Assignment #5: "树"算：概念、表示、解析、遍历

Updated 2124 GMT+8 March 17, 2024

2024 spring, Complied by 王申睿——物理学院

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 typorahttps://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 Ventura 13.4.1 (c)

Python编程环境： Spyder IDE 5.2.2, PyCharm 2023.1.4 (Professional Edition)

C/C++编程环境： Mac terminal vi (version 9.0.1424), g++/gcc (Apple clang version 14.0.3, clang-1403.0.22.14.1)

1. 题目

27638: 求二叉树的高度和叶子数目

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

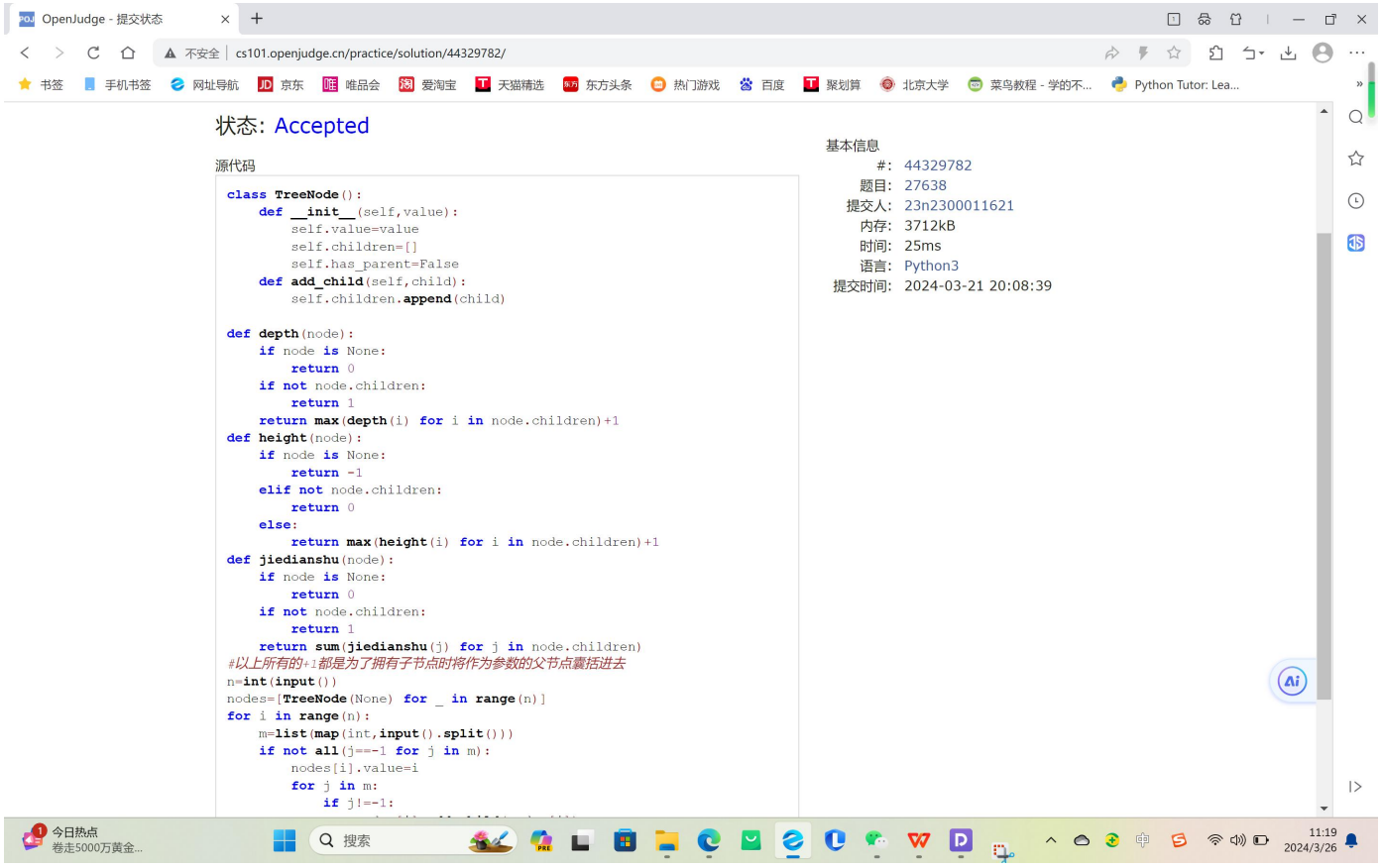
思路:

