2I006
Mini-Projet : Gestion d'une bibliothèque

Ariana Carnielli et Lisa Kacel

Dans ce projet, on s'intéresse à la gestion d'une bibliothèque, c'est-à-dire un ensemble de livres, où chaque livre a un titre, un nom d'auteur et un numéro d'enregistrement unique. L'objectif est de comparer l'implémentation de cette bibliothèque avec deux structures distinctes : une liste chainée et une table de hachage avec résolution de collision par chainage.

L'implémentation de la liste chainée a été faite de façon standard et en suivant les consignes de l'énoncé, en implémentant quelques fonctions auxiliaires et en cherchant notamment à éviter les fuites de mémoire. Pour la table de hachage, on a modifié la méthode de calcul de la clef par rapport à la méthode proposée. En effet, la somme des codes ASCII d'une chaine de caractères de taille d'environ 10 ne peut donner que des nombres entre 0 et $127 \times 10 = 1270$, ce qui donne un nombre de clefs différentes possibles trop petit par rapport à la taille de la bibliothèque donnée. On a donc choisi de multiplier les codes ASCII, en prenant soin de déclarer toutes les variables nécessaires comme `unsigned int` afin d'éviter que des calculs d'indice donnent des résultats négatifs à cause du dépassement de capacité d'un `int`. Cette modification a permis de réduire le nombre de cellules vides de la table de hachage, ce qui améliore ses performances.

La suite de ce rapport présente la comparaison des deux structures selon l'Exercice 3.

Question 3.1

La Table 1 présente les temps de calcul en millisecondes pour réaliser la recherche d'un ouvrage par son numéro ou son titre et la recherche de tous les ouvrages d'un même auteur. Cette simulation a été faite en chargeant toute la bibliothèque du fichier donné (100 000 livres), et on a choisi la taille de la table de hachage comme la moitié de la quantité de livres dans la bibliothèque, soit $m = 50000$. À chaque recherche, nous avons cherché une donnée non présente dans la bibliothèque pour tester le pire des cas. Chaque recherche a été répétée 100 000 fois.

Comme on peut voir, la recherche par auteur avec la table de hachage se fait en 0 millisecondes. Même en augmentant le nombre de répétitions de la recherche à 100 millions, ce chiffre ne change pas. Ce comportement était attendu car nous avons choisi le nom d'auteur pour le calcul de la clef. Ainsi, il suffit de rechercher les ouvrages de cet auteur parmi ceux ayant la même clef, qui se trouvent dans une même liste chainée

	Numéro	Titre	Auteur
Liste chainée	0,86	0,85	0,84
Table de hachage	3,36	4,89	0,00

Table 1 – Temps moyen de recherche en millisecondes par type de structure de données

de la table de hachage. Cette liste chainée est assez petite (en moyenne 2 éléments), la complexité de cette recherche est donc en $\Theta(1)$.

Pour toutes les autres recherches, il faut parcourir tous les éléments dans le pire des cas, elles sont donc en $\Theta(n)$. On observe que, comme la liste chainée est une structure linéaire plus simple, le temps d'exécution est plus petit. Ainsi, la liste chainée est plus appropriée pour les recherches par numéro et titre et la table de hachage pour la recherche par auteur.

Question 3.2

L'évolution des temps d'exécution des recherches par numéro, titre et auteur sont représentées sur la Figure 1. Nous avons modifié m de 5 000 à 100 000 en pas de 5 000 et, pour chaque valeur de m, chaque recherche a été répétée 5 000 fois pour le calcul des temps moyens.

