LU2IN006
Rapport TME 2 – Groupe Mono 2

Binôme :

Paul Jarski (Num Etudiant : 28710621)
Margaux Marseloo (Num Etudiant : 28630611)

Description du sujet :

Le but de ce TME est dans un premier temps de modéliser une bibliothèque et sa gestion selon deux structures distinctes qui sont une liste chaînée et une table de hachage avec chaînage. Puis, dans un second temps de comparer l'efficacité en temps de chacune de ces structures pour effectuer les fonctions de recherche.

Description des structures manipulées :

On manipule principalement 2 structures qui nous servent à modéliser une bibliothèque composée de livres (un livre possède un numéro d'enregistrement, un titre et un auteur) :

- Les listes chaînées (LC) dont la structure est définie dans le fichier *biblioLC.h* :
  - listes chaînées de livres :

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

  - bibliothèque pointant vers le début d'une liste chaîné :

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

- Les tables de hachages (H) dont la structure est définie dans le fichier *biblioH.h* :
  - liste chaînée : la clef d'un livre sert à calculer sa position dans la table de hachage

```
typedef struct livreh {
    int clef;
    int num;
    char *titre;
    char *auteur;
    struct livreh *suiv;
} LivreH;
```

- table de hachage :

```c
typedef struct table {
    int nE; /*nombre d'elements contenus dans la table de hachage*/
    int m; /*taille de la table de hachage*/
    LivreH **T;/*table de hachage avec resolution des collisions par chainage*/
} BiblioH;
```

<u>Description global du code</u> :

Notre code se divise en deux parties, les fichiers LC qui manipulent des listes chaînées et les fichiers H qui manipulent des tables de hachage, et un main qui sert à faire les tests de performances. Les fichiers composant notre code sont les suivants :

- biblioLC.c
- biblioLC.h
- entreeSortieLC.c
- entreeSortieLC.h
- mainLC.c
- testsLC.c
- testVitesseLC.c

- biblioH.c
- biblioH.H
- entreeSortieH.c
- entreeSortieH.h
- mainH.c
- testsH.C
- testVitesseH.c

- Makefile

- testVitesse{H,LC}3_{1,2,3}.c pour les graphes

<u>Description des jeux d'essais</u> :

Pour tester nos fonctions, il y a un fichier pour chaque structure. Dans chacun des fichiers, des tests sont effectués pour vérifier individuellement le fonctionnement et la validité de chaque fonction :

- testsLC.c pour les listes chaînées
- testsH.c pour les tables de hachage avec chaînage

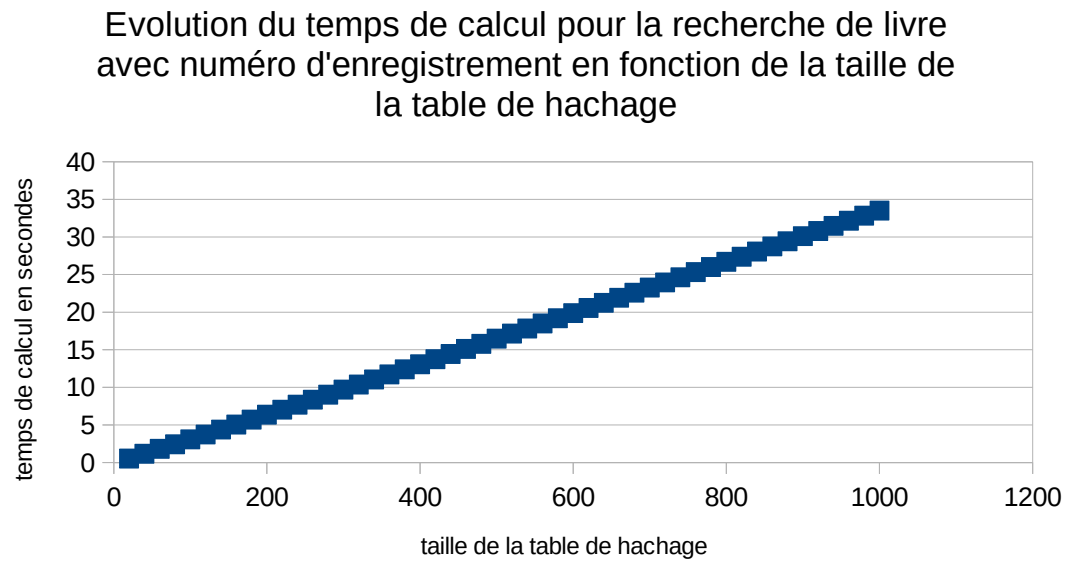
<u>Analyse des performances des programmes</u> :

