Yanis SAADI DIT SAADA
Ines HARRAOUI
Groupe 3

Compte-rendu

(mini-projet de SDD)

Introduction

Le sujet consiste à manipuler deux structures de données pour gérer une bibliothèque de livres.La première structure utilise des listes chaînées(LC dans le reste du sujet) (biblioLC.c et biblioLC.h), tandis que la deuxième utilise une table de hachage(H dans le reste du sujet) (biblioH.c et biblioH.h). Ces structures permettent d'effectuer des opérations telles que l'insertion, la recherche, la suppression, l'affichage, la fusion de bibliothèques et la recherche d'exemplaires.

Le programme principal « main.c » offre un menu interactif permettant à l'utilisateur de choisir entre ces deux structures, puis d'effectuer diverses opérations sur une bibliothèque en fonction du choix de la structure. En outre, il existe des fichiers pour la gestion des entrées/sorties pour chaque structure (entreeSortieLC.c, entreeSortieLC.c, entreeSortieH.c et entreeSortieH.h)

Descriptions globale du code

Description des structures :

- Avec liste chaînée :
    - Livre : représente un livre avec des attributs num, titre et auteur ainsi qu'un pointeur vers le livre suivant.
    - Biblio : représente une liste chaînée de livres, où chaque élément est un Livre. Cette structure est utilisée notamment pour l'insertion en tête, la suppression et la recherche.
- Avec table de hachage :
    - LivreH : représente un livre avec des attributs num, titre, auteur, une clé de hachage ainsi qu'un pointeur vers le livre suivant.
    - BiblioH : représente une table de hachage de livres, où chaque élément est une liste chaînée de LivreH ayant la même clé de hachage. Cette structure est utilisée notamment pour l'insertion en tête, la suppression et la recherche.

Description des fichiers .h :

- biblioLC.h : déclarent les structures (Livre et Biblio) et les signatures des fonctions associées aux listes chaînées
- biblioH.h : déclarent les structures (LivreH et BiblioH) et les signatures des fonctions associées aux tables de hachage
- entreeSortieLC.h/entreeSortieH.h : déclarent les fonctions pour la gestion des entrées/sorties de chacune des deux structures

Description des fichiers .c :

Les fichiers BiblioLC et BiblioH contiennent tous deux les opérations de création, d'affichage, de recherche, d'insertion et de suppression de leurs structures respectives.

entreeSortieLC.c et entreeSortieH.c contiennent les fonctions de lecture et d'écriture permettant de collecter les données de leurs structures respectives, dans des fichiers.

main.c contient la fonction «menu()» permettant à l'utilisateur d'interagir avec le programme en choisissant entre les deux structures, puis en effectuant diverses opérations comme l'insertion, la recherche, la suppression, la fusion et la recherche de doublons.

tests_LC.c et tests_H .c contiennent les tests de chaque fonction contenue dans les fichiers respectifs biblioLC.c, entreeSortieLC.c et biblioH.c, entreeSortieH.c.
Ils commencent par créer deux bibliothèques dans lesquelles sont insérés des livres, recherchent certains livres à partir de leur numéro, d'autres livres par leur titre puis fusionnent les deux bibliothèques. La bibliothèque résultante est enregistrée dans un fichier puis rechargée de toutes ses entrées. Enfin, la fonction rechargée effectue une recherche d'exemplaires stockée, puis supprime un livre spécifique et libère la mémoire de chacune des bibliothèques.

Description des fonctions principales :

- Dans biblioLC.c :
  - creer_livre_LC : crée un livre
  - liberer_livre_LC : libère la mémoire d'un livre
  - creer_biblio_LC : crée une bibliothèque Biblio
  - liberer_biblio_LC : libère la mémoire d'une bibliothèque
  - inserer_en_tete : insère un livre en tête d'une bibliothèque
  - supprimerLivre_LC : supprime un livre d'une bibliothèque
  - rechercheLivreNum_LC : recherche un livre à partir de son numéro num
  - rechercheLivreTitre_LC : recherche un livre à partir de son titre
  - LivresAuteurs_LC : retourne une nouvelle bibliothèque contenant les livres d'un auteur donné
  - Fusion_LC : fusionne deux bibliothèques Biblio