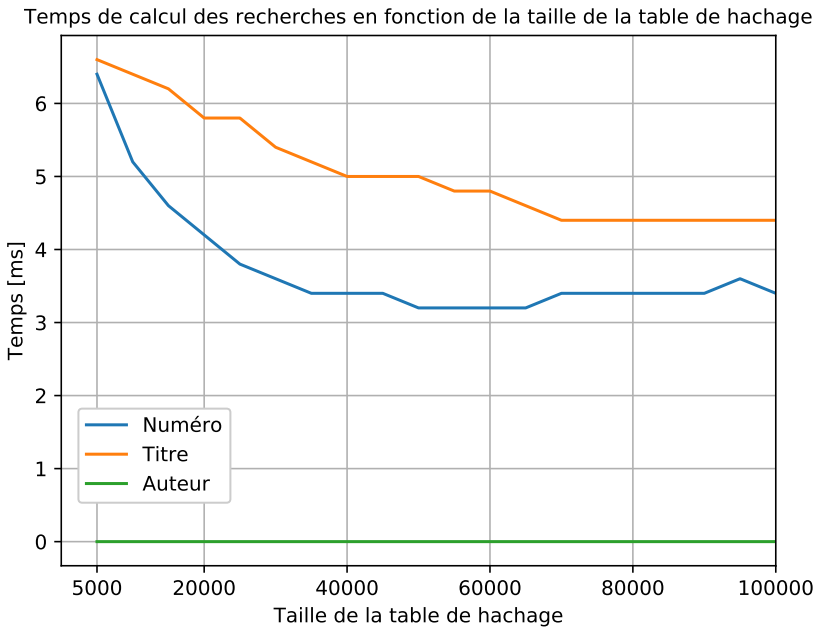
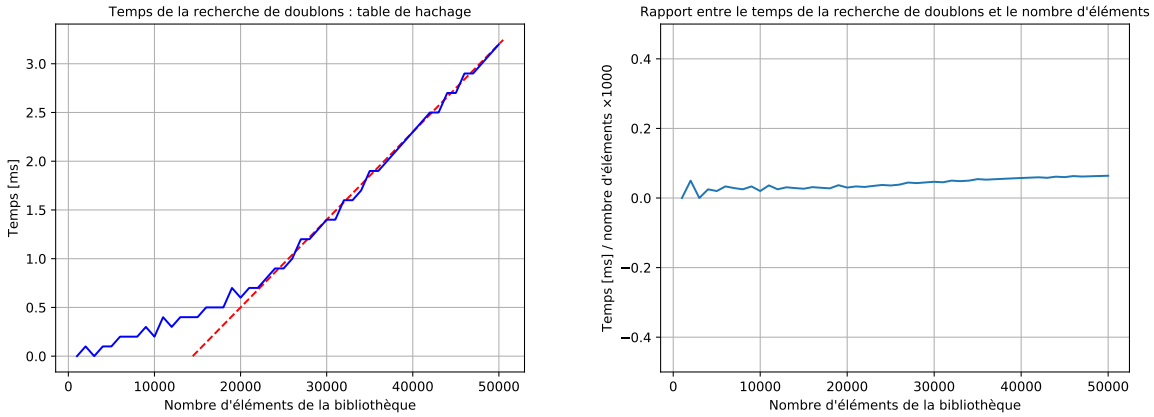


Figure 1 – Temps d'exécution des recherches

On remarque que, comme décrit dans la question précédente, le temps de recherche par auteur ne change pas, restant à zéro pour toutes les valeurs de m. Pour les autres recherches, le temps de calcul décroit lorsque m augmente, mais change très peu pour

