Experiment. No. 02

Title : write a program to implement Huffman Encoding using a greedy strategy.

objective : To understand and solve Huffman Encoding using greedy method.

Theory :

what is a Greedy method :

A greedy algorithm is an approach for solving a problem by selecting the best option available at the movement. It doesn't worry whether the current best result will bring the overall optimal result.

It works in a top-down approach

The algorithm may not produce the best result for all the problems.

Greedy Algorithm :

1. To begin with, the solution set is empty.
2. At each step, an item is added to the solution set until a solution is reached
3. If the solution set is feasible, the current item is kept
4. else, the item is rejected and never considered again.

Huffman Encoding :

Huffman coding is a technique of compres data reduce its size without losing any o details.

It was first developed by David Huffman

Huffman coding is generally useful to co the data in which there are frequently o characters.

It uses variable length coding

It is used for the lossless compressio data.

It assigns variable length code to all th characters.

The code length of a characters depend how frequently it occurs in the given text

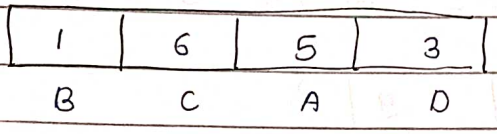
It is also known as Huffman encoding.

major steps in Huffman coding :

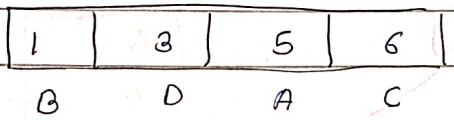
1. Building a Huffman tree from the input characters
2. Assigning code to the characters by t -g the Huffman tree.

Huffman coding is done with the help of following steps.

1. Calculate the frequency of each characters in the string

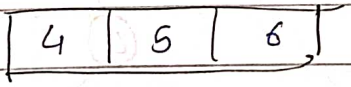


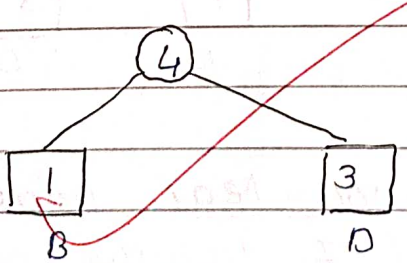
2. Sort the characters in increasing order of the frequency



3. make each unique character as a leaf node.

4. Create an empty node z. Assign the min. freque-ncy to the left child of z and assign with the second min. frequency to the right child of z. set the value of the z as the sum of the above two min frequencies

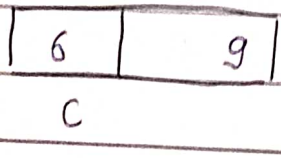


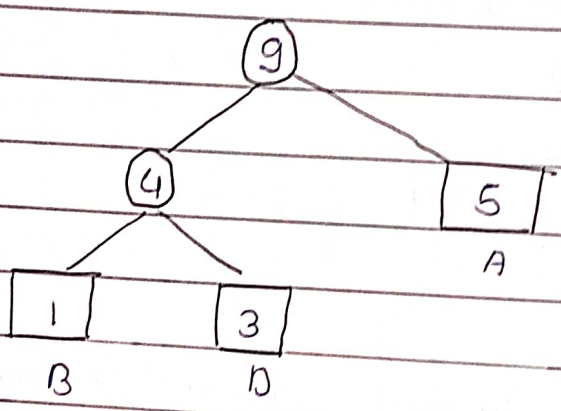


5. Remove these two min. frequencies from @ and the sum into the list of frequencies.

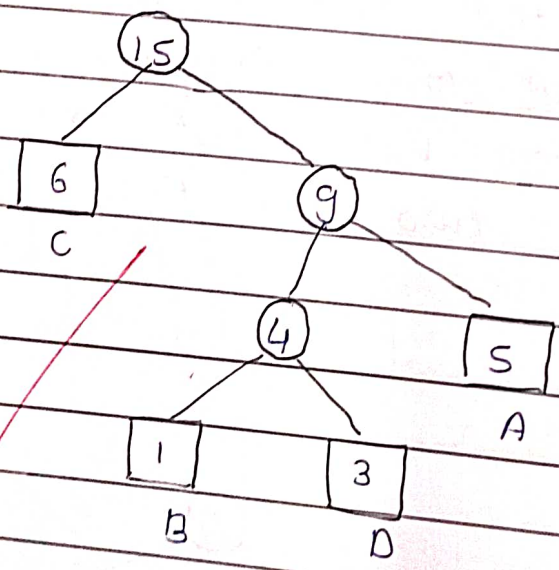
6. Insert node z into the tree

7. Repeat steps 3 to 5 for all charac

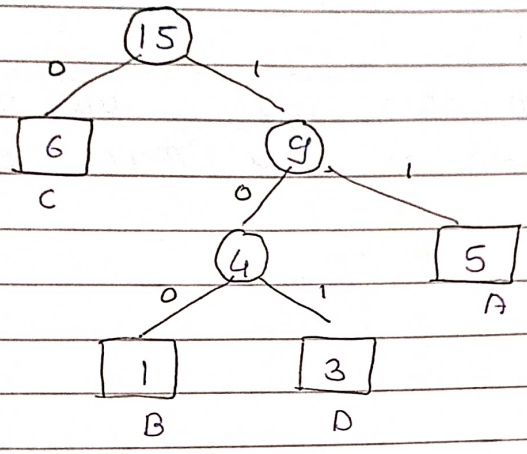




IS



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



character	Frequency	code	size
A	5	11	5 x 2 = 10
B	1	100	1 x 3 = 3
C	6	0	6 x 1 = 6
D	3	101	3 x 3 = 9
4 x 8 = 32 bits	15 bits		28 bits

without encoding the total size was 120 bits. After encoding the size reduced to 32 + 15 + 28 = 75

Time complexity -

extract min() is called 2 x (n-1) times if there are n nodes

As extract min() calls minHeapify(). It takes O(log n) time

Thus, overall time complexity of Huffman coding becomes O(n log n)