Write a Program to Implement Huffman coding.

Mr. Vikas Kumar

Assistant Professor

Industry Embedded Program

Lesson Plan

Subject/Course	Competitive Coding
Lesson Title	Write a Program to Implement Huffman coding.

Lesson Objectives
Understand the concept and purpose of Huffman Coding for data compression.
Learn how to construct a Huffman Tree using character frequencies.
Write a program to generate Huffman Codes for given inputs.
Analyze the efficiency and applications of Huffman Coding in real-world use.

Problem Statement:

Write a Program to Implement Huffman coding.

Problem Overview

Given a string S of distinct character of size N and their corresponding frequency f[ ] i.e. character S[i] has f[i] frequency. Your task is to build the Huffman tree print all the huffman codes in preorder traversal of the tree.

Note:

While merging if two nodes have the same value, then the node which occurs at first will be taken on the left of Binary Tree and the other one to the right, otherwise Node with less value will be taken on the left of the subtree and other one to the right.

Example 1:

S = "abcdef"

f[] = {5, 9, 12, 13, 16, 45}

Output:

0 100 101 1100 1101 111

Explanation:

HuffmanCodes will be:

f : 0

c : 100

d : 101

a : 1100

b : 1101

e : 111

Concept:

What is Huffman Coding?

- Huffman Coding is a lossless compression technique that reduces file size by encoding data efficiently.

- It assigns shorter binary codes to more frequent characters and longer codes to less frequent ones.

- This method ensures that no code is a prefix of another, making decoding unique and reliable.

- It was developed by David A. Huffman in 1952 and is widely used in compression tools.

Concept:

Why Huffman Coding?

- Data compression helps in saving storage space and reducing transmission time.
- Huffman coding ensures that frequently used symbols consume fewer bits.
- It provides one of the most efficient variable-length encoding schemes.
- Applications include ZIP files, JPEG, MP3, and communication protocols.

Concept:

Steps in Huffman Coding

1. Count the frequency of each character in the data.
2. Create a min-heap (priority queue) containing all characters with their frequencies.
3. Repeatedly extract two smallest nodes and merge them into one node.
4. Continue until one node remains, which becomes the root of the Huffman Tree.

Concept:

Example – Building the Tree

Given Input:
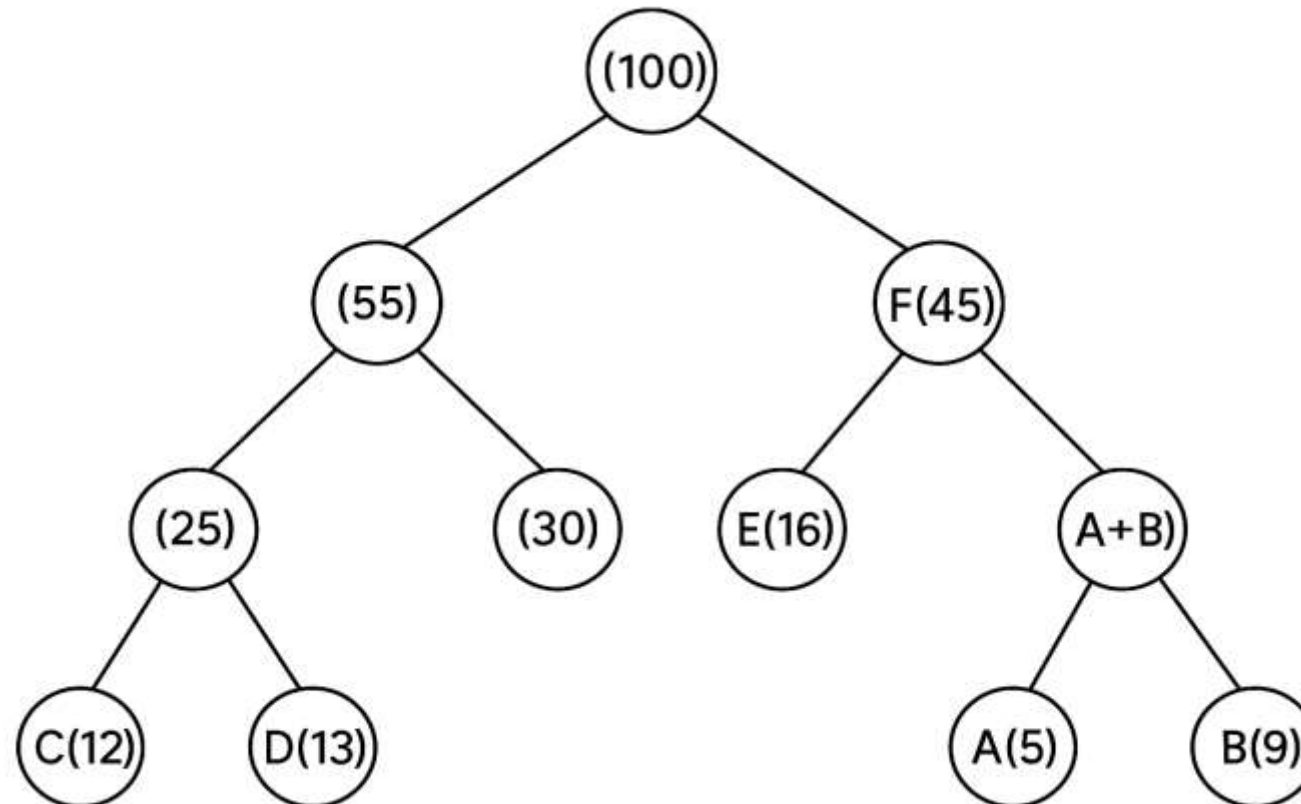
A:5, B:9, C:12, D:13, E:16, F:45