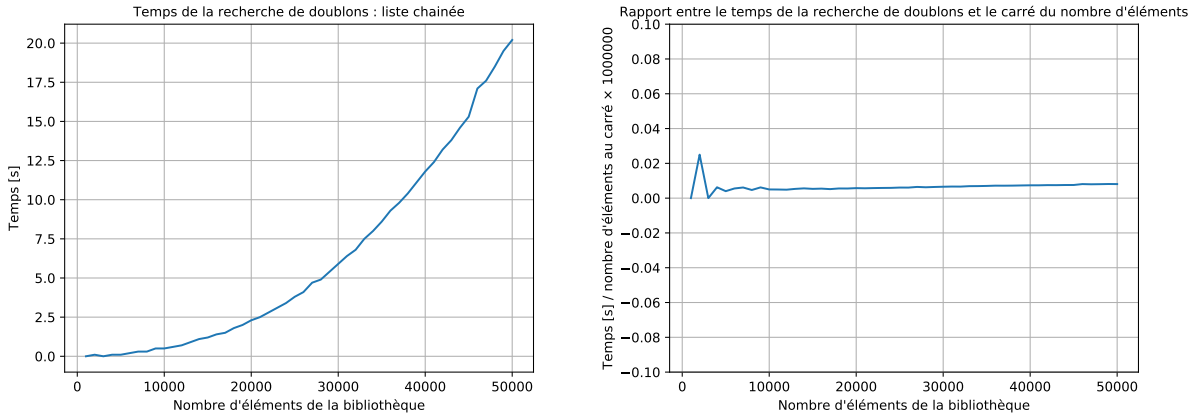
m supérieur à 50 000. Une explication possible serait que, avec $m = 50000$, on a en moyenne 2 éléments par entrée de la table de hachage, ce qui est déjà très petit, et donc on ne s'attend pas à d'importantes améliorations en augmentant le m. Cela justifie aussi notre choix de m à la question précédente.

Question 3.3



(a) (b)

FIGURE 2 – Temps de recherche de doublons pour la table de hachage



(a) (b)

FIGURE 3 – Temps de recherche de doublons pour la liste chainée

Les temps de recherche des ouvrages en doublons pour la table de hachage et la liste chainée sont donnés dans les Figures 2(a) et 3(a), respectivement. Dans le cas de la Figure 2, chaque temps a été calculé comme une moyenne de 10 000 simulations, alors que, pour la Figure 3, on a fait une moyenne de 10 simulations seulement. Nous attirons l'attention au fait que l'échelle verticale n'est pas la même pour les deux figures.

On remarque que, pour la table de hachage, à part pour quelques valeurs petites de nombre d'éléments, le comportement du temps de calcul est essentiellement linéaire. Pour mettre en évidence ce comportement, nous avons tracé, dans la Figure 2(a), la droite qui coïncide avec les données expérimentales pour $n = 30000$ et $n = 50000$, et l'on observe que le comportement expérimental est très proche de cette droite pour $n \geq 20000$. Nous avons aussi tracé, dans la Figure 2(b), le temps de calcul divisé par le nombre d'éléments, afin de vérifier que cette quantité est proche d'une constante.

Pour la liste chainée, le comportement du temps de calcul est quadratique, ce qui a été mis en évidence par la Figure 3(b), où nous avons tracé le temps de calcul divisé par le carré du nombre d'éléments, afin de vérifier que cette quantité est aussi proche d'une constante.

Question 3.4

Dans le cas de la recherche de doublons avec une liste chainée, pour chaque élément de la liste, la fonction parcourt toute la liste une fois dans le pire des cas, où elle ne trouve pas de doublon pour cet élément. Ainsi, comme il faut répéter cette boucle pour chaque élément de la liste, la complexité est en $\Theta(n^2)$ au pire cas, où il n'y a pas de doublons dans la liste.

Dans le cas de la recherche de doublons avec une table de hachage de taille m à n éléments, on sait que tous les doublons seront forcément dans la même liste chainée, car ils auront la même clef. Alors la complexité est en $\Theta\left(m\left(\frac{n}{m}\right)^2\right)$, car, pour chacune des m listes chainées, pour chacun de ses éléments, on parcourt toute cette liste à la recherche de ses doublons, et la taille moyenne d'une liste chainée est $\frac{n}{m}$.

Dans la Question 3.3, nous avons toujours choisi $m = \frac{n}{2}$, ainsi la complexité avec une table de hachage est en $\Theta\left(\frac{n}{2}\left(\frac{n}{n/2}\right)^2\right) = \Theta(2n) = \Theta(n)$.