Exploration: AVL Tree Rotations

Introduction

The AVL tree is one of several existing types of self-balancing BST. The acronym AVL is derived from the initials of the names of the tree's inventors: Adelson-Velsky and Landis. There are many other kinds of self-balancing BST. Another popular one is the red-black tree but it is significantly more complicated to implement.

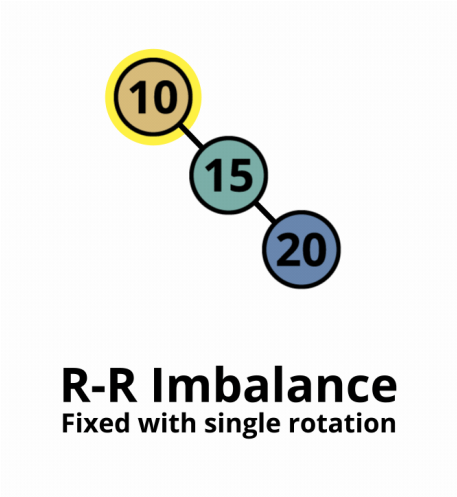
An AVL tree's operations include mechanisms to ensure that the tree always exhibits height balance. These mechanisms check the height balance of the tree after each insertion and removal of an element and perform rebalancing operations known as rotations whenever height balance is lost.

Rotations

A rotation is a simple operation that restructures an isolated region of the tree by performing a limited number of pointer updates that result in one node moving "upwards" in the tree and another node moving "downwards." Importantly, this is done in such a way as to preserve the BST property among all nodes in the tree.

Each rotation has a center and a direction. The center is the node at which the rotation is performed, and we can perform either a left rotation or a right rotation around this center node.
A left rotation moves nodes in a "counterclockwise" direction, with the center moving downwards and nodes to its right moving upwards.
Conversely, a right rotation moves nodes in a "clockwise" direction, with the center moving downwards and nodes to its left moving upwards.

For example, here's what a left rotation would look like:



Simple left rotation of an AVL tree

In the left rotation depicted above, the node with key 10 is the center of the rotation. This node moves downward in the tree, while its right child, the node with key 15, moves upward. In this simple example, the rotation restores height balance to the tree.

Sometimes, a single rotation like the one above, will be enough to restore height balance locally to the tree. Sometimes, however, a double rotation will be needed. Below, we'll explore the mechanics of these rotations in more detail.

