POLYTECHNIQUE
MONTRÉAL

LE GÉNIE
EN PREMIÈRE CLASSE

Rapport TP 2 - Automates

Dans le cadre du cours
Structures Discrètes – LOG2810

Marc-André Jolicoeur - 1846304
Alexandre Boudreault - 1486776
Mathieu Bélanger - 1850591

28 novembre 2017

Introduction

L'objectif de ce travail est de nous familiariser avec les notions d'automates, de structures de données et de théorie des langages que nous avons vus en cours. Le même stagiaire a encore besoin de notre aide, cette fois-ci pour un programme d'une compagnie de drone. Le travail est d'optimiser (d'un point de vue logistique) le parcours d'une flotte de drone de livraison à travers Montréal.

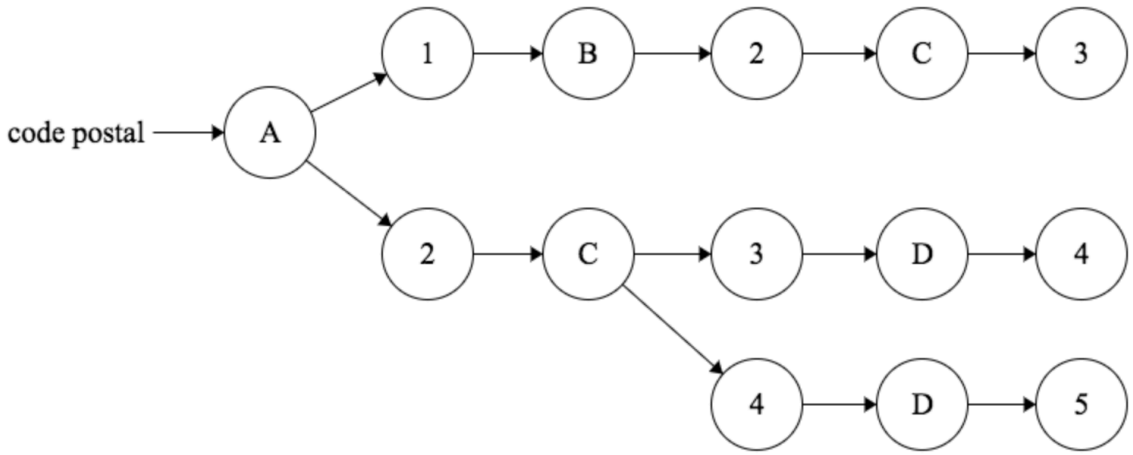
L'application est implémentée en Python. Il s'agit d'une application console qui gère les requêtes de livraison et s'assure d'optimiser leur acheminement. Elle implémente aussi un automate qui ne reconnait que les codes postaux provenant d'un fichier spécifié par l'utilisateur (ayant le format du code postal canadien standard).

Présentation de l'application produite

L'interface offre d'abord 4 choix à l'usager : (a) Créer l'automate, (b) Traiter des requêtes, (c) Afficher les statistiques et (d) Quitter.

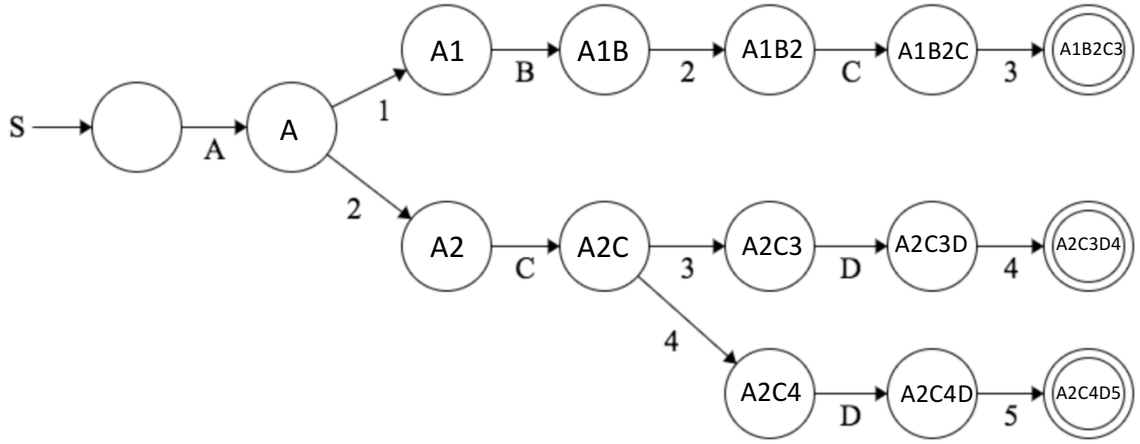
L'option **Créer l'automate** crée une banque de codes postaux à partir d'un fichier texte précisé par l'utilisateur. D'abord, l'automate vérifie si les caractères du code sont bien disposés selon les critères d'un code postal : 6 caractères ayant une alternance de lettres et de chiffres (A1B2C3). Si les critères sont respectés, le code postal peut être enregistré. La façon dont ont été enregistré les données est particulière : elle consiste en un arbre à 6 niveaux, un pour chaque caractère du code postal. Ainsi, supposons les codes « A1B2C3 », « A2C3D4 » et « A2C4D5 », « A » se retrouve au premier niveau et contient les chiffres qui lui sont associés : « A » : {1,2}. Ensuite, « 1 » qui se situe au deuxième niveau contiendra « B » et le « 2 » contiendra « C » et ainsi de suite.

