Rapport Mini-Projet

LU2IN006

Binôme :

- Noraiz AZIZ
- Wassim MARZOUGUI

Groupe : 6

Reformulation courte du sujet (introduction)

Ce mini projet s'étale sur deux semaines de travaux pratique. Il se focalise sur la gestion d'une bibliothèque contenant un ensemble de livres. L'objectif de ce projet est de comparer deux types de structures différentes. Pour tester le bon fonctionnement de ses deux structures on dispose du fichier 'GdeBiblio.txt' qui est une bibliothèque. Dans ce fichier chaque ligne est formée de la même maniéré c'est à dire un numéro qui correspond à l'identifiant suivie du titre et du nom de l'auteur. Ainsi on va voir dans un premier temps la gestion d'une bibliothèque en utilisant une liste chainée on va voir ensuite dans un second temps la gestion d'une bibliothèque en utilisant une table de hachage et pour finir on va comparer ses deux structures.

Gestion d'une bibliothèque avec une liste chainée.

Dans cette première partie du projet nous allons mettre en œuvre une bibliothèque sous la forme d'une liste chainée. Pour cela nous utiliserons deux structures, la première se nomme 'Livre' qui contient les champs **num, auteur, titre et suiv** un pointeur vers une structure de type 'Livre' représentant le livre suivant. La deuxième structure est une structure qui agit comme une tète fictif. Cette structure est composée d'un premier élément qui agit comme un élément artificiel qui est ajouté au début ou la fin de la bibliothèque. L'intérêt de cette structure est de faciliter les opérations sur la liste comme l'ajout ou bien la suppression. On a ajouté ses deux structures dans un fichier **'BiblioLC.h'**. On a par la suite ajouté dans un fichier qui se nomme **'biblioLC.c'** dans le quelle on a ajouté d'abord les fonctions de base pour manipuler des listes chainées comme la création d'un livre, sa suppression et son insertion. On a également défini des fonctions de créations d'une bibliothèque et sa libération. On a ajouté par la suite des fonctions plus spécifiques comme l'affichage ou bien la recherche d'un élément selon un critère donné en paramètre qui sont des fonctions assez basic. La fonction 'ouvrage_exemplaire' est une fonction qui nous a demandé plus de réflexion qui est une fonction censée renvoie les exemplaires ayant les même titre et auteurs, on nous impose dans l'énoncé que la complexité doit être dans le pire cas en O(n*n). Pour écrire cette fonction on a dû réfléchir à un algorithme qui utiliser deux boucles imbriquées pour comparer chaque élément de la liste. On a par la suite écrit un fichier **'entreeSortieLC.h'** qui comporte deux fonctions, une fonction qui permet de lire n lignes du fichier et une autre fonction qui permet d'écrire dans un fichiers. On a pour finir écrit un fichier main qui contient un menu qui affiche toutes les actions possibles sur la bibliothèque, on a également pensé à crée un Makefile pour automatiser le processus de compilation. Ainsi **pour compiler, il faut s'assurer que l'on est dans la racine /exo1 et utilisez la commande "make". Pour exécuter, on peut utiliser "./main GdeBiblio.txt n" où n est le nombre de lignes souhaitées.**

Gestion d'une bibliothèque avec une tableau de hachage.

Dans cette seconde partie on va voir la gestion de la bibliothèque par l'utilisation d'une table de hachage. Une table de hachage est une structure de donnée qui utilise une fonction de hachage pour associé à des valeurs des clés pour faciliter la recherche ou bien l'insertion d'éléments.

