Mini Projet

Structures de Données

LU2IN006

Maëlle LIU 21204734 - Thibaut MARCQ 21202966

21 Février 2024

Sommaire

Explication de notre projet

Nature du projet

Le sujet de ce mini-projet visait à **modéliser une bibliothèque**. Une première partie était consacrée à la modélisation par une **liste chaînée**, une seconde partie était consacrée à la modélisation par une **table de hachage**.

Dans chaque partie des fonctionnalitées étaient demandées : **charger un nombre de livres** dans la bibliothèque à partir d'un fichier, **afficher** la bibliothèque, **ajouter/supprimer** un livre dans la bibliothèque, **rechercher** un livre par : son **numéro**, son **titre** ou son **auteur**, **fusionner** deux bibliothèques dans une seule, rechercher les livres en **plusieurs exemplaires** et **enregistrer** la bibliothèque dans un fichier.

Organisation du projet

Notre projet est structuré dans un seul dossier, dans le répertoire Projet3-4. Dans ce répertoire se trouve pour chaque partie, des fichiers .c et .h :

- les fichiers de la première partie sont identifiables par leur **suffixe** `LC`,
- ceux de la deuxième partie sont identifiables par leur **suffixe** `H`,
- ceux de la troisième partie ont un **préfixe** `ex3`.

Tous les fichiers sont compilables facilement grâce au Makefile. Celui-ci génère les fichiers .o ainsi que les exécutables `main`, `ex3-1` et `ex3-3`.

Le fichier `main` contient les main des deux premières parties, chacun distinguable lors du lancement du programme (option 1 ou 2). Les fichiers de l'exercice 3 correspondent à la question qui leur est associée.

Réponses aux questions de l'exercice 3

Question 1

Voici les résultats que l'on obtient à la suite des mesures sur nos fonctions :

```
Liste Chainee, Recherche par num (cas livre present): 0.000098s
Table Hachage, Recherche par num (cas livre present): 0.000193s
Liste Chainee, Recherche par num (cas livre absent): 0.000114s
Table Hachage, Recherche par num (cas livre absent): 0.000313s

Liste Chainee, Recherche par titre (cas livre present): 0.000176s
Table Hachage, Recherche par titre (cas livre present): 0.000203s
Liste Chainee, Recherche par titre (cas livre absent): 0.000163s
Table Hachage, Recherche par titre (cas livre absent): 0.000367s

Liste Chainee, Recherche par auteur (cas livre present): 0.000197s
Table Hachage, Recherche par auteur (cas livre present): 0.000002s
Liste Chainee, Recherche par auteur (cas livre absent): 0.000198s
Table Hachage, Recherche par auteur (cas livre absent): 0.000001s
```

Résultats obtenus sur une bibliothèque de taille **5000**

D'après ces résultats, on peut voir que la **liste chainée est la plus performante** pour les recherches par **numéro** et par **titre** : dans les deux cas considérés elle est plus rapide (recherche num présent: `0.000095s`, recherche num absent: `0.000199s`, recherche titre présent: `0.000027s`, recherche titre absent: `0.000204s`).

En revanche, pour la recherche par **auteur**, la **table de hachage est la plus performante** (recherche auteur présent: `0.000195s`, recherche auteur absent : `0.000197s`). La recherche se fait presque instantannément avec elle.

Question 2

Voici les résultats obtenus en augmentant la taille de la bibliothèque :

```
Liste Chainee, Recherche par num (cas livre present): 0.001247s
Table Hachage, Recherche par num (cas livre present): 0.003058s
Liste Chainee, Recherche par num (cas livre absent): 0.001046s
Table Hachage, Recherche par num (cas livre absent): 0.004173s

Liste Chainee, Recherche par titre (cas livre present): 0.001490s
Table Hachage, Recherche par titre (cas livre present): 0.001632s
Liste Chainee, Recherche par titre (cas livre absent): 0.001132s
Table Hachage, Recherche par titre (cas livre absent): 0.003309s

Liste Chainee, Recherche par auteur (cas livre present): 0.001350s
Table Hachage, Recherche par auteur (cas livre present): 0.000008s
Liste Chainee, Recherche par auteur (cas livre absent): 0.001228s
Table Hachage, Recherche par auteur (cas livre absent): 0.000008s
```

Résultats obtenus sur une bibliothèque de taille **40000**

Ces résultats confirment nos premières observations et les accentuent. La liste chainée est toujours **plus performante** sur la **recherche par numéro et par titre** mais ne l'est **pas lors de la recherche par auteur**.

En comparaison avec les premiers résultats obtenus sur une bibliothèque plus petite, le temps de calcul de la recherche par auteur n'a **pas augmenté dans les mêmes proportions** que celui des autres recherches (avec ou sans table de hachage). Il est resté **très faible**. Ceci est dû à la **clé** de notre Table de Hachage. Celle-ci est basée sur l'**auteur** des livres de notre bibliothèque. Ainsi, cela rend les opérations sur les auteurs très rapides.

