Assignment #5: "树"算：概念、表示、解析、遍历

Updated 2124 GMT+8 March 17, 2024

2024 spring, Complied by 同学的姓名、院系

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 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：

列表表示节点，每个节点指向两个节点，筛出来根节点，高度递归

代码

```
n=int(input())
if n!=0:
    class Tree:
        def __init__(self,value):
            self.leftchild=None
            self.rightchild=None

    tree_lst=[Tree(i) for i in range(n)]
    rootjudge=[True]*n
    leaves=0

    for _ in range(n):
        lrlst=[int(x) for x in input().split()]
        l=lrlst[0]
        r=lrlst[1]
        if l!=-1:
            rootjudge[l]=False
        if r!=-1:
            rootjudge[r]=False
        if l==-1 and r==-1:
            leaves+=1
        tree_lst[_].leftchild=l
        tree_lst[_].rightchild=r
    root=0
    for i in range(n):
        if rootjudge[i]:
            root=i
            break

    def height(node):

        l=tree_lst[node].leftchild
        r=tree_lst[node].rightchild
        if l==-1 and r==-1:
            return 0
        elif l==-1 and r!=-1:
            return height(r)+1
        elif l!=-1 and r==-1:
            return height(l)+1
        else :
            return max(height(l),height(r))+1

    print(height(root),leaves)
else :
    print(-1,0)
```

