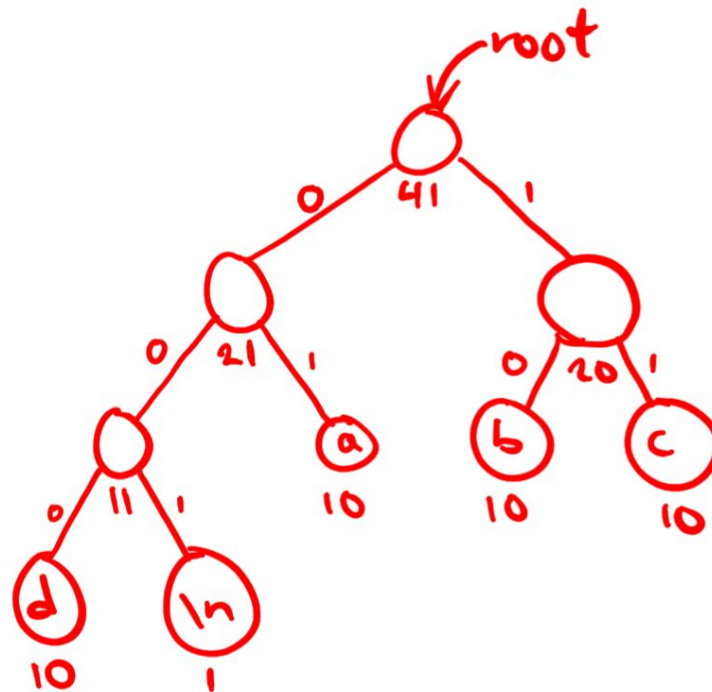


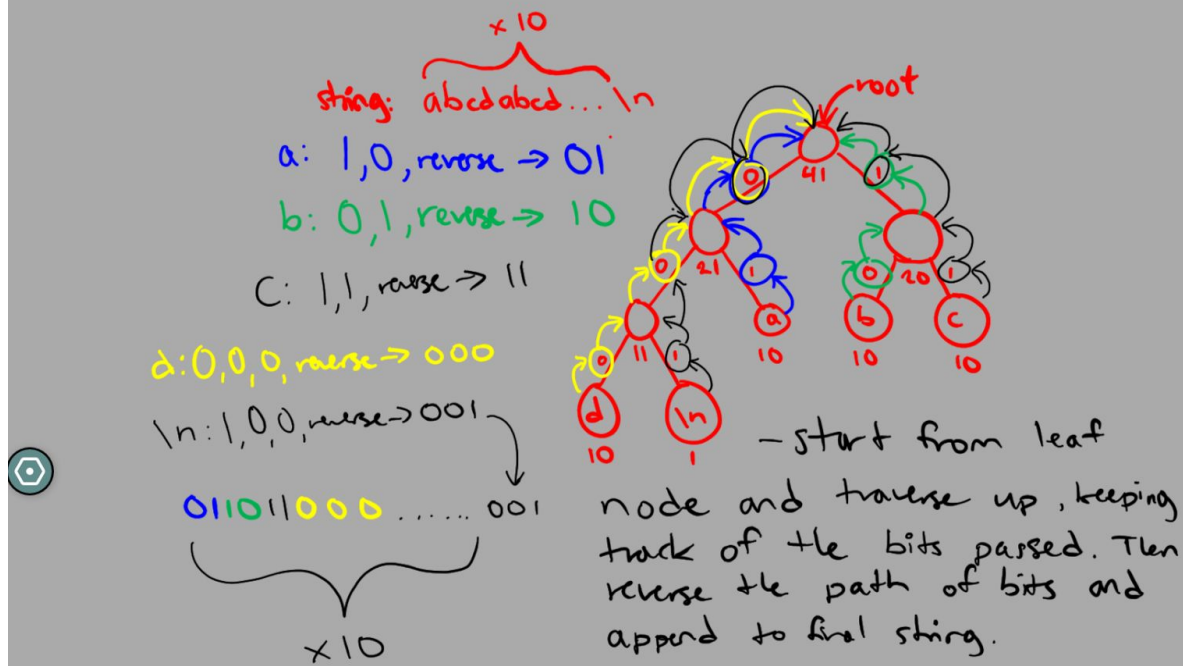
Kent Nguyen
Timothy Ferido

PA3 Report

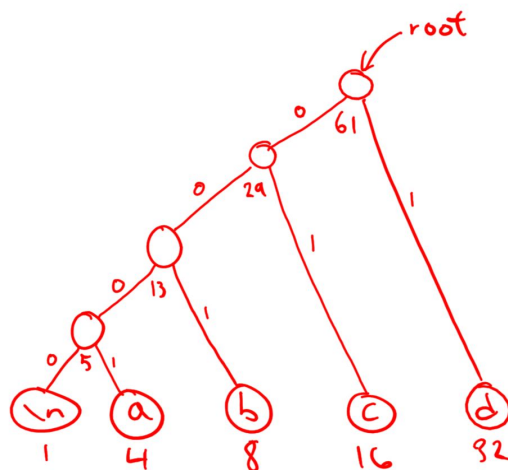
- The output for the command `./compress check1.txt checkoutput1.txt` is:
- Line 11: 1
- Line 98: 10
- Line 99: 10
- Line 100: 10
- Line 101: 10
- Encoded string:
0110110000110110000110110000110110000110110000110110000110110000110110
00011011000011011000001



- We built the tree by taking the two nodes with the lowest frequencies and pushing them onto a priority queue. Then two nodes are popped off at a time and a parent node is created that connects them and has the combined frequency. This node is then pushed back onto the priority queue and the process repeats until there is only one node, this is the root.
- To find the code word for each byte, we traverse down from the root, going right if the bit read is '1' and going left if the bit read is '0'. Once a leaf node is reached (no children) then the symbol is returned



- The output for the command `./compress check2.txt check2output.txt` is:
- Line 11: 1
- Line 98: 4
- Line 99: 8
- Line 100: 16
- Line 101: 32
- Encoded string:
 111011101101101101101010101010101000000000000000000000000000000010
 10101010101010110110110110110110110111
- The manual Huffman tree we got looks like this:



-
