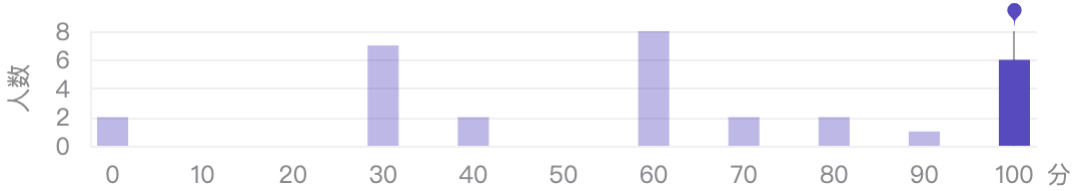
yifan-1 yifan-1

总分
100.00/100

排名
4/30

成绩分布

分数段 100 - 105 共 6 名考生



编程语言 Python3

竞赛名称:
LeetCamp 第二期第三周竞
开始时间:
2021-12-18 04:00
登录 IP 地址:
183.159.124.87(中国杭州市电信)
邮箱:
372104922@qq.com

罚时次数:
1
结束时间:
2021-12-18 14:16
登录设备:
Windows 10 | Chrome 96.0.4664

答题详情

#1 编程题 1000595

分数 30.00/30

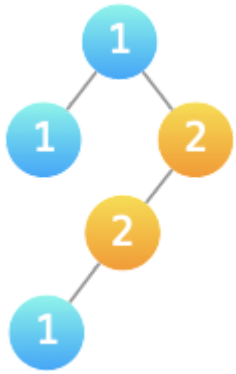
统计二叉查找树中指定值的结点个数

给定一棵二叉查找树的根结点 `root`，求其中值等于 `val` 的结点个数。

注意：本题中二叉查找树允许存在值相同的结点。

示例 1：

> 输入：`root = [1,1,2,null,null,2,null,1], val = 2`
>
> 输出：`2`
>
> 解释：
> 如图所示，橙色结点为值等于 `2` 的结点，其总数为 `2` 。



提示：

- `1 <= 结点数 <= 10^4`
- `1 <= 结点值 <= 10^4`
- `1 <= val <= 10^4`

用时 00:02:24　　提交次数 1　2021/12/18 13:00

提交结果：	通过测试用例：	语言：
通过	71/71	python3
执行用时：	消耗内存：	
448 ms	19.4 MB	

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
```

```
#         self.left = None
#         self.right = None
class Solution:
    def countNodes(self, root: TreeNode, val: int) -> int:
        self.cnt = 0
        def dfs(root):
            if not root:
                return
            if root.val == val:
                self.cnt += 1
            dfs(root.left)
            dfs(root.right)

