

An **AVL Tree** is a self-balancing **Binary Search Tree (BST)** where the height difference (balance factor) between the left and right subtrees of any node is **at most 1**.

Balance Factor (BF)

$BF = \text{Height of Left Subtree} - \text{Height of Right Subtree}$

- $BF = 0, 1, \text{ or } -1 \rightarrow \textit{Balanced}$
 - $BF > 1 \text{ or } BF < -1 \rightarrow \textit{Unbalanced}$
-

Steps for Insertion in an AVL Tree

1. **Insert the new node like in a normal BST**
 2. **Update heights of affected nodes**
 3. **Check balance factors from the inserted node up to the root**
 4. **Perform rotations if necessary**
-

Types of Rotations for Balancing

Case 1: Left-Left (LL) – Right Rotation

Occurs when a node is **left-heavy** ($BF > 1$) and the inserted node is in the **left subtree**.

Fix: Perform a **Right Rotation (Single Rotation)**

Example: Insert 10 into this tree

```
  30
 /
20
/
10
```

Right Rotate (at 30) →

```
  20
 / \
10 30
```

Case 2: Right-Right (RR) – Left Rotation

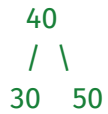
Occurs when a node is **right-heavy** ($BF < -1$) and the inserted node is in the **right subtree**.

Fix: Perform a **Left Rotation (Single Rotation)**

Example: Insert 50 into this tree



Left Rotate (at 30) →



Case 3: Left-Right (LR) – Left Rotation + Right Rotation

Occurs when a node is **left-heavy** ($BF > 1$) but the inserted node is in the **right subtree** of the left child.

Fix:

1. Perform **Left Rotation** at the left child
2. Perform **Right Rotation** at the unbalanced node

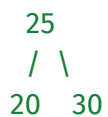
Example: Insert 25 into this tree



Step 1: Left Rotate at 20 →



Step 2: Right Rotate at 30 →



Case 4: Right-Left (RL) – Right Rotation + Left Rotation

Occurs when a node is **right-heavy** ($BF < -1$) but the inserted node is in the **left subtree** of the right child.

Fix:

1. Perform **Right Rotation** at the right child
2. Perform **Left Rotation** at the unbalanced node

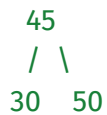
Example: Insert 45 into this tree



Step 1: Right Rotate at 50 →



Step 2: Left Rotate at 30 →



Example

Let's insert [50, 30, 70, 20, 40, 60, 80] step by step:

Insert **50** → Root

50

Insert **30** → Left of 50

50
/
30

Insert **70** → Right of 50

```
  50
 /  \
30   70
```

Insert **20** → Left of 30

```
  50
 /  \
30   70
/
20
```

Insert **40** → Right of 30

```
  50
 /  \
30   70
/  \
20  40
```

Insert **60** → Left of 70

```
  50
 /  \
30   70
/  \  /
20 40 60
```

Insert **80** → Right of 70 (Balanced)

```
  50
 /  \
30   70
/  \  /  \
20 40 60 80
```

Another example:

[(20:O), (40:S), (60:T), (80:R), (89:N), (70:E), (30:T), (10:N), (33:A), (31:H), (24:R), (32:E)]

Insert 20

20

Insert 40

```
20
 \
 40
```

$BF(20) = 1 - 0 = 0$ (Balanced)

Insert 60

```
20
 \
 40
 \
 60
```

$BF(40) = 0 - 1 = 0$

$BF(20) = 0 - 2 = -2$

- Imbalance at node 20

RR Rotation

- Rotate at node 20

```
40
 /  \
20   60
```

$BF(40) = 1 - 1 = 0$ (Balanced)

$BF(60) = 0 - 0 = 0$ (Balanced)

$BF(20) = 0 - 0 = -2$ (Balanced)

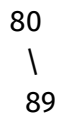
Insert 80

```
40
 /  \
20   60
      \
       80
```

