Partner1:
Name: Jiaqi Fan
PID: A12584051
Partner2:
Name: Xuanru Zhu
PID: A91064234
Date: 2/15/2017
PA3

# CheckPoint Report

## Checkpoint1.txt

Input in checkpoint1.txt = abcdabcdabcdabcdabcdabcdabcdabcdabcdabcd

**Run the Program:**

The result we obtained from running the compressor is :
111001001110010011100100111001001110010011100100111001001110010011100100111001
00

**Manually build the Huffman Tree:**

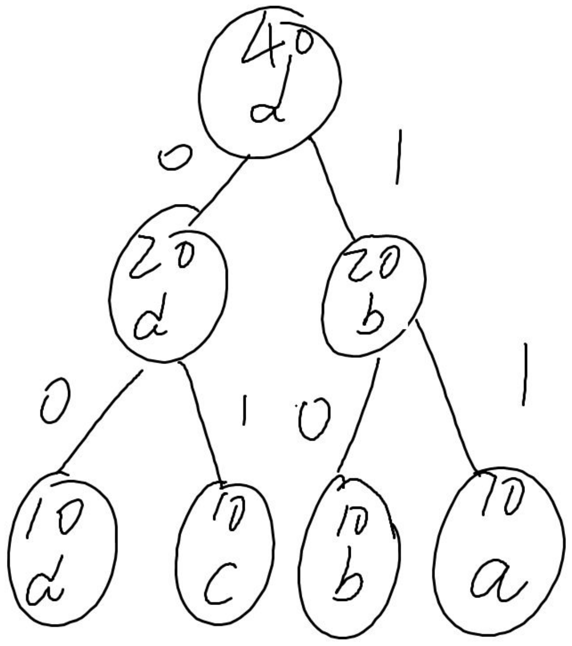
To build the Huffman Tree we have to obtain the input text, so we went to cse100 public folder and use cat command to find what is inside the checkpoint1.txt as we put the result on the top. The next step is count the frequency of each letter. We found out that frequency described in the table below.

| Character | Frequency |
|---|---|
| a | 10 |
| b | 10 |
| c | 10 |
| d | 10 |

There are two rules when we build our own Huffman Tree.

1. The most frequent element is always on the right. If there is a tie build them symbol order.
2. The left child is the 0 child, and the right child is the 1 child.

**Our tree looks like:**



**The coding will be:**

| Character | Code |
|---|---|
| a | 11 |
| b | 10 |
| c | 01 |
| d | 00 |

**Manually encode:**
Switch out the a,b,c,d with our code we get
1110010011100100111001001110010011100100111001001110010011100100111001001110010011100100
Which is exactly same from what we get by running compressor.

## Checkpoint2.txt

Input in checkpoint2.txt =
aabbbbccccccccddddddddddddddddddddddddddddddddccccccccbbbbaa

**Run the Program:**

The result we obtained from running the compressor is :
000000001001001001010101010101010111111111111111111111111111111111010101010101
0101001001001001000000

**Manually build the Huffman Tree:**

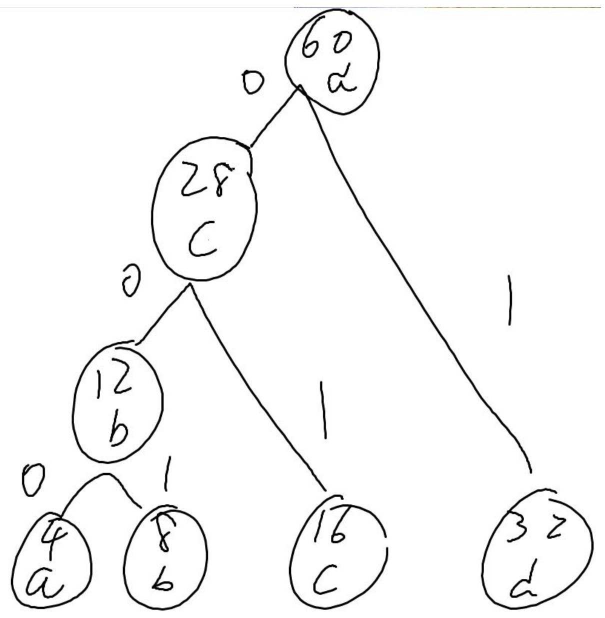
To build the Huffman Tree we have to obtain the input text, so we went to cse100 public folder and use cat command to find what is inside the checkpoint2.txt as we put the result on the top. The next step is count the frequency of each letter. We found out that frequency for each letter described in the table below.

| Character | Frequency |
|---|---|
| a | 4 |
| b | 8 |
| c | 16 |
| d | 32 |

There are two rules when we build our own Huffman Tree.

1. The most frequent element is always on the left. If there is a tie build them in alphabetical order.
2. The left child is the 0 child, and the right child is the 1 child.

**Our tree looks like:**



Thus, the coded characters would be:

| Character | Code |
|---|---|
| a | 000 |
| b | 001 |
| c | 01 |
| d | 1 |

**Manually encode:**

Switch out the a,b,c,d with our code we get

000000001001001001010101010101010111111111111111111111111111111111010101010101
0101001001001001000000

Which is exactly same from what we get by running compressor.