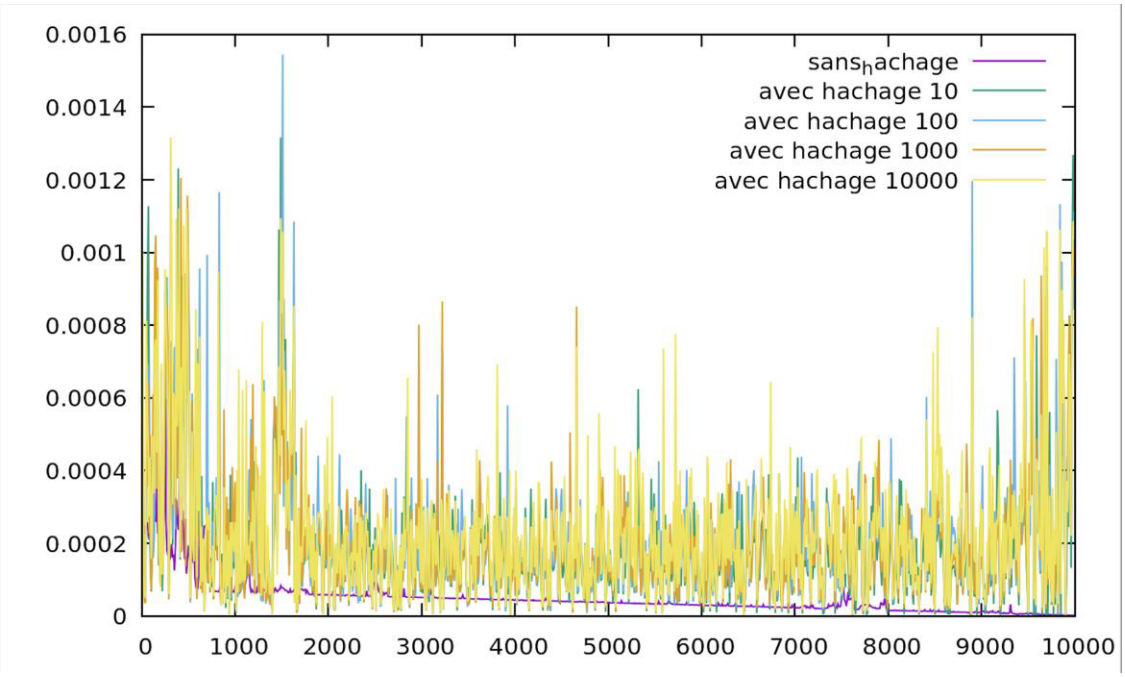
Dans cet exercice on a utilisé deux structures. La structure **'Livreh'** qui représente individuellement chaque livre, il comporte par rapport à la structure de la liste chainée un champ d'entier clés. La structure **'BiblioH'** représente l'ensemble de la table de hachage il contient notamment un champ qui

stocke le nombre d'éléments contenue dans la table, la taille de la table ainsi qu'un tableau de pointeur de type **'LivreH*'** représente la table de hachage. Chaque élément de ce tableau peut pointer vers une liste chainée de livres en cas de collision. Pour le champ clé elle corresponds à la somme de chaque lettre de son auteur en code ASCII. Dans la suite de cet exercice on a de l'adapté toutes les fonctions précédentes à la nouvelle structure qui utilise une table de hachage. Les deux seules fonctions qu'on ne retrouve pas dans la structure précédente sont la fonction de hachage qui permet de transformer la clé de chaque livre en une valeur entière .On nous a donné la fonction de hachage qui est h(k) = ⌊m(kA − ⌊kA⌋)⌋ ou k est la clé associée à un livre et A est le nombre d'or (sqrt(5)-½).L'autre fonction qui diffère est la fonction 'inserer' il faut pour cela trouver quelle case du tableau de hachage dans laquelle inséré l'élément et ensuite inséré le livre en tête de la liste chainée. On a également dû adapter le Makefile et le main pour cette exercice. De la même manière que précédemment **pour compiler il faut s'assurer que l'on ait dans la racine /exo2 et utilisez la commande "make". Pour exécuter, on peut utiliser "./main GdeBiblio.txt n" où n est le nombre de lignes souhaitées.**

Comparaison de deux structures.

Dans cette troisième et dernière partie l'objectif est de comparer les deux structures vues précédemment à savoir une liste chainée et une table de hachage en termes de temps pour effectuer des fonctions de recherche. On va effectuer une recherche simple et une recherche d'ouvrages avec plusieurs exemplaires. Nous avons également pris en compte deux scénarios qui sont la présence ou non d'un livre dans une bibliothèque.

