Avl trees research

Stephen Derenski

Neumont College of Computer Science

CSC252 – Algo and Data Structures II

Binary Search Trees are great for storing ordered data but there are some downfalls to them. For example, if there is duplicated data on one side of the tree, then that side will be longer than the other, or if the root is too small of a value to split the data evenly. An AVL tree takes care of these by rotating the root of the tree to make sure that both sides are at a tolerance level of 1. That way both sides are balanced and stay efficient when you sort through the data.

For when a branch has a height larger than one from the other branch, it will rotate the root of the tree to balance the heights of the two branches. This means any subtree in the main as well. This allows for the tree to be optimized as much as possible.

There are four different kinds of rotations to balance the tree in an AVL tree. The first one is a left rotation. When a node is inserted into the right subtree of the right subtree, then the first node on the right of the root becomes the new root and the previous root becomes the new left branch of the new one. For reference, if the root of the tree was 13, the next right child was 15, and the last right child was 17. The 15 will become the root, the 13 will be the left child, and the 17 is the right child.

A right rotation occurs when a node is inserted on the left side of the tree. Meaning, that the next left node from the root becomes the new root and the old one becomes the right branch of the new root. For reference, if the root of the tree was 13, the next left child was 11, and the last left child was 9. The 11 will become the root, the 9 will be the left child, and the 13 is the right child.

A left-right rotation is when a node has been inserted into the right subtree of the left subtree. To move the node up to become the new root of the left subtree, it performs a left rotation and then a right rotation to rebalance everything. An example of this would be 15 as the root, 10 as the left child of the root, and then 13 being the right child of 10. 13 would become the child of the root, and 10 is the left child of 13. Then 13 would become the root of the tree with 10 as the left child and 15 as the right child.

A right-left rotation is an opposite. It is when a node has been inserted into the left subtree of the right subtree. To move the node up to become the new root of the right subtree, it performs a right rotation and then a left rotation. The last example of this will be that 15 is the root of the tree, with a right child of 20, and a left child of 17 on the 20. The 17 would become the parent of the 20, meaning 20 becomes the right child of 17. Then 17 would become the root of the tree with 15 being the left child and 20 staying as the right child.