1.https://leetcode.com/problems/invert-binary-tree/

```
public class InvertBinaryTree {

    public static TreeNode invertTree(TreeNode root) {
        if (root == null) {
            return null;
        }

        TreeNode temp = root.left;
        root.left = root.right;
        root.right = temp;

        invertTree(root.left);
        invertTree(root.right);

        return root;
    }

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

```
        root.right = new TreeNode(7);
        root.left.left = new TreeNode(1);
        root.left.right = new TreeNode(3);
        root.right.left = new TreeNode(6);
        root.right.right = new TreeNode(9);

        TreeNode invertedTree = invertTree(root);

        System.out.println("The inverted tree is:");
        System.out.println(invertedTree);
    }
}
```

2.https://leetcode.com/problems/kth-smallest-element-in-a-bst/

```
public class KthSmallestElementInBST {
```

```
public static int kthSmallest(TreeNode root, int k) {
    if (root == null) {
        return -1;
    }

    Stack<TreeNode> stack = new Stack<>();
    TreeNode cur = root;
    int count = 0;

    while (cur != null || !stack.isEmpty()) {
        while (cur != null) {
            stack.push(cur);
            cur = cur.left;
        }

        cur = stack.pop();
        count++;

        if (count == k) {
            return cur.val;
        }

        cur = cur.right;
    }
```

```
        return -1;
    }

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

        int k = 1;
        int kthSmallest = kthSmallest(root, k);

        System.out.println("The kth smallest element
is: " + kthSmallest);
    }
}
```

3.https://leetcode.com/problems/path-sum/

```
public class Solution {

    public boolean hasPathSum(TreeNode root, int
targetSum) {
        if (root == null) {
            return false;
        }

        if (root.left == null && root.right == null &&
targetSum == root.val) {
            return true;
        }

        return hasPathSum(root.left, targetSum -
root.val) || hasPathSum(root.right, targetSum -
root.val);
    }
}
```

4.https://leetcode.com/problems/binary-tree-inorder-traversal/

```
class Solution {

    public List<Integer> inorderTraversal(TreeNode root) {
        List<Integer> inorder = new ArrayList<>();
        helper(root, inorder);
        return inorder;
    }

    private void helper(TreeNode root, List<Integer> inorder) {
        if (root == null) {
            return;
        }

        helper(root.left, inorder);
        inorder.add(root.val);
        helper(root.right, inorder);
    }
}
```