Homework #5

Ian Chadwick
U90596416

Thursday, April 8, 2021 1:10 PM

1. AVL Tree [20 pt]

a. In a binary tree, a child is called an "only-child" if it has a parent node but no sibling, i.e. it is the only child node of its parent. Prove that for any AVL tree with n nodes ($n > 0$), the total number of only-children is at most $n/2$.

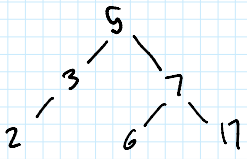
b. Is the statement "the order in which elements are inserted into an AVL tree does not matter, since the same AVL tree will be established following rotations" true? If yes, explain why. If not, give a counterexample.

a) In order for it to be a valid AVL tree, that means that the height difference between each sub-tree can be at most one by definition of an AVL tree. This means that if there are n nodes, the maximum number of leaves will be n/2 in order to preserve the AVL property.

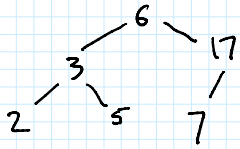
Since an only child can only be a leaf, that means that there can never be more than n/2 only children. This is true because two only children in a row would be an unbalanced chain that would be fixed with rotations.

b) False.

insert: 3, 5, 6, 7, 2, 17 results in



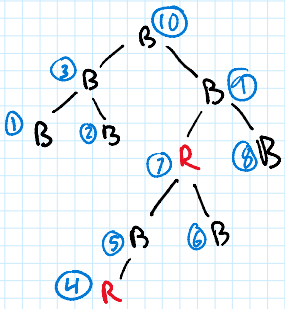
whereas insert 3, 2, 5, 17, 6, 7 results in



2. Red-Black Tree [10 pt]

A *post-order* traversal of a red-black tree on the set of numbers 1 to 10 gives the following colors: B B B R B B R B B B. Produce the tree.

In order for the post-order traversal to have the pattern BBBRBBRBBB

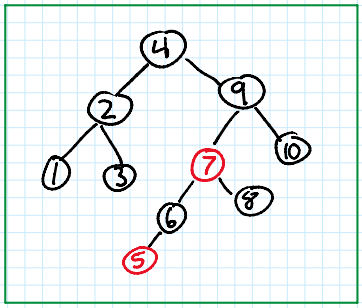


BBBRBBRBBB
1 2 3 4 5 6 7 8 9 10

order of traversal in blue

The tree would need to look like this,

Then in order to have the values 1–10 it would need to be arranged like this:



Satisfies BST properties

✓ ① all nodes to the left of every node is less than its parent node(s)

✓ ② all nodes to the right are more than their parent nodes.

✓ ③ all subtrees are also BSTs

Satisfies R/B Tree Properties

✓ ① Every node is red or black

✓ ② All Null leaves are black

✓ ③ If a node is red, both children are black

✓ ④ Every path to a NULL leaf passes through the same number of black nodes. (3 black nodes)

✓ ③ all subtrees are also BSTs

✓ Every path to a NULL leaf passes through the same number of black nodes. (3 black nodes)

3. Binary Search Tree [10 pt]

Devise an efficient algorithm that will transform any given binary search tree into any other binary search tree (with the same keys) using only ZIG and ZAG rotations. Your algorithm should run in time $O(n^3)$ or better.

① You have arguments of pointers to root node of 1) start tree and 2) target tree (with desired configuration)

② Base case would be if the tree is empty. Then the algorithm is done

③ If not empty, use search to find the root node of the target tree in the first tree.

④ Then use rotations to rotate that node to the root position of first tree

- while(!= root)
  - if (node is left child)
    - rotate node and parent right
  - else
    - rotate node and parent left

• This will preserve the BST properties, but make the two nodes "equal" in each tree (L/R)

⑤ This will create a root with two subtrees that have the same values but might be in a different configuration.

⑥ Recursively call the function on each node in the left and right subtrees until they are both in the same configuration as the target tree.

Time Complexity: since the height of the tree in the worst case is n the worst case for the rotations of the nodes would be $O(n)$ and we would have to do this for all n nodes, so the time complexity would be $O(n^2) + n \cdot O(n)$ to search in the worst case, which is $O(2n^2)$ or $O(n^2)$.