Huffman Coding

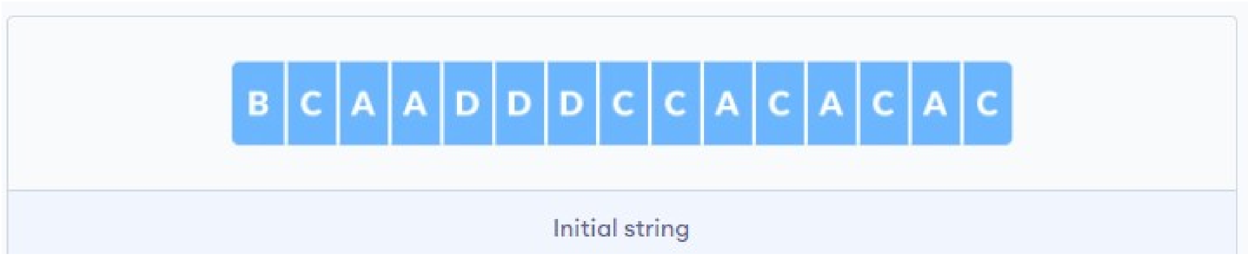
Prepared by Abidullah Khan

Huffman Coding

- 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.
- Huffman Coding is generally useful to compress the data in which there are frequently occurring characters.
- The idea is to assign variable-length codes to input characters, lengths of the assigned codes are based on the frequencies of corresponding characters.
- The most frequent character gets the smallest code and the least frequent character gets the largest code.
- The variable-length codes assigned to input characters are Prefix Codes, means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of code assigned to any other character.
- This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bitstream.

How Huffman Coding works?

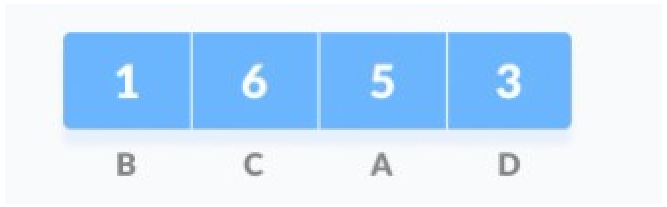
- Suppose the string below is to be sent over a network.



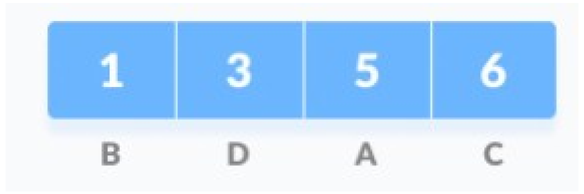
- Each character occupies 8 bits. There are a total of 15 characters in the above string. Thus, a total of 8 * 15 = 120 bits are required to send this string. Using the Huffman Coding technique, we can compress the string to a smaller size.
- Huffman coding first creates a tree using the frequencies 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 Steps

- **Step 1:** Calculate the frequency of each character in the string.

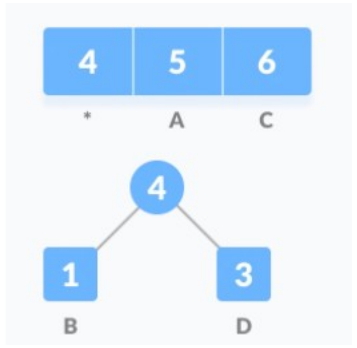


- **Step2:** Sort the characters in increasing order of the frequency. These are stored in a priority queue Q.

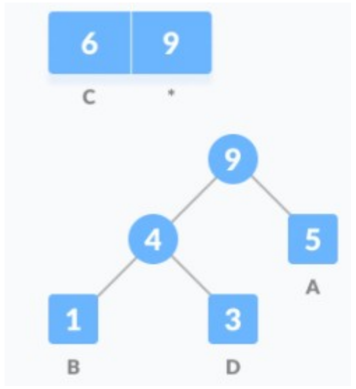


- **Step 3:** Make each unique character as a leaf node.
- **Step 4:** Create an empty node z. Assign the minimum frequency to the left child of z and assign the second minimum frequency to the right child of z. Set the value of the z as the sum of the above two minimum frequencies.

