AVL and Red Black Tree Documentation

Konstantin Polyashenko

Test Computer Specs:

PC Brand - HP CQ61

Processor - 2.1GHz AMD Sempron Processor

Memory - 2GB

Run Time for AVL Tree:

Small to Big Array (1-1000) - 0.024941808 seconds runtime

Big to Small Array (1000-1) - 0.02359099 seconds runtime

Random Filled Array (1000 elements) - 0.020297699 seconds runtime

Small to Big Array (1-100000) - 0.085088616 seconds runtime

Big to Small Array (100000-1) - 0.088094192 seconds runtime

Random Filled Array (100000 elements) - 0.122105226 seconds runtime

Run Time for Red Black Tree:

Small to Big Array (1-1000) - 0.013477709 seconds runtime

Big to Small Array (1000-1) - 0.0173235 seconds runtime

Random Filled Array (1000 elements) - 0.010881003 seconds runtime

Small to Big Array (1-100000) - 0.082631487 seconds runtime

Big to Small Array (100000-1) - 0.081463464 seconds runtime

Random Filled Array (100000 elements) - 0.061719646 seconds runtime

The average time complexity for inserting into an empty AVL tree is 0(log n), where n is the amount of integers you are inserting. The worst case in that situation is 0(log n). The average time complexity for insertion into a red black tree is 0(logn) and the worst case for it is the same. This information is provided via bigocheatsheet.com.

Looking at the results of these tests i can conclude that the best is random generated array and the worst is small to big order array, for the vast majority of cases. I also used a code snippet to test the time it takes to run the tree insertion operation. I put it after i made the arrays so only the insertion is taken into account when i calculate time.

I tested my code with the jGrasp visual debugger to see if the trees were made in the proper fashion. This visual translation of the code creating the tree in real time and gave me a drawing of both trees.