数据结构与算法B 作业8： 🌲

2025 fall

> **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本人头像，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

E108.将有序数组转换为二叉搜索树

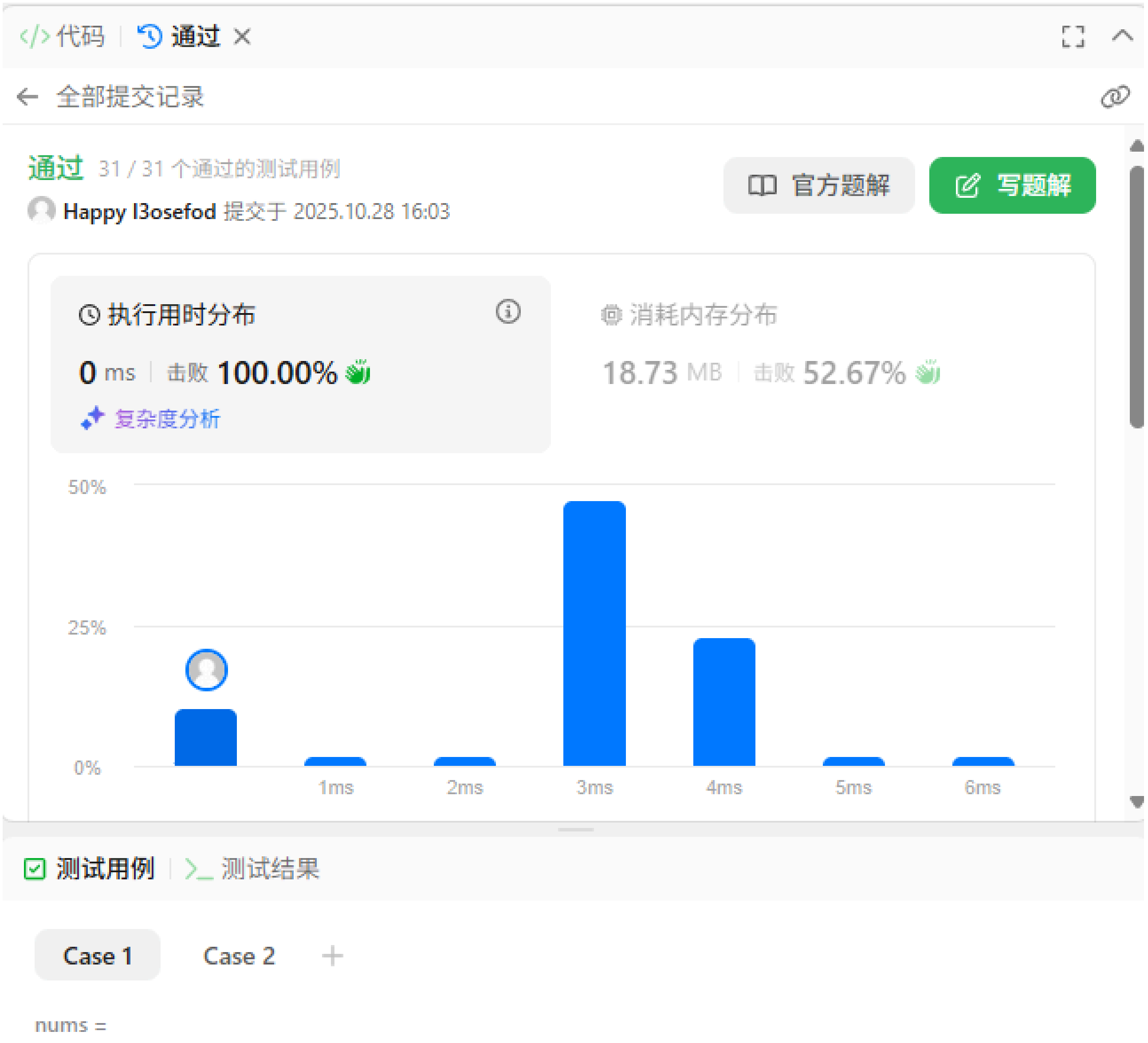
https://leetcode.cn/problems/convert-sorted-array-to-binary-search-tree/

