Homework Assignment 6

Balanced Trees

Assigned: Thursday, October 20, 2022

Due: Thursday, October 27, 2022

PROBLEM 1 5 POINTS

Complete all Homework Assignment 6 activities in the zyBook.

PROBLEM 2 10 POINTS

a) Implement insertion into an AVL tree in C++. You are provided with files `AVLTree.h` and `AVLTree.cpp` for a class AVLTree to use as a starting point. *You may add any additional data members and member functions, either private or public, to complete your implementation*. ***You do not need to implement any additional AVL tree functionality, such as search or node removal!***

Code snippet in `main()` function to test AVL tree rotations:

```
// Single rotate left
AVLTree b;
Node* bRoot = new Node(2);
b.insert(bRoot);
b.insert(new Node(1));
b.insert(new Node(3));
b.insert(new Node(4));
b.insert(new Node(5));
cout << bRoot->key << endl;
cout << bRoot->left->key << " " << bRoot->right->key
<< endl;
cout << bRoot->right->left->key << " " << bRoot-
>right->right->key << endl;

// Rotate left then right
AVLTree c;
Node* cRoot = new Node(4);
c.insert(cRoot);
c.insert(new Node(3));
c.insert(new Node(5));
c.insert(new Node(1));
```

```
c.insert(new Node(2));
cout << cRoot->key << endl;
cout << cRoot->left->key << " " << cRoot->right->key
<< endl;
cout << cRoot->left->left->key << " " << cRoot->left-
>right->key << endl;
```

Console output:
2
1 4
3 5
4
2 5
1 3

Note: an AVL tree would typically not have a function to insert a node without rebalancing. You will use this function in part (b) to simulate insertions into a regular BST without balancing. ***Submit your updated header file and implementation as*** **`AVLTree.h`** ***and*** **`AVLTree.cpp`**, ***respectively!*** (5 points)

b) For each $N \in \{10^2, 10^3, 10^4, 10^5\}$, perform the following 10 times: generate N random doubles in the interval [0, 1] and add them one at a time to both a BST without balancing and an AVL tree. (To simulate a BST without balancing, you can use your AVL tree implementation from (a) and use the function to insert a node without rebalancing.) After adding all N elements, record the height of both the BST without balancing and AVL tree.

For each N, compare the *minimum*, *average*, and *maximum* heights of the BST without balancing and the AVL tree over all 10 randomly generated sets of doubles. How much of a reduction in height is achieved by the AVL tree? How does the growth of these actual heights compare to what would be expected by best, average, and worst-case big O complexities? (5 points)