Adrian Cordova y Quiroz A12010305 Jonathan Chiu (A12113428)

CSE 100 Alvarado

Feb 14 2017

PA3 Checkpoint Report

1. Run your compressor using the provided input files: **checkpoint1.txt and checkpoint2.txt**.
2. Record the encoded output in Checkpoint.pdf

   Command line: ./compress checkpoint1.txt encode.txt

   In encode.txt what is outputted is 0 on a new line, up until lines 98, 99, 100, and 101 there is a 10 printed on each of those lines, and on the 257th line is:
   111001001110010011100100111001001110010011100100111001001110010011100100
   11100100

   Each line is a frequency so that means on line 98, there are 10 'a's since the ascii value 97 is 'a' so for each line
   Line 98: 10 … this means there are 10 'a's
   Line 99: 10 … this means there are 10 'b's
   Line 100: 10 … this means there are 10 'c's
   Line 101: 10 … this means there are 10 'd's

   And the encoding is that huge binary number on the 257th line.

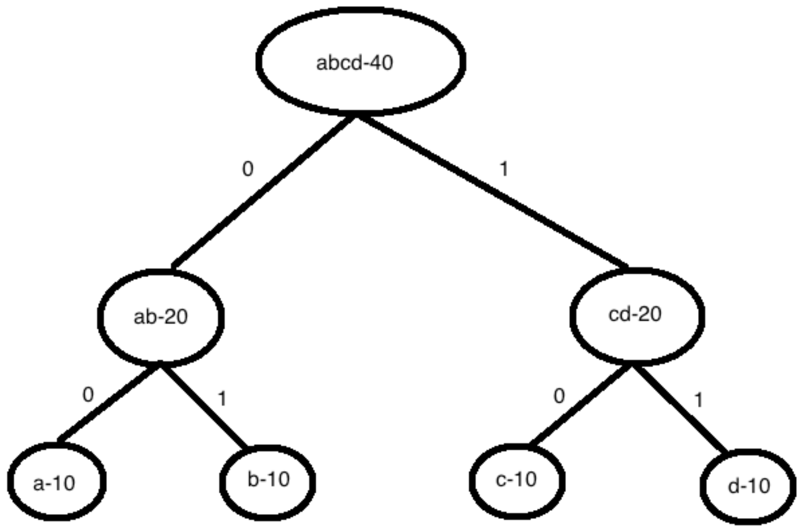
   Command line: ./compress checkpoint2.txt encode2.txt

   In encode2.txt what is outputted is 0 on a new line, up until lines 98, 99, 100, and 101 there is a 4, 8, 16, 32 respectively printed on those lines, and on the 257th line is:
   000000001001001001010101010101010111111111111111111111111111111111010101
   0101010101001001001001000000

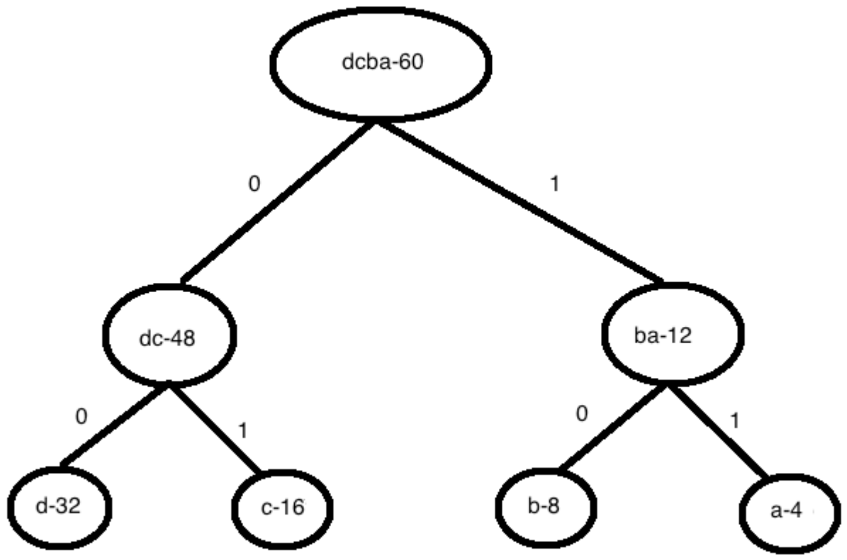
   Each line is a frequency so that means on line 98, there are 4 'a's since the ascii value 97 is 'a' so for each line
   Line 98: 4 … this means there are 4 'a's
   Line 99: 8 … this means there are 8 'b's
   Line 100: 16 … this means there are 16 'c's
   Line 101: 32 … this means there are 32 'd's

3. Use the **same input** to manually construct a Huffman coding tree. Draw the Huffman coding tree, describe how you build the tree and how you find the code word for each byte in Checkpoint.pdf.

   Checkpoint1 Huffman Tree:



   Checkpoint2 Huffman Tree:



To build the tree you write down all the characters and their frequencies, and pair the two smallest together to make the leaf node, then pair the next two to make their own leaf node and add the frequencies. You keep going until you have used all the characters and gotten to the root. To find the code for each character you just traverse the tree from the root to the node and connect the binary digit, so for a in checkpoint1 it is 00, b is 01, c is 10, and d is 11. For checkpoint 2 d is 00, c is 01, b is 10, a is 11.

The encoding for checkpoint1 in our program is
11100100111001001110010011100100111001001110010011100100111001001110010011100100
11100100
But for our handwritten Huffman code, we get
10101010101010101010101010101010101010101010101010101010101010101010101010101010
10101010
Both are the same number of bytes but different ordering because the order chosen for which character goes where on the bottom of the tree is arbitrary.

The encoding for checkpoint2 in our program is
00000000100100100101010101010101010111111111111111111111111111111111111111010101
010101010101001001001001000000
But for our handwritten Huffman code, we get
11111010101001010101010101010100000000000000000000000000000000000000000000000000
00000000000000000000000000010101010101010101101010101111
Both are the same number of bytes but different ordering because the order chosen for which character goes where on the bottom of the tree is arbitrary.