Projet: Compression/Décompression d'un texte par la méthode du codage de Huffman

Résumé

L'objectif de ce projet est de créer un programme permettant de compresser et de décompresser un texte en utilisant la méthode du codage de Huffman.
Contenu du rapport:
- Une présentation du problème
- La présentation des modules
- Choix réalisés
- Types de données
- Raffinages des programmes
- Démarches de test
- Difficultés rencontrées
- Conclusion

Introduction

Le principe du codage de Huffman est de se baser sur les occurrences des différents caractères au sein d'un texte et d'adapter en fonction la manière de le coder dans un fichier. Pour un caractère à forte occurence, il peut être de judicieux de le coder sur peu de bits, contrairement à un octet d'habitude. Comme ce caractère est très présent, la mémoire nécessaire pour coder l'ensemble du texte sera donc moins importante, et on réserve un encodage plus long aux caractères plus rares.
Le programme est donc basé sur la création d'une table de fréquence d'apparition de caractère puis puis sur un arbre de Huffman permettant la création d'un dictionnaire d'encodage écourté des caractères, qui sera créé pour la compression puis retrouvé pour la décompression du fichier.

Structure des modules:

hf_compress.c	hf_uncompress.c	types.h
hf_arbre.c	hf_uncompress.h	
hf_ecrire.c		
hf_compress.h		

Choix réalisés

Nous avons ici fait le choix de créer deux programmes: l'un compressant le texte et l'autre le décompressant. Félix s'est chargé de la partie compression et Thibault de la partie décompression.
Le code du programme de compression est divi en 3 fichiers : le programmes principal, un fichier avec des sous-programmes propre à la création de la table de fréquence et à celle de l'abri, et un propre à l'écriture des données dans le fichier compressé.

Déclaration des types:

Enregistrement freq_table:
tableau d'entiers de 256 cases
Enregistrement Noeud:
fréquence type entier
data type caractère
left type pointeur sur type Noeud
right type pointeur sur type Noeud
Enregistrement Dict:
codage type entier
data type caractère

Raffinages hf_compress:

R0:

Compresser un fichier par la méthode de Huffman

R1:

1-Créer la table de fréquence du fichier à compresser
2-Construire l'arbre de Huffman
3-Créer le dictionnaire
4-Générer le fichier compressé

R2:

R2-1:

Ouvrir le fichier
Tant que tous les caractères ne sont pas lus:
| Lire le caractère suivant
| Convertir le caractère en entier
| Incrémenter de 1 la table de fréquence a l'indice correspondant
Fin tant que
Fermer le fichier

R2-2:

Créer un tableau dynamique de Noeuds pour chaque caractère présent au moins 1 fois dans la table de fréquence sans sous-arbres (descendants).
Tant que le tableau compte plus d'un Noeud:
| Trier le tableau par «sum» croissante
| Regrouper les deux noeuds les plus à gauche en un arbre de
| fréquence la somme à deux feuilles classées par fréquences
| croissantes.
Fin Tant Que

R2-3:

Dans l'arbre binaire de Huffman associer respectivement aux branches «left» et «right» les codes 0 et 1.
Le code de chaque caractère est le nombre binaire obtenue en parcourant l'arbre depuis son sommet jusqu'au caractère.
Le dictionnaire est alors l'ensemble des codes ainsi obtenus.

R2-3:

Créer un nouveau fichier
Ouvrir le nouveau fichier
Ecrire sur 4 octets la taille du texte décompressé
Ecrire sur 1 octet la taille de l'arbre

Ecrire l'ensemble des caractères différents utilisés dans le fichier à compresser dans leur ordre d'apparition dans l'arbre de Huffman (en partant d'en bas a gauche)

Ecrire le parcours infixe de l'arbre:
Parcourir à partir du sommet de l'arbre et on commence par indiquer le chemin du caractère le plus a gauche: 0 pour descendre, 1 pour monter.
Parcourir l'arbre de gauche a droite en partant des positions Feuilles successives.
Transcrire le texte compressé.

Raffinages hf_uncompress:

R0:
Décompresser un fichier compressé par la méthode de Huffman
R1:
1-Reconstruire l'arbre de Huffman grâce au parcours infixe
2-Retrouver le dictionnaire
3-Retranscrire le fichier décompressé
R2:
R2-1:
Lire la taille du texte décompressé
Lire la taille de l'arbre sur le 5ème octet
Lire les caractères utilisés (du nombre de la taille de l'arbre)
Retrouver la longueur du parcours infixe (nombre de suites de 1)

Difficultés rencontrées

Nous n'avons pas réussi à lire correctement bit-à-bit dans le fichier compressé et donc après les caractères utilisés. Des erreurs de segmentations empêchent donc la reconstruction de l'arbre et la décompression du fichier.

Conclusion

Il reste à terminer la partie décompression du fichier.