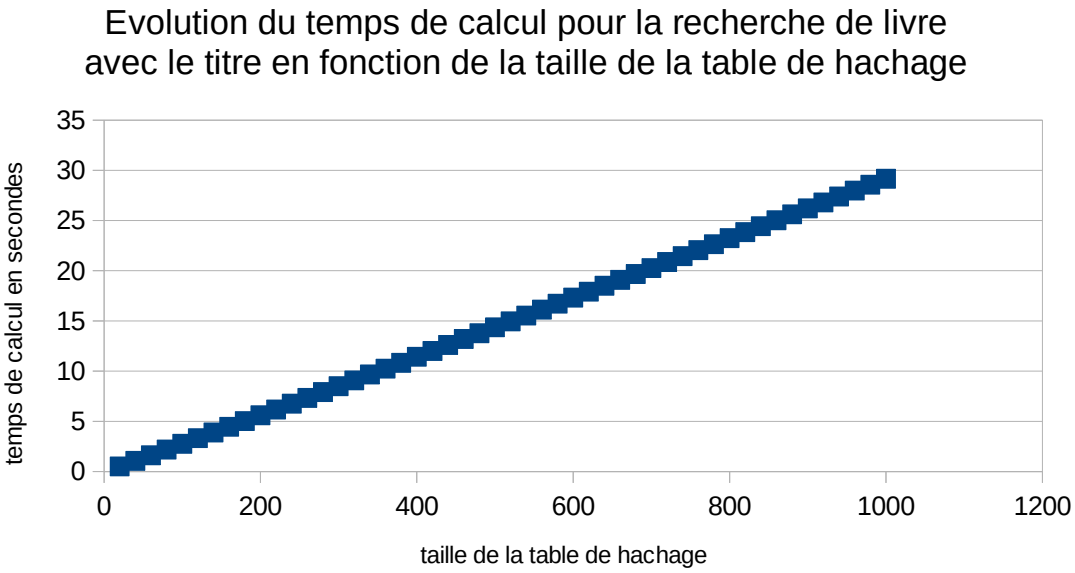
- **Tableau comparatif du temps de calcul (en secondes) pour les fonctions de recherche d’ouvrages :**
-

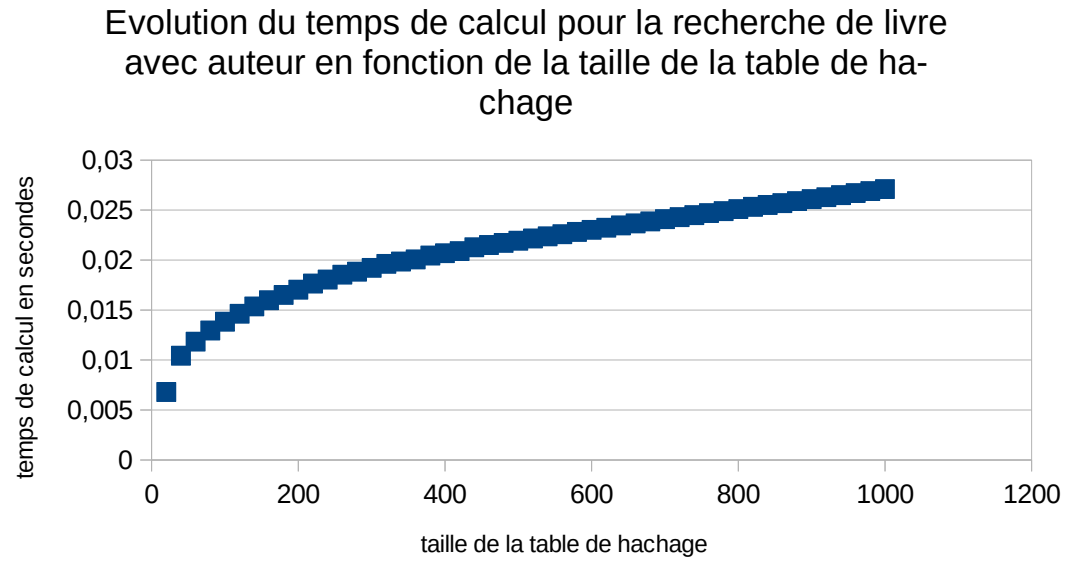
Le temps donné ici est une moyenne du temps pour 4 ouvrages recherchés 100 fois chacun pour la catégorie « Livre présent » et une moyenne du temps pour 3 ouvrages pour la catégorie « Livre absent ».

	Table de hachage avec chaînage de taille 500		Listes chaînées	
	Livre présent	Livre absent	Livre présent	Livre absent
Fonction de recherche d’un ouvrage par son numéro d’enregistrement	0.461083	0.756757	0.036869	0.07228
Fonction de recherche d’un ouvrage par son titre	0.438238	0.712567	0.038051	0.0759756
Fonction de recherche d’ouvrages par son auteur	0.000239	0.001358	0.073670	0.074569