思路：
充分利用递归的思想。由于输入的数组为有序的数组，直接选取中位数为root，将中位数的左半部分和右半部分分别backtrack，同样选取中位数、对左半部分和右半部分构建子树即可。
耗时1h

代码：

```python
from collections import deque
from typing import List, Optional
# 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 backtrack(num_half: List[int]):
            if len(num_half) == 1:
                mid = TreeNode(num_half[0])
            elif len(num_half) == 2:
                mid = TreeNode(num_half[1])
                mid.left = TreeNode(num_half[0])
            elif len(num_half) == 3:
                mid = TreeNode(num_half[1])
                mid.left = TreeNode(num_half[0])
                mid.right = TreeNode(num_half[2])
            else:
                index = len(num_half)//2
                left = backtrack(num_half[:index])
                mid = TreeNode(num_half[index])
                right = backtrack(num_half[index+1:])
                mid.left = left
                mid.right = right
            return mid

        return backtrack(nums)
```

代码运行截图 （至少包含有"Accepted"）


</> 代码 | 通过 ×

← 全部提交记录

通过 31 / 31 个通过的测试用例

Happy l3osefod 提交于 2025.10.28 16:03

官方题解 写题解

执行用时分布

0 ms | 击败 100.00%

复杂度分析

消耗内存分布

18.73 MB | 击败 52.67%

测试用例 | >_ 测试结果

Case 1 Case 2 +

nums =

M07161: 森林的带度数层次序列存储

tree, http://cs101.openjudge.cn/practice/07161/

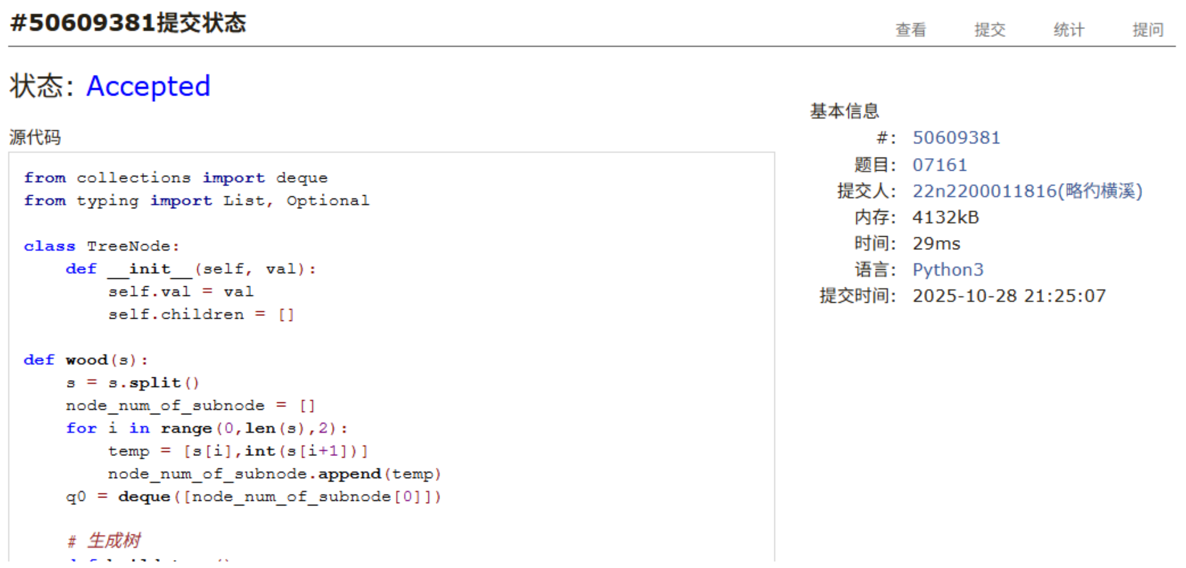
思路：
这题主要难点在于如何按照题目的顺序构建树。在这里要用到队列，将节点的val和子节点数加入队列，再按照子节点数将子节点依次加入队列中，并加入母节点的子节点内。在这一部分，一开始用了两个队列来形成树，但是应该是出现了一些问题，树形成有误，最后改了好久。耗时2h。