代码

```
class TreeNode():
    def __init__(self,value):
        self.value=value
        self.children=[]
        self.has_parent=False
    def add_child(self,child):
        self.children.append(child)



def depth(node):
    if node is None:
        return 0
    if not node.children:
        return 1
    return max(depth(i) for i in node.children)+1
def height(node):
    if node is None:
        return -1
    elif not node.children:
```

```
        return 0
    else:
        return max(height(i) for i in node.children)+1
def jiedianshu(node):
    if node is None:
        return 0
    if not node.children:
        return 1
    return sum(jiedianshu(j) for j in node.children)
#以上所有的+1都是为了拥有子节点时将作为参数的父节点囊括进去
n=int(input())
nodes=[TreeNode(None) for _ in range(n)]
for i in range(n):
    m=list(map(int,input().split()))
    if not all(j==-1 for j in m):
        nodes[i].value=i
        for j in m:
            if j!=-1:
```

```
            nodes[i].add_child(nodes[j])

            nodes[j].has_parent=True

root=None

for i in range(n):

    if nodes[i].has_parent==False:

        root=nodes[i]

print(str(height(root))+" "+str(jiedianshu(root)))
```



24729: 括号嵌套树

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

思路:

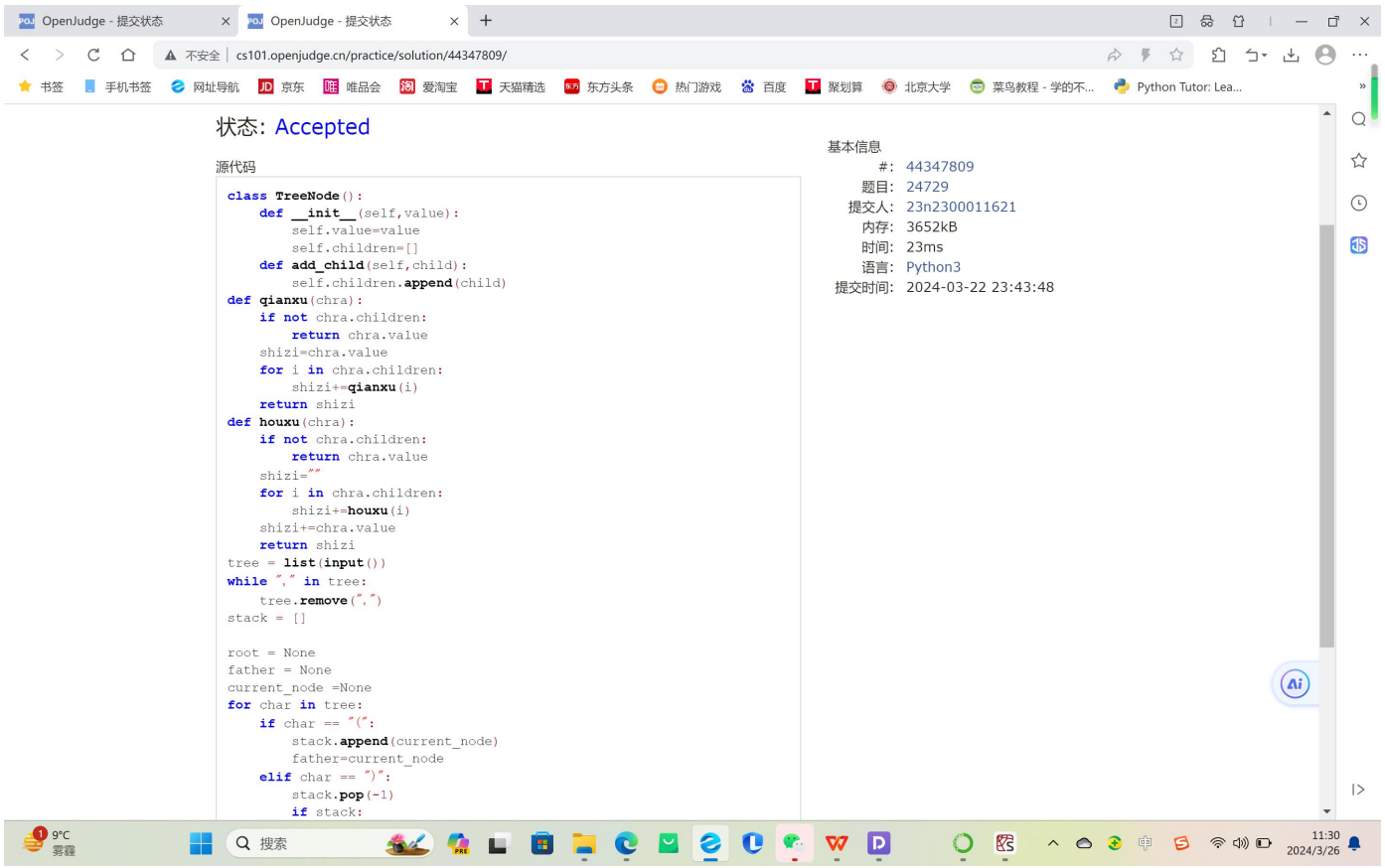
代码

```
class TreeNode():

    def __init__(self,value):

        self.value=value

        self.children=[]

    def add_child(self,child):

        self.children.append(child)

def qianxu(chra):

    if not chra.children:

        return chra.value

    shizi=chra.value

    for i in chra.children:

        shizi+=qianxu(i)
```

```
    return shizi

def houxu(chra):

    if not chra.children:

        return chra.value

    shizi=""

    for i in chra.children:

        shizi+=houxu(i)

    shizi+=chra.value

    return shizi

tree = list(input())

while "," in tree:

    tree.remove(",")

stack = []
```

```
root = None

father = None

current_node =None

for char in tree:

    if char == "(":

        stack.append(current_node)

        father=current_node

    elif char == ")":

        stack.pop(-1)

        if stack:

            father=stack[-1]

    else:

        new_node = TreeNode(char)

        if root is None:
```

```
            root = new_node

            current_node = root



        else:

            father.add_child(new_node)

            current_node = new_node

print(qianxu(root))

print(houxu(root))
```



