Assignment #5: "树"算：概念、表示、解析、遍历

Updated 2124 GMT+8 March 17, 2024

2024 spring, Complied by 夏天明 元培学院

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

操作系统：Windows 10 | 22H2

Python编程环境：Spyder IDE 5.4.3 | Python 3.11.4 64-bit

1. 题目

27638: 求二叉树的高度和叶子数目

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

思路：构造一个树，用递归求高度和叶子数目

代码

```
class Node:
    def __init__(self, name):
        self.name = name
        self.child = []
```

```
    def getHeight(self):
        return max([nd.getHeight() for nd in self.child], default=-1)+1

    def getLeafNum(self):
        return sum(nd.getLeafNum() for nd in self.child) if self.child else 1

n = int(input())
nodes = [Node(i) for i in range(n)]
not_root = set()
for nd in nodes:
    sub = list(map(int, input().split()))
    not_root.update(sub)
    nd.child.extend(nodes[s] for s in sub if s >= 0)
for i, nd in enumerate(nodes):
    if i not in not_root:
        print(nd.getHeight(), nd.getLeafNum())
        break
```

代码运行截图

24729: 括号嵌套树

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

思路：直接实现。但是注意一下叶子节点的情形。

代码

```
class Node:
    def __init__(self, name):
        self.name = name
        self.child = []

    def preSeq(self):
        return self.name + ''.join(nd.preSeq() for nd in self.child)

    def postSeq(self):
        return ''.join(nd.postSeq() for nd in self.child) + self.name

def call():
    nd = Node(s.pop())
    if s and s[-1] != '(':
        return nd
    while s:
        token = s.pop()
        if token in '(,':
            nd.child.append(call())
        else:
            return nd
    return nd

s = list(input())[::-1]
root = call()
print(root.preSeq())
print(root.postSeq())
```

代码运行截图

状态: Accepted

源代码

```
class Node:
    def __init__(self, name):
        self.name = name
        self.child = []

    def preSeq(self):
        return self.name + ''.join(nd.preSeq() for nd in self.child)

    def postSeq(self):
        return ''.join(nd.postSeq() for nd in self.child) + self.name

def call():
    nd = Node(s.pop())
    if s and s[-1] != '(':
        return nd
    while s:
        token = s.pop()
        if token in '(,':
            nd.child.append(call())
        else:
            return nd
    return nd

s = list(input())[::-1]
root = call()
print(root.preSeq())
print(root.postSeq())
```

基本信息

#: 44293093
题目: 24728
提交人: 23n2300017735(夏天明 BrightSummer)
内存: 5192kB
时间: 25ms
语言: Python3
提交时间: 2024-03-19 00:12:05

02775: 文件结构“图”

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

思路：建立文件夹类，存储有哪些子文件夹和文件，然后递归实现

代码

```
from sys import exit

class dir:
    def __init__(self, dname):
        self.name = dname
        self.dirs = []
        self.files = []

    def getGraph(self):
        g = [self.name]
        for d in self.dirs:
            subg = d.getGraph()
            g.extend(["|     " + s for s in subg])
        for f in sorted(self.files):
            g.append(f)
        return g
```

```
n = 0
while True:
    n += 1
    stack = [dir("ROOT")]
    while (s := input()) != "*":
        if s == "#": exit(0)
        if s[0] == 'f':
            stack[-1].files.append(s)
        elif s[0] == 'd':
            stack.append(dir(s))
            stack[-2].dirs.append(stack[-1])
        else:
            stack.pop()
    print(f"DATA SET {n}:")
    print(*stack[0].getGraph(), sep='\n')
    print()
```

代码运行截图

#41694325提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from sys import exit

class dir:
    def __init__(self, dname):
        self.name = dname
        self.dirs = []
        self.files = []

    def getGraph(self):
        g = [self.name]
        for d in self.dirs:
            subg = d.getGraph()
            g.extend(["|    " + s for s in subg])
        for f in sorted(self.files):
            g.append(f)
        return g

n = 0
while True:
    n += 1
    stack = [dir("ROOT")]
    while (s := input()) != "*":
        if s == "#": exit(0)
```

基本信息

#: 41694325
题目: 02775
提交人: 23n2300017735(夏天明 BrightSummer)
内存: 3720kB
时间: 36ms
语言: Python3
提交时间: 2023-10-16 10:59:56

25140: 根据后序表达式建立队列表达式

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

思路：直接实现

代码

```python
class Node:
    def __init__(self, val):
        self.val = val
        self.child = []


for o in range(int(input())):
    s = input()
    stack = []
    for token in s:
        if token.islower():
            stack.append(Node(token))
        else:
            chd = [stack.pop() for i in '12']
            stack.append(Node(token))
            stack[-1].child = chd[::-1]
    ans = []
    while stack:
        new = []
        for nd in stack:
            ans.append(nd.val)
            new.extend(nd.child)
        stack = new
    print(''.join(ans)[::-1])
```

代码运行截图

#44295904提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
class Node:
    def __init__(self, val):
        self.val = val
        self.child = []


for o in range(int(input())):
    s = input()
    stack = []
    for token in s:
        if token.islower():
            stack.append(Node(token))
        else:
            chd = [stack.pop() for i in '12']
            stack.append(Node(token))
            stack[-1].child = chd[::-1]
    ans = []
    while stack:
        new = []
        for nd in stack:
            ans.append(nd.val)
            new.extend(nd.child)
        stack = new
    print(''.join(ans)[::-1])
```

基本信息

#: 44295904
题目: 25140
提交人: 23n2300017735(夏天明BrightSummer)
内存: 3636kB
时间: 28ms
语言: Python3
提交时间: 2024-03-19 11:29:22

24750: 根据二叉树中后序序列建树

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

思路：经典题，每次从后序序列取出最后一个字母，这是当前的树根，然后利用中序序列找到子树大小

代码

```
def getPre(mid, post):
    if not mid:
        return ''
    i = mid.index(post[-1])
    return post[-1] + getPre(mid[:i], post[:i]) + getPre(mid[i+1:], post[i:-1])

print(getPre(input(), input()))
```

代码运行截图

22158: 根据二叉树前中序序列建树

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

思路：同上

代码

```python
def getPost(pre, mid):
    if not pre:
        return ''
    i = mid.index(pre[0])
    return getPost(pre[1:i+1], mid[:i]) + getPost(pre[i+1:], mid[i+1:]) + pre[0]

while True:
    try:
        print(getPost(input(), input()))
    except EOFError:
        break
```

代码运行截图

2. 学习总结和收获

树的关键思想：递归，将关于整棵树的问题化归为关于根节点和两棵子树的问题，进而递归得到解。

表达式解析：对于含有递归结构的表达式，使用维护栈的方法完成解析，或者定义一个递归函数，遇到特定token再call/return。两者本质是一样的：递归相当于维护调用栈。

桶的思想：数据结构知识的核心内容不涉及桶，但实际解题时处处能看到桶的踪影。例如，构建树时通常以列表或字典的形式维护一个桶，存储所有节点的标签所对应的实例对象的引用；以非递归的形式遍历树时，例如寻找根节点，经常用到桶的思想。桶的优势是非常易于遍历，统一处理各种数据等。