Experiment - 8

Aim :- To generate the Huffman codes for a given character set with frequencies.

Theory :-

It is an optimal prefix code used for lossless data compression. It builds up an optimal way of representing each character as a binary string.

Node structure
char
freq
left
right

Huffman Procedure (C)
n ← C - size
Q ← Priority_Queue (C)
for i in 1 to n
n = node (C[i])
Q. Push (n)
end for
while (Q.size != 1)
z = new node ()
x = Q. head ()

ANALYSIS :-

The time complexity of the Huffman Algorithm is $O(n \log n)$. Using a heap to store the weight of each tree, each iteration requires $O(\log n)$ time to determine the cheapest weight and insert the new weight. There are $O(n)$ iterations, one for each item.

```
    Q.pop().
    y = Q.head()
    Q.pop()
    z.left = x,   z.right = y
    z.freq = x.freq + y.freq
    Q.Push(z)
  end while
  return Q
end

Print (Node root, String s)
  if (root.data != '1')
    Print (root.char + ":" + str)
  Print (root.left, str + "0")
  Print (root.right, str + "1")
end
```

Result:-
The Huffman code was successfully generated and its time complexity was analyzed.

23/10/18