

Assignment #5: "树"算：概念、表示、解析、遍历

Updated 2124 GMT+8 March 17, 2024

2024 spring, Compiled 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），填写到下面作业模板中（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：照讲义写的

代码

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

def tree_height(node):
    if node is None:
        return -1 # 根据定义，空树高度为-1
```

```

        return max(tree_height(node.left), tree_height(node.right)) + 1

def count_leaves(node):
    if node is None:
        return 0
    if node.left is None and node.right is None:
        return 1
    return count_leaves(node.left) + count_leaves(node.right)

n = int(input()) # 读取节点数量
nodes = [TreeNode() for _ in range(n)]
has_parent = [False] * n # 用来标记节点是否有父节点

for i in range(n):
    left_index, right_index = map(int, input().split())
    if left_index != -1:
        nodes[i].left = nodes[left_index]
        has_parent[left_index] = True
    if right_index != -1:
        # print(right_index)
        nodes[i].right = nodes[right_index]
        has_parent[right_index] = True

# 寻找根节点，也就是没有父节点的节点
root_index = has_parent.index(False)
root = nodes[root_index]

# 计算高度和叶子节点数
height = tree_height(root)
leaves = count_leaves(root)

print(f"{height} {leaves}")

```

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：照讲义写的

代码

```

class TreeNode:
    def __init__(self, value): #类似字典
        self.value = value
        self.children = []

def parse_tree(s):
    stack = []
    node = None
    for char in s:

```

```
    if char.isalpha(): # 如果是字母, 创建新节点
        node = TreeNode(char)
        if stack: # 如果栈不为空, 把节点作为子节点加入到栈顶节点的子节点列表中
            stack[-1].children.append(node)
    elif char == '(': # 遇到左括号, 当前节点可能会有子节点
        if node:
            stack.append(node) # 把当前节点推入栈中
            node = None
    elif char == ')': # 遇到右括号, 子节点列表结束
        if stack:
            node = stack.pop() # 弹出当前节点
    return node # 根节点

def preorder(node):
    output = [node.value]
    for child in node.children:
        output.extend(preorder(child))
    return ''.join(output)

def postorder(node):
    output = []
    for child in node.children:
        output.extend(postorder(child))
    output.append(node.value)
    return ''.join(output)

# 主程序
def main():
    s = input().strip()
    s = ''.join(s.split()) # 去掉所有空白字符
    root = parse_tree(s) # 解析整棵树
    if root:
        print(preorder(root)) # 输出前序遍历序列
        print(postorder(root)) # 输出后序遍历序列
    else:
        print("input tree string error!")

if __name__ == "__main__":
    main()
```

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: 照讲义写的

代码

```

# 夏天明, 元培学院
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 == (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: 照讲义写的: 建立起表达式树, 按层次遍历表达式树的结果前后颠倒就得到队列表达式

代码

```

class TreeNode:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

```

```
def build_tree(postfix):
    stack = []
    for char in postfix:
        node = TreeNode(char)
        if char.isupper():
            node.right = stack.pop()
            node.left = stack.pop()
            stack.append(node)
    return stack[0]

def level_order_traversal(root):
    queue = [root]
    traversal = []
    while queue:
        node = queue.pop(0)
        traversal.append(node.value)
        if node.left:
            queue.append(node.left)
        if node.right:
            queue.append(node.right)
    return traversal

n = int(input().strip())
for _ in range(n):
    postfix = input().strip()
    root = build_tree(postfix)
    queue_expression = level_order_traversal(root)[::-1]
    print(''.join(queue_expression))
```

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：照讲义写的，后序遍历的最后一个元素是树的根节点。然后，在中序遍历序列中，根节点将左右子树分开。可以通过这种方法找到左右子树的中序遍历序列。然后，使用递归地处理左右子树来构建整个树。

代码

```
def build_tree(inorder, postorder):
    if not inorder or not postorder:
        return []

    root_val = postorder[-1]
    root_index = inorder.index(root_val)

    left_inorder = inorder[:root_index]
    right_inorder = inorder[root_index + 1:]

    left_postorder = postorder[:len(left_inorder)]
    right_postorder = postorder[len(left_inorder):-1]
```

```
    root = [root_val]
    root.extend(build_tree(left_inorder, left_postorder))
    root.extend(build_tree(right_inorder, right_postorder))

    return root

def main():
    inorder = input().strip()
    postorder = input().strip()
    preorder = build_tree(inorder, postorder)
    print(''.join(preorder))

if __name__ == "__main__":
    main()
```

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: 照讲义写的

代码

```
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

def build_tree(preorder, inorder):
    if not preorder or not inorder:
        return None
    root_value = preorder[0]
    root = TreeNode(root_value)
    root_index_inorder = inorder.index(root_value)
    root.left = build_tree(preorder[1:1+root_index_inorder],
                           inorder[:root_index_inorder])
    root.right = build_tree(preorder[1+root_index_inorder:],
                             inorder[root_index_inorder+1:])
    return root

def postorder_traversal(root):
    if root is None:
        return ''
    return postorder_traversal(root.left) + postorder_traversal(root.right) +
    root.value
```

```
while True:
    try:
        preorder = input().strip()
        inorder = input().strip()
        root = build_tree(preorder, inorder)
        print(postorder_traversal(root))
    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。==

反转切片的用法, `[::-1]`, 可以用来反转列表, 字符串等。

递归的思想, 可以用来解决树的问题。

这次作业难度上来了, 很多题都是不能完全靠自己想出来的, 要学习一些技巧, 比如根据后序序列建树, 根据前中序序列建树, 后序表达式建立队列等。