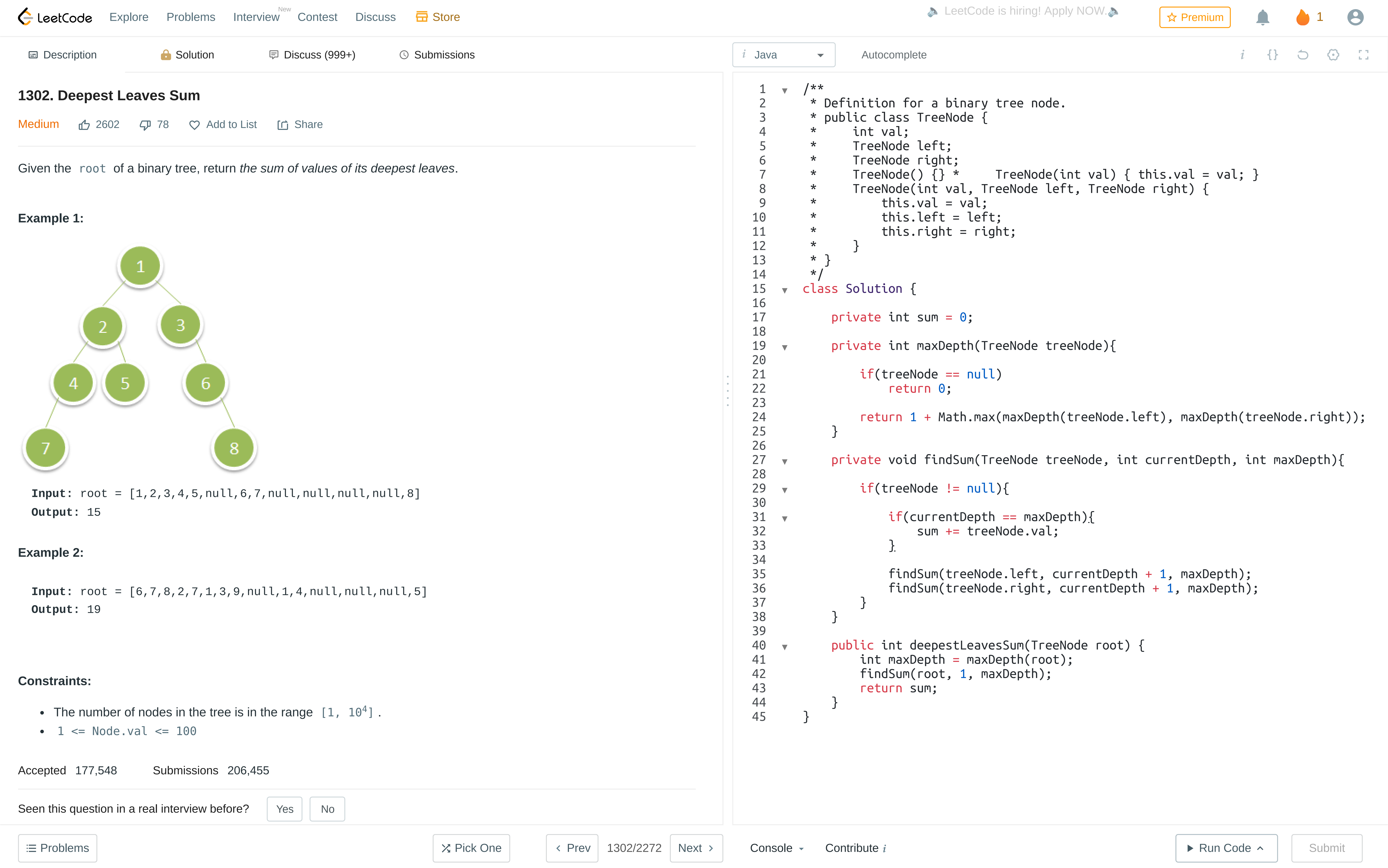

1302. Deepest Leaves Sum

Given the `root` of a binary tree, return *the sum of values of its deepest leaves*.

Example 1:

```
Input: root = [1,2,3,4,5,null,6,7,null,null,null,null,8]
Output: 15
```

Example 2:

```
Input: root = [6,7,8,2,7,1,3,9,null,1,4,null,null,null,5]
Output: 19
```

Constraints:

- The number of nodes in the tree is in the range $[1, 10^4]$.
- `1 <= Node.val <= 100`

Accepted 177,548 Submissions 206,455

Seen this question in a real interview before? Yes No

```java
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {} *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
class Solution {

    private int sum = 0;

    private int maxDepth(TreeNode treeNode){

        if(treeNode == null)
            return 0;

        return 1 + Math.max(maxDepth(treeNode.left), maxDepth(treeNode.right));
    }

    private void findSum(TreeNode treeNode, int currentDepth, int maxDepth){

        if(treeNode != null){

            if(currentDepth == maxDepth){
                sum += treeNode.val;
            }

            findSum(treeNode.left, currentDepth + 1, maxDepth);
            findSum(treeNode.right, currentDepth + 1, maxDepth);
        }
    }

    public int deepestLeavesSum(TreeNode root) {
        int maxDepth = maxDepth(root);
        findSum(root, 1, maxDepth);
        return sum;
    }
}
```