Checkpoint Writeup

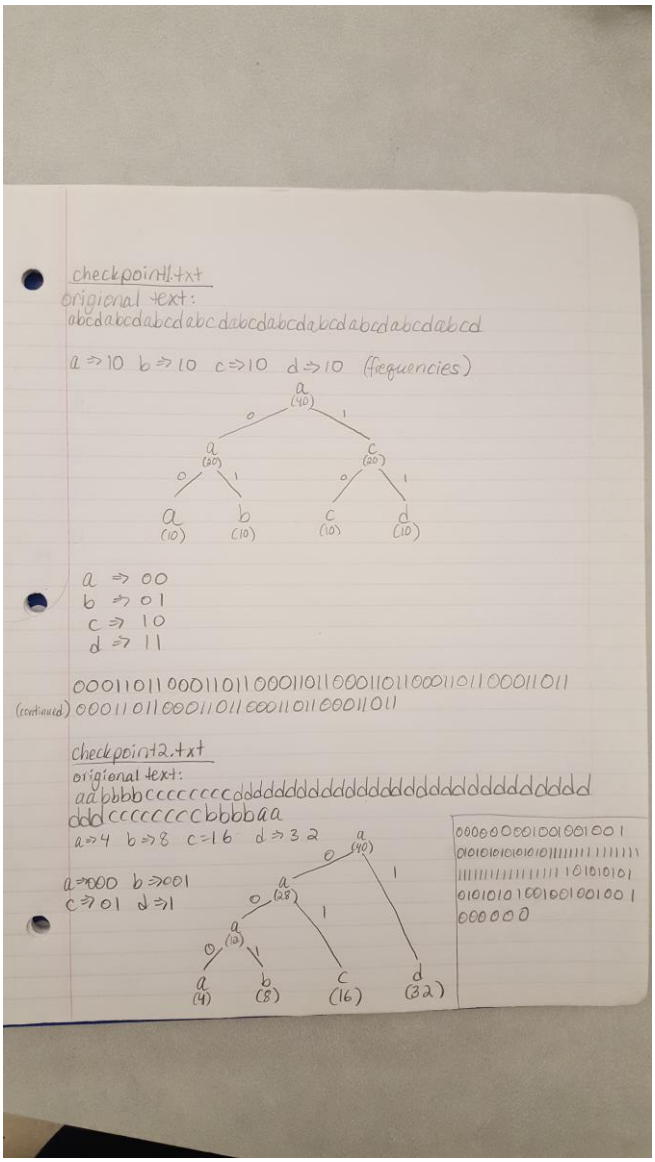
checkpoint 1

original = abcdabcdabcdabcdabcdabcdabcdabcdabcdabcd
0001101100011011000110110001101100011011000110110001101100011011000110110001101100011011
newfile = abcdabcdabcdabcdabcdabcdabcdabcdabcdabcd

checkpoint 2:

original = aabbbbccccccccddddddddddddddddddddddddddddddddddddddddcccccccbbbbaa
encode
= 0000000010010010010101010101010101111111111111111111111111111111111101010101010101
01001001001001000000

newfile = aabbbbccccccccddddddddddddddddddddddddddddddddddddddddcccccccbbbbaa



(the final frequency for checkpoint2.txt is supposed to be 60, not 40.)

The first step that I did was to count how many times each character appeared in the checkpoint1.txt file. The following steps were also used for the checkpoint.txt file. When I got the frequencies of each character as shown in the previous picture, I added up the frequencies from smallest frequency to largest frequency. In the case of checkpoint1.txt, all of the frequencies were the same, so I added “a and b” first, then “c and d”, and then the combinations of “a and b” and “c and d.” Then, for each left child of the parent node, we assigned it a 0. For each right child of the parent node, we assigned it a 1. Then, we started from the root node and worked our way down the tree until we found each symbol. Once we encoded each symbol, we looked at the string from the original text file and began to encode the text one symbol at a time using the prefix codes that were made for each .txt file. We compared our encoded by hand output to our program’s output for the same input. Both the checkpoint1.txt and checkpoint2.txt outputs are all the same when compared to our by-hand output.