COMP 550.001 - Fall 2017
Assignment 5

Part A due: Friday, November 3, 2017 (start of class)
Part B due: Monday, November 6, 2017 (4:00 p.m.)

For Part A, you should submit a physical copy of your written homework at the start of class.
For Part B, you should submit a .tar.gz or .zip file with your solutions on Sakai.

Part A: Due Friday, November 3, 2017

Be sure to include a collaboration statement with your assignment, even if you worked alone.

[8 points] Problem 1: CLRS Exercise 13.4-7

Suppose that a node x is inserted into a red-black tree with RB-INSERT and then immediately deleted with RB-DELETE, as listed in the book. Is the resulting red-black tree the same as the initial red-black tree? Justify your answer.

[16 points] Problem 2: CLRS Problem 16-1 (Greedy Coin Changing)

Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer.

a. Describe a greedy algorithm to make change consisting of quarters, dimes, nickels, and pennies. Prove that your algorithm yields an optimal solution.

b. Suppose that the available coins are in the denominations that are powers of c, i.e., the denominations are c^0, c^1, ..., c^k for some integers $c > 1$ and $k \geq 1$. Show that the greedy algorithm always yields an optimal solution.

[20 points] Problem 3: CLRS Problem 16-2 (Greedy Task Scheduling)

Suppose you are given a set $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$ of tasks, where task τ_i requires c_i units of computation time to complete once it has started. You have one computer on which to run these tasks, and the computer can run only one task at a time. Let f_i be the ***finish time*** (also called completion time) of task τ_i, that is, the time at which task τ_i finishes processing. Your goal is to minimize the average finish time, that is, to minimize $(1/n)\sum_{i=1}^{n} f_i$.

For example, suppose there are two tasks, τ_1 and τ_2, with $c_1 = 3$ and $c_2 = 5$, and consider the schedule in which τ_2 runs first, followed by τ_1. Then $f_2 = 5$, $f_1 = 8$, and the average finish time is $(5 + 8)/2 = 6.5$. If task τ_1 runs first, however, then $f_1 = 3$, $f_2 = 8$, and the average finish time is $(3 + 8)/2 = 5.5$.

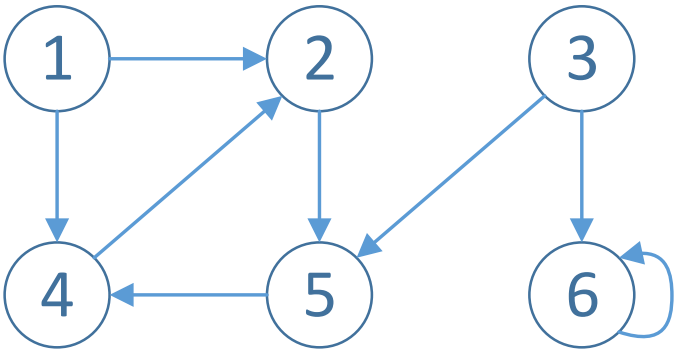
a. Give an algorithm that schedules the tasks so as to minimize the average finish time. Each task must run non-preemptively, that is, once task τ_i starts, it must run continuously for c_i units of time. Prove that your algorithm minimizes the average finish time, and state the running time of your algorithm.

b. Suppose now that the tasks are not all available at once. That is, each task cannot start until its ***release time*** r_i. Suppose also that we allow ***preemption***, so that a task can be suspended and restarted at a later time. For example, a task τ_i with execution time $c_i = 6$ and release time $r_i = 1$ might start running at time 1 and be preempted at time 4. It might then resume at time 10 but be preempted at time 11, and it might finally resume at time 13 and complete at time 15. Task τ_i has run for a total of 6 time units, but its running time has been divided into three pieces. In this scenario, τ_i's finish time is 15.

Give an algorithm that schedules the tasks so as to minimize the average finish time in this new scenario. Prove that your algorithm minimizes the average finish time, and state the running time of your algorithm.

[8 points] Problem 4: CLRS Exercise 22.2-1 (Breadth-First Search)

Show the d and π values that result from running breadth-first search on the directed graph below, using vertex 3 as the source.



[8 points] Problem 5: Exercise 22.3-5 part a (Depth-First Search)

Show that edge (u, v) is a tree edge or forward edge if and only if $u.d < v.d < v.f < u.f$.

Part B: Due Monday, November 6, 2017

[40 points] Text Encoding, inspired by CLRS Exercise 16.3-8

For this problem, you will implement both fixed-length and variable-length encoding schemes, and compare their compression rates on two different input files.

You are given two data files: `random.txt`, and `assignment.txt`. Both files contain the same number of characters, but with different distributions. The file `assignment.txt` contains the text of this assignment part B, whereas `random.txt` contains characters randomly selected from along upper- and lower-case letters, numbers, spaces, and the characters , . : - ˜( ) { } ” ’ @ # % ˆ & * + / \ \n.

You will implement three encoding schemes. For each scheme, encoding is done by looking up each character in a table, and replacing that character with a given sequence of bits. To keep it easy to compare and reason about the results, each encoder will generate a string of 0s and 1s, rather than actual binary output. This will make it easiest for you to compare the lengths of the encoded strings.

a. First, implement the fixed-length encoder in `FixedLengthEncoder`. The code to build the encoding table (and decoding table) are provided for you in `buildEncodingTable`. This builds two hashmaps: one for turning a string of characters to a string of 0s and 1s (`encodingTable`), and one for decoding a series of 0s and 1s into a string of characters (`decodingTable`).

You should fill in the implementations for the functions `encodeString` and `decodeString`. You are given a test function in `Main.java`, `testEncoder`, which will tell you if your implementation correctly encodes and decodes a simple message.

b. Next, fill in the unimplemented functions in the `HuffmanEncoder` class. You'll first need to implement `buildEncodingTable` and `buildEncodingTableFromTree` to build the encoding tree and the corresponding table from a provided dictionary of frequencies. Then, fill in the `encodeString` and `decodeString` functions.

Again, you can test your implementation using the function `testEncoder` in Main.

c. Now that you've implemented both a fixed-length encoder and a Huffman encoder, try them out! Uncomment the calls to `runEncoder` in Main. They are set up right now to encode/decode `syllabusText`, provided in `syllabus.txt`. In your writeup, list the lengths of the input string for `syllabusText`, as well as the length of the encoded strings using the fixed-length encoder, a Huffman encoder built using `assignment.txt`'s frequencies, and a Huffman encoder built using `syllabus.txt`'s frequencies. Which one is shortest? How much longer is the second shortest?

d. Finally, compare your encoders on a random input file: `random.txt`. Change the calls to `runEncoder` and `buildFrequencyTable` to use `randomText` instead. How long is each encoded string now? How do these compare to each other?