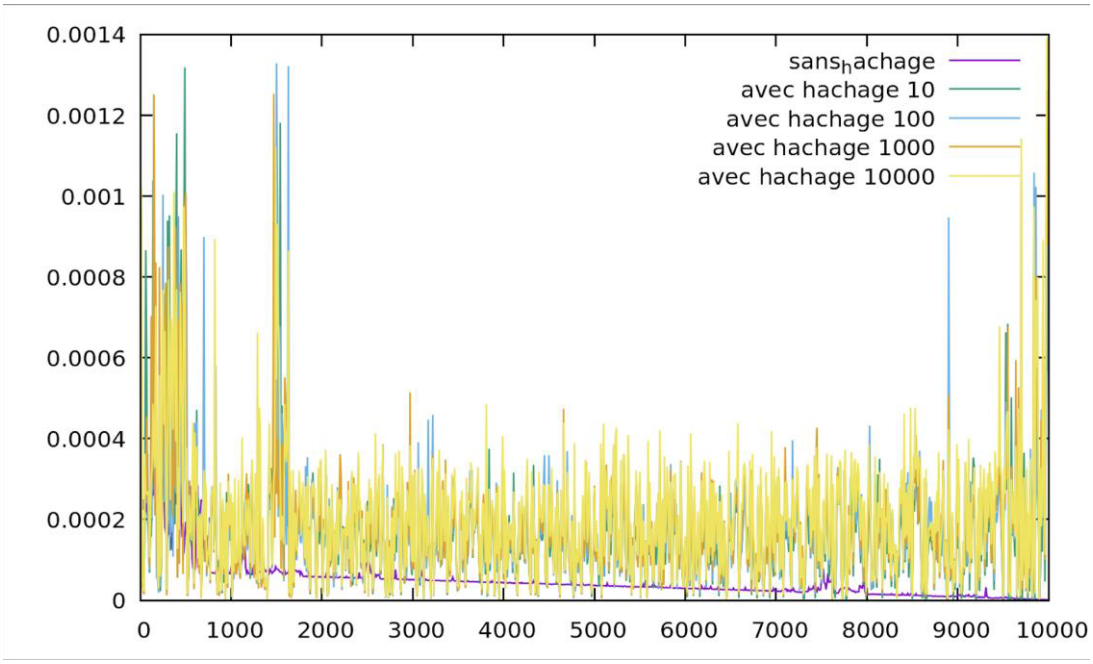
(Question1 fonctions dans 'exo3.c) Pour la recherche par numéro, la courbe de vitesse pour la recherche d'un ouvrage par numéro indique que la structure par liste chainée semble plus efficace en termes de temps, surtout lorsque la recherche concerne un numéro proche du dernier numéro. Cette performance est due au fait qu'une liste chainée est linéaire qui permet ainsi un accès direct au éléments consécutifs favorisant ainsi une recherche rapide.



(voir manuel d'utilisation)

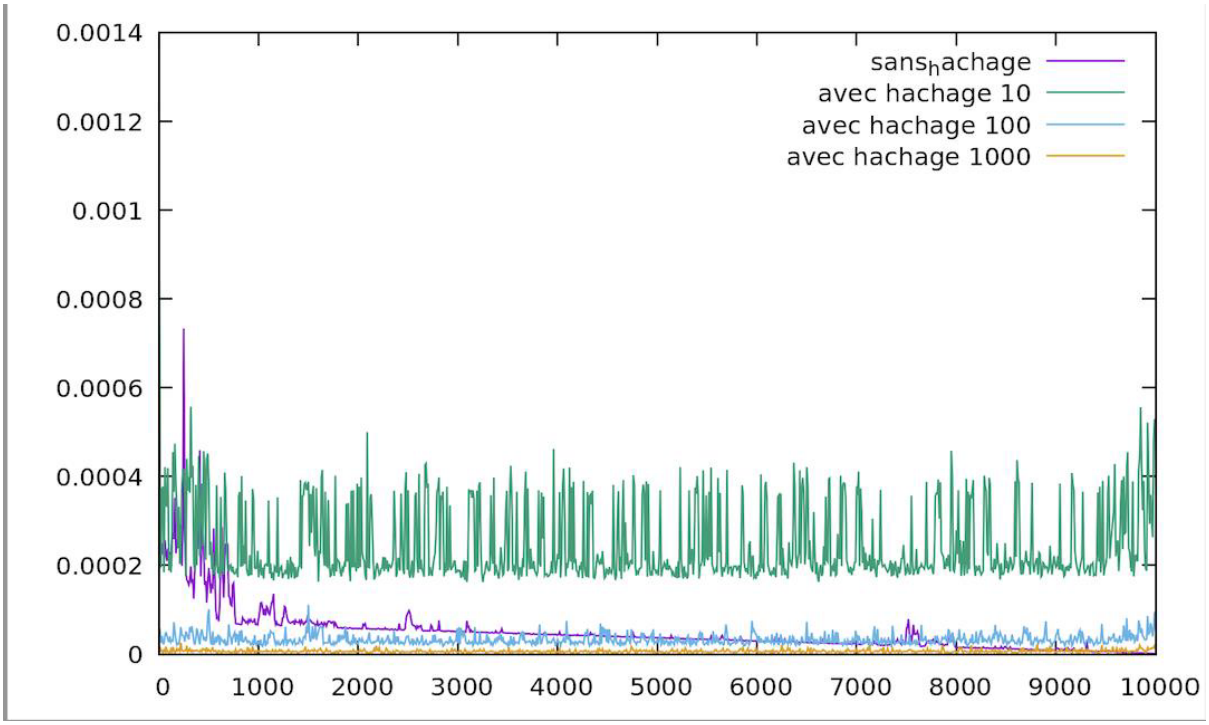
Pour la recherche d'un ouvrage par titre, la courbe de vitesse suggère une efficacité légèrement supérieure pour la liste chainée comparé à la table de hachage

