Assignment #5: "树"算：概念、表示、解析、遍历

Updated 2124 GMT+8 March 17, 2024

2024 spring, Complied by ==陈奕好 工学院==

说明：

1）The complete process to learn DSA from scratch can be broken into 4 parts:

Learn about Time complexities, learn the basics of individual Data Structures, learn the basics of Algorithms, and practice Problems.

2）请把每个题目解题思路（可选），源码Python, 或者C++（已经在Codeforces/Openjudge上AC），截图（包含Accepted），填写到下面作业模版中（推荐使用 typora https://typoraio.cn ，或者用word）。AC 或者没有AC，都请标上每个题目大致花费时间。

3）提交时候先提交pdf文件，再把md或者doc文件上传到右侧“作业评论”。Canvas需要有同学清晰头像、提交文件有pdf、"作业评论"区有上传的md或者doc附件。

4）如果不能在截止前提交作业，请写明原因。

编程环境

==（请改为同学的操作系统、编程环境等）==

操作系统：macOS Sonoma 14.4 (23E214)

Python编程环境：PyCharm 2023.3.1 (Professional Edition)

1. 题目

27638: 求二叉树的高度和叶子数目

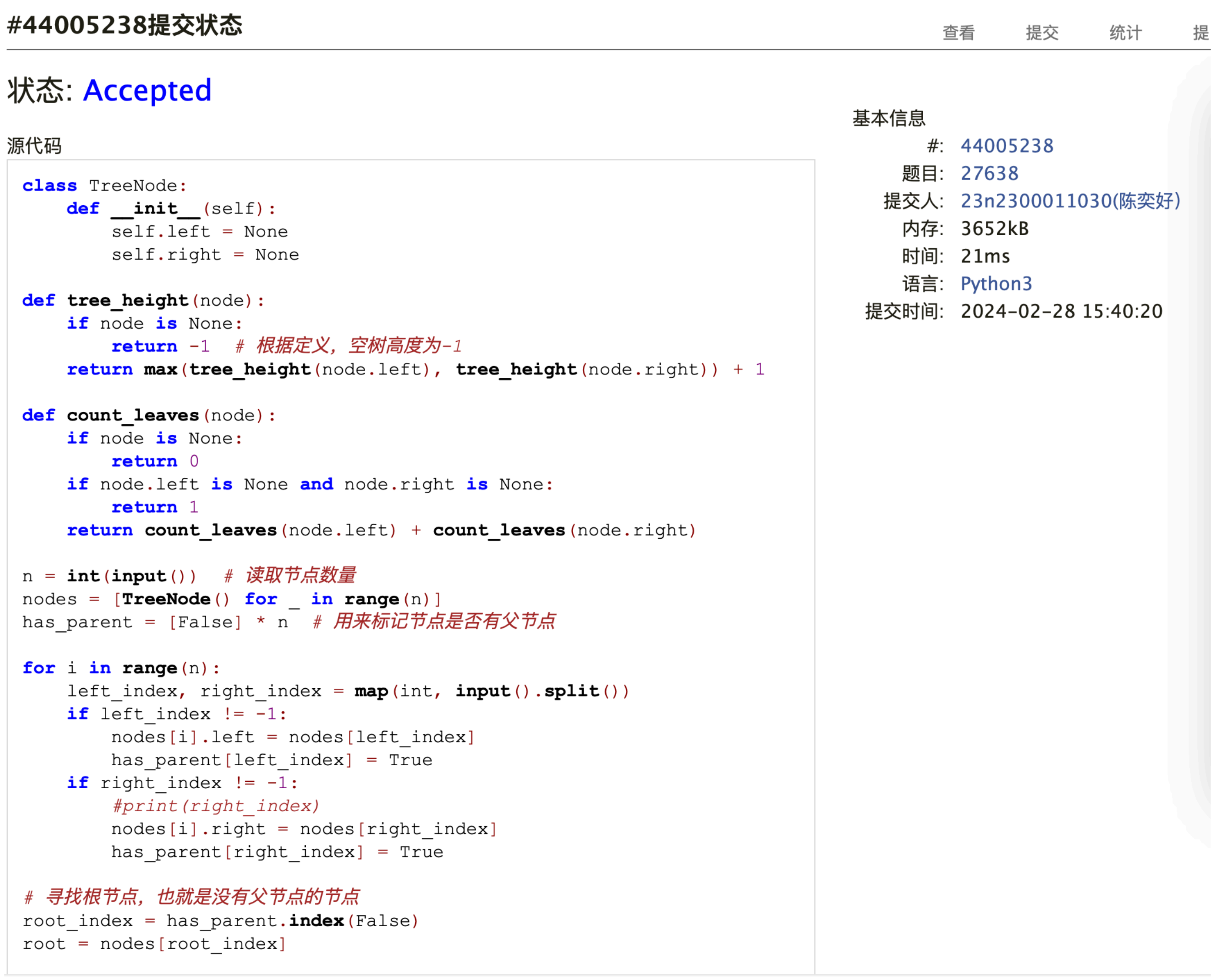
http://cs101.openjudge.cn/practice/27638/

