There are mainly two major parts in Huffman Coding

1) Build a Huffman Tree from input characters.

2) Traverse the Huffman Tree and assign codes to characters.

Steps to build Huffman Tree

Input is array of unique characters along with their frequency of occurrences and output is Huffman Tree.

1. Create a leaf node for each unique character and build a min heap of all leaf nodes (Min Heap is used as a priority queue. The value of frequency field is used to compare two nodes in min heap.

Initially, the least frequent character is at root)

2. Extract two nodes with the minimum frequency from the min heap.

3. Create a new internal node with frequency equal to the sum of the two nodes frequencies. Make thefirst extracted node as its left child and the other extracted node as its right child. Add this node to themin heap.

4. Repeat steps#2 and #3 until the heap contains only one node. The remaining node is the root nodeand the tree is complete.

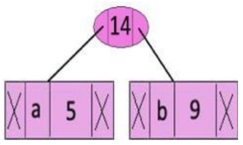
Example:

Let us understand the algorithm with an example:

character	Frequency
a	5
b	9
c	12
d	13
e	16
f	45

Step 1: Build a min heap that contains 6 nodes where each node represents root of a tree with singlenode.

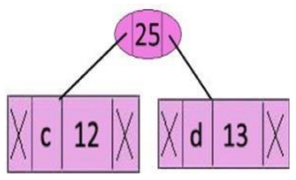
Step 2 **:** Extract two minimum frequency nodes from min heap. Add a new internal node with frequency 5 + 9 = 14.



Now min heap contains 5 nodes where 4 nodes are roots of trees with single element each, and oneheap node is root of tree with 3 elements,

Character	
Frequencyc	12
d	13
Internal Node	14
e	16
f	45

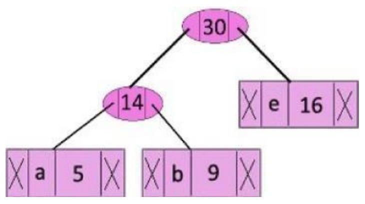
Step 3: Extract two minimum frequency nodes from heap. Add a new internal node with frequency12 + 13 = 25



Now min heap contains 4 nodes where 2 nodes are roots of trees with single element each, and twoheap nodes are root of tree with more than one nodes.

Character	Frequency
Internal Node	14
e	16
Internal Node	25
f	45

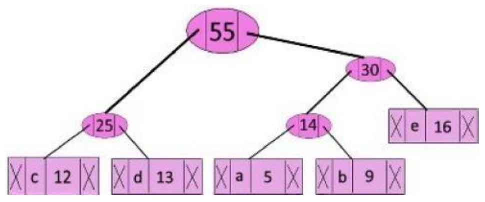
Step 4: Extract two minimum frequency nodes. Add a new internal node with frequency 14 + 16 =30



Now min heap contains 3 nodes.

Character	Frequency
Internal Node	25
Internal Node	30
f	45

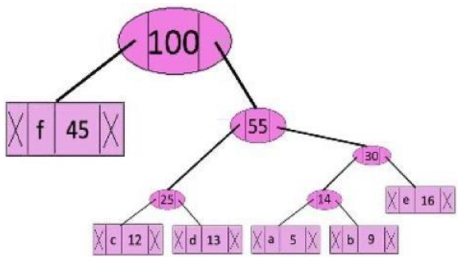
Step 5: Extract two minimum frequency nodes. Add a new internal node with frequency 25 + 30 =55



Now min heap contains 2 nodes.

Character	Frequency
f	45
Internal Node	55

Step 6: Extract two minimum frequency nodes. Add a new internal node with frequency 45 + 55 =100

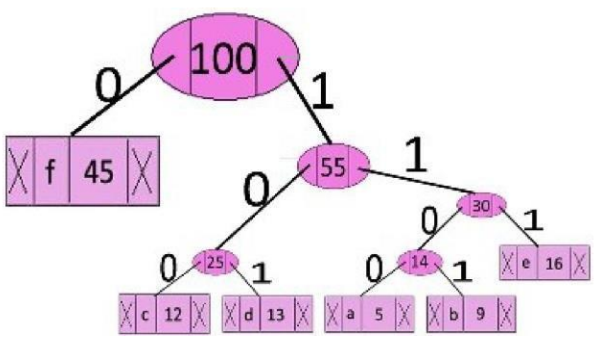


Now min heap contains only one node.

Character	Frequency
Internal Node	100.

Since the heap contains only one node, the algorithm stops here.

Steps to print codes from Huffman Tree:
Traverse the tree formed starting from the root. Maintain an auxiliary array. While moving to the left child, write 0 to the array. While moving to the right child, write 1 to the array. Print the arraywhen a leaf node is encountered.



The codes are as follows:

Character	code-word
f	0
c	100
d	101
a	1100
b	1101
e	111

Algorithm for Huffman code

Input:-Number of message with frequency count.Output: - Huffman merge tree.

```
1. Begin
2. Let Q be the priority queue,
3. Q= {initialize priority queue with frequencies of all symbol or message}
4. Repeat n-1 times
5. Create a new node Z
6. X=extract_min(Q)
7. Y=extract_min(Q)
8. Frequency(Z) =Frequency(X) +Frequency(y);
9. Insert (Z, Q)
10. End repeat
11. Return (extract_min(Q))
12. End.
```

Time Complexity-

O(nlogn) where n is the number of unique characters. If there are n nodes, extractMin() is called 2*(n – 1) times. extractMin() takes O(logn) time as it calles minHeapify(). So, overall complexity is O(nlogn).
Thus, Overall time complexity of Huffman Coding becomes O(nlogn).If the input array is sorted, there exists a linear time algorithm.

Conclusion: In this way concept of Huffman Encoding is explored using greedy method.