COMP 210 - Project 1 Lab Worksheet

Spring 2016

Due by the next class meeting

1. Table 1 provides a variable-length, prefix-free code for a five letter alphabet.

letter	a	b	c	d	e
code	010	00	10	011	11

Table 1: A five letter alphabet with variable length codes

For this code you need to:

- Draw the binary tree that represents this code.
- Encode the message *adedbbc* with the code.
- Decode the message 0110101000 with this code.

2. The following set of problems deal with using trees to develop codes for an 8 letter alphabet consisting of a through h.

   (a) Construct a fixed length, binary code for this alphabet that uses the least number of bits per possible and draw the binary tree that represents that code. As a "bonus" see if you can figure out how many possible fixed length codes with this same length you could have for this language.

(b) In table 2 below you're given a relative frequency for each letter. Use these to construct a Huffman code for this language. Try to place trees of smaller value in the left subtree and when choosing between two trees of equal value choose the smaller (in size) tree to go left or if both trees are of size one, then choose the alphabetically first tree to go left. If both of the trees being combined are the same size and value then choose the one that has been in the forest/collection longest to go on the left. In theory, following these guides will ensure that we all get the same tree.

letter	a	b	c	d	e	f	g	h
frequency	.184	.105	.053	.184	.053	.105	.079	.237

Table 2: An eight letter alphabet with relative frequencies

(c) Re-draw your Huffman tree but label each by enumerating the nodes. Start with one at the root and then count left to right and top to bottom. Nodes with a depth of 1 should be 2 and 3, then depth 2 should get 4, 5, 6, etc. Once your tree is drawn, carry out a preorder, inorder, and postorder traversal of the resultant tree and list the order in which the nodes are visited for each traversal.