COHEN-SOLAL Léa 28604425
DE RICHEMONT Jules 28717601
SAKKOUR Zein 28707255

Compte rendu projet LU2IN006

Dans ce projet nous nous intéressons à la gestion d'une bibliothèque de livres.
Dans ce projet nous allons comparer l'efficacité de cette gestion avec des listes chainées et des tables de hachages.
Les caractéristiques d'un livre et d'une bibliothèque sont stockés dans un fichier BiblioLC.h (pour l'utilisation des listes chainées) et dans un fichier BiblioH.h (pour l'utilisation des tables de hachages).

```c
typedef struct livre{
    int num;
    char *titre;
    char *auteur ;
    struct livre *suiv;
} Livre ;

typedef struct {
    Livre* L ;
} Biblio ;
```

fichier biblioLC.h.

```c
typedef struct LivreH {
    int clef ;
    int num;
    char *titre;
    char *auteur ;
    struct LivreH * suivant ;
} LivreH ;

typedef struct table {
    int nE ;        //nombre d'elements dans la table de hachage
    int m ;         //taille de la table de hachage
    LivreH ** T ;
} BiblioH ;
```

fichier biblioH.h

Le fonctionnement du Makefile

1) Les fichiers .h

Les fichiers .h sont principalement constitués des prototypes des fonctions définis dans les fichiers .c.

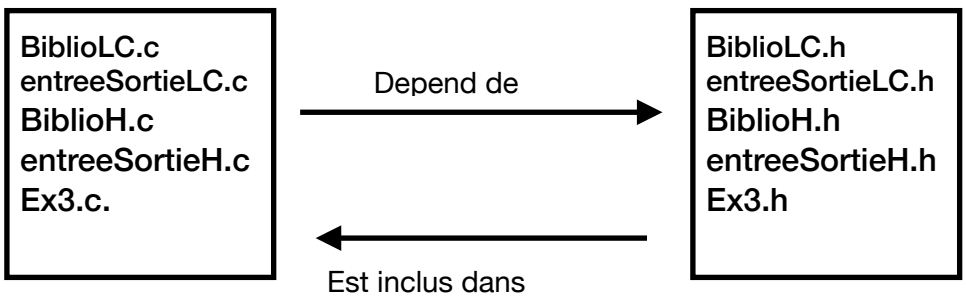
2) Les fichiers .c

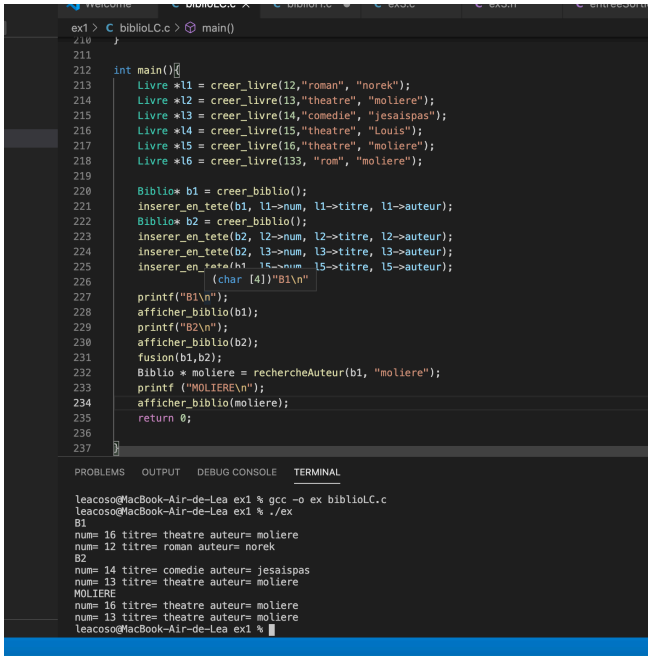
Les fichiers .c correspondent aux corps des fonctions différents prototypes présents dans les fichiers .h.

3) Makefile

Fichiers .c

Fichiers.h





Jeux d'essais sur le BiblioLC.c

Les différentes structures utilisées

Les listes chainées

La première structure utilisée dans notre programme est la liste chaînée.
Chaque livre de la liste chaînée sera constitué du numéro, du titre, du nom d'auteur ainsi qu'un pointeur sur l'élément suivant.

Tableau de hachage

La deuxième structure utilisée dans notre programme est le tableau de hachage.
Ici le livre aura comme caractéristiques un numéro, un titre, un auteur, une clé lui permettant de connaitre directement sa « place » dans la table de hachage (pour éviter les collisions) ainsi qu'un pointeur vers l'élément suivant.