代码：

```python
from collections import deque
from typing import List, Optional

class TreeNode:
    def __init__(self, val):
        self.val = val
        self.children = []
```

```python
def wood(s):
    s = s.split()
    node_num_of_subnode = []
    for i in range(0,len(s),2):
        temp = [s[i],int(s[i+1])]
        node_num_of_subnode.append(temp)
    q0 = deque([node_num_of_subnode[0]])

    # 生成树
    def build_tree():
        root = TreeNode(node_num_of_subnode[0][0])
        q0 = deque([[root,node_num_of_subnode[0][1]]])
        index = 0
        while q0:
            current = q0.popleft()
            for _ in range(current[1]):
                index += 1
                child = TreeNode(node_num_of_subnode[index][0])
                q0.append([child,node_num_of_subnode[index][1]])
                current[0].children.append(child)
        return root

    head = build_tree()
    result = []

    # 利用迭代来排序
    def preorder(node: TreeNode):
        for i in node.children:
            preorder(i)
        result.append(node.val)

    preorder(head)
    return result

def main():
    n = int(input())
    result = []
    for i in range(n):
        result += wood(input())
    print(' '.join(result))

if __name__ == '__main__':
    main()
```

代码运行截图 （至少包含有"Accepted"）



#50609381提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
from collections import deque
from typing import List, Optional

class TreeNode:
    def __init__(self, val):
        self.val = val
        self.children = []

def wood(s):
    s = s.split()
    node_num_of_subnode = []
    for i in range(0,len(s),2):
        temp = [s[i],int(s[i+1])]
        node_num_of_subnode.append(temp)
    q0 = deque([node_num_of_subnode[0]])

    # 生成树
```

基本信息

#:	50609381
题目:	07161
提交人:	22n2200011816(略约横溪)
内存:	4132kB
时间:	29ms
语言:	Python3
提交时间:	2025-10-28 21:25:07

M27928: 遍历树

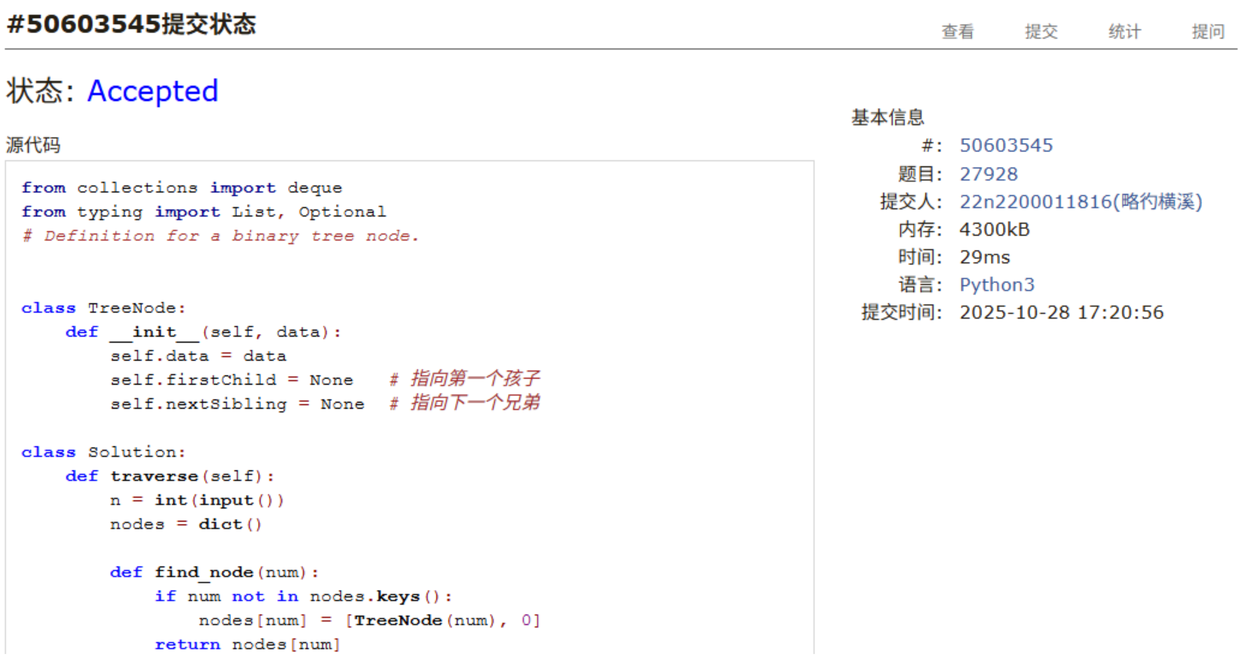
adjacency list, dfs, http://cs101.openjudge.cn/practice/27928/

思路:
这里练习了用课件中介绍的将树转变为二叉树（也就是孩子-兄弟表示法）来表示树。首先是构建树，在依次读入时，使用字典来存储已经变成TreeNode的数据（find），从而帮组构建树。
随后是按照要求遍历。在这里，题目其实描述的不是很清楚，要求应该是：对一个节点，按照孩子和父亲由小到大进行遍历，如果遍历到的节点有孩子，就按照上面的规则继续迭代，否则输出节点的value。最后应用迭代的写法输出即可。用时1h

代码:

```python
from collections import deque
from typing import List, Optional
# Definition for a binary tree node.


class TreeNode:
    def __init__(self, data):
        self.data = data
        self.firstChild = None   # 指向第一个孩子
        self.nextSibling = None  # 指向下一个兄弟

class Solution:
    def traverse(self):
        n = int(input())
        nodes = dict()

        # 使用字典存储和读取node
        def find_node(num):
            if num not in nodes.keys():
                nodes[num] = [TreeNode(num), 0]
            return nodes[num]

        head = None
        for i in range(n):
            temp = [int(x) for x in input().split()]
            for j in range(len(temp)-1):
                temp_node1 = find_node(temp[j])
                temp_node2 = find_node(temp[j + 1])
                if j == 0:
                    temp_node1[0].firstChild = temp_node2[0]
                else:
                    temp_node1[0].nextSibling = temp_node2[0]
                temp_node2[1] += 1
                nodes[temp[j]] = temp_node1
                nodes[temp[j+1]] = temp_node2

        for i in nodes.keys():
            if nodes[i][1] == 0:
                head = nodes[i][0]
                break

        def smallorder(root: Optional[TreeNode]):
            if not root:
                return
            q = [[root, root.data]]
            child = root.firstChild
            while child:
                q.append([child, child.data])
                child = child.nextSibling
            q = deque(sorted(q, key=lambda x: x[1]))
            while q:
                temp_node = q.popleft()
                if temp_node[0] == root:
                    print(temp_node[1])
                else:
                    smallorder(temp_node[0])

        smallorder(head)
        return 0


if __name__ == '__main__':
    solut = Solution()
    solut.traverse()
```

代码运行截图 （至少包含有"Accepted"）


#50603545提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque
from typing import List, Optional
# Definition for a binary tree node.


class TreeNode:
    def __init__(self, data):
        self.data = data
        self.firstChild = None    # 指向第一个孩子
        self.nextSibling = None   # 指向下一个兄弟

class Solution:
    def traverse(self):
        n = int(input())
        nodes = dict()

        def find_node(num):
            if num not in nodes.keys():
                nodes[num] = [TreeNode(num), 0]
            return nodes[num]
```

