Assignment #5: "树"算：概念、表示、解析、遍历

Updated 2124 GMT+8 March 17, 2024

2024 spring, Complied by ==祁轩宇、经济学院==

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：Windows 11, version 23H2

Python编程环境：VSCode 1.87.1

C/C++编程环境：

1. 题目

27638: 求二叉树的高度和叶子数目

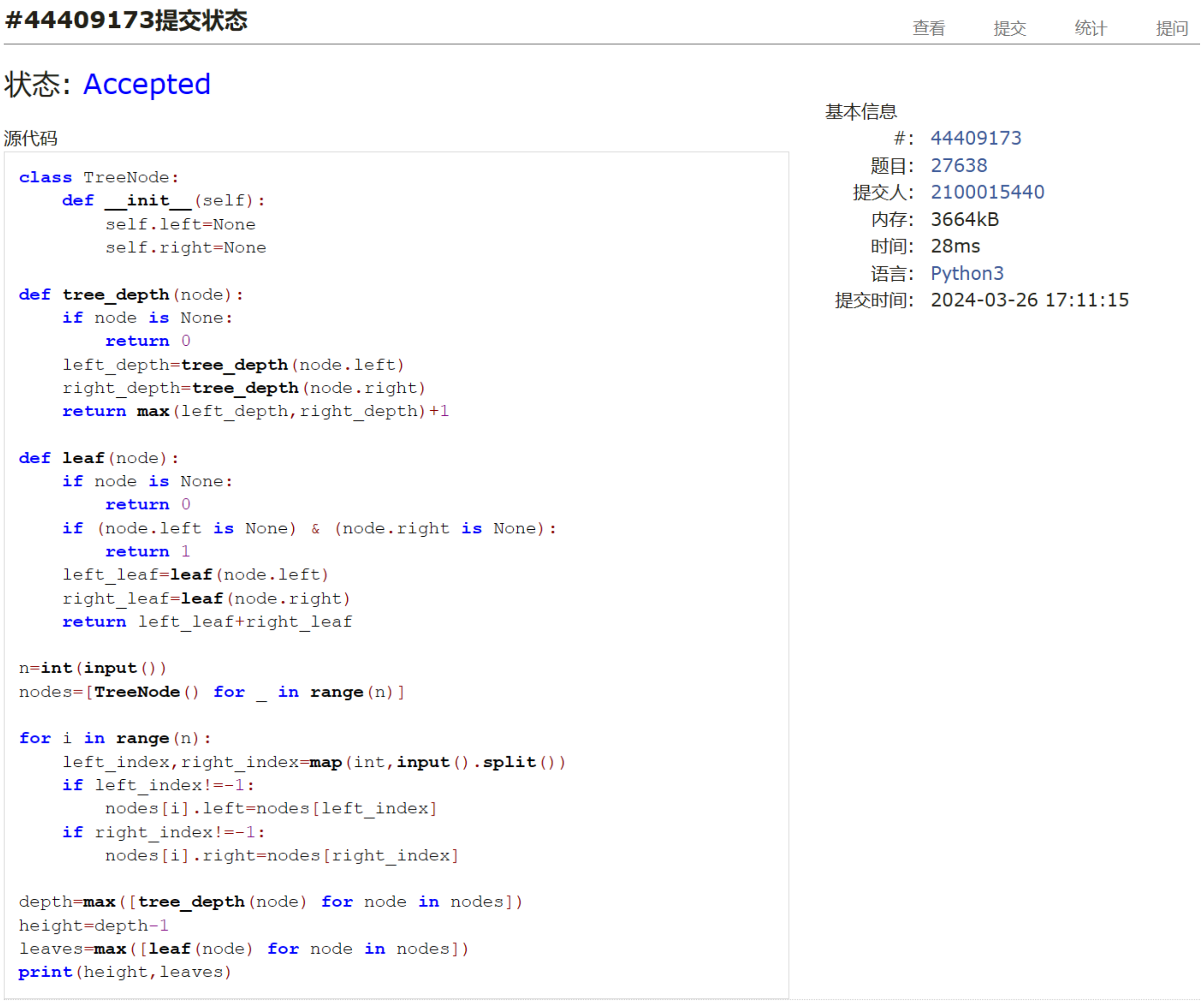
http://cs101.openjudge.cn/practice/27638/