Dans les deux structures, nous avons implémenté un certains nombre de fonctions qui ajoutent, retirent des livres d'une bibliothèque, recherchent des livres en fonctions de leur caractéristiques et peuvent même fusionner deux bibliothèques

Compilation et execution

Pour compiler la partie sur les listes chainées

```
Make all
./main GdeBiblio.txt 100
```

Pour compiler la partie sur les tables de hachage

```
Make all1
./mainH GdeBiblio.txt 100
```

Pour exécuter le main

```
./main GdeBiblio.txt 100
```

Pour exécuter le mainh

```
./mainh GdeBiblio.txt 100
```

Comparaison entre les 2 structures

Recherche dans le cas où le livre est absent

On réalise la recherche du livre suivant : « Bonjour » qui n'est pas présent dans la bibliothèque.

Voici un extrait des temps de recherches obtenu en fonction du type de structure de stockage utilisé :

Ce qui est dans la deuxième colonne correspond au temps de recherche dans une liste chainée
Ce qui est dans la troisième colonne correspond au temps de recherche dans une table de hachage.

2510	0.000272.	0.000004
2520	0.000299	0.000004
2530	0.000476	0.000004

On en conclut que l'on obtient en moyenne un temps de recherche 74,75 fois plus long pour

une liste chainée que pour une table de hachage. (0.000299/0.000004=74,75).

Recherche dans le cas où le livre est présent

On réalise la recherche du livre suivant : « WLRBBMQBHCDARZOWK » qui est présent dans la bibliothèque.

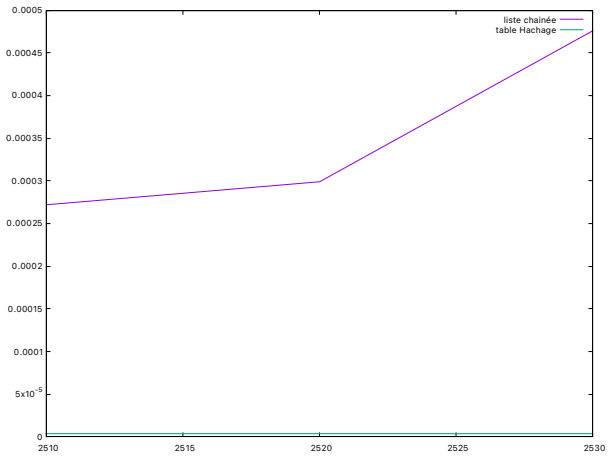
Voici un extrait des temps de recherches obtenu en fonction du type de structure de stockage utilisé :

Ce qui est dans la deuxième colonne correspond au temps de recherche dans une liste chainée
Ce qui est dans la troisième colonne correspond au temps de recherche dans une table de hachage.

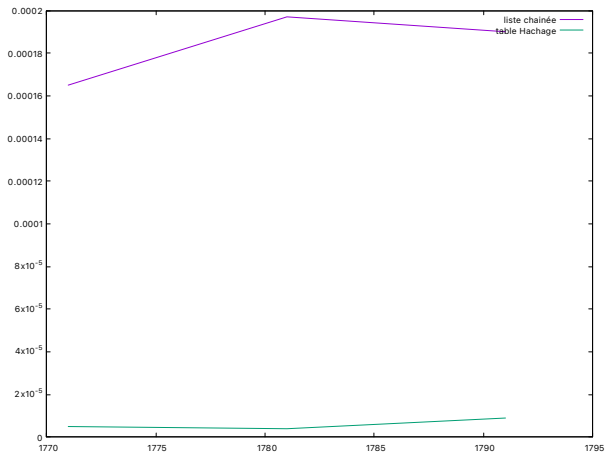
1771	0.000165	0.000005
1781	0.000197	0.000004
1791	0.000190	0.000009

On en conclut que dans le cas où le livre est présent on obtient en moyenne un temps de recherche 33 fois plus long pour une liste chainée que pour une table de hachage. (0.000165/0.000005=33).

En effectuant une recherche par titre ou par numéro on remarque que cette recherche est légèrement plus rapide en utilisant comme structure une liste chainée. Ceci est dû au fait que la table de hachage nécessite des opérations supplémentaires.

