Assignment #5: "树"算：概念、表示、解析、遍历

Updated 2124 GMT+8 March 17, 2024

2024 spring, Complied by 李佳霖，心理与认知科学学院

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

（请改为同学的操作系统、编程环境等）

操作系统：macOS Sonoma 14

Python编程环境：VSCode

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：需要注意的一个点在于需要知道谁是root，不然树的高度可能会出问题。另外一个就是空树的高度是-1。

代码

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

def tree_depth(node):
    if node is None:
        return -1 # 避免NoneType返回，即度为0的结点返回None值会报错

    left = tree_depth(node.left)
    right = tree_depth(node.right)

    return max(left, right)+ 1

def leaf_node(nodelist):
    counter = 0
    for i in range(len(nodelist)):
        if nodelist[i].left == None and nodelist[i].right == None:
            counter +=1

    return counter

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

for i in range(len(nodes)):
    if nodes[i].parent is None:
        root = nodes[i]

print(tree_depth(root), leaf_node(nodes))
```

代码运行截图 （至少包含有"Accepted"）

#44336310提交状态 查看 提交 统计 提问

状态: Accepted

源代码

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

基本信息

#: 44336310
题目: 27638
提交人: 李佳霖2000013713
内存: 3692kB
时间: 23ms
语言: Python3
提交时间: 2024-03-22 13:43:43

24729: 括号嵌套树

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

思路：看了题解。这里parse_tree函数来构建二叉树的方法值得学习。通过栈来存储当前结点所有的子结点。且由于数据的input本身已经存在顺序，所以也不需要区分左右子树，一股脑全部扔进list就好。在输出前序和后序表达式的时候，通过递归的方式将所有子结点extend到output的列表当中，前序和后序的唯一区别就是先扔进去还是后append进去。整体代码我个人认为非常优雅。

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"）

#44389640提交状态

查看 提交 统计 提问

状态: Accepted

源代码

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

def parse_tree(s):
    stack = []
```

基本信息

#: 44389640
题目: 24729
提交人: 李佳霖2000013713
内存: 3672kB
时间: 24ms
语言: Python3
提交时间: 2024-03-24 20:54:43

02775: 文件结构“图”

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

思路：同样看了题解。通过字典和递归的方式分别进行数据存储和输出。

代码

```
#
def print_structure(node,indent=0):
 #indent为缩进个数
  prefix='|   '*indent
  print(prefix+node['name'])
  for dir in node['dirs']:
   #若为目录继续递归
    print_structure(dir,indent+1) #对目录进行递归的方式输出，indent控制缩进
  for file in sorted(node['files']):
   #若为文件直接打印
    print(prefix+file)

dataset=1
datas=[]
temp=[]

#读取输入
while True:
  line=input()
  if line=='#':
    break
  if line=='*':
    datas.append(temp)
    temp=[]
  else:
    temp.append(line)

for data in datas:
  print(f'DATA SET {dataset}:')
  root={'name':'ROOT','dirs':[],'files':[]}
  stack=[root]
  #用栈实现后进先出
  for line in data:
    if line[0]=='d': #如果是目录，直接添加子目录到dirs的键中
      dir={'name':line,'dirs':[],'files':[]}
      stack[-1]['dirs'].append(dir)
```

```
            stack.append(dir)
        elif line[0]=='f': #如果是文件，添加到files的键中
            stack[-1]['files'].append(line)
        else: #某种结束符
            stack.pop()
    print_structure(root)
    if dataset<len(datas):
        print()
    dataset+=1
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44369671提交状态

查看 提交 统计 提问

状态: **Accepted**

源代码

