Rapport du Projet compresseur de Huffman

HLIN303 - Systèmes d'exploitation

Rémi Cérès Jérémie Dautheribes

14 décembre 2016

1 Introduction

Dans le cadre de l'UE Systèmes d'exploitation (HLIN303) nous avons réalisé un compresseur sans perte utilisant l'algorithme de Huffman.

Cet algorithme se base sur le calcul de la fréquence de répartition de chaque caractère du fichier à compresser. Il génère ensuite des codes binaires pour chaque caractère, les caractères les plus fréquents ayant les codes les plus cours. Ces codes sont par la suite placés dans un nouveau fichier dit "compressé" qui aura dans la plupart des cas une taille inférieure à celle d'origine.

Nous allons commencer par expliquer nos choix techniques, puis les spécifications du programme, enfin nous répondrons aux différentes questions données dans le sujet.

2 Choix techniques

2.1 Découpage du code

Notre code est divisé en deux programmes séparés : le compresseur et le décompresseur. Chacune d'entre elles est elle-même divisée en plusieurs parties.

Compresseur :

defNoeud.h : Définit la structure `Noeud`. C'est un noeud d'un arbre binaire. Cette structure contient `pere,fg,fd,frequences` qui correspondent respectivement à l'indice du noeud père, à l'indice du fils gauche, à l'indice du fils droit et enfin à la fréquence d'apparition du caractère lié à ce noeud.

frequences.h : Contient la définition de la fonction calculant la fréquence d'apparition de chaque caractère du fichier en lecture.

constrcArbre.h : Contient la définition de la fonction qui construit l'arbre à partir du tableau de fréquences préalablement généré.

codeBin.h : Contient la définition de la fonction générant les codes binaires liés à chaque caractère lu dans le fichier en lecture. Il utilise pour cela l'arbre qui vient d'être créé.

generation.h : Contient la définition de la fonction qui va constitué le fichier compressé. L'entête est générée à partir de l'arbre et le corps est généré à partir des caractères lus et de leur code binaire préalablement obtenu.

Décompresseur :

defNoeud.h : Définit la structure `Noeud`.

Decompresseur.h : Contient la définition de la fonction de décompression.

2.2 Formalisme du fichier compressé

Nous avons défini notre propre format de fichier compressé. Nous avons inséré les informations nécessaires à la décompression du fichier tout en essayant de minimiser leur taille et d'éviter tout doublons.

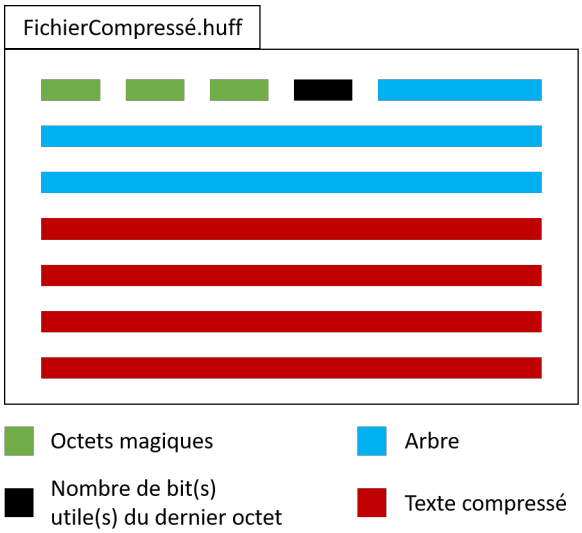

FIGURE 1 – Schèma du format du fichier compressé

Les "octets magiques" correspondent à une suite de trois octets qui sont uniquement présents dans les fichiers qui ont été compressés par notre programme. C'est en quelques sortes une clé d'identification. S'ils ne sont pas présents, alors le programme ne décompresse pas le fichier désiré.

Vient ensuite le nombre de bits utiles du dernier octet. En effet, cet octet final peut contenir des 0 inutiles générés lors de la compression. Cela nous per-

met de savoir quand s'arrêter lors de la décompression.

