# Assignment #8: 树为主

Updated 1704 GMT+8 Apr 8, 2025

2025 spring, Complied by 同学的姓名、院系

> **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：

根节点为中间值，左右子树递归构造即可。约2min。

代码：

```
class Solution:
    def sortedArrayToBST(self, nums: List[int]) -> Optional[TreeNode]:
        if nums:return
TreeNode(nums[len(nums)//2],self.sortedArrayToBST(nums[:len(nums)//2]),self.sorte
dArrayToBST(nums[len(nums)//2+1:]))
```

代码运行截图 （至少包含有"Accepted"）


代码 | Python3

```python
# 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]:
        if nums:return TreeNode(nums[len(nums)//2],self.sortedArrayToBST(nums[:le
```

## M27928:遍历树

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

思路：

用字典套列表的方式建树，并确定根节点。建树时把每个节点都视作自己的子节点，于是从根节点开始遍历，会经过除了根节点之外每个节点2次，第一次经过时不输出，第二次经过的时候再输出，即可保证次序。约20min。

代码：

```
n=int(input())
dic={}
not_root=set()
for _ in range(n):
    lst=list(map(int,input().split()))
    dic[lst[0]]=sorted(lst)
    not_root.update(lst[1:])

for r in dic.keys():
```

```
    if r not in not_root:
        root=r
        break
visited={root}
def visit(r):
    for _ in dic[r]:
        if _ not in visited:
            visited.add(_)
            visit(_)
        else:print(_)

visit(root)
```

代码运行截图 （至少包含有"Accepted"）

#48860376提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
n=int(input())
dic={}
not_root=set()
for _ in range(n):
    lst=list(map(int,input().split()))
    dic[lst[0]]=sorted(lst)
    not_root.update(lst[1:])

for r in dic.keys():
    if r not in not_root:
        root=r
        break
visited={root}
def visit(r):
    for _ in dic[r]:
        if _ not in visited:
            visited.add(_)
            visit(_)
        else:print(_)

visit(root)
```

基本信息

#: 48860376
题目: 27928
提交人: 24n2400010617
内存: 3736kB
时间: 20ms
语言: Python3
提交时间: 2025-04-09 16:02:44



English 帮助 关于

## LC129.求根节点到叶节点数字之和

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

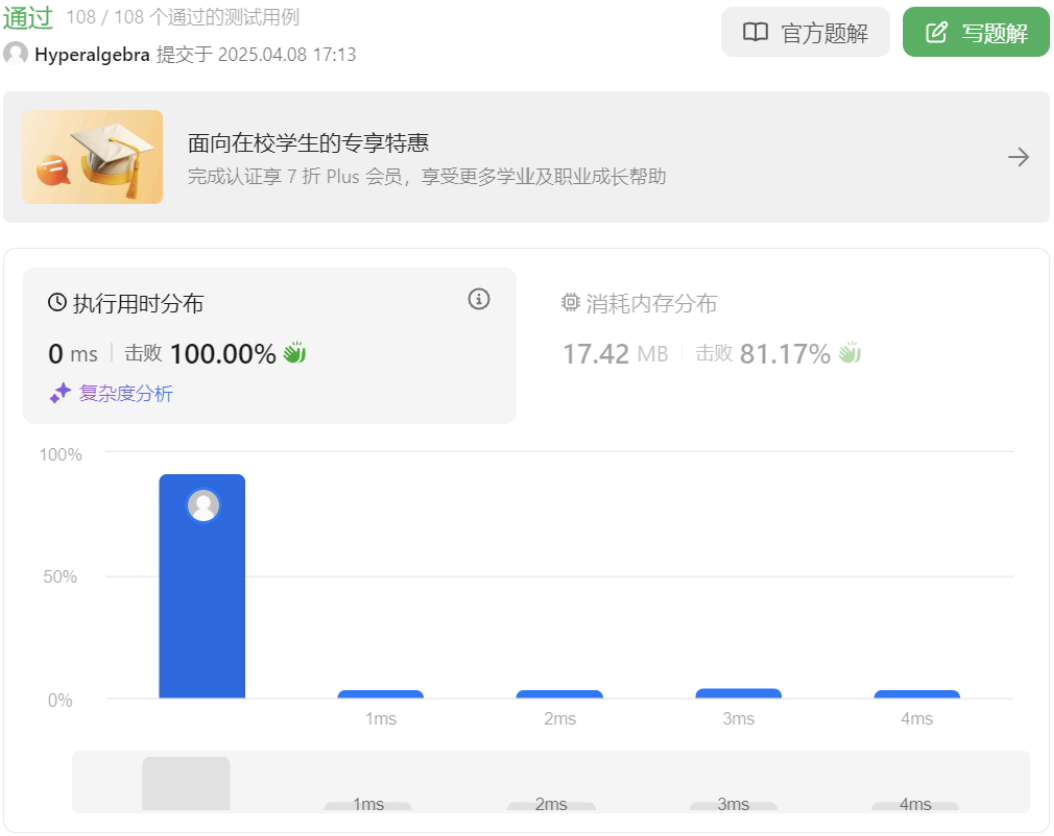
思路:

直接DFS遍历树，输出所有可能的数字字符串的列表，最后求和即可。约5min。

代码:

```
class Solution:
    def sumNumbers(self, root: Optional[TreeNode]) -> int:
        def dfs(r):
            if not r:return []
            if not r.left and not r.right:return [str(r.val)]
            return list(map(lambda x:str(r.val)+x,dfs(r.left)))+list(map(lambda
x:str(r.val)+x,dfs(r.right)))
        return sum(map(int,dfs(root)))
```

代码运行截图 （至少包含有"Accepted"）




代码 | Python3

```python
# 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:
        def dfs(r):
            if not r:return []
            if not r.left and not r.right:return [str(r.val)]
            return list(map(lambda x:str(r.val)+x,dfs(r.left)))+list(map(lambda x
        return sum(map(int,dfs(root)))
```

收起

## M22158:根据二叉树前中序序列建树

tree, http://cs101.openjudge.cn/practice/24729/

思路：

不建树，直接从前序、中序序列中分离出根节点、左子树和右子树，然后按照后序的规则递归即可。约5min。

代码：

```
while 1:
    try:pre_order,in_order=input(),input()
    except:break
    def postorder(preorder,inorder):
        if preorder:
            root=preorder[0]
            ind=inorder.find(root)
            postorder(preorder[1:ind+1],inorder[:ind])
            postorder(preorder[ind+1:],inorder[ind+1:])
            print(root,end='')
    postorder(pre_order,in_order)
    print()
```

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：

不建树，将原字符串删去括号、逗号就直接是前序遍历序列，后序遍历序列只需从原字符串中分离出根节点和所有子树，根节点就是第一个字母，子树只要在每个逗号前检测左右括号个数是否相等，若相等则分离出一棵子树即可。约10min。

代码：

```
s=input()
print(s.replace('(','').replace(')','').replace(',',''))
def postorder(tree):
    left=right=0
    cur=2
    if len(tree)>2:
        for i in range(2,len(tree)-1):
            if tree[i]=='(':left+=1
            elif tree[i]==')':
                right+=1
```

```
            elif tree[i]==',' and left==right:
                postorder(tree[cur:i])
                cur=i+1
        if cur<len(tree)-1:postorder(tree[cur:-1])
    print(tree[0],end='')
postorder(s)
```

代码运行截图 （至少包含有"Accepted"）

#48861242提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
s=input()
print(s.replace('(','').replace(')','').replace(',',''))
def postorder(tree):
    left=right=0
    cur=2
    if len(tree)>2:
        for i in range(2,len(tree)-1):
            if tree[i]=='(':left+=1
            elif tree[i]==')':
                right+=1
            elif tree[i]==',' and left==right:
                postorder(tree[cur:i])
                cur=i+1
        if cur<len(tree)-1:postorder(tree[cur:-1])
    print(tree[0],end='')
postorder(s)
```

基本信息

#: 48861242
题目: 24729
提交人: 24n2400010617
内存: 3636kB
时间: 22ms
语言: Python3
提交时间: 2025-04-09 16:43:58



English 帮助 关于

## 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:

用`prev`、`next`列表模拟双向链表，直接用原数组的下标表示结点。另外开一个`heapq`记录相邻两数和及下标，同时用数组`sums`维护`nums`数组中"实际"的相邻两数和，采用懒更新，每次从堆顶弹出元素时检验是否与`sums`中的数值相符，如果相符再进行操作。同时为了检验数组是否递增，维护相邻两个元素的大小关系中"左小于等于右"的个数，即`inc_count`，如果总数等于数组长度减1即说明数组递增。此外，在数组两端加上`float('inf')`"保护圈"，避免对边界的特判。每次操作删除两个元素，依次更新操作次数`counts`、实际总和`sums`、堆`q`、数组`nums`、链表连接关系`prev`、`next`、递增判据`inc_count`。直到数组递增时输出即可。

