Assignment #8: 树为主

Updated 1704 GMT+8 Apr 8, 2025

2025 spring, Complied by 任宇桐 物理学院

> **说明：**
>
> 1. **解题与记录：**
>
> 对于每一个题目，请提供其解题思路（可选），并附上使用Python或C++编写的源代码（确保已在OpenJudge， Codeforces，LeetCode等平台上获得Accepted）。请将这些信息连同显示“Accepted”的截图一起填写到下方的作业模板中。（推荐使用Typora https://typoraio.cn 进行编辑，当然你也可以选择Word。）无论题目是否已通过，请标明每个题目大致花费的时间。
>
> 2. **提交安排：**提交时，请首先上传PDF格式的文件，并将.md或.doc格式的文件作为附件上传至右侧的“作业评论”区。确保你的Canvas账户有一个清晰可见的头像，提交的文件为PDF格式，并且“作业评论”区包含上传的.md或.doc附件。
> 3. **延迟提交：**如果你预计无法在截止日期前提交作业，请提前告知具体原因。这有助于我们了解情况并可能为你提供适当的延期或其他帮助。
>
> 请按照上述指导认真准备和提交作业，以保证顺利完成课程要求。

1. 题目

LC108.将有序数组转换为二叉树

dfs, https://leetcode.cn/problems/convert-sorted-array-to-binary-search-tree/

思路：

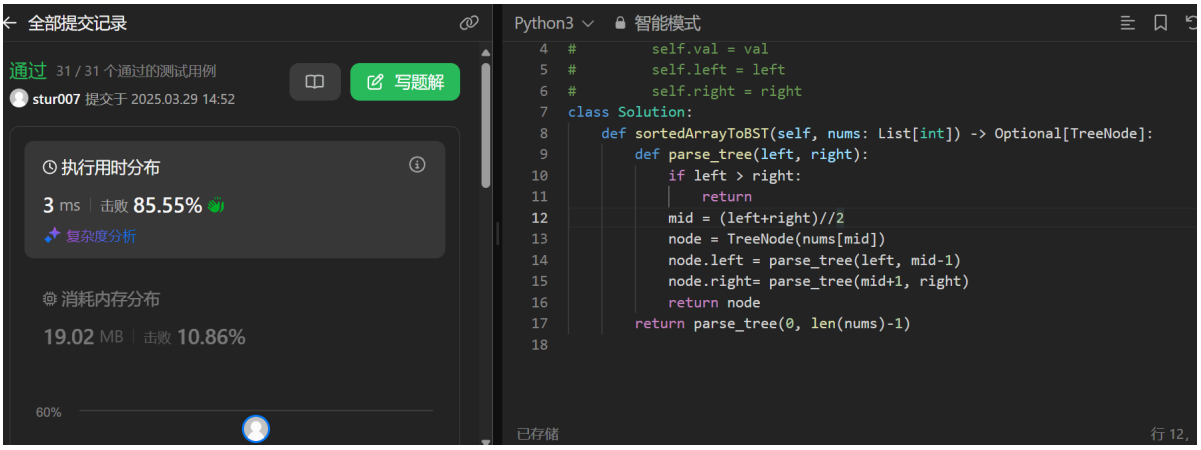
每次找到中间的节点，对两边分别递归即可。

代码：

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def sortedArrayToBST(self, nums: List[int]) -> Optional[TreeNode]:
        def parse_tree(left, right):
            if left > right:
                return
            mid = (left+right)//2
            node = TreeNode(nums[mid])
            node.left = parse_tree(left, mid-1)
            node.right= parse_tree(mid+1, right)
            return node
```

