CIS 26B
Advanced C Programming
Honors Project

Honors Project: A variation of the Huffman encoding algorithm

100 Points

Write a program that finds the frequency of each character in a text and builds the corresponding Huffman tree. After the tree has been built, demonstrate encoding and decoding text using the Huffman tree in a menu-driven program as explained below.

Building the Huffman Tree

Start with one binary tree of one node for each character; then build new trees by combining two existing trees at a time. Eventually we will obtain only one tree, the Huffman tree. Each leaf in a tree has as data the character and its weight. The other nodes have just the accumulated weight (the character data member is not used).

Use a singly-linked list to keep track of the binary trees. The list is sorted in ascending order by frequency. Each node in the list has as data a pointer to a root of a binary tree.

Combining two existing trees:
Merge the two nodes with minimum weight from the linked list into one; the new linked list is to contain a pointer to a new tree node; the weight of the new tree node is to be the sum of the original weights; its left child is one of the two nodes with the minimum weight, and its right child is the other one. The process continues until the linked list is reduced to only one node that is containing a pointer to the Huffman tree.

Write a menu-driven program with the following options:

- Print the character weights (frequency)
- Print Tree (Right-Root-Left – indented format)
- Print the Huffman codes for all of the characters in the list
- Enter one character – print its Huffman code
- Enter a word – print its ASCII binary representation (as strings of 0s and 1s), then its Huffman code (as strings of 0s and 1s)
- Enter an encoded word, decode it, then display it to the screen
- Enter the name of a text file, encode it, and save it to another file.
- Enter the name of the encoded file, decode it, then save it to another file.
- Quit

Test your program using the following text (**`in.txt`**).

In computer science and information theory, a Huffman code is a particular type of optimal prefix code that is commonly used for lossless data compression. The process of finding and/or using such a code proceeds by means of Huffman coding, an algorithm developed by David A. Huffman while he was a student at MIT, and published in the 1952 paper "A Method for the Construction of Minimum-Redundancy Codes".

The output from Huffman's algorithm can be viewed as a variable-length code table for encoding a source symbol (such as a character in a file). The algorithm derives this table from the estimated probability or frequency of occurrence (weight) for each possible value of the source symbol. As in other entropy encoding methods, more common symbols are generally represented using fewer bits than less common symbols. Huffman's method can be efficiently implemented, finding a code in time linear to the number of input weights if these weights are sorted. However, although optimal among methods encoding symbols separately, Huffman coding is not always optimal among all compression methods.

Source: https://en.wikipedia.org/wiki/Huffman_coding

Note

This project is a comprehensive project based on several topics covered in this class:

1. Sorted Singly Linked-Lists
2. Hash Tables
3. Binary Trees
4. Bit Manipulation
5. Binary Files

Project Requirements / Instructions

1. The honors project is an individual project, but it can be changed to a group project with the instructor's approval.
2. If you are already working on a project that meets the needs of an honors project, you may submit this project instead of the assigned project, with the instructor's approval.
3. Create a (pdf or ppt) presentation file (4 to 6 pages or slides). Include a title page, author(s), purpose, design (could be explain in words and / or structure charts and data structure diagrams), include sample outputs, and a conclusion.
4. Create a short version of this document (1 page, like a poster) to share it with your classmates.
5. Write a short report (one page) about your project (similar to the regular homework assignments reports)
6. Upload a .zip file that with the following content:
    a. Report (one per student)
    b. Source, header, input, output files (one per group)
    c. Presentation file(s) (one per group)