Rapport Huffman

Paul BUNEL & Esteban BARON

17 Décembre 2019

Partie 1

Compresseur

1.1 Fréquences

Les fréquences sont représentées dans un tableau de taille 256 (pour les 256 caractère ASCII), tel que les indices correspondent aux caractères de la table ASCII.
Ainsi pour chaque caractère on récupère le nombre de fois qu'il apparaît dans le fichier et on calcule sa fréquence.

1.2 Arbre

Chaque noeud est représenté par un objet d'une structure noeud, ayant pour attributs : le noeud père, les noeuds fils gauche et fils droit, ainsi que la fréquence. L'ensemble de l'arbre est représenté par un tableau de 511 noeuds : les 256 premiers correspondants aux feuilles, et donc aux éventuels caractères ASCII, et les 255 autres aux noeuds non-feuilles.
Il est construit par la fonction creationArbre(), qui associe à chaque feuille la fréquence du caractère correspondant, puis qui détermine les arbres à construire au fur et à mesure pour obtenir l'arbre Huffman final.

1.3 Génération du code Huffman

On crée un tableau de caractères d'une taille de 256 (pour les 256 caractère ASCII) et on définit le code de chaque élément du tableau en parcourant l'arbre. On appelle la fonction récursivement tel qu'on ajoute un "0" pour aller à gauche et un "1" pour aller à droite par exemple.

1.4 Compression dans le fichier

Nous avons choisi d'utiliser la méthode qui consiste à encoder l'arbre Huffman en binaire puis l'écrire dans le fichier avant le code, afin de pouvoir ensuite déchiffrer ce dernier grâce à l'arbre.
Notre fichier compressé est donc composé de la manière suivante : d'abord 3 bits pour indiquer le nombre de bits restants en trop à la fin (si notre Buffer se

termine avant le dernier bit d'un octet), puis n bits pour l'arbre et le reste (sauf les bits restant) pour le code à déchiffrer.
L'arbre, lui, est encodé de la manière suivante : on part de la racine ; si le noeud actuel n'est pas une feuille, on écrit un bit 0, si c'en est une on écrit 1 suivit du code binaire du caractère ASCII correspondant.

Partie 2

Décompresseur

2.1 Reconstitution du buffer

Nous allons tous simplement recréer le buffer du fichier pour le lire bit à bit.

2.2 Bits restants

Nous lisons les trois premiers bits qui servent à savoir combien de bits il y aura en trop dans le dernier octet du buffer. Pour cela nous utilisons notre fonction "bitrestants()".

2.3 Récupération de l'arbre

Dans cette partie, les noeuds sont représentés par une structure noeud ayant pour attributs un caractère et ses fils gauches et droits. L'arbre est retrouvé grâce à la fonction recupArbre : grâce à la structure BitRead, on lit les bits du buffer : si c'est un 1 (donc une feuille) on crée une feuille qui stocke le caractère correspondant aux 8 bits suivants ; sinon, on crée un nouveau noeud et on rappelle récursivement la fonction recupArbre pour ses deux fils.

2.4 Déchiffrage des caractères binaires

Dans cette partie nous déchiffrons le code binaire du reste du buffer à l'aide de l'arbre et le mettons dans une chaine de caractère qui nous servira pour restituer le fichier d'origine.
Pour cela nous utilisons notre fonction "dechiffrage()".