UNIVERSITÉ PARIS 13

Projet Structures de données et algorithmes
Codage binaire des caractères

Binôme

Mohand Lounis **BENSEKHRI** 11710457

Sofiane **HAMMOUM** 11508506

Année Universitaire : 2018 / 2019

L2 informatique

M. HAMIDI

M. VALENCIA

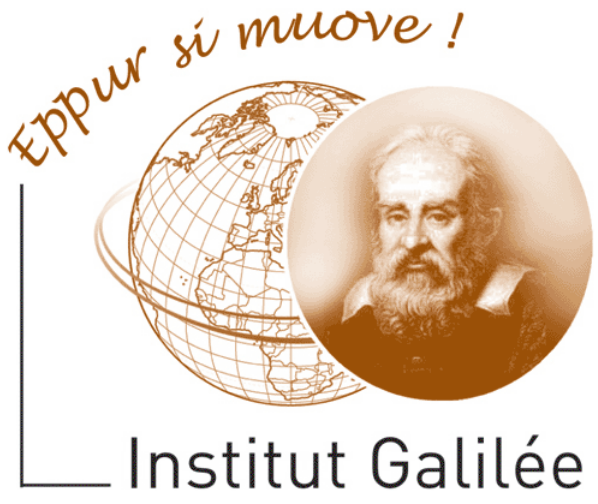

TABLE DES MATIERES

Introduction

Au cours de ce projet de Structures de données et algorithme nous avons pu mettre en pratique toutes les nouvelles connaissances que nous avons acquises au cours de ce premier semestre sur les arbres binaires et les tas.

Ce projet ne nous a pas pris beaucoup de temps pour le mettre en œuvre, c'est pour cela que nous avons choisi d'envoyer une première version de ce que nous avons qui sera nommé 1.1 qui est le mini projet demandé qui aura un seul fichier.c et ce rapport. Dans la deuxième version nous comptons faire un logiciel de compression plus performant, en effet il aura comme but de prendre un fichier texte de type .txt par exemple et créera un fichier codé en binaire et nous remarquerons bien sûr les bienfaits de ce codage vis-à-vis de la mémoire car comme nous le savons un entier prend beaucoup moins de mémoire qu'un char.

A propos de cette première version elle est assez basique car elle permet à l'utilisateur d'entrer une série de caractères avec leurs fréquences respectives. Ensuite créée un tas, un arbre de codage et enfin renvoie les codes binaire des caractères entrées.

Explications sur le code

A propos du code, on a essayé de le performer au mieux et cela en utilisant toutes les connaissances acquises et cela pour couvrir et éviter tout bug de la par du logiciel.

Le fichier.c nommé codage.c est constitué de trois parties tout en haut la première partie constitué des structures définies et les prototypes des fonctions utilisateurs juste après la deuxième partie qui est le main principal où l'on peut trouver les différentes instructions qui mettent en marche le logiciel et notamment les libérations de mémoires occupées par les données et enfin la troisième et dernière partie juste après le main là où on trouve les définitions de toutes les fonctions déclarées dans la partie 1.

Nous nous sommes contentés dans la majorité du temps d'utiliser que les fonctions à implémenter qui nous est demandé sauf dans quelques cas où nous avons préféré en mettre davantage pour plus de performances ou même pour faciliter d'autre fonction plus complexe. Par exemple pour la création de l'arbre de codage, nous avons rajouté deux fonctions juste avant pour trier un tableau de nœuds (tri_insertion) et un tas (trier_tas) pour que à chaque fois que nous ferons une insertion du nœud fusion crée à partir des deux minimum du tas, on met à jour le tableau de nœuds contenu dans le tas en le triant. Nous avons aussi rajouté une fonction (imprimer_codes_complet) qui est la fonction qui revoie le code des caractères entrés par l'utilisateur et cela d'une grande partie grâce à la fonction (imprimer_codes), cette fonction nous l'avons rajouté car nous avons remarqué que si l'utilisateur demande le code d'un seul caractère le code que nous avons écrit de la fonction (imprimer_codes) ne pouvait pas le fournir car cette dernière ne fonctionne que si la racine a au moins un fils. A propos du cas ou l'utilisateur demande le code de 0 caractères c'est-à-dire que le nombre de caractère de son alphabet est égale à 0 le logiciel prend fin.

A propos des explications des fonctions et tout ce qui est des paramètres et des return nous avons tout mentionnée dans les commentaires qui se trouve dans l'en tête de chaque fonction.