Assignment #5: "树"算：概念、表示、解析、遍历

Updated 2124 GMT+8 March 17, 2024

2024 spring, Complied by ==张宇帆 心理与认知科学学院==

编程环境

==（请改为同学的操作系统、编程环境等）==

Python编程环境：Spyder IDE 5.2.2

1. 题目

27638: 求二叉树的高度和叶子数目

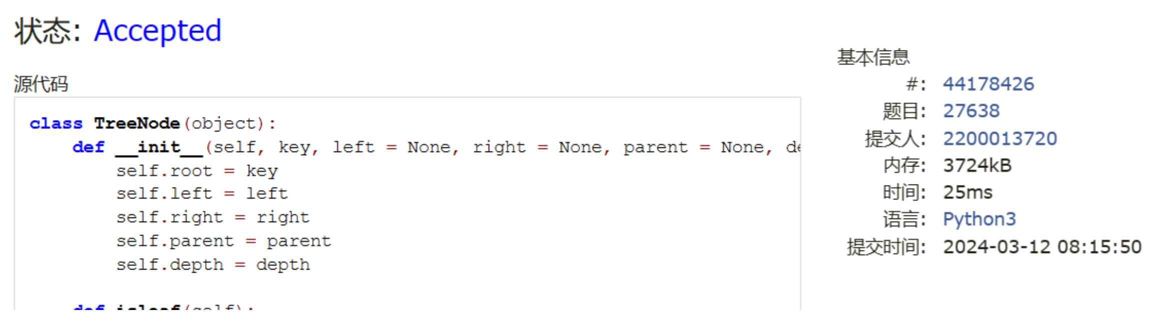
http://cs101.openjudge.cn/practice/27638/

思路：设置节点的depth参数，叶节点为1，父节点为子节点中最大depth + 1；叶子数目在输入时即可统计；由于输入无法分辨谁为根节点，所以还写了个寻找根节点的函数。

代码

```
# 
class TreeNode(object):
    def __init__(self, key, left = None, right = None, parent = None, depth = 1):
        self.root = key
        self.left = left
        self.right = right
        self.parent = parent
        self.depth = depth

    def isleaf(self):
        return self.left == None and self.right == None

def finddepth(treenode, trees):
    if treenode.isleaf():
        treenode.depth = 1
    if not treenode.left == None:
        finddepth(treenode.left, trees)
        treenode.depth = max(treenode.left.depth + 1, treenode.depth)
    if not treenode.right == None:
        finddepth(treenode.right, trees)
        treenode.depth = max(treenode.right.depth + 1, treenode.depth)
    return treenode.depth

def findroot(trees):
    for tree in trees:
        if tree.parent == None:
            return tree.root

n = int(input())
trees = [TreeNode(i) for i in range(n)]
numleaf = 0
for i in range(n):
    left, right = map(int,input().split())
    if not left == -1:
        trees[i].left = trees[left]
        trees[left].parent = trees[i]
    if not right == -1:
        trees[i].right = trees[right]
        trees[right].parent = trees[i]
    if left == -1 and right == -1:
        numleaf += 1
root = findroot(trees)
depth = finddepth(trees[root], trees)
print("%s %s"%(depth-1, numleaf))
```

代码运行截图 ==（至少包含有"Accepted"）==


状态: Accepted

源代码

```
class TreeNode(object):
    def __init__(self, key, left = None, right = None, parent = None, de
        self.root = key
        self.left = left
        self.right = right
        self.parent = parent
        self.depth = depth

    def isleaf(self):
```

基本信息

#:	44178426
题目:	27638
提交人:	2200013720
内存:	3724kB
时间:	25ms
语言:	Python3
提交时间:	2024-03-12 08:15:50