La suite et fin de l'entête contient l'arbre sous forme compressée. Nous avons choisi le format suivant pour le stocker : Nous parcourons notre arbre de manière récursive : si le noeud l'arbre comporte deux fils, on écrit un 1 et on rappelle l'algorithme sur le sous-arbre gauche puis sur le sous-arbre droit. Sinon, s'il s'agit d'une feuille, on écrit un 0 puis on écrit le code ASCII du caractère correspondant à cette feuille.
Nous avons choisi de stocker l'arbre plutôt que la table de caractères associés à leur code. En effet l'arbre est plus petit en terme de taille de stockage ainsi qu'en efficacité lors de la décompression (temps d'accès moins long).

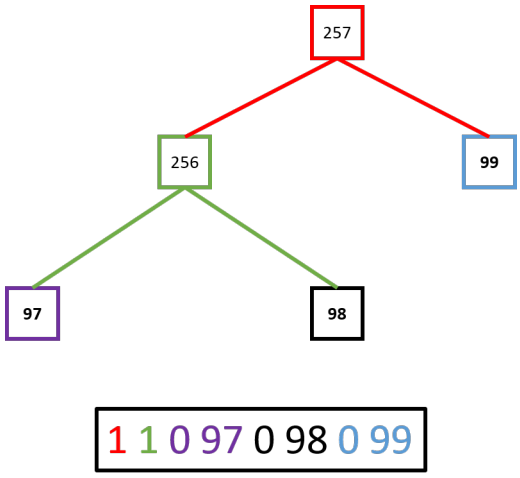

FIGURE 2 – Exemple de représentation compressée d'un arbre

Après l'entête, le corps du fichier contient une suite de codes binaires correspondants au texte compressé.

3 Spécifications des programmes

```
# Pour compresser :
huf [Fichier entrée] [Fichier de sortie]

# Pour décompresser (vers un fichier "résultat" ainsi
# que la sortie standard) :
dehuf [fichier entrée]
```

4 Questions / Réponses

Quel est le nombre maximum de caractères (char) différents ?
Le nombre maximum de caractères est 256.

Comment représenter l'arbre de Huffman ? Si l'arbre est implémenté avec des tableaux (fg, fd, parent), quels sont les indices des feuilles ? Quelle est la taille maximale de l'arbre (nombre de noeuds) ?
L'arbre de huffman est représenté par une structure possédant les variables *pere*,*fg*, *fd* et *frequences*.
Si l'arbre est implémenté avec des tableaux (fg, fd, parent), les indices des feuilles corresponde au code du carractère.
L'arbre peut avoir au maximum 511 noeuds.

Comment les caractères présents sont-ils codés dans l'arbre ?
Ils sont codsé par leur code ASCII.

Le préfixe du fichier compressé doit-il nécessairement contenir l'arbre ou les codes des caractères ou bien les deux (critère d'efficacité) ?
Le préfixe du fichier compressé doit contenir soit l'arbre soit les codes des caractères.
Stocker l'arbre est plus efficace en terme de taille de stockage ainsi qu'en efficacité lors de la décompression.

Quelle est la taille minimale de ce préfixe (expliquer chaque champ et sa longueur) ?
Le préfixe est constitué en trois parties : Tout d'abord 3 octet magiques (pas obligatoires) permettant d'identifier les fichiers pouvant être décompressés, puis d'un octet représentant le nombre de bits utiles dans le dernier octet et enfin l'arbre.

Si le dernier caractère écrit ne finit pas sur une frontière d'octet, comment le compléter ? Comment ne pas prendre les bits de complétion pour des bits de données ?
On le complète avec des 0 inutiles.
Lors de la décompression, on vérifie si on est sur le dernier octet, si c'est le cas on ne traite que les bits utiles. Le nombre de bits utiles a été récupéré dans l'entête.

Le décompresseur doit-il reconstituer l'arbre ? Comment ?
Oui, à partir de l'entête. Dans notre cas, l'arbre est reconstruit dans un tableau de *Noeuds*.