Assignment #5: "树"算：概念、表示、解析、遍历

Updated 2124 GMT+8 March 17, 2024

2024 spring, Complied by 周添 物理学院

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。

1. 题目

27638: 求二叉树的高度和叶子数目

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

代码

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


def calculate_depth(node):
    if node is None:
        return -1
    return max(calculate_depth(node.left), calculate_depth(node.right)) + 1


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

```
n = int(input())
nodes = [TreeNode() for _ in range(n)]
you_wu_fu_mu = [False] * n

for i in range(n):
    l, r = map(int, input().split())
    if l != -1:
        nodes[i].left = nodes[l]
        you_wu_fu_mu[l] = True
    if r != -1:
        nodes[i].right = nodes[r]
        you_wu_fu_mu[r] = True

father = nodes[you_wu_fu_mu.index(False)]
print(calculate_depth(father), count(father))
```

状态: Accepted

源代码

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


def calculate_depth(node):
    if node is None:
        return -1
    return max(calculate_depth(node.left), calculate_depth(node.right))


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


n = int(input())
nodes = [TreeNode() for _ in range(n)]
you_wu_fu_mu = [False] * n

for i in range(n):
    l, r = map(int, input().split())
    if l != -1:
        nodes[i].left = nodes[l]
        you_wu_fu_mu[l] = True
    if r != -1:
        nodes[i].right = nodes[r]
        you_wu_fu_mu[r] = True

father = nodes[you_wu_fu_mu.index(False)]
```

24729: 括号嵌套树

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

代码

```
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.children = []


def parse_nested_tree(s):
    stack = []
    root = None
    ls = len(s)
    for c in range(ls):
        char = s[c]
        if char.isalpha():
            if stack:
                node = TreeNode(char)
                stack[-1].children.append(node)
                if c<ls-1 and s[c+1] != ',' and s[c+1] != ')':
                    stack.append(node)
            else:
                root = TreeNode(char)
                stack.append(root)
        elif char == '(':
            continue
        elif char == ')':
            stack.pop()
    return root

def preorder_traversal(root):
    if not root:
        return
    print(root.value, end='')
    for child in root.children:
        preorder_traversal(child)
    return


def postorder_traversal(root):
    if not root:
        return
    for child in root.children:
        postorder_traversal(child)
    print(root.value, end='')
    return


input_str = input()
root = parse_nested_tree(input_str)
preorder_traversal(root)
print()
```

```
postorder_traversal(root)
```

状态: Accepted

源代码

```
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.children = []


def parse_nested_tree(s):
    stack = []
    root = None
    ls = len(s)
    for c in range(ls):
        char = s[c]
        if char.isalpha():
            if stack:
                node = TreeNode(char)
                stack[-1].children.append(node)
                if c<ls-1 and s[c+1] != ',' and s[c+1] != ')':
                    stack.append(node)
            else:
                root = TreeNode(char)
                stack.append(root)
        elif char == '(':
            continue
        elif char == ')':
            stack.pop()
    return root

def preorder_traversal(root):
    if not root:
        return
    print(root.value, end='')
    for child in root.children:
```

基本信息

#:	44363075
题目:	24729
提交人:	23n2300011538
内存:	3684kB
时间:	25ms
语言:	Python3
提交时间:	2024-03-23 17:00:17

02775: 文件结构“图”

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

代码

```
class TreeNode:
    def __init__(self, name):
        self.name = name
        self.f_sons = []
        self.d_sons = []


def print_tree(s, t):
    print('|     '*(t-1) + s.name)
    for d in s.d_sons:
        print_tree(d, t+1)
    s.f_sons.sort()
    for f in s.f_sons:
        print('|     '*(t-1) + f)


stack = [TreeNode('ROOT')]
```

```
v = 1
while True:
    a = input()
    if a == '#':
        break
    if a == '*':
        print(f'DATA SET {v}:')
        print_tree(stack[0], 1)
        print()
        v+=1
        stack = [TreeNode('ROOT')]
    else:
        if a[0] == 'f':
            stack[-1].f_sons.append(a)
        if a[0] == 'd':
            stack.append(TreeNode(a))
        if a[0] == ']':
            sp = stack.pop()
            stack[-1].d_sons.append(sp)
```

状态: Accepted

源代码

```
class TreeNode:
    def __init__(self, name):
        self.name = name
        self.f_sons = []
        self.d_sons = []


def print_tree(s, t):
    print('|     '*(t-1) + s.name)
    for d in s.d_sons:
        print_tree(d, t+1)
    s.f_sons.sort()
    for f in s.f_sons:
        print('|     '*(t-1) + f)


stack = [TreeNode('ROOT')]
v = 1
while True:
    a = input()
    if a == '#':
        break
    if a == '*':
        print(f'DATA SET {v}:')
        print_tree(stack[0], 1)
        print()
        v+=1
        stack = [TreeNode('ROOT')]
    else:
        if a[0] == 'f':
            stack[-1].f_sons.append(a)
```

基本信息

#:	44369969
题目:	02775
提交人:	23n2300011538
内存:	3660kB
时间:	24ms
语言:	Python3
提交时间:	2024-03-23 21:55:16

25140: 根据后序表达式建立队列表达式

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

代码

```python
def diff(character):
    if ord(character) < 91:
        return True  # 大写
    else:
        return False


