CSCI 4041, Fall 2018, Programming Assignment 6

Due Tuesday, 10/16/18, 10:30 AM (submission link on Canvas)

This is not a collaborative assignment; you must design, implement and test the solution(s) on your own. You may not consult or discuss the solution with anyone other than the course instructor or TAs. In addition, you may not include solutions or portions of solutions obtained from any source other than those provided in class. Obtaining or sharing solutions to any programming assignment for this class is considered academic misconduct. If you are not sure what this means, consult the class syllabus or discuss it with the course instructor.

Your shadow organization has realized that chicken nugget purchases are not a good method of relaying information, and are on the lookout for some other method of encoding their messages. You have received information that they will now be communicating with binary signals modulated into a local radio broadcast, using Huffman Coding rather than Morse Code to optimize the length of the message.

Since decoding a message encoded with an optimal Huffman Code requires knowledge about the content of the messages, this idea also seems questionable to you, but in order to relay that doubt to your superiors, you will need to create a program that can construct a Huffman Code. In particular, you just need to develop a function that takes a string representing the message and outputs the encoded string of 0s and 1s.

Download the PA6.py template from the course website, the four test messages message0.txt, message1.txt, message2.txt, and message3.txt, and their correctly encoded versions message0encoded.txt, message1encoded.txt, message2encoded.txt, and message3encoded.txt from **Canvas**. The tests are all contained in a single ZIP file, PA6tests.zip, which is available in the Files tab on Canvas The template contains a few functions and an implementation of a min-heap priority queue, but unlike in previous assignments, you are not required to use this code to solve the problem. So long as your code takes as input a string and outputs the optimal encoding of that string using the Huffman Coding algorithm in the textbook, the implementation and data structures used are up to you; the helper code is there only as a guide.

Requirements:

- You must download the template file PA6.py and edit the `huffmanEncode` function. You are permitted to use, edit, or delete any of the other code that occurs above the DO NOT EDIT BELOW THIS LINE marker.
- The code provided reads the test file to get an input string, uses a dictionary to track the frequency of each character within the string, and then constructs a min priority queue based on those frequencies. You must complete the functionality of huffmanEncode by constructing a Binary Huffman Code tree, and then using that tree to return a new string with each character in the input string replaced by the equivalent string of 0's and 1's based on the optimal Huffman Code.

- Your program must run without errors on the version of Python installed on the CSELabs machines, Python 3.5.2. (if you're testing this on CSELabs, you need to type python3 or idle3 instead of python or idle to start the correct version from the terminal)
- You are not permitted to use any built-in Python sorting routines like the `sorted()` function or the `.sort()` list method. You are also not allowed to use any Python function that asks for user input, such as `input()`, as this will break the grading script.
- You must implement the Huffman Coding algorithm found in Chapter 16 of the textbook. Any other algorithm will receive very little credit.
- However, note that while the textbook algorithm describes how to generate a binary tree that represents the encoding, your algorithm will have to actually take an input string and encode it using the constructed tree, outputting a string of 0's and 1's.
- This assignment will be graded automatically based on the number of test cases your program passes. There will be several secret test cases in addition to the ones included in the template to ensure you're not hard-coding in the solutions.
- This program will only run test cases until you fail one, avoiding the problem of having to scroll through test output to find the one broken test case.
- The grading breakdown for this assignment is as follows:
  - 30%: File runs without syntax errors
  - 70%: Passing test cases without breaking any requirements.
- The unedited template file already runs without syntax errors. This means that if your program causes syntax errors, you will get a better score by just submitting the original template unedited.
- Submit your edited PA6.py file to the Programming Assignment 6 link on Canvas before 10:30 AM on 10/16/18. No credit will be given for late submissions. Do not submit the test files.