- Dans biblioH.c :
  - fonctionClef : calcule la clé de hachage à partir du nom d'un auteur

- fonctionHachage : calcule l'indice de la table de hachage à partir d'une clé
- creer_livre_H : crée un livre LivreH
- liberer_livre_H : libère la mémoire d'un livre
- creer_biblio_H : crée une bibliothèque BiblioH
- liberer_biblio_H : libère la mémoire d'une bibliothèque
- inserer: insère un livre dans la table de hachage d'une bibliothèque
- supprimerLivre_H : supprime un livre de la table de hachage d'une bibliothèque
- rechercheLivreNum_H : recherche un livre dans la table de hachage d'une bibliothèque, à partir de son numéro num
- rechercheLivreTitre_H : recherche un livre dans la table de hachage d'une bibliothèque, à partir de son titre
- LivresAuteurs_H : retourne une nouvelle bibliothèque contenant les livres d'un auteur donné
- fusion_H : fusionne deux bibliothèques BiblioH

- Dans entreeSortieLC.c :
  - gère les entrées/sorties des bibliothèques en listes chaînées
- Dans entreeSortieH.c :
  - gère les entrées/sorties des bibliothèques avec table de hachage
- Dans main.c : permet à l'utilisateur de manipuler les structures avec listes chaînées ou avec table de hachage et d'effectuer diverses opérations sur la bibliothèque en fonction du choix de la structure
- Dans tests_LC.c : effectue des tests pour chaque fonction contenues dans les fichiers biblioLC.c et entreeSortieLC.c
- Dans tests_H.c : effectue des tests pour chaque fonction contenues dans les fichiers biblioH.c et entreeSortieH.

Description des algorithmes

Algorithmes pour les structures en listes chaînées :

- inserer_en_tete_LC :
  1. Crée un nouveau livre avec les données fournies.
  2. Déplace le pointeur suivant le nouveau livre vers la tête de la bibliothèque
  3. Déplace la tête de la bibliothèque sur le nouveau livre à insérer
  4. Sinon, le nouveau livre est le seul livre de la bibliothèque.

- rechercheLivreNum_LC/rechercheLivreTitre_LC:
  1. Parcourt la bibliothèque
     a. Vérifie si le numéro de chaque livre égale le numéro/titre donné
        i. Retourne le livre si le numéro/titre est trouvé
     b. déplace la tête de la bibliothèque sur le livre suivant

  2. Retourne NULL si aucun livre ne contient le numéro/titre donné

-LivresAuteurs_LC:

1. Crée une nouvelle bibliothèque
2. Parcourt la bibliothèque donnée
    a. Si le livre actuel est écrit par l'auteur donné
        i. Insère en tête tous les livres écrits par l'auteur donné
    b. Déplace la tête de la bibliothèque sur le livre suivant
3. Retourne la nouvelle bibliothèque contenant les livres écrits par l'auteur donné

- supprimerLivre_LC:

1. Parcourt la bibliothèque pour trouver le livre à supprimer en fonction de ses attributs
2. Si le livre est trouvé:
    a. Si le pointeur du livre précédent est NULL, le livre est en tête de liste:
        i. Déplace la tête de la bibliothèque sur le livre qui suit le livre à supprimer
    b. Sinon, le livre à supprimer est au milieu de la liste:
        i. Crée un pointeur "tmp" pointant sur le livre à supprimer
        ii. Libère la mémoire de tmp
        iii. Déplace le pointeur qui suit le livre précédent vers le pointeur suivant le livre à supprimer.

