Solution 29-11-23

701. Insert into a Binary Search Tree

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online

class TreeNode {
  int val;
  TreeNode left;
  TreeNode right;

  TreeNode(int x) {
    val = x;
  }
}

public class Solution701 {
  public static TreeNode insertIntoBST(TreeNode root, int val) {
    if (root == null) {
      return new TreeNode(val);
    }

    if (val < root.val) {
      root.left = insertIntoBST(root.left, val);
    } else {
      root.right = insertIntoBST(root.right, val);
    }

    return root;
  }
```

```
  public static void inOrderTraversal(TreeNode root) {
    if (root != null) {
      inOrderTraversal(root.left);
      System.out.print(root.val + " ");
      inOrderTraversal(root.right);
    }
  }

  public static void main(String[] args) {
    // Example input
    // You can customize the input values according to your requirements
    int[] values = { 4, 2, 7, 1, 3 };
    int insertValue = 5;

    // Construct the initial BST
    TreeNode root = null;
    for (int value : values) {
      root = insertIntoBST(root, value);
    }

    // Insert a new value into the BST
    root = insertIntoBST(root, insertValue);

    // Print the in-order traversal of the modified BST
    System.out.print("In-order traversal after insertion: ");
    inOrderTraversal(root);
  }
}
```

Output:-

```
PS C:\Users\Ajeet\Desktop\java> java Solution701
In-order traversal after insertion: 1 2 3 4 5 7
```

652. Find Duplicate Subtrees

```
import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;

class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int x) {
        val = x;
    }
}

public class FindDuplicateSubtrees {

    public static List<TreeNode> findDuplicateSubtrees(TreeNode root) {
        List<TreeNode> result = new LinkedList<>();
        Map<String, Integer> map = new HashMap<>();

        traverse(root, map, result);

        return result;
    }

    private static String traverse(TreeNode node, Map<String, Integer> map, List<TreeNode> result) {
        if (node == null) {
            return "#";
        }
```

```
        String key = node.val + "," + traverse(node.left, map, result) + "," + traverse(node.right, map, result);

        map.put(key, map.getOrDefault(key, 0) + 1);

        if (map.get(key) == 2) {
            result.add(node);
        }

        return key;
    }

    public static void main(String[] args) {
        TreeNode root = new TreeNode(1);
        root.left = new TreeNode(2);
        root.right = new TreeNode(3);
        root.left.left = new TreeNode(4);
        root.right.left = new TreeNode(2);
        root.right.right = new TreeNode(4);
        root.right.left.left = new TreeNode(4);

        // Example Output
        List<TreeNode> duplicates = findDuplicateSubtrees(root);

        System.out.println("Duplicate Subtrees:");
        for (TreeNode duplicate : duplicates) {
            // You can customize the way you want to print the duplicate subtrees
            printTree(duplicate);
            System.out.println();
        }
```

```
    }

    private static void printTree(TreeNode node) {
        if (node == null) {
            return;
        }
        System.out.print(node.val + " ");
        printTree(node.left);
        printTree(node.right);
    }
}
```

Output:-

```
PS C:\Users\Ajeet\Desktop\java> javac FindDuplicateSubtrees.java
PS C:\Users\Ajeet\Desktop\java> java FindDuplicateSubtrees
Duplicate Subtrees:
4
2 4
```