startup possède dix drones de catégorie 1 et cinq drones de catégorie 2.
- Chaque colis à livrer a un poids qui lui est associé.
- Il faut toujours vérifier si le poids des colis transportés en groupe respecte la capacité maximale des drones.

- Advenant qu'aucun drone ne soit disponible dans un centre de dépôt pour effectuer la livraison d'un ou de plusieurs colis, ces requêtes doivent être reportées au cycle suivant.
- Les colis doivent être livrés en toute priorité selon le principe FIFO (*first in first out*).
- Le fichier `CodesPostaux.txt` contient les codes postaux considérés comme valides. Un automate qui reconnaît ce langage doit être construit. L'automate ne doit reconnaître que les codes postaux se trouvant dans `CodesPostaux.txt`.
- Le système doit bien réagir face aux requêtes des utilisateurs qu'il lit à partir de fichiers. Les fichiers de requêtes sont du type `requeteX.txt` et leur format est expliqué en annexe. Un fichier contient toutes les requêtes données pour un cycle de requêtes. Il n'y a pas d'ordre particulier pour la lecture de ces fichiers ; l'ordre sera déterminé au courant de l'exécution du programme par l'utilisateur.

Voici un schéma possible d'un automate reconnaissant les mots 'H3T' et 'H3S' :

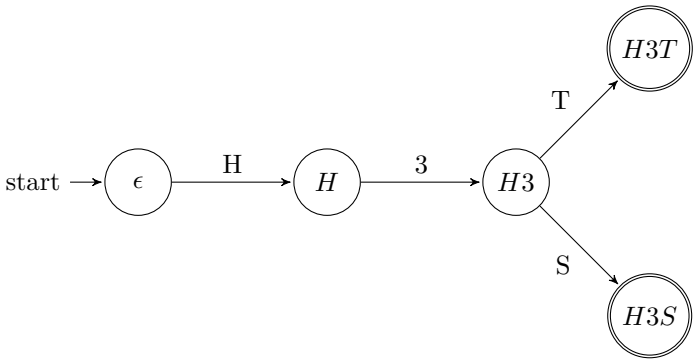

FIG. 1: Automate de validation

Comme mentionné précédemment, le fichier `CodesPostaux.txt` contient une liste de codes postaux valides se trouvant à Montréal. L'automate doit être capable de reconnaître chaque code postal de cette liste, et seulement les codes postaux de cette liste. Un automate reconnaît un code postal lorsqu'il existe une suite de transitions allant de l'état initial de l'automate à une sortie qui permet de former le code postal (état final de l'automate). Ordre : la première partie du code postal doit correspondre à la première transition, et ainsi de suite. Un **code postal** est une chaîne de six caractères alphanumériques, qui utilise le format A1B2C3, alternant lettres et chiffres.

5 Composants à implémenter - Automates

C1. Écrire une fonction « creerArbreAdresses() » qui lit le fichier texte contenant des adresses postales et qui crée l'automate responsable de valider les codes postaux soumis par les clients pour la livraison d'un colis. Cette fonction prend en paramètre le nom du fichier à lire.

C2. Écrire une fonction « equilibrerFlotte() » qui permet d'équilibrer le nombre de quadricoptères présents dans chaque quartier de la ville.

C3. Écrire une fonction « assignerLesColis() » qui permet d'assigner des colis à un drone en respectant les limitations physiques du drone et en s'assurant que tous les colis d'un drone vont vers le même quartier.

C4. Écrire une fonction « traiterLesRequetes() » qui lit le fichier texte contenant des requêtes d'usagers qui souhaitent envoyer des colis, qui valide les requêtes, qui assigne les colis à des drones et qui envoie les drones vers les stations d'autres quartiers. Cette fonction prend comme paramètre le nom du fichier contenant les requêtes à traiter.

C5. Écrire une fonction « imprimerLesStatistiques() » qui indique les items suivants dans le format précisé à l'annexe :

- Le nombre de requêtes traitées depuis le lancement du programme ;
- Le nombre de requêtes invalides depuis le lancement du programme ;
- Le nombre de drones dans chaque quartier suite au rééquilibrage ;
- Le nombre moyen de colis transportés par un drone à petite capacité depuis le lancement du programme ;
- Le nombre moyen de colis transportés par un drone à grande capacité depuis le lancement du programme ;

Ce requis est optionnel (vous ne pouvez pas perdre de points, juste en gagner), mais fortement conseillé puisqu'il aide au dépistage des erreurs.

C6. Faire une interface console qui affiche le menu suivant :

(a) Créer l'automate.
(b) Traiter des requêtes.
(c) Afficher les statistiques.
(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 un nouveau fichier de codes postaux afin de créer l'automate correspondant. Il est possible que de nouveaux codes postaux s'ajoutent ou s'enlèvent à la banque des adresses, alors le mécanisme conséquent doit être disponible dans votre code. Pour lire un nouveau fichier de codes postaux, le nom du fichier doit être demandé à l'utilisateur.
- L'option (b) permet de lire un fichier contenant des requêtes, de traiter les requêtes et de déplacer les drones vers de nouveaux quartiers où ils doivent effectuer leur livraison. À la fin de l'exécution de cette option, les drones seront redistribués dans la ville pour rééquilibrer la flotte et s'assurer d'avoir toujours des machines prêtes à opérer dans chaque secteur.
- L'option (c) permet d'afficher les statistiques sur l'état du système de livraison.

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.

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), option (b)), 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.
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, etc. Vous pouvez également indiquer le temps passé sur ce TP à des fins d'ajustement pour la prochaine 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.	3
Composants implémentés : respect des requis, logique de développement, etc.	
C1	3
C2	3
C3	2
C4	3
C5	2
C6	2
Total de points	18

8 Documentation

- `https://www.draw.io/` (Diagrammes UML)
- `https://www.lucidchart.com/pages/examples/uml_diagram_tool` (UML)
- `learnxinyminutes.com` (Apprendre rapidement les bases d'un nouveau langage)

Annexe

1 Affichage des statistiques

L'affichage des statistiques doit se faire selon le format suivant :

```
/* AFFICHAGE DES STATISTIQUES */
REQUETES TRAITEES  : X
REQUETES INVALIDES : Y

------------------------
REPARTITION DE LA FLOTTE
Quartier  |  Drones faible capacite  | Drones forte capacite
A3A 3A3   |  ...                     | ...
B7B 7B7   |  ...                     | ...
.
.
------------------------

NOMBRE MOYEN DE COLIS PAR DRONE :
FAIBLE CAPACITE : P
FORTE  CAPACITE : Q
/* FIN */
```

2 Format des fichiers de requêtes

Chaque fichier `requeteN.txt` comporte plusieurs lignes (le nombre de lignes varie). Sur chaque ligne, il y a trois éléments séparés par des espaces :

1. Code postal de la zone de départ, au format A1B2C3 (donc sans espace au milieu du code postal)
2. Code postal de la zone d'arrivée, au même format
3. Poids (en g) du colis