Zhang Yuxiang & Lecomte Antoine

Mini-Projet 2 : Gestion d'une bibliothèque

Nous avons travaillé dans ce mini-projet avec des structures en langage C permettant de gérer des bibliothèques composées de livres. Dans l'ensemble des exercices, un livre est décrit et identifié à l'aide de son titre, son auteur et un numéro unique.

Dans la première partie de ce travail, nous utilisons une liste simplement chaînée pour la gestion de bibliothèques. La structure Livre est donc composée des trois éléments mentionnés précédemment, ainsi que d'un pointeur suiv sur la structure, afin de ranger les livres dans une liste chaînée. La deuxième structure utilisée, Biblio, correspond donc à une bibliothèque, elle est donc simplement composée de livres.

Pour organiser notre code correspondant à cette partie, nous avons donc eu recourt à deux fichiers ".c" et deux ".h". Le fichier header biblioLC.h contient nos structures que nous avons définies ainsi que les entêtes des fonctions principales pour la création/suppression des livres/bibliothèques. Nous incluons ce fichier dans le deuxième header (entreeSortieLC.h), où nous avons stocké le reste des fonctions utilisées pour ce mini-projet. Les fichiers ".c" respectifs contiennent les codes des fonctions que nous manipulons.

Dans la deuxième partie du travail, nous utilisons cette fois une table de hachage pour gérer nos bibliothèques. La structure LivreH est la même que la structure Livre avec un attribut clef en plus (Remarque : nous avons remplacé le pointeur suiv par suivant). La deuxième structure, BiblioH, est désormais gérée avec une table de hachage, elle a une taille m, un nombre d'éléments nE et un double pointeur de LivreH T. Ce dernier correspond à la table de hachage (2 dimensions) où sont stockés les livres. L'organisation pour cette partie est similaire à la première (deux fichiers ".c" et deux ".h", l'un de ces fichiers header inclut l'autre).

Le neuvième fichier (main.c) est le fichier principal, on y utilise à l'intérieur tout ce qu'on a défini au préalable dans les autres fichiers. Il contient un menu (fonction d'affichage pour informer sur les actions possibles à réaliser sur les bibliothèques), une fonction pour évaluer les performances en terme de temps d'exécution des programmes, afin de comparer en fonction des cas (recherches de livres avec numéro, titre, auteur, livres en multiples exemplaires) quelle est la méthode la plus efficace (utilisation de listes simplement chaînées ou de tables de hachage) et une fonction main, qui contient notamment l'ensemble des instructions à effectuer en fonction des entrées de l'utilisateur pendant l'exécution du programme. L'utilisateur peut effectuer 16 actions pour la gestion des livres et bibliothèques et lorsque l'utilisateur a terminé, il peut quitter le programme lorsque le programme demande le choix de l'action à réaliser en entrant la commande "0".

Pour des questions évidentes en termes d'optimisation, nous avons également créé un Makefile puisque nous travaillons avec plusieurs fichiers, et qu'il est essentiel de pouvoir gérer efficacement la compilation de tous ces fichiers en même temps, ce qui permet un gain de temps considérable et renforce la propreté du code.

Résultats obtenus sur les temps d'exécutions :

Taille de la table de hachage : 100050

Recherche d'un ouvrage par son numéro :

Livre présent :

Recherche par numéro (liste chaînée):
Entrez le numéro de l'ouvrage : 1
Ouvrage trouvé : Numéro: 1, Titre: SCDXRJ, Auteur: owfrx
numéro (LC) : Temps de recherche : 0.005979 secondes

Recherche par numéro (table de hachage):
Entrez le numéro de l'ouvrage : 1
Ouvrage trouvé : Clé: 566, Numéro: 1, Titre: SCDXRJ, Auteur: owfrx
numéro (H) : Temps de recherche : 0.013116 secondes

Livre absent :

Recherche par numéro (liste chaînée):
Entrez le numéro de l'ouvrage : 1000001
Aucun ouvrage trouvé avec ce numéro.
numéro (LC) : Temps de recherche : 0.005751 secondes

Recherche par numéro (table de hachage):
Entrez le numéro de l'ouvrage : 1000001
Aucun ouvrage trouvé avec ce numéro.
numéro (H) : Temps de recherche : 0.016669 secondes

Conclusion :
La complexité de rechercher_par_numero est O(n) mais la complexité de rechercher_par_numeroH est O(m*n), donc dans ce cas la liste chaînée est plus efficace.

