Huffman Class Programmer's Manual

Programmer's Manual
Huffman Class

1. Problem Description

The Huffman class consists of a struct which holds the data for each character as well as pointers to it's children and parent in the huffman tree. The struct also contains operator overloads for < to allow sorting and << to allow printing. The class contains the functions which encode and decode a message according to the huffman tree constructed from encoding as well as allowing the user to print the tree and the code values for each character.

2.1 Class Huffman

Private Data Members:

bool populated	flag determining whether the tree has data in it
vector<huffNode> nodes	a vector of huffNodes representing the tree
vector<char> input	the input string which is then sorted
string inString	the original input string
string fullCode	the code generated from decoding

Private Member Functions:

makeTree	constructs the huffman tree
readFile	reads in the input file
countChars	counts the characters and builds the huffNodes for the tree

Public Member functions:

Huffman	constructor for a Huffman object
encode	builds a huffman tree from the input file and encodes the input string
decode	decodes a message encoded from the corresponding huffman tree
printTree	prints each node with corresponding children and parent nodes
printTable	prints the table of coded values for each character

2.2 Struct huffNode

Data Members:

string name	name of character in the node
int freq	frequency of the character
huffNode* left	left child of the node
huffNode* right	right child of the node
huffNode* parent	parent of the node
string code	code for each node

Operator Overloads:

<	allows for sorting by frequency
<<	print the name, frequency, and code

3. High Level Program Solution

Huffman
- set populated to false

readFile
- if populated is true, reset the values in the tree
- read in input file
- save copy of original input
- sort the input by character frequency
- set populated to true

countChars
- loop from a to z and create huffNodes for each character
- set each huffNode's values to 0 or null
- count the number of each character
- sort the vector by the frequency

makeTree
- construct the inner nodes of the tree
  - name is T plus an increasing value
  - set frequency to the added frequency of the children
  - initialize the inner nodes values to 0 or null
  - push inner node into vector
  - sort the vector
- assign all of the pointers of each inner node to it's corresponding left and right child
- assign all of the pointers of each node to it's corresponding parent

encode
- use readFile to get the input file
- use countChars to count the characters and create the nodes
- use makeTree to construct the huffman tree
- get the file to encode
- read the input string from the file and find the code in the tree

decode
- get the file to be decoded
- set the code to empty to begin
- read the code and look through the tree
- if the code is a 0, go left
- if the code is a 1, go right
- append the character code found to the final code
- repeat until the input code has been fully decoded

printTree
- iterate through the vector of nodes

print out the name of the node
if the node has a left child, print the name
if the node has a right child, print the name
if the node has a parent, print the name
print the code for each node

printTable

open file to save table to
iterate through the vector of nodes
if the node is a leaf node, meaning it is a character, print the name and the code
write each node to the output file