思路：正常建树，从下往上从左往右遍历树，每次到无子节点就是叶子节点，ans += 1。这题还有一个坑就是根节点的寻找，学习copilot的标记找法，在输入的过程中，打好标记点。

代码

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

def tree_height(node):
    if node is None:
        return -1  # 根据定义，空树高度为-1
    return max(tree_height(node.left), tree_height(node.right)) + 1

def count_leaves(node):
    if node is None:
        return 0
    if node.left is None and node.right is None:
        return 1
    return count_leaves(node.left) + count_leaves(node.right)

n = int(input())  # 读取节点数量
nodes = [TreeNode() for _ in range(n)]
has_parent = [False] * n  # 用来标记节点是否有父节点

for i in range(n):
    left_index, right_index = map(int, input().split())
    if left_index != -1:
        nodes[i].left = nodes[left_index]
        has_parent[left_index] = True
    if right_index != -1:
        #print(right_index)
        nodes[i].right = nodes[right_index]
        has_parent[right_index] = True

# 寻找根节点，也就是没有父节点的节点
root_index = has_parent.index(False)
root = nodes[root_index]

# 计算高度和叶子节点数
height = tree_height(root)
leaves = count_leaves(root)

print(f"{height} {leaves}")
```

代码运行截图 ==（至少包含有"Accepted"）==



#44005238提交状态

查看 提交 统计 提

状态: Accepted

源代码

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

def tree_height(node):
    if node is None:
        return -1  # 根据定义，空树高度为-1
    return max(tree_height(node.left), tree_height(node.right)) + 1

def count_leaves(node):
    if node is None:
        return 0
    if node.left is None and node.right is None:
        return 1
    return count_leaves(node.left) + count_leaves(node.right)

n = int(input())  # 读取节点数量
nodes = [TreeNode() for _ in range(n)]
has_parent = [False] * n  # 用来标记节点是否有父节点

for i in range(n):
    left_index, right_index = map(int, input().split())
    if left_index != -1:
        nodes[i].left = nodes[left_index]
        has_parent[left_index] = True
    if right_index != -1:
        #print(right_index)
        nodes[i].right = nodes[right_index]
        has_parent[right_index] = True

# 寻找根节点，也就是没有父节点的节点
root_index = has_parent.index(False)
root = nodes[root_index]
```

基本信息

#:	44005238
题目:	27638
提交人:	23n2300011030(陈奕好)
内存:	3652kB
时间:	21ms
语言:	Python3
提交时间:	2024-02-28 15:40:20