```
        return parse_tree(0, len(nums)-1)
```

代码运行截图 （至少包含有"Accepted"）



M27928:遍历树

adjacency list, dfs, http://cs101.openjudge.cn/practice/27928/

思路:

注意特判list中直接是根节点的情形。

代码:

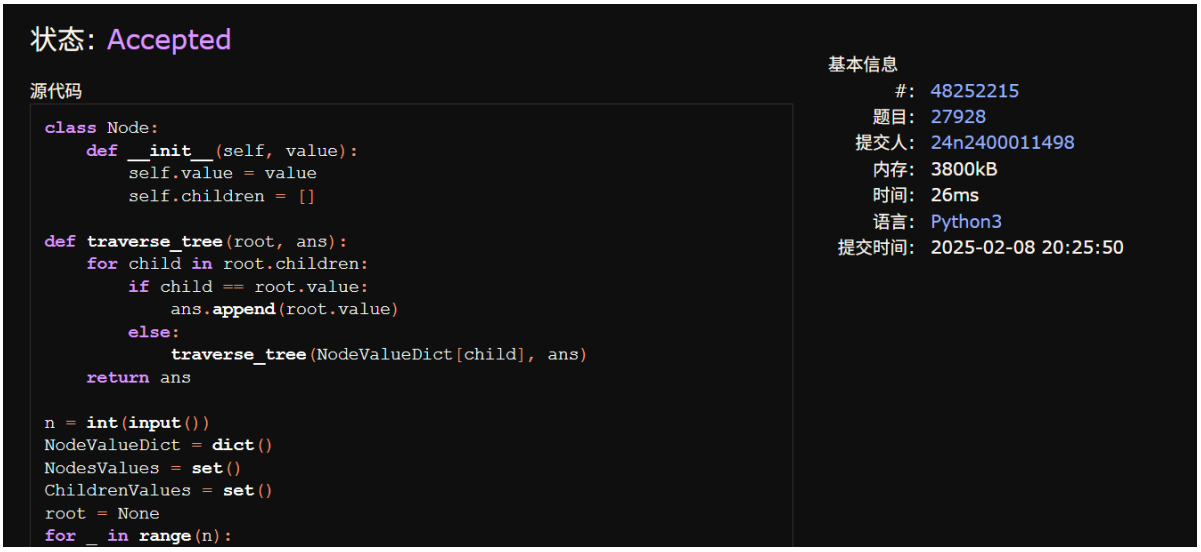
```
class Node:
    def __init__(self, value):
        self.value = value
        self.children = []

def traverse_tree(root, ans):
    for child in root.children:
        if child == root.value:
            ans.append(root.value)
        else:
            traverse_tree(NodeValueDict[child], ans)
    return ans

n = int(input())
NodeValueDict = dict()
NodesValues = set()
ChildrenValues = set()
root = None
for _ in range(n):
    s = list(map(int, input().split()))
    node = Node(s[0])
    NodeValueDict[s[0]] = node
    node.children = sorted(s)
    NodesValues |= set(s)
    if len(s) > 1:
        ChildrenValues |= set(s[1:])
```

```
RootValue ,= NodesValues - ChildrenValues
root = NodeValueDict[RootValue]
ans = traverse_tree(root, [])
for i in ans:
    print(i)
```

代码运行截图（至少包含有"Accepted"）



LC129.求根节点到叶节点数字之和

dfs, https://leetcode.cn/problems/sum-root-to-leaf-numbers/

思路：

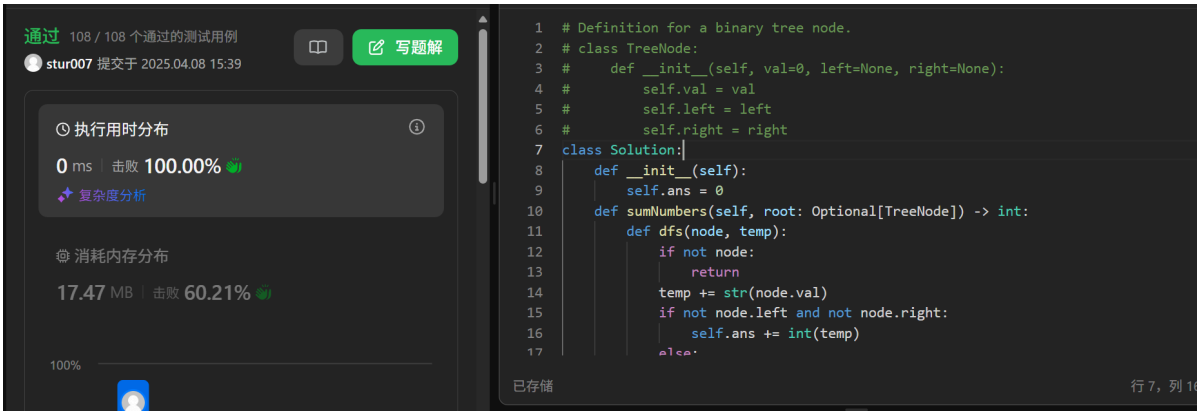
直接搜索即可。

代码：

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def __init__(self):
        self.ans = 0
    def sumNumbers(self, root: Optional[TreeNode]) -> int:
        def dfs(node, temp):
            if not node:
                return
            temp += str(node.val)
            if not node.left and not node.right:
                self.ans += int(temp)
            else:
                dfs(node.left, temp)
                dfs(node.right, temp)
```

```
            temp = temp[:-1]
        dfs(root, '')
        return self.ans
```

