ECE368 Project 2

Milestone 1

Xiao Xiao

Overall

The main purpose of this project is to write two programs called huff.c and unhuff.c. The program huff.c is desired to compress a regular input file using Huffman coding and write out the compressed output to a new file with ".huff" appended. For example, if "example.txt" is the name of an input file, the output file should be "example.txt.huff". The other program unhuff.c is used to decompress the input file and write out the output to a file with ".unhuff" appended. For example, if "example.txt.huff" is the name of an input file, the output file should be "example.txt.huff.unhuff".

huff.c Program

First of all, write some of the basic primitives of linked list, tree and queue. Functions like enqueue, dequeue, construct node, construct tree, delete node, insert node will be implemented as individual functions for further using.

Building the Table for Compression

In order to use Huffman coding method, the first thing is to build a table of per-character encodings.

1. Use a linked list to implement a queue that will store each unique character in the input file. Each node will contain a tree node, an integer value, and a character in the type definition. Count the number of times every character occurs using linked list traversal to search repeated elements. Afterwards, each node will have a unique character and a weight equal to the number of times the character occurs.
2. Basically use the Huffman algorithm to build a single tree. If possible, sort the linked list stably. Otherwise it is necessary to do a linked list traversal for combing two trees. Start with the elements with the lowest weights in the linked list and combine the two nodes to a new tree with combined weights. Then repeat the process to find the lowest weights elements in the linked list and combine them until there is only one node in the linked list.

Compress the file

1. Do an in order traversal of the Huffman tree and record the steps, which indicates whether it is a 0 or 1 for the bit sequence. Read the file again and process one character at a time. Compare the character with the character in the tree node during the tree traversal. Once the character matched, and write the bit sequence recorded into the output file. After recording, go back to the root and repeat the traversal process.
2. The other things are the header information and EOF. I will store the tree at the beginning by using pre order traversal of the tree. For each non-leaf node, I put character 0. For each leaf node, I put 1 followed by its character value. For EOP, I may use a pseudo-EOF character to indicate the end of file.

unhuff.c Program

The same as huff.c, write some of the basic primitives for tree. Functions like construct node, construct tree, delete node, insert node will be implemented as individual functions for further using.

Building Huffman tree

Read the information in header to build the same Huffman tree in the huff.c program. Read a character each time and see whether it is 0 or 1. If it is 1, which indicating non-leaf node, read and store the character value in the tree node. If it is 0, which indicating a leaf node, put 0 in the tree node. Repeat the process until the end of the file.

Decompress the file

Read the bit one at a time from the input file after the header information. Follow the input bit to traverse the Huffman coding tree. If a leaf is reached, store the character into the output file unless the character is pseudo-EOF, then decompression done.