# **Day 17 - 12 July 2025**

**Document Name:**Day 17 - hmuvvala@ - Hari Gopal Muvvala

### Task 1: AVL Tree Insertion Algorithm

#### Algorithm: AVL Tree Insertion

1. **Create a new node** with the given key and value.
2. **If the tree is empty**, set this node as the root.
3. **Otherwise**, insert the node following **Binary Search Tree (BST)** rules:
   * Go left if the key is smaller.
   * Go right if the key is larger.
4. **After insertion**, update the height of each ancestor node going up.
5. **At each node**, calculate the **balance factor**:
   * balance = height(left subtree) - height(right subtree)
6. If balance factor becomes:
   * **> +1** → left heavy → rotate
     + If inserted into left-left → **Right Rotation**
     + If inserted into left-right → **Left-Right Rotation**
   * **< -1** → right heavy → rotate
     + If inserted into right-right → **Left Rotation**
     + If inserted into right-left → **Right-Left Rotation**
7. After rotation (if needed), **resume updating heights and checking balances** up to the root.

### Task 3: Red-Black Tree – Insertion Algorithm

#### Step 1: Insert as in a normal Binary Search Tree (BST)

* Place the new node in the correct position using BST rules.
* Color the new node **RED**.

#### Step 2: Fix Red-Black Violations

Repeat the following **while parent of the new node is RED**:

##### Case 1:

Uncle is RED

* Recolor:
  + Parent → **BLACK**
  + Uncle → **BLACK**
  + Grandparent → **RED**
* Move focus up to the **grandparent** and repeat fix.

##### Case 2:

Uncle is BLACK (or null)

###### Case 2a:

Triangle case (Left-Right or Right-Left)

* Rotate to convert into line case:
  + Left-Right → Left rotate the parent
  + Right-Left → Right rotate the parent

###### Case 2b:

Line case (Left-Left or Right-Right)

* Rotate the grandparent:
  + Left-Left → Right rotate
  + Right-Right → Left rotate
* Swap colors between parent and grandparent.

#### Step 3: Ensure Root is Black

* After all rotations and recoloring, **make the root BLACK**.

#### Notes:

* Newly inserted nodes are always **red**.
* Tree is rebalanced using **rotation + recoloring**.
* This maintains all 5 Red-Black properties.
* Guarantees **O(log n)** for insertion.