Assignment #9: 图论：遍历，及 树算

Updated 1739 GMT+8 Apr 14, 2024

2024 spring, Complied by 王申睿——物理学院

说明:

1）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 typorahttps://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。

2）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。

3）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

04081: 树的转换

http://cs101.openjudge.cn/dsapre/04081/

思路:

代码

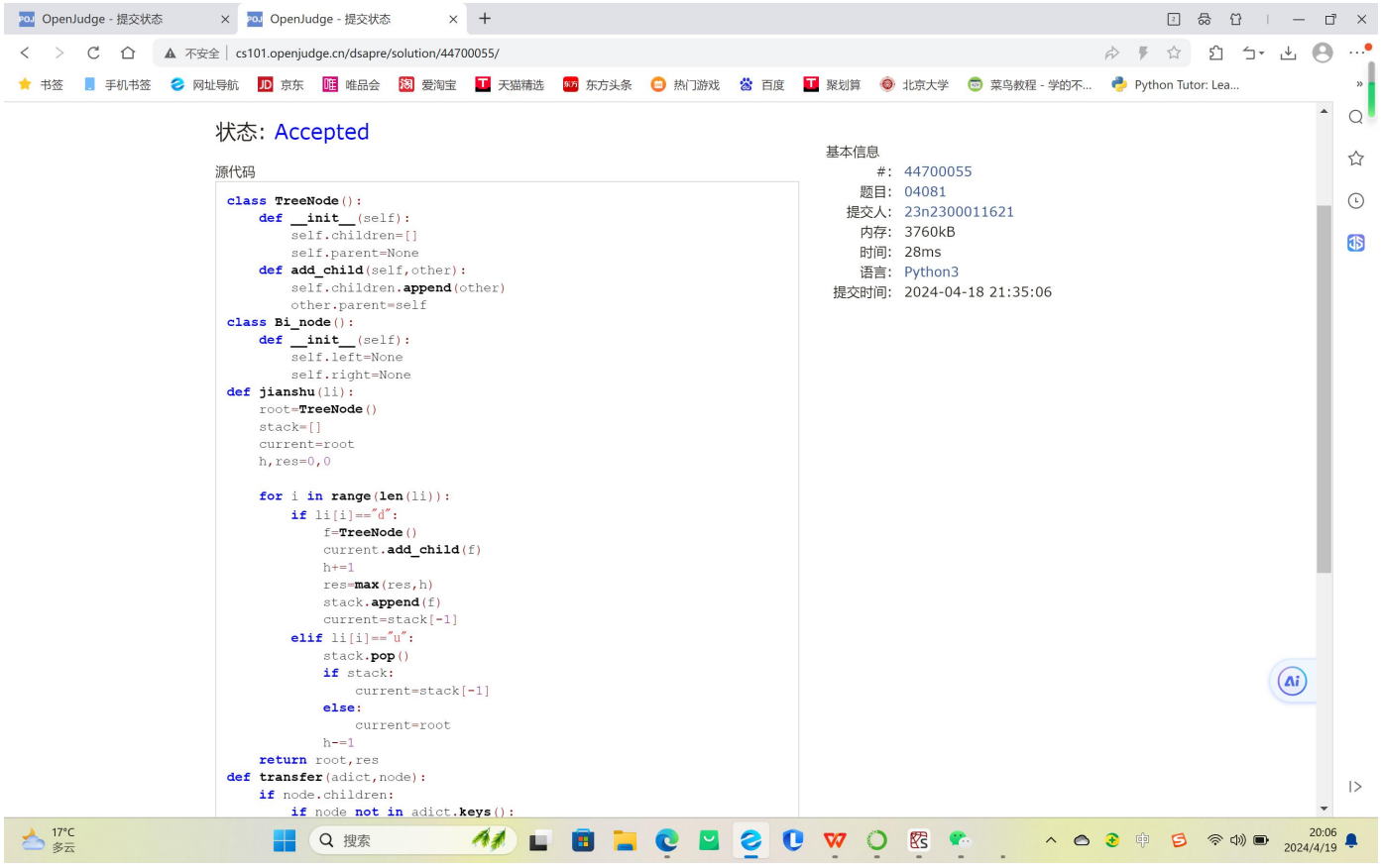
```
class TreeNode():

    def __init__(self):
```

```
        self.children=[]

        self.parent=None

    def add_child(self,other):

        self.children.append(other)

        other.parent=self

class Bi_node():

    def __init__(self):

        self.left=None

        self.right=None

def jianshu(li):

    root=TreeNode()

    stack=[]

    current=root

    h,res=0,0
```

```
for i in range(len(li)):
    if li[i]=="d":
        f=TreeNode()
        current.add_child(f)
        h+=1
        res=max(res,h)
        stack.append(f)
        current=stack[-1]
    elif li[i]=="u":
        stack.pop()
        if stack:
            current=stack[-1]
        else:
```

```
            current=root

        h-=1

    return root,res

def transfer(adict,node):

    if node.children:

        if node not in adict.keys():

            adict[node]=Bi_node()

        for child in node.children:

            if child not in adict.keys():

                adict[child]=Bi_node()

        adict[node].left=adict[node.children[0]]

        for i in range(len(node.children)-1):

            adict[node.children[i]].right=adict[node.children[i+1]]

        for i in range(len(node.children)):
```

```
            transfer(adict,node.children[i])

def height2(x):

    if x is None:

        return -1

    elif x.left is None and x.right is None:

        return 0

    else:

        return max(height2(x.left),height2(x.right))+1

dfs=list(input())

root=jianshu(dfs)[0]




ercha={}

transfer(ercha,root)

print(str(jianshu(dfs)[1])+" => "+str(height2(ercha[root])))
```