02775: 文件结构“图”

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

思路:

代码

```
#

```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

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

```
#

```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

24750: 根据二叉树中后序序列建树

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

思路:

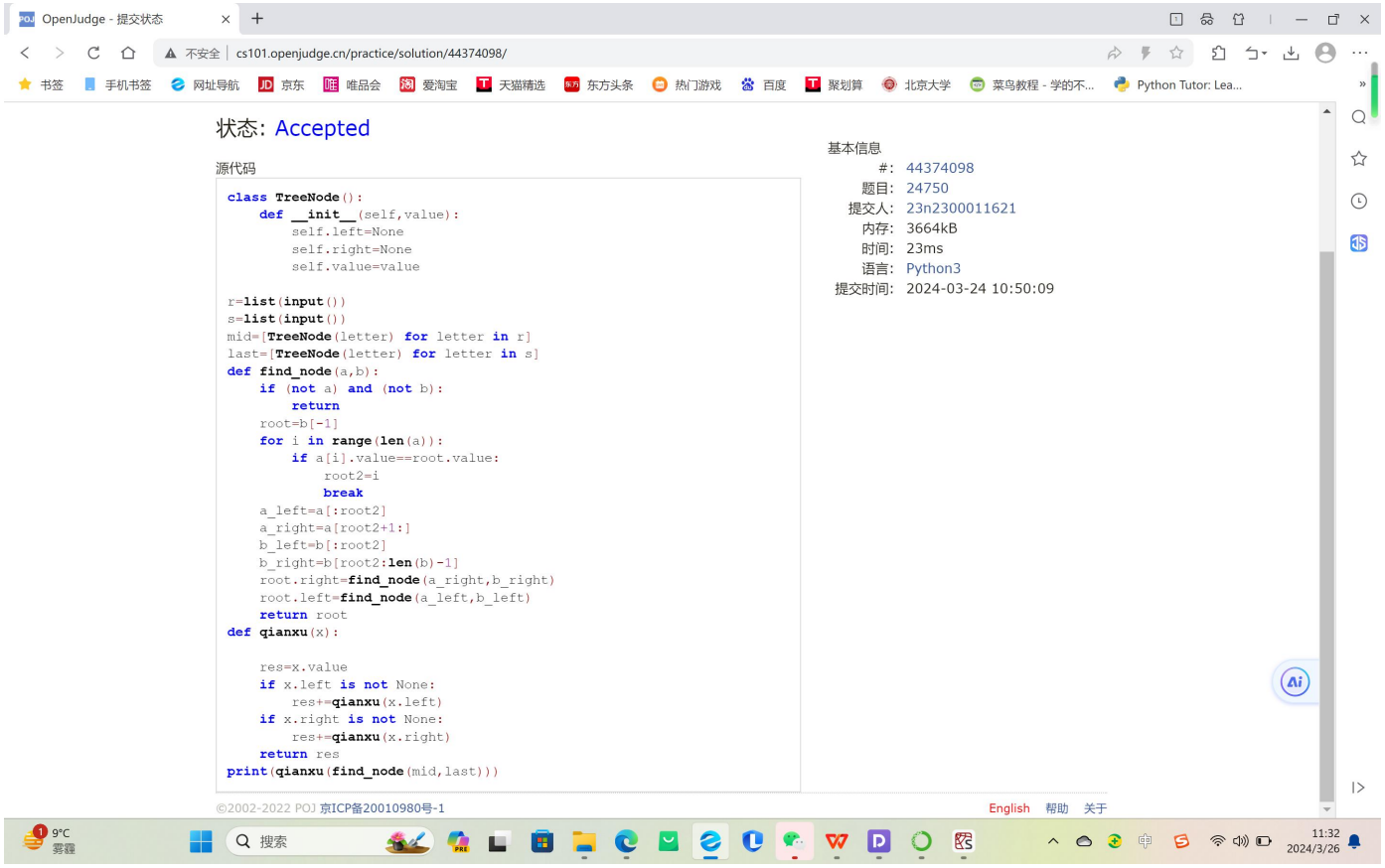
代码

```
class TreeNode():

    def __init__(self,value):

        self.left=None

        self.right=None

        self.value=value
```

```
r=list(input())

s=list(input())

mid=[TreeNode(letter) for letter in r]

last=[TreeNode(letter) for letter in s]

def find_node(a,b):

    if (not a) and (not b):

        return

    root=b[-1]

    for i in range(len(a)):

        if a[i].value==root.value:

            root2=i

            break

    a_left=a[:root2]
```

```
    a_right=a[root2+1:]

    b_left=b[:root2]

    b_right=b[root2:len(b)-1]

    root.right=find_node(a_right,b_right)

    root.left=find_node(a_left,b_left)

    return root

def qianxu(x):



    res=x.value

    if x.left is not None:

        res+=qianxu(x.left)

    if x.right is not None:

        res+=qianxu(x.right)

    return res
```

print(qianxu(find_node(mid,last)))