Questions 3 et 4

Détermination du temps de recherche des **ouvrages en plusieurs exemplaires**

(Q. 3)
En redirigeant le code de `ex3-3.c` dans un fichier texte, on peut obtenir (dans GnuPlot) le graphique suivant:

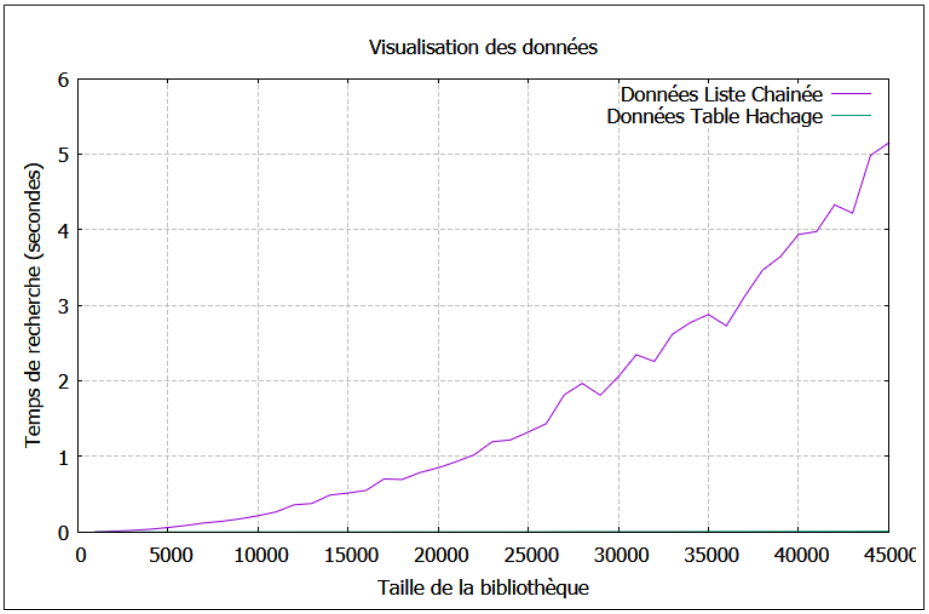

Figure 1: Evolution des **temps de recherche** des deux structures en fonction de la **taille de la bibliothèque** (pas de 1000)

D'après ce graphique, on peut voir que **plus la taille de la bibliothèque augmente**, plus le **temps de recherche** pour la fonction en Liste Chainée **augmente** aussi. On ne distingue que **très peu** l'évolution des données de la Table de Hachage.

Pour mieux visualiser l'évolution des **recherches par Table de Hachage** :

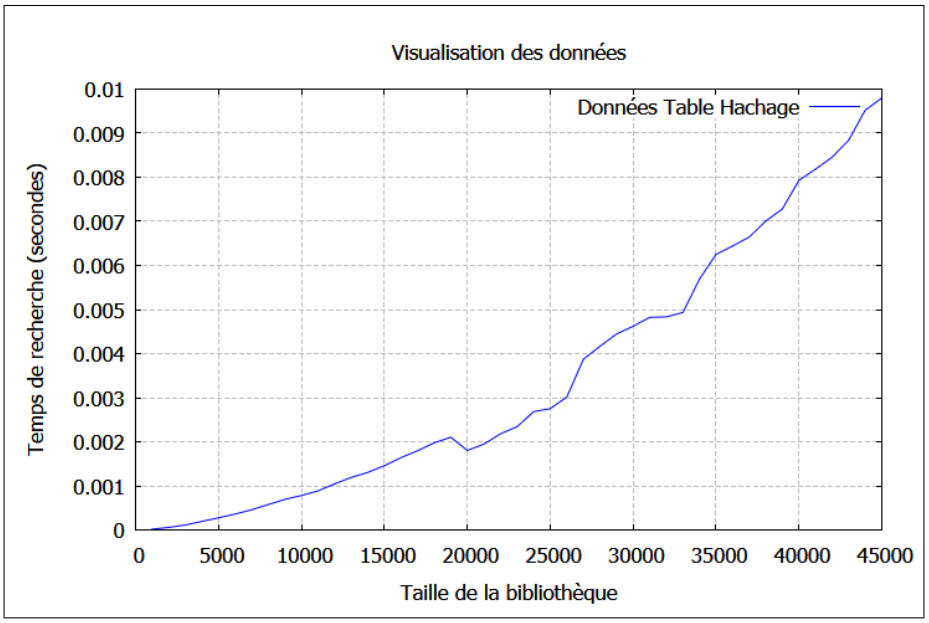

Figure 2: Evolution des **temps de recherche** de la **Table de Hachage** en fonction de la **taille de la bibliothèque** (pas de 1000)

Ce graphique permet de visualiser **plus précisément** l'évolution des temps de recherche des livres en plusieurs exemplaires par la table de hachage. On se rend compte d'une **augmentation très faible** du temps de recherche. Contrairement à celle sur la Liste Chainée, qui évoluait sur **[0, 6] secondes**, celle sur la Table de Hachage évolue seulement sur **[0, 0.1] secondes**.

(Q. 4)
Ceci est dû à la **complexité des deux fonctions** de recherche utilisées ici. La fonction en Liste Chainée a une **complexité temps pire-cas en** $O(n^2)$, tandis que celle en Table de Hachage a une complexité temps pire-cas **plus faible**. Pour chercher les livres en plusieurs exemplaires, elle s'intéresse en premier aux **livres qui ont le même auteur**. Comme nous l'avons vu à la question précédente, la **recherche par auteur** dans une Table de Hachage est très performante.