OpenJudge - 提交状态
OpenJudge - 提交状态
不安全 | cs101.openjudge.cn/dsapre/solution/44700055/
书签 手机书签 网址导航 京东 唯品会 爱淘宝 天猫精选 东方头条 热门游戏 百度 聚划算 北京大学 菜鸟教程 - 学的不... Python Tutor: Lea...
状态: Accepted
源代码
class TreeNode():
def __init__(self):
self.children=[]
self.parent=None
def add_child(self,other):
self.children.append(other)
other.parent=self
class Bi_node():
def __init__(self):
self.left=None
self.right=None
def jianshu(li):
root=TreeNode()
stack=[]
current=root
h,res=0,0
for i in range(len(li)):
if li[i]=="d":
f=TreeNode()
current.add_child(f)
h+=1
res=max(res,h)
stack.append(f)
current=stack[-1]
elif li[i]=="u":
stack.pop()
if stack:
current=stack[-1]
else:
current=root
h-=1
return root,res
def transfer(adict,node):
if node.children:
if node not in adict.keys():
基本信息
#: 44700055
题目: 04081
提交人: 23n2300011621
内存: 3760kB
时间: 28ms
语言: Python3
提交时间: 2024-04-18 21:35:06
17°C 多云
搜索
20:06 2024/4/19

