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 « jerome ermont ». Il est composé de 13 caractères. En utilisant le codage ASCII, ce texte utilise 13 octets (1 caractère est codé sur un octet).

Supposons maintenant la table de codage de la table 1.

Caractère	j	e	r	o	m	' ␣ '	n	t
Code	1110	01	100	101	110	1111	000	001

TABLE 1 – Table de Huffman pour le texte « jerome ermont »

Le texte est alors codé sur 5 octets :
1110.01.100.101.110.01.1111.01.100.110.101.000.001

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. déterminer le code de chaque caractère depuis 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.

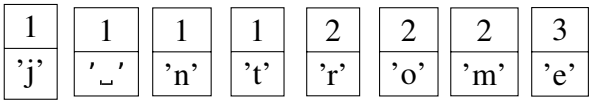
Par exemple, pour le texte « jerome ermont », on obtient :

Caractère	j	e	r	o	m	' ␣ '	n	t
Fréquence	1	3	2	2	2	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 ou égale à celle de la racine du sous-arbre droit. La figure 7 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 correspondant aux caractères 'j' et '␣'. La liste ainsi modifiée est donnée à la figure 1.

Les étapes de construction de l'arbre sont détaillées figures 1 à 6. Seules les feuilles de l'arbre de Huffman correspondent à des caractères du texte original. Pour les nœuds, seule la fréquence est utilisée. On a donc mis deux apostrophes sans caractère pour symboliser que cette valeur n'est pas significative.

L'algorithme se termine lorsqu'il ne reste plus qu'un seul arbre dans la liste (figure 7).