24729: 括号嵌套树

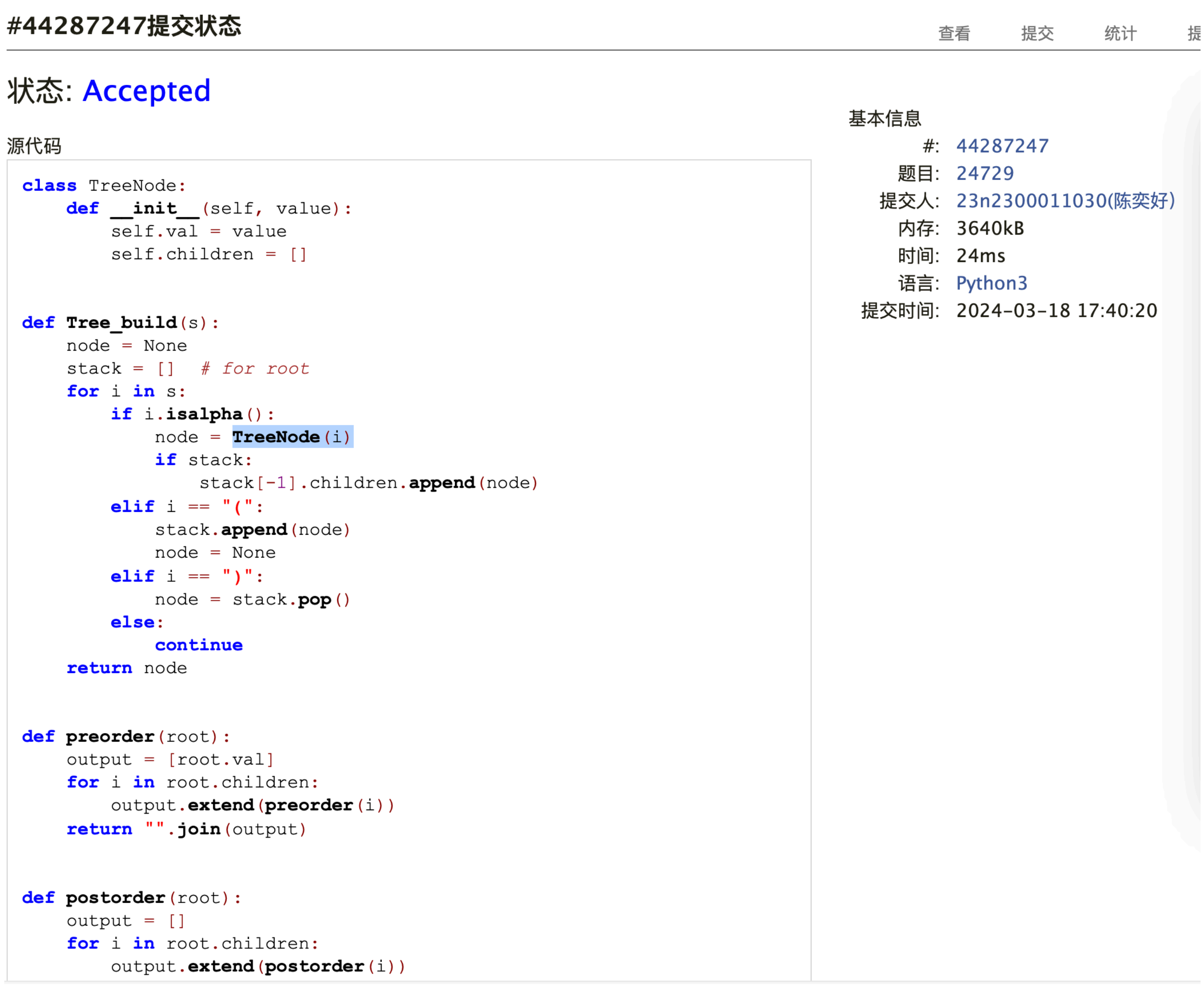
http://cs101.openjudge.cn/practice/24729/

思路：很标准的一道树题。首先建好树的类，stack用于存储节点，node用于遍历指针，如果是字母，先建树，如果stack里已经有元素了，则加入children；如果遇到“（”，node成为子节点，加入stack，则将之后的第一个“）”之前的节点纳入node的children里，stack再弹出node。这里有点recursion的感觉，重点在分层。

代码

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


def Tree_build(s):
    node = None
    stack = []  # for root
    for i in s:
        if i.isalpha():
            node = TreeNode(i)
            if stack:
                stack[-1].children.append(node)
        elif i == "(":
            stack.append(node)
            node = None
        elif i == ")":
            node = stack.pop()
        else:
            continue
    return node


def preorder(root):
    output = [root.val]
    for i in root.children:
        output.extend(preorder(i))
    return "".join(output)


def postorder(root):
    output = []
    for i in root.children:
        output.extend(postorder(i))
    output.append(root.val)
    return "".join(output)


s = input()
root = Tree_build(s)
print(preorder(root))
print(postorder(root))
```

代码运行截图 ==（至少包含有"Accepted"）==



#44287247提交状态

状态: Accepted

源代码

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


def Tree_build(s):
    node = None
    stack = []  # for root
    for i in s:
        if i.isalpha():
            node = TreeNode(i)
            if stack:
                stack[-1].children.append(node)
        elif i == "(":
            stack.append(node)
            node = None
        elif i == ")":
            node = stack.pop()
        else:
            continue
    return node


def preorder(root):
    output = [root.val]
    for i in root.children:
        output.extend(preorder(i))
    return "".join(output)


def postorder(root):
    output = []
    for i in root.children:
        output.extend(postorder(i))
```

基本信息

#:	44287247
题目:	24729
提交人:	23n2300011030(陈奕好)
内存:	3640kB
时间:	24ms
语言:	Python3
提交时间:	2024-03-18 17:40:20

02775: 文件结构“图”