08581: 扩展二叉树

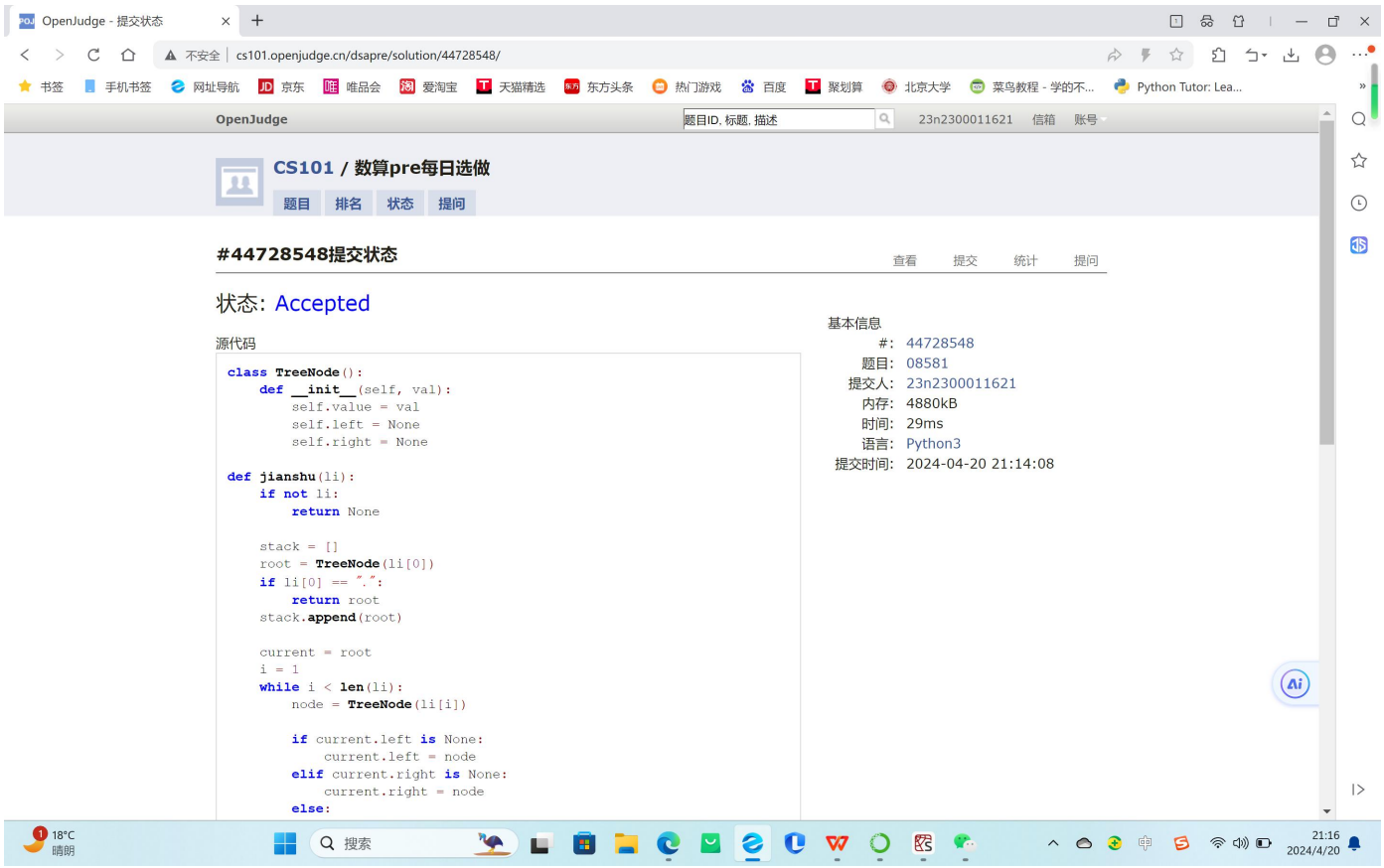
http://cs101.openjudge.cn/dsapre/08581/

思路:

