Structure de données

Compression de texte par le codage de Huffman

1. Cahier des charges

Le codage de Huffman (1952) est un codage statistique utilisé pour la compression de données telles que les textes, les images (fichiers JPEG) ou les sons (fichiers MP3). L'objectif de ce projet est de programmer un compresseur et un décompresseur de fichiers texte en s'appuyant sur ce codage.

Le codage de Huffman par l'exemple. Supposons le texte « mohamed maachaoui ». Il est composé de 17 caractères. En utilisant le codage ASCII, ce texte utilise 17 octets (1 caractère est codé sur un octet).

Supposons maintenant la table de codage de la table 1.

Caractère	'm'	'o'	'h'	'a'	'e'	'd'	' '	'c'	'u'	'i'
Code	111	011	010	00	1011	1010	1000	1001	1101	1100

TABLE 1 – Table de Huffman pour le texte « mohamed maachaoui »

Le texte est alors codé sur 7 octets :

111.011.010.00.111.1011.1010.1000.111.00.00.1001.010.00.011.1101.1100

Cette table de codage est un codage de Huffman. Les codes de chaque caractère sont obtenus à partir de leur fréquence d'apparition dans le texte. Les étapes pour construire cette table de codage seront :

1. Déterminer la fréquence d'apparition de chaque caractère dans le texte ;
2. Construire un arbre de Huffman ;
3. Lire le code de chaque caractère dans l'arbre.

Fréquence d'apparition des caractères. Il s'agit de lire le texte en entrée pour comptabiliser le nombre d'occurrences des caractères qu'il contient.

Par exemple, pour le texte « mohamed maachaoui », on obtient :

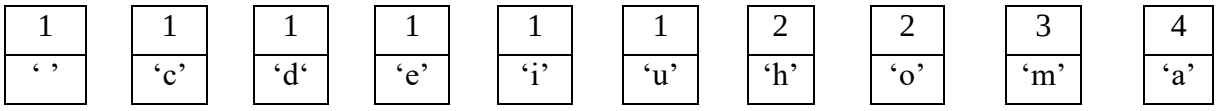
Caractère	'm'	'o'	'h'	'a'	'e'	'd'	' '	'c'	'u'	'i'
Code	3	2	2	4	1	1	1	1	1	1

Algorithme de Huffman. Le codage de Huffman a pour but d'attribuer à un caractère un code qui est d'autant plus court que ce caractère apparaît fréquemment dans le texte.

Pour ce faire, l'algorithme de Huffman utilise un arbre binaire dont chaque feuille enregistre un caractère et sa fréquence dans le texte et dont chaque nœud possède une fréquence qui est la somme des fréquences des nœuds racines de ses sous-arbres droit et gauche. Pour chaque nœud, la valeur de la racine du sous-arbre gauche est inférieure à celle de la racine du sous-arbre droit.

La figure 6 présente l'arbre correspondant à notre exemple.

Initialement, on construit une liste d'arbres possédant une unique feuille correspond à un caractère et sa fréquence.



L'algorithme de Huffman consiste alors à remplacer dans la liste les deux arbres ayant les fréquences les plus petites par un nouvel arbre dont les sous-arbres sont les deux enlevés. Par exemple, dans la liste précédente, les arbres ayant pour fréquence 1 sont retirés puis un nouvel arbre de fréquence 2 est inséré dans la liste. Ce nouvel arbre a pour sous-arbres les arbres corresponds aux caractères ' ' et 'c'. La liste ainsi modifiée est donnée à la figure 1.