思路：

代码

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

def tree_depth(node):
    if node is None:
        return 0
    left_depth=tree_depth(node.left)
    right_depth=tree_depth(node.right)
    return max(left_depth,right_depth)+1

def leaf(node):
    if node is None:
        return 0
    if (node.left is None) & (node.right is None):
        return 1
    left_leaf=leaf(node.left)
    right_leaf=leaf(node.right)
    return left_leaf+right_leaf

n=int(input())
nodes=[TreeNode() for _ in range(n)]

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

depth=max([tree_depth(node) for node in nodes])
height=depth-1
leaves=max([leaf(node) for node in nodes])
print(height,leaves)
```

代码运行截图 ==（至少包含有"Accepted"）==



#44409173提交状态

查看 提交 统计 提问

状态: Accepted

源代码

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

def tree_depth(node):
    if node is None:
        return 0
    left_depth=tree_depth(node.left)
    right_depth=tree_depth(node.right)
    return max(left_depth,right_depth)+1

def leaf(node):
    if node is None:
        return 0
    if (node.left is None) & (node.right is None):
        return 1
    left_leaf=leaf(node.left)
    right_leaf=leaf(node.right)
    return left_leaf+right_leaf

n=int(input())
nodes=[TreeNode() for _ in range(n)]

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

depth=max([tree_depth(node) for node in nodes])
height=depth-1
leaves=max([leaf(node) for node in nodes])
print(height,leaves)
```

基本信息

#:	44409173
题目:	27638
提交人:	2100015440
内存:	3664kB
时间:	28ms
语言:	Python3
提交时间:	2024-03-26 17:11:15

24729: 括号嵌套树

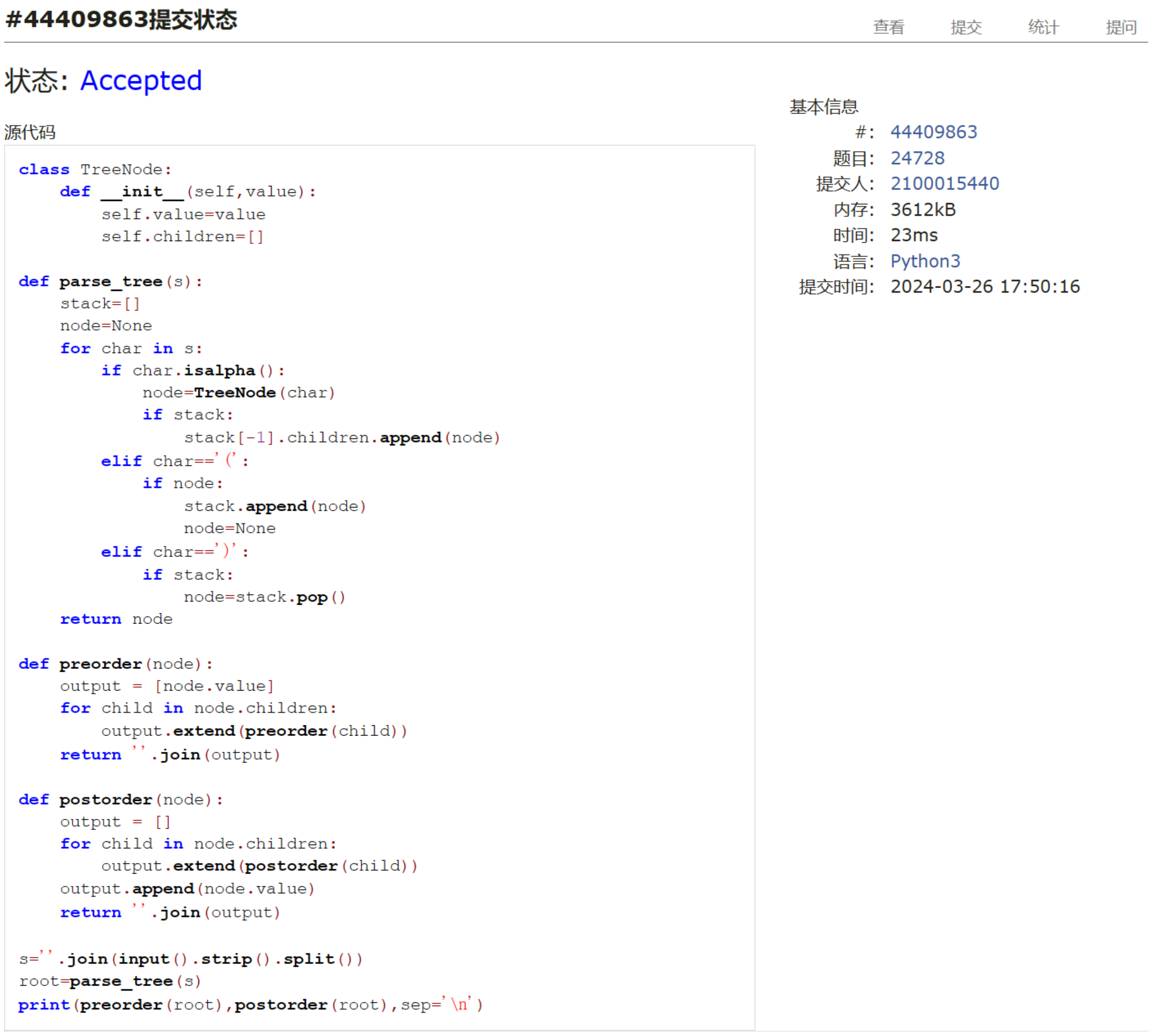
http://cs101.openjudge.cn/practice/24729/