```
def print_structure(node,indent=0):
        #indent为缩进个数
    prefix='|      '*indent
    print(prefix+node['name'])
    for dir in node['dirs']:
        #若为目录继续递归
        print_structure(dir,indent+1)
    for file in sorted(node['files']):
        #若为文件直接打印
        print(prefix+file)
```

基本信息

#:	44369671
题目:	02775
提交人:	李佳霖2000013713
内存:	3584kB
时间:	28ms
语言:	Python3
提交时间:	2024-03-23 21:35:18

25140: 根据后序表达式建立队列表达式

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

思路：非常综合的一道题，其中利用了树，栈和队列等多个结构。首先将后缀表达式转换成表达式树，然后通过栈将树中的元素从根结点到叶结点，从左子树到右子树的顺序添加进去，利用队列结构进行中转。考虑栈结构先进后出的特性将栈中的元素反转输出。

代码

```
#
from collections import deque

class Node:
    def __init__(self, x):
        self.value = x
```

```
        self.left = None
        self.right = None

def build_tree(postfix):
    stack = []
    for item in postfix:
        if item in op:
            node = Node(item)
            node.right = stack.pop()
            node.left = stack.pop()
        else:
            node = Node(item)
        stack.append(node)
    return stack[0]

# def dequeExp(root):
#     if root.right == None and root.left == None:
#         print(root.value)
#         return
#     dequeExp(root.right)
#     dequeExp(root.left)
#     print(root.value)
#     return

def dequeExp(root):
    if root is None:
        return

    stack = [] #用栈先进后出的特性，将树中的元素依次从根节点到叶结点依次存入
    queue = deque() #通过队列中转，先进先出
    queue.append(root)

    while queue:
        node = queue.popleft()
        stack.append(node)

        if node.left:
            queue.append(node.left)
        if node.right:
            queue.append(node.right)

    values = [node.value for node in stack[::-1]]
    print("".join(values))


