Assignment #5: "树"算：概念、表示、解析、遍历

Updated 2124 GMT+8 March 17, 2024

2024 spring, Complied by 城环 吴至超

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：Windows11

Python编程环境：Pycharm2023.2.3

1. 题目

27638: 求二叉树的高度和叶子数目

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

思路：递归取两边的最大高度

代码

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

n=int(input())#结点个数0~n-1
```

```python
arr=[tree() for _ in range(n)]#建立树
yezi=0
for i in range(n):
    x,y=map(int,input().split())
    arr[i].left=x
    arr[i].right=y
    if x==-1 and y==-1:
        yezi+=1


def count(tree):
    if tree.left!=-1 and tree.right!=-1:
        return 1+max(count(arr[tree.left]),count(arr[tree.right]))
    elif tree.left==-1 and tree.right!=-1:
        return 1+count(arr[tree.right])
    elif tree.left!=-1 and tree.right==-1:
        return 1+count(arr[tree.left])
    else:
        return 1
cnt=count(arr[0])
for m in arr[1::]:
    p=count(m)#记录叶子数
    if p>cnt:
        cnt=p
print(cnt-1,yezi)
```

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：结合栈来构建树，前序与后序遍历的递归写法

代码

```
# #括号嵌套树
class tree:
    def __init__(self,name):
        self.name=name
        self.children=[]
sample=input()

#A(B(E),C(F,G),D(H(I)))
def buildtree(sam):
    stack=[]

    for i in sam:
        if i.isalpha():
            unix=tree(i)
            if stack:
                stack[-1].children.append(unix)
        elif i=="(":#表示该字母带孩子
            stack.append(unix)
        elif i==")":
            unix=stack.pop()#被pop出去后仍然保留其形式不变
    return unix#根节点，类型tree

def qianxu(root):#dfs
    output=[root.name]
    for m in root.children:
        output.extend(qianxu(m))
    return "".join(output)

def houxu(root):
    output=[]
    for t in root.children:
        output.extend(houxu(t))
    output.append(root.name)
    return "".join(output)

print(qianxu(buildtree(sample)))
print(houxu(buildtree(sample)))
```

代码运行截图 ==（至少包含有"Accepted"）==

02775: 文件结构“图”

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

思路：如何处理这样的输入—>先放在一个[][][[ ],[ ],[ ]]之中再去遍历

如何处理这样的输出—>写一个递归的print

代码

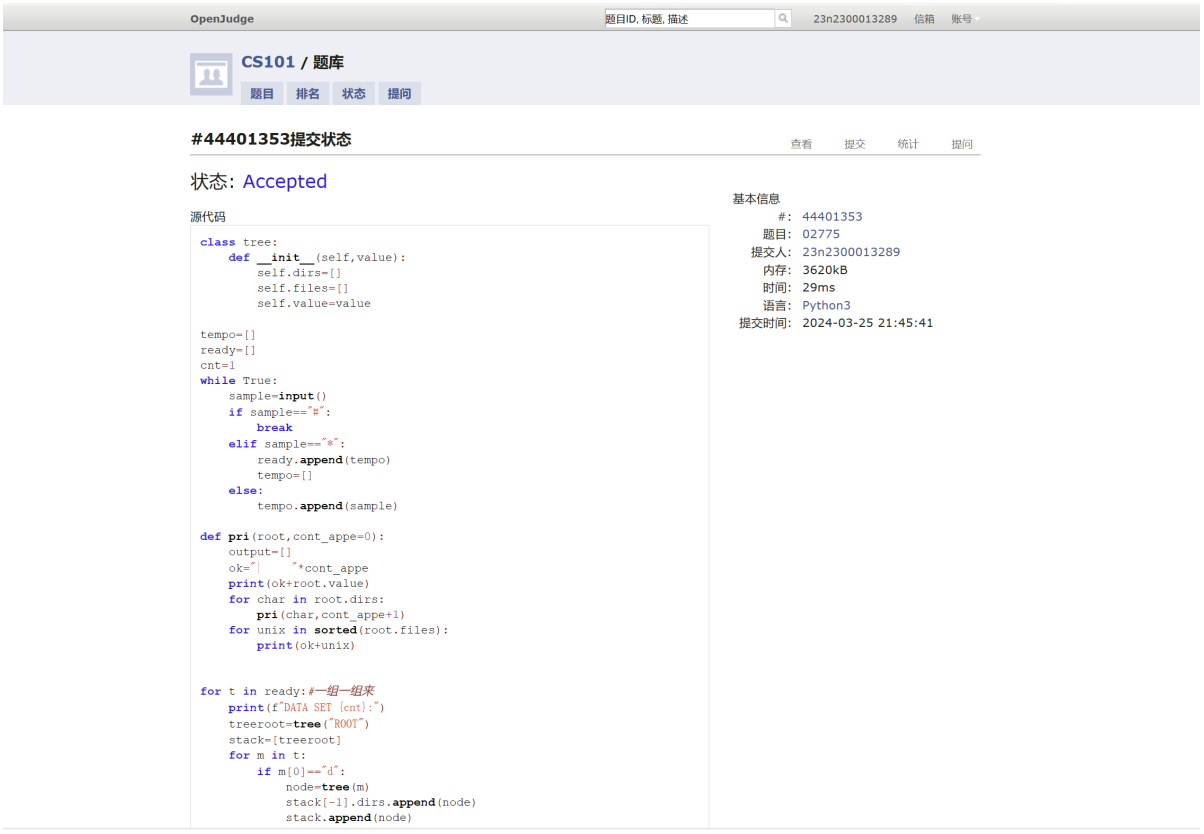
```
# class tree:
    def __init__(self,value):
        self.dirs=[]
        self.files=[]
        self.value=value

tempo=[]
ready=[]
cnt=1
while True:
    sample=input()
    if sample=="#":
        break
    elif sample=="*":
        ready.append(tempo)
        tempo=[]
    else:
        tempo.append(sample)
```