代码:

```
class Solution:
    def minimumPairRemoval(self, nums: List[int]) -> int:
        nums=[float("inf")]+nums+[float('inf')]
        n=len(nums)
        prev,next=[0]+list(range(n-1)),list(range(1,n))+[n-1]
        sums,q=[float('inf')]+[0] * (n-3)+[float('inf')]*2,[]
        inc_count=0
        for i in range(1,n-2):
            sums[i]=nums[i]+nums[i+1]
            q.append((sums[i],i))
```

```
            inc_count+=int(nums[i+1]>=nums[i])
        counts=0
        heapify(q)
        while 1:
            if inc_count==n-3-counts:return counts
            s,ind=heappop(q)
            if s==float('inf') or sums[ind]!=s:continue
            counts+=1

            sums[prev[ind]]+=nums[next[ind]]
            sums[ind]+=nums[next[next[ind]]]
            sums[next[ind]]=float('inf')

            heappush(q,(sums[prev[ind]],prev[ind]))
            heappush(q,(sums[ind],ind))


 before=int(nums[ind]>=nums[prev[ind]])+int(nums[next[ind]]>=nums[ind])+int(nums[
next[next[ind]]]>=nums[next[ind]])

            nums[ind]+=nums[next[ind]]

            temp=next[next[ind]]
            next[ind]=next[next[ind]]
            prev[temp]=ind

            cur=int(nums[ind]>=nums[prev[ind]])+int(nums[next[ind]]>=nums[ind])
            inc_count+=cur-before
```

代码运行截图 （至少包含有"Accepted"）

通过 680 / 680 个通过的测试用例

Hyperalgebra 提交于 2025.04.07 09:21

写题解

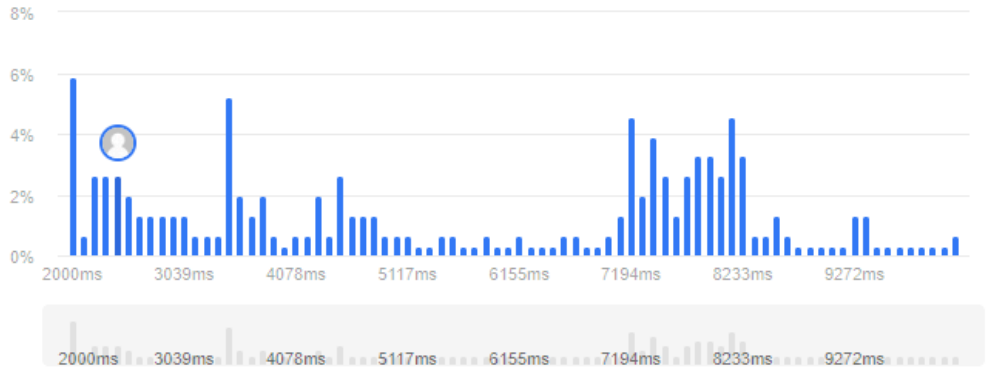
执行用时分布

2450 ms | 击败 85.62%

复杂度分析

消耗内存分布

63.61 MB | 击败 24.18%







代码 | Python3

```python
class Solution:
    def minimumPairRemoval(self, nums: List[int]) -> int:
        nums=[float("inf")]+nums+[float('inf')]
        n=len(nums)
        prev,next=[0]+list(range(n-1)),list(range(1,n))+[n-1]
        sums,q=[float('inf')]+[0] * (n-3)+[float('inf')]*2,[]
        inc_count=0
        for i in range(1,n-2):
            sums[i]=nums[i]+nums[i+1]
            q.append((sums[i],i))
            inc_count+=int(nums[i+1]>=nums[i])
        counts=0
        heapify(q)
        while 1:
            if inc_count==n-3-counts:return counts
            s,ind=heappop(q)
            if s==float('inf') or sums[ind]!=s:continue
            counts+=1

            sums[prev[ind]]+=nums[next[ind]]
            sums[ind]+=nums[next[next[ind]]]
            sums[next[ind]]=float('inf')

            heappush(q,(sums[prev[ind]],prev[ind]))
            heappush(q,(sums[ind],ind))

            before=int(nums[ind]>=nums[prev[ind]])+int(nums[next[ind]]>=nums[ind]

            nums[ind]+=nums[next[ind]]

            temp=next[next[ind]]
            next[ind]=next[next[ind]]
            prev[temp]=ind

            cur=int(nums[ind]>=nums[prev[ind]])+int(nums[next[ind]]>=nums[ind])
            inc_count+=cur-before
```

收起

## 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。

这两周准备期中考，LeetCode周赛没空打了。这周作业最后一题直接放了周赛压轴，原来比赛完自己看题时只想到一个复杂到不想动手实现的思路，但是读了这题的标签后，感觉想法就非常自然了，写起来负担也不是特别大。感觉还是得学会选用合适的数据结构，特别是链表的作用值得关注。

现在考试比较忙，就每天完成一下每日选做（如果有寒假没做过的），等到考完之后再多练一些题吧。