Applications of Algorithms

Assignment 2

Binary Search Trees and Order-statistic Trees

(A) In the first part of this assignment you will investigate the claim in Theorem 12.4 (of the 3rd edition textbook) that a randomly built binary search tree on n distinct keys has expected height $\mathcal{O}(\log n)$.
In addition, you will investigate the time taken to build and destroy binary search trees, and the time to perform an Inorder-Tree-Walk.

(i) Code up the TREE-INSERT, TREE-DELETE and INORDER-TREE-WALK algorithms from Chapter 12.
The algorithms must be coded in either Java, C, or C++.
Code the algorithms following the methods given in the textbook. The algorithms must be coded from scratch and should not use any packages or any code that's not your own.

(ii) Run experiments to build binary search trees from randomly shuffled lists of keys by repeatedly calling TREE-INSERT. Record the height of each tree built. Run experiments for a range of values (the number of keys) and plot a graph of the **expected height** of a randomly built binary search tree.

(iii) For each binary search tree constructed in part (ii), record also the time taken to build the tree and plot a graph of the **expected run-time** to build a binary search tree.

(iv) For each binary search tree constructed in part (ii), destroy the tree by repeatedly calling TREE-DELETE on the root node, until the tree is empty. Record the time taken to destroy the binary search trees and plot a graph of the **expected run-time** to destroy a binary search tree.

(v) Run experiments to confirm the $\Theta(n)$ run-time for INORDER-TREE-WALK and plot a graph showing your results.

(B) In the second part of this assignment, you will code up functions on order-statistic trees. The order-statistic trees must be based on binary search trees (**not** Red-Black Trees).

(i) Code up the TREE-INSERT, TREE-DELETE, OS-SELECT and OS-RANK algorithms on a binary search tree whose nodes are augmented with the *size* attribute.
The algorithms must be coded in either Java, C, or C++.

Code the algorithms following the methods given in the textbook. The algorithms must be coded from scratch and should not use any packages or any code that's not your own.

Make sure your TREE-INSERT and TREE-DELETE algorithms maintain the *size* attribute of all nodes in the tree. Part of your assignment is to explain the changes to TREE-INSERT and TREE-DELETE that are required to maintain the *size* attribute of the nodes.

Note that there are some small changes required to OS-SELECT and OS-RANK as given in the textbook since the tree is not a Red-Black tree.

(ii) Run some experiments to confirm the expected run-times of the four algorithms and present your results using graphs. Compare the run-times for TREE-INSERT and TREE-DELETE on order-statistic trees to their run-times on binary search trees.

*****

You must submit the following to the AA moodle page by **Monday 13th October** at **23h00**:

(1) Your source code for all the algorithms in (A) and (B).

(2) A document with:

(a) all the graphs required in (A) and (B) and a description of how the graphs were obtained (range of dimensions, key values, number of trees of each size, etc.).

(b) An explanation, using pseudocode and sentences, of the changes to TREE-INSERT and TREE-DELETE that are required to maintain the *size* attribute of the nodes.