Assignment #9: 图论：遍历，及 树算

Updated 1739 GMT+8 Apr 14, 2024

2024 spring, Complied by 尹柚鑫 光华管理学院 2100015878

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 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。

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：Win11

Python编程环境：jupter notebook

1. 题目

04081: 树的转换

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

耗时：40mins

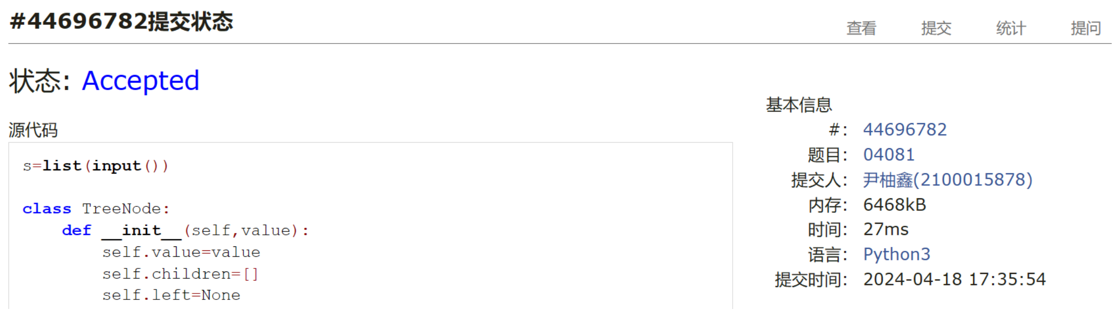
思路：先根据DFS的结果建立树，算一下高度，再把建立的树转化为左儿子右兄弟的二叉树，再算一下高度

代码

```python
#
s=list(input())

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

def buildnormaltree(s):
    index=1
    root=TreeNode(0)
    node=root
    stack=[node]
    for i in s:
        if i =='d':
            newnode=TreeNode(index)
            node.children.append(newnode)
            stack.append(newnode)
            node=newnode
            index+=1
        elif i=='u':
            stack.pop()
            parent=stack[-1]
            node=parent
    return root

firsttree=buildnormaltree(s)

def getlengthnormaltree(root):
    if root.children==[]:
        return 0
    allheight=[getlengthnormaltree(i) for i in root.children]
    return max(allheight)+1

firstheight=getlengthnormaltree(firsttree)

def buildsecondTree(root):
    stack=[root]

    while stack:
        node=stack.pop()
        if len(node.children)==0:
            pass
        elif len(node.children)==1:
            node.left=node.children.pop(0)
            stack.append(node.left)
        else:
            node.left=node.children.pop(0)
            stack.append(node.left)
            firstnode=node
            node=node.left
            while firstnode.children:
                node.right=firstnode.children.pop(0)
                stack.append(node.right)
                node=node.right
    return root

secondtree=buildsecondTree(firsttree)

def getlengthsecondtree(root):
    if root==None:
        return 0
    allheight=[getlengthsecondtree(i) for i in (root.left,root.right)]
    return max(allheight)+1

secondheight=getlengthsecondtree(secondtree)-1
```

```
print(f"{firstheight} => {secondheight}")
```

代码运行截图 ==（至少包含有"Accepted"）==


#44696782提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
s=list(input())

