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 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.

(i) Code up the Tree-Insert and Tree-Delete algorithms from Chapter 12 of the textbook (in C, C++ or Java).

(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 different values of n (the number of keys) to illustrate the asymptotic growth of the height as n increases. To get the 'expected height' of a randomly built binary search tree, repeat the experiment a number of times for each value of n and take the average height. Plot the growth of the avergae height of the randomly built binary search trees.

(iii) For each binary search tree constructed in part (ii), record also the time taken to build the tree and plot your results on a graph.

(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 your results on a graph.

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

(i) Code up the Tree-Insert, Tree-Delete, OS-Search and OS-rank algorithms on a binary search tree whose nodes are augmented with the *size* attribute. (Use C, C++ or Java.)

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-Search and OS-rank in the textbook since the tree is not a Red-Black tree.

You must submit the following to the AA Moodle page by **Sunday 6 October** at **23h00**:

(1) Your source code for all the algorithms in (A) and (B) (coded in C, C++ or Java).

(2) A document with:

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

(ii) 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.