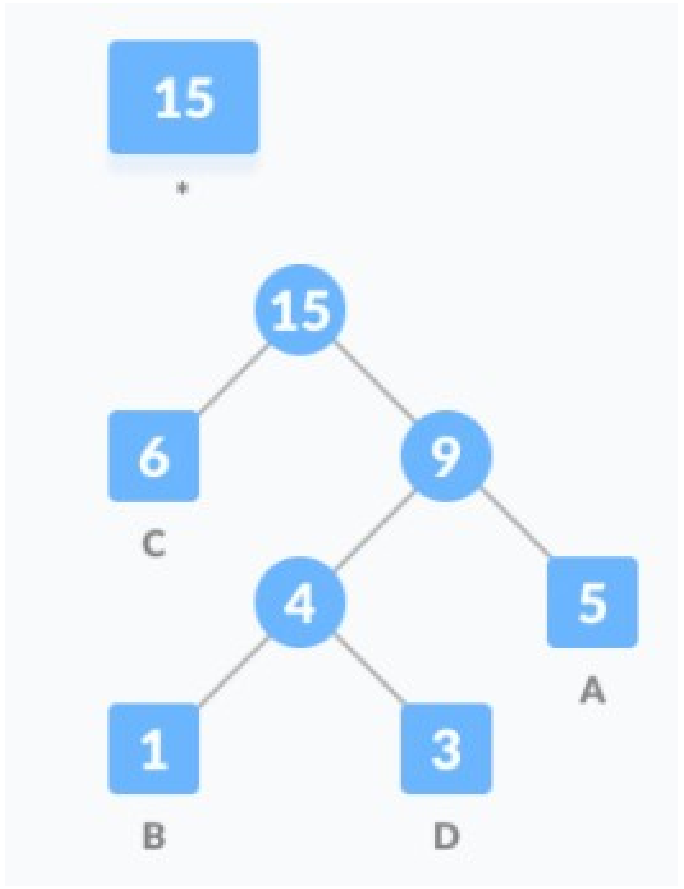
Huffman Coding Steps



- **Step 5:** Remove these two minimum frequencies from Q and add the sum into the list of frequencies (* denote the internal nodes in the figure above).
- **Step 6:** Insert node z into the tree.
- **Step 7:** Repeat steps 3 to 5 for all the characters.