思路：

代码

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

def parse_tree(s):
    stack=[]
    node=None
    for char in s:
        if char.isalpha():
            node=TreeNode(char)
            if stack:
                stack[-1].children.append(node)
        elif char=='(':
            if node:
                stack.append(node)
                node=None
        elif char==')':
            if stack:
                node=stack.pop()
    return node

def preorder(node):
    output = [node.value]
    for child in node.children:
        output.extend(preorder(child))
    return ''.join(output)

def postorder(node):
    output = []
    for child in node.children:
        output.extend(postorder(child))
    output.append(node.value)
    return ''.join(output)

s=''.join(input().strip().split())
root=parse_tree(s)
print(preorder(root),postorder(root),sep='\n')
```

代码运行截图 ==（至少包含有"Accepted"）==



#44409863提交状态

查看 提交 统计 提问

状态: Accepted

源代码

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

def parse_tree(s):
    stack=[]
    node=None
    for char in s:
        if char.isalpha():
            node=TreeNode(char)
            if stack:
                stack[-1].children.append(node)
        elif char=='(':
            if node:
                stack.append(node)
                node=None
        elif char==')':
            if stack:
                node=stack.pop()
    return node

def preorder(node):
    output = [node.value]
    for child in node.children:
        output.extend(preorder(child))
    return ''.join(output)

def postorder(node):
    output = []
    for child in node.children:
        output.extend(postorder(child))
    output.append(node.value)
    return ''.join(output)

s=''.join(input().strip().split())
root=parse_tree(s)
print(preorder(root),postorder(root),sep='\n')
```

基本信息

#:	44409863
题目:	24728
提交人:	2100015440
内存:	3612kB
时间:	23ms
语言:	Python3
提交时间:	2024-03-26 17:50:16

02775: 文件结构“图”

