Claude Chibout 21202978
Derya Keskin 21111316

Compte rendu Mini Projet 2: gestion d'une bibliotheque

Intro:

Dans ce projet, on traite la gestion d'une bibliothèque, composée d'un ensemble de livres.
Les livres ont un numéro, un auteur et un titre.
Les bibliothèques et livres sont représentés sous forme de structures.
Dans le projet nous allons comparer:

-les liste simplement chaînée struct (Partie 1).
-les tables de hachage struct (Partie 2).

Nous avons 2 structures Biblio et Livre qui utilise les listes chaînées.
Et 2 autres structures BiblioH et LivreH qui utilisent des tables de hachage.

Le projet comporte les fichiers suivant:

-bilbioC.c // où sont codé toutes les fonctions manipulant les livres et bibliothèques (Liste chainée)

-bilbioC.h // où on retrouve les prototypes des fonction dans biblioC.c et les structures bibliothèque et livres

-entreeSortieLC.c // qui implémente 2 fonction pour gérer les entrées et sortie avec les listes chainée

-entreeSortieLC.h // Header qui contient les prototypes des fonctions de entreeSortieLC.c

-biblioH.c // où sont codé toutes les fonctions manipulant les livres et bibliothèques (Table Hachage)

-biblioH.h // où on retrouve les prototypes des fonction dans biblioC.h et les structures bibliothèque et livres

-entreeSortieH.c // qui implémente 2 fonction pour gérer les entrées et sortie avec les tables de hachage

-entreeSortieH.h // header

main.c // contient 2 main un pour les listes chainée (mis en commentaire) et l'autre pour les tables de hachages

exo3.c // un fichier qui contient les réponses de l'exercice 3

exo1_test.c // fichier qui teste toutes les fonctions de l'exo 1

exo2_test.c // fichier qui teste toutes les fonctions de l'exo 2

Réponses exercice 3 :

Q1: La recherche par titre et par numéro est plus rapide avec les listes chainée
Car notre fonction de hachage utilisée prends en compte le nom de l'auteur
puisqu'on en a pas connaissance lors de ces recherches on ne peut pas
utiliser la puissance des tables de hachage

Alors que dans le cas de la recherches de tous les livres d'un auteur
on peut utiliser notre fonction de hachage, donc le plus rapide sera
la recherche avec la table de hachage

Q2: Augmenter le nombre de cases du tableau ne vas pas diminuer drastiquement
le temps pris pour les recherches

Q3: Graphe_temps_table_hachage_vs_liste_chaine.jpg est le graphe qui compare les deux versions
avec en ordonnée le temps en secondes
et en abscisse le nombre de livres dans notre bibliothèque

Graphe_zoom_table_Hachage.jpg est le graphe qui zoom sur la courbe pour la table de hachage

Q4: Le résultat des courbes s'explique grace à la complexité des algorithmes
Celui avec les listes chainée comprends deux boucles imbriquées
qui parcourent toutes les deux entièrement la liste

Alors que notre algo avec les tables de hachage parcours toutes
les listes avec la première boucle
(for + while)

ensuite on reparcours la liste chainée actuelle dans b->T[i]
dans le while le plus imbriqué

donc forcément on parcours moins d'élément
donc plus rapide

Essaie exo1:

Dans exo1_test.c on test toutes les fonctions demandé dans l'exo1
avec un affichage pour s'assurer que tout fonctionne
avec les structures qui implémente les listes chaînées

Essaie exo2:

Dans exo2_test.c on test toutes les fonctions demandé dans l'exo2
avec un affichage pour s'assurer que tout fonctionne
avec les structures qui implémente les tables de hachage