9th DSA

09 June 2025 09:40

9th June 2025

BST with Dead End

You are given a Binary Search Tree (BST) containing unique positive integers greater than 0. Your task is to determine whether the BST contains a dead end. Note: A dead end is a leaf node in the BST such that no new node can be inserted in the BST at or below this node while maintaining the BST property and the constraint that all node values must be > 0.

Examples: Input: root[] = [8, 5, 9, 2, 7, N, N, 1] Output: true Explanation: Node 1 is a Dead End in the given BST. Input: root[] = [8, 7, 10, 2, N, 9, 13] Output: true Explanation: Node 9 is a Dead End in the given BST.

Constraints: 1 <= number of nodes <= 3000 1 <= node->data <= 105

Expected Complexities Time Complexity: O(n) Auxiliary Space: O(h)

Solution:-

```
class Solution {

  // Definition for Tree Node
  static class Node {
    int data;
    Node left, right;
    Node(int item) {
      data = item;
      left = right = null;
    }
  }

  // Function to check if the BST contains a dead end
  public boolean isDeadEnd(Node root) {
    return checkDeadEnd(root, 1, Integer.MAX_VALUE);
  }

  // Helper function to check dead ends in the given range
  private boolean checkDeadEnd(Node node, int min, int max) {
```

```
        if (node == null) return false;

        // If this is a leaf node and no room to insert new nodes
        if (min == max) return true;

        // Check in left and right subtrees
        return checkDeadEnd(node.left, min, node.data - 1) ||
               checkDeadEnd(node.right, node.data + 1, max);
    }
}
```