Exemple de banque de code postaux



Ceci nous permet de transformer la recherche des codes postaux en automate. En effet, chaque lettre du mot permet à l'automate de changer d'état. Si le caractère n'est pas reconnu, le mot est refusé. C'est seulement lorsque le mot est rendu au 6^{e} état qu'il est validé par l'automate. Il s'agit en fait du principe de reconnaissance d'un langage, notion que nous avons abordée dans la théorie des langages. Ainsi, l'automate créé ne reconnait que les codes postaux spécifiés, et aucun autre mot.

Exemple d'automate qui reconnait les codes postaux « A1B2C3 », « A2C3D4 » et « A2C4D5 »



Une fois l'automate établi, l'application procède en créant une flotte de drone à un endroit choisi, comme le garage de la compagnie par exemple. Pour l'instant, ce « garage » est le premier code postal valide enregistré dans l'automate. Ensuite, deux types de drones sont créés : les drones de catégorie 1 qui peuvent transporter jusqu'à 1000 g et ceux de catégorie 2 qui transporte jusqu'à 5000 g. Dix drones de catégorie 1 et cinq de catégorie 2 sont ajoutés à la flotte de drones. Les drones sont ensuite envoyés vers des centres de dépôt aléatoires pour mieux servir les futures requêtes. Ainsi, il est plus probable qu'un drone se trouve déjà à l'origine d'une commande au moment de traiter les requêtes.

L'option **Traiter des requêtes** coordonne les commandes de livraison qui se trouvent dans un fichier texte (spécifié par l'utilisateur). Les commandes doivent suivre le format ci-dessous :

Code Postal de l'Origine *Code Postal de la Destination* *Poids du colis*

Les codes postaux de l'origine et de la destination sont injectés dans l'automate, qui en vérifie l'appartenance au langage créé précédemment. Si les codes sont reconnus par l'automate, un objet de type *Delivery* est créé avec les propriétés *origin*, *destination* et

weight, puis est ajouté à la file d'attente *request_queue*. L'opération est exécutée pour chaque ligne du fichier texte. Les commandes qui n'adhèrent pas au format requis ou qui comportent un (ou plusieurs) code postal invalide sont ignorées. Le nombre de commandes ignorées est sauvegardé et mis à jour à chaque appel de la fonction *traiter_requetes*. Il est à noter que l'automate doit être créé avant de traiter les données, sinon un message d'erreur est affiché.

Après avoir traité un fichier de requêtes, l'application procède à la distribution des tâches. Chaque élément de la request_queue (un objet de type Delivery()) est traité une à la fois. Si un drone se situe déjà à l'origine de la livraison (préférablement un petit drone), la livraison lui est attribuée. Si l'attribution de la livraison est impossible pour une raison quelconque (poids trop lourd pour drone, drone a déjà trop de colis, aucuns drones ne se situent à l'origine de cette livraison), un drone est assigné à aller récupérer le colis et la livraison est ajouté à une liste temporaire nommée *temp_queue*. S'il est impossible d'envoyer un drone vers l'origine d'une livraison, la livraison est ajoutée à la queue temporaire pour être traitée lors du prochain cycle. Une fois que les tâches sont tous distribuées (livraison ou récupération), les drones qui ne sont pas assignés à une tâche sont assignés une destination selon les emplacements qui ne seront pas visités lors de ce cycle (soit la fonction *reequilibrate_fleet*). Le contenu de la file temporaire est injecté dans la file *request_queue* afin de préserver leur priorité sur les prochaines requêtes. Finalement, le signale est donné aux drones. L'appel à la fonction *deliver_packages* de la classe DroneFleet ordonne aux drones de partir vers leur destination, que ça soit pour effectuer une livraison, récupérer un colis pour une livraison lors du prochain cycle, ou pour simplement assurer une uniformité du nombre de drone dans les arrondissements.

