Assignment #8: 树为主

Updated 1704 GMT+8 Apr 8, 2025

2025 spring, Complied by <mark>汤伟杰，信息管理系</mark>

> **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：

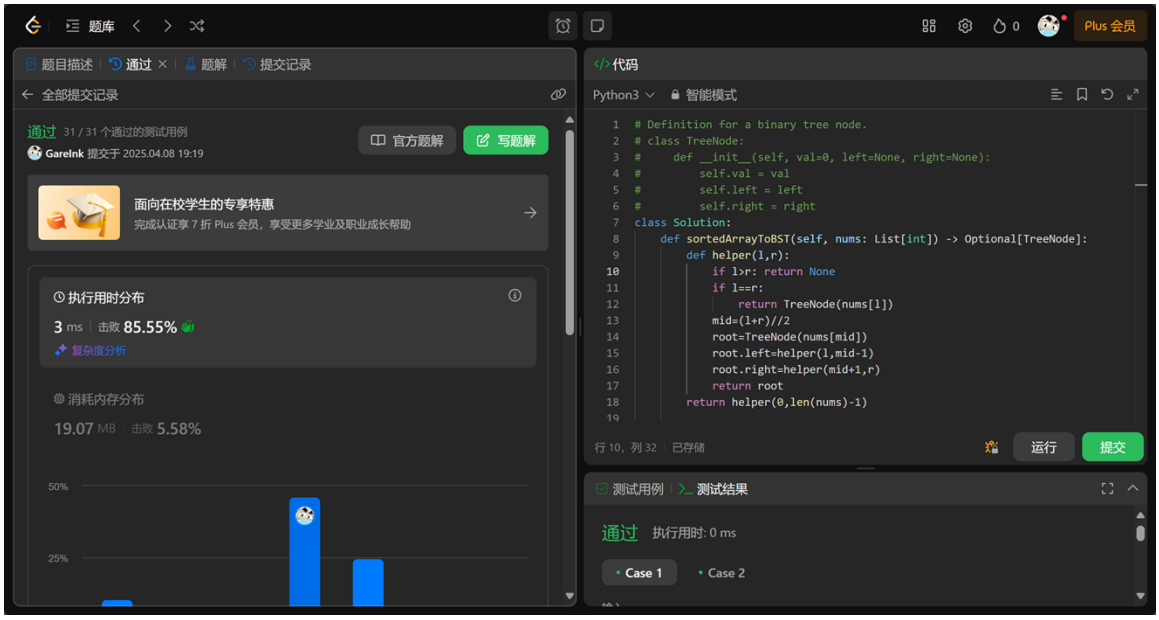
利用一个 helper 函数，为了保证二叉树是平衡的，每次都从有序数组的中间位置取值作为根节点，接着就是递归调用的神奇之处了：由于设置了退出条件为 `l>r`，此时返回 `None`，而这恰好对应叶子节点。于是在设定中间位置为根节点之后，只需要 `root.left=helper(l,mid-1); root.right=helper(mid+1,r)`两行即可实现左右子树的建立。目前感觉这种递归调用来直接对 root.left 和 root.right 赋值的题目还挺多的（在 oj）。

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 helper(l,r):
            if l>r: return None
            if l==r:
```

```
                return TreeNode(nums[l])
            mid=(l+r)//2
            root=TreeNode(nums[mid])
            root.left=helper(l,mid-1)
            root.right=helper(mid+1,r)
            return root
        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：

这题是看的题解，自己没有做出来，一是题目有点没读懂，二是想用 oop 建树但是由于节点的编号不是从 0 开始的，导致不好进行父子节点的构建关系（总之想了半天也没想到一个比较合适的构建方法）。于是用了答案给的只用字典和集合来建树的方法；同时又学到了答案的遍历方法，感觉是很新的递归。好难。。

代码：

```
from collections import defaultdict

def dfs(root):
    curr=[root]+nodes[root]
    curr.sort()
    for num in curr:
        if num==root:
            print(root)
        else:
            dfs(num)

nodes=defaultdict(list)
```

```
all,childs=set(),set()
n=int(input())
for _ in range(n):
    root,*child=list(map(int,input().split()))
    nodes[root].extend(child)
    all.update([root]+child)
    childs.update(child)
root=list(all-childs)[0]

dfs(root)
```

