CS608 - Fall 2020 - Assignment #2

Assigned: September 17th, 2020
Due: September 25th, 2020 (Extra credit: December 1st, 2020)
NO LATE SUBMISSIONS.
Non-coding answers may be resubmitted once, after receiving feedback on the first attempt.

Collaboration policy: The goal of the assignment 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 study group is unable to solve a problem, it is your responsibility to get help from the instructor before the assignment is due.

For this assignment, you can form a team of up to three members. Each team must write up each problem solution and/or code any programming assignment without external assistance, even if you collaborate with others outside your team for discussions. You are asked to identify your collaborators outside your team. **If you did not work with anyone outside your team, 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 any member of the team cannot orally explain to the instructor.** No other student or team 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: Your team has to submit your work on Blackboard (no email) by the due date. Only one submission per team is necessary. For each class in the programming assignments you must use the header template provided in Blackboard. Make sure that you identify your team members in the header, and any collaborators outside your team, if none, write "none". Your code must follow the code formatting standards as posted in Blackboard. Format will also be part of your grade. To complete the submission, you have to upload two files to Blackboard: your source file and your class file. Your answers to questions that do not require coding must be included in the remarks section of the header. **The submission will not be accepted in any other format.**

Style and Correctness: Keep in mind that your goal is to communicate. Full credit will be given only to the correct solution which is described clearly. Convoluted and obtuse descriptions might receive low marks, even when they are correct. Also, aim for concise solutions, as it will save you time spent on write-ups, and also help you conceptualize the key idea of the problem.

Assignment #2 Grading Rubric

Coding:

Program characteristic	Program feature	Credit possible		
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%	Comments, name	5%		
	Indentation	5%		
	TOTAL	100%		

Non-coding:
Embedded in questions.

1(20)	2(50)	3(30)	**TOTAL**(100)	EC

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 assignment 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. **(50 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. **(30 points)** How the results obtained compare with your conjecture? Do they match or not? It is not enough to simply say: yes, they match. You have to justify your claim based on the running times measured. No points without justification. For instance, one of your answers could be something like the following.

   For the Skewed BST, we observe in row ... columns ... and ... that when the input size grows by ... digit the running time grows by ... digit. Likewise, from column ... to column ..., the running time grows by ... digit. Hence, the rate of growth is ...

   If the results differ from your conjecture, investigate the reason by looking carefully at the code of the `BinarySearchTree` class, and explain what happened. No points if you say they match and they do not or viceversa. No points deducted if you do not find the reason they do not match. Provide your answers in the remarks section of the code header.

4. **Extra credit:** 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 .