- fusion_LC:

1. Initialise un pointeur "cour2" la bibliothèque "b2"
2. Tant que cour2 n'est pas NULL
    a. Insère en tête de la bibliothèque b1, le livre pointé par cour2
    b. déplace cour2 sur le livre suivant
3. Libère la mémoire de "b2"
4. Retourne la bibliothèque b1 mise à jour avec les livres de b2

- recherchePlusieurs_LC:

1. Crée une nouvelle bibliothèque "bibliodoublon"
2. Crée un pointeur livreActuel parcourant la bibliothèque b
3. Tant que livreActuel n'est pas NULL:
    a. On initialise un pointeur livreCompare parcourant la bibliothèque b
    b. Tant que livreCompare n'est pas NULL:
        i. Si les numéros de livreActuel et livreCompare sont différents
        ii. Si les titres de livreActuel et livreCompare sont identiques
        iii. Si les auteurs de livreActuel et livreCompare sont identiques
        iv. Insère en tête livreActuel dans la bibliothèque "bibliodoublon"
        v. Déplace livreCompare sur le livre suivant
            a. Déplace livreActuel sur le livre suivant
4. Retourne la bibliothèque "bibliodoublon"

Algorithmes pour les structures avec table de hachage :

- inserer :

1.Calcule l'indice "ind" de hachage du livre à insérer à l'aide de son auteur et de la fonction fonctionClef

2.Crée un nouveau livre avec les attributs donnés

3. Déplace le pointeur suivant du nouveau livre vers la tête de la liste d'indice "ind" du tableau de hachage

4. Incrémente le nombre d'éléments de la bibliothèque

- rechercheLivreNum_H/rechercheLivreTitre_H:

1.Parcourt la table de hachage de la bibliothèque b
   a.Crée un pointeur "l" parcourant la liste chaînée à l'indice i
   b.Tant que l n'est pas NULL
      i.Retourne l si le numéro/titre du livre "l" égale le numéro/titre donné
      ii.Déplace le pointeur l vers le livre suivant dans la liste
   c.Incrémente l'indice i et passe à la liste chaînée suivante dans la table
2.Retourne NULL si aucun livre ne possède le numéro/titre donné

- LivresAuteurs_H:

1.Crée une nouvelle bibliothèque "res"
2.Parcourt chaque liste chaînée de la table de hachage de res
   a.Crée un pointeur "l" parcourant la liste chaînée à l'indice i
   b.Tant que l n'est pas NULL
      i.Insère le livre pointé par l dans res, si l'auteur du livre l est égal à l'auteur donné
      ii.Déplace l sur le livre suivant
   c.Incrémente l'indice i et passe à la liste chaînée suivante dans la table
3.Retourne la bibliothèque res contenant les livres de l'auteur donné

- supprimerLivre_H:

1.Calcule l'indice ind de la liste chaînée de l'auteur du livre à supprimer dans la table de hachage de la bibliothèque donnée.
2.Crée les pointeurs prec=NULL et cour à la tête de la liste chaînée d'indice ind
3.Si cour correspond au livre à supprimer.
   a.Si prec est NULL, alors le livre à supprimer se trouve en tête de liste
      i.Déplace la tête de la liste sur le livre qui suit le livre à supprimer "cour->suivant"
   b.Relie le livre précédent à cour->suivant, sinon.
   c.Libère l'espace mémoire du livre pointé par cour
   d.Décrémente le nombre d'éléments de la table de hachage
      1. Déplace prec sur le cour actuel et déplace cour sur le livre suivant ,sinon.

- fusion_H:
    1. Libère la mémoire de b2 et retourne b1, si la bibliothèque b2 est vide
    2. Parcourt chaque liste chaînée de la table de hachage de b2
        a. Crée un pointeur tmp contenant le livre actuel
        b. Insère dans b1 chaque livre trouvé dans la table
        c. Incrémente l'indice i et passe à la liste chaînée suivante dans la table
    3. Libère l'espace mémoire de b2

- recherchePlusieurs_H:
    1. Crée une nouvelle bibliothèque "bibliodoublon"
    2. Parcourt chaque liste chaînée de la table de hachage de b
        a. Crée un pointeur liveActuel parcourant chaque livre de la liste chaînée d'indice i
            i. Calcule l'indice de hachage "ind" de chaque livreActuel afin de parcourir la liste de livres de l'auteur du livreActuel
            ii. Crée un pointeur livreCompare parcourant la liste d'indice "ind"
    3. Si livreCompare possède les mêmes titres et auteurs que livreActuel, avec des numéros différents
        a. Insère livreActuel dans biblioDoublon
            i. Passe au livre suivant de la liste chaînée
        b. Passe au livre actuel suivant
    4. Retourne bibliodoublon