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


def postfix_to_expression_tree(postfix):
    stack = []
    for token in postfix:
        if not diff(token):
            node = TreeNode(token)
            stack.append(node)
        else:
            right = stack.pop()
            left = stack.pop()
            node = TreeNode(token)
            node.left = left
            node.right = right
            stack.append(node)
    return stack[0]


def level_order_traversal(root):
    if not root:
        return []

    result = []
    queue = [root]

    while queue:
        current_level = []
        next_level = []

        for node in queue:
            current_level.append(node.value)
            if node.left:
                next_level.append(node.left)
            if node.right:
                next_level.append(node.right)

        result.append(current_level)
        queue = next_level

    print(''.join(reversed(list(''.join([''.join(item) for item in result])))))


n = int(input())
```

```
for i in range(n):
    m = input()
    root = postfix_to_expression_tree(m)
    level_order_traversal(root)
```

状态: Accepted

源代码

```
def diff(character):
    if ord(character) < 91:
        return True  # 大写
    else:
        return False


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


def postfix_to_expression_tree(postfix):
    stack = []
    for token in postfix:
        if not diff(token):
            node = TreeNode(token)
            stack.append(node)
        else:
            right = stack.pop()
            left = stack.pop()
            node = TreeNode(token)
            node.left = left
            node.right = right
            stack.append(node)
    return stack[0]


def level_order_traversal(root):
    if not root:
        return []

    result = []
    queue = [root]

    while queue:
        current_level = []
```

基本信息

#: 44371695
题目: 25140
提交人: 23n2300011538
内存: 5684kB
时间: 29ms
语言: Python3
提交时间: 2024-03-24 01:44:13

24750: 根据二叉树中后序序列建树

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

代码

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

def build_tree(inorder, postorder):
    if not inorder or not postorder:
        return None
```

```
    root_val = postorder.pop()
    root = TreeNode(root_val)

    mid = inorder.index(root_val)

    root.right = build_tree(inorder[mid+1:], postorder)
    root.left = build_tree(inorder[:mid], postorder)

    return root

def preorder_traversal(root):
    if not root:
        return []

    traversal = [root.val]
    traversal += preorder_traversal(root.left)
    traversal += preorder_traversal(root.right)

    return traversal


inorder = list(input())
postorder = list(input())


root = build_tree(inorder, postorder)


preorder = preorder_traversal(root)
print(''.join(preorder))
```

源代码

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

def build_tree(inorder, postorder):
    if not inorder or not postorder:
        return None

    root_val = postorder.pop()
    root = TreeNode(root_val)

    mid = inorder.index(root_val)

    root.right = build_tree(inorder[mid+1:], postorder)
    root.left = build_tree(inorder[:mid], postorder)

    return root

def preorder_traversal(root):
    if not root:
        return []

    traversal = [root.val]
    traversal += preorder_traversal(root.left)
    traversal += preorder_traversal(root.right)

    return traversal


inorder = list(input())
postorder = list(input())


root = build_tree(inorder, postorder)


preorder = preorder_traversal(root)
print(''.join(preorder))
```

22158: 根据二叉树前中序序列建树

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

代码

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

def build_tree(preorder, inorder):
```

```
    if not preorder or not inorder:
        return None

    root_val = preorder[0]
    root = TreeNode(root_val)

    root_idx = inorder.index(root_val)

    root.left = build_tree(preorder[1:1+root_idx], inorder[:root_idx])
    root.right = build_tree(preorder[1+root_idx:], inorder[root_idx+1:])

    return root

def preorder_to_postorder(preorder, inorder):
    root = build_tree(preorder, inorder)

    def postorder_traversal(node):
        if node is None:
            return []

        traversal = []
        traversal += postorder_traversal(node.left)
        traversal += postorder_traversal(node.right)
        traversal.append(node.val)

        return traversal

    return postorder_traversal(root)


while True:
    try:
        preorder = list(input())
        inorder = list(input())
        postorder = preorder_to_postorder(preorder, inorder)
        print(''.join(postorder))
    except EOFError:
        break
```

状态：Accepted

源代码

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

def build_tree(preorder, inorder):
    if not preorder or not inorder:
        return None

    root_val = preorder[0]
    root = TreeNode(root_val)

    root_idx = inorder.index(root_val)

    root.left = build_tree(preorder[1:1+root_idx], inorder[:root_idx])
    root.right = build_tree(preorder[1+root_idx:], inorder[root_idx+1:])

    return root

def preorder_to_postorder(preorder, inorder):
    root = build_tree(preorder, inorder)

    def postorder_traversal(node):
        if node is None:
            return []

        traversal = []
        traversal += postorder_traversal(node.left)
        traversal += postorder_traversal(node.right)
        traversal.append(node.val)

        return traversal
```

基本信息

#：44370911
题目：22158
提交人：23n2300011538
内存：3632kB
时间：24ms
语言：Python3
提交时间：2024-03-23 23:17:08

2. 学习总结和收获

体会到了树是一种递归形式，递归又常常用栈来实现；大概就是用递归思考，用栈实现

（暂时来说）用树做的题步骤就是“ 定义树 - 构建树（解析树） - 输出树（or树的特征）”

这星期事情比较多，没能直接上手做更多的题，但是欣赏了很多代码