代码运行截图 （至少包含有"Accepted"）

24729: 括号嵌套树

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

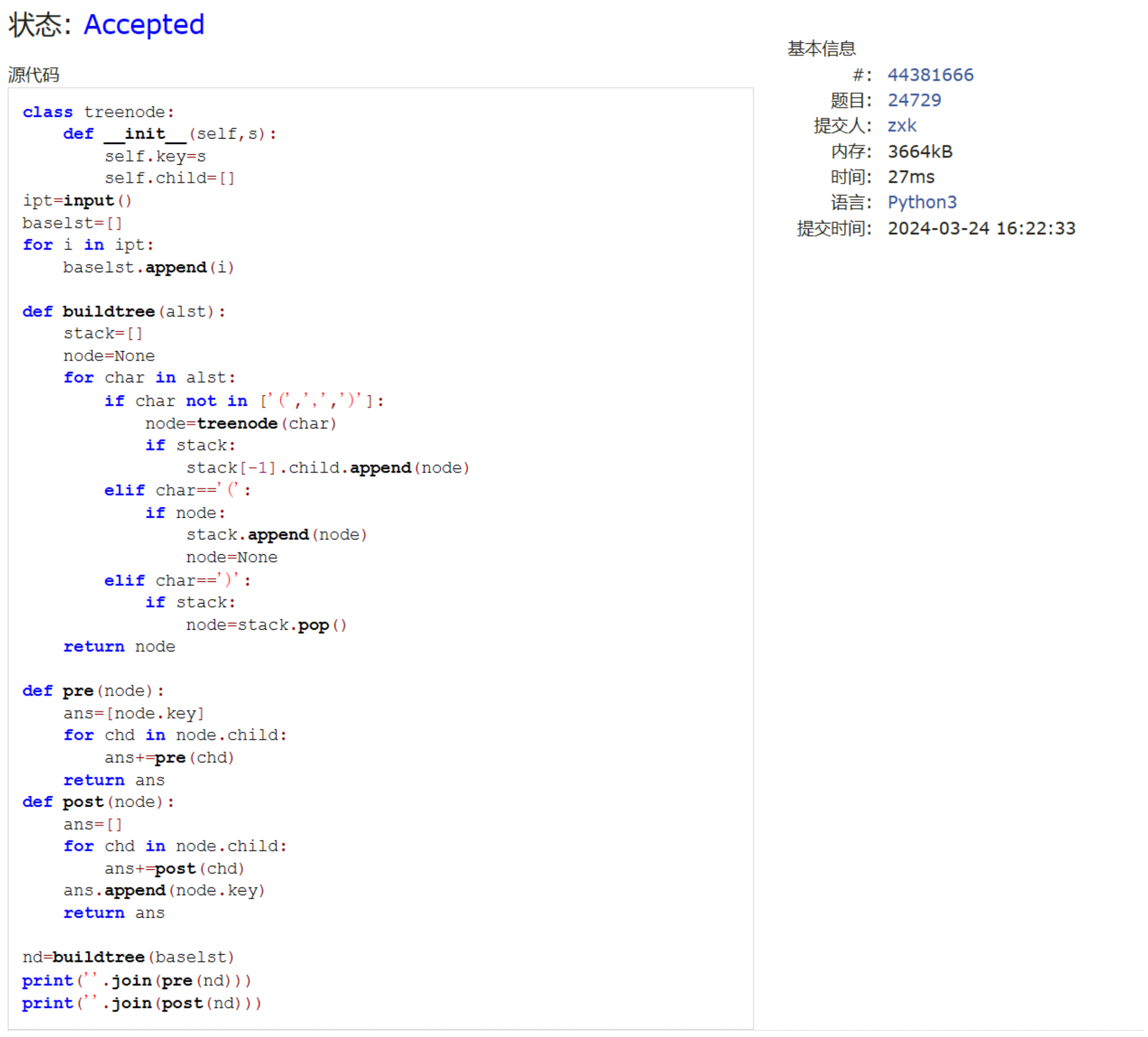
思路：

对于不确定子节点数目的树，可以采用节点+child列表的方法，构建树的过程是需要记住的，然后根据前序和后序的要求递归遍历

代码

```python
class treenode:
    def __init__(self,s):
        self.key=s
        self.child=[]
ipt=input()
baselst=[]
for i in ipt:
    baselst.append(i)

def buildtree(alst):
    stack=[]
    node=None
    for char in alst:
        if char not in ['(',',',')']:
            node=treenode(char)#构建节点
            if stack:
                stack[-1].child.append(node)#如果此时栈内有节点，就把这个节点放入最后一个节点的child中
        elif char=='(':
            if node:
                stack.append(node)#如果这时候有node，说明这个括号是这个node的child的开端，把node压入栈中，此时栈中最后一个元素是当前括号对应的node
                node=None#重新分析节点，将旧的节点重置
        elif char==')':
            if stack:
                node=stack.pop()#当前节点编辑结束，重新返回编译好的父节点
    return node

def pre(node):
    ans=[node.key]
    for chd in node.child:
        ans+=pre(chd)
    return ans
def post(node):
    ans=[]
    for chd in node.child:
        ans+=post(chd)
    ans.append(node.key)
    return ans

nd=buildtree(baselst)
print(''.join(pre(nd)))
print(''.join(post(nd)))
```

代码运行截图 （至少包含有"Accepted"）


状态: Accepted

源代码

```python
class treenode:
    def __init__(self,s):
        self.key=s
        self.child=[]
ipt=input()
baselst=[]
for i in ipt:
    baselst.append(i)

def buildtree(alst):
    stack=[]
    node=None
    for char in alst:
        if char not in ['(',',',')']:
            node=treenode(char)
            if stack:
                stack[-1].child.append(node)
        elif char=='(':
            if node:
                stack.append(node)
                node=None
        elif char==')':
            if stack:
                node=stack.pop()
    return node

def pre(node):
    ans=[node.key]
    for chd in node.child:
        ans+=pre(chd)
    return ans
def post(node):
    ans=[]
    for chd in node.child:
        ans+=post(chd)
    ans.append(node.key)
    return ans

nd=buildtree(baselst)
print(''.join(pre(nd)))
print(''.join(post(nd)))
```

基本信息

#:	44381666
题目:	24729
提交人:	zxk
内存:	3664kB
时间:	27ms
语言:	Python3
提交时间:	2024-03-24 16:22:33

02775: 文件结构“图”

