Name : Satyam Sharma
Project Report : Huffman Code
Course : CSC 220
Prof. Stephen Lucci

Files in the project

Driver.cpp (Contains the main method)

Opens the `Raven.txt` as an `ifstream` object.
`ifstream` allows `istreambuf_iterator` to iterate through the characters of the file and store them as a `string` called `content`.
Creates a `HuffmanCode` with `content` as the arguement.
Prints the binary string assinged to each character.
Opens the `Compressed_Raven.bin` as an `fstream` object, and prints the encoded binary string to this file.
Prints the size of both files using `tellg()` fuction.

HuffmanCode.h / HuffmanCode.cpp (Creates an instance of a Huffman Coding Problem)

```
//Constructor
HuffmanCode(string);
//Creates a heap of the size equal to total number of characters.
//Calls buildMap()
//Calls buildHeap();

void buildMap()
//Builds the map<character, frequency>. Where the frequency counts the number
//of occurence of the corresponding character in the string.

void buildHeap()
//Builds a Min Heap using the map<character, frequency>

void printHuffmanTrie();
//Prints the Heap as a table
```

HuffmanHeap.h / HuffmanHeap.cpp (Creates an instance of a Heap)

```
HuffmanHeap(int);
void insert(HuffmanNode*);
HuffmanNode* removeMin();
```

HuffmanNode.h / HuffmanNode.cpp (Creates an instance of a node)

```
HuffmanNode();
```

*Only the major functions are shown.

Raven.txt

The poem by Edgar Allen Poe

Compressed_Raven.bin

Output file containing the poem in Huffman binary

Application Analysis

Size of the input file `Raven.txt` : 6336 Bytes
Size of the output file `Compressed_Raven.bin` : 28349 Bytes
Compression Rate : 0.223 Bytes

```
Compression not successful
This is because the binary is being stored as string.
```

Expected size of the output file `Compressed_Raven.bin:` 28349 Bits ~ 3543.625 Bytes
(Estimated by counting the character 1s and 0s in the output file, assuming each character takes 1 bit)

Expected Compression Ratio : 1.788