Fanxiang ZENG
Raphaël FRANÇOISE

Compte rendu TME2 - Structure de données

L'objectif de ce TME est de s'intéresser à la gestion d'une bibliothèque. Cette dernière est composée de livres. On distingue un livre grâce à son titre, son auteur et son numéro d'enregistrement.
Dans la première partie nous manions une liste simplement chainée de struct Livre. Nous travaillons aussi avec une structure Biblio qui pointe le premier élément de la liste de livres.
Dans la deuxième partie nous allons gérer la bibliothèque avec les tables de hachage.
Enfin dans une dernière partie nous allons comparer les complexité des différentes fonctions de recherche des deux structures données pour voir quelle structure de donnée est plus efficace pour la gestion d'une bibliothèque.

Notre dossier de code est composé des fichiers :

- GdeBiblio.txt : une bibliothèque de livres
- biblioLC.h : contient la définition des deux structures précédemment énoncées. Ce fichier contient de même la signature des différentes fonctions codées pour ces structures.
- biblioLC.c : contient la définition de toutes les fonctions manipulant ces structures.
- biblioH:c : idem biblioLC.c mais structure donné table hachage
- biblioH.h : idem biblioLC.h
- entreeSortieLC.h : contient les signatures des fonctions permettant de manipuler des fichiers.
- entreeSortieLC.c : contient la définition des fonctions permettant de manipuler des fichiers type GdeBiblio.txt.
- entreeSortieH:c : idem entreeSortieLC.c
- entreeSortieH.h : idem entreeSortieH:h
- main.c : contient le main et affichant le menu des actions possible sur la bibliothèque.
- mainH.c : idem main.c
- test.txt : pour faire des test de la fonction fusion
- comparaisonLC : mesure de la complexité des fonctions de recherche pour les listes chainees
- comparaisonH : mesure de la complexité des fonction de recherche pour les table de hachage
- Makefile : pour simplifier la compilation.

LC : structure liste chainé

H : table de hachage

Complexité pour les différentes fonctions de notre programme :

Pour LC :

Complexité des différentes fonctions du programme :

- O(1) : creer_livre, liberer_livre, creer_biblio, inserer_en_tete, afficheLivre
- O(n) : liberer_biblio, recherche_ouvrage_num, recherche_ouvrage_titre, recherche_ouvrage_auteur, affciherBiblio, suppression_ouvrage,, fusion_biblio_2, charger_n_entrees, enregister_biblio
- $O(n^2)$: recherche_meme_ouvrage

Pour H :

Complexité des différentes fonctions du programme :

- O(1) : creer_livre, liberer_livre, creer_biblio, inserer_en_tete, afficheLivre recherche_ouvrage_auteur
- O(n) : liberer_biblio, recherche_ouvrage_num, recherche_ouvrage_titre, fusion_biblio, charger_n_entrees, enregister_biblio
- $O(n^2)$: recherche_meme_ouvrage
-

n : le nombre d'éléments.

Les structures manipulées :

LC :

Livre : structure qui représente un livre, caractérisé par son titre, son auteur et son numéro d'enregistrement.

```
typedef struct livre {

    int num ;

    char * titre ;

    char * auteur ;
```

```
    struct livre * suiv ;
} Livre ;
```

Biblio : tête fictive qui pointe sur la première élément de la liste

```
typedef struct{ /* Tete fictive */
    Livre * L ; /* Premier élément */
} Biblio ;
```

Table de hachage :

LivreH : structure qui représente un livre, caractérisé par son titre, son auteur, son numéro d'enregistrement et sa clé.

```
typedef struct  livreh{
    int clef;
    int num ;
    char * titre ;
    char * auteur ;
    struct livreh * suivant ;
}LivreH;
```

BiblioH : structure qui contient notre table de hachage par liste chaînée, la taille du tableau, et le nombre d'éléments en totalité dans la table de hachage.

```
typedef struct table{
    int nE; //nombre d'elements contenus dans la table de hachage
    int m ; /*taille de la table de hachage */
    LivreH** T ; /*table de hachage avec resolution des collisions par chainage */

}BiblioH;
```

Réponse aux exercice :

Exercice 1 : voir code et Makefile

Exercice 2 : voir code

Exercice 3 :

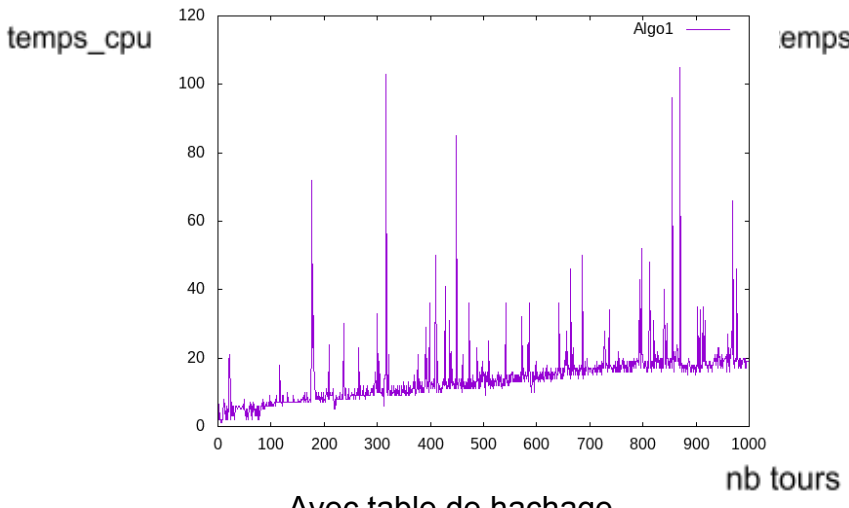
Réponses aux questions :