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

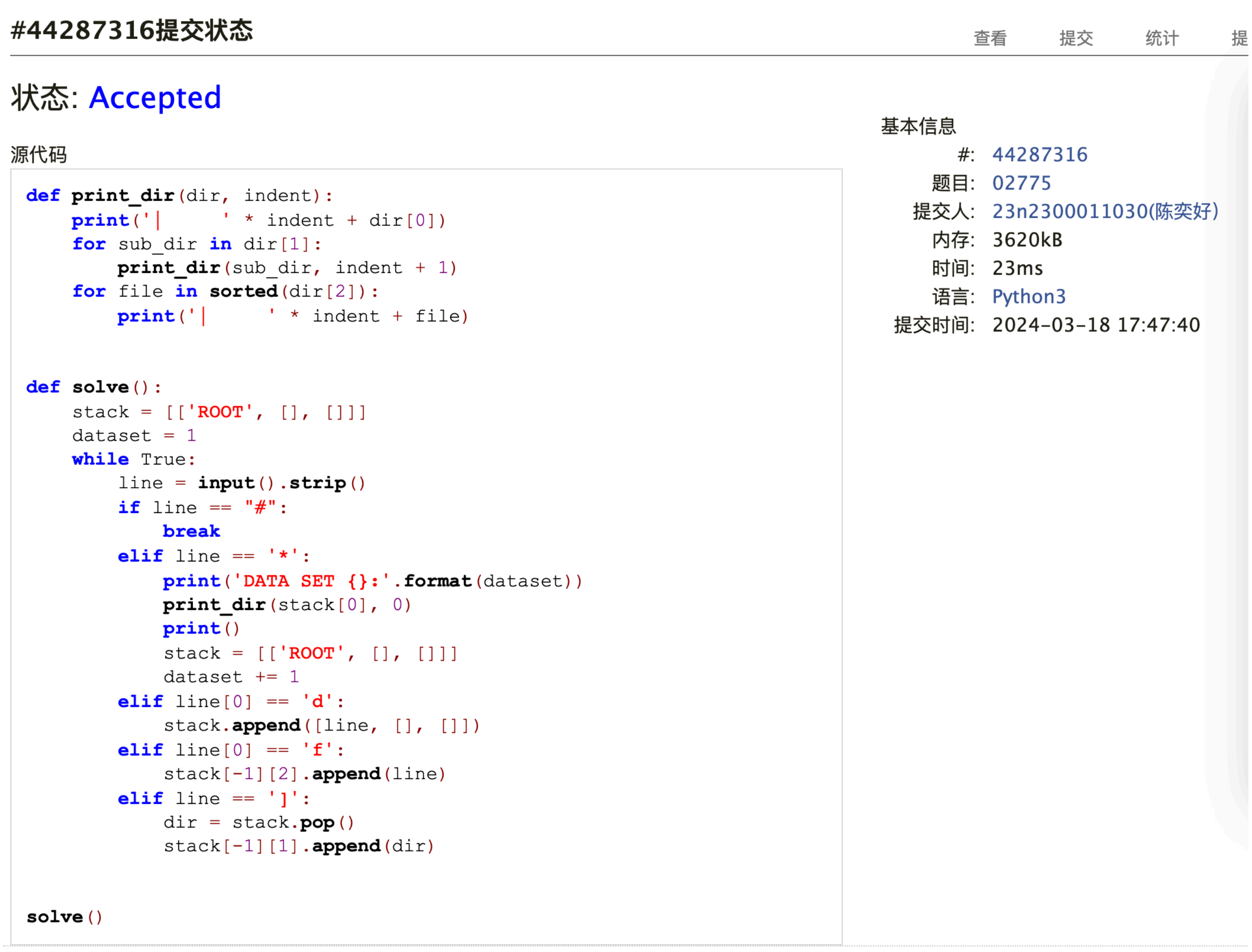
思路：这里用的是列表模拟树，其实可以在left放上文件夹，right放文件的。重点在文件夹的优先级高于文件，因此用了递归来输出答案

代码

```python
def print_dir(dir, indent):
    print('|     ' * indent + dir[0])
    for sub_dir in dir[1]:
        print_dir(sub_dir, indent + 1)
    for file in sorted(dir[2]):
        print('|     ' * indent + file)
```

```python
def solve():
    stack = [['ROOT', [], []]]
    dataset = 1
    while True:
        line = input().strip()
        if line == "#":
            break
        elif line == '*':
            print('DATA SET {}:'.format(dataset))
            print_dir(stack[0], 0)
            print()
            stack = [['ROOT', [], []]]
            dataset += 1
        elif line[0] == 'd':
            stack.append([line, [], []])
        elif line[0] == 'f':
            stack[-1][2].append(line)
        elif line == ']':
            dir = stack.pop()
            stack[-1][1].append(dir)


solve()
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==


#44287316提交状态

查看 提交 统计 提

状态: Accepted

源代码

