Mark Barnes
Zehua Liu
PA 3
14 February 2017

Checkpoint PDF

1 & 2) We ran the checkpoint1.txt and checkpoint2.txt files and these were the outputs that we had when we did the compression of the file:

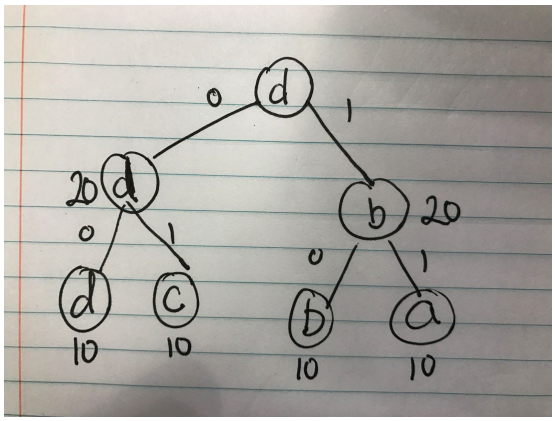
checkpoint1.txt:
This is what the header had -
Lines 98, 99, 100, and 101 had the value of 10. This indicates that the frequencies of ASCII values of “line # - 1” (‘a’, ‘b’, ‘c’, and ‘d’ in this case) were all 10. The rest of the header consisted of 0’s.

This is the huffman compression -
11100100111001001110010011100100111001001110010011100100111001001110010011100100

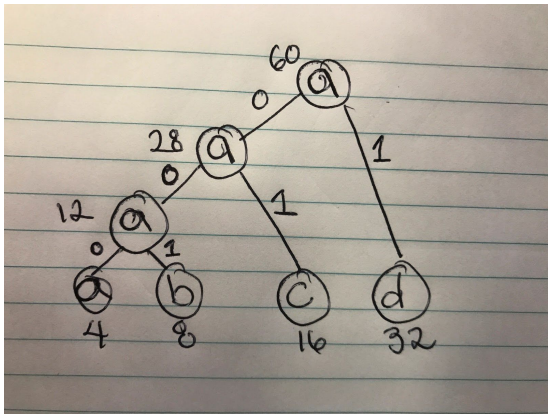
checkpoint2.txt:
This is what the header had-
Lines 98, 99, 100, and 101 had the respective values of 4, 8, 16, and 32. This indicates that the frequency of ‘a’ was 4, ‘b’ was 8, ‘c’ was 16, and ‘d’ was 32. The rest of header consisted of 0’s.

This is the huffman compression -
00000000100100100101010101010101011111111111111111111111111111111010101010101010100100100100100100000000

3) Manual Construction of Huffman Tree:
checkpoint1.txt



checkpoint2.txt



Description of checkpoint1.txt and checkpoint2.txt construction:
The tree is constructed so that the nodes with the lower frequencies are left nodes. When nodes have the same frequency, we compare alphabetically with ASCII value, and the larger ASCII value node is the left node. The parent node takes the symbol of the left node, and its frequency is the sum of its right and left nodes.

How to find coded character for each byte:
Start from a leaf, and work your way up the tree, checking if node is left or right node. If the node is the left then store a ‘0’ in the stack, and if it is a right node, then store a ‘1’ on the stack. Once the node has reached the root (the parent of root is NULL), then we are done retrieving the coded value for the character, so we top() and pop() from the stack in order to retrieve the correct order for descending the tree to finding that leaf.

How to find the coded word:
Use the key for each character -- that was developed from traversing down the Huffman Tree to each leaf node -- to translate the word into its coded ‘0’ and ‘1’ equivalent.

4) The manually encoded values matched our compressor’s output. We did not have an issue with encoding or decoding checkpoint1.txt or checkpoint2.txt.