基本信息

#: 50603545
题目: 27928
提交人: 22n2200011816(略约横溪)
内存: 4300kB
时间: 29ms
语言: Python3
提交时间: 2025-10-28 17:20:56

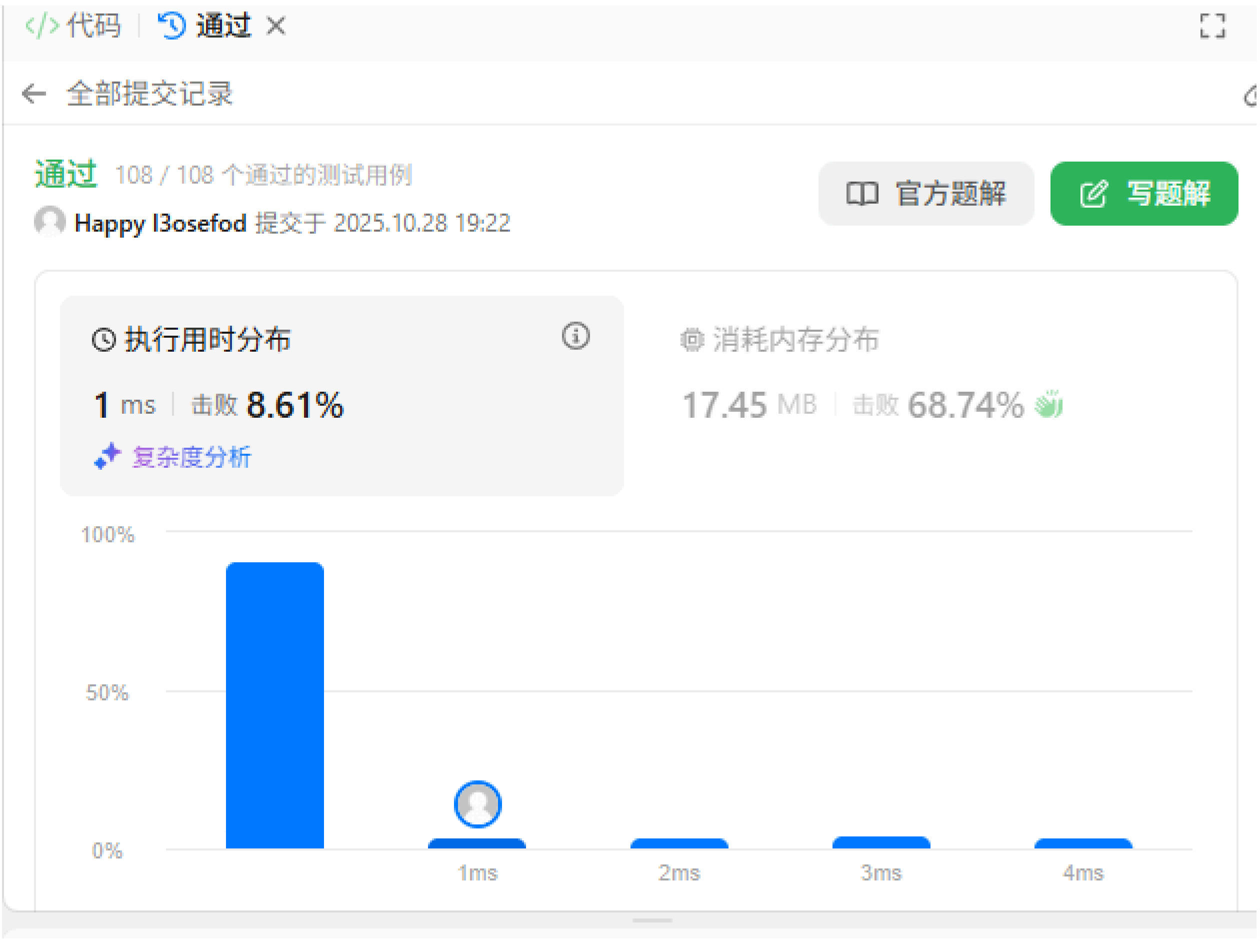
M129.求根节点到叶节点数字之和

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

思路：
这次作业里面比较简单的树的题目。只要按照前序遍历的方式遍历，随后将遍历到的节点加入trace中，如果遍历到底，就将trace加入到result内，最后将result组合、相加、输出即可。用时0.5 h。
代码

```python
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:
        trace = []
        result = []
        def backtrack(node):
            if node is None:
                return None
            trace.append(node.val)
            if node.left is None and node.right is None:
                result.append(trace[:])
            else:
                backtrack(node.left)
                backtrack(node.right)
            trace.pop()
        backtrack(root)
        sum = 0
        for i in result:
            temp = 0
            for j in range(len(i)):
                temp = temp*10 + i[j]
            sum += temp
        return sum
```

代码运行截图（至少包含有"Accepted"）



通过 108 / 108 个通过的测试用例

Happy l3osefod 提交于 2025.10.28 19:22

执行用时分布

1 ms | 击败 8.61%

消耗内存分布

17.45 MB | 击败 68.74%

M24729: 括号嵌套树

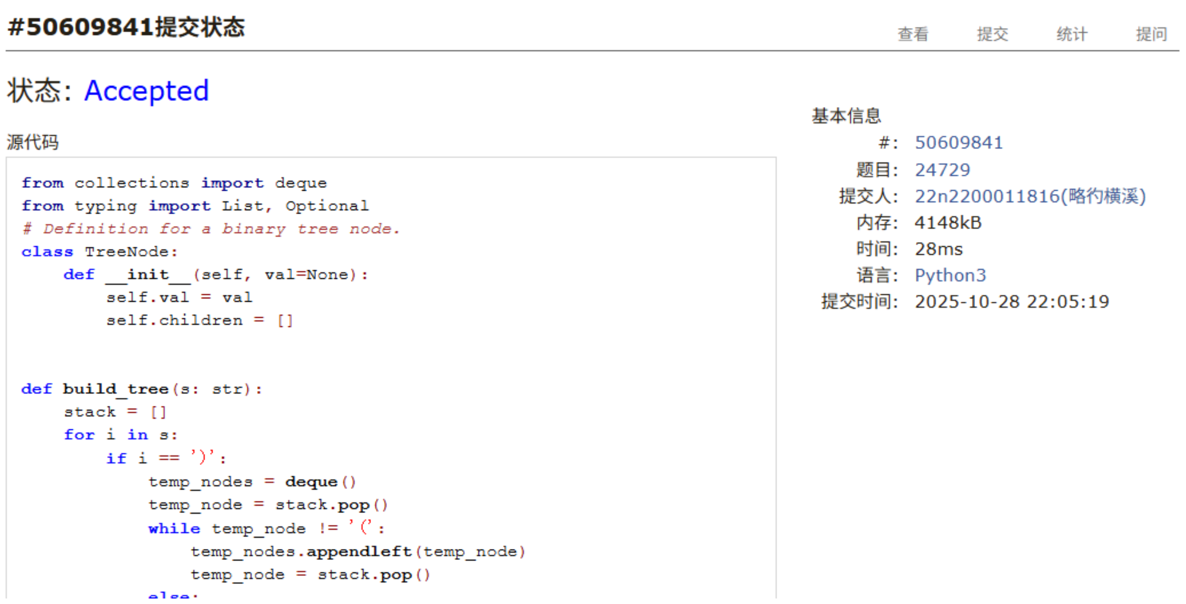
dfs, stack, http://cs101.openjudge.cn/practice/24729/