Recherche d'un ouvrage par son titre :

Livre présent :

Recherche par titre (liste chaînée):
Entrez le titre de l'ouvrage : UQMNAC
Ouvrage trouvé : Numéro: 49999, Titre: UQMNAC, Auteur: mzeita
titre (LC) : Temps de recherche : 0.004276 secondes

Recherche par titre (table de hachage):
Entrez le titre de l'ouvrage : UQMNAC
Ouvrage trouvé : Clé: 650, Numéro: 49999, Titre: UQMNAC, Auteur: mzeita
titre (H) : Temps de recherche : 0.015028 secondes

Livre absent :

Recherche par titre (liste chaînée):
Entrez le titre de l'ouvrage : Hello
Aucun ouvrage trouvé avec ce titre.
titre (LC) : Temps de recherche : 0.004966 secondes

Recherche par titre (table de hachage):
Entrez le titre de l'ouvrage : Hello
Aucun ouvrage trouvé avec ce titre.
titre (H) : Temps de recherche : 0.018572 secondes

Conclusion :
La complexité de rechercher_par_titre est O(n) mais la complexité de rechercher_par_titreH est O(m*n), donc dans ce cas la liste chainée reste la plus efficace.

Recherche d'un ouvrage par son auteur :

Livre présent :

Recherche par l'auteur (liste chaînée):
Entrez l'auteur de l'ouvrage : sdwjm
Ouvrages trouvés de l'auteur sdwjm :
Bibliothèque :
Numéro: 50000, Titre: BZKRVBWCL, Auteur: sdwjm
auteur (LC) : Temps de recherche : 0.005242 secondes

Recherche par l'auteur (table de hachage):
Entrez l'auteur de l'ouvrage : sdwjm
Ouvrages trouvés de l'auteur sdwjm :
Bibliothèque :
Clé: 549, Numéro: 549, Titre: BZKRVBWCL, Auteur: sdwjm
auteur (H) : Temps de recherche : 0.001828 secondes

Livre absent :

Recherche par titre (liste chaînée):
Entrez le titre de l'ouvrage : Yuxiang
Aucun ouvrage trouvé avec ce titre.
titre (LC) : Temps de recherche : 0.004908 secondes

Recherche par l'auteur (table de hachage):
Entrez l'auteur de l'ouvrage : Yuxiang
Ouvrages trouvés de l'auteur Yuxiang :
Bibliothèque :
auteur (H) : Temps de recherche : 0.001624 secondes

Conclusion :
La complexité de la fonction rechercher_par_auteur est O(n) et la complexité de la fonction rechercher_par_auteurH est O(max(m, n)), où m est la taille de la table de hachage et n est la longueur moyenne de la chaîne.
Donc ici l'utilisation d'une table de hachage pour la recherche d'ouvrages par auteur est généralement plus efficace que l'utilisation d'une liste chaînée. Cela est dû à la complexité temporelle inférieure de la recherche par auteur dans une table de hachage par rapport à une liste chaînée.