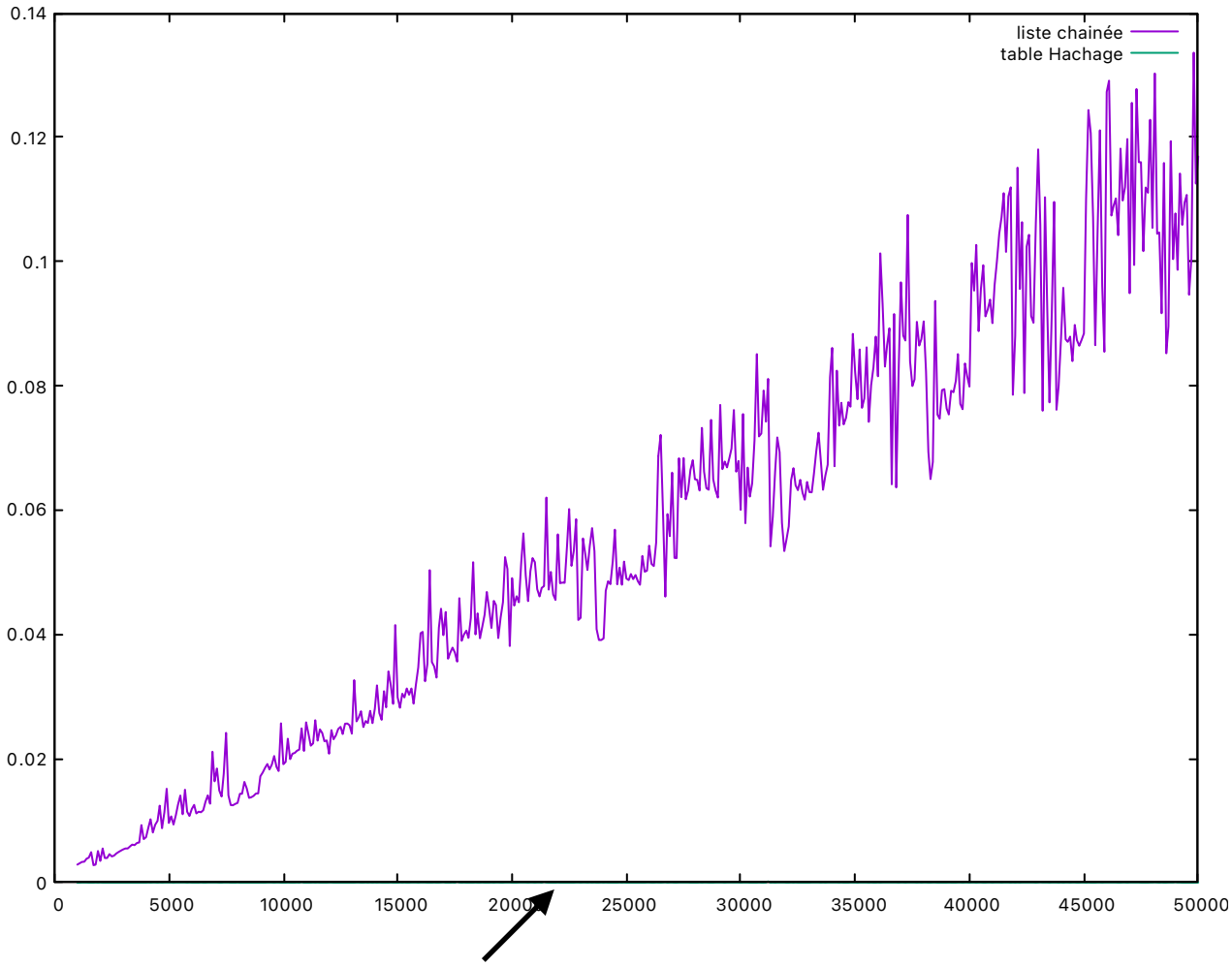


Comparaison entre la vitesse d'execution d'une recherche dans la bibliothèque lorsque le livre est absent



Comparaison entre la vitesse d'execution d'une recherche dans la bibliothèque lorsque le livre est present

Pour voir la différence d'efficacité entre une table de hachage et une liste chainée on compile nos nos deux programmes avec *make all* et *make all1* + on execute *.mainh >sortie_vitesse.txt* + on affiche nos courbes avec *gnuplot -p <commande.txt*



En vert tout en bas (la courbe de la table de hachage)

Comme en témoigne ces courbes, on remarque que le temps de recherche en fonction d'un auteur en utilisant une table de hachage est pratiquement nul par rapport à l'utilisation d'une liste chainée.

La table de hachage permet de disperser une liste d'entrées de manière homogène. Pour un taille m de la table et n éléments on aura m listes chainées de collisions contenant chacune en moyenne n/m elements.

Deux entrées de clef différente seront différentes. Donc pour trouver des doublons dans une table de hachage, il faut trouver des doublons dans chacune des listes chainées qui est de $O(n^2)$

On obtient donc une complexité de $O(m(n/m)^2)$ en moyenne. = $O(n^2/m)$ donc m fois plus rapide que une liste chainée.