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

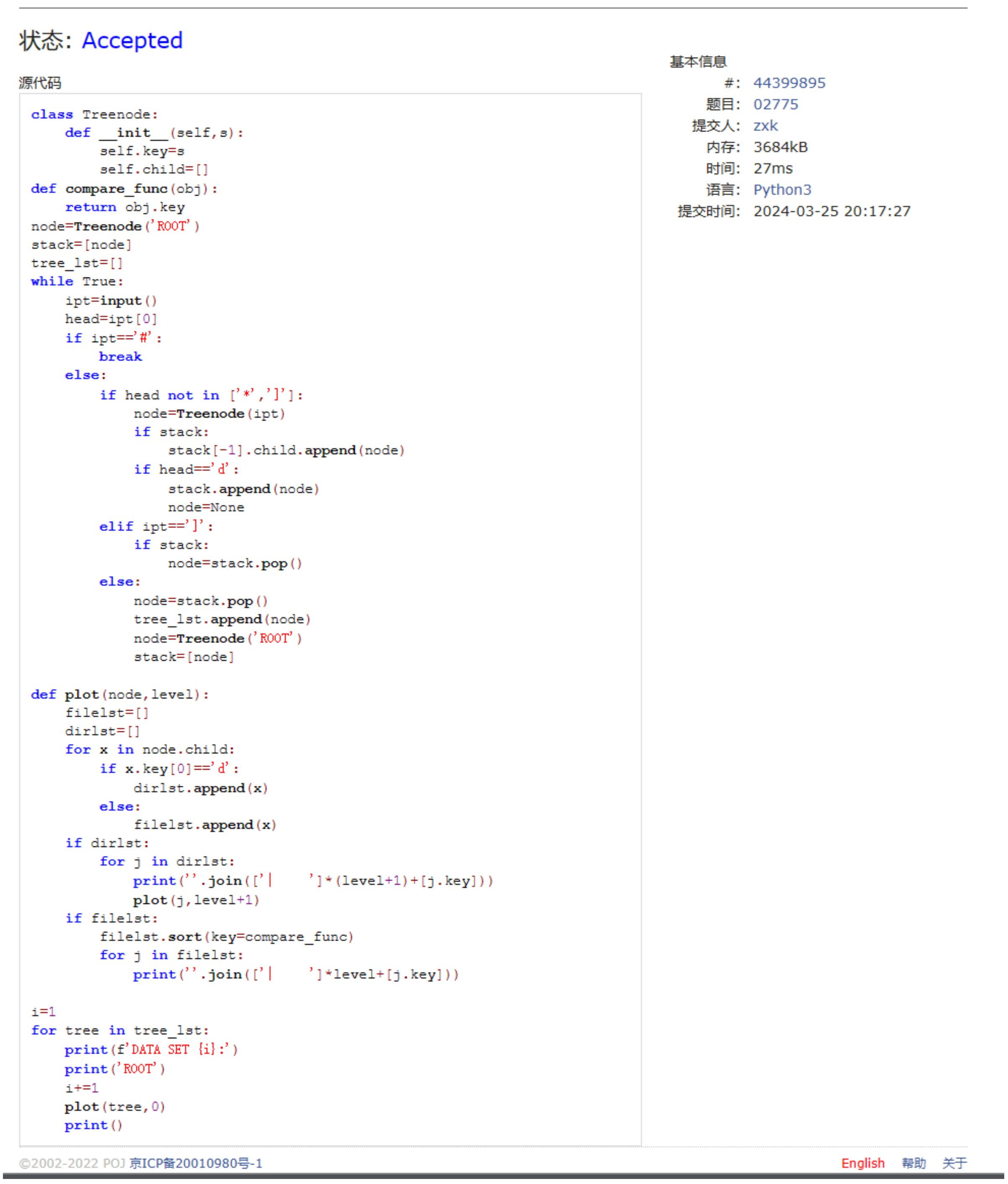
思路:

复刻括号嵌套树，改改接口导入

代码

```python
class Treenode:
    def __init__(self,s):
        self.key=s
        self.child=[]
def compare_func(obj):
    return obj.key
node=Treenode('ROOT')
stack=[node]
tree_lst=[]
while True:
    ipt=input()
    head=ipt[0]
    if ipt=='#':
        break
    else:
        if head not in ['*',']']:
            node=Treenode(ipt)
            if stack:
                stack[-1].child.append(node)
            if head=='d':
                stack.append(node)
                node=None
        elif ipt==']':
            if stack:
                node=stack.pop()
        else:
            node=stack.pop()
            tree_lst.append(node)
            node=Treenode('ROOT')
            stack=[node]

def plot(node,level):
    filelst=[]
    dirlst=[]
    for x in node.child:
        if x.key[0]=='d':
            dirlst.append(x)
        else:
            filelst.append(x)
    if dirlst:
        for j in dirlst:
            print(''.join(['|     ']*(level+1)+[j.key]))
            plot(j,level+1)
    if filelst:
        filelst.sort(key=compare_func)
        for j in filelst:
            print(''.join(['|     ']*level+[j.key]))

i=1
for tree in tree_lst:
    print(f'DATA SET {i}:')
    print('ROOT')
    i+=1
    plot(tree,0)
    print()
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）


状态: Accepted

源代码

```python
class Treenode:
    def __init__(self,s):
        self.key=s
        self.child=[]
def compare_func(obj):
    return obj.key
node=Treenode('ROOT')
stack=[node]
tree_lst=[]
while True:
    ipt=input()
    head=ipt[0]
    if ipt=='#':
        break
    else:
        if head not in ['*',']']:
            node=Treenode(ipt)
            if stack:
                stack[-1].child.append(node)
            if head=='d':
                stack.append(node)
                node=None
        elif ipt==']':
            if stack:
                node=stack.pop()
        else:
            node=stack.pop()
            tree_lst.append(node)
            node=Treenode('ROOT')
            stack=[node]

def plot(node,level):
    filelst=[]
    dirlst=[]
    for x in node.child:
        if x.key[0]=='d':
            dirlst.append(x)
        else:
            filelst.append(x)
    if dirlst:
        for j in dirlst:
            print(''.join(['|     ']*(level+1)+[j.key]))
            plot(j,level+1)
    if filelst:
        filelst.sort(key=compare_func)
        for j in filelst:
            print(''.join(['|     ']*level+[j.key]))