Réponses aux questions et analyse des performances des fonctions de recherches

La recherche par numéro d'une liste chaînée peut nécessiter un parcours complet de la liste, ce qui est peu efficace. Or, la recherche par numéro d'une table de hachage peut aussi nécessiter un parcours complet. Ce qui reste moins efficace que le parcours complet d'une liste chaînée.

La recherche par titre d'une liste chaînée peut nécessiter un parcours complet de la liste, ce qui est peu efficace mais le coût de cette recherche peut être comparable à celui d'une table de hachage.

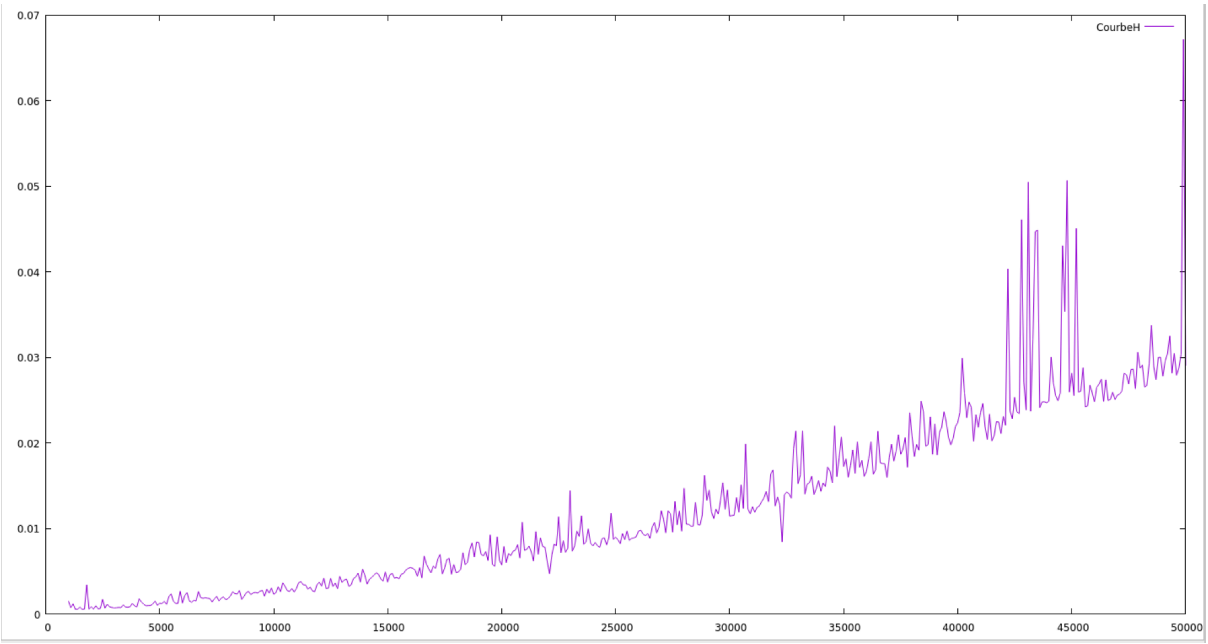
La recherche par titre d'une table de hachage peut être moins efficace car elle nécessite potentiellement une recherche séquentielle sur plusieurs indices d'une table de hachage. Les tables de hachage sont particulièrement adaptées à la recherche par auteur. Elle permet un accès direct à la liste des auteurs grâce aux clés de hachage.
En revanche, une liste chaînée peut nécessiter un parcours complet, ce qui est moins efficace que la recherche par hachage.

En résumé, la table de hachage semble plus adaptée pour la recherche par auteur. Cependant, la liste chaînée peut être plus appropriée pour la recherche par numéro et par titre.

