POTD30.1. Encrypted Messaging With Huffman Trees

Download and Extract

An initial setup of files is provided to you via a shell script: Download potd-q30

Using a terminal, extract the initial files by running the shell script you just downloaded (you will need to navigate to the directory where you saved the file):

```
sh potd-q30.sh
```

Your files for this problem will be in the `potd-q30` directory.

The Story

Thanks to your heroic adventures during your time so far in CS 225, the FBI and CIA have requested your help. You are ready to accept their requests, Now, you must encrypt the messages you send to them and decrypt all messages they have sent to you. With your impressive understanding of trees from CS 225, you will utilize a trees provided to you by these agencies to help you encrypt and decrupt messages. The type of tree they have provided you is a huffman tree, which we will provide you with more information on below. Your heroic efforts will also help you accept offers from Gooogle and Facebook who have require you to send your replies to them using their own message encoding.

The Problem

You will utilize Huffman Encoding that is represented in a Huffman Tree in order to turn strings into a secret binary encrypted message, and turn binary encrypted messages into human readable strings. The Huffman Trees left paths represent a 0 in the binary message and taking a right oath represents a 1. The exampe below can help clarify.

Complete the member function `stringToBinary` and `binaryToString` in the `HuffmanUtils` file. More information is in the files

Example

In `HuffmanUtils.cpp`, a simple example of a huffman tree we can provide:

```
      6
  0 /   \ 1
  'E'    4
     0 /   \ 1
      2     2
   0 / \1 0/ \ 1
   'S' 'R''C''T'
```

The numbers in each node represent character frequencies. The number on each edge represent the direction taken on the tree. For instance to encode `S`, the value is `100`.

NOTE: In the huffman tree, the FBI, CIA, Facebook, and Google have all decided based on the functions you are writing to be optimal and not store the frequencies in each non-leaf node since your code will not need to utilize the frequency values.

Some Tips About Strings

If you are not sure how to analyze each letter of a string in your code, here are a few ways you can iterate over a string.

POTD 30

Total points: 0/1

Score: 0%

Question

Value: 1

History:

Awarded points: 0/1

Report an error in this question

Previous question

Next question

```
string str = "cs225agent";

// Prints out every chracter in the string on a new line
for (const char& c : str) {
    // NOTE: This method does not let you edit the inner elements in the string
    std::cout << c << std::endl;
}
```

```
string str = "cs225agent";

// Print out every character in the string
for (char& c: str) {
    std::cout << c << std::endl;
}
```

```
string str = "cs225agent";

// Another way to iteratore over characters in the string
for (size_t i = 0; i < str.length(); ++i) {
    std::cout << str[i] << std::endl;
}
```

Upload Solution

Drop files here or click to upload.

Only the files listed below will be accepted—others will be ignored.

Files

HuffmanUtils.cpp

not uploaded

Save & Grade | Save only