i=1
for tree in tree_lst:
    print(f'DATA SET {i}:')
    print('ROOT')
    i+=1
    plot(tree,0)
    print()
```

基本信息

#:	44399895
题目:	02775
提交人:	zxk
内存:	3684kB
时间:	27ms
语言:	Python3
提交时间:	2024-03-25 20:17:27

©2002-2022 POJ 京ICP备20010980号-1　　English 帮助 关于

25140: 根据后序表达式建立队列表达式

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

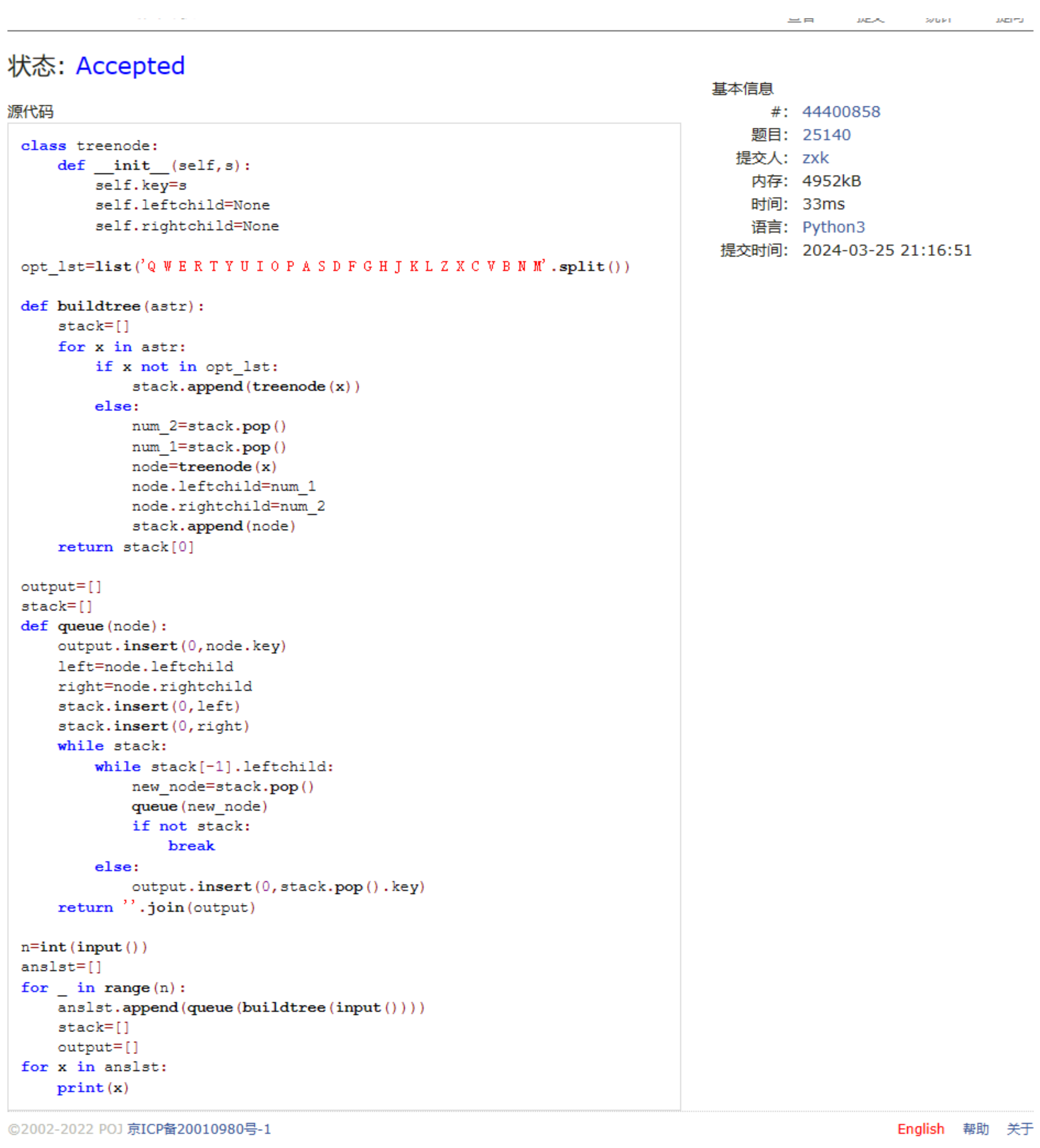
思路:

先根据后序表达式建树，然后对树按照队列表达式的方式反推出来原始的队列表达式

代码

