Huffman Coding Implementation

This project implements the Huffman Coding algorithm, a popular technique for data compression. Huffman Coding assigns variable-length codes to input characters, with shorter codes assigned to more frequent characters, resulting in an efficient encoding scheme.

How It Works

Huffman Coding is a greedy algorithm that constructs an optimal prefix-free binary tree (Huffman Tree) based on character frequencies in the input data. The steps to build the Huffman Tree and generate the corresponding codes are as follows:

Steps in Huffman Coding

1. **Calculate Frequency of Characters**:
    - Traverse the input text and calculate the frequency of each character. Store these frequencies in a map.
2. **Create Leaf Nodes**:
    - For each character in the frequency map, create a leaf node in the Huffman Tree. These nodes are stored in a priority queue (min-heap) where the node with the lowest frequency has the highest priority.
3. **Build the Huffman Tree**:
    - While there is more than one node in the priority queue:
        - Extract the two nodes with the lowest frequencies.
        - Create a new internal node with these two nodes as its children. The frequency of this new node is the sum of the frequencies of its children.
        - Insert the new node back into the priority queue.
    - The remaining node in the queue is the root of the Huffman Tree.
4. **Generate Huffman Codes**:
    - Traverse the Huffman Tree from the root, assigning a '0' for the left branch and a '1' for the right branch. The path from the root to each leaf node gives the Huffman Code for the corresponding character.
    - Store the generated codes in a map for easy access.
5. **Encode the Input Text**:
    - Replace each character in the input text with its corresponding Huffman Code from the map. This produces the encoded string.
6. **Decode the Encoded String**:
    - Traverse the Huffman Tree based on the bits in the encoded string to reconstruct the original text.