RAPPORT PROJET ALGORITHMIQUE

# COMPRESSION DE DONNÉES PAR CODAGE DE HUFFMAN

**Descriptif général**

Le codage de Huffman, du nom de son concepteur, est une méthode statistique de compression de données. Son principe est de remplacer un caractère (ou symbole) par une suite de bits de longueur variable. L'idée sous-jacente est de coder ce qui est fréquent sur peu de bits et au contraire ce qui est rare sur des séquences de bits plus longues. Il permet une compression sans perte, c’est-à-dire qu’une suite de bits strictement identique à l’originale est restituée par décompression. Il nécessite cependant que soit connues (ou estimées) les fréquences d’apparition des différents symboles à coder. Il existe ainsi plusieurs variantes de l’algorithme de Huffman (statique, semi-adaptatif ou adaptatif) aujourd’hui utilisées dans des algorithmes de compression de fichiers tels que gzip. Ce sujet concerne la version semi-adaptative de l’algorithme dans laquelle le texte à coder est tout d’abord lu intégralement de façon à construire l’alphabet et déterminer les fréquences d’apparition des éléments de l’alphabet.

Etape 1 : Détermination de l’alphabet et des fréquences de caractères

L’alphabet sera composé des caractères présents dans le texte fourni et uniquement de ceux-ci. La fréquence des différents caractères de l’alphabet dans le texte sera déterminée. Le terme fréquence est ici, et dans toute la suite, utilisé pour une fréquence absolue, c’est-à-dire un nombre d’occurrences des caractères dans le texte. L’ordre des caractères de l’alphabet sera maintenu par fréquence croissante puis par ordre de codage des caractères ASCII.

Afin de réaliser cette première étape j’ai tout d’abord réalisé plusieurs fonctions.

Une image contenant texte

Description générée automatiquement

Cette première fonction récupère les données d’un fichier tel que tous les caractères du fichier.

Une image contenant texte

Description générée automatiquement

Cette deuxième fonction permet d’ajouter à un dictionnaire tous les différents caractères du fichier texte, en comptant leurs fréquences. Puis récupères tous les items du dictionnaire afin de les classer en fonction de leur fréquence puis dans l’ordre alphabétique. Et cette fonction retourne une liste de tuples de la forme (« caractère », fréquence).

Etape 2 : Construction de l’arbre

L’algorithme est décrit dans l’article de son créateur publié en 1952. Il repose sur une structure d’arbre binaire où tous les nœuds internes ont exactement deux successeurs. Les feuilles sont étiquetées avec les caractères de l’alphabet, les branches par 0 (fils gauche) et 1 (fils droit). Les chemins depuis la racine jusqu’aux feuilles constituent les codes des caractères.

La construction de l’arbre est réalisée de la manière suivante :

Créer un arbre (feuille) pour chaque caractère de l’alphabet avec la fréquence associée Répéter

Déterminer les 2 arbres t1 et t2 de fréquence minimale avec t1.freq  t2.freq. Créer un nouvel arbre t avec t1 et t2 comme sous-arbres respectivement gauche et droite avec t.freq = t1.freq  t2.freq. Jusqu’à ce qu’il ne reste plus qu’un seul arbre.

Pour la réalisation de cette étape, j’ai aussi fait plusieurs fonctions afin de me simplifier la programmation de cette partie.

Une image contenant texte

Description générée automatiquement

Tout d’abord, j’ai réalisé cette fonction qui permet de parcourir la liste de tuples que j’ai créé au-dessus afin de transformer cette liste de tuples en une liste de nœuds.

Une image contenant texte

Description générée automatiquement

Ensuite j’ai effectué cette fonction qui permet de créer petit à petit des sous-arbres et de les classer jusqu’à ce qu’il ne reste que l’arbre.