代码运行截图（至少包含有"Accepted"）



M22158:根据二叉树前中序序列建树

tree, http://cs101.openjudge.cn/practice/22158/

思路:

根据中序找到前序的分界点。

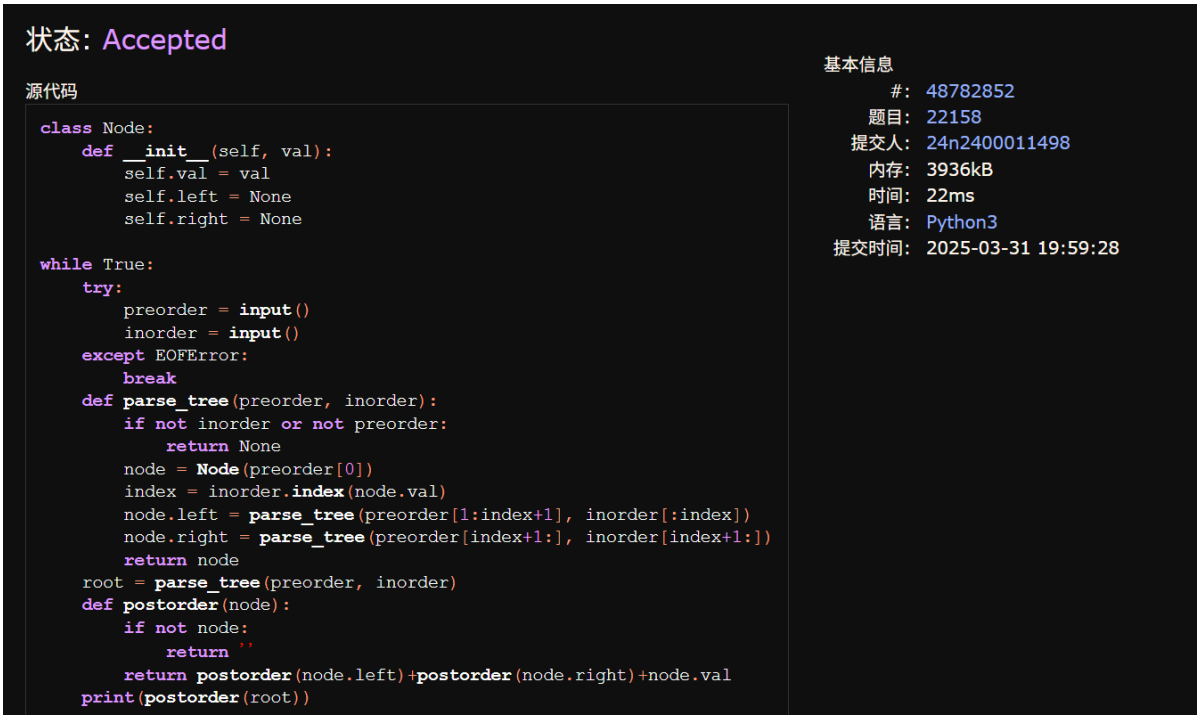
代码:

```
class Node:
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None

while True:
    try:
        preorder = input()
        inorder = input()
    except EOFError:
        break
    def parse_tree(preorder, inorder):
        if not inorder or not preorder:
            return None
        node = Node(preorder[0])
        index = inorder.index(node.val)
        node.left = parse_tree(preorder[1:index+1], inorder[:index])
        node.right = parse_tree(preorder[index+1:], inorder[index+1:])
        return node
    root = parse_tree(preorder, inorder)
    def postorder(node):
        if not node:
            return ''
```

```
        return postorder(node.left)+postorder(node.right)+node.val
    print(postorder(root))
```