Taille de la table de hachage : 50000

Recherche d'un ouvrage par son numéro :

Livre présent :

Recherche par numéro (liste chaînée):
Entrez le numéro de l'ouvrage : 50000
Ouvrage trouvé : Numéro: 50000, Titre: BZKRVBWCL, Auteur: sdwjm
numéro (LC) : Temps de recherche : 0.001389 secondes

Recherche par numéro (table de hachage):
Entrez le numéro de l'ouvrage : 50000
Aucun ouvrage trouvé avec ce numéro.
numéro (H) : Temps de recherche : 0.007908 secondes

Livre absent :

Recherche par numéro (liste chaînée):
Entrez le numéro de l'ouvrage : 200000
Aucun ouvrage trouvé avec ce numéro.
numéro (LC) : Temps de recherche : 0.004252 secondes

Recherche par numéro (table de hachage):
Entrez le numéro de l'ouvrage : 200000
Aucun ouvrage trouvé avec ce numéro.
numéro (H) : Temps de recherche : 0.007562 secondes

Recherche d'un ouvrage par son titre :

Livre présent :

Recherche par titre (liste chaînée):
Entrez le titre de l'ouvrage : UQMNAC
Ouvrage trouvé : Numéro: 49999, Titre: UQMNAC, Auteur: mzeita
titre (LC) : Temps de recherche : 0.001523 secondes

Recherche par titre (table de hachage):
Entrez le titre de l'ouvrage : UQMNAC
Ouvrage trouvé : Clé: 650, Numéro: 49999, Titre: UQMNAC, Auteur: mzeita
titre (H) : Temps de recherche : 0.005931 secondes

Livre absent :

Recherche par titre (liste chaînée):
Entrez le titre de l'ouvrage : Hello
Aucun ouvrage trouvé avec ce titre.
titre (LC) : Temps de recherche : 0.003847 secondes

Recherche par titre (table de hachage):
Entrez le titre de l'ouvrage : Hello
Aucun ouvrage trouvé avec ce titre.
titre (H) : Temps de recherche : 0.008429 secondes

Recherche d'un ouvrage par son auteur :

Livre présent :

Recherche par l'auteur (liste chaînée):
Entrez l'auteur de l'ouvrage : dygb
Ouvrages trouvés de l'auteur dygb :
Bibliothèque :
Numéro: 49978, Titre: ZVFSYFRHMM, Auteur: dygb
auteur (LC) : Temps de recherche : 0.003902 secondes

Recherche par l'auteur (table de hachage):
Entrez l'auteur de l'ouvrage : dygb
Ouvrages trouvés de l'auteur dygb :
Bibliothèque :
Clé: 422, Numéro: 422, Titre: ZVFSYFRHMM, Auteur: dygb
auteur (H) : Temps de recherche : 0.001104 secondes

Livre absent :

Recherche par l'auteur (liste chaînée):
Entrez l'auteur de l'ouvrage : Yuxiang
Ouvrages trouvés de l'auteur Yuxiang :
Bibliothèque :
auteur (LC) : Temps de recherche : 0.003965 secondes

Recherche par l'auteur (table de hachage):
Entrez l'auteur de l'ouvrage : Yuxiang
Ouvrages trouvés de l'auteur Yuxiang :
Bibliothèque :
auteur (H) : Temps de recherche : 0.000811 secondes

Conclusion générale (temps d'exécutions) :

Dans l'ensemble, nous pouvons conclure que la méthode de recherche varie en efficacité en fonction du type de recherche effectuée et de la taille de la table de hachage. Pour les recherches par numéro ou par titre, la liste chaînée semble être plus efficace car on obtient des temps de recherche plus rapides. Egalement, lorsque la taille de la table de hachage est petite, la méthode utilisant la table de hachage semble être moins efficace que celle qui utilise la liste chaînée.

En revanche, pour les recherches par auteur, la méthode de la table de hachage est généralement préférable, et bien que la différence de performance est peu significative pour les petites tailles de table de hachage, elle l'est davantage pour des tables plus grandes. Enfin, lorsque le livre n'est pas trouvé, la méthode utilisant la table de hachage semble offrir de meilleures performances si la recherche était effectuée par auteur, sinon c'est la méthode avec les listes chaînées qui prend le dessus. Les performances de recherche sont assez stables en faisant varier le nombre de répétitions pour le calcul du temps.