Il est possible qu'un drone se charge de la livraison de plusieurs colis en même temps, pourvu que ces colis proviennent de la même origine, sont destinés vers le même code postal et que leur poids combiné ne dépasse pas la capacité maximale du drone en question. Lorsque le drone arrive à sa destination par contre, il peut seulement déposer un colis par cycle. De cette manière, il prend donc une pénalité de 1 cycle par colis additionnel transporté. Après avoir complété toutes ses livraisons, un drone redevient disponible.

L'option **Afficher les statistiques**, comme son nom l'indique, permet d'afficher différentes informations sur la flotte de drones et les commandes traitées depuis le lancement du programme.

- Le nombre de requêtes traitées
- Le nombre de requêtes invalides

- Le nombre de drones dans chaque quartier suite au rééquilibrage
- Le nombre moyen de colis transportés par un drone à petite capacité
- Le nombre moyen de colis transportés par un drone à grande capacité

Pour arriver à garder ces données en mémoire, un objet de type « *RecordKeeper* » est créé et mis à jour pendant le traitement des requêtes.

Le menu est affiché tant que l'option **Quitter** n'est pas sélectionné.

Difficultés rencontrées

Une difficulté majeure du travail pratique est le grand nombre de conditions qui sont prises en compte à chaque opération. En effet, il faut d'abord vérifier si les codes postaux de la commande sont valides. Ensuite, il faut vérifier si un des drones de la flotte est disponible pour la livraison. Il faut vérifier si d'autres colis ont la même destination afin d'optimiser les déplacements d'un drone. Il faut ensuite vérifier si la capacité du drone permet l'ajout du nouveau colis et ainsi de suite.

Une difficulté fut l'écriture du code tout en clarifiant l'énoncé. Au fur et à mesure que le projet avançait, les informations étaient plus claires et les travaux effectués ont dû être modifiés pour se conformer aux requis.

De plus, le fait qu'un drone puisse transporter plusieurs colis à la fois rend le rééquilibrage plus complexe. Par exemple, si deux colis (de petite taille) ont les mêmes origine et destination, un seul drone pourrait transporter les deux colis. Ainsi, il faut que lors du traitement des requêtes, le programme fasse preuve d'une certaine intelligence afin d'éviter l'envoi de deux drones dans un tel cas. Plutôt, pour minimiser les déplacements des drones, il faut regrouper les livraisons avec la même origine et destination et les assigner à un seul drone, en tenant compte des deux masses de colis.

Structures de données et classes

- PostalCodeAutomaton s'occupe de la validation des codes postaux et de la banque de codes postaux valides.
- Delivery : Objet de base d'une requête, il contient le lieu de départ, d'arrivée et le poid d'une requête.
- DeliveryRequest : Gère la Queue de requêtes venant des fichiers d'entrée. Effectue les traitements de répartition des drones et d'assignement des tâches de livraison
- Drone : Objet de base effectuant les livraisons. Peut être de différente taille (poids de livraison maximale)
- DroneFleet : Coordonne les drones en symbiose avec DeliveryRequest
- RecordKeeper : Garde un historique des statistiques de livraison

Conclusion

Nous sommes encore venus à l'aide de notre collègue. La prochaine fois, nous allons devoir lui charger pour notre travaille. Le programme gère de façon autonome la flotte de drone de la compagnie. En fonction d'une banque de codes postaux, les requêtes sont validées et traitées par la flotte de drones. Un historique des statistiques est cumulé et peut être affiché à la demande de l'utilisateur.

De plus, le programme est très flexible en permettant notamment de modifier le nombre de drones de chaque type, d'ajouter un nouveau type de drones, de mettre à jour la banque de codes postaux reconnus ainsi que l'emplacement du garage de la compagnie. Ainsi, en cas d'expansion, la start-up pourra augmenter le nombre de drone dans sa flotte, ou changer la localisation du garage de la compagnie. En mettant à jour l'arbre de codes postaux, de nouveaux centres de dépôt peuvent être ajoutés à la banque de données.