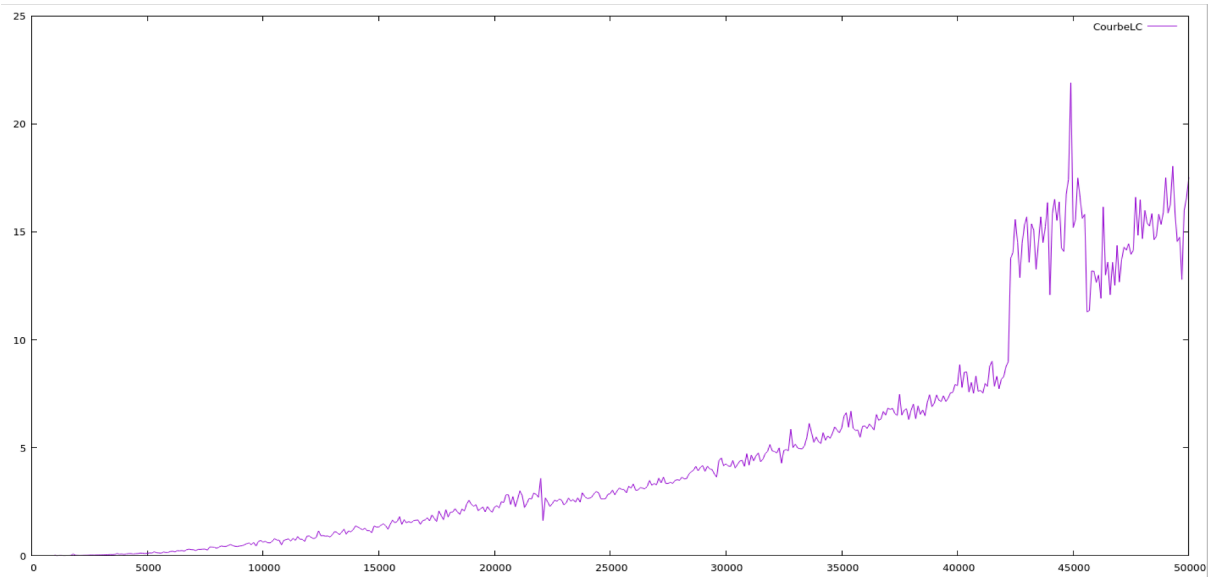
Une taille trop petite pour une table de hachage peut causer de nombreuses collisions selon les auteurs des livres insérés dans la table.
Ainsi, la taille d'une table influence uniquement la recherche par auteur.
Les recherches de numéro et de titre peuvent néanmoins être affectées par une taille trop grande, allongeant le temps de recherche.

Passons maintenant à la comparaison des deux structures pour la recherche de doublons dans une bibliothèques.

En prenant un pas de 100 avec une table de hachage de taille 80000, on peut constater, qu'en amoindrissant le pas, la courbe de recherche avec hachage contient de nombreuses variations à très petite échelle de temps. Malgré certains pics, qu'on a bien une augmentation du temps de calcul avec le nombre d'entrées ce qui peut ne peut pas être vu lorsque les courbes H et LC sont mises ensemble.



Lorsque que la courbe LC est toute seule, on remarque très peu de changements contrairement au cas où les deux courbes sont ensemble. On remarque tout de même des fluctuations au même niveau que celles de H.Surement dû à l'augmentation des entrées ou bien à la présence de livres en doublon.



Enfin lorsque les courbes sont mises ensemble, on remarque clairement que la recherche du modèle Hachage est énormément plus rapide que celle par Liste chaînée. On peut expliquer cela grâce au fait que, malgré que la complexité des deux pires des cas soit inclus dans $O(n^2)$, il est plus facile de trouver un livre associé à un auteur dans notre méthode de recherche avec Hachage. En effet, cela est dû au fait que chaque auteur est associé à un indice. Par conséquent, deux livres de même auteur sont toujours dans la même liste chaînée et on trouve quasiment instantanément un livre de cet auteur dans cette liste chaînée. On ne parcourt donc quasiment jamais la liste chaînée associée au livre d'un auteur en son intégralité.