

Assignment #5: "树"算：概念、表示、解析、遍历

Updated 0002 GMT+8 March 18, 2024

2024 spring, Compiled by 钟明衡

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_NT x64 10.0.19045

Python编程环境：Visual Studio Code 1.76.1

C/C++编程环境：Visual Studio Code 1.76.1

1. 题目

27638: 求二叉树的高度和叶子数目

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

思路：

输入时建树，左右都是-1的就是叶子节点，而始终没出现在子节点中的那个就是根结点，从这个节点开始遍历树即可得到高度

代码

```
1 left, right = [], []
2 height, leaf = 0, 0
```

```
3 not_root = set()
4
5
6 def dfs(x, h):
7     global left, right, height
8     if left[x]+1:
9         dfs(left[x], h+1)
10    if right[x]+1:
11        dfs(right[x], h+1)
12    height = max(height, h)
13
14
15 n = int(input())
16 for i in range(n):
17     l, r = map(int, input().split())
18     left.append(l)
19     right.append(r)
20     not_root.add(l)
21     not_root.add(r)
22     if l == r == -1:
23         leaf += 1
24 for root in range(n):
25     if root not in not_root:
26         dfs(root, 0)
27 print('%d %d' % (height, leaf))
28
```

代码运行截图

状态: **Accepted**

源代码

```
left, right = [], []
height, leaf = 0, 0
not_root = set()

def dfs(x, h):
    global left, right, height
    if left[x]+1:
        dfs(left[x], h+1)
    if right[x]+1:
        dfs(right[x], h+1)
    height = max(height, h)

n = int(input())
for i in range(n):
    l, r = map(int, input().split())
    left.append(l)
    right.append(r)
    not_root.add(l)
    not_root.add(r)
    if l == r == -1:
        leaf += 1
for root in range(n):
    if root not in not_root:
        dfs(root, 0)
print('%d %d' % (height, leaf))
```

基本信息

#: 44280857

题目: 27638

提交人: 23n2300011505(12号娱乐选手)

内存: 3620kB

时间: 23ms

语言: Python3

提交时间: 2024-03-17 23:00:42

24729: 括号嵌套树

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

思路:

利用递归建树, 用一个count来记录括号层数, 当存在层数时就压入栈, 层数被清空就用之前的栈继续建树

输出结果直接遍历即可

代码

```
1 from collections import defaultdict
2 dic = defaultdict(lambda: [])
3
4
5 def build(root, s):
6     global dic
7     count = 0
8     new_root = new_leaf = ''
9     for i in range(len(s)):
10         if s[i] == ')':
11             count -= 1
12         if s[i] not in '(), ' and not count:
13             if new_leaf:
14                 build(new_root, new_leaf)
```

```
15         new_leaf = ''
16         new_root = s[i]
17         dic[root].append(s[i])
18         if count:
19             new_leaf += s[i]
20         if s[i] == '(':
21             count += 1
22         if new_leaf:
23             build(new_root, new_leaf)
24
25
26 def pre(root):
27     global dic
28     temp = ''
29     for leaf in dic[root]:
30         temp += pre(leaf)
31     return root+temp
32
33
34 def suf(root):
35     global dic
36     temp = ''
37     for leaf in dic[root]:
38         temp += suf(leaf)
39     return temp+root
40
41
42 s = input()
43 build(s[0], s[2:-1] if len(s) > 1 else '')
44 print(pre(s[0]))
45 print(suf(s[0]))
46
```

代码运行截图

状态: Accepted

源代码

```
from collections import defaultdict
dic = defaultdict(lambda: [])

def build(root, s):
    global dic
    count = 0
    new_root = new_leaf = ''
    for i in range(len(s)):
        if s[i] == ')':
            count -= 1
        if s[i] not in '()' and not count:
            if new_leaf:
                build(new_root, new_leaf)
                new_leaf = ''
            new_root = s[i]
            dic[root].append(s[i])
        if count:
            new_leaf += s[i]
        if s[i] == '(':
            count += 1
    if new_leaf:
        build(new_root, new_leaf)

def pre(root):
    global dic
    temp = ''
    for leaf in dic[root]:
        temp += pre(leaf)
    return root+temp

def suf(root):
    global dic
    temp = ''
    for leaf in dic[root]:
        temp += suf(leaf)
    return temp+root

s = input()
build(s[0], s[2:-1] if len(s) > 1 else '')
print(pre(s[0]))
print(suf(s[0]))
```

基本信息

#: 44280890

题目: 24729

提交人: 23n2300011505(12号娱乐选手)

内存: 3676kB

时间: 22ms

语言: Python3

提交时间: 2024-03-17 23:03:33

02775: 文件结构“图”

