Description

Problem 4.34 asks to write a method to generate an N-node random binary search tree with distinct keys 1 through N.

In order to achieve this, I implemented a simple Binary Search Tree class. In the class is where I included the method this problem asks for, which I called generateRandomBST.

My implementation first gets the integers 1 through N into an array list using a simple for loop. I then used the collections utility to shuffle the array list so the numbers would be in a random order. Finally I used my insert method to insert the keys into the tree using a loop.

I also included inorder and preorder traversals in the class to help test the generateRandomBST method. In the main class I tested the method using the traversals.

Complexity Analysis

The generateRandomBST method in my implementation has an average case time complexity of O(n log n), where n is the number of nodes in the binary search tree. This is because I used a loop to insert each element into the tree, and the insert method has an average case time complexity of O(log n), as the height of the tree will be logarithmic with respect to the number of nodes n. Thus, the for loop, which is O(n), and the insert method inside the for loop, multiply to give an average case time complexity of O(n log n).

However, the worst case time complexity is $O(n^2)$. This is because in the worst-case scenario, where the tree is completely unbalanced and looks like a linked list, the height of the tree will be n and each insertion operation will take O(n) time. Thus the two O(n) operations multiply to give a worst case time complexity of $O(n^2)$.