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

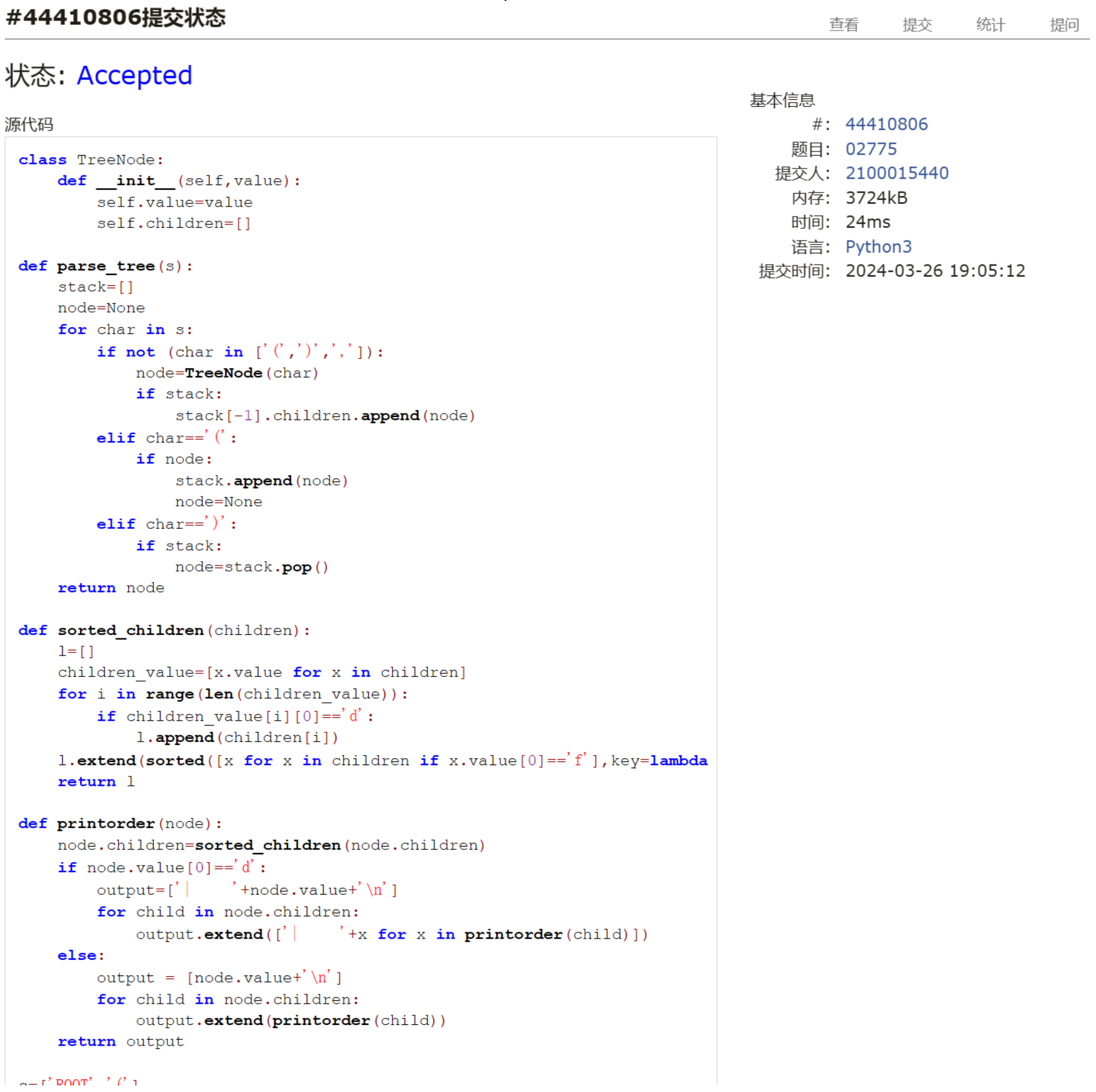
思路：

我做这个题的时候套了括号嵌套树的模板，不过还是题解的方法更简便。

代码

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

def parse_tree(s):
    stack=[]
    node=None
    for char in s:
        if not (char in ['(',')',',']):
            node=TreeNode(char)
            if stack:
                stack[-1].children.append(node)
        elif char=='(':
            if node:
                stack.append(node)
                node=None
        elif char==')':
            if stack:
                node=stack.pop()
    return node

def sorted_children(children):
    l=[]
    children_value=[x.value for x in children]
    for i in range(len(children_value)):
        if children_value[i][0]=='d':
            l.append(children[i])
    l.extend(sorted([x for x in children if x.value[0]=='f'],key=lambda
x:x.value))
    return l