24729: 括号嵌套树

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

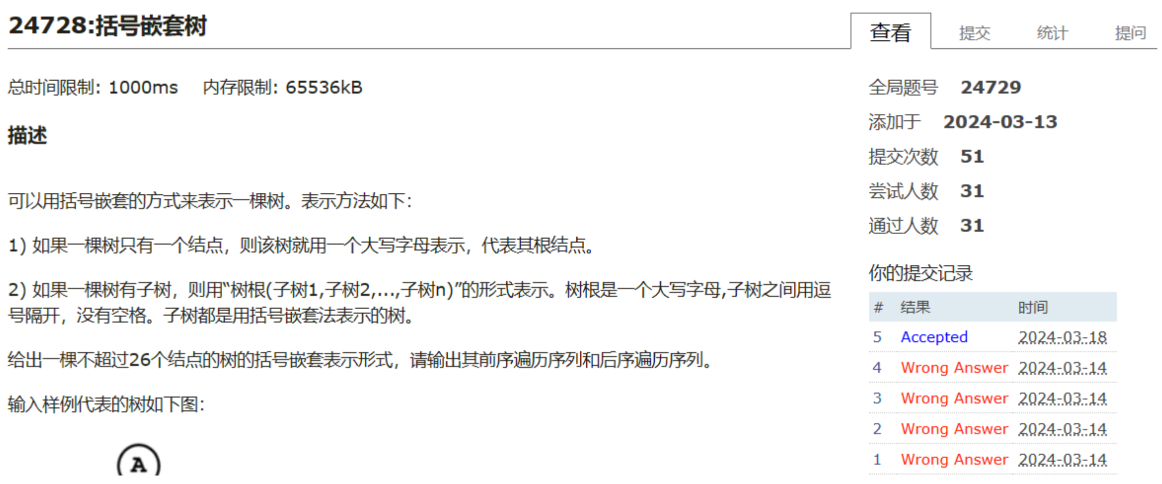
思路：关键在于如何区分出每一层的子节点，一开始样例过了但WA了好多次都想不明白错在哪。多天之后又在样例的F里面套了个子树才发现问题：添加 ',' 链接子节点部分的判断错了。修改了一下就AC了。可能是因为当初做的时候匆匆忙忙被其他课程搞得焦头烂额吧。以后还是要放平心态对待每道题。

代码

```
#
class TreeNode(object):
    def __init__(self, root, child = []):
        self.root = root
        self.child = child

def preloop(tree, result):
    if not tree == None:
        result.append(str(tree.root))
        for child in tree.child:
```

```python
            result = preloop(child, result)
    return result

def postloop(tree, result):
    if not tree == None:
        for child in tree.child:
            result = postloop(child, result)
        result.append(str(tree.root))
    return result

def rebuild(treestr):
    if len(treestr) == 1:
        return TreeNode(treestr)
    elif len(treestr) > 1:
        temp = TreeNode(treestr[0], child = [])
        children = treestr[2:-1].split(',')
        child = ''
        for i in children:
            child += i
            if len(child) == 1:
                temp.child.append(rebuild(child))
                child = ''
            elif not child.count('(') == child.count(')'):
                child += ','
            else:
                temp.child.append(rebuild(child))
                child = ''
        return temp
s = input()
tree = rebuild(s)
result = []
print(''.join(preloop(tree, result)))
result = []
print(''.join(postloop(tree, result)))
```

代码运行截图 == （至少包含有"Accepted"） ==



24728:括号嵌套树

查看 提交 统计 提问

总时间限制: 1000ms 内存限制: 65536kB

描述

可以用括号嵌套的方式来表示一棵树。表示方法如下:

1) 如果一棵树只有一个结点，则该树就用一个大写字母表示，代表其根结点。

2) 如果一棵树有子树，则用"树根(子树1,子树2,...,子树n)"的形式表示。树根是一个大写字母,子树之间用逗号隔开，没有空格。子树都是用括号嵌套法表示的树。

给出一棵不超过26个结点的树的括号嵌套表示形式，请输出其前序遍历序列和后序遍历序列。

输入样例代表的树如下图:

A

全局题号 24729
添加于 2024-03-13
提交次数 51
尝试人数 31
通过人数 31

你的提交记录

#	结果	时间
5	Accepted	2024-03-18
4	Wrong Answer	2024-03-14
3	Wrong Answer	2024-03-14
2	Wrong Answer	2024-03-14
1	Wrong Answer	2024-03-14

02775: 文件结构“图”