```python
class treenode:
    def __init__(self,s):
        self.key=s
        self.leftchild=None
        self.rightchild=None

opt_lst=list('Q W E R T Y U I O P A S D F G H J K L Z X C V B N M'.split())

def buildtree(astr):
    stack=[]
    for x in astr:
        if x not in opt_lst:
            stack.append(treenode(x))
        else:
            num_2=stack.pop()
            num_1=stack.pop()
            node=treenode(x)
            node.leftchild=num_1
            node.rightchild=num_2
            stack.append(node)
    return stack[0]

output=[]
stack=[]
def queue(node):
    output.insert(0,node.key)
    left=node.leftchild
    right=node.rightchild
    stack.insert(0,left)
    stack.insert(0,right)
    while stack:
        while stack[-1].leftchild:
            new_node=stack.pop()
            queue(new_node)
            if not stack:
                break
        else:
            output.insert(0,stack.pop().key)
    return ''.join(output)

n=int(input())
anslst=[]
for _ in range(n):
    anslst.append(queue(buildtree(input())))
    stack=[]
    output=[]
for x in anslst:
    print(x)
```

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,s):
        self.key=s
        self.leftchild=None
        self.rightchild=None

opt_lst=list('Q W E R T Y U I O P A S D F G H J K L Z X C V B N M'.split())

def buildtree(astr):
    stack=[]
    for x in astr:
        if x not in opt_lst:
            stack.append(treenode(x))
        else:
            num_2=stack.pop()
            num_1=stack.pop()
            node=treenode(x)
            node.leftchild=num_1
            node.rightchild=num_2
            stack.append(node)
    return stack[0]

output=[]
stack=[]
def queue(node):
    output.insert(0,node.key)
    left=node.leftchild
    right=node.rightchild
    stack.insert(0,left)
    stack.insert(0,right)
    while stack:
        while stack[-1].leftchild:
            new_node=stack.pop()
            queue(new_node)
            if not stack:
                break
        else:
            output.insert(0,stack.pop().key)
    return ''.join(output)

n=int(input())
anslst=[]
for _ in range(n):
    anslst.append(queue(buildtree(input())))
    stack=[]
    output=[]
for x in anslst:
    print(x)
```

基本信息

#:	44400858
题目:	25140
提交人:	zxk
内存:	4952kB
时间:	33ms
语言:	Python3
提交时间:	2024-03-25 21:16:51

