**HUFFMAN CODING**

**Team members:**

Keshav Upadhyaya D (01FB16ECS165)

Guruprasad M (01FB16ECS126)

Gurudas K Bhat (01FB16ECS125)

**Introduction:**

Huffman coding is a lossless data compression algorithm. The idea is to assign variable-length codes to input characters, lengths of the assigned codes are based on the frequencies of corresponding characters. The most frequent character gets the smallest code and the least frequent character gets the largest code.  
The variable-length codes assigned to input characters are prefix codes, means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not prefix of code assigned to any other character. This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bit stream.

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.

**The simplest Huffman tree construction algorithm uses a priority queue where the node with lowest frequency is given highest priority**:

1. Create a leaf node for each symbol and add it to the priority queue.

2. While there is more than one node in the queue:

Remove the two nodes of highest priority (lowest frequency) from the queue

Create a new internal node with these two nodes as children and with frequency equal to the sum of the two nodes' frequencies.

Add the new node to the queue.

3. The remaining node is the root node and the tree is complete.

**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 array when a leaf node is encountered.

We have implemented Huffman coding for both characters and strings.

**Working:**

The user can input a string or the data can be read from a file.

A table is formed containing the characters and the corresponding frequencies.

The table is passed to a createQueue function.

The queue is then passed to the createHuffmanTree function which creates the binary tree.