⚠️ Problem with BST (Not Balanced)

- When we keep inserting sorted or nearly sorted data in a BST, it becomes unbalanced.
- Example: Inserting 10, 20, 30, 40, 50; the tree grows in one direction (right skewed).
- The height of tree increases → making search, insert, and delete operations slow.
- In worst case (completely unbalanced BST), Time Complexity = O(n) instead of O(log n).
- So, an unbalanced BST behaves almost like a linked list which defeats the purpose of using BST.

🌲 What is an AVL Tree?

- AVL Tree is a self-balancing Binary Search Tree (BST).
- It was invented by Adelson-Velsky and Landis in 1962.
- For every node, the height difference between its left and right subtree is at most 1.
- This height difference is called the Balance Factor (BF).
- Balance Factor = Height(Left Subtree) - Height(Right Subtree)
- Valid range → {-1, 0, +1}
- AVL Tree automatically rotates nodes to maintain balance after every insertion or deletion.

🎯 Why We Use AVL Tree

- Maintains O(log n) time complexity for search, insertion, and deletion.
- Prevents tree from becoming skewed or unbalanced.
- Provides faster search compared to normal BSTs, because the height is always minimal.
- Commonly used where read/search operations are frequent, e.g. databases and indexing systems.

⚙️ How AVL Tree Works (4 Rotation Cases)

Whenever a new node insertion makes the balance factor go out of range (-1, 0, +1),
the tree performs one of four rotations to regain balance 👇

1) Left-Left (LL) Rotation

- Occurs when a node is inserted into the left subtree of the left child.
- Solution → Perform a single right rotation.

2) Right-Right (RR) Rotation

- Occurs when a node is inserted into the right subtree of the right child.
- Solution → Perform a single left rotation.

3) Left-Right (LR) Rotation

- Occurs when a node is inserted into the right subtree of the left child.
- Solution → Perform a left rotation on left child, then right rotation on root.

4) Right-Left (RL) Rotation

- Occurs when a node is inserted into the left subtree of the right child.
- Solution → Perform a right rotation on right child, then left rotation on root.