        dfs(root)
        return self.cnt
```

#2 编程题 1000596 分数 30.00/30

统计不一致结点个数

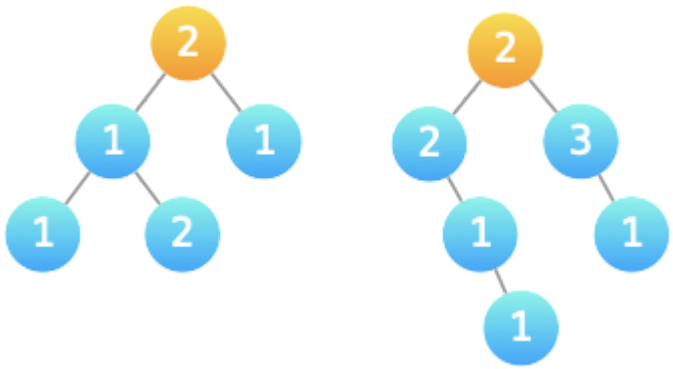
两棵树中的 「**不一致结点**」 的定义如下：

- 某一位置的结点只在其中一棵树中存在，另一棵树中对应位置为空；
- 或者两棵树中某一相同位置的结点都不为空，但是结点值不相同。

给定两棵二叉树的根结点 `root1` 和 `root2`，求这两棵树中的 「**不一致结点**」 个数。

示例 1：

> 输入：`root1 = [2,1,1,1,2], root2 = [2,2,3,null,1,null,1,null,1]`
>
> 输出：`9`
>
> 解释：
> 如图所示，除橙色结点外，其余结点都为 「**不一致结点**」 。

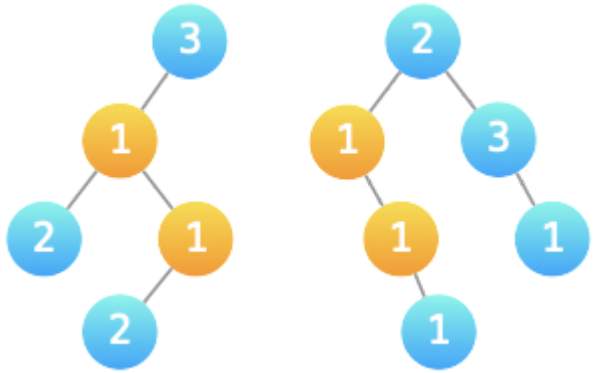


示例 2：

输入：`root1 = [3,1,null,2,1,null,null,1], root2 = [2,1,3,null,1,null,1,null,1]`

输出：`7`

解释：
如图所示，除橙色结点外，其余结点都为 「**不一致结点**」 。



提示：

- `1 <= 每棵树的结点数 <= 10^4`
- `1 <= 结点值 <= 10^4`

用时 00:15:53　　提交次数 1　2021/12/18 13:16

提交结果：	通过测试用例：	语言：

通过	81/81	python3
执行用时： 1172 ms	消耗内存： 22 MB	

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None
class Solution:
    def countInconsistentNode(self, root1: TreeNode, root2: TreeNode) -> int:
        self.cnt = 0
        def dfs(root1, root2):
            if not root1 and not root2:
                return
            elif root1 and not root2:
                self.cnt += 1
                dfs(root1.left, root2)
                dfs(root1.right, root2)
            elif not root1 and root2:
                self.cnt += 1
                dfs(root1, root2.left)
                dfs(root1, root2.right)
            else:
                if root1.val != root2.val:
                    self.cnt += 2
                dfs(root1.left, root2.left)
                dfs(root1.right, root2.right)
        dfs(root1, root2)
        return self.cnt
```

#3 编程题 1000597 分数 20.00/20

平衡二叉查找树

给定一棵二叉查找树的根结点 `root` 和一个二维数组 `operations`，其中 `operations[i] = [type,val]`：

- 若 `type = 0`，将 `val` 插入给定的二叉查找树中；
- 若 `type = 1`，删除树中的最大值，若有多个最大值，只需删除其中的任意一个即可。

你需要完成 `operations` 中的每个操作，并返回一个 `ans` 数组，其中 `ans[i]` 表示完成当前操作后二叉查找树中的最大值。

注意：

- 本题中二叉查找树允许存在值相同的结点；
- 测试数组保证每次操作完成后，二叉查找树的结点数恒大于 0。

示例 1：

> 输入：`root = [1,1,2,null,1,null,2], operations = [[0,3],[0,2],[1,0]]`
>
> 输出：`[3,3,2]`
>
> 解释：
>
> - 将 `3` 插入树中，此时二叉查找树中的最大值为 `3`；
> - 将 `2` 插入树中，此时二叉查找树中的最大值为 `2`；
> - 删除树中的最大值 `3`，此时二叉查找树中的最大值为 `2`。

提示：

- `1 <= 结点数 <= 10^4`
- `1 <= 结点值 <= 10^4`
- `1 <= operations.length <= 10^4`
- `0 <= operations[i][0] <=1`
- `0 <= operations[i][1] <=10^4`
- 只有当 `operations[i][0] = 1` 时，`operations[i][1] = 0`，此时 `operations[i][1]` 无意义。

用时 00:20:19 提交次数 2

提交结果：	通过测试用例：	语言：
通过	81/81	python3
执行用时：	消耗内存：	罚时次数
996 ms	32.3 MB	1

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None
class Solution:
    def solve(self, root: TreeNode, operations: List[List[int]]) -> List[int]:
        self.nums = []
        def inorder(root):
            if not root:
                return
            inorder(root.left)
            self.nums.append(-root.val)
            inorder(root.right)