```python
def print_dir(dir, indent):
    print('|     ' * indent + dir[0])
    for sub_dir in dir[1]:
        print_dir(sub_dir, indent + 1)
    for file in sorted(dir[2]):
        print('|     ' * indent + file)


def solve():
    stack = [['ROOT', [], []]]
    dataset = 1
    while True:
        line = input().strip()
        if line == "#":
            break
        elif line == '*':
            print('DATA SET {}:'.format(dataset))
            print_dir(stack[0], 0)
            print()
            stack = [['ROOT', [], []]]
            dataset += 1
        elif line[0] == 'd':
            stack.append([line, [], []])
        elif line[0] == 'f':
            stack[-1][2].append(line)
        elif line == ']':
            dir = stack.pop()
            stack[-1][1].append(dir)


solve()
```

基本信息

#:	44287316
题目:	02775
提交人:	23n2300011030(陈奕好)
内存:	3620kB
时间:	23ms
语言:	Python3
提交时间:	2024-03-18 17:47:40

25140: 根据后序表达式建立队列表达式

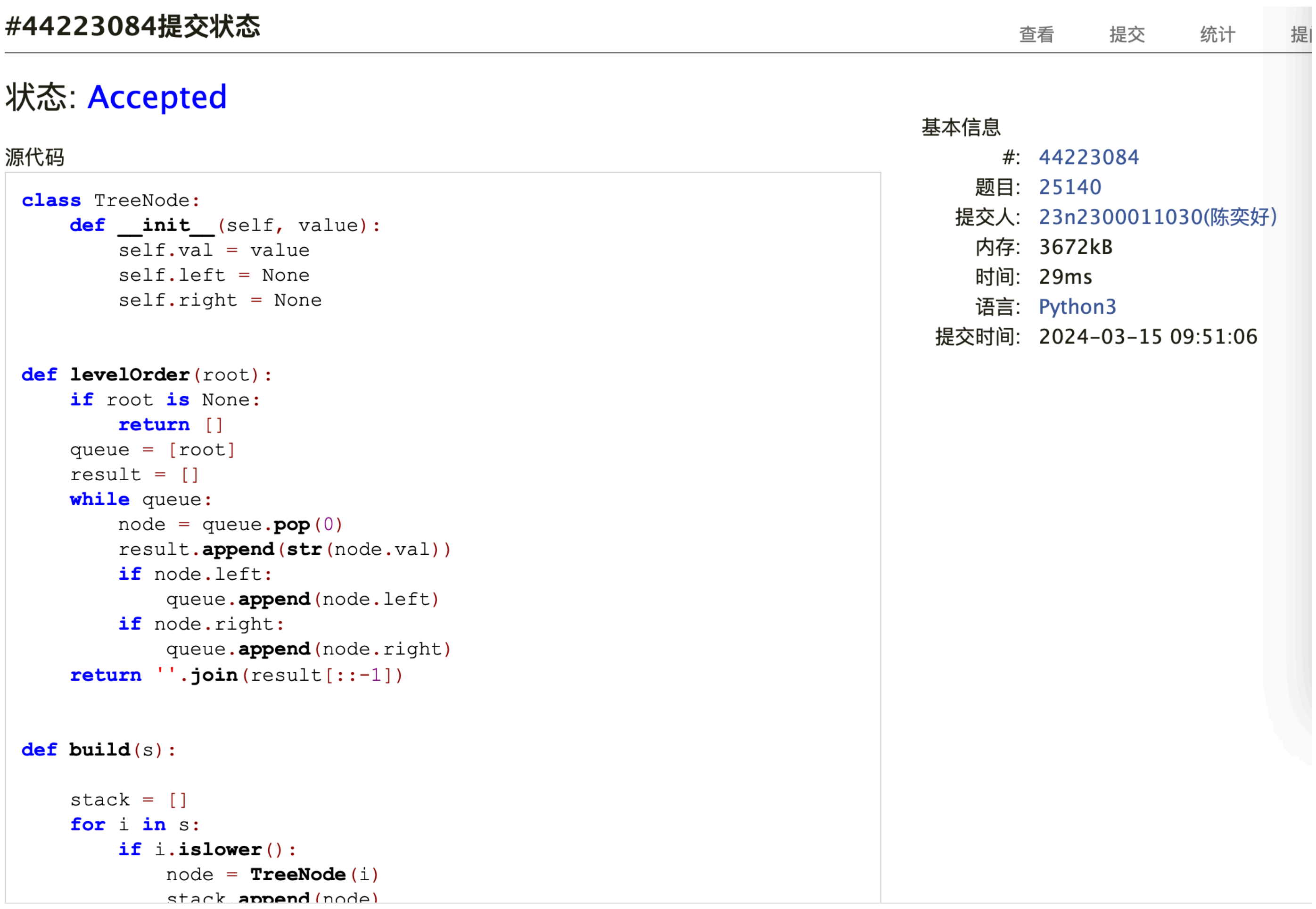
http://cs101.openjudge.cn/practice/25140/

思路：第一次想了很久，总以为是树结构有问题，其实只是输出表达错了。

代码

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


def levelOrder(root):
    if root is None:
        return []
    queue = [root]
    result = []
    while queue:
        node = queue.pop(0)
        result.append(str(node.val))
        if node.left:
            queue.append(node.left)
        if node.right:
            queue.append(node.right)
    return ''.join(result[::-1])


def build(s):

    stack = []
    for i in s:
        if i.islower():
            node = TreeNode(i)
            stack.append(node)
        else:
            node = TreeNode(i)
            node.right = stack.pop()
            node.left = stack.pop()
            stack.append(node)
    return stack[0]


for i in range(int(input())):
    s = input()
    root = build(s)
    ans = ""
    for _ in levelOrder(root):
        ans += ("".join(_))
    print(ans)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



#44223084提交状态

查看 提交 统计 提问

状态: Accepted

源代码

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


def levelOrder(root):
    if root is None:
        return []
    queue = [root]
    result = []
    while queue:
        node = queue.pop(0)
        result.append(str(node.val))
        if node.left:
            queue.append(node.left)
        if node.right:
            queue.append(node.right)
    return ''.join(result[::-1])


def build(s):

    stack = []
    for i in s:
        if i.islower():
            node = TreeNode(i)
            stack.append(node)
```

