AVL Tree Rotations in C - Simple Explanation with Example

Understanding AVL Tree Rotations - Detailed Explanation

This handout explains the logic and variables used in AVL Tree rotations with simple examples. It helps understand how and why rotations are performed in C programming.

Right Rotation (RR) - Step-by-Step

When?

Used when the LEFT subtree of the LEFT child is heavier (Left-Left imbalance).

Imagine inserting 30 -> 20 -> 10, creating:

```
    y = 30
   /
  x = 20
 /
T2 = 10
```

This is unbalanced at node 30, so we do a right rotation.

```
struct Node* rightRotate(struct Node* y) {
    struct Node* x = y->left;
    struct Node* T2 = x->right;

    x->right = y;
    y->left = T2;

    y->height = max(height(y->left), height(y->right)) + 1;
    x->height = max(height(x->left), height(x->right)) + 1;

    return x;
}
```

Variables Explained:

- y: Unbalanced node (30)

AVL Tree Rotations in C - Simple Explanation with Example

- x: Left child of y (20), becomes new root

- T2: Right child of x, reattached as left of y

After rotation:

```
    x=20
   /   \
 T2     y=30
```

Left Rotation (LL) - Step-by-Step

When?

Used when the RIGHT subtree of the RIGHT child is heavier (Right-Right imbalance).

Imagine inserting 10 -> 20 -> 30, creating:

```
 x = 10
   \
    y = 20
      \
       T2 = 30
```

Unbalanced at node 10, so we perform a left rotation.

```
struct Node* leftRotate(struct Node* x) {
    struct Node* y = x->right;
    struct Node* T2 = y->left;

    y->left = x;
    x->right = T2;

    x->height = max(height(x->left), height(x->right)) + 1;
    y->height = max(height(y->left), height(y->right)) + 1;

    return y;
```

```
}
```

Variables Explained:

- x: Unbalanced node (10)
- y: Right child of x (20), becomes new root
- T2: Left child of y, reattached as right of x

After rotation:

```
    y=20
   /   \
 x=10   T2=30
```

Summary Table

Rotation	When to Use	New Root	Steps
RR	Left-Left Case	x	Right Rotate
LL	Right-Right Case	y	Left Rotate

Tips for Students

- Understand the pattern of imbalance before applying rotation.
- Always update height after rotation.
- Practice with different insertion orders to visualize tree shapes.