思路：
利用了栈的写法，将除了'）'以外的所有元素都加入栈中，而如果遍历到'）'，则将栈末尾的所有的元素出栈直至遇到'（'，并将出栈的所有节点放入栈末最后一个元素的字节点中。用时1 h
代码

```
from collections import deque
from typing import List, Optional
# Definition for a binary tree node.
class TreeNode:
    def __init__(self, val=None):
```

```python
        self.val = val
        self.children = []


def build_tree(s: str):
    stack = []
    for i in s:
        if i == ')':
            temp_nodes = deque()
            temp_node = stack.pop()
            while temp_node != '(':
                temp_nodes.appendleft(temp_node)
                temp_node = stack.pop()
            else:
                stack[-1].children = list(temp_nodes)
        elif i == '(':
            stack.append('(')
        elif i != ',':
            stack.append(TreeNode(i))
    return stack[-1]


def preorder(node: TreeNode, result: str):
    result += node.val
    for i in node.children:
        result = preorder(i, result)
    return result


def postorder(node: TreeNode, result: str):
    for i in node.children:
        result = postorder(i, result)
    result += node.val
    return result

def main():
    s = input()
    root = build_tree(s)
    print(preorder(root, ''))
    print(postorder(root, ''))


if __name__ == '__main__':
    main()
```

代码运行截图（至少包含有"Accepted"）



#50609841提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque
from typing import List, Optional
# Definition for a binary tree node.
class TreeNode:
    def __init__(self, val=None):
        self.val = val
        self.children = []


def build_tree(s: str):
    stack = []
    for i in s:
        if i == ')':
            temp_nodes = deque()
            temp_node = stack.pop()
            while temp_node != '(':
                temp_nodes.appendleft(temp_node)
                temp_node = stack.pop()
            else:
```

基本信息

#:	50609841
题目:	24729
提交人:	22n2200011816(略约横溪)
内存:	4148kB
时间:	28ms
语言:	Python3
提交时间:	2025-10-28 22:05:19

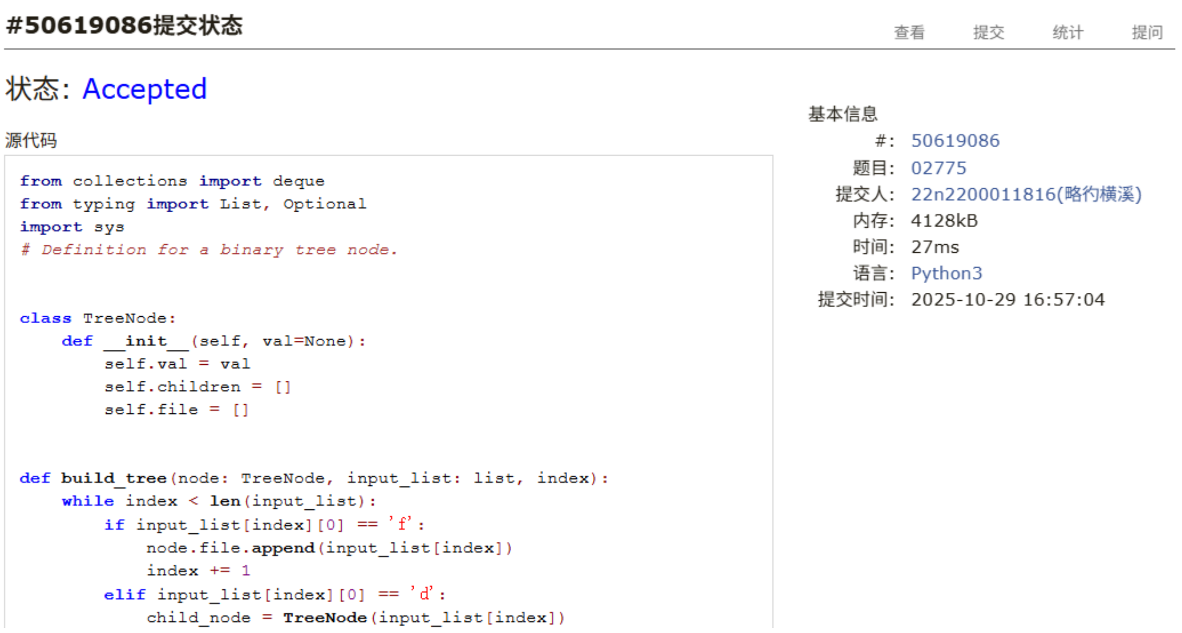
T02775: 文件结构“图”

tree, http://cs101.openjudge.cn/practice/02775/

思路：
由于最近都在练习树的解法，这题也很自然地想到了树。
首先是存储文件结构，对树稍作修改，self.children用于目录下的存储dir，self.file用于存储file，随后用迭代的写法，如果读取到dir，则进入下一层；读到']'，则退出本层迭代返回上一层即可。
其次是输出，采用类似前序遍历的写法对首先对节点的dir进行读取，并在最后附上file即可。
由于‘DATA SET 1:’这一行少输出了一个冒号，导致一直WA，openjudge不能给数据错在哪里，真是垃圾-_-。用时3h
代码：

