Assignment #8: 树为主

Updated 1704 GMT+8 Apr 8, 2025

2025 spring, Complied by <mark>同学的姓名、院系</mark>

姓名：李彦臻
学号：2300010821
学院：数学科学学院

> **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/

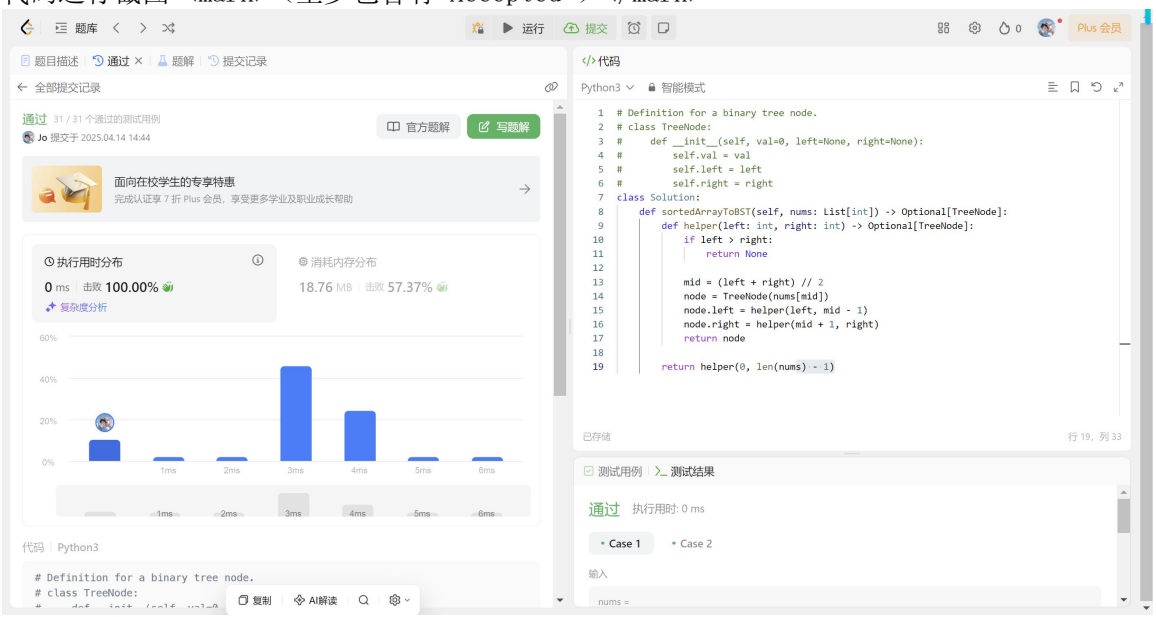
思路：使用分治递归的方法，每次取数组的中间元素作为当前子树的根节点，这样能保证左右子树的节点数量尽量接近，从而实现平衡；每次递归将数组划分为左右两部分，分别构建左右子树，直到构建完整棵树为止。
用时：15mins

代码：

```python

```

```
class Solution:
    def sortedArrayToBST(self, nums: List[int]) -> Optional[TreeNode]:
        def helper(left: int, right: int) -> Optional[TreeNode]:
            if left > right:
                return None

            mid = (left + right) // 2
            node = TreeNode(nums[mid])
            node.left = helper(left, mid - 1)
            node.right = helper(mid + 1, right)
            return node

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

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



M27928:遍历树

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

思路：先输入树的结构，并将每个节点及其子节点存储在字典中；接着，查找根节点，并使用递归遍历树，在遍历过程中，将当前节点的子节点按值分为小于和大于当前节点的两部分，并分别递归处理即可。

用时：25mins

代码：

```python

```
```
n = int(input())  # 输入节点数
tree = {}  # 用于存储每个节点及其子节点
values = []  # 用于存储所有节点的值
root = None  # 根节点初始化为空
children_map = {}  # 用于记录每个节点的父节点

# 输入每个节点及其子节点
for _ in range(n):
    parts = list(map(int, input().split()))
    value = parts[0]  # 当前节点的值
    children = parts[1:] if len(parts) > 1 else []  # 当前节点的子节点
    tree[value] = children  # 将当前节点和子节点存入 tree 字典
    values.append(value)  # 将节点值添加到 values 列表
    for child in children:
        children_map[child] = value  # 记录子节点的父节点

# 找到根节点（即没有父节点的节点）
root_candidates = set(tree.keys()) - set(children_map.keys())
root = root_candidates.pop()  # 从候选的根节点中选出一个根

