CPSC2108 Data Structures, Spring 2014

ASSIGNMENT 6 – DUE BEFORE MIDNIGHT, WEDNESDAY, MAY 7

Binary Search Tree Performance

The purpose of this assignment (worth 100 points) is to investigate the performance of a binary search
tree. It consists of the following two parts:

Part A: Binary Search Tree Class
Extend the BST class given in chapter 6 of the text book by adding two additional methods:

1. `public int height(Node<E> localRoot)`

Method returns the height of a BST instance

2. `public int size(Node<E> localRoot)`

Method returns the size of a BST instance in terms of its number of nodes

Part B: Experiments
After extending binary search tree class, write a tester program, class BSTTester, to run the following experiments on the BST class and record the results in a table.

Create a BST instance and insert 1000 randomly generated Integer
objects. Here is one way of doing this:

```
int n = 1000;
Random random = new Random();
BinarySearchTree<Integer> testBST = new
BinarySearchTree<Integer>();
for(int i = 0; i < n; i++){
   testBST.add(random.nextInt());
}
```

Record the time it takes to do this in nanoseconds using the `Java.lang.System.nanoTime()` method, which returns time elapsed in nanoseconds since some fixed but arbitrary time.

Record the height of the tree after inserting the 1000 items.

Record the size of the tree. (It may not be 1000 because the random number generator may produce duplicates.).

Run this experiment 10 times and note the average time to do the insertions, average height of the tree, and average size of the tree.

Repeat this experiment of getting the above average values in your BSTTester program for the following values of *n*: 2,000, 4,000, 8,000, 16,000, 32,000, 64,000, 128,000, 256,000, 512,000, and 1,024,000.

Record all information in a table as shown below:

Problem size *n*	Average tree size	Average tree height	Average insertion time
1000			
2,000			
4,000			
8,000			
16,000			
32,000			
64,000			
128,000			
256,000			
512,000			
1,024,000			

Answer the following question:
Did the growth in execution time with increasing problem size *n* meet your expectation?
To answer this question, create a spreadsheet with the data you have recorded and do a chart, using the spreadsheet chart tool, plotting the growth in height with increasing problem size, *n*. Write your conclusion based on this chart.
(Hint: Reading the section on Big O in chapter 2 of the text book should help).

What to submit:
A single zip file containing the following two items:

- ✓ The **.jar file** (including source code files BinarySearchTree.java and BSTTester.java) of your project.
- ✓ A **Word or .rtf document** with your answers to the question above. Include the table of execution data generated by your program and the growth chart.

Submit your zip file using CougarView Assignment 6 drop box before midnight, Wednesday, May 7. You MUST make sure your submitted .zip file can be unzipped and your assignment project runs successfully on a different computer.

BE SURE TO NOTE ANY CODE THAT IS NOT YOURS.

Grading Rubric

1	Good documentation (follow style given in textbook program listings) • Header documentation • Class and method documentation	10 points
2	Good programming style • Good use of white space • Good variable names • Good indentation	10 points
4	Program • Fulfilled all requirements • Good program logic and structure • Good use of data structures • Efficient algorithm design	55 points
5	Presentation of work • Table of execution data, line chart, answer to question	20 points
6	Submission instructions followed (see under	5 points