(voir manuel d'utilisation)



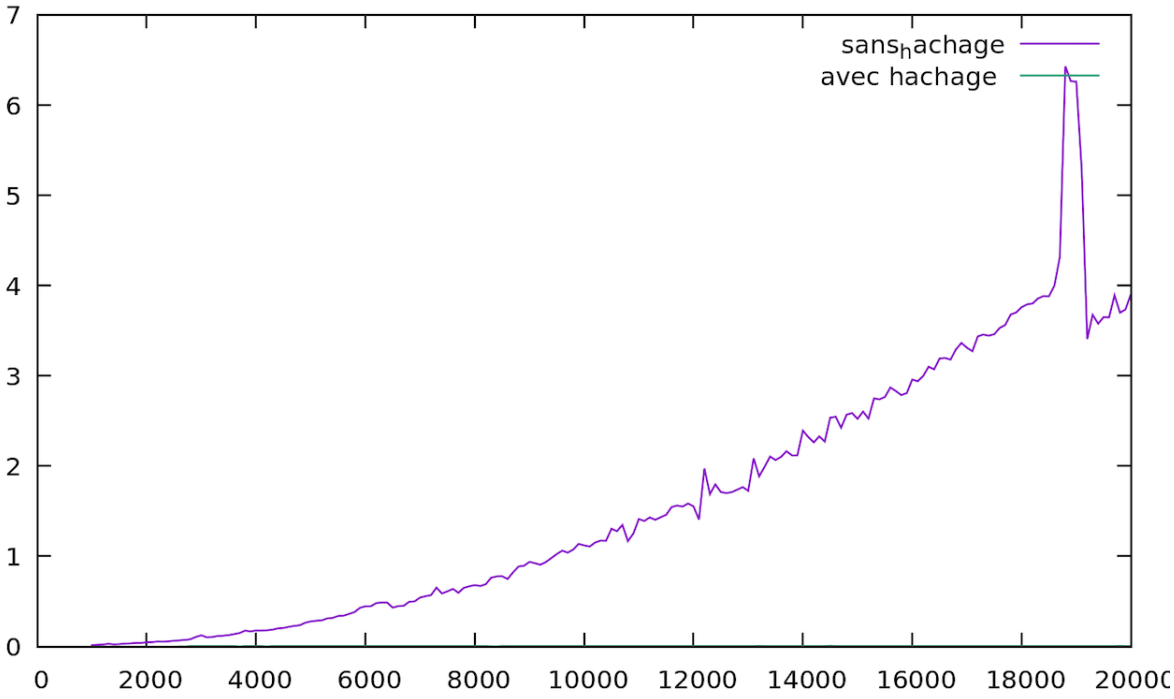
Pour la recherche d'un ouvrage par auteur, on voit que la courbe de vitesse montre que la structure par table de hachage est plus efficace. La nature de la table de hachage permet une localisation plus rapide des éléments puisque la fonction de hachage dépend de clef donc de auteur, ce qui devient plus avantageux pour la recherche par auteur (on recherche que dans la case correspondante).

(voir manuel d'utilisation)



(Question 2) Nous avons modifié la taille de la table de hachage est on observe très clairement que à chaque augmentation de la table, le temps de calcul diminue de manière significative. Ce qui peut être expliqué par le fait que plus la table est grands moins on a de collision, de même plus une table de hachage est grande plus les opérations de recherche sont rapides.

(Question3-4) On veut maintenant déterminer le temps de recherche des ouvrages en plusieurs exemplaires en fonction de la taille de la bibliothèque et de la structure de données utilisée (fichier **'exemplaires.c'**). La courbe de vitesse pour la recherche d'ouvrages avec exemplaires montre clairement que la structure par hachage est plus efficace surtout lorsque la table comporte un grand nombre d'éléments. Cela est due également à la différence de performance entre les deux structures, la table de hachage comporte un champ clé, ainsi la recherche est simplifiée car il suffit de localiser la colonne correspondant à l'auteur permettant un accès plus direct. En revanche pour la liste chainée l'approche est moins optimisée qui nécessite un parcoure intégral de tous les éléments de la bibliothèque. On remarque que plus la taille augmente plus l'efficacité de la structure par liste chainée diminue.



(voir manuel d'utilisation)