```python
from collections import deque
from typing import List, Optional
import sys
# Definition for a binary tree node.


class TreeNode:
    def __init__(self, val=None):
        self.val = val
        self.children = []
        self.file = []


def build_tree(node: TreeNode, input_list: list, index):
    while index < len(input_list):
        if input_list[index][0] == 'f':
            node.file.append(input_list[index])
            index += 1
        elif input_list[index][0] == 'd':
            child_node = TreeNode(input_list[index])
            index += 1
            child_node, index = build_tree(child_node, input_list, index)
            node.children.append(child_node)
        elif input_list[index] == ']':
            index += 1
            return node, index
    return node, index


def output(node: TreeNode, data_set_num):
    print(f'DATA SET {data_set_num}:')
    print(f'ROOT')
    output_list = []

    def backtrack(node1, depth):
        for i in node1.children:
            output_list.append([depth, i.val])
            backtrack(i, depth + 1)
        node1.file = sorted(node1.file)
        for j in node1.file:
            output_list.append([depth - 1, j])

    backtrack(node, depth=1)
    for i in output_list:
        print('|     '*i[0] + i[1])


def main():
    lines = sys.stdin.read().splitlines()
    dataset_count = 0
    first_dataset = True

    i = 0
    index_left = 0
    while i < len(lines) and lines[i] != "#":
        if lines[i] == "*":
            dataset_count += 1
            root = TreeNode()
            root, _ = build_tree(root, lines[index_left:i], index=0)
            index_left = i+1
            # 在非第一个数据集前输出空行
            if not first_dataset:
                print()
            first_dataset = False

            output(root, dataset_count)

            # 输出数据集内容...
        i += 1

if __name__ == '__main__':
    main()
```

代码运行截图 （至少包含有"Accepted"）



#50619086提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque
from typing import List, Optional
import sys
# Definition for a binary tree node.


class TreeNode:
    def __init__(self, val=None):
        self.val = val
        self.children = []
        self.file = []


def build_tree(node: TreeNode, input_list: list, index):
    while index < len(input_list):
        if input_list[index][0] == 'f':
            node.file.append(input_list[index])
            index += 1
        elif input_list[index][0] == 'd':
            child_node = TreeNode(input_list[index])
```

基本信息

#:	50619086
题目:	02775
提交人:	22n2200011816(略仢横溪)
内存:	4128kB
时间:	27ms
语言:	Python3
提交时间:	2025-10-29 16:57:04

2. 学习总结和个人收获

首先，对树、dfs、递归的理解加深了很多，实践了很多写法。
其次，对oop的理解深了很多，现在看到题目就不由自主地将程序拆解成各个模块，分别编程。
最后，我花了一些时间在github上建了一个仓库Xiwei-1D20/Data-Structure-and-Algorithm-B: 用于保存个人在完成数据结构与算法期间的代码。每天将学习结果放上去还是很有成就感的。