AVL Tree Cheatsheet

Overview

- AVL tree is a self-balancing binary search tree.
- It is named after its inventors, Adelson-Velsky and Landis.
- In an AVL tree, the heights of the two child subtrees of any node differ by at most one.

Operations

Insertion

```
def insert_node(node, key):
    if not node:
        return Node(key)
    elif key < node.key:
        node.left = insert_node(node.left, key)
    else:
        node.right = insert_node(node.right, key)

    node.height = 1 + max(get_height(node.left), get_height(node.right))

    balance = get_balance(node)

    if balance > 1 and key < node.left.key:
        return right_rotate(node)

    if balance < -1 and key > node.right.key:
        return left_rotate(node)

    if balance > 1 and key > node.left.key:
        node.left = left_rotate(node.left)
        return right_rotate(node)

    if balance < -1 and key < node.right.key:
        node.right = right_rotate(node.right)
        return left_rotate(node)

    return node
```

Deletion

```
def delete_node(root, key):
    if not root:
        return root

    elif key < root.key:
        root.left = delete_node(root.left, key)
```

```
    elif key > root.key:
        root.right = delete_node(root.right, key)

    else:
        if root.left is None:
            temp = root.right
            root = None
            return temp

        elif root.right is None:
            temp = root.left
            root = None
            return temp

        temp = get_min_value_node(root.right)
        root.key = temp.key
        root.right = delete_node(root.right, temp.key)

    if root is None:
        return root

    root.height = 1 + max(get_height(root.left), get_height(root.right))

    balance = get_balance(root)

    if balance > 1 and get_balance(root.left) >= 0:
        return right_rotate(root)

    if balance < -1 and get_balance(root.right) <= 0:
        return left_rotate(root)

    if balance > 1 and get_balance(root.left) < 0:
        root.left = left_rotate(root.left)
        return right_rotate(root)

    if balance < -1 and get_balance(root.right) > 0:
        root.right = right_rotate(root.right)
        return left_rotate(root)

    return root
```

Traversal

In-order Traversal

```
def in_order_traversal(node):
    if node:
        in_order_traversal(node.left)
        print(node.key)
        in_order_traversal(node.right)
```

Time Complexity

- Insertion: O(log n)
- Deletion: O(log n)
- Traversal: O(n)

Resources

- AVL Tree Wikipedia
- GeeksforGeeks: AVL Tree
- AVL Tree Visualization