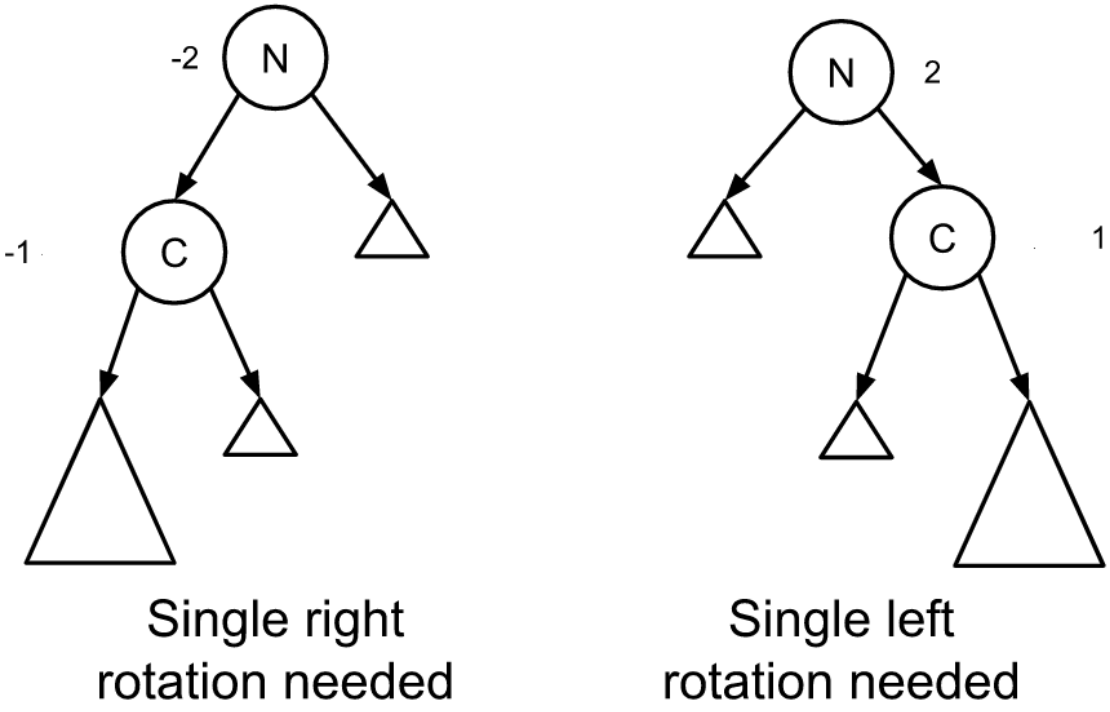
Determining the Rotations Needed

A rotation of some kind (i.e., single or double) will be needed any time an insertion into or removal from an AVL tree leaves the tree (temporarily) with a node whose balance factor is either –2 or 2. In other words, a rotation is needed when height balance is lost at a specific node in the tree. Let's call this node N.

If N has a balance factor of –2, this means N is left-heavy. If, on the other hand, N has a balance factor of 2, this means N is right-heavy. Regardless of the direction of N's heaviness, let's refer to the heavier of N's children as C.

The node C itself will have a balance factor of –1, or 1, which, respectively, means that C itself is left-heavy or right-heavy.

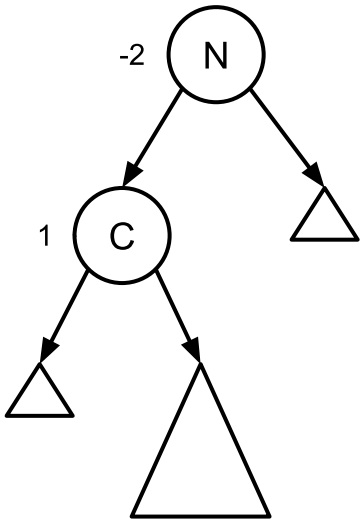
If N and C are heavy in the same direction (i.e., if the balance factor has the same sign at both N and C) then a single rotation is needed around N in the opposite direction as N's heaviness. In other words, these are the situations in which a single rotation is needed:



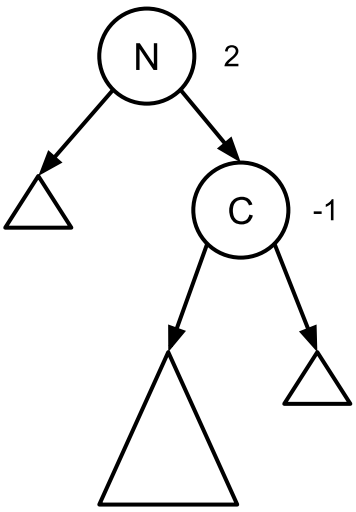
A depiction of an unbalanced node with an unbalanced child on the same side

If, on the other hand, N and C are heavy in opposite directions (i.e., if the balance factor has opposite signs at N and C), then a double rotation is needed.
- If N is left-heavy and C is right-heavy (i.e., N has a negative balance factor and C has a positive balance factor) then we first rotate left around C then right around N.
- If N is right-heavy and C is left-heavy (i.e., N has a positive balance factor and C has a negative balance factor) then we first rotate right around C then left around N.

In other words, these are the situations in which a double rotation is needed:



Double rotation needed: left around C, right around N



Double rotation needed: right around C, left around N

Trees requiring a double rotation to balance

Lecture


CS 261: AVL R ns
Oregon State
University
0:00 / 8:04
1x