PA2b: Implementing Huffman's Algorithm

Overview

You will follow your plan from PA2a to implement Huffman's Algorithm using your frequency table. You will write a reflection documenting the time complexity of your code and what you learned.

Structure

This project is broken up into two parts:

1. Complete the huffman_encoder interface.
2. Answer the questions for the reflection.

Complete the Interface

New for part b: This project will build on top of the code you wrote in part a. Start by downloading the three new project files from Canvas (navigate to Files > SP20 > PA2 and download "huffman_encoder.cpp", "huffman_encoder.h", and "huffman_tests.cpp". Place these files in the same folder as "frequency_table.cpp" and frequency_table.h".

You can use any text editor to open the project source files. As with PA1, if you are using an IDE like Visual Studio or CLion, take care that you are **only** using it to edit the source files, **not** to compile and run the code. You must do that on the terminal using the provided makefile (covered in the next section).

When completing the interface, you may add new public or private functions or member variables. However, you may not modify or remove existing functions or member variables. Additionally, you may not change the way the tests are written. E.g. You may not modify the tests so that a new function you create is called in the test case. Do not create a main function. The testing framework will generate it automatically.

You may use anything in the C++ STL to complete the assignment. We encourage you to explore the new features added in the C++11, C++14, and C++17 standards.

huffman_encoder Interface Details

The huffman_encoder class represents a Huffman tree which can encode and decode files whose characters are contained in the tree.

Method	**Description**
huffman_encoder(const frequency_table &table);	Constructs a huffman_encoder object. This should build a Huffman tree using all the **non-zero** entries of the table (i.e. do not make leaves or generates codes for characters which are not present in the table or have a frequency of 0 in the table).
~huffman_encoder();	Destructs the huffman_encoder object. Should free any dynamically allocated memory. Can be left empty if no memory is dynamically allocated for the object.
std::string get_character_code(char c) const;	Returns the binary code representation of the character c as a string of binary digits. Returns an empty string if c is not present in the tree.
std::string encode(const std::string &file_name) const;	Encodes the contents of the file given by file_name by converting each character to its character code

	and concatenating them. If the file contains a letter not present in the tree, return an empty string.
std::string decode(const std::string &string_to_decode) const;	Decodes string_to_decode by converting each character code to it's ASCII character representation. If the file does not contain a valid Huffman encoding (i.e. characters other than '0' or '1') return an empty string.

File Details

To aid in completing the project, we have provided the new files on Canvas. Here is a short description of the files provided:

- huffman_encoder.h – Contains the declaration of the huffman_encoder class.
- huffman_encoder.cpp – Contains the definition of the member functions of the huffman_encoder class.
- huffman_tests.cpp – Contains sample Catch unit tests for the huffman_encoder class. You may add your own test cases to this file.

Testing

Testing uses the same process as PA2a. See the PA2a document for details.

Compiling and Running Locally

Compiling uses the same process as PA2a. See the PA2a document for details.

Compiling and Running Remotely

Compiling uses the same process as PA2a. See the PA2a document for details.

Reflection

Answer the following questions about assignments 2a and 2b:

a) What is the computational complexity of huffman_encoder (the constructor), get_character_code, encode, and decode?
b) What was the hardest part of the assignment?
c) What did you learn from this assignment?

Submission

Submit the following files on canvas:

- Your completed frequency_table.h file.
- Your completed frequency_table.cpp file.
- Your completed huffman_encoder.h file.
- Your completed huffman_encoder.cpp file.
- Your reflection in docx or pdf format.

When submitting, attach each file to your submission **separately**. **Do not** zip or tar the files.

Frequently Asked Questions

Why am I getting an error when I compile or run my code?

There are several things that could cause it. Before asking for help, try running `make clean` in your terminal to remove the build folder, then run `make` and `./build/test_pa2` again.

References

For details on opening a file in C++, see the documentation of the `std::ifstream` class:
https://en.cppreference.com/w/cpp/io/basic_ifstream