3.1) Nous avons observé grâce à nos tests que les fonctions de recherche par titre et par numéro sont sensiblement aussi rapides, que ce soit pour la liste chaînée ou bien pour la table de hachage. Cela s'explique par le fait que la table de hachage ne fait pas appel à sa clé dans ces deux fonctions. Le numéro et le titre n'influent pas sur la clé de la table de hachage. Pour ces deux recherches il importe donc peu d'utiliser une liste chaînée plutôt qu'une table de hachage (vis et versa).
Par ailleurs, nous avons observé que pour la fonction de recherche par auteur, l'utilisation d'une table de hachage s'avérait plus rapide que l'utilisation d'une liste chaînée. Cela est dû au fait que pour effectuer sa recherche, la fonction utilise la clé de la table de hachage, qui dépend de l'auteur, ce qui réduit fortement le temps de recherche car il y a juste à chercher directement dans le tableau.
Ces observations valent aussi bien que l'élément soit dans la liste ou pas.

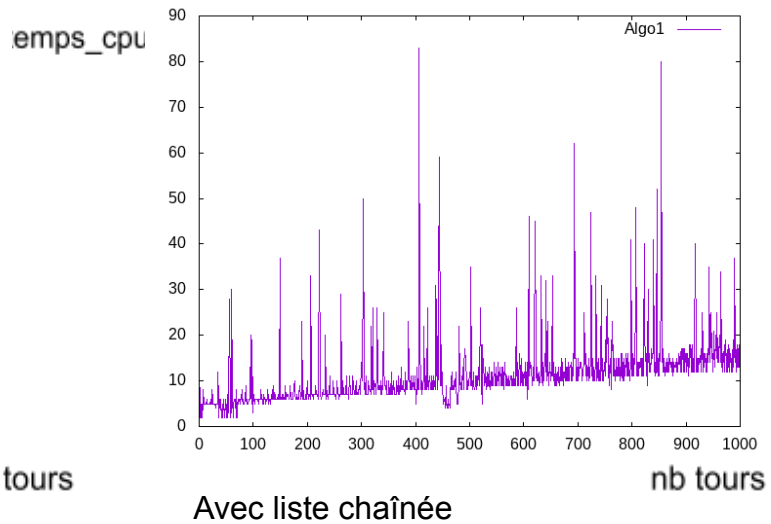
3.2) Si on augmente la taille de la table de hachage, le temps de calcul augmente dans le cas de la recherche par titre et par numéro. Cette modification n'influe pas quant à elle la fonction recherche par auteur.

3.3)

Courbes pour les fonctions de recherche par **numéro** :

