

24/10/2017

Rapport TP1 LOG2810

(STRUCTURES DISCRETES)

**Auteurs :**

Etienne Laval 1916458

Constantin Bouis 1783438

Xavier Ralay 1786307

**Tables des matières :**

[1.Introduction 2](#_Toc496648259)

[2. Drones 3](#_Toc496648260)

[3.Recettes 4](#_Toc496648261)

[4.Difficultés et solutions 5](#_Toc496648262)

[5.Conclusion 5](#_Toc496648263)

# **1.Introduction**

Dans le but de venir en aide à un étudiant du département du génie informatique et génie logiciel, nous avons mis au point un projet qui répond a ses attentes. Ce dernier a notamment eu besoin de notre aide dans le cadre de ses stages.

Pour commencer, venant juste de se trouver un stage auprès d’une start-up de drones qui aimerait populariser les livraisons de colis sur l’île de Montréal par des drones donc par voies aériennes, la start-up le met déjà en charge d’un grand projet. En effet, il sera en charge d’un algorithme ayant pour but d’optimiser les chemins qu’emprunteront les drones lors de la livraison d’un colis pour ainsi minimiser les coûts d’exploitation. Cet algorithme rappelle plusieurs notions théoriques de graphs du cours de LOG2810: Structures Discrètes. Cependant, ayant stipulé dans son C.V avoir réussi le cours avec succès, le nouveau stagiaire n’a malheureusement pas très bien assimilé les notions du cours. Il a donc désespérément besoin de notre aide. Pour cela, nous l’avons aidé à implémenter quelques composantes de son algorithme.

Ensuite, l’étudiant ne s’en arrête pas là. En effet, dans le cadre de son prochain stage, cependant cette fois ci dans une autre compagnie. Pour ce stage, le jeune étudiant doit remettre un autre programme ou il s’agit encore de notions du cours de Structures Discrètes. En effet, son programme doit pouvoir déterminer un ordre et afficher un diagramme de Hasse d’après une liste de sommets et d’arêtes orientées entre ces sommets dont il aura tracé le graphe. Il a donc encore besoin de notre aide.

Par la suite, nous allons vous expliquer les moyens que nous avons utilisés pour mener à bien ce TP. Puis lors de l’élaboration du TP, nous avons rencontré notamment quelques difficultés que nous vous expliquerons ensuite.

# **2. Drones**

Pour la mise en place de l’architecture des graphs, nous avons repris le travail qu’Etienne Laval avait déjà fait pour un autre projet d’algorithmique dans son école en France. (ConsultablesurGithubhttps://github.com/AlgoAvanc/ProjetAlgo).

Il s’ agit de largement en élaguer les morceaux inutiles puis de construire les fonctionnalités requises par ce TP dessus.

Cette architecture va comme suit :

Trois classes : Graph, Edge et Node.

Les nodes (nœuds) contiennent leur identifiant (un String) ainsi que mention des edges (arcs) qui partent d’eux sous la forme d’une liste.

Les edges sont orientés, ils contiennent une référence à leur nœud de départ et à leur nœud d’arrivée ainsi qu’un poids.

Les graphs contiennent trois listes : celle de leur edges, celle de leur nodes et celle des id de leur nodes (pour les retrouver plus rapidement).

A cela nous avons adapté un module de lecture de fichiers .txt et ajouté le fait d’avoir des nœuds de recharges de drones (par booléen sur les nœuds)

Ayant la fonction de création de graph, la mise à jour du graph n’est qu’une recréation de ce graph.

Pour ce qui est de l’implémentation du calcul de chemin par un drone c’était assez simple, il y a deux variantes par rapport au Dijkstra proposé :

-On tient aussi un compte de la charge de batterie une fois arrivée à un nœud (en plus de renseigner le nœud précédent et la distance/temps au premier nœud)

-On ne met à jour la distance/temps à un nœud que si celui-ci est assez proche pour que le drone ne tombe pas en panne (on ajoute bien sûr le temps de recharge dans le décompte du temps le cas échéant, de même que la recharge elle-même).

Pour l’utilisation de la batterie, on entre en paramètre le taux de consommation (en part de batterie par minute de la technique testée). Ainsi, si une fois l’algorithme testé, le nœud visé a un parent dans l’algorithme de Dijkstra, alors, il y a bien une possibilité de trajet. Si ce n’est pas le cas, on le réessaie avec en taux de consommation d’énergie, celui des drones plus puissants et si cela ne marche toujours pas, on sait que le voyage est impossible.

En termes d’architecture cet algorithme et les tableaux de temps/précédent/consommation sont dans une classe (nommée DijkstraPlus). Ainsi, on peut faire correspondre les données déjà calculées dans un objet rattaché à chaque nœud.