- Balanced at all nodes

Insert 89

```
40
 /  \
20   60
      \
       \
```



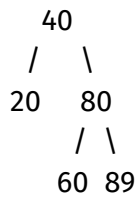
$BF(80) = 0 - 1 = -1$ (Balanced)

$BF(60) = 0 - 2 = -2$ (Unbalanced)

- Imbalance at node 60

RR rotation

- Rotate on node 60

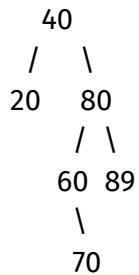


$BF(80) = 1 - 1 = 0$ (Balanced)

$BF(60) = 0 - 0 = 0$ (Balanced)

$BF(40) = 1 - 2 = -1$ (Balanced)

Insert 70



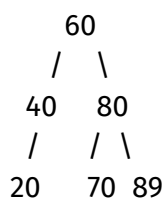
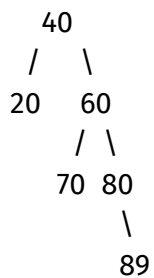
$BF(80) = 2 - 1 = 0$ (Balanced)

$BF(60) = 0 - 1 = -1$ (Balanced)

$BF(40) = 1 - 3 = -2$ (Unbalanced)

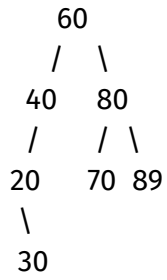
- Imbalance at node 40

RL rotation



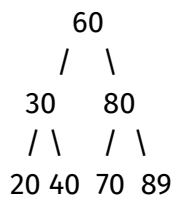
- **Balanced**

Insert 30

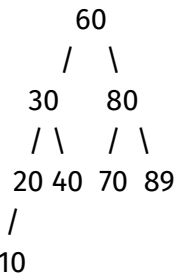


- Imbalance at node 40

LR rotation

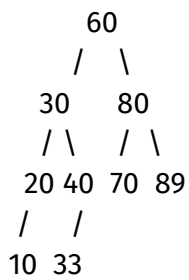


Insert 10

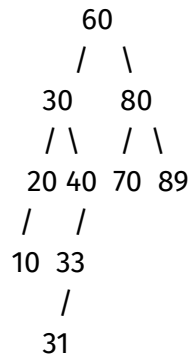


- Still balanced

Insert 33

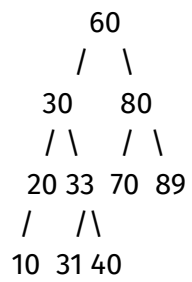


Insert 31

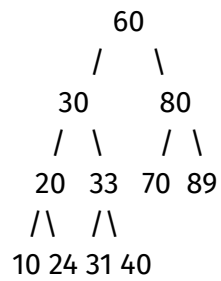


- Imbalance at node 30

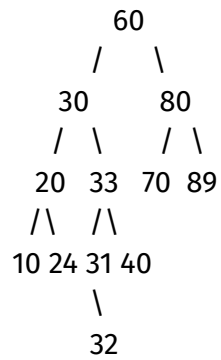
LL rotation



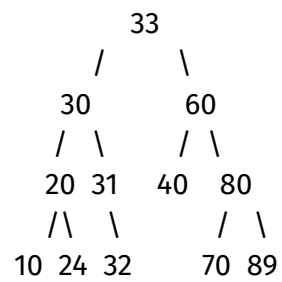
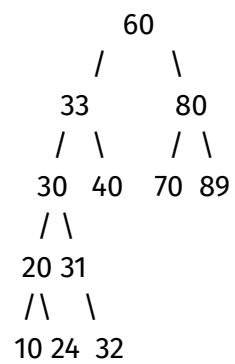
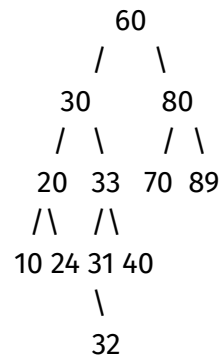
Insert 24



Insert 32



- Imbalance at node 60



- **Balanced**