代码

```
class TreeNode():
    def __init__(self, val):
        self.value = val
        self.left = None
        self.right = None
def jianshu(li):
    if not li:
        return None

    stack = []
    root = TreeNode(li[0])
    if li[0] == ".":
        return root
    stack.append(root)

    current = root
    i = 1
    while i < len(li):
        node = TreeNode(li[i])

        if current.left is None:
            current.left = node
        elif current.right is None:
            current.right = node
        else:
            while stack and current.left and current.right:
                current = stack.pop()

            if current.right is None:
                current.right = node

        if li[i] != ".":
            stack.append(node)
            current = node
        i += 1

    return root
def zhongxu(node):
    if not node:
        return ""
    res = ""
```

```
    if node.left:
        res += zhongxu(node.left)
    res += node.value
    if node.right:
        res += zhongxu(node.right)
    return res
def houxu(node):
    if not node:
        return ""
    res = ""
    if node.left:
        res += houxu(node.left)
    if node.right:
        res += houxu(node.right)
    res += node.value
    return res

alist = list(input().strip())
root = jianshu(alist)
ans1 = zhongxu(root)
ans2 = houxu(root)
ans1_b = ans1.replace('.', '')
ans2_b = ans2.replace('.', '')print(ans1_b)print(ans2_b)
```



22067: 快速堆猪

http://cs101.openjudge.cn/practice/22067/

思路:

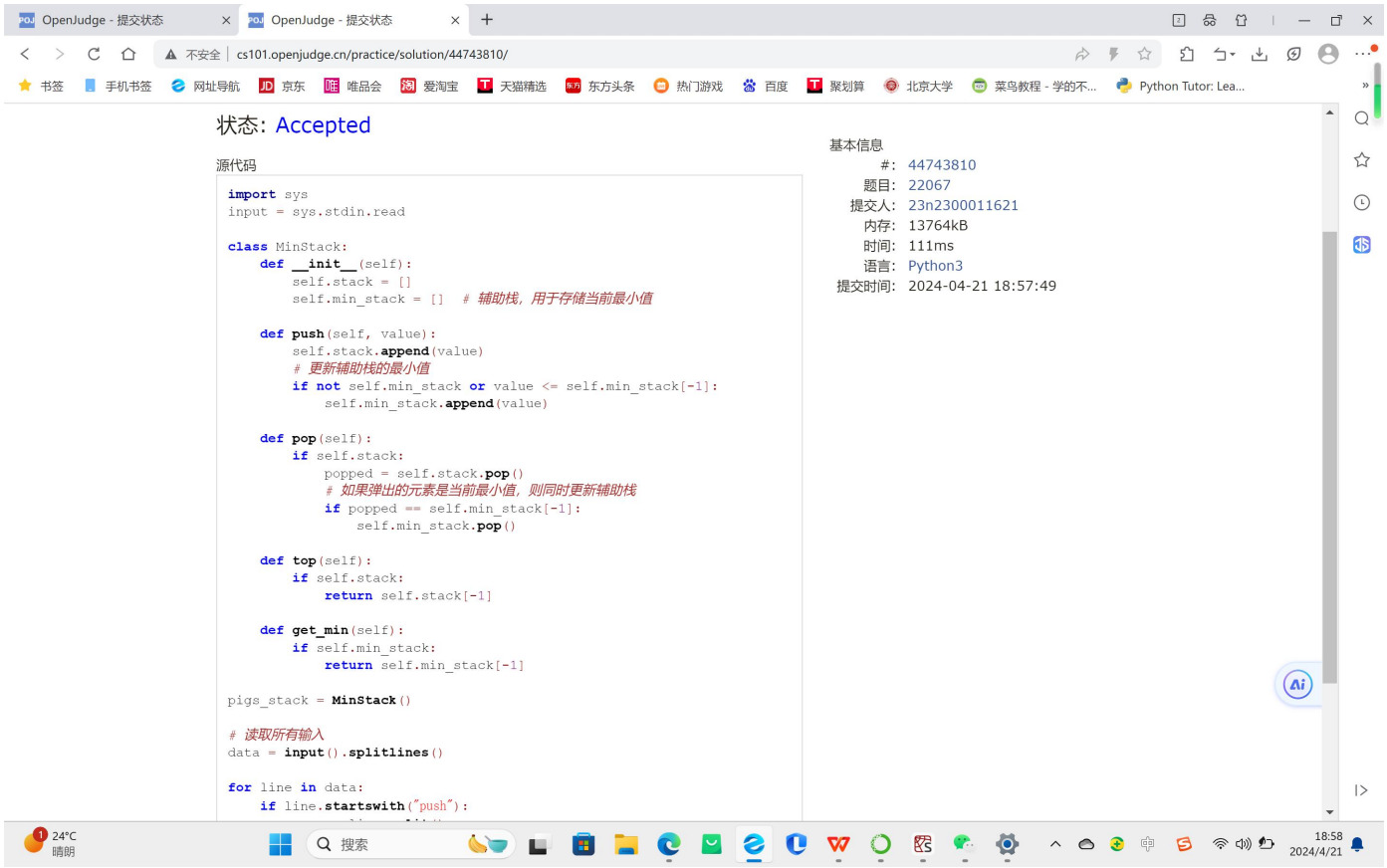
代码

```
import sys

input = sys.stdin.read




class MinStack:

    def __init__(self):

        self.stack = []

        self.min_stack = []  # 辅助栈，用于存储当前最小值



    def push(self, value):

        self.stack.append(value)
```

```
    # 更新辅助栈的最小值
    if not self.min_stack or value <= self.min_stack[-1]:
        self.min_stack.append(value)

def pop(self):
    if self.stack:
        popped = self.stack.pop()
        # 如果弹出的元素是当前最小值，则同时更新辅助栈
        if popped == self.min_stack[-1]:
            self.min_stack.pop()

def top(self):
    if self.stack:
        return self.stack[-1]
```

```
    def get_min(self):
        if self.min_stack:
            return self.min_stack[-1]

pigs_stack = MinStack()

# 读取所有输入
data = input().splitlines()

for line in data:
    if line.startswith("push"):
        _, n = line.split()
        weight = int(n)
```

```
        pigs_stack.push(weight)

    elif line == "pop":

        pigs_stack.pop()

    elif line == "min":

        min_weight = pigs_stack.get_min()

        if min_weight is not None:

            print(min_weight)
```



