AVL trees are a special type of binary search tree that keep themselves balanced. They're named after their inventors Adelson-Velsky and Landis. Here's a simple explanation of AVL trees:

What is an AVL Tree?

An AVL tree is like a regular binary search tree, but with an extra rule: the heights of the left and right subtrees of any node can't differ by more than one[1][2]. This difference is called the balance factor. The balance factor can only be -1, 0, or +1[3].

Why Use AVL Trees?

AVL trees help solve the problem of regular binary search trees becoming unbalanced and slow. By keeping themselves balanced, AVL trees ensure that operations like searching, inserting, and deleting always take a reasonable amount of time (specifically, O(log n) time)[2].

How AVL Trees Work

1. Balance Factor: For each node, we calculate its balance factor by subtracting the height of its right subtree from the height of its left subtree[3].
2. Self-Balancing: When we add or remove nodes, the tree might become unbalanced. If this happens, the tree performs special operations called rotations to fix itself[4].

Types of Rotations

There are four types of rotations:

1. Left Rotation (LL Rotation): When a node becomes too heavy on the right side[4].
2. Right Rotation (RR Rotation): When a node becomes too heavy on the left side[4].
3. Left-Right Rotation (LR Rotation): A combination of a left rotation followed by a right rotation[4].
4. Right-Left Rotation (RL Rotation): A combination of a right rotation followed by a left rotation[4].

Examples of Insertions and Rotations

Let's look at some examples of inserting nodes and the rotations that might be needed:

1. Simple Insertion (No Rotation Needed):
   Start with: 5
   Insert 3:
   5
   /
   3
   No rotation needed, tree is balanced.
2. Left Rotation Example:
   Start with:
   5

   7
   Insert 8:
   5

   7

   8
   This triggers a left rotation:
   7
   /
   5 8
3. Right Rotation Example:
   Start with:
   7
   /
   5
   Insert 3:
   7
   /
   5
   /
   3
   This triggers a right rotation:
   5
   /
   3 7
4. Left-Right Rotation Example:
   Start with:

7
/
3
Insert 5:
7
/
3

5
This triggers a left-right rotation:
5
/
3 7

5. Right-Left Rotation Example:
   Start with:
   3

   7
   Insert 5:
   3

   7
   /
   5
   This triggers a right-left rotation:
   5
   /
   3 7

After each insertion, the tree checks its balance and performs rotations if needed to stay balanced[7].
Remember, the goal of all these rotations is to keep the tree balanced, which means keeping the height difference between left and right subtrees to no more than one for every node[12].

Example 1: Simple Insertion (No Rotation)
Initial:
5
Action: Insert 3

Result:
5
/
3
Rotation: None
Final:
5
/
3
Example 2: Left Rotation
Initial:
5

7
Action: Insert 8
Result:
5

7

8
Rotation: Left
Final:
7
/
5 8
Example 3: Right Rotation
Initial:
7
/
5
Action: Insert 3
Result:
7
/
5
/
3
Rotation: Right

Final:
5
/
3 7
Example 4: Left-Right Rotation
Initial:
7
/
3
Action: Insert 5
Result:
7
/
3

5
Rotation: Left-Right
Final:
5
/
3 7
Example 5: Right-Left Rotation
Initial:
3

7
Action: Insert 5
Result:
3

7
/
5
Rotation: Right-Left
Final:
5
/
3 7

Example 6: Multiple Insertions (No Rotation)
Initial:
5
Action: Insert 3, then 7
Result:
5
/
3 7
Rotation: None
Final:
5
/
3 7
Example 7: Left Rotation after Multiple Insertions
Initial:
5
/
3 7
Action: Insert 8, then 9
Result:
5
/
3 7

8

9
Rotation: Left
Final:
7
/
5 8
/
3 9
Example 8: Right Rotation after Multiple Insertions
Initial:
5
/
3 7

Action: Insert 2, then 1
Result:
5
/
3 7
/
2
/
1
Rotation: Right
Final:
3
/
2 5
/
1 7
Example 9: Left-Right Rotation after Multiple Insertions
Initial:
7
/
3
Action: Insert 2, then 5
Result:
7
/
3
/
2 5
Rotation: Left-Right
Final:
5
/
3 7
/
2
Example 10: Right-Left Rotation after Multiple Insertions
Initial:
3

7
Action: Insert 8, then 5
Result:
3

7
/
5 8
Rotation: Right-Left
Final:
5
/
3 7

8

Example 11: Double Rotation (Left-Right)
Initial:
8
/
4
Action: Insert 6
Result:
8
/
4

6
Rotation: Left-Right
Final:
6
/
4 8
Example 12: Double Rotation (Right-Left)
Initial:
2

6
Action: Insert 4

Result:
2

6
/
4
Rotation: Right-Left
Final:
4
/
2 6
Example 13: Multiple Rotations
Initial:
5
/
3 7
Action: Insert 1, then 2
Result:
5
/
3 7
/
1

2
Rotation: Left-Right, then Right
Final:
3
/
2 5
/
1 7
Example 14: Insertion Causing Multiple Rotations
Initial:
5
/
3 7
/
1 9

Action: Insert 0
Result:
5
/
3 7
/
1 9
/
0
Rotation: Right, then Right
Final:
3
/
1 5
/ \
0 2 7

9
Example 15: Insertion at Root
Initial:
Empty tree
Action: Insert 5
Result:
5
Rotation: None
Final:
5
Example 16: Balancing a Skewed Tree
Initial:
1

2

3

4
Action: Insert 5
Result:
1

2

3

4

5
Rotation: Left (multiple times)
Final:
3
/
2 4
/
1 5
Example 17: Insertion Causing Alternating Rotations
Initial:
5
/
3 7
Action: Insert 4, then 2
Result:
5
/
3 7
/
2 4
Rotation: Left-Right
Final:
4
/
3 5
/
2 7
Example 18: Complex Insertion Scenario
Initial:
8
/
4 12

/
2 6
Action: Insert 1, then 3
Result:
8
/
4 12
/
2 6
/
1 3
Rotation: Right, then Left-Right
Final:
4
/
2 8
/ \
1 3 12
/
6
Example 19: Insertion Causing Multiple Level Rotations
Initial:
10
/
5 15
/ \
3 7 20
Action: Insert 4
Result:
10
/
5 15
/ \
3 7 20
/
4
Rotation: Left-Right, then Right
Final:
10

/
5 15
/ \
4 7 20
/
3
Example 20: Insertion in a Perfectly Balanced Tree
Initial:
4
/
2 6
/ \ /
1 3 5 7
Action: Insert 8
Result:
4
/
2 6
/ \ /
1 3 5 7

8
Rotation: Left
Final:
4
/
2 6
/ \ /
1 3 5 7

8