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：

思路是借用前序遍历的 dfs 框架，每次到叶子节点就把当前的数字存入 ans，然后为了保证递归右子树的时候不会受到递归左子树时对 curr 造成的改变，我选择了在 dfs 之前先把当前的值 pop 掉，然后在递归传入的参数中写成 `curr+[str(root.val)]`，总之效果就是：在遍历左子树和右子树的时候，保证 curr 均为以当前根节点数值结尾的状态，不要把左子树的更新状态带入到右子树了。

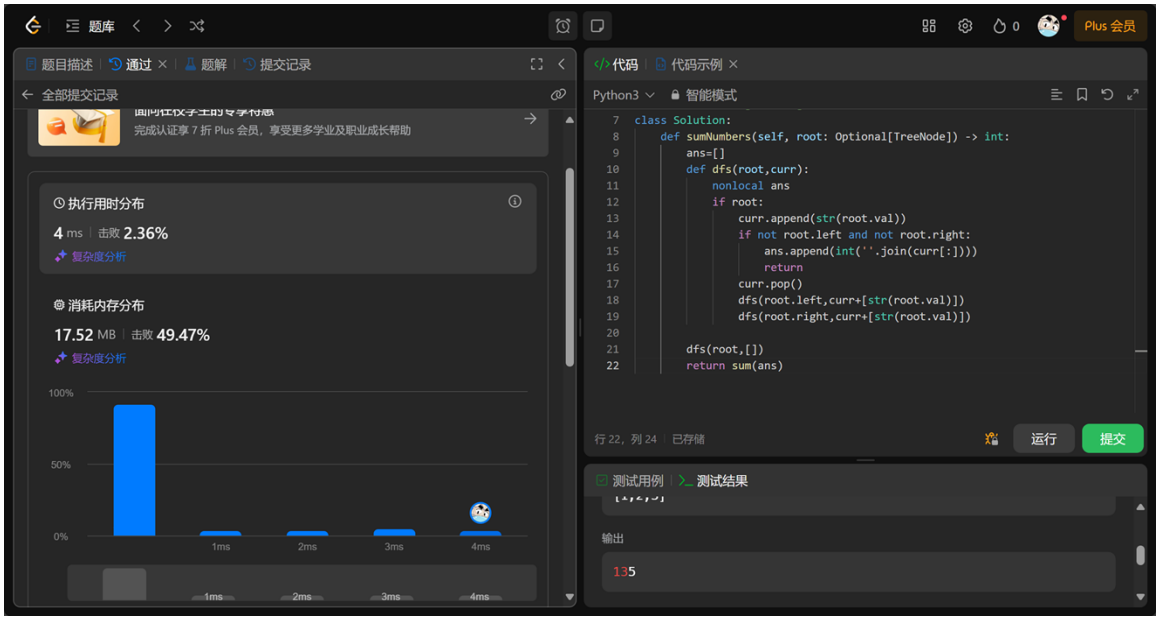
看了一圈题解感觉写法很简洁但是好像对我来说有点难理解。。虽然我的代码只击败了不到 5%的人但是对于我自己来说比较好理解，毕竟是树的前序遍历+dfs。。

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 sumNumbers(self, root: Optional[TreeNode]) -> int:
        ans=[]
        def dfs(root,curr):
            nonlocal ans
            if root:
                curr.append(str(root.val))
                if not root.left and not root.right:
                    ans.append(int(''.join(curr[:])))
                    return
                curr.pop()
                dfs(root.left,curr+[str(root.val)])
                dfs(root.right,curr+[str(root.val)])

        dfs(root,[])
        return sum(ans)
```

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/24729/

思路：

寒假记忆。。前序的第一个是根节点，找到中序这个根节点，其左边右边为左子树和右子树。用到了和作业第一题类似的递归构造树的写法

代码：

