HUFFMAN BASED TEXT COMPRESSOR

Text compression is the process of reducing the size of a text file by encoding its content in a more compact form. This can be useful when storing or transmitting large amounts of text, as it allows the text to take up less space and be transferred more quickly.

There are several different techniques that can be used for text compression, including:

1. Huffman coding: This is a lossless compression algorithm that works by creating a prefix code for each character in a given input string, such that the length of the code for each character is inversely proportional to the character's frequency of occurrence in the input string.

2. Lempel-Ziv-Welch (LZW) algorithm: This is a lossless compression algorithm that works by replacing repeated sequences of characters in the input string with a reference to a previously occurring occurrence of the same sequence.

3. Run-length encoding: This is a lossless compression technique that works by replacing runs of repeated characters with a single instance of the character and a count of the number of times it appears.

4. Arithmetic coding: This is a lossless compression algorithm that works by representing the input string as a fraction, where the numerator represents

the position of the character in the string and the denominator represents the total number of characters.

5. Statistical compression: This is a lossless compression technique that works by analyzing the probability of occurrence of different characters in the input string and using this information to encode the string in a more compact form.

6. Text compression can be useful for a variety of applications, such as reducing the size of documents for storage or transmission, or for improving the efficiency of text-based communication systems.

Huffman Coding

Huffman coding is a lossless data compression algorithm. It works by creating a prefix code for each character in each input string, such that the length of the code for each character is inversely proportional to the character's frequency of occurrence in the input string. The resulting prefix code can be used to represent the input string in a more compact form, taking up less space than the original string.

The Huffman coding algorithm works by constructing a binary tree, with the most frequent characters at the root of the tree and the least frequent characters at the leaves. The tree is constructed such that the path from the root to any leaf represents the prefix code for that character.

To create the tree, the Huffman coding algorithm uses the following steps:

- Count the frequency of each character in the input string.
- Create a leaf node for each character, with the frequency of the character as its weight.
- Sort the leaf nodes in ascending order of weight.
- Repeatedly merge the two nodes with the lowest weight, creating a new internal node with the sum of the weights of the two nodes as its weight and the two nodes as its children.
- Continue merging nodes until there is only one node left, which is the root of the binary tree.
- Once the tree is constructed, the prefix codes for each character can be easily derived by traversing the tree and adding a '0' for each left child and a '1' for each right child.

Our Code:

Our code reads in a file specified by the user and stores the contents of the file in a string. It then counts the frequency of each character in the string and creates a leaf node for each character, with the frequency of the character as its weight. The leaf nodes are then sorted in ascending order of weight, and the algorithm repeatedly merges the two nodes with the lowest weight, creating a new internal node with the sum of the weights of the two nodes as its weight and the two nodes as its children. This process continues until there is only one node left, which is the root of the binary tree. The prefix codes for each character are then derived by traversing the tree and adding a '0' for each left child and a '1' for each right child. The resulting prefix codes can be used to compress the original input string.

Limitations in our project:

- We were unable to create a GUI for this program.
- We did not implement the decompression algorithm for this compressor; hence it does not restore the file to its original format.

Both restrictions occurred due to lack of time.