BST DELETION LOGIC

M. HAADHEE SHEERAZ MIAN

Step 1 — Search for the node to delete

1. Start from the root.
2. If the key to delete is smaller than the current node's key, go to the left subtree.
3. If the key to delete is greater, go to the right subtree.
4. If the key matches:
    - **Case 1: Node has no child (leaf)** — Delete it directly.
    - **Case 2: Node has one child** — Replace the node with its only child.
    - **Case 3: Node has two children** —
a. Find the **inorder successor** (smallest node in the right subtree).
b. Copy the successor's key to the current node.
c. Recursively delete the successor.

Step 2 — Update the height

- After deletion, update the height of the current node:

sql

CopyEdit

height = 1 + max(height(left), height(right))

Step 3 — Check balance factor

- Calculate:

ini

CopyEdit

balance = height(left) – height(right)

- If the balance factor is **> 1 or < -1**, rebalance.

Step 4 — Rebalancing cases

1. **LL Case**:
   If balance > 1 and getBalance(node->left) >= 0 → Right Rotation.
2. **LR Case**:
   If balance > 1 and getBalance(node->left) < 0 →
   Left Rotate on node->left, then Right Rotate on node.
3. **RR Case**:
   If balance < -1 and getBalance(node->right) <= 0 → Left Rotation.
4. **RL Case**:
   If balance < -1 and getBalance(node->right) > 0 →
   Right Rotate on node->right, then Left Rotate on node.

Step 5 — Return the updated root

- Return the updated node pointer up the recursion chain.