num = 'abcdefghijklmnopqrstuvwxyz'
op = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

```
n = int(input())
for i in range(n):
    seq = list(input())
    root = build_tree(seq)
    dequeExp(root)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44370595提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque

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

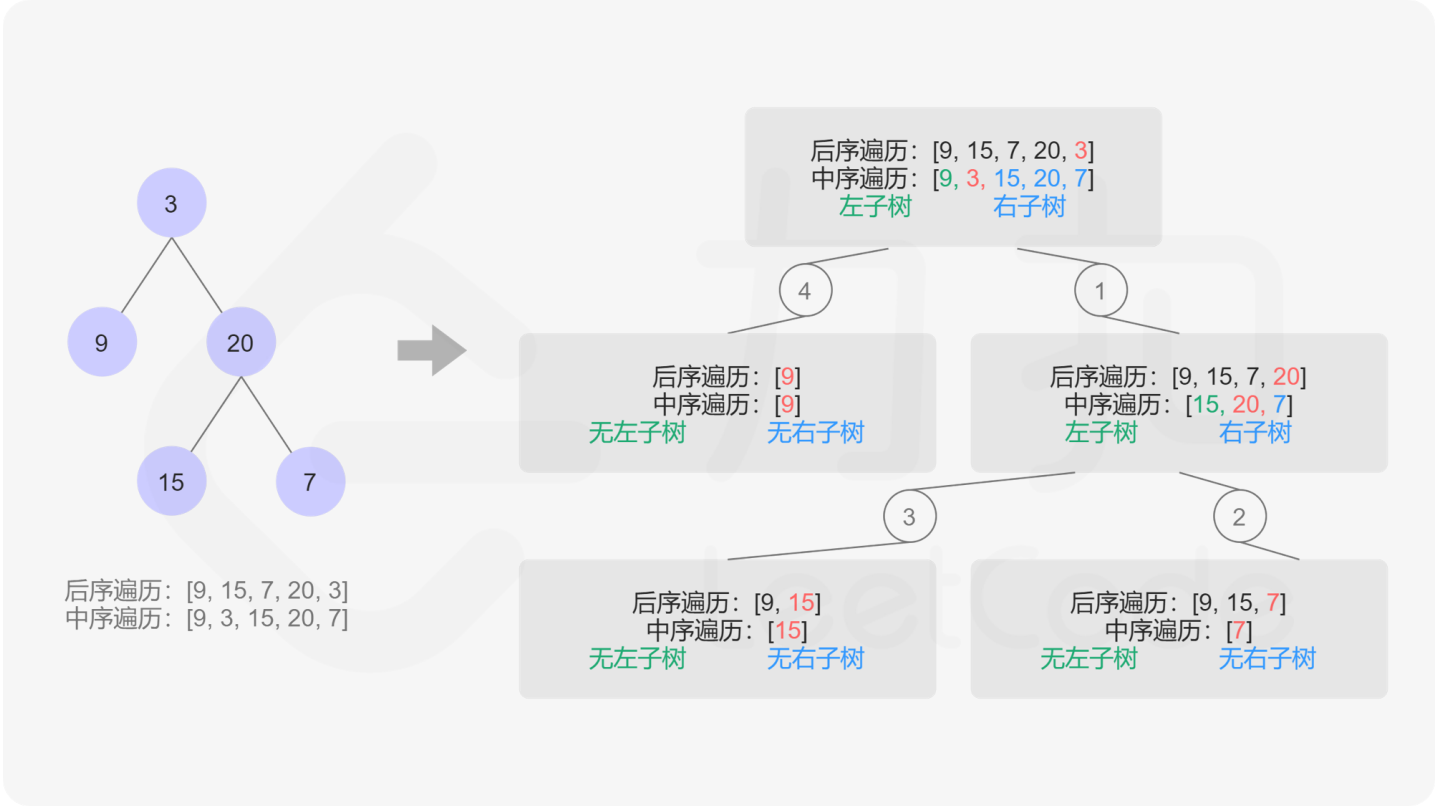
基本信息

#: 44370595
题目: 25140
提交人: 李佳霖2000013713
内存: 3684kB
时间: 29ms
语言: Python3
提交时间: 2024-03-23 22:45:24

24750: 根据二叉树中后序序列建树

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

思路：参考LeetCode官方题解。**后序遍历的数组最后一个元素代表的即为根节点**。知道这个性质后，我们可以利用已知的根节点信息在中序遍历的数组中找到根节点所在的下标，然后根据其将中序遍历的数组分成左右两部分，左边部分即左子树，右边部分为右子树，针对每个部分可以用同样的方法继续递归下去构造。



作者：力扣官方题解
链接：https://leetcode.cn/problems/construct-binary-tree-from-inorder-and-postorder-traversal/solutions/426738/cong-zhong-xu-yu-hou-xu-bian-li-xu-lie-gou-zao-14/
来源：力扣（LeetCode）

代码

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

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

    root_val = postorder.pop()
    root = TreeNode(root_val)
    root_index = inorder.index(root_val) #找到根结点在中序遍历中的位置

    #将序列切分成右子树和左子树
```

```
    root.right = buildTree(inorder[root_index+1:], postorder)
    root.left = buildTree(inorder[:root_index], postorder)

    return root

def preorderTraversal(root):
    if not root:
        return []
    stack = [root]
    result = []
    while stack:
        node = stack.pop()
        result.append(node.val)
        if node.right:
            stack.append(node.right)
        if node.left:
            stack.append(node.left)
    return result

inorder = input().strip()
postorder = input().strip()

# 构建二叉树
root = buildTree(list(inorder), list(postorder))

# 前序遍历并输出结果
result = preorderTraversal(root)
print(''.join(result))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44384620提交状态

查看 提交 统计 提问

状态: Accepted

源代码