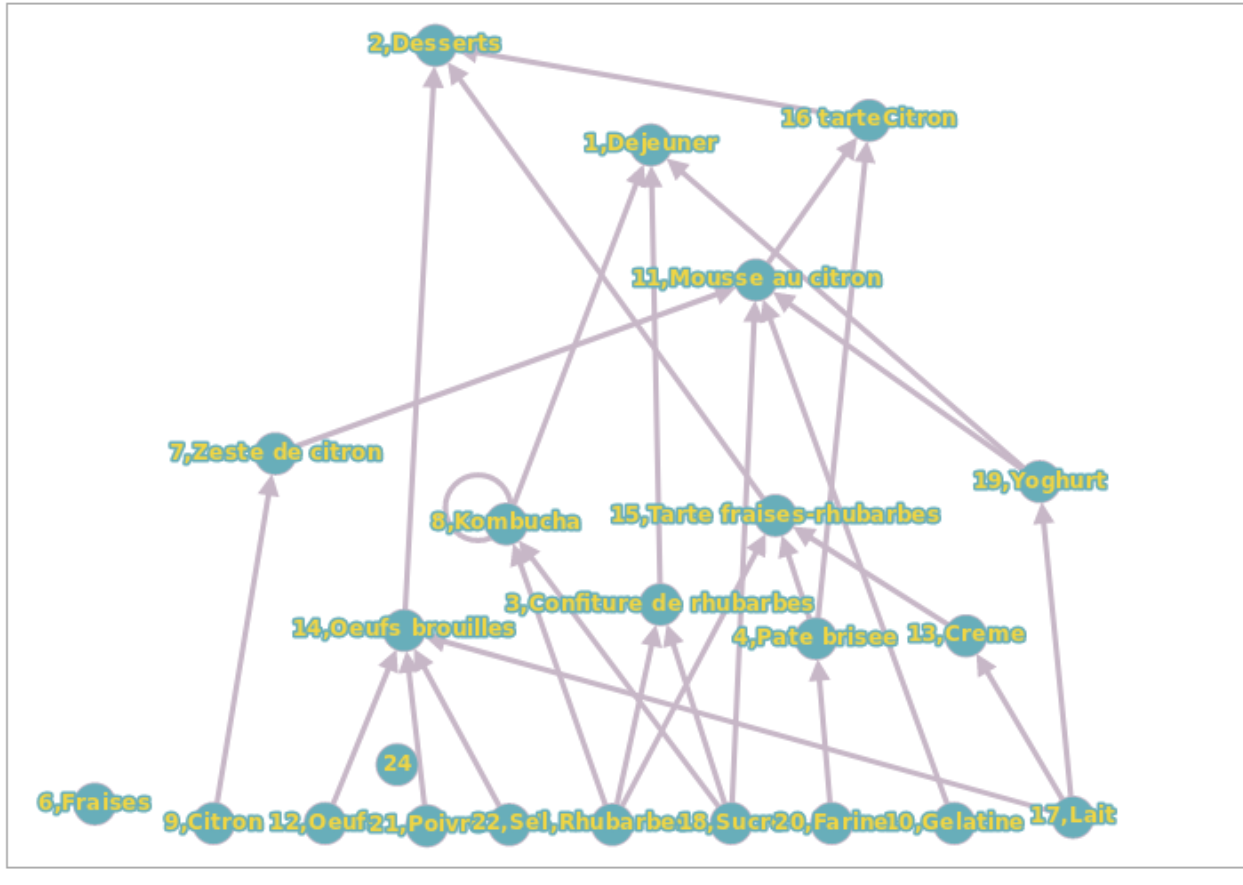
# 3.Recettes

- Pour la partie " Déjeuner et desserts" nous avons pu prendre la fonction creerGraph et la changer légèrement pour l'adapter à des graphs orientés.

Nous avons également apporté une légère modification à la fonction LireGraph ce qui nous permis de lire les deux fichiers textes (exercice 1 et 2) à partir d’une même fonction.

Ainsi dans la classe Hasse, nous n'avons eu qu'à implémenter les fonctions qui représentent les étapes de construction d'un diagramme de Hasse. Tout d'abord nous avons la fonction qui permet de retirer la réflexivité, puis nous avons la fonction qui permet de retirer la transitivité. Ensuite, dans la même fonction qui affiche le diagramme de Hasse, nous avons implémenté un algorithme permettant d’afficher nos éléments dans un ordre valide.

Voici le graphe généré à partir du fichier manger.txt, c’est sur ce graphe que nous avons appliqué les différentes étapes de construction d’un diagramme de Hasse



Enfin, pour le main(), nous avons opté pour des switch-case imbriqués dans des boucles while pour y afficher en tout temps le menu.

# **4.Difficultés et solutions**

C’est donc comme cela que nous avons pu élaborer ce TP. Cependant, on a tout de même rencontré plusieurs problèmes lors de cette élaboration. Pour commencer, une des principales difficultés est l’organisation. En effet, n’ayant pas bien organisé nos périodes de réunion cela a beaucoup affecté notre efficacité. Cependant, pour y remédier, nous avons établi à l’aide d’un logiciel, un emploi du temps des disponibilités de chacun pour ainsi avoir une meilleure vue sur la fixation des réunions. Ensuite, le non-respect du temps prévu pour l’accomplissement de la tâche à accomplir est un des facteurs nocifs à notre performance. Cependant, pour faire face à cette difficulté, nous avions dû fixer les objectifs à atteindre au début de chaque réunion. Par suite, la difficulté au niveau de la compréhension des énoncés nous a beaucoup retardé surtout lors du commencement de l’élaboration de notre TP. Ainsi, pour y remédier nous avons fixé les objectifs principaux du TP. Enfin, le fait que la majorité du groupe n’était pas familier avec le langage Java a eu un gros impact sur l’efficacité du travail de groupe. Cependant faire ce TP nous a permis de gagner en expérience. En effet, déjà le fait de coder en Java a permis à la partie de l’équipe de se familiariser avec ce langage. De plus, le fait de mieux s’organiser nous a appris une meilleure méthode de travail pour les prochains TP .

# **5.Conclusion**

Finalement, ce TP nous a permis de revoir les bases d’un travail en équipe, autant du point de vue de l’organisation que du point de vue sociale et relationnel. Nous avons pu revoir certains aspects du cours log2810 tels que l’algorithme de Djikstra ou encore les diagrammes de Hasse. Ce TP nous a également permis de revoir des anciens concepts vus dans les cours précédents. Enfin, nous avons pu mieux comprendre certaines spécificités de Java, le langage de programmation que nous avons utilisé ici.