```
# 前序的第一个是根节点，找到中序这个根节点，其左边右边为左子树和右子树
```

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

def buildTree(preorder,inorder):
    if len(inorder)==1:
        return TreeNode(inorder[0])
    if len(inorder)==0 or len(preorder)==0:
        return None
    root=TreeNode(preorder[0])
    rootIdx=inorder.index(root.val)

    root.left=buildTree(preorder[1:rootIdx+1],inorder[:rootIdx])
    root.right=buildTree(preorder[rootIdx+1:],inorder[rootIdx+1:])

    return root

def postorderTraversal(root):
    if root:
        return
postorderTraversal(root.left)+postorderTraversal(root.right)+root.val
    return ''

while True:
    try:
        preorder=input()
        inorder=input()
        root=buildTree(preorder,inorder)
        print(postorderTraversal(root))
    except EOFError:
        break
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

T24729:括号嵌套树

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

思路：

主要是建树部分的栈的使用，这道题不是二叉树，孩子用 child 来存。最后保证 stack 都要 pop 掉，每次 pop 都用 root 来接，这样最后的 root 就一定是根节点了。后面的前序和后序就很常规了。

代码：

```
class TreeNode:
    def __init__(self,val,child=None):
        self.val=val
        self.child=child if child is not None else []

def build(s):
    stack=[]
    if not s:
        return None
    if len(s)==1: return TreeNode(s[0])
    # root=TreeNode(s[0])
    curr=None

    for i in s:
        if i=='(':
            stack.append(curr)
        elif i.isalpha():
            curr=TreeNode(i)
            if stack:
                stack[-1].child.append(curr)
        elif i==')':
            root=stack.pop()

    return root

def pre(root,ans):
    if root:
        if isinstance(root,TreeNode):
            ans.append(root.val)
            pre(root.child,ans)
        else:
            for node in root:
                ans.append(node.val)
                pre(node.child,ans)
    return ''

def post(root,ans):
    if root:
        if isinstance(root,TreeNode):
            post(root.child,ans)
            ans.append(root.val)
        else:
            for node in root:
```

```
                post(node.child,ans)
                ans.append(node.val)
    return ''


s=input()
root=build(s)
# print(root.child)
ans1,ans2=[],[]
pre(root,ans1)
post(root,ans2)
print(''.join(ans1))
print(''.join(ans2))
```

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：

需要维护的信息：

1. nums 中**相邻的元素和**以及**这一对元素的左边一个的索引**，通过一个升序的数据结构维护，方便从第一位查找当前从左侧开始找的最小数对和（这也是为什么要把索引考虑进去排序的原因），需要增加查找删除，借用**堆+懒删除**或者**有序集合**的方法；
2. 维护剩余下标，每次在进行加法运算时，需要得知当前 i 位置的左侧尚存在的 pre 索引和右侧尚存在的 nxt 以及 nxt2 索引，对这几个数组对的和进行更新（在 1 中的 heap），维护左右两侧最近的索引，使用**两个数组**或者**有序集合**或者**两个并查集**。

3. 每次加法运算后，如果都采用 O(nlogn)来统计当前逆序对太慢；考虑统计相邻元素逆序个数，降低到每次 O(n)；还可以在一开始建立第一步中的堆时统计起始相邻逆序个数 dec，在第二步的每次更新中对新出现的数对关系进行加减操作，这样时间复杂度将更低。

4. 最后完成的条件是 dec=0；维护答案 ans，每次 dec 循环一次就加 1.

法一：使用有序集合（Python 为 SortedList，cpp 为 set<>）

法二：堆+懒删除

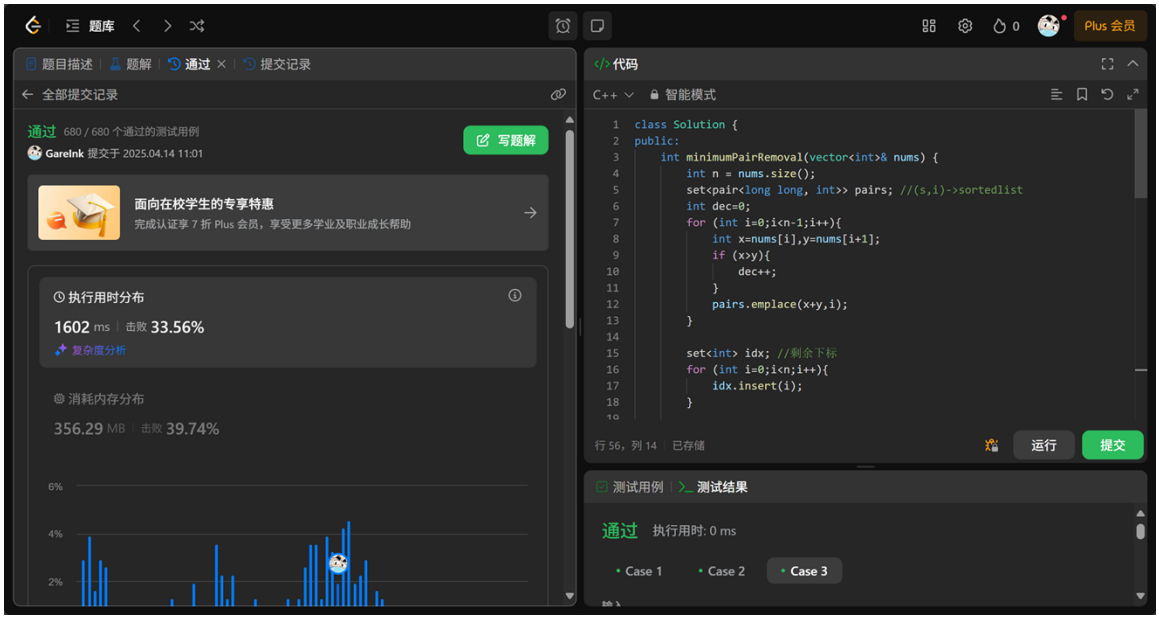
最后模拟双向链表：更新 nums[nxt]的 left 和 right 位置的索引，使得 right[nxt]的新 left 值与 left[nxt]相等，left[nxt]的新 right 值与 right[nxt]相等(因为 nxt 被删除了，有点并查集的 find 函数的找根节点的含义)

代码：

