Que-2

Implement Huffman coding and decoding algorithm?

Huffman coding is famous Greedy Algorithm. It is used for the lossless compression of data. It uses variable length code to all the characters. The code length of a character depends on how frequently it occurs in the given text.
The character which occurs most frequently gets the smallest code. The character which occurs least frequently gets the largest code.

Prefix Rule:-

- Huffman coding implements a rule known as prefix rule.
- This is to prevent the ambiguities while decoding.
- It ensures that the code assigned to any character is not a prefix of the code assigned to any other character.

Major steps in Huffman Coding:

There are two major steps in Huffman coding-

(i) Building a Huffman Tree from the input characters.

(ii) Assigning code to the characters by traversing the Huffman Tree.

Huffman Tree :

The steps involved in the construction of Huffman Tree are as follows:

step-1

- Create a leaf node for each character of the text.
- Leaf node of a character contains the occurring frequency of that character.

step-2

- Arrange all the nodes in increasing order of their frequency value.

step-3

consider the first two nodes having minimum frequency.

- create a new internal node.
- The frequency of new node is the sum of frequency of those two nodes.
- Make the first node as a left child and the other node as a right child of the newly created node.

step-4

- Keep repeating step-02 and step-3 until all the nodes from a single tree.
- The tree finally obtained is the desired Huffman Tree.

So we get Huffman code for characters.

To write huffman code for any character, traverse the Huffman Tree from root node to the leaf node of that character.

- a = 1100
- b = 1101
- c = 100
- d = 101
- e = 111
- f = 0

From here, we observe that

- characters occurring less frequency in the text are assigned the larger code.
- characters occurring more frequently in the text are assigned the smaller code.

- Average code length:-

Avg. code length = $\sum(\text{frequency}_i \times \text{code length}_i) / \sum(\text{frequency}_i)$

$$= \frac{(5\times4 + 9\times4 + 12\times3 + 13\times3 + 16\times3 + 45\times1)}{(5+9+12+13+16+45)}$$

$= 2.24$

- Length of Huffman Encoded message-

$= 100 \times 2.24$

$= 224$ bits

Example:

Characters	a	b	c	d	e	f
frequency	5	9	12	13	16	45

Suppose we are given above characters with the frequencies as shown above. We can try to find

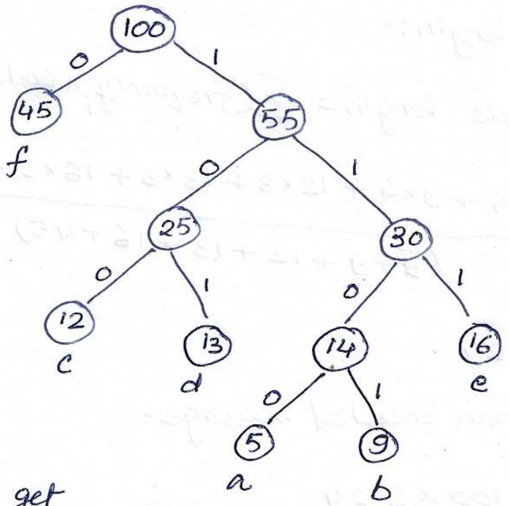
(i) Huffman code for each character.

(ii) Average code length.

(iii) Length of Huffman encoded message (in bits).

Solution:

First let us construct the Huffman Tree.



From tree we get huffman coding :

f : 0
c : 100
d : 101
a : 1100
b : 1101
e : 111

Time complexity:

$O(n\log n)$

Where n is number of unique characters. If there are n nodes, extractMin() is called $2*(n-1)$ times. extractMin() takes $O(\log n)$ time as it calls minHeapify(), so overall

time complexity = $O(n\log n)$