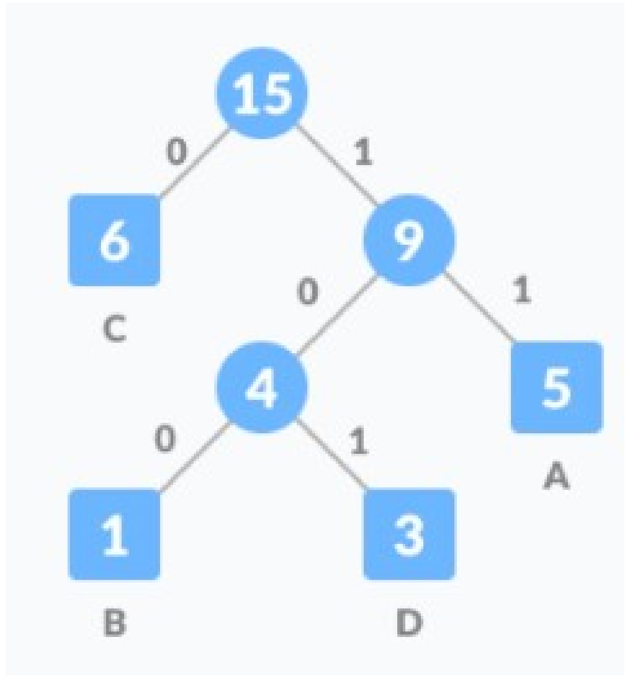


Huffman Coding Steps



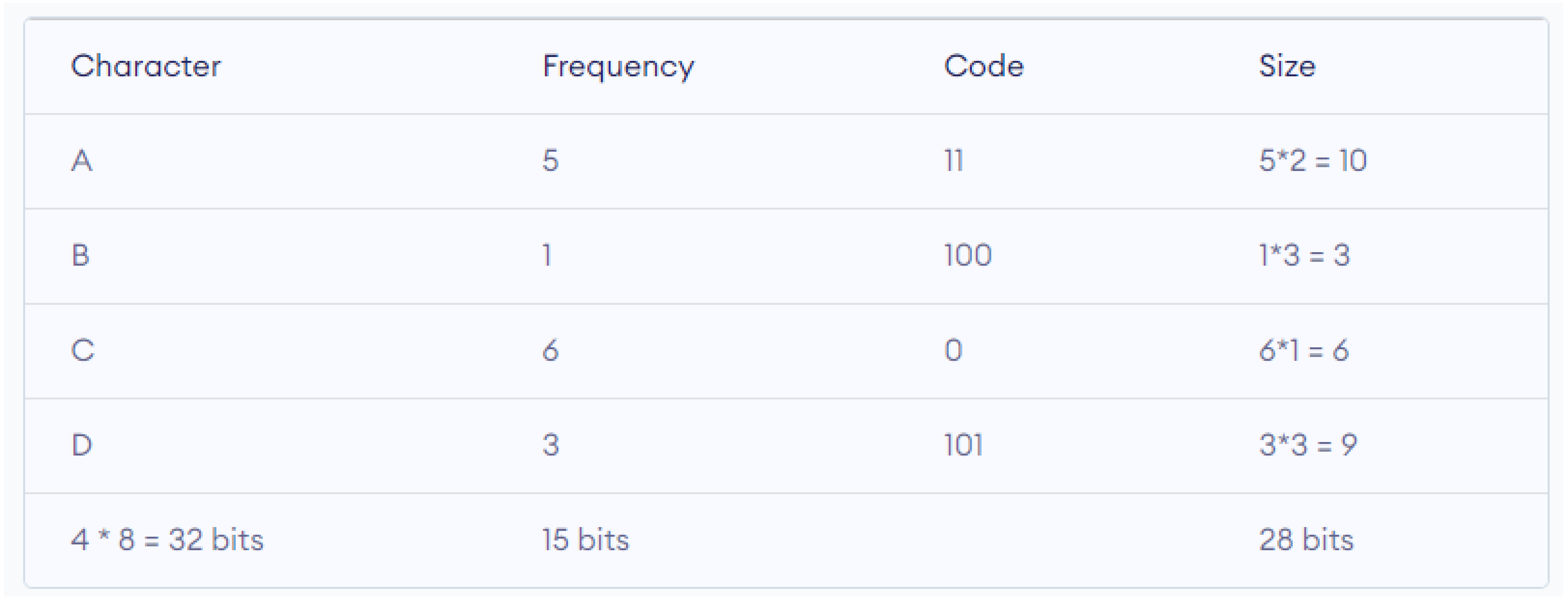
Huffman Coding Steps

Step 8: For each non-leaf node, assign 0 to the left edge and 1 to the right edge.



Huffman Coding Steps

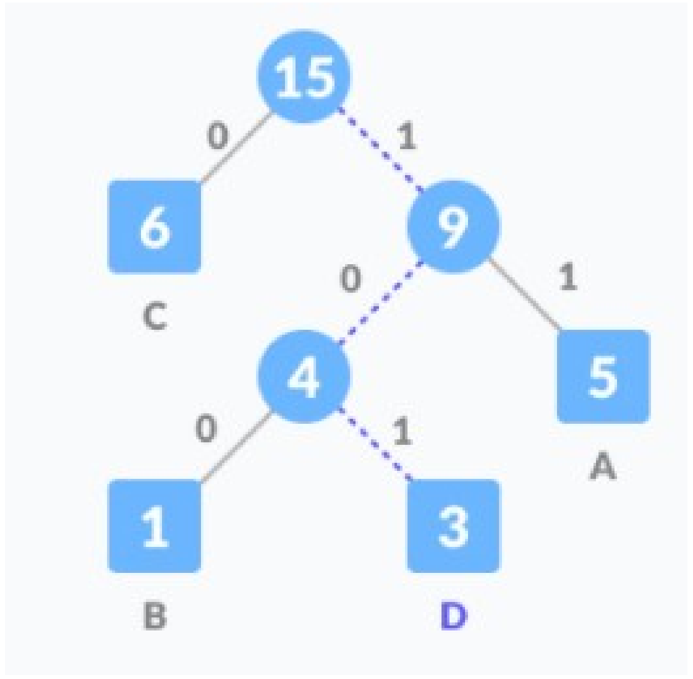
- For sending the above string over a network, we have to send the tree as well as the above compressed-code. The total size is given by the table below.



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

Huffman Coding Steps

- **Decoding the code:** For decoding the code, we can take the code and traverse through the tree to find the character. Let 101 is to be decoded, we can traverse from the root as in the figure below



Huffman Coding Algorithm

1. Create a priority queue Q consisting of each unique character.
2. Sort them in ascending order of their frequencies.
3. For all the unique characters:
4. Create a **newNode**
5. Extract the most minimum value from Q and assign it to **leftChild** of **newNode**
6. Extract the second minimum value from Q and assign it to **rightChild** of **newNode**
7. Calculate the sum of these two minimum values and assign it to the value of **newNode**
8. Insert this **newNode** into the tree
9. Return **rootNode**