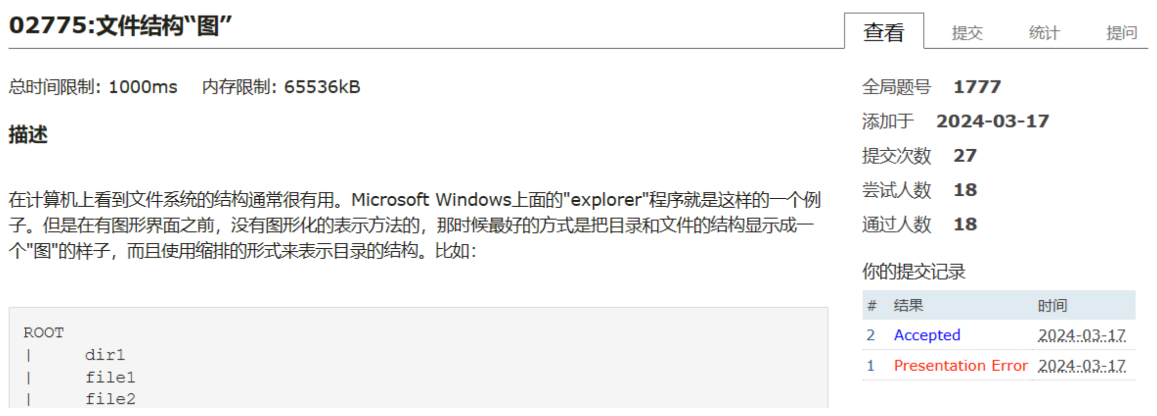
http://cs101.openjudge.cn/practice/02775/

思路：由于对图还不是很了解，所以本来想尝试着写一下图，后来想想能不能用树实现一下，然后发现其实还不算很难（虽然用的还是图的思路），之后会尝试写一下图的。

代码

```python
#
class TreeNode(object):
    def __init__(self, key, dirs = [], files = [], layer = 0, pre = None):
        self.key = key
        self.dirs = dirs
        self.files = files
        self.layer = layer
        self.pre = pre

def write(treenode):
    print('|     '*treenode.layer + treenode.key)
    for dr in treenode.dirs:
        write(dr)
    treenode.files.sort()
    for fl in treenode.files:
        print('|     '*treenode.layer + fl)


code = 0
word = 'ROOT'
Node = TreeNode(word)
while not word == '#':
    if word == '*':
        code += 1
        print('DATA SET %s:'%code)
        write(Node)
        print()
        Node = TreeNode('ROOT', dirs = [], files = [], layer = 0)
    elif word[0] == 'f':
        Node.files.append(word)
    elif word[0] == 'd':
        newNode = TreeNode(word, dirs = [], files = [], layer = 0)
        Node.dirs.append(newNode)
        newNode.pre = Node
        newNode.layer = Node.layer + 1
        Node = newNode
    elif word == ']':
        Node = Node.pre
    word = input()
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==


02775:文件结构“图”

查看 提交 统计 提问

总时间限制: 1000ms 内存限制: 65536kB

描述

在计算机上看到文件系统的结构通常很有用。Microsoft Windows上面的"explorer"程序就是这样的一个例子。但是在有图形界面之前，没有图形化的表示方法的，那时候最好的方式是把目录和文件的结构显示成一个"图"的样子，而且使用缩排的形式来表示目录的结构。比如:

```
ROOT
|     dir1
|     file1
|     file2
```

全局题号 1777
添加于 2024-03-17
提交次数 27
尝试人数 18
通过人数 18

你的提交记录

#	结果	时间
2	Accepted	2024-03-17
1	Presentation Error	2024-03-17

25140: 根据后序表达式建立队列表达式

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

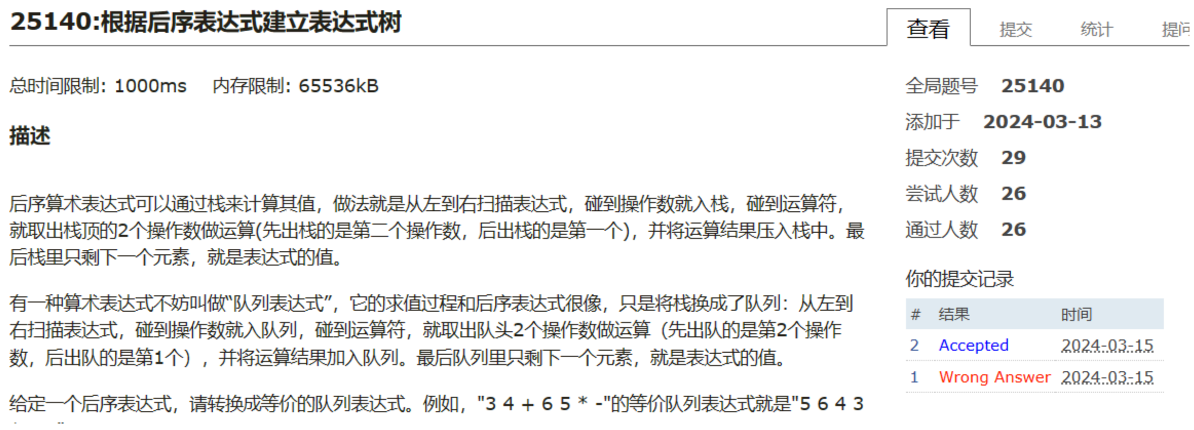
思路：感觉就是常规的思路，不过还是很有收获的，因为知道了层次遍历怎么写。一开始总想着跟前序、后序遍历一样简单，但就是实现不了，最后用dfs才完成的。

对了，这次还发现了之前没有发觉的一点，在函数中已经设置了默认值（如layerloop里的result），那么在函数结束并输出后，再重新调用时他并不会变为[]，而是保留了上一个函数所产生的值（运行的时候就发现了，提交的时候忘改了导致了第一次WA），但是如果设置为None就不会有这一点。所以以后要设置参数还是先设置为None然后在函数里定义就好了。

代码

```python
#
class TreeNode(object):
    def __init__(self, root, left = None, right = None, parent = None):
        self.root = root
        self.left = left
        self.right = right
        self.parent = parent

def trans_to_tree(formula):
    if formula:
        key = formula.pop()
        if key.islower():
            temp = TreeNode(key)
        elif key.isupper():
            temp = TreeNode(key)
            temp.left = trans_to_tree(formula)
            temp.right = trans_to_tree(formula)
    return temp

def layerloop(treenode, result = []):
    queue = [treenode]
    while queue:
        for _ in range(len(queue)):
            node = queue.pop(0)
            result.append(node.root)
            if node.right:
                queue.append(node.right)
            if node.left:
                queue.append(node.left)
    return ''.join(result)