def printorder(node):
    node.children=sorted_children(node.children)
    if node.value[0]=='d':
        output=['|     '+node.value+'\n']
        for child in node.children:
            output.extend(['|     '+x for x in printorder(child)])
    else:
        output = [node.value+'\n']
        for child in node.children:
            output.extend(printorder(child))
    return output

s=['ROOT','(']
i=1
x=''
while x!='#':
    x=input()
    if x=='*':
        s.append(')')
        root=parse_tree(s)
        print(f'DATA SET {i}:')
        print(''.join(printorder(root)))
        i+=1
        s=['ROOT','(']
    elif x[0]=='f':
        s.extend([x,','])
    elif x[0]=='d':
        s.extend([x,'('])
    elif x==']':
        s.append(')')
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



#44410806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.value=value
        self.children=[]

def parse_tree(s):
    stack=[]
    node=None
    for char in s:
        if not (char in ['(',')',',']):
            node=TreeNode(char)
            if stack:
                stack[-1].children.append(node)
        elif char=='(':
            if node:
                stack.append(node)
                node=None
        elif char==')':
            if stack:
                node=stack.pop()
    return node

def sorted_children(children):
    l=[]
    children_value=[x.value for x in children]
    for i in range(len(children_value)):
        if children_value[i][0]=='d':
            l.append(children[i])
    l.extend(sorted([x for x in children if x.value[0]=='f'],key=lambda
    return l

def printorder(node):
    node.children=sorted_children(node.children)
    if node.value[0]=='d':
        output=['|     '+node.value+'\n']
        for child in node.children:
            output.extend(['|     '+x for x in printorder(child)])
    else:
        output = [node.value+'\n']
        for child in node.children:
            output.extend(printorder(child))
    return output

