Huffman Static Coding

Akshett Rai Jindal
Shivang Gupta
PiedPiper
akshett.jindal@research.iiit.ac.in
shivang.gupta@students.iiit.ac.in

October 24, 2020

I. Introduction

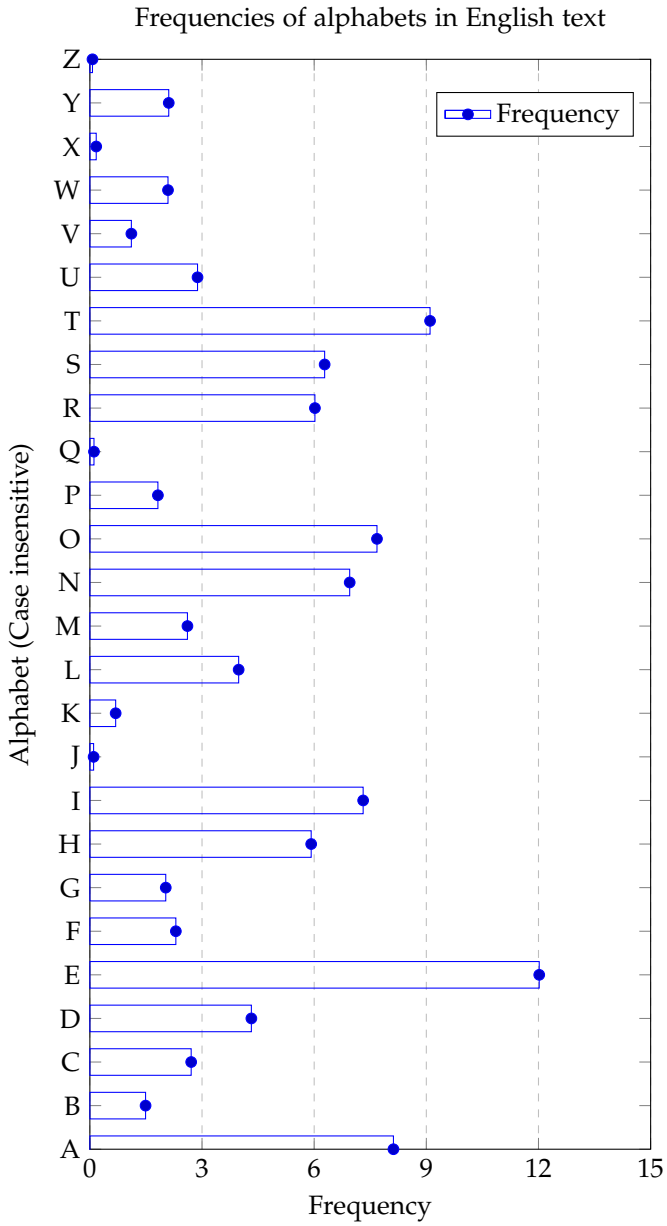
In normal text files each character is stored as a group of 8 bits know as 1 byte. The number represented by these 8 bits in binary system is the ASCII value for that character. These values range from 0 to 255 ($2^8 - 1$). These values represent a variety of characters ranging from alphabets (a-z, A-Z), digits (0-9), special symbols (&, *, /, + etc.), paranthesis, brackets, EOF, newline characters etc.

Representation of 'A':
01000001
So, its ASCII value is:
$1 * 2^0 + 1 * 2^6 = 65$

So, if there are n characters in a file, then its size would be equal to $(n * 8)$ bits = n bytes.

II. Idea

Now, what happens is that if we take a huge text, then we see that the frequencies of the characters is not same for all the characters. Some have high frequencies and some have very low frequencies. For example, the general frequencies for an English text containing 40,000 words are:



So, somehow if we decrease the number of bits for storing the high frequency characters and increase the bits for low frequency characters, then there are very good chances that the memory required to store the text will decrease.

III. Requirements

If we code the characters in the method mentioned above, then it should satisfy two conditions. These are:

1. Code for any character should not have any other character's code as a prefix. This type of code is called "prefix code".
2. The length of the new text should not be bigger than the original text.

If the first condition is violated, then there will be a loss of data as it can become uncertain whether to convert the prefix or the whole code into character. For example, consider the mapping for two elements:

$$a = 10011$$
$$b = 01011$$
$$c = 10011010$$
$$d = 1111$$

and some other characters

Then, if encoder is given the text which contains 'cd' and some other characters, then it will turn it into 100110101111 and there will be some bits that follow this from other characters. This can confuse the decoder whether to convert:

1. 10011010 into c and 1111 into d
2. 10011 to a, 01011 to b and 11 along with other following bits into some other character

If the second condition is violated, then there is no use of converting the characters into codes as they would take more space than original.

IV. Huffman Coding

David A. Huffman gave a frequency based binary coding technique which he proved to be most efficient. This method uses bottom-up approach which guarantees optimality.

What we do is take all the characters along with their frequencies. We then take the two characters with the least frequencies and replace them with a node having the frequency as the sum of the two frequencies. This is done until there is only one character is left.

This generates a binary tree in which all the original characters are present only at the leaf nodes. Then the head node of the tree is assigned an empty code. Then a convention is fixed that the left child will be given 0 and the right child 1 (This can be reversed also, but the order is fixed for the whole tree.) Then if parent node has code c, where c is a binary number, then the left child will be assigned the code $c0$, i.e. c appended with 0 and the right child will have the code $c1$. This will give the huffman codes for the leaf nodes, i.e. the actual characters.

V. Our Code

i. Working

What the code does is, reads the file and counts the occurences of the 255 characters and then adds them to a min heap. After that we have the binary tree ready and hence the codes are ready. Then we also had to handle the cases where the binary string generated did not have a length which is multiple of 8 because C can write data in multiple of bytes. Currently we write data in bytes but plan to cahnge it to multiple bytes.

ii. Benchmark

The program compressed a lorem ipsum file from 966.4 MB to 516.4 MB in 1 minute 14 seconds and decompressed it in 58 seconds without any loss in the data.