CSCI 3333 Homework: AVL Trees

1 Height-Balanced

Problem 1. Give a BST of height 4 with each of the following properties (if one exists):

- Balanced and height-balanced.
- Balanced but not height-balanced.
- Height-balanced but not balanced.
- Neither balanced nor height balanced.

Problem 2. Call a node *almost-height-balanced* if the heights of its left and right subtrees differ by at most 2. Call a binary tree *almost-height-balanced* if all of its nodes are almost-height-balanced.

- Give a recurrence relation \( f(h) \) for the minimum number of nodes in an almost-height-balanced tree of height \( h \).
- Use repeated substitution to obtain a recursive inequality of the form \( f(h) > a \cdot f(h - b) \) where \( a, b \) are integers larger than 1.
- Find a closed form for the equality \( f(h) = a \cdot f(h - b) \).
- Give an asymptotic lower bound for \( f(h) \), knowing a closed form for \( f(h) = a \cdot f(h - b) \) and that \( f(h) > a \cdot f(h - b) \).
- Give an asymptotic upper bound for the height of an almost-height-balanced tree with \( n \) nodes.

2 Algorithm Examples

Problem 3. Draw a smallest tree that cannot be height-balanced with one rotation, but can be height-balanced with two rotations.

Problem 4. Draw the AVL tree that results from inserting the following elements in the given order: 1, 2, 3, 4, 5, 6. How many total rotations were done during the insertions?

Problem 5. Draw the AVL tree that results from inserting the following elements in the given order: 6, 1, 2, 3, 4, 5. How many total rotations were done during the insertions?

3 Algorithm Implementation

Problem 6. Implement a C++ function that returns whether a tree is height-balanced. You may assume that a C++ function with prototype `int height(Node* root)` has already been implemented.

Problem 7. Implement a C++ function that rotates a tree rightward around its root.