Huffman Coding

In this tutorial, you will learn how Huffman Coding works. Huffman Coding is a technique of compressing data to reduce its size without losing any of the details. It was first developed by David Huffman in 1951. Huffman Coding is generally useful to compress the data in which there are frequently occurring characters.

How Huffman Coding works?

Suppose the string below is to be sent over a network.

Each character occupies 8 bits. There are a total of 15 characters in the above string. Thus, a total of `8 * 15 = 120` **bits** are required to send this string. Using the Huffman Coding technique, we can compress the string to a smaller size.

Huffman coding first creates a tree using the frequencies of the character and then generates code for each character. Once the data is encoded, it has to be decoded. Decoding is done using the same tree.

Huffman Coding prevents any ambiguity in the decoding process using the concept of **prefix code** ie. a code associated with a character should not be present in the prefix of any other code.

Huffman coding is done with the help of the following steps.

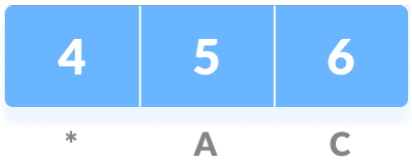
Calculate the frequency of each character in the string.

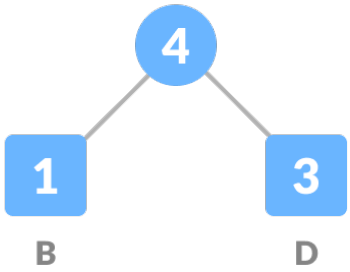
Sort the characters in increasing order of frequency. These are stored in a priority queue *Q*.

Make each unique character as a leaf node.

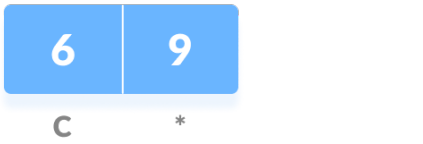
Create an empty node *z*. Assign the minimum frequency to the left child of **z** and assign the second minimum frequency to the right child of *z*. Set the value of the *z* as the sum of the above two minimum frequencies.

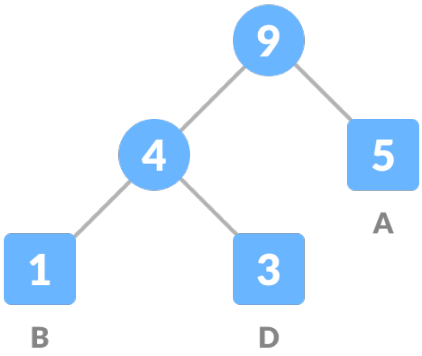
Remove these two minimum frequencies from *Q* and add the sum into the list of frequencies (* denote the internal nodes in the figure above).

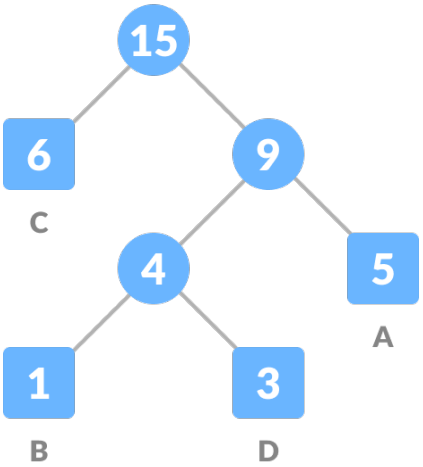




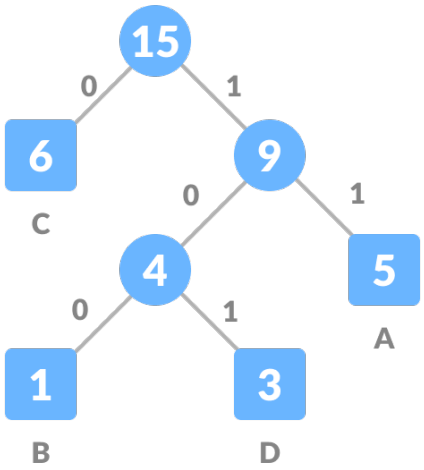
Repeat these steps for all the characters in the alphabet.







For each non-leaf node, assign 0 to the left edge and 1 to the right edge.



For sending the above string over a network, in general we have to send the tree as well as the above compressed-code. The total size is given by the table below.

Character	Frequency	Code	Size
A	5	11	5*2 = 10
B	1	100	1*3 = 3
C	6	0	6*1 = 6
D	3	101	3*3 = 9
4 * 8 = 32 bits	15 bits		28 bits

Without encoding, the total size of the string was 120 bits. After encoding the size is reduced to `32 + 15 + 28 = 75`.

Decoding the code

For decoding the code, we can take the code and descend the tree starting from the root until we reach a leaf node. The label of the node is the decoded character.
Let 101 the code to be decoded, we can traverse the tree from the root as in the figure below, and we get character D.