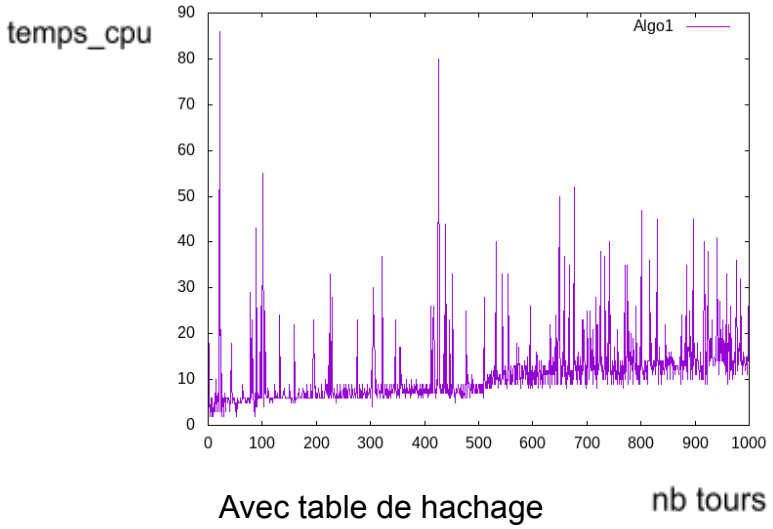


Avec table de hachage

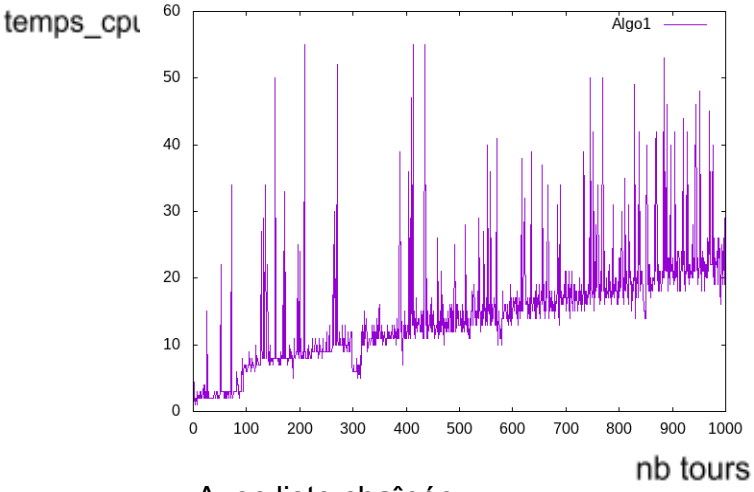


Avec liste chaînée

Courbes pour les fonctions de recherche par **auteur** :

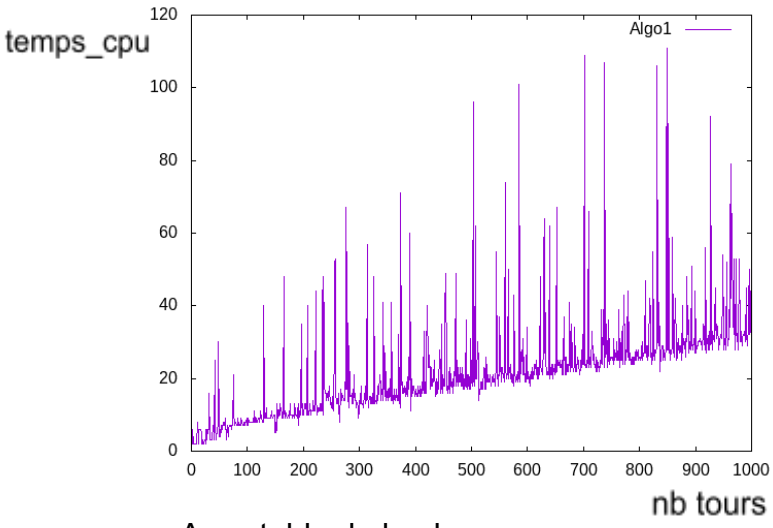


Avec table de hachage

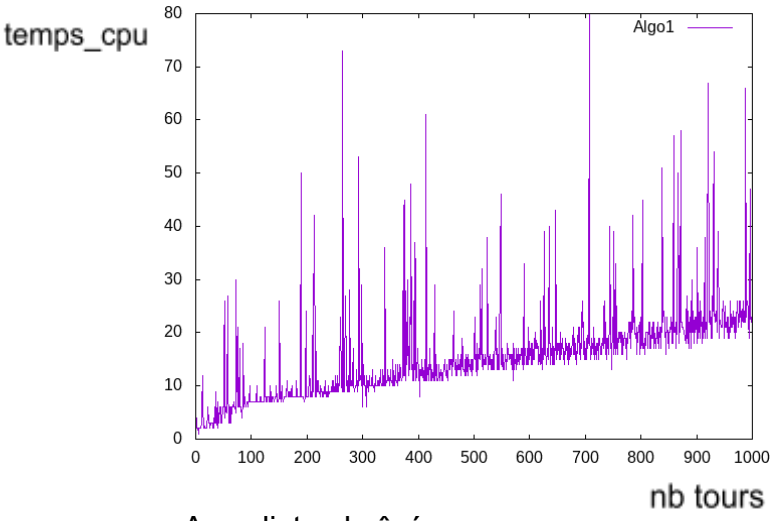


Avec liste chaînée

Courbes pour les fonctions de recherche par **titre** :



Avec table de hachage



Avec liste chaînée

3.4) Nous pouvons observer sur les courbes que pour les fonctions de recherche par numéro et par titre le temps de calcul est similaire. Cela s'explique par le fait que, dans le cas d'une liste chaînée ou d'une table de hachage, les deux fonctions ont la même complexité dans le pire cas O(n).
Cependant pour la fonction de recherche par auteur, le temps de calcul dans le cas de la table de hachage est moins important que pour la liste chaînée. On peut expliquer cette différence par le fait que dans le cas de la table de hachage la recherche par auteur est en O(1) dans le pire cas, tandis que pour la liste chaînée elle est en O(n) dans le pire cas.

Jeu de test :

nous avons deux fichiers pour faire les jeux de test, main.c pour faire les test des fonctions de manipulation de la bibliothèque sous forme de liste chainées , mainH. pour les fonctions de manipulation de la bibliothèque sous forme de table de hachage

En suivant les indices sur le terminal, vous l'utilisateur pourrait afficher, insérer, supprimer, rechercher, des ouvrages dans la bibliothèque. Puis, le jeu test nous permet également d'enregistrer la bibliothèque modifié dans un autre fichier, de fusionner le bibliothèque courant avec une autre bibliothèque (stocker dans un fichier), et de rafraîchir le bibliothèque avec un nouveau nombre de ligne