LABORATOIRE #1
Algorithme de Huffman
Version 1.4

Patrick Cardinal **Durée :** 3 semaines

Introduction

L'algorithme de Huffman est une méthode statistique de compression de données. Un fichier texte (ou binaire) est représenté par une série d'octets (séquence de 8 bits). Le principe de l'algorithme est de transformer chacun de ces octets en une séquence de bits dont la longueur est variable. La longueur de la nouvelle séquence qui représente un octet spécifique est calculée en fonction de sa fréquence dans le fichier. L'algorithme va faire en sorte que les octets plus rare vont être codés par une séquence plus longue (et possiblement de plus de 8 bits) tandis que les octets plus fréquents seront codés par une séquence plus courte et inférieure à 8 bits. Il a été démontré que les gains obtenus par les octets les plus fréquents surpasse les pertes causées par les codes plus long des éléments les moins fréquents. Le résultat net est donc un fichier dont la taille est inférieure à sa taille originale.

Une propriété importante de l'algorithme est qu'aucun code n'est le préfix d'un autre code. Ceci veut dire que si par exemple un octet est codé par la séquence '001', aucun autre code va commencer par cette séquence. Cette propriété assure qu'il n'y aura pas de confusion lors du décodage.

L'algorithme

Table de fréquences

L'algorithme de compression se décompose en trois phases. La première phase consiste à lire le fichier à compresser et de construire une table de fréquences. Cette table contient le nombre d'occurences de chacun des caractères dans le fichier à compresser. L'algorithme ci-dessus décrit cette opération.

CréerTableFréquences

```
1: T ← ∅
2: for all c ∈ File do
3:    if c ∈ T then
4:       T[c] ← T[c] + 1
5:    else
6:       T ← T ∪ {c}
7:       T[c] ← 1
8: return T
```

La table de fréquences doit ensuite être triée en ordre décroissant selon la fréquence des caractères. La figure 1 représente une table de fréquences à la fin du processus.

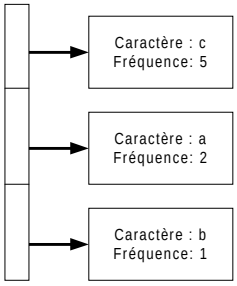

FIG. 1 – *Table de fréquences.*

Vous remarquerez que les éléments de la table sont en fait des feuilles de l'arbre binaire qui sera créé lors de la deuxième phase. Cette technique permettra de faciliter la construction de l'arbre binaire.

Construction de l'arbre binaire

La seconde phase consiste à créer l'arbre binaire en fonction de la table de fréquences. L'algorithme effectue des réductions sucessives de la table de fréquences. L'opération de réduction consiste à extraire de la table les deux éléments ayant les fréquences les plus faibles et à les combiner. La combinaison amène la création d'un nouveau noeud de l'arbre dont les enfants seront les deux éléments extrait de la table. La fréquence du noeud ainsi créé se trouve à être la somme des fréquences de ses deux enfants.

La figure 2 montre comment les deux éléments de la table représentée à la figure 1 ayant les fréquences les plus faibles sont combinés afin de former un nouvel élément.

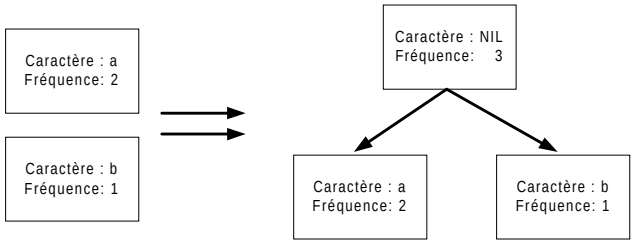

FIG. 2 – *Combinaison de deux éléments de la table de fréquences.*

Le nouvel élément ainsi créé est replacé dans la table à la position correspondant à la fréquence combinée des deux noeuds enfants. **Il est important que la table demeure toujours en ordre décroissant**. L'opération est ainsi répétée jusqu'à ce que la table ne contienne qu'un seul élément qui sera la tête de notre arbre binaire.

La figure 3 nous montre l'arbre binaire créé à partir de la table de fréquences de la figure 1.

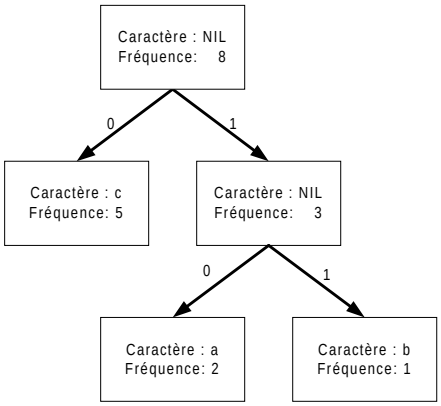

FIG. 3 – *Arbre binaire construit à partir de l'arbre de fréquences de la figure 1.*

Encoder le fichier source

La troisième phase de la compression consiste à encoder chacun des caractères du fichier source à l'aide des codes obtenus en traversant l'arbre binaire. Vous remarquerez que chacune des branches de l'arbre correspond à une valeur binaire (0 ou 1). Le code d'un caractère est donc la suite de bits obtenus en traversant l'arbre à partir de la tête jusqu'au caractère à encoder. Par exemple, le caractère 'a' sera codé '10' tandis que le caractère 'c' sera codé '0'.

Compresser le fichier consiste donc à remplacer les codes ascii (toujours codés sur 8 bits) par les codes déterminés par la structure de l'arbre. Dans notre exemple, nous avions 1 'b', 2 'a' et 5 'c' qui occupait un espace de 64 bits. Encoder, le même texte occupera un espace de 11 bits.

Cependant, il ne faut pas oublier qu'il faut aussi sauvegarder les informations nécessaires au décodage afin de pouvoir décompresser le fichier. Il est clair que dans le cas de petits fichiers, ces informations supplémentaires risquent de créer un fichier compressé plus gros que le fichier original !

Décompresser un fichier

La décompression d'un fichier encodé est relativement simple. Premièrement, il faut récupérer les informations nécessaires à la reconstruction de l'arbre binaire. Ces informations sont typiquement sauvegardés dans l'en-tête du fichier compressé. Deuxièmement, il suffit de traverser l'arbre binaire en considérant les bits du fichier compressé. À chaque fois qu'on atteind une feuille de l'arbre, un caractère à été décodé, on continue de cette manière jusqu'à la fin du fichier.

Vérification du programme

À la date déterminée par le chargé de laboratoire, vous devrez faire la démonstration que votre programme fonctionne. À ce moment, le chargé de laboratoire vous fournira un fichier que vous devrez compresser et décompresser afin de montrer le bon fonctionnement de votre programme.

Noter bien que le fichier de test peut contenir n'importe quel octet entre 0 et 255.

Rapport de laboratoire

Le rapport de laboratoire doit contenir les éléments suivants :

1. Description de votre programme,
2. Analyse de complexité (asymptotique) des algorithmes,
3. Description des problèmes rencontrés,
4. Description des améliorations que vous avez implémentées.

Votre rapport doit avoir un **maximum de 4 pages**. Toutes les pages supplémentaires seront ignorées lors de la correction.

Barème de correction

Fonctionnement du programme	4
Performances	
-Vitesse d'execution	1
-Taux de compression	1
Rapports	4
Total	10

Les points pour la vitesse d'execution seront accordés si le temps de traitement est raisonnable (évalué à l'oeil). Les points pour le taux de compression seront accordés aux équipes dont le programme surpasse la performance du démo d'au moins 400 bytes.