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

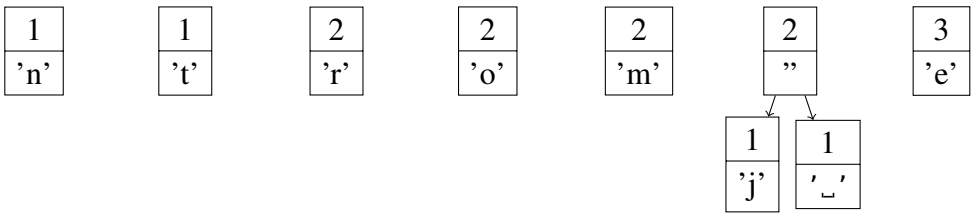


FIGURE 1 – Construction de l'arbre : étape 1

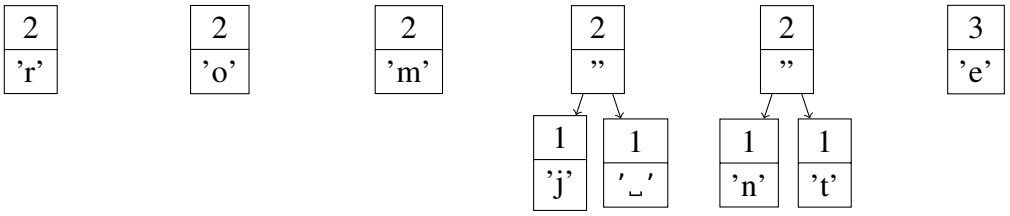


FIGURE 2 – Construction de l'arbre de Huffman : étape 2

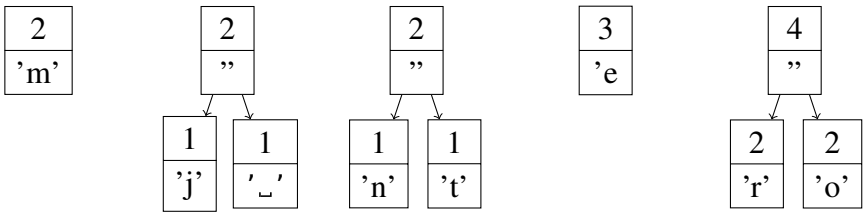


FIGURE 3 – Construction de l'arbre de Huffman : étape 3

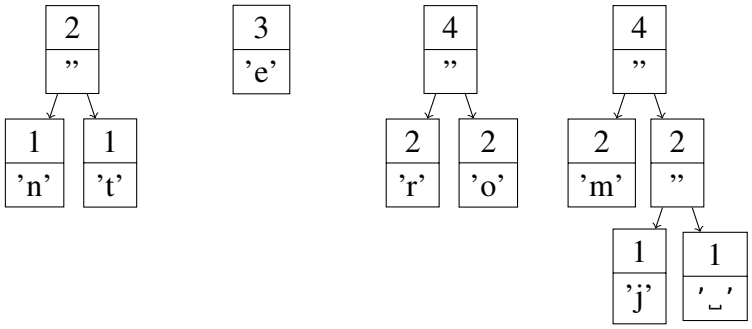


FIGURE 4 – Construction de l'arbre de Huffman : étape 4

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 7). La table de Huffman ainsi obtenue est celle donnée en table 1. Le caractère 'e', présent 3 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 décompressé : 13 sur l'exemple,
- un octet pour la taille de l'arbre (le nombre de caractères) : 8 sur l'exemple,
- chacun des caractères de l'arbre (parcours infixe, en profondeur) : 'n', 't', 'e', 'r', 'o', 'm', 'j' et ' ' 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 : 000101101100010110010 0101,

La suite du fichier contient le message encodé : 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`.

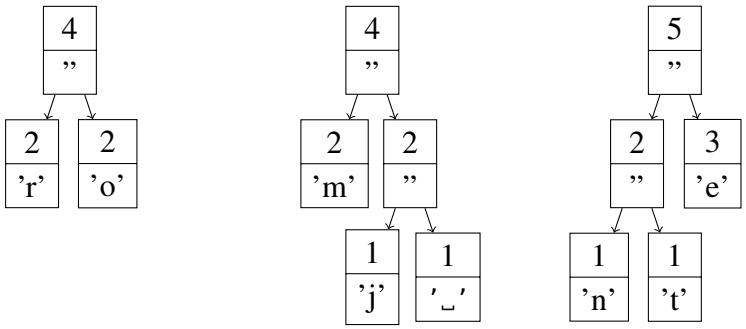


FIGURE 5 – Construction de l’arbre de Huffman : étape 5

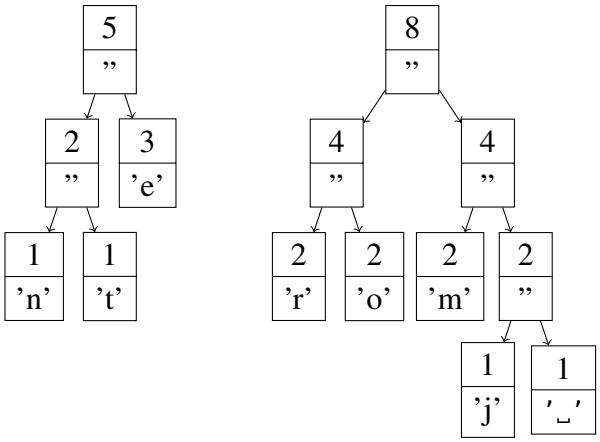


FIGURE 6 – Construction de l’arbre de Huffman : étape 6

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 8. Un autre sous-programme permettra d’afficher la table de Huffman tel que présentée figure 9 (les caractères sont affichés dans l’ordre lexicographique, en fait ASCII).

2 Contraintes

Le projet sera réalisé en binôme en utilisant le langage C. Les deux membres du binôme doivent faire partie du même groupe de TD.

Ils ne doivent pas déjà avoir fait partie d’un même groupe de projet.

Toutes les notions vues en cours, TD ou TP peuvent (et doivent !) être utilisées.

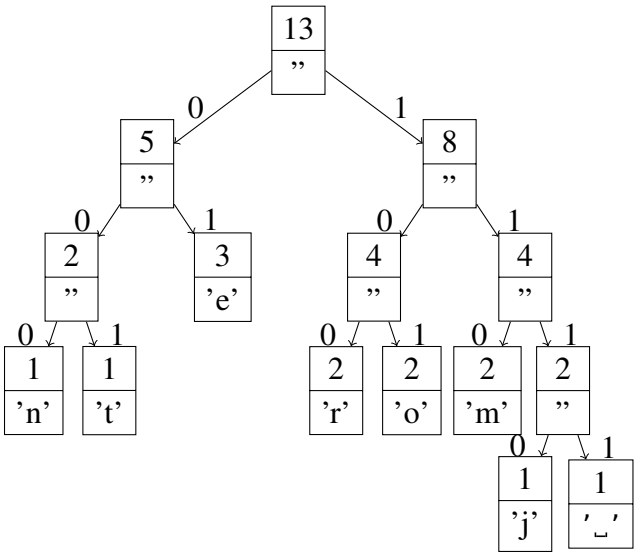


FIGURE 7 – Arbre de Huffman final

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 l'objectif et le contenu du rapport (10 lignes maxi),
  — une introduction qui présente le problème traité et le plan du document
  — l'architecture de l'application modules
  — la présentation des principaux choix réalisés
  — la présentation des principaux algorithmes et types de données
  — la démarche adoptée pour tester le programme
  — les difficultés rencontrées et les solutions adoptées en justifiant vos choix (en particulier quand vous avez envisagé plusieurs solutions)
  — une conclusion donnant un état d'avancement du projet et les perspectives d'amélioration / évolution