Step 1: Combine A and B → Node(14)

Step 2: Combine C and D → Node(25)

Step 3: Combine Node(14) and E → Node(30)

Step 4: Continue merging until one root node remains → Node(100)

Huffman Tree Visualization

Algorithm / Pseudocode

1. Insert all characters into a priority queue with their frequencies.

1. While more than one node exists, remove two smallest nodes and combine them.

1. Insert the new merged node back into the queue.

1. When one node remains, traverse the tree assigning '0' for left and '1' for right to generate codes.

Code Implementation

```java
import java.util.PriorityQueue;

class Node {
    char ch;
    int freq;
    Node left, right;

    Node(char ch, int freq) {
        this.ch = ch;
        this.freq = freq;
        this.left = null;
        this.right = null;
    }
}
```

```
class HuffmanCoding {
    // Print Huffman Codes from the root of Huffman Tree
    static void printCodes(Node root, String s) {
        if (root == null) return;

        // If it's a leaf node, print the character and its code
        if (root.left == null && root.right == null) {
            System.out.println(root.ch + ": " + s);
            return;
        }

        printCodes(root.left, s + "0");
        printCodes(root.right, s + "1");
    }
```

```
public static void main(String[] args) {
    char[] chars = {'A', 'B', 'C', 'D', 'E', 'F'};
    int[] freq = {5, 9, 12, 13, 16, 45};

    // Create a priority queue (min-heap) based on frequency
    PriorityQueue<Node> pq = new PriorityQueue<>((a, b) -> a.freq - b.freq);

    // Create leaf nodes for each character and add to the queue
    for (int i = 0; i < chars.length; i++) {
        pq.add(new Node(chars[i], freq[i]));
    }
```

```
while (pq.size() > 1) {
        // Extract the two nodes with minimum frequency
        Node left = pq.poll();
        Node right = pq.poll();

        // Create a new internal node with a combined frequency
        Node sum = new Node('$', left.freq + right.freq);
        sum.left = left;
        sum.right = right;

        // Add the new node to the priority queue
        pq.add(sum);
    }

    // Print the Huffman codes
    printCodes(pq.peek(), "");
  }
}
```

Output

Output (for given input):

F: 0

C: 100

D: 101

A: 1100

B: 1101

E: 111

Summary

- Huffman Coding efficiently compresses data by reducing total bits used.

- It uses frequency-based tree construction to assign unique variable-length codes.

- The algorithm's complexity is $O(n \log n)$ due to the use of a min-heap.

- It remains one of the most widely used lossless compression techniques in computing.

Practice Questions:

1 Kth Largest Element in an Array

Link: https://leetcode.com/problems/kth-largest-element-in-an-array/

Description:
Find the k-th largest element in an unsorted array using a min-heap or priority queue.

Concept Reinforced: Heap operations, element comparison, and top-k pattern.

Difficulty: ⬜ Medium

Practice Questions:

2 Top K Frequent Elements

Link: https://leetcode.com/problems/top-k-frequent-elements/?utm_source=chatgpt.com

Description:
Given an integer array, return the k most frequent elements using hash map + heap.

Concept Reinforced: Frequency counting, priority queue usage — similar to Huffman frequency logic.

Difficulty: ☐   Medium

Practice Questions:

3) Kth Smallest Element in a Sorted Matrix

Link: https://leetcode.com/problems/kth-smallest-element-in-a-sorted-matrix/?utm_source=chatgpt.com

Description:
Find the k-th smallest element in a sorted matrix using a min-heap or binary search approach.

Concept Reinforced: Heap-based selection and efficient traversal — extends Huffman Tree's greedy concept.

Difficulty: ⬜ Medium

Thanks