CPS 2232 - Fall 2015 - Homework #5

Assigned: November 4th, 2015
Due: November 11th, 2015

Collaboration policy: The goal of homework is to give you practice in mastering the course material. Consequently, you are encouraged to collaborate with others. In fact, students who form study groups generally do better on exams than do students who work alone. If you do work in a study group, however, you owe it to yourself and your group to be prepared for your study group meeting. Specifically, you should spend at least 30–45 minutes trying to solve each problem beforehand. If your group is unable to solve a problem, it is your responsibility to get help from the instructor before the assignment is due. **You must write up each problem solution and/or code any programming assignment by yourself** without assistance, even if you collaborate with others to solve the problem. **You are asked to identify your collaborators.** If you did not work with anyone, you must write "Collaborators: none." If you obtain a solution through research (e.g., on the web), acknowledge your source, but write up the solution in your own words. **It is a violation of this policy to submit a problem solution that you cannot orally explain to the instructor.** No other student may use your solutions; this includes your writing, code, tests, documentation, etc. It is a violation of this policy to permit anyone other than the instructor and yourself read-access to the location where you keep your solutions.

Submission Guidelines: You have to submit your work on Blackboard by the due date. For each of the programming assignments you must **use the header template provided in Blackboard**. The header must contain, your name, course number, semester, homework number, problem number, and **list of collaborators** (if any). Your answers to questions that do not require coding must be included in this header as well. Your code must follow the Java formatting standards posted in Blackboard. Format will also be part of your grade.

To submit your assignments, you have to produce a capture file following the steps below. **The submission will not be accepted otherwise.**

1. Initiate an output capture.
   (On eve, enter `startrec yourname-hwknumber-problemnumber`)

2. Display the source code.
   (On eve, enter `cat filename.java`)

3. Compile the source code.
   (On eve, enter `javac filename.java`)

4. Execute the program.
   (On eve, enter `java filename`)

5. End capture and generate the output file.
   (On eve, enter `Control-D`)

Homework 5

Programming Assignment Grading Rubric:

The following rubric applies only to the programming assignment.

Program characteristic	Program feature	Credit possible	**Part 2**
Design 30%	Algorithm	30%	
Functionality 30%	Program runs without errors	20%	
	Correct result given	10%	
Input 15%	User friendly, typos, spacing	10%	
	Values read in correctly	5%	
Output 15%	Output provided	10%	
	Proper spelling, spacing, user friendly	5%	
Format 10%	Documentation: name, collaborators, header, etc.	5%	
	Clarity: comments, indentation, etc.	5%	
	TOTAL	100%	

1(20)	2(60)	3(20)	**TOTAL**(100)

Assignment:

The way that data is stored in a Binary **Search** Tree (BST) depends on the order in which the values are inserted. For example, if we insert the numbers $1, 2, 3$ in that order, the resulting tree has 1 in the root, 2 as a right child of the root, and 3 as a right child of 2. However, if we insert first 2, then 2 will be stored in the root, and 1 and 3 will be the left and right child of the root respectively. Moreover, not only the values are stored in different places but also the shape of the tree obtained is different. The first one is skewed whereas the second one is balanced. As a consequence, although both trees contain the same data, the worst-case cost of searching differs. The purpose of this homework is to highlight this striking difference in running time, creating a skewed BST and a (roughly) balanced BST, both with a large number of nodes, and measuring the execution time of searching on each.

1. **(20 points)** Estimate the asymptotic running time of searching in a skewed BST and a balanced BST. Justify your conjecture explaining which operations of the `BinarySearchTree` class (attached) you would use, and explain how do you obtain the overall running time from the running times of those operations. You can use asymptotic notation (big-O).

2. **(60 points)** Write a program to do the following.

   - Input an integer x. (Should work with "big" numbers.)
   - Create a completely-skewed BST S containing $1, 2, \ldots, x$.
   - Create a BST R containing x integers without repetitions generated at random. (To minimize the risk of repetitions, you can multiply the value returned by `random()` by a big number.) Given that the numbers are generated uniformly at random, the tree will likely be balanced.
   - Measure the time to search in S for a number that is not in the tree.
   - Measure the time to search in R for a new random number.
   - Display the time taken for each search.

Fill in a chart like the following with the times in nanoseconds measured. You may need to adjust the values of n according to your platform. That is, if your program takes too long to complete, or if you run out of memory, etc., reduce the range of n as needed. Your chart must have enough cells filled to be able to answer the following question.

	$n = 10^3$	$n = 10^4$	$n = 10^5$	$n = 10^6$
Skewed BST				
Balanced BST				

3. **(20 points)** How the results obtained compare with your conjecture? If the results differ from your conjecture, investigate the reason by looking carefully at the code of the `BinarySearchTree` class, and explain what happened.

4. **Extra credit** (up to 20%): Carry out the same experiments on the Java API class `TreeMap`. Compare the measurements with the skewed tree and random tree. Argue why the running time functions observed are (roughly) equal/different .