```
class Solution:
    def minimumPairRemoval(self, nums: List[int]) -> int:
        n=len(nums)
        heap=[]
        dec=0
        lazy=defaultdict(int)
        left=list(range(-1,n))
        right=list(range(1,n+1))
        ans=0

        for i in range(n-1):
            if nums[i]>nums[i+1]:
                dec+=1
            heappush(heap,(nums[i]+nums[i+1],i))

        while dec:
            ans+=1
            while lazy[heap[0]]:
                lazy[heappop(heap)]-=1
            s,i=heappop(heap)

            nxt=right[i]
            #i,nxt
            if nums[i]>nums[nxt]:
                dec-=1

            #pre,i
            pre=left[i]
            if pre>=0:
                if nums[pre]>nums[i]:
                    dec-=1
                if nums[pre]>s:
                    dec+=1
                heappush(heap,(nums[pre]+s,pre))
                lazy[(nums[pre]+nums[i],pre)]+=1

            #nxt,nxt2
```

```
            nxt2=right[nxt]
            if nxt2<n:
                if nums[nxt]>nums[nxt2]:
                    dec-=1
                if s>nums[nxt2]:
                    dec+=1
                heappush(heap,(s+nums[nxt2],i))
                lazy[(nums[nxt]+nums[nxt2],nxt)]+=1

            nums[i]=s
            #更新 nums[nxt]的 left 和 right 位置的索引，使得 right[nxt]的新 left 值与 left[nxt]相等，left[nxt]的新 right 值与 right[nxt]相等(因为 nxt 被删除了，有点并查集的 find 函数的找根节点的含义)
            left[right[nxt]],right[left[nxt]]=left[nxt],right[nxt]
        return ans
```

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>

最近一直在准备高数期中，没有在数算上花功夫，这次是交作业最迟的一次（最后一题真的太难了，周日用了一个上午把灵神的题解看懂了，数据结构用的太巧了，有点非人类了。。。学习了 cpp 的 set 的用法，原来 set 功能这么强大，但是感觉 python 语法更简洁易懂一点，容错率很大）

前面几道树的题是前几天做的，有点忘了，，不过只记得用递归写法建树是很巧妙的，root 确定下来，左子树和右子树无脑递归就好了。

加油！