CSE250 Project #3

Sheng Liu

June 2021

1 Introduction

In this project, you will implement AVL tree, *i.e.*, a special kind of binary search tree (BST). AVL trees may be as foundations to implement other data structures, *e.g.*, maps (though most instantiations of C++ implement map using another type of tree). A BST is an AVL tree if it is balanced.

Balanced: A tree is balanced if the absolute difference between height of the left sub-tree and the right sub-tree is smaller than 1.

2 Functions

You will implement two functions, *i.e.*, `insert(), delete()` that perform insertion and deletion, such that an AVL tree remains an AVL tree (in other words, it is still balanced) after the insertion or deletion. There are six other functions, that you **may** choose to implement, they can be used as helper functions in `insert(), delete()`.

We will learn how insertion and deletion can be performed on AVL trees. The way we will learn rely on the part of the six helper functions. We will only test `insert(), delete()`. If your implementation does not rely on the six helper functions and satisfies the requirement that an AVL tree remains an AVL tree after the insertion or deletion, it is fine!

- `void AVLTree<T>::insert(const T &value)`: inserts a node that stores `value` to the AVL tree.
- `void AVLTree<T>::delete(const T &value)`: removes a node that stores `value`. You can assume that the node exists.
- `void AVLTree<T>::rotateLeft(Node<T> *& node)`: rotates a tree whose root is `node` to the left.
- `void AVLTree<T>::rotateRight(Node<T> *& node)`:rotates a tree whose root is `node` to the right.

- `void AVLTree<T>::rotateLeftRight(Node<T> *& node)`: performs left-right rotation on a tree whose root is `node`.
- `void AVLTree<T>::rotateRightLeft(Node<T> *& node)`: performs right-left rotation on a tree whose root is `node`.
- `void AVLTree<T>::find(const T &value)`: finds a node that stores `value` in the AVL tree. You can assume that the node exists.

3 Submission

You **only** need to submit the cpp file named "AVLTree.hpp" to AutoLab. Do **NOT** modify the code we provided in the cpp file or use (include) any additional function defined in C++ std namespace. As your projects will be graded automatically, modifying the provided code may cause our grading script to fail. If this happens, the grade will be **0**. Grades for the projects that use additional std functions will also be **0**.