基本信息

#:	44223084
题目:	25140
提交人:	23n2300011030(陈奕好)
内存:	3672kB
时间:	29ms
语言:	Python3
提交时间:	2024-03-15 09:51:06

24750: 根据二叉树中后序序列建树

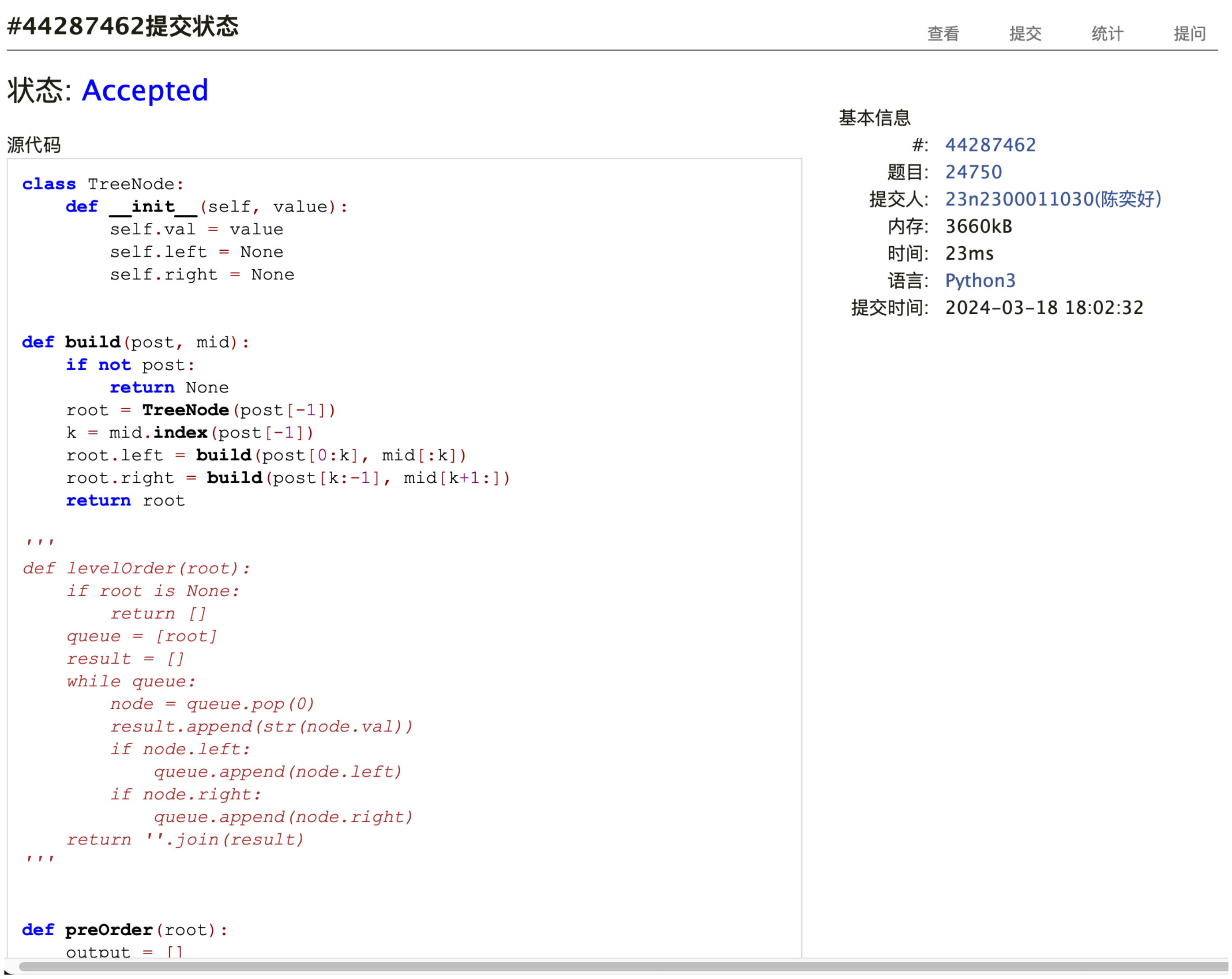
http://cs101.openjudge.cn/practice/24750/

