COMP 550.001 - Fall 2017
Assignment 4

Part A due: Monday, October 16, 2017 (start of class)
Part B due: Wednesday, October 18, 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 Monday, October 16, 2017

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

[20 points] Problem 1: CLRS Problem 9-4 (Alternative Analysis of Randomized Selection)

In this problem, we use indicator random variables to analyze the RANDOMIZED-SELECT procedure in a manner akin to our analysis of RANDOMIZED-QUICKSORT in Section 7.4.2.

As in the quicksort analysis, we assume that all elements are distinct, and we rename the elements of the input array A as $z_1, z_2, ..., z_n$, where z_i is the ith smallest element. Thus, the call RANDOMIZED-SELECT$(A, 1, n, k)$ returns z_k.

For $1 \leq i < j \leq n$ let

$$X_{ijk} = I\{z_i \text{ is compared with } z_j \text{ sometime during the execution of the algorithm to find } z_k\}.$$

a. Give an exact expression for $\mathrm{E}[X_{ijk}]$. (*Hint:* Your expression may have different values, depending on the values of i, j, and k.)

b. Let X_k denote the total number of comparisons between elements of array A when finding z_k. Show that

$$\mathrm{E}[X_k] \leq 2\left(\sum_{i=1}^{k}\sum_{j=k}^{n}\frac{1}{j-i+1} + \sum_{j=k+1}^{n}\frac{j-k-1}{j-k+1} + \sum_{i=1}^{k-2}\frac{k-i-1}{k-i+1}\right).$$

c. Show that $\mathrm{E}[X_k] \leq 4n$.

d. Conclude that, assuming all elements of array A are distinct, RANDOMIZED-SELECT runs in expected time $O(n)$.

[9 points] Problem 2: Exercise 8.3-2 (Stable Sorts)

a. Which of the following sorting algorithms are stable: insertion sort, merge sort, heapsort, and quicksort? Explain your answers.

b. Give a simple scheme that makes any comparison sort stable. How much additional time and space does your scheme entail?

[12 points] Problem 3: CLRS Exercise 11.4-3 (Hashing)

Consider an open-address hash table with uniform hashing. Give upper bounds on the expected number of probes in an unsuccessful search and on the expected number of probes in a successful search when the load factor is 3/4 and when it is 7/8. Explain how you got your answers.

[10 points] Problem 4: Exercise 12.2-1 (BST Review)

Suppose that we have numbers between 1 and 1000 in a binary search tree, and we want to search for the number 363. Which of the following sequences could *not* be the sequence of nodes examined? For each, say why or why not.

a. $2, 252, 401, 398, 330, 344, 397, 363$.

b. $924, 220, 911, 244, 898, 258, 362, 363$.

c. $925, 202, 911, 240, 912, 245, 363$.

d. $2, 399, 387, 219, 266, 382, 381, 278, 363$.

e. $935, 278, 347, 621, 299, 392, 358, 363$.

[9 points] Problem 5: Exercise 13.1-1 (Red-Black Trees)

In the style of Figure 13.1(a) in the book, draw the complete binary search tree of height 3 on the keys $1, 2, ..., 15$. Add the NIL leaves and color the nodes in three different ways such that the black-heights of the resulting red-black trees are 2, 3, and 4.

Part B: Due Wednesday, October 18, 2017

[40 points] Debugging Red-Black Trees

In this problem, you are given an *almost*-correct implementation of Red-Black Trees, and a suite of unit tests. In this problem, you need to find, document, and fix the bugs. You will submit this documentation in a report.pdf file, alongside the updated code that passes all of the unit tests.

Remember, you can only work with one other person on Part B. Make sure you do not discuss the bugs or fixes with more than one other person, and cite who you discussed it with in your readme file.

a. You should first look through the unit tests to study which tests are failing. For each failing test, you should include in the write-up the test case that is failing, indicating which function is being called, what the input is, what the expected output is, and what the actual output is. In the case of an error (e.g., if the program throws an exception) you should note the line of code that is failing rather than the actual output.

b. Now that you have described the bugs, you should investigate them! For each failing test case, you need to find bug fix(es) that make(s) the test pass (without causing any additional failures, ideally). Note that it is possible that more than one test is failing because of a single bug, or that multiple bugs cause a single test to fail. Include in your report, for each unit test, a description of the bug(s) and the fixes you had to make to get the test to pass (list the line numbers and both the original and new versions of the code).