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

思路:

写了一个File类，默认名称为'ROOT'，当出现file就存储，dir就建一个新的类接收，直到']'结束

输出时，先输出原顺序的dir，要在每一行以前加上'| '，这个可以自动嵌套，然后输出排序了的file

代码

```
1 class File:
2     def __init__(self):
3         self.name = 'ROOT'
```

```

4         self.files = []
5         self.dirs = []
6
7     def __str__(self):
8         return '\n'.join([self.name]+'|'+s for d in self.dirs for s in
str(d).split('\n')]+sorted(self.files))
9
10    def build(self, parent, s):
11        if s[0] == 'f':
12            parent.files.append(s)
13        else:
14            dir = File()
15            dir.name = s
16            parent.dirs.append(dir)
17            while True:
18                s = input()
19                if s == ']':
20                    break
21                dir.build(dir, s)
22
23
24    x = 0
25    while True:
26        s = input()
27        if s == '#':
28            break
29        x += 1
30        root = File()
31        while s != '*':
32            root.build(root, s)
33            s = input()
34        print('DATA SET %d:' % x)
35        print(root, end='\n\n')
36

```

代码运行截图

状态: **Accepted**

源代码

```
class File:
    def __init__(self):
        self.name = 'ROOT'
        self.files = []
        self.dirs = []

    def __str__(self):
        return '\n'.join([self.name]+'|'+s for d in self.dirs for s

def build(self, parent, s):
    if s[0] == 'f':
        parent.files.append(s)
    else:
        dir = File()
        dir.name = s
        parent.dirs.append(dir)
        while True:
            s = input()
            if s == ']':
                break
            dir.build(dir, s)

x = 0
while True:
    s = input()
    if s == '#':
        break
    x += 1
    root = File()
    while s != '*':
        root.build(root, s)
        s = input()
    print('DATA SET %d:' % x)
    print(root, end='\n\n')
```

基本信息

#: 44280935
题目: 02775
提交人: 23n2300011505(12号娱乐选手)
内存: 3596kB
时间: 23ms
语言: Python3
提交时间: 2024-03-17 23:07:19

25140: 根据后序表达式建立队列表达式

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

思路:

采用了一个很特殊的方法: 叠层数

用 $r[h]$ 来表示当前层 h 所需的数值, 规则如下:

无论出现大写还是小写, 当前层的 r 值减一 (无论是大写还是小写, 都可以作为运算符的输入)。当出现大写字母就加一层, 并且给当前层的 r 值加上二 (因为一个运算符需要两个输入)。如果当前的 r 被清空了, 则下落直到有一层 r 不为0。

每输入一个新的字母, 就把它添加到当前所在层的队列末尾。最后输出结果, 层数从高到低, 按顺序将队列输出即可。

代码

```
1 from collections import defaultdict as D
2
```

```

3
4 for _ in range(int(input())):
5     s = input()
6     d = D(str)
7     r = D(int)
8     h = 0
9     for i in range(len(s)-1, -1, -1):
10         d[h] += s[i]
11         r[h] -= 1
12         if s[i] < 'a':
13             h += 1
14             r[h] += 2
15         while not r[h]:
16             h -= 1
17     a = ''
18     for i in range(max(d.keys()), -1, -1):
19         a += d[i]
20     print(a)
21

```

代码运行截图

#44280984提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: Accepted

源代码

```

from collections import defaultdict as D

for _ in range(int(input())):
    s = input()
    d = D(str)
    r = D(int)
    h = 0
    for i in range(len(s)-1, -1, -1):
        d[h] += s[i]
        r[h] -= 1
        if s[i] < 'a':
            h += 1
            r[h] += 2
        while not r[h]:
            h -= 1
    a = ''
    for i in range(max(d.keys()), -1, -1):
        a += d[i]
    print(a)

```

基本信息

#: 44280984

题目: 25140

提交人: 23n2300011505(12号娱乐选手)

内存: 3628kB

时间: 26ms

语言: Python3

提交时间: 2024-03-17 23:11:22

24750: 根据二叉树中后序序列建树

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

思路:

后序的最后一位必定为root，在中序中找到root位置，root以前就是左子节点中序遍历，以后就是右子节点中序遍历，这在后序的相同长度同样位置也是如此，将它们分别作为新的中序和后序，递归地求出前序即可（根左右）

代码

```
1 def pre(mid, suf):
2     if len(mid) > 1:
3         root = suf[-1]
4         n = mid.index(root)
5         left = pre(mid[:n], suf[:n])
6         right = pre(mid[n+1:], suf[n:-1])
7         return root+left+right
8     else:
9         return mid
10
11
12 mid = input()
13 suf = input()
14 print(pre(mid, suf))
15
```