- Diagram illustrating a binary tree structure for Huffman coding. The root node is labeled "root" and has a value of 61. The tree structure is as follows:
- Root (61) branches into:
 - Left child (29)
 - Left child (13)
 - Left child (5)
 - Left child (1) (leaf node labeled 'ln')
 - Right child (4) (leaf node labeled 'a')
 - Right child (8)
 - Left child (4)
 - Left child (2)
 - Left child (1) (leaf node labeled 'b')
 - Right child (1) (leaf node labeled 'c')
 - Right child (2)
 - Left child (1) (leaf node labeled 'd')
 - Right child (1) (leaf node labeled 'e')
 - Right child (4)
 - Left child (2)
 - Left child (1) (leaf node labeled 'f')
 - Right child (1) (leaf node labeled 'g')
 - Right child (2)
 - Left child (1) (leaf node labeled 'h')
 - Right child (1) (leaf node labeled 'i')
 - Right child (16)
 - Left child (8)
 - Left child (4)
 - Left child (2)
 - Left child (1) (leaf node labeled 'j')
 - Right child (1) (leaf node labeled 'k')
 - Right child (2)
 - Left child (1) (leaf node labeled 'l')
 - Right child (1) (leaf node labeled 'm')
 - Right child (8)
 - Left child (4)
 - Left child (2)
 - Left child (1) (leaf node labeled 'n')
 - Right child (1) (leaf node labeled 'o')
 - Right child (2)
 - Left child (1) (leaf node labeled 'p')
 - Right child (1) (leaf node labeled 'q')

Annotations on the right side of the diagram:

