Nicolas Ramirez
CMPT 225
David Mitchell
Assignment 2

Assignment 2 Report Part 1

In this experiment, we have constructed two Binary Search Trees (BST). The first has been made to be “tall and skinny” and the second has been made to be more “natural”. Both trees have been created with the same keys. Each key has been inserted in a different order for each tree in order to create a tree fitting the aforementioned description. For the tall and skinny tree, an ordered sequence of values was inserted into the tree. For the natural tree however, I took this ordered array of values and shuffled their arrangement within the array and then inserted each value into the natural tree. From here, our experiment is as follows: Perform a sequence of membership checks on each tree, accounting for the time it took to perform the check of each key in each tree. Then, we take the average of these checks for each tree and compare the average membership check time for each tree. In addition to this, we found the average depth of search for each of the membership checks. The following table outlines our results:

For trees with 50,000 nodes seeking 50,000 keys with randomized values between 0 and 49999 which are all present within the tree:

	Average Time taken for membership test [ms]	Average depth of search for membership test	Height
Binary Search Tree 1 (Tall and skinny)	0.140648	24999.5	49999
Binary Search Tree 2 (Natural)	0.00238566	245.922	562

As seen from the above table of values, the natural BST performs several orders of magnitude faster than the tall and skinny BST. This is due to the height difference between each tree which causes traversal and search times to be much longer for the tall and skinny BST than the natural BST.

Nicolas Ramirez
CMPT 225
David Mitchell
Assignment 2

Assignment 2 Report Part 2

In this experiment, we have constructed a Binary Search Tree (BST) and an AVL tree. Both trees have been created with the same keys with each key being inserted in the same order for each tree. From here, our experiment is as follows: Determine the average insertion time for each tree. Compare the height and average key depth of each tree. From here we will perform a sequence of membership checks on each tree, accounting for the time it took to perform the check of each key in each tree. Then, we find the minimum, maximum and mean time of these checks for each tree. In addition to this, we found the minimum, maximum and mean depth of search for each of the membership checks. The following table outlines our results:

For trees with 50,000 nodes and all nodes being searched are within both trees:

	AVL Tree	BST
Average Insertion Time [ms]	0.0009677	0.0228492
Height	13	9999
Average Key Depth	11.3631	4999.29
Minimum Membership Check Time [ms]	0	0
Mean Membership Check Time [ms]	0.000553481	0.0254065
Maximum Membership Check Time [ms]	0.022	0.1
Minimum Depth of Search	0	0
Mean Depth of Search	11.3631	4999.29
Maximum Depth of Search	13	9999

As seen from the above table of values, the AVL tree performs multiple orders of magnitude faster than the standard BST. This is a direct consequence of the fact that the AVL tree is self balancing and thereby ensures that the height of the tree is the lowest it can be for the given set of values. However, for the Binary search tree, the order of insertion of values can greatly vary the size and shape of the BST which causes traversal and search times to be much longer than the AVL tree.