```
def pri(root,cont_appe=0):
    output=[]
    ok="|      "*cont_appe
    print(ok+root.value)
    for char in root.dirs:
        pri(char,cont_appe+1)
    for unix in sorted(root.files):
        print(ok+unix)


for t in ready:#一组一组来
    print(f"DATA SET {cnt}:")
    treeroot=tree("ROOT")
    stack=[treeroot]
    for m in t:
        if m[0]=="d":
            node=tree(m)
            stack[-1].dirs.append(node)
            stack.append(node)
        elif m[0]=="f":
            stack[-1].files.append(m)
        else:
            stack.pop()
    pri(stack[0])
    if ready.index(t) !=len(ready)-1:
        print()
    cnt+=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队列表达式

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

思路：如何建树—>结合栈

如何进行层次遍历—>在列表中一遍遍历一遍拉长列表

代码

```
# class tree:
    def __init__(self,value):
        self.value=value
        self.left=None
        self.right=None
def buildtree(sample):#把树建好
    aa=sample[::-1]
    stack=[tree(aa[0])]#装未满子节点的树
    for y in aa[1::]:
        node=tree(y)
        if y.isupper():
            while stack[-1].right!=None and stack[-1].left!=None:
                stack.pop()
            if stack[-1].right==None:
                stack[-1].right=node
            else:
                stack[-1].left=node
            stack.append(node)
        else:
            while stack[-1].right!=None and stack[-1].left!=None:
                stack.pop()
            if stack[-1].right==None:
                stack[-1].right=node
            else:
                stack[-1].left=node
    return stack[0]

n=int(input())
for p in range(n):
    sample=input()
    queue=[buildtree(sample)]



    flag=0
    while len(queue)<len(sample):
        if queue[flag].left !=None:
            queue.append(queue[flag].left)
        if queue[flag].right !=None:
            queue.append(queue[flag].right)
        flag+=1
```

```
    output=[x.value for x in queue]

    print("".join(output[::-1]))
```

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：如何建立树——>使用递归，好难想，需要熟悉后序遍历的逻辑

代码

```
#
#24750
zhongxu=list(input())
houxu=list(input())
class tree:
    def __init__(self,value):
        self.left=None
        self.value=value
        self.right=None
def buildtree(zhongxu,houxu):
    if not zhongxu or not houxu:
        return None
    a = zhongxu.index(houxu[-1])
```

```
    rootnode=tree(houxu.pop())
    new1=zhongxu[0:a]
    new2=zhongxu[a+1::]
    rootnode.right=buildtree(zhongxu[a+1::],houxu)#一直使用同一个列表houxu，就成了全局变量
    rootnode.left=buildtree(zhongxu[:a],houxu)
    return rootnode
def preorder(root):
    stack = []#注意！
    if root:#终止条件
        stack.append(root.value)
        stack.extend(preorder(root.left))
        stack.extend(preorder(root.right))
    return stack
print("".join(preorder(buildtree(zhongxu,houxu))))
```

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：如何建立树——>使用递归，好难想，需要熟悉前序遍历的逻辑

代码

```
# #22158
class tree:
    def __init__(self,value):
```

```
        self.left=None
        self.right=None
        self.value=value
def buildtree(preorder,inorder):
    if not preorder or not inorder:
        return None
    a=inorder.index(preorder[0])
    treeroot=tree(preorder.pop(0))
    treeroot.left=buildtree(preorder,inorder[:a])
    treeroot.right=buildtree(preorder,inorder[a+1::])
    return treeroot
def postorder(root):
    stack=[]
    if root:
        stack.extend(postorder(root.left))
        stack.extend(postorder(root.right))
        stack.append(root.value)
    return stack
while True:
    try:
        preorder=list(input())
        inorder=list(input())
        print("".join(postorder(buildtree(preorder,inorder))))
    except EOFError:
        break
```

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

==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。==

第一题自己成功把一个完整的递归程序写出来了，激动！

第二题参考答案了解了用栈来建树的办法

于是做出了第四题

第三题被一开始被题目绕晕了，后来参考了答案打印函数的递归程序

第五题按照答案来理解，对那种递归程序tutor+手画了蛮久的

第六题类似第五题的思路可以解出

总体来说感觉到了递归在树中方方面面的使用，递归太强大的同时自己又写不出来，只是有那个想法但是望而却步，伤心💔

总的来说花了好多时间，但也熟悉了栈构建树的方式，children类的树，前后序中序层次遍历的写法