Compte-rendu Huffman

Oussama SAADAOUI & Mehdi ADJA

Attestation sur l'honneur

Nous soussignés Mehdi ADJA et Oussama SAADAOUI déclarons sur l'honneur que ce projet est le résultat de notre travail personnel et que nous n'avons pas copié tout ou partie du code source d'autrui afin de le faire passer pour le nôtre.

Comment compiler et exécuter vos programmes

Make: Le fichier exécutable est huffman. On attend comme paramètre un fichier d'entrée .txt (disponibles dans ascii). Une fois exécuté, trois principaux nouveaux fichiers seront créés:
- current-tree.pdf: Contient une représentation visuelle de l'arbre d'Huffman du fichier d'entrée
- encodage.txt: Contient la version encodée du fichier d'entrée, avec 3 majeures lignes:
    -1er: Arbre binaire du fichier
    -2ème: Nombre de caractères du fichier
    -3ème: Version binaire du fichier
- decodage.txt: Contient la version décodée du fichier d'encodage, soit le même fichier que le fichier d'entrée (on peut tester avec `diff ascii/entree.txt decodage.txt`)

A chaque Make, les fichiers encodage.txt et decodage.txt sont nettoyés
Un Make clean est aussi disponible pour tout nettoyer (encodage, decodage et current-tree inclus)

Lesquels parmi les fichiers d'exemples fournis sur elearning peuvent être correctement traités et les bugs connus

La majorité des problèmes de Huffman vient du write_tree, qui ne s'attend pas à devoir écrire certains caractères.

ASCII
- 2city11.txt: Encodage/décodage fonctionnel, Huffman non fonctionnel
- 6502.txt: Encodage/décodage fonctionnel, Huffman non fonctionnel
- banana01.txt: Encodage/décodage fonctionnel, Huffman semi fonctionnel
- beer-gui.txt: Encodage/décodage fonctionnel, Huffman non fonctionnel
- dna.txt: Encodage/décodage fonctionnel, Huffman non fonctionnel
- wave.txt: Encodage/décodage fonctionnel, Huffman non fonctionnel (problème d'UTF ?)

Binaires: non traités (on traite uniquement des fichiers txt)

Les problèmes découverts pendant la soutenance bêta et comment vous les avez résolus

Deux problèmes majeurs ont été relevé lors de la soutenance bêta par M. De Mesmay:
les problèmes d'allocation et libération et la manière dont l'une de nos fonctions a été codée (fill_tab_code)

- Allocation/Libération: Le majeur problème était la manière dont on allouait notre tableau de code (code_table). Notre soucis était de savoir quelles cases devaient être alloués pour contenir un caractère, et quelles cases ne le sont pas afin de savoir quelles cases libérer.

Pour palier à ce problème, à l'initialisation, toutes les cases de notre table sont initialisées à NULL et lorsqu'on remplira notre tableau, on remplacera les cases des caractères du fichier par sa version binaire. Lorsqu'on va libérer la table, on libère toutes les cases de notre table, et on ne fera ainsi pas l'erreur de libérer des pointeurs non initialisées.
Programme vérifié avec valgrind (valgrind -v --leak-check==full)

-fill_tab_code: Le problème majeur sur cette fonction était qu'à chaque transition pour stocker pour chaque caractère sa version binaire (0 si on va à gauche, 1 si on va à droite), au début de la fonction, on initialise une chaîne de caractères vide, et on ajoute à cette chaîne de caractères un 0 ou un 1 en créant une nouvelle chaîne contenant le bon chiffre et en utilisant l'utilitaire strcpy pour ajouter le chiffre à notre chaîne actuel avant de faire un appel récursif, ce qui prenait beaucoup de mémoire.

Pour corriger ce soucis, il nous a suffit d'ajouter un nouveau paramètre à notre fonction, qui est l'indice sur lequel on pointe sur notre chaîne de base str (str_pos), et pour ajouter un 0 ou un 1, on effectue simplement str[str_pos] = "0" ou "1" ET str[str_pos] = "\0", pour garder la fin de la chaîne.

On se retrouve avec une fonction beaucoup plus lisible et moins lourde, qui utilise moins de mémoire.

Comment vous avez mené le travail en binôme

On s'est réparti le travail en exercices (un exercice chacun), en étant en même temps en conversation vocale pour bien comprendre ce qu'a fait l'autre et s'aider.
On a utilisé Git pour versionner et partager notre code ainsi que de travailler en dynamique sur les mêmes fichiers.

Les difficultés rencontrées et solutions apportées, leçons tirées

Par manque de temps (nécessité d'avancer sur le projet Java et Web), nous n'avons pas pu faire la partie 5.
On a rencontré quelques problèmes de fuites mémoires lorsque l'on se servait de malloc et de boucles mais avec les conseils de M. De Mesmay, on a réussi rapidement à les corriger.

On a un mis un peu de temps à adapter la prioqueue à nos fonctions. (notamment qu'il fallait modifier notre structure node en y ajoutant un champs priority)