©2002-2022 POJ 京ICP备20010980号-1　　English 帮助 关于

24750: 根据二叉树中后序序列建树

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

思路：

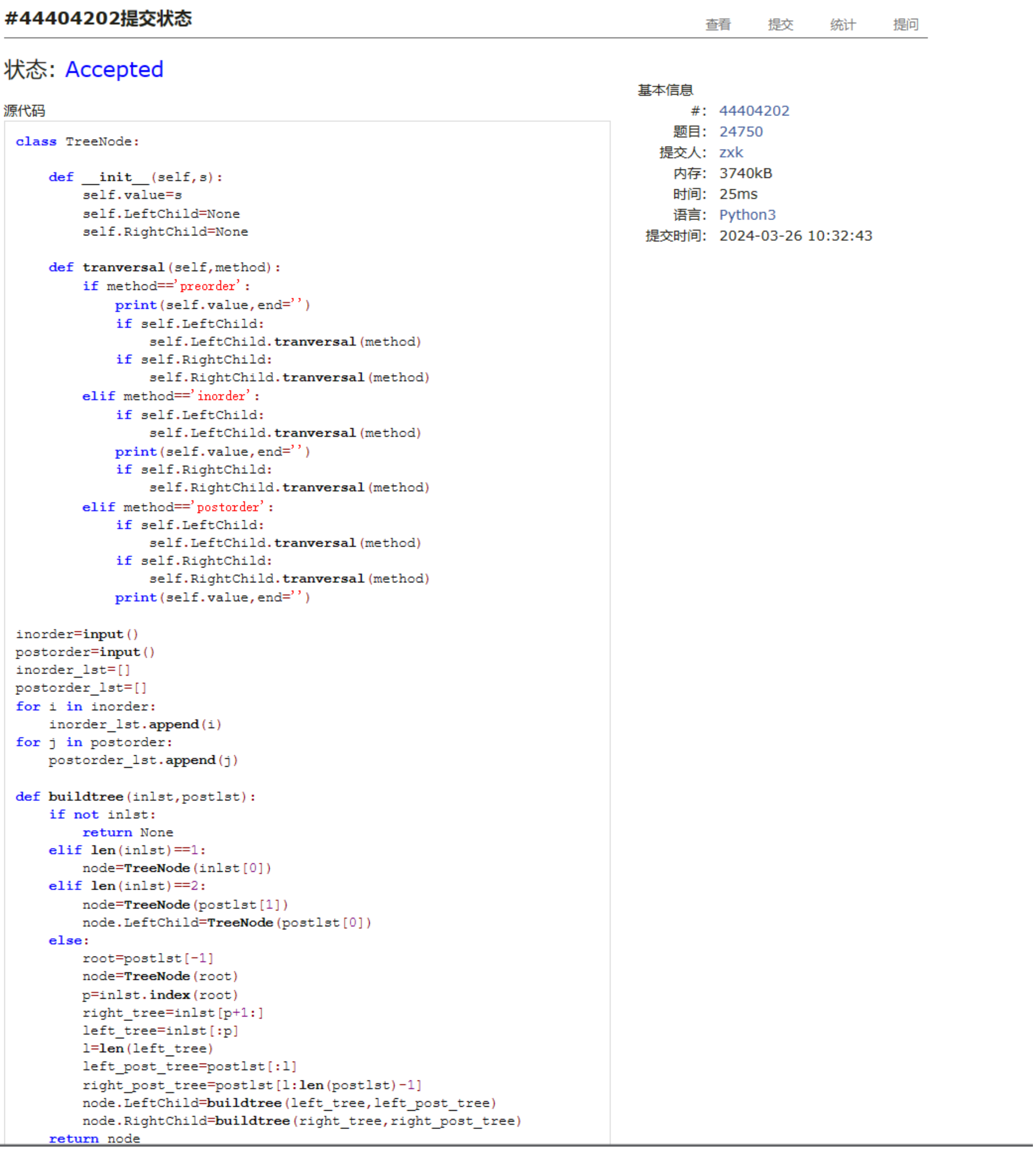
建树，通过后序的最后一个对中序查找，划分，然后递归建左右树，最后前序遍历

代码

```
class TreeNode:

    def __init__(self,s):
        self.value=s
        self.LeftChild=None
        self.RightChild=None

    def tranversal(self,method):
```

```python
        if method=='preorder':
            print(self.value,end='')
            if self.LeftChild:
                self.LeftChild.tranversal(method)
            if self.RightChild:
                self.RightChild.tranversal(method)
        elif method=='inorder':
            if self.LeftChild:
                self.LeftChild.tranversal(method)
            print(self.value,end='')
            if self.RightChild:
                self.RightChild.tranversal(method)
        elif method=='postorder':
            if self.LeftChild:
                self.LeftChild.tranversal(method)
            if self.RightChild:
                self.RightChild.tranversal(method)
            print(self.value,end='')

inorder=input()
postorder=input()
inorder_lst=[]
postorder_lst=[]
for i in inorder:
    inorder_lst.append(i)
for j in postorder:
    postorder_lst.append(j)

def buildtree(inlst,postlst):
    if not inlst:
        return None
    elif len(inlst)==1:
        node=TreeNode(inlst[0])
    elif len(inlst)==2:
        node=TreeNode(postlst[1])
        node.LeftChild=TreeNode(postlst[0])
    else:
        root=postlst[-1]
        node=TreeNode(root)
        p=inlst.index(root)
        right_tree=inlst[p+1:]
        left_tree=inlst[:p]
        l=len(left_tree)
        left_post_tree=postlst[:l]
        right_post_tree=postlst[l:len(postlst)-1]
        node.LeftChild=buildtree(left_tree,left_post_tree)
        node.RightChild=buildtree(right_tree,right_post_tree)
    return node

tree=buildtree(inorder_lst,postorder_lst)

tree.tranversal('preorder')#
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



#44404202提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class TreeNode:

    def __init__(self,s):
        self.value=s
        self.LeftChild=None
        self.RightChild=None

    def tranversal(self,method):
        if method=='preorder':
            print(self.value,end='')
            if self.LeftChild:
                self.LeftChild.tranversal(method)
            if self.RightChild:
                self.RightChild.tranversal(method)
        elif method=='inorder':
            if self.LeftChild:
                self.LeftChild.tranversal(method)
            print(self.value,end='')
            if self.RightChild:
                self.RightChild.tranversal(method)
        elif method=='postorder':
            if self.LeftChild:
                self.LeftChild.tranversal(method)
            if self.RightChild:
                self.RightChild.tranversal(method)
            print(self.value,end='')

inorder=input()
postorder=input()
inorder_lst=[]
postorder_lst=[]
for i in inorder:
    inorder_lst.append(i)
for j in postorder:
    postorder_lst.append(j)

def buildtree(inlst,postlst):
    if not inlst:
        return None
    elif len(inlst)==1:
        node=TreeNode(inlst[0])
    elif len(inlst)==2:
        node=TreeNode(postlst[1])
        node.LeftChild=TreeNode(postlst[0])
    else:
        root=postlst[-1]
        node=TreeNode(root)
        p=inlst.index(root)
        right_tree=inlst[p+1:]
        left_tree=inlst[:p]
        l=len(left_tree)
        left_post_tree=postlst[:l]
        right_post_tree=postlst[l:len(postlst)-1]
        node.LeftChild=buildtree(left_tree,left_post_tree)
        node.RightChild=buildtree(right_tree,right_post_tree)
    return node
```

基本信息

#:	44404202
题目:	24750
提交人:	zxk
内存:	3740kB
时间:	25ms
语言:	Python3
提交时间:	2024-03-26 10:32:43