        inorder(root)
        heapq.heapify(self.nums)
        res = []
        for op in operations:
            if op[0] == 0:
                heapq.heappush(self.nums, -op[1])
                res.append(-self.nums[0])
            elif op[0] == 1:
                heapq.heappop(self.nums)
                res.append(-self.nums[0])
        return res
```

#4 编程题 1000598 

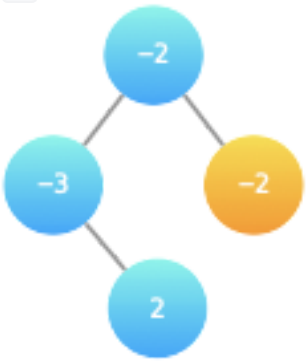
分数 10.00/10

切割二叉树

给定一个二叉树的根结点 `root` ，切断 **除根结点以外** 任意一个结点与其父结点的连边可以使整棵树分成两部分，求这两部分的结点值之和的差值的 **绝对值** 的最小值。

示例 1：

> 输入：`root = [-2,-3,-2,null,2]`
>
> 输出：`1`
>
> 解释：
> 如图所示，将橙色结点与其父结点的连边切断后可将整棵树为两部分，这两部分的结点值之和的差值的绝对值为 `|(-2 + -3 + 2) - (-2)| = 1` 。



提示：

- `2 <= 结点数 <= 10^4`
- `-10^4 <= 结点值 <= 10^4`

用时 00:19:15

提交次数 1 2021/12/18 13:56

提交结果：	通过测试用例：	语言：

通过	82/82	python3
执行用时: 776 ms	消耗内存: 19.8 MB	

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None
class Solution:
    def splitTree(self, root: TreeNode) -> int:
        self.nums = []
        def dfs(root):
            if not root:
                return 0
            L = dfs(root.left)
            R = dfs(root.right)
            self.nums.append(root.val + L + R)
            return L + R +root.val
        dfs(root)
        allsum = self.nums[-1]
        res = sys.maxsize
        for i in range(0, len(self.nums) - 1):
            res = min(abs(allsum - 2 * self.nums[i]), res)
        return res
```

#5 编程题 1000599 分数 10.00/10

树上结点的最短距离

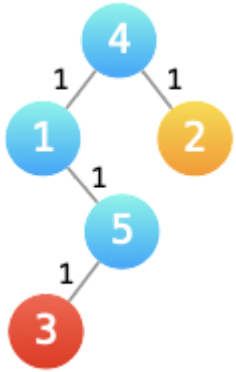
给定一棵含有 `n` 个结点的二叉树的根结点 `root`，求结点值为 `num1` 的结点到结点值为 `num2` 的结点的最短距离。

注意：

- 给定的二叉树中的所有结点值均不相等；
- 给定的二叉树中一定含有结点值为 `num1` 和 `num2` 的结点；
- 二叉树中两个相邻结点的距离为 1。

示例 1：

> 输入：`root = [4,1,2,null,5,null,null,3], num1 = 3, num2 = 2`
>
> 输出：`4`
>
> 解释：
> 如图所示，结点值为 `3` 的结点到结点值为 `2` 的结点的最短距离为 `4` 。



提示：

- `2 <= 结点数 <= 10^4`
- `1 <= 结点值 <= 10^4`
- `num1 != num2`

用时 00:19:50　　提交次数 1　2021/12/18 14:15

提交结果：	通过测试用例：	语言：
通过	81/81	python3
执行用时：	消耗内存：	

836 ms	20.3 MB

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None
class Solution:
    def getDistance(self, root: TreeNode, num1: int, num2: int) -> int:
        node_parent = dict()
        self.target = TreeNode(num1)

        def dfs_find_parent(node: TreeNode) -> None:
            if node:
                if node.left:
                    node_parent[node.left] = node
                if node.right:
                    node_parent[node.right] = node
                if node.val == num1:
                    self.target = node
                dfs_find_parent(node.left)
                dfs_find_parent(node.right)

        dfs_find_parent(root)
        res = sys.maxsize
        Q = collections.deque()
        visited = set()
        Q.append(self.target)
        visited.add(self.target)
        level = 0
        while Q:
            level += 1
            for _ in range(len(Q)):
                x = Q.popleft()
                for y in[node_parent[x] if x in node_parent else None, x.left, x.right]:
                    if y and y not in visited:
```

```
                    if y.val == num2:
                        res = min(res, level)
                    Q.append(y)
                    visited.add(y)
        return res
```