# 遍历树并按规则输出
def traverse(node):
    children = tree.get(node, [])  # 获取当前节点的子节点
    # 将子节点分成比当前节点小的和比当前节点大的两部分
    less = [child for child in children if child < node]
    greater = [child for child in children if child > node]
    # 对小于当前节点的子节点排序
    less.sort()
    # 对大于当前节点的子节点排序
    greater.sort()
    # 先处理小于当前节点的子节点
    for child in less:
        traverse(child)
    # 输出当前节点
    print(node)
    # 然后处理大于当前节点的子节点
    for child in greater:
        traverse(child)
```

traverse(root) # 从根节点开始遍历

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

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：用 dfs 来遍历树，从根节点到叶子节点的路径代表一个数字；在递归过程中累积路径值，然后在到达叶子节点时返回完整的数字。最后，累积所有叶子节点的数字即可得到答案。
用时：20mins

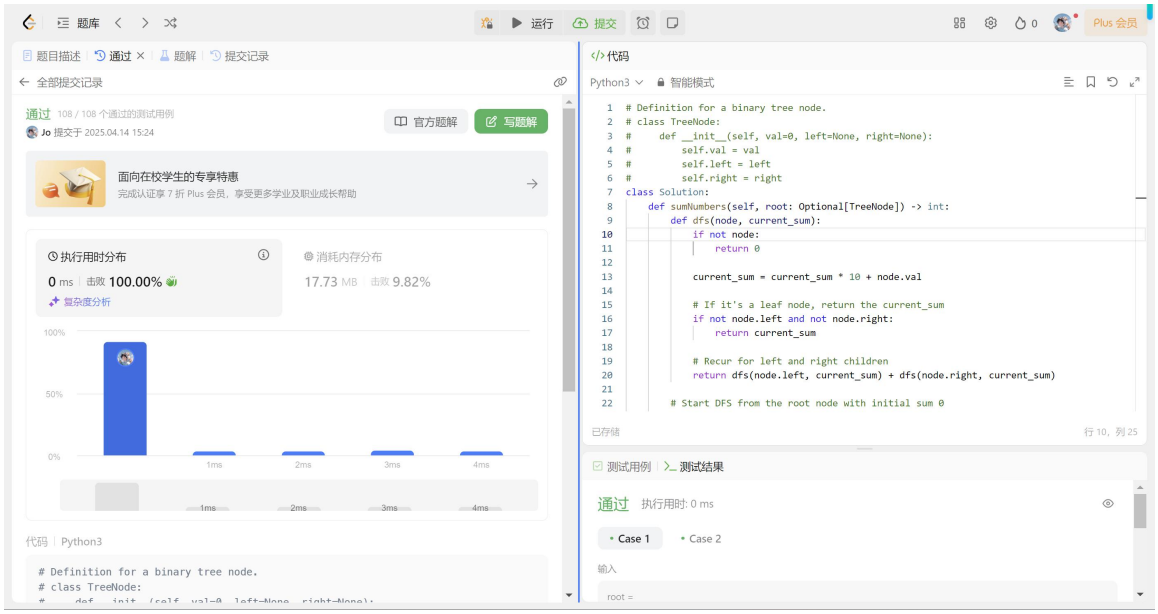
代码：

```python

```

```
class Solution:
    def sumNumbers(self, root: Optional[TreeNode]) -> int:
        def dfs(node, current_sum):
            if not node:
                return 0
```

```
        current_sum = current_sum * 10 + node.val

        # If it's a leaf node, return the current_sum
        if not node.left and not node.right:
            return current_sum

        # Recur for left and right children
        return dfs(node.left, current_sum) + dfs(node.right, current_sum)

    # Start DFS from the root node with initial sum 0
    return dfs(root, 0)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



M22158:根据二叉树前中序序列建树

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

思路：先重建二叉树，在中序遍历序列中找到根节点的位置，根节点左侧的子序列构成左子树的中序遍历，右侧的子序列构成右子树的中序遍历；根据左子树的节点数量，可以在前序遍历序列中分割出左子树的前序遍历和右子树的前序遍历。递归地对左子树和右子树重复上述过程，直到序列为空，此时子树构建完成。接着，递归地访问左子树和右子树，最后访问根节点，将节点值按顺序收集起来，即为后序遍历序列。
用时：20mins

代码：

```python

```
```
def build_tree(preorder, inorder):
    if not preorder or not inorder:
        return None
    root_val = preorder[0]
    root = root_val
    root_pos_in_inorder = inorder.index(root_val)

    left_inorder = inorder[:root_pos_in_inorder]
    right_inorder = inorder[root_pos_in_inorder+1:]

    left_preorder = preorder[1:1+len(left_inorder)]
    right_preorder = preorder[1+len(left_inorder):]

    left_subtree = build_tree(left_preorder, left_inorder)
    right_subtree = build_tree(right_preorder, right_inorder)

    return (root, left_subtree, right_subtree)

def postorder_traversal(tree):
    if not tree:
        return []
    root, left, right = tree
    return postorder_traversal(left) + postorder_traversal(right) + [root]

def solve():
    import sys
    input_lines = sys.stdin.read().splitlines()
    idx = 0
    while idx < len(input_lines):
        preorder = input_lines[idx].strip()
        idx += 1
        inorder = input_lines[idx].strip()
        idx += 1
        tree = build_tree(preorder, inorder)
        postorder = postorder_traversal(tree)
        print(''.join(postorder))

solve()
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

M24729:括号嵌套树

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

思路：

代码：

```python
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

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：

代码：

```python

```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

2. 学习总结和收获

<mark>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。</mark>

最近两周一直在忙活各科的期中考试，用在数算上的时间相对较少。新接触到一些概念理解的还不是很透彻，加之时间紧张，因此这次作业有好几道题都是在 ai 或者题解的提示下才完成的；这一段时间疏于练习数算题目，也导致我在每日选做里落下了不少进度。等忙完这一段时间的期中考之后，得赶紧把这些内容全部都补起来啊 qwq~