CS 4310 - Algorithms - Fall 2017

HomeWork3

Given: October 17, 2017

Due: October 31, 2017

General Comments: *Show all your work, otherwise no partial credit. No credit without proper justifications. State all your assumptions. No Algorithm is complete without its time and space complexity. When presenting an algorithm, first indicate if it is similar to a well-known algorithm, second describe intuitively how the algorithm works (which may be supported by examples), third give its pseudo code and finally analyze its time and space complexity.* Always describe general idea of the algorithm before giving its pseudo-code. Do not reinvent the wheel, i.e., if a well-known algorithm can be modified to solve a problem efficiently, use that solution and clearly indicate the changes required. Do not unnecessarily complicate a solution, i.e., if a simple but efficient solution exists then we should use it. Finally, if you just write pseudo-code of a well-known algorithm without indicating how it applies or modified to the problem at hand, no credit will be given.

1. In this homework you will implement a version of zip/compress application using Huffman algorithm. Your code will be able to encode a text such that the letters which appeared more inside the text, need fewer bits to be stored. There are several steps to this process:
   - **Generate the input file**
     Your input file should be a simple text that you want to compress. For simplicity, only use lower case letters. this is an example of input file
   - **Build the Huffman tree**
     Use the pseudocode provided in page 442 of GT book for constructing the Huffman tree. In the algorithm you will be working with a priority queue. Use min heap using a size-balanced binary tree in its explicit representation (i.e., using pointers / references explicitly stored for the parent, leftchild and right child) for constructing the priority queue.
   - **Encode**
     Encode the input file based on the Huffman tree. Produce a map from letters to their binary representations. While traversing Huffman tree to generate the binary code for each letter, assign 1 while following the left branch and 0 while following

the right branch (unlike conventional Huffman algorithm which assigns 0 to left branches and 1 to right branches). Using the map you can encode the string into shorter binary presentation.

- **Decode**
  Use the Huffman tree to decode the encoded string. For decoding process, read each bit from the encoded string one at a time, and traverse the Huffman tree. If the bit is 1 move to the left otherwise move to the right. Continue traversing until you visit a leaf node which represents a character. Append the character to the decoded string and start with the next bit.

The output should be another text file automatically generated by your program. In your output file, first show the binary code that is assigned to each character by Huffman tree (one per line), then show the compressed version of input text by writing corresponding code of each letter in its position. Decoded string should be followed in the next line. this is an example of output file.

Testing

Run several tests (at least 20 runs) with different input strings (strings with different lengths and different number of distinct letters). Measure the running times of your implementation.

Report

Submit a report along with your code with the following contents:

- A brief explanation of the problem and Huffman algorithm.
- Show an example of the input string you used and the output generated by your code.
- Discuss how this approach helps to decrease the size of the input.
- Discuss about the running time of your implementation based on different inputs and compare it with the theoretically derived complexity.

Please provide a read me file containing the instructions for debugging and running your code and any other useful information which helps using your code easier.

For easy access to GT problems needed in this homework, click on this link.

General Instructions on submitting your homeworks.

- For programming assignments, submit a SINGLE zipped file of your source codes, scripts (to run your program if any) and a brief report along with a copy of a couple of sample executions of your solution to the class's Elearning. No need to say, but you should be using good conventions and programming practices in developing your programs [just in case you forgot, refresh them from some of the coding conventions etc links provided on the TopicsCovered page.]

- Use <hw#cs4310_yourlastname_mmddyy.{zip,ppt,doc,tex}> as the naming convention for your zipped, ppt, MS-Word, or LaTex files when submitting on Elearning. Replace '#' with the appropriate homework number.

- There will be significant point penalties for not following the naming convention above, good coding practices, submitting a different format of archive file or if your program does not run. Make sure it is a .zip and NOT another format (no .rar, .tar, .tar.gz, etc)

Any student may be asked to show and discuss his or her solution in class, so be ready with your presentation.