代码运行截图

#44281174提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: Accepted

源代码

```
def pre(mid, suf):
    if len(mid) > 1:
        root = suf[-1]
        n = mid.index(root)
        left = pre(mid[:n], suf[:n])
        right = pre(mid[n+1:], suf[n:-1])
        return root+left+right
    else:
        return mid

mid = input()
suf = input()
print(pre(mid, suf))
```

基本信息

#: 44281174
题目: 24750
提交人: 23n2300011505(12号娱乐选手)
内存: 3616kB
时间: 21ms
语言: Python3
提交时间: 2024-03-17 23:26:24

22158: 根据二叉树前中序序列建树

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

思路:

和上题完全一致，前序第一个必为root，中序root前是left，后是right，前序则为root left right，用它们作为新的前序和中序，递归地求出后序即可（左 右 根）

代码

```
1 def suf(pre, mid):
2     if len(pre) > 1:
3         root = pre[0]
4         n = mid.index(root)
5         left = suf(pre[1:n+1], mid[:n])
6         right = suf(pre[n+1:], mid[n+1:])
7         return left+right+root
8     else:
9         return pre
10
11
12 while True:
13     try:
14         pre = input()
15         mid = input()
16     except EOFError:
17         break
18     print(suf(pre, mid))
19
```

代码运行截图

#44281212提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: Accepted

源代码

```
def suf(pre, mid):
    if len(pre) > 1:
        root = pre[0]
        n = mid.index(root)
        left = suf(pre[1:n+1], mid[:n])
        right = suf(pre[n+1:], mid[n+1:])
        return left+right+root
    else:
        return pre

while True:
    try:
        pre = input()
        mid = input()
    except EOFError:
        break
    print(suf(pre, mid))
```

基本信息

#: 44281212
题目: 22158
提交人: 23n2300011505(12号娱乐选手)
内存: 3548kB
时间: 24ms
语言: Python3
提交时间: 2024-03-17 23:29:25

2. 学习总结和收获

每天都在跟进每日选做，感觉刷这些题对我的帮助很大。

初次见到二叉树遍历问题的时候，有点摸不着头脑。实际上前序、中序、后续，实际上遍历顺序是一样的，都是先往左再往右，只是输出顺序不一样：

前序: $\text{root} \rightarrow \text{left} \rightarrow \text{right}$ 中序: $\text{left} \rightarrow \text{root} \rightarrow \text{right}$ 后序: $\text{left} \rightarrow \text{right} \rightarrow \text{root}$

了解了这一点, 二叉树遍历问题就不困难了。

二叉树通过将元素往左/右放, 和二分法的作用相同, 能将许多 n 复杂度变为 $\log n$, 二叉搜索树 ([OpenJudge - 05455:二叉搜索树的层次遍历](#)) 就是一个很好的例子。

OJ上大部分树题目其实不用类也能做, 用defaultdict存索引还是很方便的。

另外, 递归思想在很多方面都可以使用, 是一种省事而且优美的方法, 最近的题目几乎都要用到。此外我个人比较喜欢的如下:

[OpenJudge - 04147:汉诺塔问题\(Tower of Hanoi\)](#)

```
1 def H(n, a, b, c):
2     if n:
3         H(n-1, a, c, b)
4         print('%d:%s->%s' % (n, a, c))
5         H(n-1, b, a, c)
6
7
8 n, a, b, c = input().split()
9 H(int(n), a, b, c)
```

[OpenJudge - 01941:The Sierpinski Fractal](#)

```
1 def t(n):
2     if n == 1:
3         return [' /\', '/_\\']
4     T = t(n-1)
5     N = 2**(n-1)
6     return [' '*N+T[i] for i in range(N)]+[T[i]+' '*(N-1-i)+T[i] for i in
7         range(N)]
8
9 while True:
10     n = int(input())
11     if not n:
12         break
13     for s in t(n):
14         print(s)
15     print('')
```

[OpenJudge - 26573:康托集的图像表示](#)

```
1 def c(n):
2     return c(n-1)+'-'*(3**(n-1))+c(n-1) if n else ''
3
4
5 print(c(int(input())))
```

我对此的理解是，规定一个base case，然后告诉计算机大概要做什么，让它自己用同一套逻辑去算就好了。包括写类的时候，指针指向同样是这个类的元素，也是一种递归想法。举个例子，建树的时候，把左右子节点都看成一棵新的树，大概就是这么个想法。这种思路在各种场景下都有不错的表现。

写[OpenJudge - 25140:根据后序表达式建立表达式树](#)的时候想到了一个奇怪的方法，见上面。