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

Attention : Le rapport est le point d'entrée sur les documents que vous rendez, en particulier les sources de votre programme. Il n'est donc pas utile d'y copier/coller votre programme. En revanche, il faut donner les informations qui permettront au lecteur de comprendre votre programme et les choix faits et ainsi savoir quelles parties du code pour avoir les détails.

```
(13)
  \--0--
        (5)
          \--0--
                (2)
                  \--0--
                        (1) 'n'
                  \--1--
                        (1) 't'
          \--1--
                (3) 'e'
  \--1--
        (8)
          \--0--
                (4)
                  \--0--
                        (2) 'r'
                  \--1--
                        (2) 'o'
          \--1--
                (4)
                  \--0--
                        (2) 'm'
                  \--1--
                        (2)
                          \--0--
                                (1) 'j'
                          \--1--
                                (1) ' '
```

FIGURE 8 – Version textuelle de l'arbre de Huffman

```
' ' -> 1111
'e' -> 01
'j' -> 1110
'm' -> 110
'n' -> 000
'o' -> 101
'r' -> 100
't' -> 001
```

FIGURE 9 – Version textuelle de la table de Huffman

4 Principales dates

Voici les **principales dates** du projet :

— mardi 5 janvier 2015, 10h-12h : distribution du sujet, compréhension du sujet, constitution des groupes de projet.

— vendredi 8 janvier 2015 : dépôt sur le SVN des principaux choix : structuration en modules, premiers niveaux de raffinages, principaux types de données. Ceci sera la première version de votre rapport (fichier rapport.pdf) à mettre dans le dossier `livrables` de votre dépôt SVN.

— mercredi 27 janvier 2015 : remise sur le SVN, dans le répertoire livrables, du rapport (rapport.pdf) et des sources (sources.tgz).

— jeudi 28 janvier 2015 : remise de la présentation oral (oral.pdf) dans le répertoire livrable du SVN.

— vendredi 29 janvier 2015, 8-12 h : oral et recette du projet

— les autres dates seront communiquées sur le site Web du module

Tournez SVP...

5 Conseils

Voici le résumé et les deux dernières parties [1] d'un rapport technique [2] rédigé en novembre 2004 par Marlène Beray, 3TR. En espérant que vous saurez tirer profit de l'expérience de vos prédécesseurs...

5.1 Abstract

During a computing project, technical issues often occur. In order to correct the errors, tests are performed on the whole program. Several recommendations should permit to avoid some errors which involve a waste of time on the program development and to finish the project in time.

5.2 Recommandations

The project was finished in time. However, some technical issues could have been avoided. In order to produce an efficient program, tests should be performed regularly. Before beginning to write the code, students should think about the points of issue established by the specifications and the tests to execute. Developing the program, students should check that there is no error in the writing code by compiling it. They could also test each function and each unit (group of functions) to avoid looking for errors in the whole program and wasting time. The report has major place in the project and should not be written during the last days. On the contrary, the report should evolve in its structure and its contents. Consequently, its writing should be done progressively, throughout the project.

5.3 Conclusion

In the engineer training, projects are crucial since they give students the opportunity to work as team members, to organise themselves and produce results within required times. They allow them to assimilate lessons more easily and practice. The technical issue met during the computing project was not serious and, once found, was rapidly corrected. This shows that insignificant errors could lead to critical situations. Consequently, it should be advantageous to perform tests regularly on the whole program. Moreover, the work concerning the different parts of the project should progress in parallel. Each assignment is specific and presents various issues, which permits to gain experience and become able to manage more impressive projects.

1. Ce rapport contient en fait trois parties supplémentaires.
2. Ce rapport a été rédigé dans le cadre des cours d'anglais sur un sujet choisi par l'étudiante.