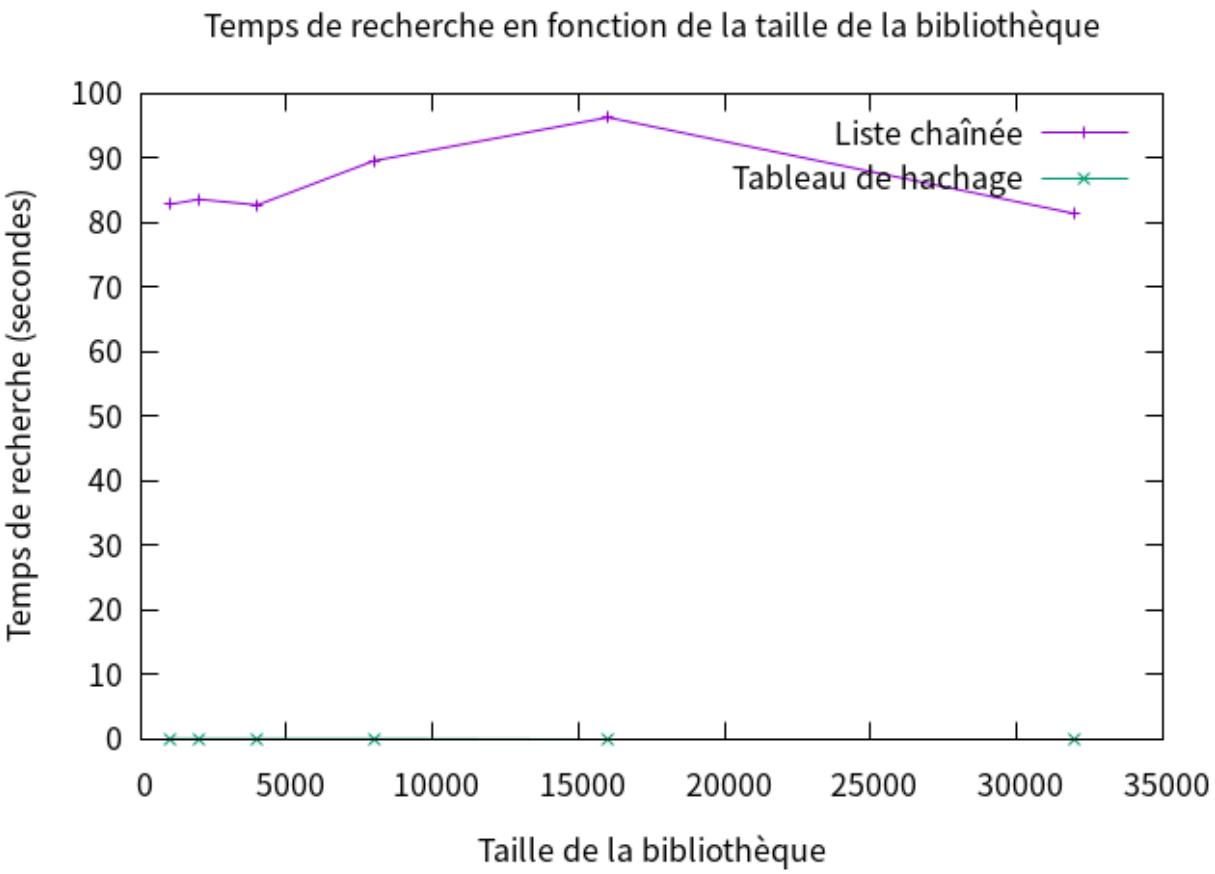
Que faire ? Le choix entre une liste chaînée et une table de hachage se porte alors sur nos attentes sur l'application : la taille de la bibliothèque, la fréquence des opérations de recherche et d'insertion, ainsi que les contraintes de mémoire. Gardons en tête que si la mémoire est à prendre en compte car on veut limiter au maximum l'espace disque occupé, si la quantité de mémoire disponible dans l'ordinateur est pauvre, ou si la taille de la bibliothèque est relativement petite, l'utilisation d'une liste chaînée sera plus appropriée.
Sans contrainte particulière, opter plutôt pour les fonctions de hachage sera avantageux.