n = int(input())
for i in range(n):
    formula = list(input())
    result = []
    print(layerloop(trans_to_tree(formula), result)[::-1])
```

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表达式树

查看 提交 统计 提问

总时间限制: 1000ms 内存限制: 65536kB

描述

后序算术表达式可以通过栈来计算其值，做法就是从左到右扫描表达式，碰到操作数就入栈，碰到运算符，就取出栈顶的2个操作数做运算(先出栈的是第二个操作数，后出栈的是第一个)，并将运算结果压入栈中。最后栈里只剩下一个元素，就是表达式的值。

有一种算术表达式不妨叫做“队列表达式”，它的求值过程和后序表达式很像，只是将栈换成了队列：从左到右扫描表达式，碰到操作数就入队列，碰到运算符，就取出队头2个操作数做运算（先出队的是第2个操作数，后出队的是第1个），并将运算结果加入队列。最后队列里只剩下一个元素，就是表达式的值。

给定一个后序表达式，请转换成等价的队列表达式。例如，"3 4 + 6 5 * -"的等价队列表达式就是"5 6 4 3

全局题号 25140
添加于 2024-03-13
提交次数 29
尝试人数 26
通过人数 26

你的提交记录

#	结果	时间
2	Accepted	2024-03-15
1	Wrong Answer	2024-03-15

24750: 根据二叉树中后序序列建树

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

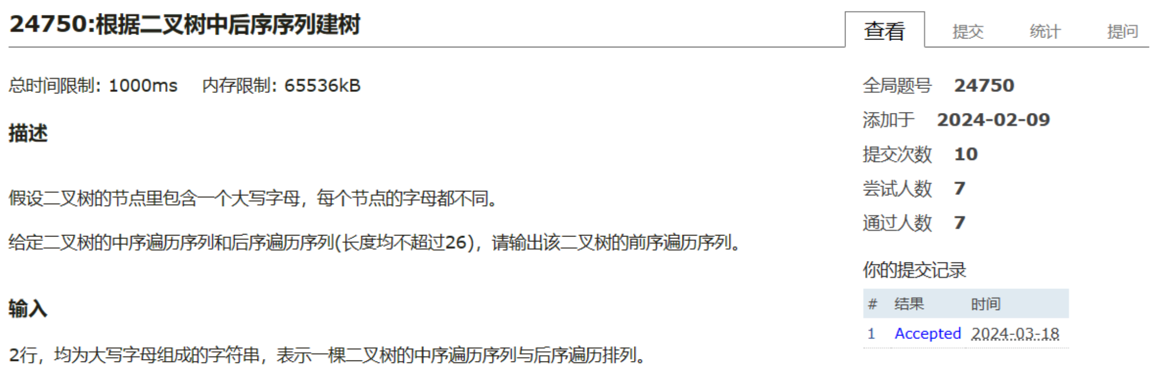
思路：这道题似乎和每日选做里的另一道题很像：22158:根据二叉树前中序序列建树。当初做完这道题后就直接写了个通用的模板，写的过程中发现中序遍历对于形成一棵树是必要的，很有意思。

还有就是，其实我一开始的思路不是直接建树（以下给的是建树的代码），而是直接截取字符串然后拼接，我感觉那个会更简单而且更好理解，不过写通用的时候考虑到：①以为中序是可以去掉的，用字符串方法写起来会有点麻烦；②更多情境下其实更需要你构建起完整的树。所以就写了个建树的通用。字符串的方法我会在作业最后给出，就以“22158:根据二叉树前中序序列建树”为例

代码

```python
#
class TreeNode(object):
    def __init__(self, root, left = None, right = None, parent = None):
        self.root = root
        self.left = left
        self.right = right
        self.parent = parent

def trans_to_trees(pre = None, mid = None, post = None):
    if post == None:
        if len(pre) <= 1 or len(mid) <= 1:
            return TreeNode(pre)
        else:
            root = pre[0]
            idx = mid.find(root)
            temp = TreeNode(root)
            temp.left = trans_to_trees(pre[1:1+idx], mid[:idx], post)
            temp.right = trans_to_trees(pre[1+idx:], mid[idx+1:], post)
            return temp

    elif mid == None:
        error = 'mid is necessary'
        return error

    elif pre == None:
        if len(mid) <= 1 or len(post) <= 1:
            return TreeNode(mid)
        else:
            root = post[-1]
            idx = mid.find(root)
            temp = TreeNode(root)
            temp.left = trans_to_trees(pre, mid[:idx], post[:idx])
            temp.right = trans_to_trees(pre, mid[idx+1:], post[idx:-1])
            return temp

def preloop(tree, result):
    if (not tree == None) and (not tree.root == '*') and (not tree.root == ''):
        result.append(str(tree.root))
        result = preloop(tree.left, result)
        result = preloop(tree.right, result)
    return result

def midloop(tree, result):
    if (not tree == None) and (not tree.root == '*') and (not tree.root == ''):
        result = midloop(tree.left, result)
        result.append(str(tree.root))
        result = midloop(tree.right, result)
    return result