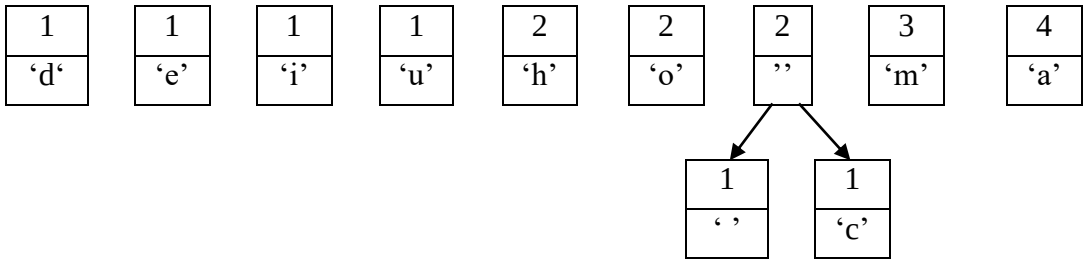


FIGURE 1 – Construction de l'arbre : étape 1

Les étapes de construction de l'arbre sont détaillées dans les figures 1 à 5

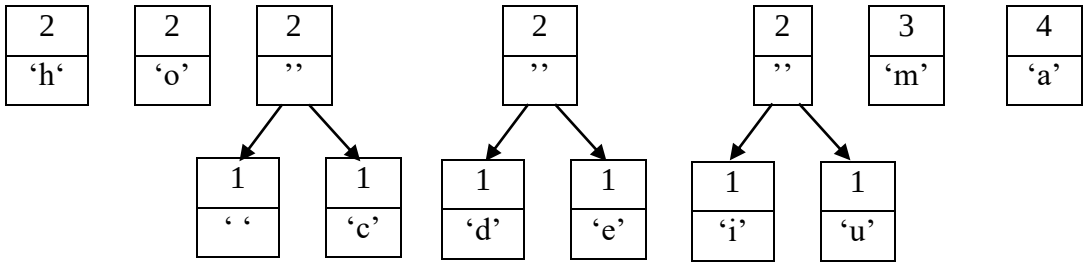


FIGURE 2 – Construction de l'arbre de Huffman : étape 2

L'algorithme se termine lorsqu'il ne reste plus qu'un seul arbre dans la liste (figure 6).

Remarque : L'arbre de Huffman n'est pas unique. Il dépend en particulier de l'ordre des feuilles dans la liste initiale.