Pour déterminer les temps de recherche des livres en plusieurs exemplaires, on a ajouté la fonction mesurer_temps_recherche_multiples dans le fichier main.c, puis nous avons fait une boucle afin de lire les n premières lignes du fichier allant de 1000 à 50000, n augmentant de manière croissante : on double sa valeur à chaque itération, et on a obtenu les résultats suivants :

Taille de la bibliothèque : 1000
Temps de recherche (Liste chaînée) : 82.785562 secondes
Temps de recherche (Table de hachage) : 0.020616 secondes
Taille de la bibliothèque : 2000
Temps de recherche (Liste chaînée) : 83.563714 secondes
Temps de recherche (Table de hachage) : 0.023964 secondes
Taille de la bibliothèque : 4000
Temps de recherche (Liste chaînée) : 82.676732 secondes
Temps de recherche (Table de hachage) : 0.020151 secondes
Taille de la bibliothèque : 8000
Temps de recherche (Liste chaînée) : 89.536116 secondes
Temps de recherche (Table de hachage) : 0.025495 secondes
Taille de la bibliothèque : 16000
Temps de recherche (Liste chaînée) : 96.236846 secondes
Temps de recherche (Table de hachage) : 0.020484 secondes
Taille de la bibliothèque : 32000
Temps de recherche (Liste chaînée) : 81.326962 secondes
Temps de recherche (Table de hachage) : 0.020547 secondes

Les résultats montrent que le temps de recherche avec une liste chaînée augmente considérablement avec la taille de la bibliothèque, tandis que le temps de recherche avec une table de hachage reste relativement constant, ce qui met en évidence cet avantage en termes d'efficacité et cela confirme une nouvelle fois l'avantage d'utiliser une table de hachage pour les opérations de recherche lorsque la taille des données devient importante, car on observe peu de différences malgré que l'on demande d'effectuer plus de recherches.

A l'aide de l'outil Gnuplot, on a représenté nos résultats graphiquement.



Nous pouvons voir que quand nous voulons faire une recherche d'exemplaires de livres multiples dans une bibliothèque, c'est l'utilisation de la table de hachage qui est plus efficace (approximativement 0 secondes quelle que soit la taille de la bibliothèque), mais en utilisant une liste chaînée, le temps d'exécution est toujours au-delà de 80 secondes.
Ainsi, les courbes obtenues illustrent clairement les différences de performances entre une liste chaînée et une table de hachage en fonction de la taille de la bibliothèque. Voici une justification basée sur la complexité pire cas attendue pour chaque structure de données :

Liste chaînée :

- La recherche dans une liste chaînée se fait en parcourant séquentiellement les éléments jusqu'à trouver celui recherché.
- Dans le pire des cas, lorsque l'élément recherché est le dernier de la liste ou n'existe pas du tout, on doit parcourir toute la liste.
- La complexité de la recherche dans une liste chaînée est donc linéaire par rapport à la taille de la liste, soit O(n), où n est le nombre d'éléments dans la liste.
- Comme illustré par les résultats, le temps de recherche avec une liste chaînée augmente linéairement avec la taille de la bibliothèque. Nous remarquons néanmoins un problème au-delà d'une bibliothèque de taille de 15000, les résultats sont inattendus mais nous pensons qu'il s'agit d'une défaillance des performances de l'ordinateur, car nous savons qu'une machine ne peut avoir des performances absolument constantes au fil du temps, surtout lorsqu'elle est mise à rude épreuve.

Table de hachage :

- La recherche dans un tableau de hachage se fait en calculant l'index de l'élément recherché, puis en accédant directement à cet index dans le tableau.
- Dans le cas idéal, où il n'y a pas de collisions, la recherche est constante, soit la complexité est O(1).
- Même dans des cas plus globaux avec des collisions, la complexité moyenne de la recherche dans un tableau de hachage reste constante dans la plupart des situations pratiques.
- Comme illustré par les résultats, le temps de recherche avec une table de hachage reste relativement constant même lorsque la taille de la bibliothèque augmente.