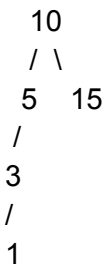
Here are examples for each AVL tree balancing case:

1. Left-Left Imbalance (Single Rotation)

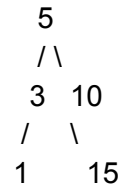
Example:

- Start with a balanced tree: 10 as root, 5 as left child, 15 as right child
- Insert 3 as left child of 5
- Insert 1 as left child of 3
- Now the tree is imbalanced with left side too heavy (height difference of 2)

Before rotation:



After single right rotation:

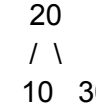


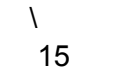
2. Left-Right Imbalance (Double Rotation)

Example:

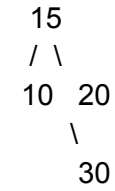
- Start with: 20 as root, 10 as left child, 30 as right child
- Insert 15 as right child of 10
- Now node 20 has a left-right imbalance

Before rotations:





After left rotation on 10, then right rotation on 20:

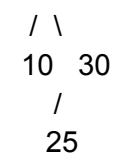


3. Right-Left Imbalance (Double Rotation)

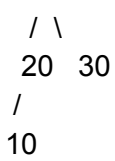
Example:

- Start with: 20 as root, 10 as left child, 30 as right child
- Insert 25 as left child of 30
- Now node 20 has a right-left imbalance

Before rotations:



After right rotation on 30, then left rotation on 20:



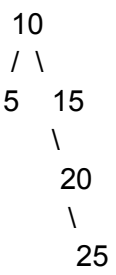
4. Right-Right Imbalance (Single Rotation)

Example:

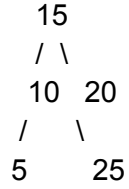
- Start with a balanced tree: 10 as root, 5 as left child, 15 as right child

- Insert 20 as right child of 15
- Insert 25 as right child of 20
- Now the tree is imbalanced with right side too heavy

Before rotation:



After single left rotation:



Practical Applications:

Each of these rotations is triggered by specific insertion or deletion operations:

- **Left-Left Case**: Often happens when inserting a sequence of decreasing values into an initially empty tree (like 50, 40, 30...)
- **Right-Right Case**: Often happens when inserting a sequence of increasing values (like 10, 20, 30...)
- **Left-Right Case**: Can occur when inserting values that zigzag (like 50, 30, 40)
- **Right-Left Case**: Can occur when inserting values that zigzag in the opposite direction (like 10, 30, 20)

These rotations ensure the tree remains balanced so that operations maintain O(log n) time complexity rather than degrading to O(n).