s=['ROOT', '(']
```

基本信息

#:	44410806
题目:	02775
提交人:	2100015440
内存:	3724kB
时间:	24ms
语言:	Python3
提交时间:	2024-03-26 19:05:12

25140: 根据后序表达式建立队列表达式

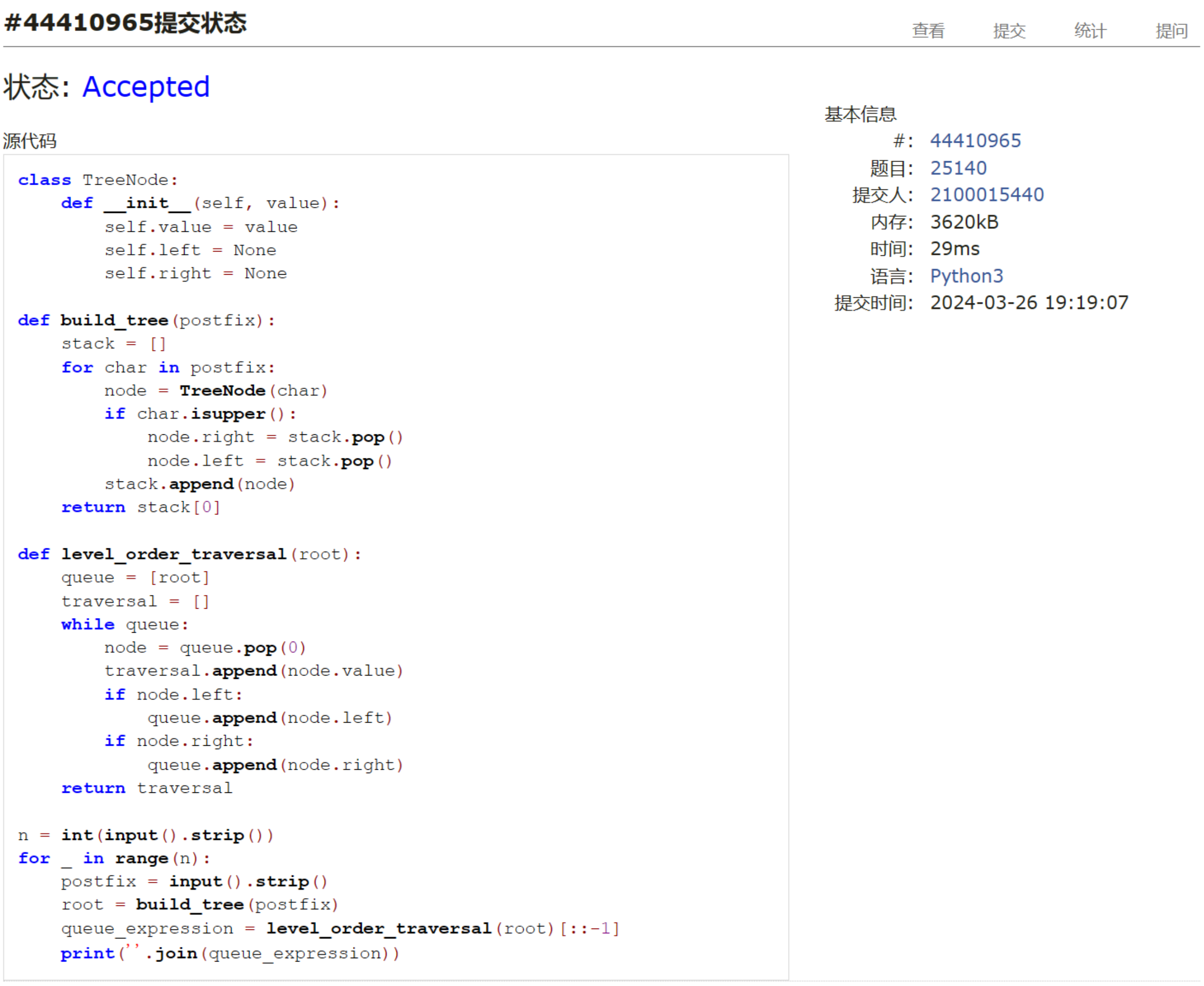
http://cs101.openjudge.cn/practice/25140/

思路：

代码

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

def build_tree(postfix):
    stack = []
    for char in postfix:
        node = TreeNode(char)
        if char.isupper():
            node.right = stack.pop()
            node.left = stack.pop()
        stack.append(node)
    return stack[0]

def level_order_traversal(root):
    queue = [root]
    traversal = []
    while queue:
        node = queue.pop(0)
        traversal.append(node.value)
        if node.left:
            queue.append(node.left)
        if node.right:
            queue.append(node.right)
    return traversal

n = int(input().strip())
for _ in range(n):
    postfix = input().strip()
    root = build_tree(postfix)
    queue_expression = level_order_traversal(root)[::-1]
    print(''.join(queue_expression))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



#44410965提交状态

查看 提交 统计 提问

状态: Accepted

源代码

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

def build_tree(postfix):
    stack = []
    for char in postfix:
        node = TreeNode(char)
        if char.isupper():
            node.right = stack.pop()
            node.left = stack.pop()
        stack.append(node)
    return stack[0]

def level_order_traversal(root):
    queue = [root]
    traversal = []
    while queue:
        node = queue.pop(0)
        traversal.append(node.value)
        if node.left:
            queue.append(node.left)
        if node.right:
            queue.append(node.right)
    return traversal

n = int(input().strip())
for _ in range(n):
    postfix = input().strip()
    root = build_tree(postfix)
    queue_expression = level_order_traversal(root)[::-1]
    print(''.join(queue_expression))
```

基本信息

#:	44410965
题目:	25140
提交人:	2100015440
内存:	3620kB
时间:	29ms
语言:	Python3
提交时间:	2024-03-26 19:19:07

24750: 根据二叉树中后序序列建树

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

思路：

代码

```
# 
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

22158: 根据二叉树前中序序列建树

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

思路：

代码

```
```

```
#
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

2. 学习总结和收获

- 题目感觉难度上比上一次低一点，不过写起来比较繁琐。感觉还需要加强类的书写，有些不熟练。
- 做题debug的时候使用了Python Tutor，对树节点之间的引用关系有了更清楚的认识。
- 一部分题目还没有搞懂，之后结合题解再自己打一遍。