22158: 根据二叉树前中序序列建树

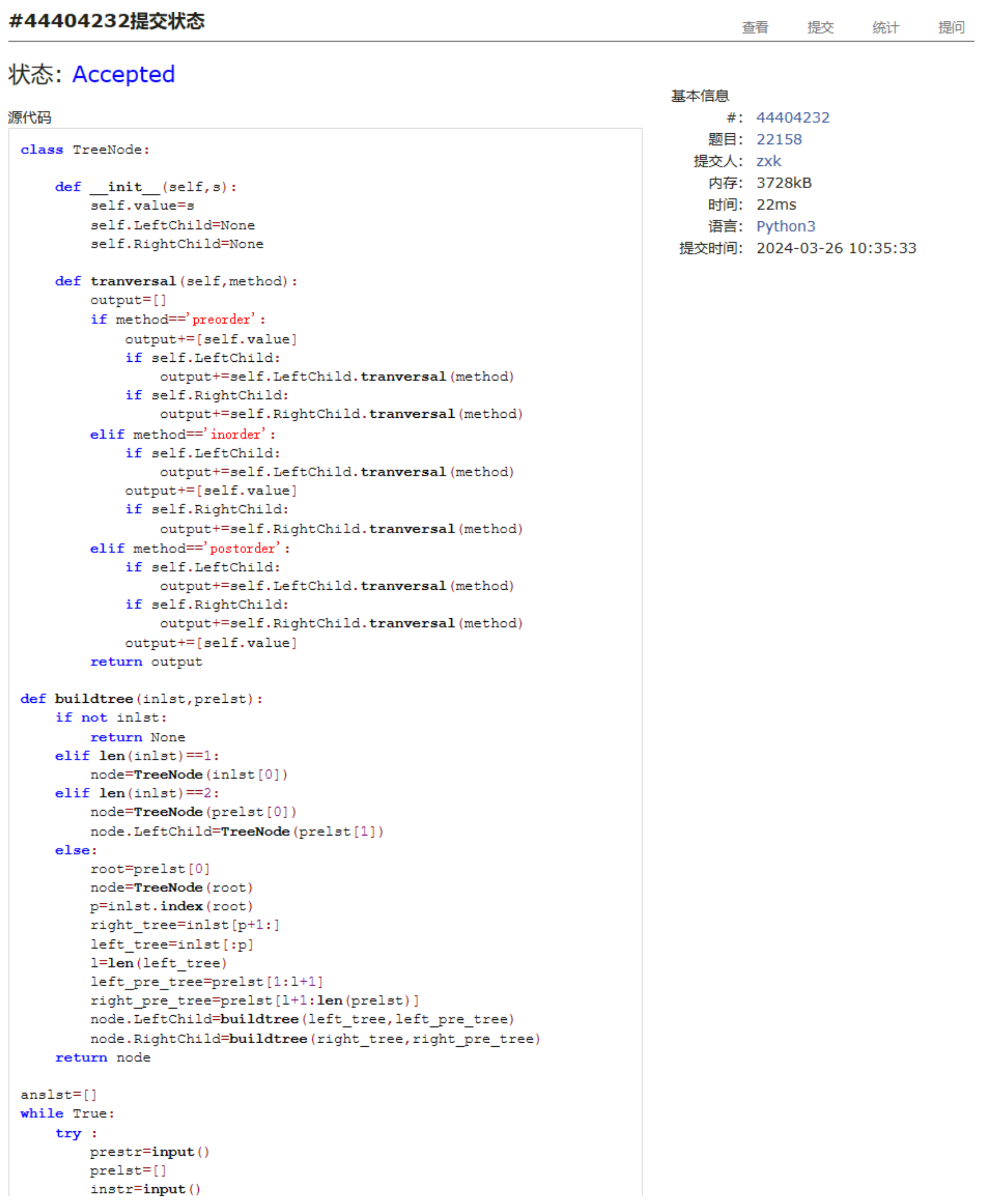
http://cs101.openjudge.cn/practice/22158/

