CMPS 340 Fall 2016
HW #8: Huffman Coding
Due: 1pm, Tuesday, November 22

Suppose that we have a file that is to be compressed. Suppose, further, that we decide to view it as being composed of 4-bit blocks (half-bytes), interpreting each such block as being an occurrence of a symbol. Under this interpretation, there are sixteen symbols in the *source alphabet*, one for each of the distinct bit strings of length four. Let Σ be the name of this alphabet. The frequency with which each member of Σ occurs in the file is given in the following table. (For convenience, we refer to the sixteen symbols as a_0, a_1, ..., a_{15}, where a_k corresponds to the binary numeral of length four that represents k.)

Character	Frequency	Character	Frequency
a_0 (0000)	8%	a_8 (1000)	1%
a_1 (0001)	3%	a_9 (1001)	0%
a_2 (0010)	5%	a_{10} (1010)	0%
a_3 (0011)	0%	a_{11} (1011)	1%
a_4 (0100)	9%	a_{12} (1100)	15%
a_5 (0101)	21%	a_{13} (1101)	6%
a_6 (0110)	7%	a_{14} (1110)	0%
a_7 (0111)	2%	a_{15} (1111)	22%

(a) Construct a Huffman tree corresponding to this table of frequencies. Label each leaf with the symbol a_i to which it corresponds. Only symbols with non-zero frequency should be included.

Note: In order to ensure that the correct answer to part **(b)** below is unique, do not label the edges of the tree until you have constructed it completely. At that point, iterate through the symbols a_0, a_1, ..., a_{15} and, for each symbol a_k, do this:

If a_k has a non-zero frequency, find its corresponding leaf node. Beginning there, follow edges toward the root. If an edge is unlabled, label it with 0 and label the "sibling edge" emanating from the same parent node with 1. Once you get to an already-labeled edge, stop.

Now redraw the tree so that all edges labeled 0 (respectively, 1) go to a parent node's left (respectively, right) child. **End of note.**

(b) Show the codeword table (i.e., the mapping from Σ to binary codewords) implied by your answer to (a).

(c) Show the bit string that would be used to encode the codeword table, in accord with the method discussed in lecture. Specifically, that would be the bit string representing the Huffman Tree (recall the recursive algorithm that does a preorder traversal of the tree, emitting 0 for every interior node and 1 for every leaf) followed by the "native codes" of the symbols corresponding to the leaves of the tree, going from left to right (under the assumption that edges to left children are labeled 0 and edges to right children are labeled 1). (Because it is understood that all the native codes are of length four, they need not be preceded by length indicators.)

Annotate the bit string so that it is clear where the boundaries are between adjacent elements within it.

(d) Compute the ratio between the lengths of the compressed version of the file and the original file. Assume that the length of the original file is $200,000$ bytes (which is $1,600,000$ bits). Show your work. Keep in mind that the number of bits used, on average, for encoding a symbol in the compressed version of the file is the sum

$$\sum_{x \in \Sigma \ \wedge \ freq(x) \neq 0} len(x) \cdot freq(x)$$

where $len(x)$ is the length (in bits) of the codeword for x and $freq(x)$ is the frequency with which x occurs in the original file.

Recall that the compressed version of the file contains the metadata that is the subject of part (c) above.

(e) Now suppose that we had assigned codewords to the symbols in Σ in accord with **canonical** Huffman coding. Recall that a canonical Huffman tree is one in which the depths of the leaves are in descending order as you go from left to right across the fringe of the tree.

Follow the rule that if a_i and a_j have codewords of the same length and $i < j$, then the codeword assigned to a_i should be numerically/lexicographically smaller than the codeword assigned to a_j.

Show the canonical mapping from Σ to codewords.