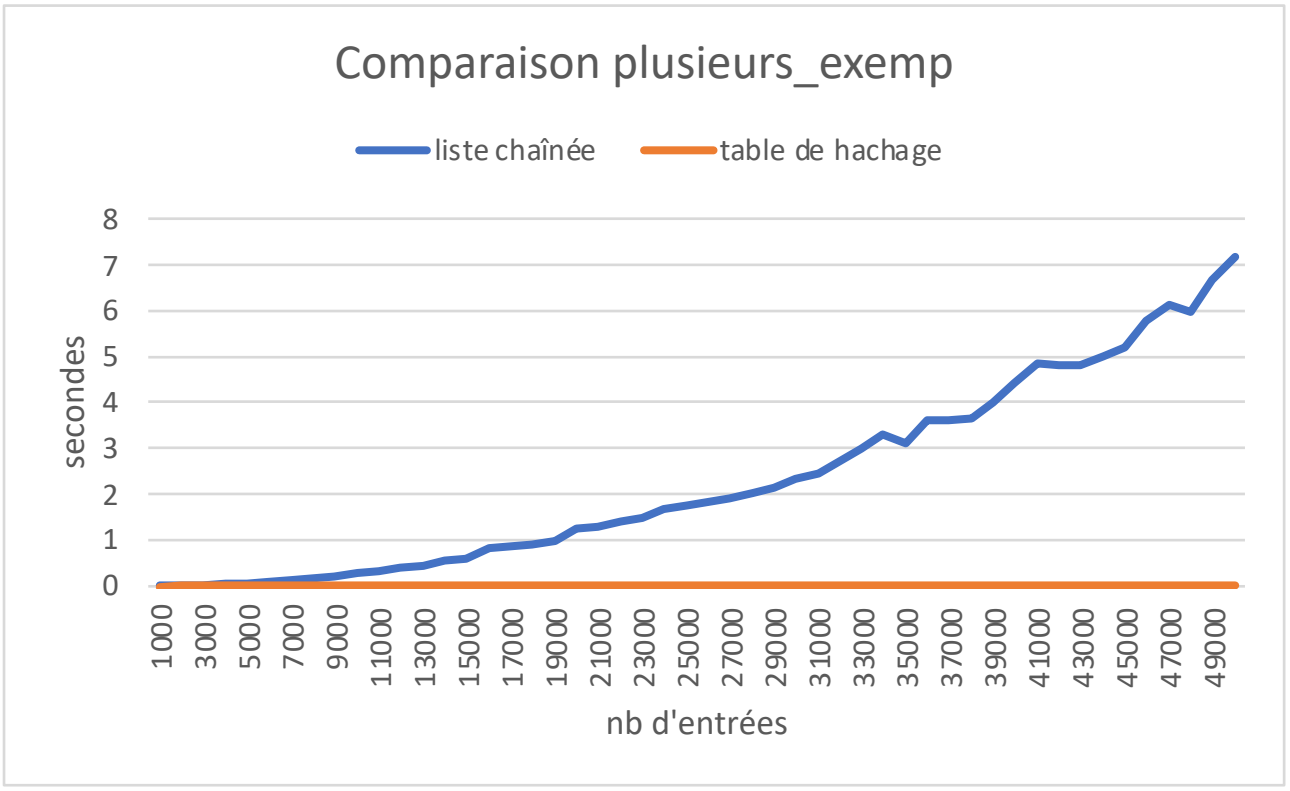
- **Temps de calcul en fonction de la taille de la table de hachage**







- **Graphe comparatif du temps de calcul de la recherche des ouvrages en plusieurs exemplaires en fonction de la taille de la bibliothèque.**



<u>Conclusion</u> :

Finalement, on remarque que le temps nécessaire aux listes chaînées restent à peu près le même peu importe la recherche contrairement aux tables de hachage qui sont en moyenne, d'après nos données, moins efficaces sauf lorsqu'il s'agit de la recherche à partir de l'auteur. Cette différence parait logique car la position dans la table est calculée à partir des auteurs des livres et donc on ne parcourt qu'une seule liste chaînée qui dans le pire des cas contient tous les éléments et donc est équivalente à la recherche dans une seule structure de liste chaînée. Néanmoins, selon la théorie, les tables de hachage devrait en moyenne être plus performante que des listes chaînées et moins performantes dans le pire des cas. **[Q3.1]** On se demande alors si le problème ne provient pas de la taille que l'on a choisi pour la table de hachage.

En analysant les courbes représentant le temps de calcul nécessaire pour faire une recherche en fonction de la taille de la table de hachage, nous observons que plus la table est grande, plus le temps de calcul est long. Donc effectivement nous aurions plutôt intérêt à l'exercice précédant de diminuer la taille de notre table de hachage.**[Q3.2]** Néanmoins, encore une fois, la performance ne devrait pas décroître lorsque la taille de la table grandit. Au contraire elle devrait augmentée car il y aurait moins de collisions et la recherche devrait approcher le O(1). Il y a ici une incohérence entre notre expérience et la théorie.

Sur le dernier graphe, on remarque que les tables de hachages sont cependant bien plus performantes que les listes chaînées lorsqu'il s'agit de la rechercher des ouvrages à plusieurs exemplaires. Cela est du au fait que les éléments sont répartis sur le tableau, et donc que les listes chaînées de chaque case sont de tailles réduites par rapport à une seule grande liste chaînée. Finalement, dans le pire des cas, c'est-à-dire si la fonction de hachage renvoyait toujours la même valeur, on aurait le même temps de recherche car la même structure à parcourir : une liste chaînée de taille n. **[Q3.4]** Cette observation est cohérente avec la théorie.