COSC 311 WINTER 2010 Project #2 Binary Search Tree

Distributed: February 18, 2010 **Due**: March 9, 2010

Brief description:
It is claimed that a binary search tree has average O(log n) performance for search, insertion, and delete of a single item because the average height of a BST with n elements is log n.

You will write and run experiments that examine that assertion.

Experiment #1

Create a very large BST by inserting 10^6 to it. The data type is long int. The data are generated randomly (use the maximum range).

Keep track of the height of the tree by updating a height field for each node upon each insertion.

For the following seven element insertions: k={10^3, $0.5*10^4$, 10^4, $0.5*10^5$,10^5, $0.5*10^6$, 10^6} output the time in milliseconds to insert and the height of the BST.

Run Experiment #1 five times, starting with different seed each time.

Assessing results
Make a (clean) spreadsheet showing n versus tree height and n versus insertion time, for the indicated seven elements.

Graph the results.

Write a short statement (250 words or fewer) summarizing the result of the experiment. The statement should include whether or not the O(log n) assertion is supported by your experiment.

Experiment #2

It has been claimed that BST become skewed (approach degeneracy) after multiple insertion, deletion pairs. This experiment will test that. We will assess 'skewness' very approximately using the balance at the root. The balance of a node is `right.height - left.height`.

2.a.

Create a large BST by inserting 10^4 elements to an initially empty BST. The data type is `int` (not `long`). Again, keep track of the height of each node.

Now insert a random element and delete a random element to the BST 10^6 times. N.B. the value of the inserted element and the value of the deleted element are different.
For the following seven element insertion/deletion pairs: k={10^3, $0.5*10^4$, 10^4, $0.5*10^5$,10^5, $0.5*10^6$, 10^6} output the following: time in milliseconds to insert, the height of the BST, and the balance at the root.

A deleted node is always replaced by the largest child in the left sub-tree (except when deleting the root, of course).

2.b.
Do the same as in 2.a., except that a deleted node is replaced by alternating between the largest child in the left sub-tree and the smallest child in the right sub-tree.

Run experiment 2.a. and 2.b. five times each.

Assessment of Experiment #2

Put the results in a table, graph the result, and make a statement (250 words or fewer) on the result of the experiments. 2.a. and 2.b. may be put on separate graphs, if appropriate. Your statement must include comment on the relative effect of the two different approaches to deletion.

Turn in:

Hardcopy of code
Output
Laboratory write-ups

Grade based on:

Meeting specifications:	65%
Code standards:	10%
Elegance:	10%
Write-up:	15%