思路：先找节点，在mid找index，递归建树

代码

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


def build(post, mid):
    if not post:
        return None
    root = TreeNode(post[-1])
    k = mid.index(post[-1])
    root.left = build(post[0:k], mid[:k])
    root.right = build(post[k:-1], mid[k+1:])
    return root

'''
def levelOrder(root):
    if root is None:
        return []
    queue = [root]
    result = []
    while queue:
        node = queue.pop(0)
        result.append(str(node.val))
        if node.left:
            queue.append(node.left)
        if node.right:
            queue.append(node.right)
    return ''.join(result)
'''


def preOrder(root):
    output = []
    if root:
        output.append(root.val)
        output.extend(preOrder(root.left))
        output.extend(preOrder(root.right))
    return output



mid = input()
post = input()
root = build(post, mid)
print(''.join(preOrder(root)))


# 2024.03.15 有点难度，需要多练习
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



状态: Accepted

源代码

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


def build(post, mid):
    if not post:
        return None
    root = TreeNode(post[-1])
    k = mid.index(post[-1])
    root.left = build(post[0:k], mid[:k])
    root.right = build(post[k:-1], mid[k+1:])
    return root

'''
def levelOrder(root):
    if root is None:
        return []
    queue = [root]
    result = []
    while queue:
        node = queue.pop(0)
        result.append(str(node.val))
        if node.left:
            queue.append(node.left)
        if node.right:
            queue.append(node.right)
    return ''.join(result)
'''