 - String: $a \dots b \dots c \dots d \dots c \dots b \dots a \dots$ with weights $\times 2, \times 4, \times 8, \times 32, \times 8, \times 4, \times 2$ above it.
 - Binary strings and their reverse:
 - $a: 1, 0, 0, 0, \text{reverse} \rightarrow 0001$
 - $b: 1, 0, 0, \text{reverse} \rightarrow 001$
 - $c: 1, 0, \text{reverse} \rightarrow 01$
 - $d: 1, \text{reverse} \rightarrow 1$

Encoded string at the bottom:

ENCODED: $0001 \dots 001 \dots 01 \dots 1 \dots 01 \dots 001 \dots 0001 \dots$

Weights below the encoded string: $\times 2, \times 4, \times 8, \times 32, \times 8, \times 4, \times 2$

- ## Efficient Header Re-Design

- In order to implement a more efficient header when compressed, we decided to represent the *non-zero* frequencies as a 4 byte combination of two parts. The first part is a 1 byte number representing the line number / index of the freqs vector to build the Huffman tree. The second part is a 3 byte number representing the frequency. Additionally, the newline characters were removed. Here is a hexdump of the redesigned

header:

```

BitOutputStream.h      HCTree.h      check1output2.txt      output.txt
BitOutputStream.o      HCTree.o      check2.txt             output2.tx
CSE100_assignment3.pdf Makefile      compress               refcompres
MacBook-Pro-8:PA3 timferido$ xxd -b output.txt
00000000: 00001010 00000000 00000000 00000001 00100001 00000000 ....!.
00000006: 00000000 00000001 00100011 00000000 00000000 00000111 ..#...
0000000c: 00100100 00000000 00000000 00000110 00100110 00000000 $....&.
00000012: 00000000 00000010 00101000 00000000 00000000 00001000 ..(...
00000018: 00101001 00000000 00000000 00000010 00101010 00000000 )...*.
0000001e: 00000000 00000110 00101111 00000000 00000000 00000001 ../...
00000024: 00110000 00000000 00000000 00000011 00110010 00000000 0...2.
0000002a: 00000000 00000011 00110011 00000000 00000000 00000010 ...3...
00000030: 00111000 00000000 00000000 00000001 00111001 00000000 8...9.
00000036: 00000000 00000100 00111010 00000000 00000000 00000001 ...:..
0000003c: 01000000 00000000 00000000 00000011 01001010 00000000 @...J.
00000042: 00000000 00000001 01001111 00000000 00000000 00000010 ...0...
00000048: 01010000 00000000 00000000 00000001 01010101 00000000 P...U.
0000004e: 00000000 00000101 01100001 00000000 00000000 00001001 ..a...
00000054: 01100010 00000000 00000000 00000001 01100101 00000000 b...e.
0000005a: 00000000 00000001 01100110 00000000 00000000 00000001 ..f...
00000060: 01101001 00000000 00000000 00000001 01101010 00000000 i...j.
00000066: 00000000 00000011 01101111 00000000 00000000 00000010 ..o...
0000006c: 01110010 00000000 00000000 00000010 00000000 10011001 r....
00000072: 01110001 01011011 11101101 11001100 01101000 00100001 q[...h!
00000078: 11000001 10000110 10000100 00000100 01011011 01000011 ....[C
0000007e: 01000000 11001000 00111010 11111110 00111000 00010010 @...8.
00000084: 11011011 00000000 10011110 00100001 00101001 00101100 ...!),
0000008a: 11010110 01101001 00011101 10111001 01011111 00110001 .i..._1
00000090: 10011110 10101011 10001110 01110100 00101110 11101101 ...t..
00000096: 01000110 11101010 11001000 01100100 01001001 10010010 F..dI.
0000009c: 00000110 00000110 ..
MacBook-Pro-8:PA3 timferido$

```

Line number
Frequency of Symbol
tree
Delimiter
Encoding

- The only “entries” are frequencies that are non-zero. The first “Line Number Byte” that is 00000000, and is not the first byte, is read to be the delimiter and the remaining bytes create the encoding, using the tree.