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

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

基本信息

#:	44384620
题目:	24750
提交人:	李佳霖2000013713
内存:	3660kB
时间:	24ms
语言:	Python3
提交时间:	2024-03-24 18:02:37

22158: 根据二叉树前中序序列建树

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

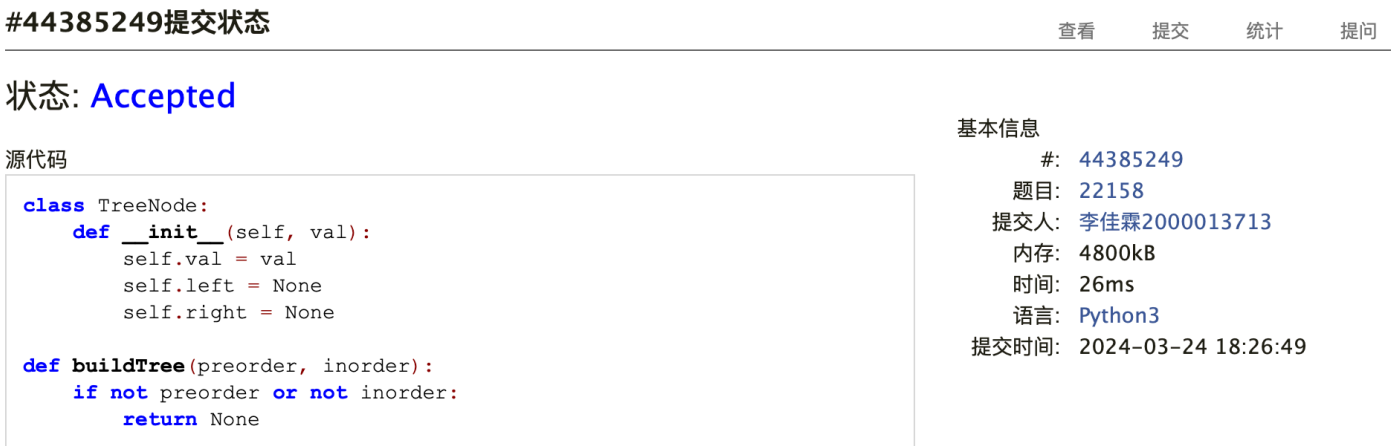
思路：有了上一题的经验，这题也是照猫画虎。前序遍历中的第一个元素是根结点，据此找到中序遍历中的左子树和右子树，然后分别建树。这里有一个顺序要求，就是先建左树后建右树。在输出后序遍历的结果时也是先遍历左树后遍历右树最后添加结点值。还有一个细节就是善用try...except句子

代码

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

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

    root_val = preorder.pop()
    root = TreeNode(root_val)
    root_index = inorder.index(root_val) #找到根结点在中序遍历中的位置

    #将序列切分成右子树和左子树
    root.left = buildTree(preorder, inorder[:root_index])
    root.right = buildTree(preorder, inorder[root_index+1:])

    return root

def postorderTraversal(root):
    result = []
    if root:
        result.extend(postorderTraversal(root.left))
        result.extend(postorderTraversal(root.right))
        result.append(root.val)

    return result

while True:
    try:
        preorder = input().strip()
        inorder = input().strip()
```

```
    # 构建二叉树
    root = buildTree(list(preorder)[::-1], list(inorder))

    # 前序遍历并输出结果
    result = postorderTraversal(root)
    print(''.join(result))
  except:
    break
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



#44385249提交状态

查看 提交 统计 提问

状态: Accepted

源代码

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

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

基本信息

#: 44385249
题目: 22158
提交人: 李佳霖2000013713
内存: 4800kB
时间: 26ms
语言: Python3
提交时间: 2024-03-24 18:26:49

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。

感觉这周作业难度比较大，但题目考查的都非常综合。分析自己主要做的不太顺的原因是还是不能够非常清晰的知道什么情况下需要用到什么样类型的数据结构。例如，第一次看括号嵌套树的时候并不会想到通过栈来处理输入的数据，虽然之前做过类似波兰表达式的题目。此外，做题的时候发现递归和数据结构的结合使用非常频繁，基本上每一题都有涉及。感觉之后还是需要多看一些好一点的代码，从中总结经验和精华。还学到了一些比较好的用法例如try...except语句，例如反转列表输出list[::-1]，以及extend列表，以及字典嵌套字典等等。