22158: 根据二叉树前中序序列建树

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

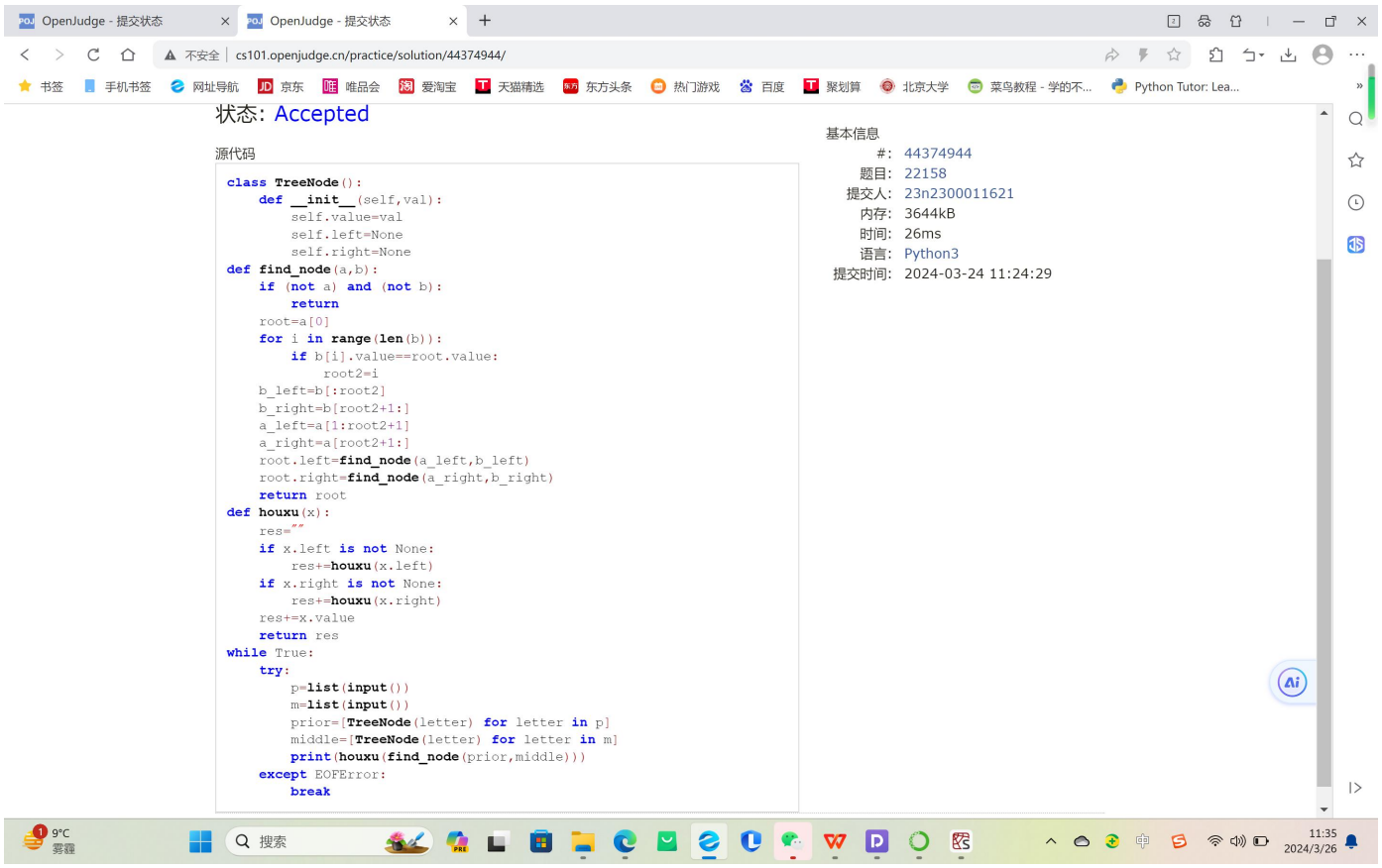
思路:

代码

```
class TreeNode():

    def __init__(self,val):

        self.value=val

        self.left=None

        self.right=None

def find_node(a,b):

    if (not a) and (not b):

        return

    root=a[0]

    for i in range(len(b)):

        if b[i].value==root.value:

            root2=i

    b_left=b[:root2]
```

```
    b_right=b[root2+1:]

    a_left=a[1:root2+1]

    a_right=a[root2+1:]

    root.left=find_node(a_left,b_left)

    root.right=find_node(a_right,b_right)

    return root

def houxu(x):

    res=""

    if x.left is not None:

        res+=houxu(x.left)

    if x.right is not None:

        res+=houxu(x.right)

    res+=x.value

    return res
```

```
while True:
    try:
        p=list(input())
        m=list(input())
        prior=[TreeNode(letter) for letter in p]
        middle=[TreeNode(letter) for letter in m]
        print(houxu(find_node(prior,middle)))
    except EOFError:
        break
```



2. 学习总结和收获

因原文件丢失，本周总结从简：

数算的精华在于复用，但从节点数和高度一题可以得出，应注意视题目具体要求进行微调；

本周在括号嵌套树上花费了较多时间，但能感受到自己的肉眼调代码的能力有了明显提升，甚至能发现GPT的错误；

前序，后序表达式的优势在于根节点便于确定，应当充分利用；

创建树节点的过程会像字典一样覆盖，原因是每次创建该类的时候所有的属性都设置了默认值，会覆盖修改的值；

后序右节点优先，前序左节点优先；