Code Analysis for Encode

TCSS 342: Data Structures
Jesse Bannon

Functions and Notable Lines within Encode

```
while ((c = fgetc(in)) != EOF):
```

Description: Loops through the entire input text file assigning character c to the current letter the stream is at.

Value: Let n represent the number of characters within the input text file.

$$g(n, contents) = \sum_{i=1}^{n} [Contents\ within\ the\ loop] \quad , \quad O(g(n)) = O(n)$$

buildTree:

Description: Builds a Huffman Tree using an array of character frequencies. Starts by creating a tree for every letter and placing them in a queue from least to greatest based on frequency. A while loop will merge the two trees of the lowest weight together combining their weight and continue looping until there is one tree remaining. It then returns the tree.

Value: Let *m* represent the number of characters with frequency greater than zero. Let $c_{build\ leaf}$ represent the number of instructions it takes to build a new tree.

$$\sum_{i=1}^{m} c_{build\ leaf} = m * c_{build\ leaf} \quad , \quad O(g(m)) = O(m)$$

buildTable:

Description: Builds the codeword table from a Huffman Tree and assigns it to a t_node table. It traverses to every leaf in the tree and stores the traversal as an unsigned integer and the character found within the leaf.

Value: Let m represent the number of characters within the Huffman tree. Let $c_{traverse\ tree}$ represent the number of instructions it takes to traverse to each character.

$$\sum_{i=1}^{m} c_{traverse\ tree} = m * c_{traverse\ tree} \quad , \quad O(g(m)) = O(m)$$

writeHeader:

Description: Writes every character and associated codeword to the top of the binary output.

Value: Let m represent the number of characters. Let $c_{\text{write code}}$ represent the number of instructions it takes to write to the file.

$$\sum_{i=1}^{m} c_{\text{write code}} = m * c_{\text{write code}} \quad , \quad O(g(m)) = O(m)$$

printCodewords:

Description: Prints each character and their codeword to a readable text file.

Value: Let m represent the number of characters. Let $c_{\text{print code}}$ represent the number of instructions it takes to write to the text file.

$$\sum_{i=1}^{m} c_{\text{print code}} = m * c_{\text{print code}} \quad O(g(m)) = O(m)$$

getCodeWord:

Description: Returns the codeword from the table for the character given.

Value: Let m represent the number of characters.

Best case: $O(g(m)) = O(1)$

Worst case: $O(g(m)) = O(m)$

Ave case: $\frac{m}{2}$

Runtime of Encode

```
int Encode(FILE *in,                                                  1
             FILE *out)                                               1
{
    unsigned int c, bufferSize, bitBuffer, codeword, charCount, end;  6
    unsigned int frequency[CHAR_RANGE] = { 0 };                       1
    int bindex;                                                       1
    t_node** table;                                                   1

    while ((c = fgetc(in)) != EOF) frequency[c]++;                    g(n, 2);
    frequency[END_OF_TEXT]++;                                         1
    rewind(in);                                                       1

    tree_t* head = buildTree(frequency, &charCount);                  m*c_build leaf

    table = buildTable(head, charCount);                              m*c_traverse tree

    writeHeader(out, table, charCount);                               m*c_write code
    printCodewords(table, charCount);                                 m*c_print code

    bitBuffer = 0;                                                    1
    bufferSize = 0;                                                   1
    end = 0;                                                          1
    while ((c = fgetc(in)) != EOF) {                        LOOP_1    g(n,LOOP_1)
        codeword = getCodeWord(table, charCount, c);                  m/2
end_of_file:                                                   JUMP_1
        bindex = 31;                                                  1
        while (!CHECK_BIT(bindex--, codeword));                       c_check bit
        while (bindex >= 0) {                                  LOOP_2 1
            if (CHECK_BIT(bindex--, codeword))                        1
                ENCODE_BIT(bufferSize++, bitBuffer);                  1
            else
                bufferSize++;                                         1

            if (bufferSize == 32) {                                   1
                fwrite(&bitBuffer, sizeof(unsigned int), 1, out);     1
                bufferSize = 0;                                       1
                bitBuffer = 0;                                        1
            }                                                  END_LOOP2
        }
    }                                                       END_LOOP1 END_JUMP1
    if (!end) {                                                       1
        end = 1;                                                      1
        codeword = getCodeWord(table, charCount, END_OF_TEXT);        m/2
        goto end_of_file;                                             1 + JUMP_1
    }
    if (bufferSize)                                                   1
        fwrite(&bitBuffer, sizeof(unsigned int), 1, out);             1
}
```

Runtime Analysis

Let $f(n)$ represent the function Encode
Let n represent the amount of characters within the file
Let m represent the amount of unique characters within the file

$$f(n)=1+1+6+1+1+1+g(n,2)+1+1+m*c_{build\,leaf}+m*c_{traverse}tree+m*c_{write\,code}+m*c_{print\,code}$$
$$+1+1+1+g(n,LOOP_1)+1+1+\frac{m}{2}+1+JUMP_1+1+1$$

$$f(n)=g(n,2)+m(c_{build\,leaf}+c_{traverse\,tree}+c_{write\,code}+c_{print\,code}+½)+g(n,LOOP_1)+(c_{checkbit}+8)+21$$

Let $c_{unique\,chars}$ represent $c_{build\,leaf}+c_{traverse\,tree}+c_{write\,code}+c_{print\,code}+½$
Let c_{jump} represent $(c_{checkbit}+8)+21$

$$f(n)=g(n,2)+g(n,LOOP_1)+m*c_{unique\,chars}+c_{jump}$$

$$f(n)=\sum_{i=1}^{n}[2]+\sum_{i=1}^{n}[\frac{m}{2}+1+c_{checkbit}+LOOP_2]+m*c_{unique\,chars}+c_{jump}$$

$$f(n)=2n+n[\frac{m}{2}+1+c_{checkbit}+8]+m*c_{unique\,chars}+c_{jump}$$

$$f(n)=n[2+\frac{m}{2}+1+c_{checkbit}+8]+m*c_{unique\,chars}+c_{jump}$$

Because $1\leq m\leq 256$, we will let c_1 represent m

$$f(n)=n[2+\frac{c_1}{2}+1+c_{checkbit}+8]+c_1*c_{unique\,chars}+c_{jump}$$

Let c_2 represent $2+\frac{c_1}{2}+1+c_{checkbit}+8$

$$f(n)=n*c_2+c_1*c_{unique\,chars}+c_{jump}$$

Let c_3 represent $c_1*c_{unique\,chars}+c_{jump}$

$$f(n)=n*c_2+c_3$$

Thus, the function $f(n)\in O(n)$