mid = input()
post = input()
tree = trans_to_trees(None, mid, post)
result = []
print(''.join(preloop(tree, result)))
```

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

查看 提交 统计 提问

总时间限制: 1000ms 内存限制: 65536kB

描述

假设二叉树的节点里包含一个大写字母，每个节点的字母都不同。

给定二叉树的中序遍历序列和后序遍历序列(长度均不超过26)，请输出该二叉树的前序遍历序列。

输入

2行，均为大写字母组成的字符串，表示一棵二叉树的中序遍历序列与后序遍历排列。

全局题号	24750
添加于	2024-02-09
提交次数	10
尝试人数	7
通过人数	7

你的提交记录

#	结果	时间
1	Accepted	2024-03-18

22158: 根据二叉树前中序序列建树

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

思路：刚写完前一道的思路才发现接下来就是这一道）那我就写写字符串的思路吧：

其实也很简单，分为三步：①确定根节点。由于前序遍历根节点在前，所以根节点必然是第一个字母

②确定左子树长度。由于中序遍历的根节点往左就是左子树，所以找到根节点在中序遍历中的idx即为左子树的长度。

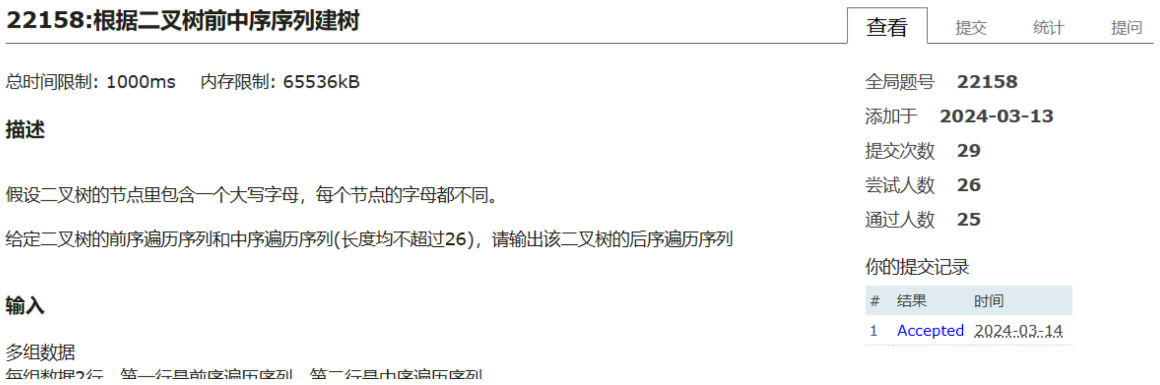
③形成递归。pre[1:1+idx]即为左子树的前序遍历，mid[:idx]即为左子树的中序遍历；pre[1+idx:]即为右子树的前序遍历，mid[idx+1:]即为右子树的中序遍历，最后当节点数为1时返回节点值即可。应当注意的是，得把pre[0]和mid[idx]去掉（因为是根节点）

以下代码中的 <=1其实是有缺陷的，会导致空字符的返回，可以改进一下，不过最后输出结果相同，但在搭建树的过程中就需要考虑了。

代码

```python
# 
def trans_to_post(pre, mid):
    if len(pre) <= 1 and len(mid) <= 1:
        return pre
    else:
        root = pre[0]
        idx = mid.find(root)
        return trans_to_post(pre[1:1+idx], mid[:idx]) + 
trans_to_post(pre[1+idx:], mid[idx+1:]) + root

while True:
    try:
        pre = input()
        mid = input()
        print(trans_to_post(pre, mid))
    except EOFError:
        break
```

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

查看 提交 统计 提问

总时间限制: 1000ms 内存限制: 65536kB

描述

假设二叉树的节点里包含一个大写字母，每个节点的字母都不同。

给定二叉树的前序遍历序列和中序遍历序列(长度均不超过26)，请输出该二叉树的后序遍历序列

输入

多组数据

全局题号 22158

添加于 2024-03-13

提交次数 29

尝试人数 26

通过人数 25

你的提交记录

#	结果	时间
1	Accepted	2024-03-14

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。==

现在每天就跟着每日选做做题，主要时间花在了其他课程上，但一空闲下来就会去CF和OJ上逛逛看看有没有感兴趣的题目。每日题中图的出现让我意识到该开始加快点进度了。AVL树还是不知道怎么实现（就是没有亲手敲出代码解决题目过），但是对于一般的树题目我感觉还是能解决的。