思路：

代码

```python
class TreeNode:

    def __init__(self,s):
        self.value=s
        self.LeftChild=None
        self.RightChild=None

    def tranversal(self,method):
        output=[]
        if method=='preorder':
            output+=[self.value]
            if self.LeftChild:
                output+=self.LeftChild.tranversal(method)
            if self.RightChild:
                output+=self.RightChild.tranversal(method)
        elif method=='inorder':
            if self.LeftChild:
                output+=self.LeftChild.tranversal(method)
            output+=[self.value]
            if self.RightChild:
                output+=self.RightChild.tranversal(method)
        elif method=='postorder':
            if self.LeftChild:
                output+=self.LeftChild.tranversal(method)
            if self.RightChild:
                output+=self.RightChild.tranversal(method)
            output+=[self.value]
        return output

def buildtree(inlst,prelst):
    if not inlst:
        return None
    elif len(inlst)==1:
        node=TreeNode(inlst[0])
    elif len(inlst)==2:
        node=TreeNode(prelst[0])
        node.LeftChild=TreeNode(prelst[1])
    else:
        root=prelst[0]
        node=TreeNode(root)
        p=inlst.index(root)
        right_tree=inlst[p+1:]
        left_tree=inlst[:p]
        l=len(left_tree)
        left_pre_tree=prelst[1:l+1]
        right_pre_tree=prelst[l+1:len(prelst)]
        node.LeftChild=buildtree(left_tree,left_pre_tree)
        node.RightChild=buildtree(right_tree,right_pre_tree)
    return node

anslst=[]
while True:
    try :
        prestr=input()
        prelst=[]
        instr=input()
        inlst=[]
        for i in instr:
            inlst.append(i)
        for j in prestr:
            prelst.append(j)
        tree=buildtree(inlst,prelst)
        anslst.append(tree.tranversal('postorder'))
    except EOFError:
        break
for i in anslst:
    print(''.join(i))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



#44404232提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class TreeNode:

    def __init__(self,s):
        self.value=s
        self.LeftChild=None
        self.RightChild=None

    def tranversal(self,method):
        output=[]
        if method==’preorder’:
            output+=[self.value]
            if self.LeftChild:
                output+=self.LeftChild.tranversal(method)
            if self.RightChild:
                output+=self.RightChild.tranversal(method)
        elif method==’inorder’:
            if self.LeftChild:
                output+=self.LeftChild.tranversal(method)
            output+=[self.value]
            if self.RightChild:
                output+=self.RightChild.tranversal(method)
        elif method==’postorder’:
            if self.LeftChild:
                output+=self.LeftChild.tranversal(method)
            if self.RightChild:
                output+=self.RightChild.tranversal(method)
            output+=[self.value]
        return output

def buildtree(inlst,prelst):
    if not inlst:
        return None
    elif len(inlst)==1:
        node=TreeNode(inlst[0])
    elif len(inlst)==2:
        node=TreeNode(prelst[0])
        node.LeftChild=TreeNode(prelst[1])
    else:
        root=prelst[0]
        node=TreeNode(root)
        p=inlst.index(root)
        right_tree=inlst[p+1:]
        left_tree=inlst[:p]
        l=len(left_tree)
        left_pre_tree=prelst[1:l+1]
        right_pre_tree=prelst[l+1:len(prelst)]
        node.LeftChild=buildtree(left_tree,left_pre_tree)
        node.RightChild=buildtree(right_tree,right_pre_tree)
    return node

anslst=[]
while True:
    try :
        prestr=input()
        prelst=[]
        instr=input()
```

基本信息

#:	44404232
题目:	22158
提交人:	zxk
内存:	3728kB
时间:	22ms
语言:	Python3
提交时间:	2024-03-26 10:35:33

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。

1.二叉树高度：二叉树里的节点是可以只有一个子节点的，看来是对概念没记住啊！递归求高度的方法倒是还没问题

2.括号嵌套树：普通解析树的模板，用节点值加列表表示子树，加入cheating paper；不同题目的区别是不同的接口

3.文件结构“图”：代码复用，区别是如何解析输入的内容，遍历同样还是递归，带一个level的参数可以方便判断输出是第几层；值得注意的是有一个"def function(x):\n return x.key"+"xxx.sort(key=function)"的方法对子节点按其节点值排序，学到了

4.后序表达式转队列表达式：核心思路是逆着队列表达式读取的方式对其进行还原，当stack中的最后一项是一个非叶节点时，对其进行递归拆开，直到是叶节点时，把叶节点从前端嵌入output里，直到stack空掉，输出output；这个题开始的时候报错，stack是空的但是还在执行某个循环，然后感觉应该是某个地方stack已经空了但是没有跳出循环，不知道具体问题出在哪，于是随便改了一个地方检查是否空栈，空栈则跳出，没想到莫名其妙就跑通了……然后就秉着能跑就别动的原则投到oj上，没想到就ac了！赢

5.二叉树中后序序列建树：思路其实很简单，卡住的问题在于对前中后序表达式的定义没理解，如果只有一个子树，例如AB，中序和后序有可能一致有可能不一致，后序一定是先子后父，但是中序不一定，因为子不一定是左或者右

6.同上

整体来说会比之前做题顺畅一些，但是个别的代码有一定的问题，主要是针对某一个概念的理解或者某一段代码的思路，整体上对建树、遍历、递归等方法还是掌握的更深了一些，每日选择还是没顾上做，感觉压力比较大，争取之后做一些关于数算的题，把课件上的题都自己编下来