CS325: Analysis of Algorithms, Fall 2016

Group Assignment 3

Due: Tue, 11/15/16

Homework Policy:

1. Students should work on group assignments in groups of preferably three people. Each group submits to TEACH a zip file that includes their source code and their report. Each group, also, hands in a printed hard copy of the report in class or slides the hard copy under my door before the midnight of the due day. The hard copy will be graded, and the codes submitted to teach will be tested.
2. The goal of the homework assignments is for you to learn solving algorithmic problems. So, I recommend spending sufficient time thinking about problems individually before discussing them with your friends.
3. You are allowed to discuss the problems with other groups, and you are allowed to use other resources, but you *must* cite them. Also, you must write everything in your own words, copying verbatim is plagiarism.
4. *I don't know policy:* you may write "I don't know" *and nothing else* to answer a question and receive 25 percent of the total points for that problem whereas a completely wrong answer will receive zero.
5. Algorithms should be explained in plain english. You can use pseudocodes if it helps your explanation, but the grader will not try to understand a complicated pseudocode.
6. More items might be added to this list. ☺

In this assignment, we implement two data compression algorithms: run length encoding and Huffman codes. Of course, you can find implementations of these encoding algorithms on the internet, but, implementing them is easier and more fun.

Report (50%). In your report, give answers to the following questions.

(1) Give a short description of Huffman codes. What do you need to include in the encoded file for the decoder to be able to retrieve data?

(2) Give a short description of Run Length Encoding. What do you need to include in the encoded file for the decoder to be able to retrieve data?

(3) Give a short description of prefix-free codes. Give an example of a none prefix-free code that can be decoded in different ways.

(4) Give an example for which run length encoding is much better than Huffman codes, and an example for which the Huffman code is much shorter.

(5) The Huffman codes for a file composed of letters a, b, c, d, e, and f is $code(a) = 0$, $code(b) = 10$, $code(c) = 110$, $code(d) = 1110$, $code(e) = 11110$, and $code(f) = 11111$. What is the minimum length of this file?

Code(50%) We would like to encode a word that is composed of lower case letters. Consider the following methods:

(A) Simple encoding, that assigns a 5 bit code to each letter.

(B) Run length encoding, which is a sequence of pairs (ℓ, n), where ℓ is a letter specified in 5 bits, and n is the number of times that ℓ is repeated specified in 19 bits.

(C) Huffman codes, which are the prefix-free codes we saw in class.

The input to your program is a word of letters {'a', 'b',..., 'z'}, all lower case, in file "input.txt". The length of the word is at most 2^{19}.

Your output must be written in "output.txt". It should be three numbers, one per line: (1) the length of an unencoded file, (2) the length of the run length encoded file, and (3) the length of the compressed file by Huffman code (don't count the header size, the required size to specify the prefix-free codes).

Sample Input (1):
aaaaa

Sample Output (1):
25
24
5

Sample Input (2):
aabbacb

Sample Output (1):
35
120
11

Sample Input (3):
aaabbbcccddd

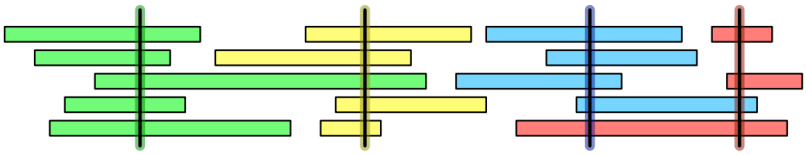
Sample Output (1):
60
96
24

Submission Each group submits to TEACH a zip file that includes their source code (*which must be just one file with name "hufffman.cpp", "huffman.java", or "huffman.py"*) and their report. This file can be submitted by any member of the group, but all names must be listed in the submitted report. Each group, also, hands in a printed hard copy of the report in class or slides the hard copy under my door before the midnight of the due day. The hard copy will be graded, and the codes submitted to teach will be tested.

Test your code with the sample test files (`http://web.engr.oregonstate.edu/~nayyeria/CS325/Fall16/hws/test3.zip`) before submitting them, to make sure there is no formatting error.

Do *not* submit a solution for the following problem, it is for practice.

Practice Problem. [1] Let X be a set of n intervals on the real line. We say that a set P of points stabs X if every interval in X contains at least one point in P. Describe and analyze an efficient algorithm to compute the smallest set of points that stabs X. Assume that your input consists of two arrays $X_L = [1..n]$ and $X_R = [1..n]$, representing the left and right endpoints of the intervals in X. Prove that your algorithm is correct.



A set of intervals stabbed by four points (shown here as vertical segments)

[1] This problem is from Jeff Erickson's lecture notes. Looking into similar problems from his lecture notes on greedy algorithms is recommended.