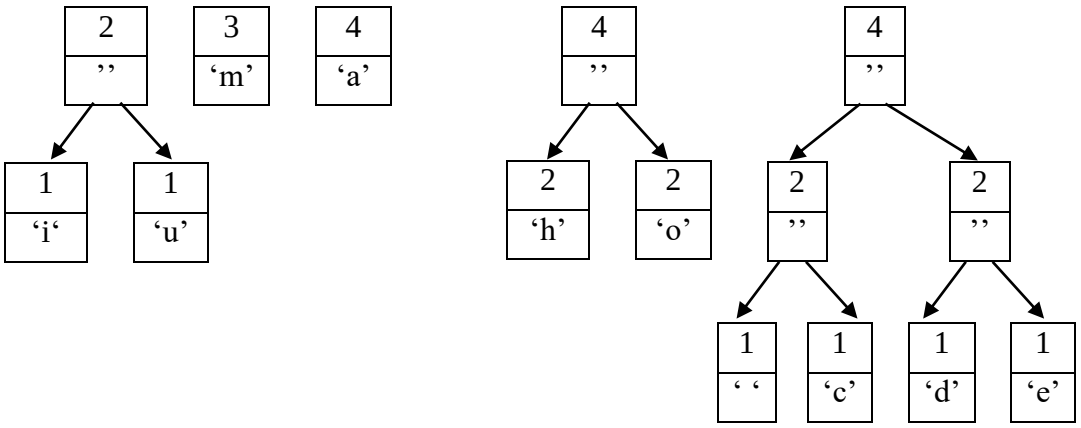


FIGURE 3 – Construction de l'arbre de Huffman : étape 3

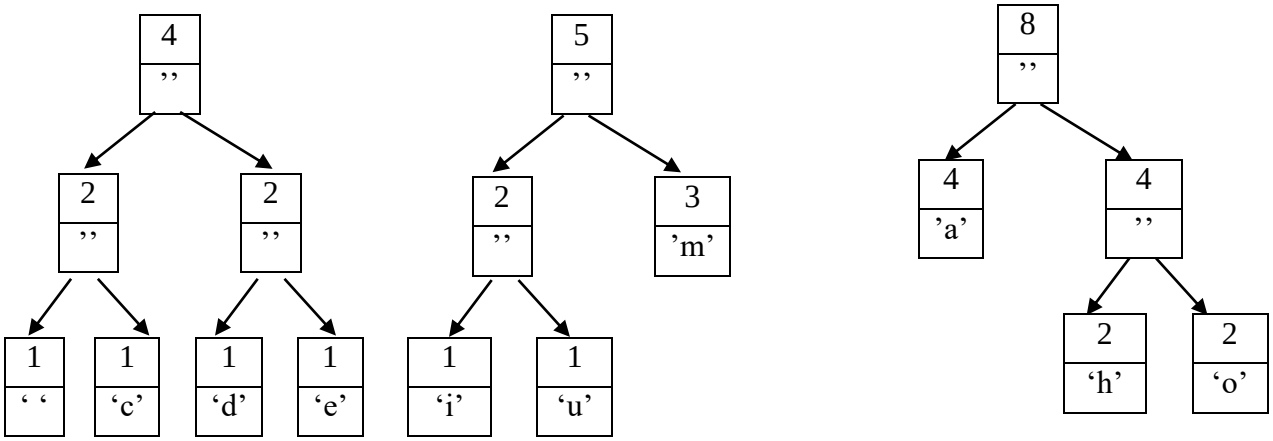


FIGURE 4 – Construction de l'arbre de Huffman : étape 4

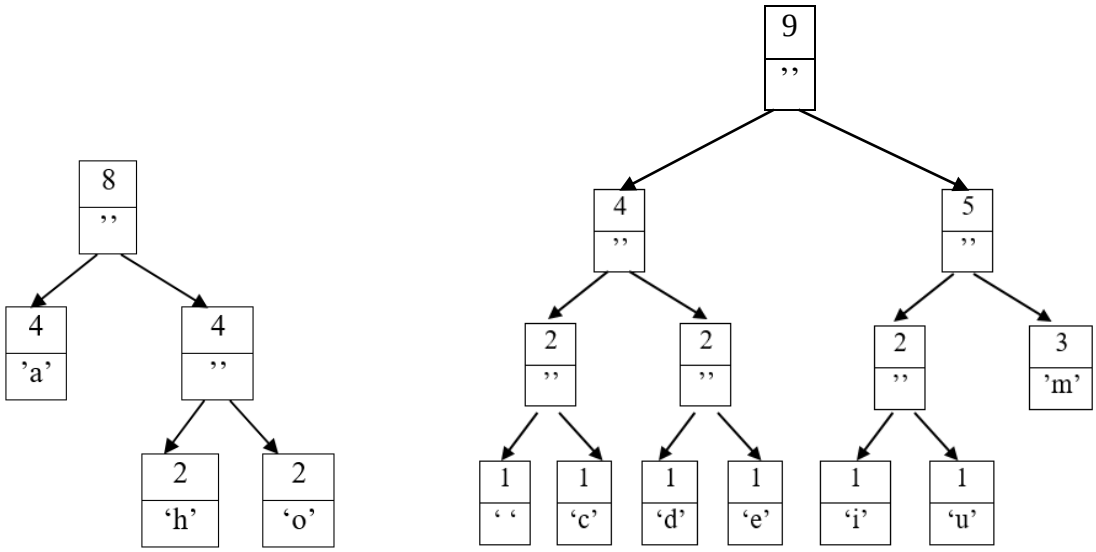


FIGURE 5 – Construction de l'arbre de Huffman : étape 5

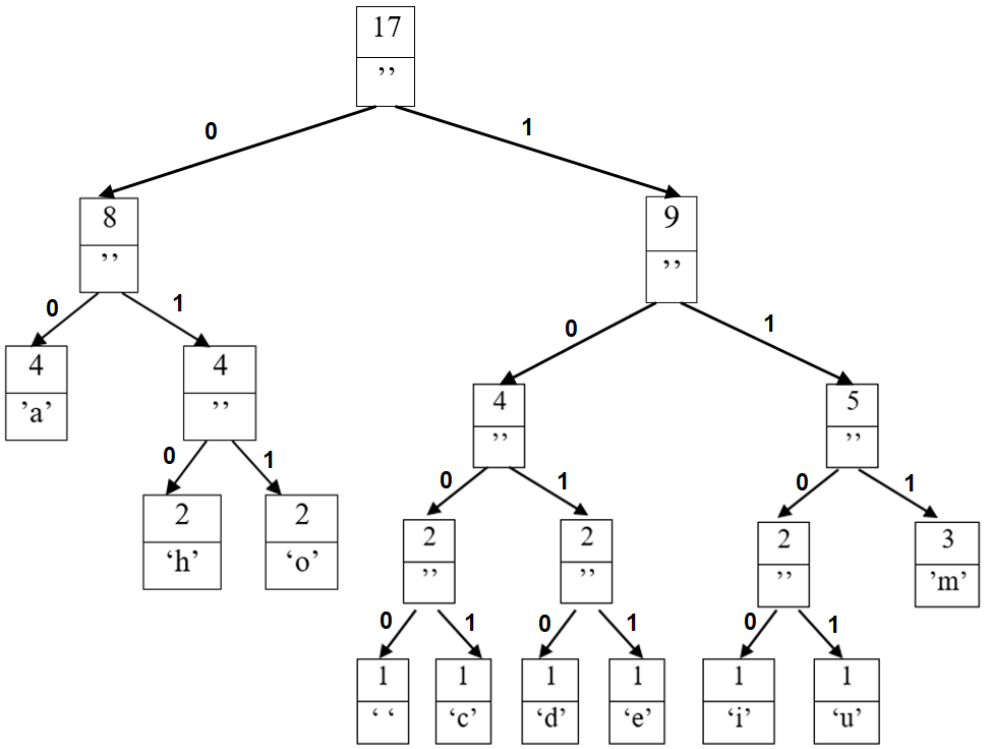


FIGURE 6 – Arbre de Huffman final

Table de Huffman Chaque branche de l'arbre est étiquetée par la valeur 0 pour le sous-arbre gauche et la valeur 1 pour le sous-arbre droit (voir figure 6). La table de Huffman ainsi obtenue est celle donnée en table 1. Le caractère 'a', présent 4 fois dans le texte, possède le code le plus petit (2 bits).

Encodage du fichier compressé. Dans le fichier engendré, il faudra commencer par encoder la table de Huffman de manière à pouvoir ensuite décoder le fichier. Le format choisi est le suivant :

- 4 octets pour la taille du texte non décompressé : 17 sur l'exemple,
- un octet pour la taille de l'arbre (le nombre de caractères) : 10 sur l'exemple,
- chacun des caractères de l'arbre (parcours infixe, en profondeur) : 'a', 'h', 'o', ' ', 'c', 'd', 'e', 'i', 'u' et 'm' sur l'exemple,
- des bits 0 ou 1 correspondant au parcours infixe de l'arbre : 0 on descend (d'abord à gauche, puis à droite) et 1 on remonte. On ne met qu'un seul 1 pour la dernière feuille. Cet encodage marche ici car chaque nœud de l'arbre a deux sous-arbres. Sur l'exemple, on a donc : 0010010111000010110010111000101101,

La suite du fichier contient le code donné par la table de huffman de chacun des caractères de l'entrée.

Interface utilisateur. L'interface avec l'utilisateur se fera au moyen de la ligne de commande.

On définira une première commande appelée `compress` qui prend un seul argument correspondant au fichier à compresser. Elle engendre le fichier compressé dans un fichier de même nom avec l'extension "`.hf`". Ainsi, `compress exemple.txt` produit un fichier `exemple.txt.hf`

Une deuxième commande appelée uncompress permettra de décompresser le fichier donné en argument sur la sortie standard.

On écrira un sous-programme qui permet d'afficher un arbre tel que présenté à la figure 7.

