# Rapport de Mini Projet

21006

XIANG YANGFAN

3300401

Mercredi 18 février 2015

# I. Description globale de mon code

Le code du projet est constitué globalement de deux parties. La première est l'ensemble de fichier pour la structure de Liste composé de *liste.h* et *liste.c* qui implémente tous les fonctions nécessaires pour la manipulation des listes. La deuxième est les fichiers pour la structure de Tableau de Hachage composé de *tabHachage.h* et *tabHachage.c* ainsi que deux jeux de test *mainTest* qui test que tous les fonctions implémenté fonctionne bien et ne produise pas d'erreur et *mainCompare* qui compare l'efficacité de fonction *rechercheLivreEnPlusDeDouble* pour les deux structures. Mon code est également accompagné d'un *Makefile* pour automatiser la compilation.

## II. Quelque explication sur les questions du sujet

Pour la partie de Tableau de Hachage, le sujet nous impose qu'il faut implémenter les fonctions de recherche d'un livre par son numéro et par son titre comme dans le cas de la liste. Mais comme la clef de hachage est déterminé par l'auteur du livre alors on perd l'intérêt de faire un tel recherche par son numéro et par son titre car dans ces cas là la recherche revient au même cas de la liste à savoir de parcourir tout la structure pour trouver le livre correspondant. Donc au final je n'ai pas implémenté ces deux fonctions pour le Tableau de Hachage.

### III. Description des jeux d'essais

#### 1. mainTest

Mon premier jeu d'essais consiste à tester toutes les fonctions implémentées pour les deux structures et de vérifier qu'ils sont bien valides. Je commande d'abord à l'ouverture du fichier bibliothèque *GdeBiblio.txt* puis un première lecture des livres en quantité limitée avec un affichage de ces livres qui vient d'être lu et suivi de deux suppressions par numéro et par titre avec un affichage pour vérifier la liste des livres.

Un deuxième lecture en plus grande quantité pour ensuite effectuer une recherche des livres de même auteur de nom définie par le #define au début du programme. Et le jeux se termine par la recherche des livres en plus de double avec affichage et la durée émis pour le jeux de teste. Même procédure pour la structure de Tableau de Hachage.

#### 2. mainCompare

Le deuxième jeu d'essais commence aussi par ouvrir le fichier de bibliothèque. Pour un pas définie par #define et avec un total de 50000 livres lu, a chaque pas je teste la fonction de recherche des livres en plus de double pour les deux structures qui me donne le temps utiliser pour effectuer la recherche dans les deux cas. Il affiche aussi le temps et de l'écrire dans un fichier nommé resultat.txt.

## IV. Analyse de performance (fonction de recherche de plus de double)

J'ai pour la taille du tableau de hachage une valeur de m=100 pour avoir des temps de recherche plus significatif et plus facile a interprété.

Tableau représentant le temps émis pour la recherche en fonction de la taille de la structure de donnée.

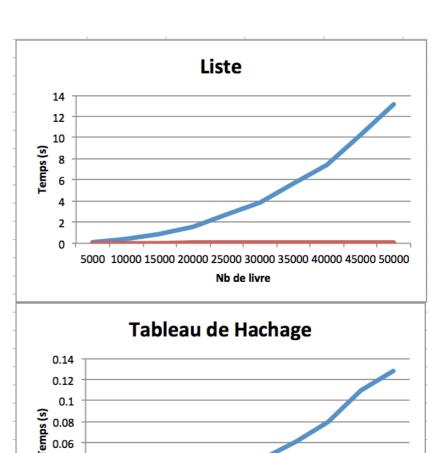
On remarque que le temps pour le tableau de hachage est la centième du temps pour les listes, ce qui n'est pas un hasard et correspond au m= 100 qu'on a choisi.

Nb de	livre	liste(sec)	hachage (sec)	
	5000	0, 099375	0,001172	
	10000	0, 397873	0,004636	
	15000	0, 90237	0,010121	
	20000	1,600213	0, 01899	
	25000	2,660013	0, 029832	
	30000	3, 832853	0, 045399	
	35000	5, 667694	0, 061227	
	40000	7, 463765	0, 07977	
	45000	10, 309776	0, 10959	
	50000	13, 140905	0, 127798	

La diminution du temps est donc en

fonction de la taille m qu'on a choisi puisqu'au final on divise la liste long en m liste plus petit en supposant que c'est bien une répartition uniforme. La recherche s'effectue donc uniquement dans cette liste plus petite puisque la hachage d'un même livre est la même.

En effectuant un analyse de complexité on retrouve que pour les listes la fonction à une complexité de  $O(n^2)$  car on a deux boucles imbriqués et pour le Tableau de Hachage une complexité de  $O(\frac{n^2}{m})$  avec n le nombre de livre et m la taille du tableau de hachage.



5000 10000 15000 20000 25000 30000 35000 40000 45000 50000 **Nb de livre** 

0.04 0.02 0