Rapport Projet Huffman
2018-2019

Etudiants sur le projet :

Canta Thomas
Desgenetez Charles
Reiter Maxime

Notes : Une grande partie du code est extensivement commenté donc ce rapport ne sera pas sur la description des fonctions. N'hésitez pas à regarde le listings_functions.pdf

Nouveaux aspects découverts du langage C

Les structures :

Les structures étaient très intéressantes à abordé, en effet c'est la première fois que on pouvait les utilisez de façon extensible ce qui nous a permit de coder de façon différente notre programme, qu'avec des tableaux simples.
Avec les structures nous avons appris à utiliser différentes fonctions comme le malloc(), calloc(), free() ou même des memset() qui sont des éléments essentiels à l'utilisation des tableau dynamique, ce sont aussi des fonctions pas facile à comprendre sans les avoirs utilisés plusieurs fois.

Traces écrites

Les traces écrites sont souvent renier face au codage pur direct alors que cela nous a débloqué a plusieurs de nos problèmes lors de l'écriture de certaines fonctions compliquées comme celle de la création de l'arbre et de son codage binaire associé. *(On l'admet vous avez raison)*

Implémentation du codage de l'arbre d'Huffman

Création d'un tableau d'occurrence

La création du tableau d'occurrence fut l'étape la plus simple du projet, effectivement celle-ci fût écrite en cours lors d'un Travaux Pratique.

Création de l'arbre (arbre.c, arbre.h)

La création de l'arbre d'Huffman fut la partie la plus compliqué : il y'a beaucoup de condition à vérifier lors de sa création ce qui peut provoquer beaucoup d'erreurs si une des conditions est fautives.
Nous utilisons principalement deux structures :

- nœud : sous-arbre reliant un père à ses deux fils et la somme de leurs probabilités
- dictionnaire : définie un dictionnaire composé d'un caractère, de son code binaire ainsi que la taille du code.

Compresseur :

La partie la plus compliqué de cette partie était de trouvé comment écrire dans un fichier le codage binaire, en effet il ne suffit pas de juste écrire les 0 et les 1 dans un fichier de façon quelconque, sinon le codage binaire afficher n'est pas binaire mais seulement une suite de 0 et de 1 en forme de caractères.
Une fois la solution connue on a pût créer un entête permettant, lors du décompressage, de récupérer les informations nécessaires pour pouvoir décoder. Puis par la suite, le codage binaire caractère par caractère du fichier.

La partie la plus compliqué devient alors de prendre en compte par exemple, si le nombre de bits nécessaire afin de coder le nombre de caractère total est supérieur à 8 il va falloir le coder sur plusieurs octets. Dans l'entête de notre fichier compressé on retrouve donc :

- Le nombre de caractères différents
  - Coder sur 1 octet maximum car il n'existe que 255 caractères dans la table ASCII
- Le nombre d'octet nécessaire pour coder le nombre de caractères totaux
  - Peut être codé sur plusieurs octets
- Le nombre de caractère totaux présent dans le fichier source
- Puis l'arbre de Huffman dont 3 octets sont affiliés à chaque caractères
  - 1er octet : le codage binaire du caractère
  - 2ème octet : la taille du code associé au caractère
  - 3ème octet : son codage binaire

Décompresseur :

Pour décompresser le fichier on eu les plus grosses difficultés lors de la lecture de l'entête fût celle de récupérer le nombre de caractère totaux si celui-ci était supérieur à 1 octet. Après, la création du dictionnaire et le décompressage fût plutôt simpliste.