1. Worst-case algorithmic asymptotic complexity:
    a. Add a book – O(logn) : Because this is a self-balancing tree, the worst case complexity should be O(logn). In the worst case, you traverse to the bottom of the tree of height log(n). In my implementation, the complexity actually gets dominated by the updateHeights() function. For simplicity's sake, my implementation of this function checks and updates the height of every node in the tree. It is really only necessary to update the heights of the nodes on the path to the final insertion spot. I could achieve this by adding a new flag field to my node struct corresponding to whether it was on the path or not. So, my actual implementation has complexity O(nlogn), because it must traverse every node in the tree, but it could be improved upon to have worst-case complexity O(logn).
    b. Search for a book – O(logn): In the worst case, you must traverse to the bottom of a tree of height logn.
2. Average-case complexity:
    a. Add a book – O(logn) : this is the same as the worst case. On average, you will traverse the tree to a height of (log(n) / 2). For a complexity analysis, we can ignore the constant (1/2), so this simplifies to O(logn).
    b. Search a book – O(logn) : this is the same as the worst case. On average, you will traverse the tree to a height of (log(n) / 2). For a complexity analysis, we can ignore the constant (1/2), so this simplifies to O(logn).