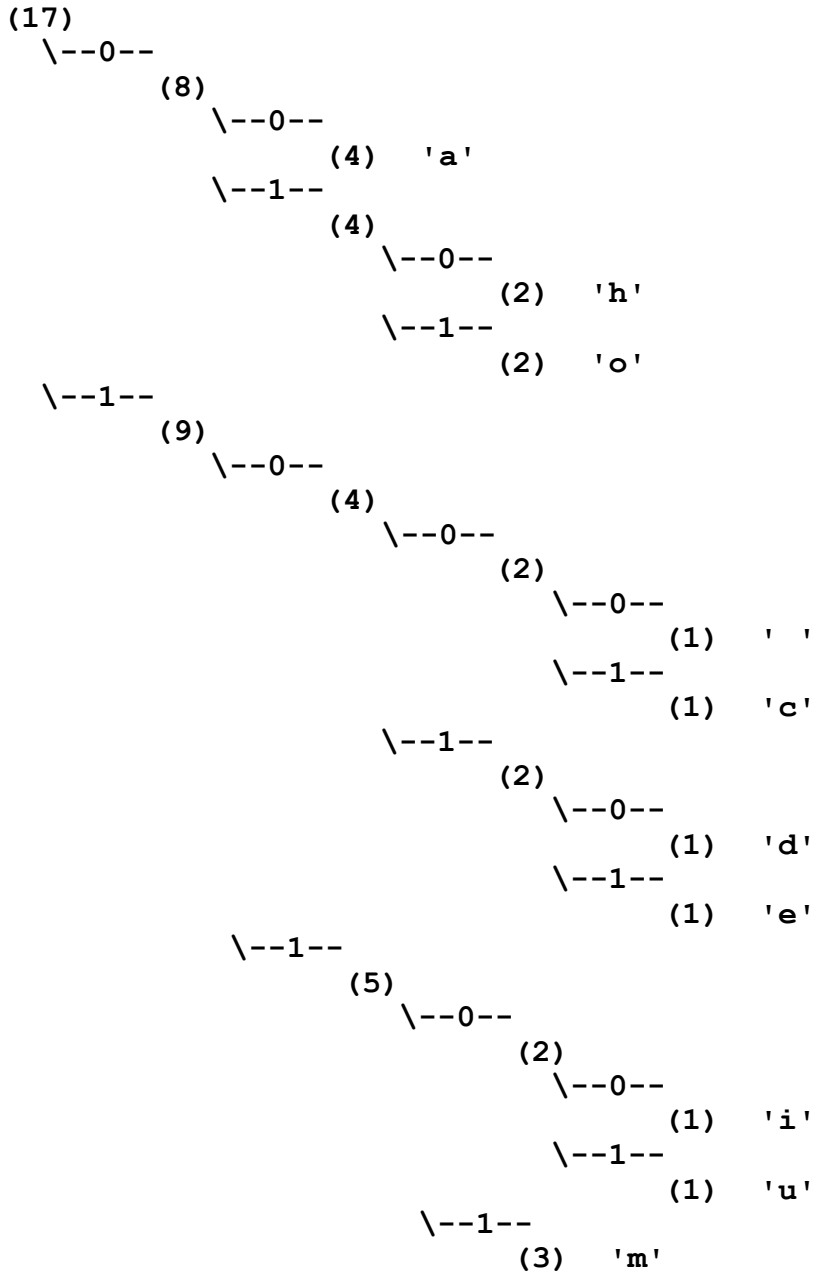


FIGURE 7 – Version textuelle de l'arbre de Huffman

Un autre sous-programme permettra d'afficher la table de Huffman tel que présentée figure 8 (les caractères sont affichés dans l'ordre lexicographique).

```
' ' -> 1000
'a' -> 00
'c' -> 1001
'd' -> 1010
'e' -> 1011
'h' -> 010
'i' -> 1100
'm' -> 111
'o' -> 011
'u' -> 1101
```

FIGURE 8 – Version textuelle de la table de Huffman

2. Contraintes

Le projet sera réalisé en trinôme en utilisant le langage C. Toutes les notions vues en cours, TD ou TP peuvent (doivent !) être utilisées.

3. Documents à rendre

Les documents à rendre sont :

- Les sources de votre projet, y compris les programmes de test. Pour chaque programme de test, il faut préciser ce qu'il permet de tester ;
- Le rapport qui doit comporter au moins :
    - un résumé qui décrit le contenu du rapport
    - une introduction qui présente le problème traité et le plan du document
    - un rappel succinct du sujet ;
    - l'architecture de l'application sous la forme d'un diagramme de classe ;
    - la présentation des principaux choix réalisés ;
    - la présentation des principaux algorithmes ;
    - l'explication de la manière dont le programme a été mis au point ;
    - la liste des difficultés rencontrées et les solutions adoptées en justifiant vos choix (en particulier quand vous avez envisagé plusieurs solutions) ;
    - une conclusion expliquant l'état d'avancement du projet et les perspectives d'amélioration / évolution ;

Remarque : Le rapport ne doit pas dépasser 10 pages, listing non compris. Il s'agit de la limite supérieure. Il n'est donc pas nécessaire de l'atteindre !

4. Principales dates

Voici les principales dates du projet :

- Mercredi 05 février 2020 : distribution du sujet
- Mercredi 19 février 2020, 18 h : remise du rapport (rapport.pdf) et des sources (sources.zip) par mail à l'adresse m.maachaoui@gmail.com