def preOrder(root):
    output = []
```

基本信息

#:	44287462
题目:	24750
提交人:	23n2300011030(陈奕好)
内存:	3660kB
时间:	23ms
语言:	Python3
提交时间:	2024-03-18 18:02:32

22158: 根据二叉树前中序序列建树

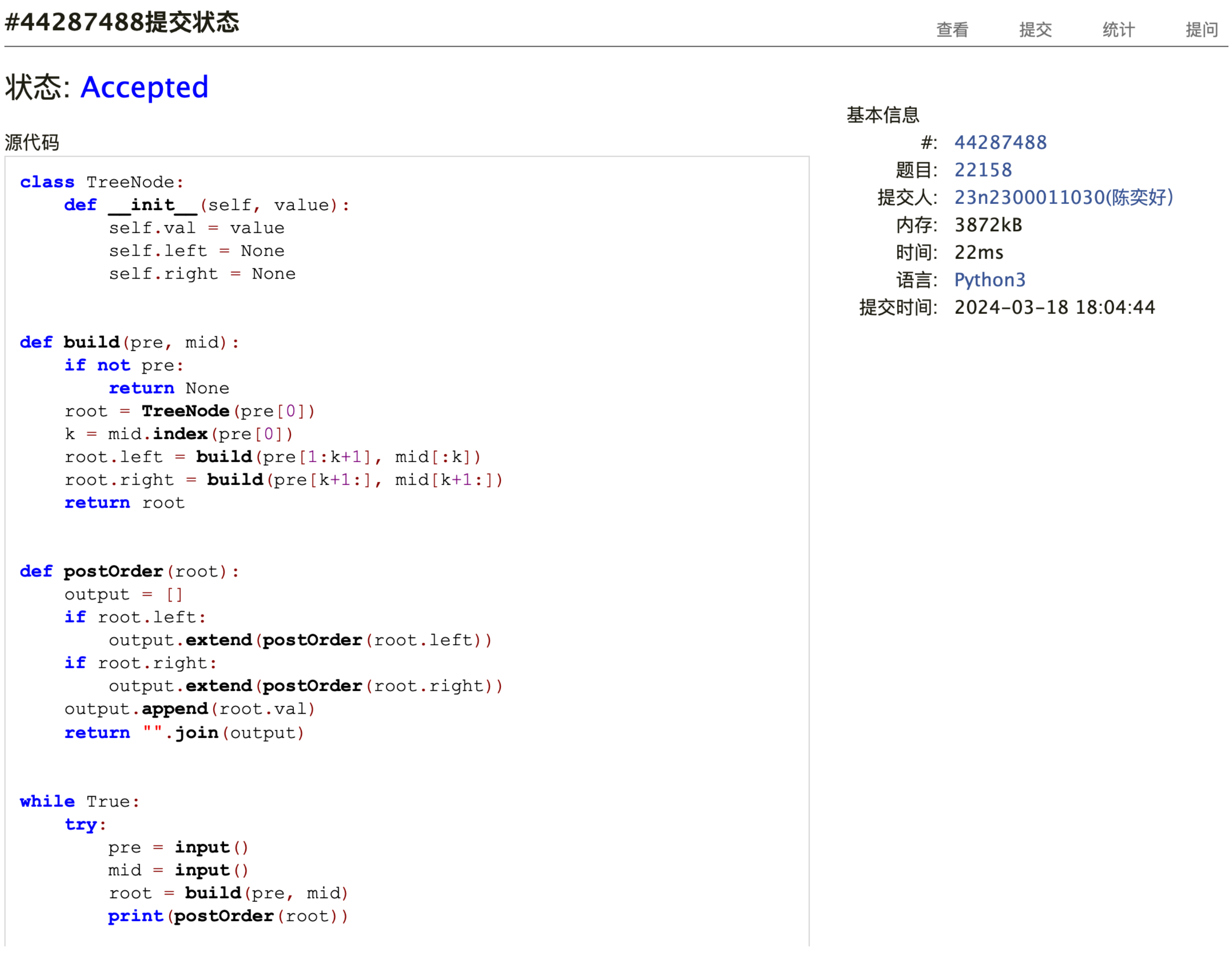
http://cs101.openjudge.cn/practice/22158/

思路：跟上面一题差一个参数

代码

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


def build(pre, mid):
    if not pre:
        return None
    root = TreeNode(pre[0])
    k = mid.index(pre[0])
    root.left = build(pre[1:k+1], mid[:k])
    root.right = build(pre[k+1:], mid[k+1:])
    return root


def postOrder(root):
    output = []
    if root.left:
        output.extend(postOrder(root.left))
    if root.right:
        output.extend(postOrder(root.right))
    output.append(root.val)
    return "".join(output)


while True:
    try:
        pre = input()
        mid = input()
        root = build(pre, mid)
        print(postOrder(root))

    except EOFError:
        break

# 2024.03.15 有点难度，需要多练习
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==


#44287488提交状态

状态: Accepted

源代码

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


def build(pre, mid):
    if not pre:
        return None
    root = TreeNode(pre[0])
    k = mid.index(pre[0])
    root.left = build(pre[1:k+1], mid[:k])
    root.right = build(pre[k+1:], mid[k+1:])
    return root


def postOrder(root):
    output = []
    if root.left:
        output.extend(postOrder(root.left))
    if root.right:
        output.extend(postOrder(root.right))
    output.append(root.val)
    return "".join(output)


while True:
    try:
        pre = input()
        mid = input()
        root = build(pre, mid)
        print(postOrder(root))
```

基本信息

#:	44287488
题目:	22158
提交人:	23n2300011030(陈奕好)
内存:	3872kB
时间:	22ms
语言:	Python3
提交时间:	2024-03-18 18:04:44

2. 学习总结和收获

==如果作业题目简单，有否额外练习题目，比如：OJ“2024spring每日选做”、CF、LeetCode、洛谷等网站题目。==

为什么都是每日选做的题，我觉得每日选做应该是附加题，用来练手的；

现在每天写完每日选做，连作业都直接CV了。