75 Days of Code
Day 42
Problem no : 450.
Problem Title : Delete Node in a BST
Type : tree /BST

Given a root node reference of a BST and a key, delete the node with the given key in the BST. Return the root node reference (possibly updated) of the BST.

Basically, the deletion can be divided into two stages:

Search for a node to remove.
If the node is found, delete the node.

Example 1:

Input: root = [5,3,6,2,4,null,7], key = 3
Output: [5,4,6,2,null,null,7]
Explanation: Given key to delete is 3. So we find the node with value 3 and delete it.
One valid answer is [5,4,6,2,null,null,7], shown in the above BST.
Please notice that another valid answer is [5,2,6,null,4,null,7] and it's also accepted.

Example 2:

Input: root = [5,3,6,2,4,null,7], key = 0
Output: [5,3,6,2,4,null,7]
Explanation: The tree does not contain a node with value = 0.
Example 3:

Input: root = [], key = 0
Output: []

```
function deleteNode(root: TreeNode | null, key: number): TreeNode | null {
    if (!root) {
      return null;
    }

    // Helper function to find the minimum node in a tree
    function findMin(node: TreeNode): TreeNode {
      while (node.left) {
        node = node.left;
      }
      return node;
    }

    // Search for the node to be deleted and keep track of its parent
    let parent: TreeNode | null = null;
    let current: TreeNode | null = root;
    while (current && current.val !== key) {
      parent = current;
      if (key < current.val) {
        current = current.left;
      } else {
        current = current.right;
      }
    }

    // If the key is not found, return the root as is
    if (!current) {
      return root;
    }

    // Node to be deleted has no children
    if (!current.left && !current.right) {
      if (parent) {
        if (parent.left === current) {
          parent.left = null;
        } else {
          parent.right = null;
        }
      } else {
        root = null;
      }
    }
```

```
  // Node to be deleted has one child
  else if (!current.left || !current.right) {
    const child = current.left ? current.left : current.right;
    if (parent) {
      if (parent.left === current) {
        parent.left = child;
      } else {
        parent.right = child;
      }
    } else {
      root = child;
    }
  }

  // Node to be deleted has two children
  else {
    const successor = findMin(current.right);
    current.val = successor.val;
    current.right = deleteNode(current.right, successor.val);
  }

  return root;
}
```

Accepted

Editorial

Runtime

Details

89 ms

Beats 63.19% of users with TypeScript

Memory

51.44 MB

Beats 62.50% of users with TypeScript

Shubham Agrahari

Shubham Agrahari