Ce travail est à faire en binôme. Le compte rendu (un par binôme) est à déposer sur la page Moodle du cours en respectant les consignes.

Objectifs

Cette Apnée porte sur le problème du codage d'un texte à l'aide de l'algorithme de Huffman. Ce code consiste à associer à chaque caractère du texte une séquence de bits dont la longueur est généralement inversement proportionnelle à la fréquence d'apparition de ce caractère, ce qui permet d'obtenir un fichier codé de taille réduite.

Le travail demandé est de compléter les deux programmes huff_encode.c et huff_decode.c puis de valider votre programme de compression et d'en évaluer l'efficacité dans différentes situations.

Ressources fournies

- huff_encode.c et huff_decode.c, programmes à compléter
- Un module d'arbres binaires : arbrebin.c et arbrebin.h
- Un module de files d'attentes avec priorités : fap.c et fap.h (**attention :** dans cette implémentation un entier petit signifie une priorité forte)
- Un module de lecture/écriture bit à bit : bfile.c et bfile.h
- Un Makefile
- Deux fichiers textes pour les expérimentations initiales : candide.txt et gargantua.txt

1. Encodage

Exercice 0. Construction de la table d'occurrences

Complétez la procédure ConstruireTableOcc. Cette procédure doit permettre de construire le tableau TableOcc, tel que TableOcc[i] est le nombre d'occurrences du caractère de code i dans le fichier d'entrée.

Exercice 1. Construction de l'arbre de Huffman

Complétez maintenant les fonctions InitHuffman et ConstruireArbre, afin de construire un arbre de Huffman à partir de TableOcc. L'algorithme a été vu en cours.

Indications

- Cet algorithme nécessite une file à priorités. Vous pouvez récupérer celle fournie.
- Pour mémoriser l'arbre de Huffman, utilisez le module d'arbres binaires fourni.

Testez cette étape sur de petits exemples. Essayez de construire à la main l'arbre de Huffman pour quelques exemples, comparez avec l'arbre obtenu à l'exécution de votre programme.

Exercice 2. Construction de la table des codes

L'arbre de Huffman obtenu à l'étape précédente fournit implicitement un code de Huffman (une séquence de bits) pour chaque caractère du code Ascii. Complétez maintenant la procédure ConstruireCode. Cette procédure va construire la table HuffmanCode, contenant le code de Huffman correspondant à l'arbre de Huffman passé en paramètre :

- HuffmanCode[i] contiendra le code du caractère de code Ascii i ;
- HuffmanCode[i].lg est la longueur (en nombre de bits) de ce code ;
- HuffmanCode[i].code est un tableau d'entiers, contenant ce code.

Exercice 3. Encodage

Complétez enfin la procédure Encoder, qui utilisera la table HuffmanCode afin de produire, dans le fichier fichier_encode, le code de Huffman du fichier d'entrée fichier.

Indications

- il ne faut pas oublier d'écrire l'arbre de Huffman en début de fichier, afin de permettre son décodage ;
- pour lire le fichier à coder, vous pouvez vous inspirer du schéma donné pour ConstruireTableOcc ;
- vous pouvez vous servir du module d'écriture bit à bit.

Expérimentation

huff_encode.c est maintenant complet : expérimentez l'encodage sur des exemples de fichiers : le source de votre programme, le texte de Candide, de Gargantua, la librairie Xlib.h, des fichiers créés par vous pour l'occasion, etc. Comment pouvez-vous vérifier que votre programme est correct à ce stade ?

?

2. Décodage

Un squelette de décodeur se trouve dans huff_decode.c. Ce programme ouvre le fichier donné en premier argument de la ligne de commande, et commence par y lire un arbre de Huffman.

Exercice 4. Décodage à partir d'un arbre de Huffman

Complétez la procédure Decoder, permettant de lire le fichier d'entrée, et utilisant l'arbre de Huffman pour le décoder.

Indication

Vous pouvez ici encore vous servir du module de lecture bit à bit.

Exercice 5. Tests

Testez conjointement l'encodage et le décodage, sur de petits exemples, puis sur des textes plus conséquents, de façon à vérifier si votre programme fonctionne comme attendu. Là encore, n'hésitez pas à varier les tests et à créer des fichiers pour observer ce qui se passe.

Essayez également de déterminer sur quels genres de fichiers la compression est efficace ou non, et si possible proposez des explications selon les cas.

Compte-rendu

Votre compte-rendu devra comprendre :

- les fichiers huff_encode.c et huff_decode.c complétés ;
- un rapport contenant
  - si vous n'avez pas réussi à terminer, un récapitulatif de ce qui a été fait, des problèmes rencontrés ;
  - les tests commentés (nature des fichiers testés, performance de la compression) ;
  - un commentaire court, indiquant de quelle manière les modules fournis ont été utilisés (structure de l'arbre de Huffman, etc.) ;
  - une conclusion concernant la correction et l'efficacité de votre programme, au regard des tests réalisés.

Modifié le: jeudi 14 novembre 2024, 08:47

?