AVL tree is a self-balancing binary search tree. It maintains the balance by ensuring the difference between the heights of the left and right subtrees of any node is at most one. This guarantees that the worst-case time complexity of search, insertion, and deletion operations is logarithmic (O(log n)).

AVL trees use tree rotations to maintain balance. There are two main types of rotations: left rotation and right rotation. A left rotation is performed when the right subtree of a node is heavier than the left subtree by more than one. A right rotation is performed when the left subtree of a node is heavier than the right subtree by more than one.

When inserting a node into an AVL tree, we first insert the node like a normal binary search tree. Then, we travel up the tree from the inserted node and check if the balance condition is still satisfied. If the balance condition is violated, we perform a rotation to restore the balance.

Deleting a node from an AVL tree is similar to deleting a node from a binary search tree. However, we also need to check the balance condition after deletion and perform rotations if necessary.