04123: 马走日

dfs, http://cs101.openjudge.cn/practice/04123

思路：（已参考题解）

```
maxn = 10;sx = [-2,-1,1,2, 2, 1,-1,-2]sy = [ 1, 2,2,1,-1,-2,-2,-1]
ans = 0;
 def Dfs(dep: int, x: int, y: int):
    #是否已经全部走完
    if n*m == dep:
        global ans
        ans += 1
        return

    #对于每个可以走的点
    for r in range(8):
        s = x + sx[r]
        t = y + sy[r]
        if chess[s][t]==False and 0<=s<n and 0<=t<m :
            chess[s][t]=True
            Dfs(dep+1, s, t)
            chess[s][t] = False; #回溯

for _ in range(int(input())):
    n,m,x,y = map(int, input().split())
    chess = [[False]*maxn for _ in range(maxn)]  #False表示没有走过
    ans = 0
    chess[x][y] = True
    Dfs(1, x, y)
    print(ans)
```

代码

```
#

```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

28046: 词梯

bfs，http://cs101.openjudge.cn/practice/28046/

思路：

代码

```
#

```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

28050: 骑士周游

dfs, http://cs101.openjudge.cn/practice/28050/

思路:

代码

```
#

```

。

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

对两种建树的方法做个总结，提醒自己：一种是分冶法，如果能够确切地知道根节点的位置和子树的范围，则可以以子树的范围对列表切片，对每个切片递归调用函数；另一种是设栈法，如果表达式体现了明显的从属关系，则可以考虑用栈来缓存层级关系，其中当前父节点一般是stack[-1]。

自我批评！又踩了重复建节点的坑，看了PythonTutor才反应过来。这一点应该写进cheatsheet里面作为警示。

快速堆猪看似是用堆，但是用了你就掉进了陷阱，因为有序性无法维护。正确的做法是双栈，也是再一次体会到了栈的强大。（本周还在期中季，实在是精疲力尽，问题代码只能依赖GPT纠正总体思路。）