代码运行截图（至少包含有"Accepted"）



T24729:括号嵌套树

dfs, stack, http://cs101.openjudge.cn/practice/24729/

思路:

尝试了一下用二叉树模拟多叉树。

代码:

```
class Node:
    def __init__(self, val):
        self.val = val
        self.FirstChild = None
        self.NextSibling = None
def parse_tree(s):
    stack = []
    for i in range(len(s)):
        if s[i].isalpha():
            node = Node(s[i])
            if stack:
                if stack[-1] == '(':
                    stack[-2].FirstChild = node
                else:
                    stack[-1].NextSibling = node
            stack.append(node)
        elif s[i] == '(':
            stack.append('(')
```

```
        elif s[i] == ')':
            while stack[-1] != '(':
                stack.pop()
            stack.pop()
    return stack.pop()
def preorder(node):
    if node:
        return node.val+preorder(node.FirstChild)+preorder(node.NextSibling)
    return ''
def postorder(node):
    if node:
        return postorder(node.FirstChild)+node.val+postorder(node.NextSibling)
    return ''
s = input()
root = parse_tree(s)
print(preorder(root))
print(postorder(root))
```

代码运行截图（至少包含有"Accepted"）

LC3510.移除最小数对使数组有序II

doubly-linked list + heap, https://leetcode.cn/problems/minimum-pair-removal-to-sort-array-ii/

思路:

按照tag想法比较直接，就是代码写起来比较麻烦，感觉必须非常清醒才能写对.....

代码:

```
class Node:
    def __init__(self,val, idx):
        self.val = val
        self.idx = idx
        self.prev = None
        self.next = None
class Solution:
```

```
    def minimumPairRemoval(self, nums: List[int]) -> int:
        least_sums = []
        wrong_positions = set()
        idx_for_node = dict()
        if nums:
            node = Node(nums[0], 0)
            idx_for_node[0] = node
            for i in range(1, len(nums)):
                temp = Node(nums[i], i)
                idx_for_node[i] = temp
                node.next = temp
                temp.prev = node
                node = node.next
        for i in range(len(nums)-1):
            heapq.heappush(least_sums, (nums[i]+nums[i+1], i, i+1))
            if nums[i+1]<nums[i]:
                wrong_positions.add((i, i+1))
        cnt = 0
        while wrong_positions:
            val, idx_1, idx_2 = heapq.heappop(least_sums)
            if not idx_for_node[idx_1] or not idx_for_node[idx_2] or val !=
idx_for_node[idx_1].val+idx_for_node[idx_2].val:
                continue
            cnt += 1
            node_1 = idx_for_node[idx_1]
            node_2 = idx_for_node[idx_2]
            prev_node = node_1.prev
            next_node = node_2.next
            if prev_node and (prev_node.idx, idx_1) in wrong_positions:
                wrong_positions.remove((prev_node.idx, idx_1))
            if (idx_1, idx_2) in wrong_positions:
                wrong_positions.remove((idx_1, idx_2))
            if next_node and (idx_2, next_node.idx) in wrong_positions:
                wrong_positions.remove((idx_2, next_node.idx))
            new_node = Node(val, idx_1)
            idx_for_node[idx_1]=new_node
            idx_for_node[idx_2]=None
            if prev_node:
                prev_node.next = new_node
                new_node.prev = prev_node
                if val < prev_node.val:
                    wrong_positions.add((prev_node.idx, idx_1))
                heapq.heappush(least_sums, (val+prev_node.val, prev_node.idx,
new_node.idx))
            if next_node:
                new_node.next = next_node
                next_node.prev = new_node
                if val > next_node.val:
                    wrong_positions.add((idx_1, next_node.idx))
                heapq.heappush(least_sums, (val+next_node.val, new_node.idx,
next_node.idx))
        return cnt
```

代码运行截图 （至少包含有"Accepted"）

2. 学习总结和收获

如果发现作业题目相对简单，有否寻找额外的练习题目，如“数算2025spring每日选做”、LeetCode、Codeforces、洛谷等网站上的题目。

这周期中考试有点多，仅仅完成了作业，下周要开始补这周落下的每日选做了。