class TreeNode:
    def __init__(self,value):
        self.value=value
        self.children=[]
        self.left=None
```

基本信息

#:	44696782
题目:	04081
提交人:	尹柚鑫(2100015878)
内存:	6468kB
时间:	27ms
语言:	Python3
提交时间:	2024-04-18 17:35:54

08581: 扩展二叉树

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

耗时：30mins

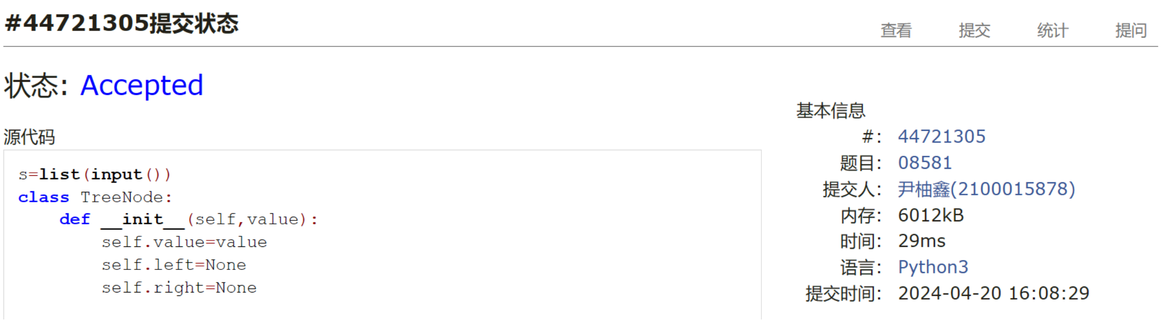
思路：可以直接建一个包括点的树，然后遍历，输出的时候不输出点就行

代码

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


def buildTree(s):
    if s:
        root=TreeNode(s[0])
        node=root
        stack=[root]

        for i in s[1:]:
            if i!='.':
                if node.left==None:
                    node.left=TreeNode(i)
                    stack.append(node.left)
                    node=node.left
                elif node.right==None:
                    node.right=TreeNode(i)
                    stack.append(node.right)
                    node=node.right
                else:
                    while node.left!=None and node.right!=None:
                        stack.pop()
```

```python
                        node=stack[-1]
                    node.right=TreeNode(i)
                    stack.append(node.right)
                    node=node.right
            else:
                if node.left==None:
                    node.left=TreeNode('.')
                elif node.right==None:
                    node.right=TreeNode('.')
                else:
                    while node.left!=None and node.right!=None:
                        stack.pop()
                        node=stack[-1]
                    node.right=TreeNode(i)

    return root

root=buildTree(s)

list_in=[]
list_after=[]
def inorder(root):
    if root is None:
        return
    inorder(root.left)
    list_in.append(root.value)
    inorder(root.right)

def afterorder(root):
    if root is None:
        return
    afterorder(root.left)
    afterorder(root.right)
    list_after.append(root.value)

inorder(root)
afterorder(root)

print(''.join([i for i in list_in if i !='.']))
print(''.join([i for i in list_after if i !='.']))
```

代码运行截图 ==（至少包含有"Accepted"）==



#44721305提交状态

查看 提交 统计 提问

状态: Accepted

源代码

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

基本信息

#:	44721305
题目:	08581
提交人:	尹柚鑫(2100015878)
内存:	6012kB
时间:	29ms
语言:	Python3
提交时间:	2024-04-20 16:08:29

22067: 快速堆猪

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

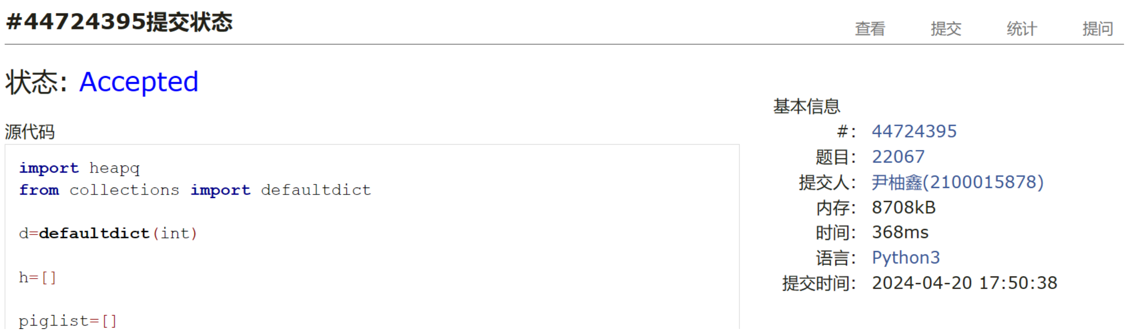
耗时：20mins

思路：用堆实现输出最小值，用字典实现对猪的删除，而不是真的删除

代码

```python
#
import heapq
from collections import defaultdict

d=defaultdict(int)

h=[]

piglist=[]


while True:
    try:
        s=input().split()
        if len(s)==0:
            break
        if s[0]=='pop':
            if len(piglist)>0:
                pig=piglist.pop()
                d[pig]-=1

        elif s[0]=='min':
            if len(piglist)>0:
                while d[h[0]]==0:
                    heapq.heappop(h)
                print(h[0])

        elif s[0]=='push':
            piglist.append(int(s[1]))
            heapq.heappush(h,int(s[1]))
            d[int(s[1])]+=1
    except EOFError:
        break
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==


#44724395提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import heapq
from collections import defaultdict

d=defaultdict(int)

h=[]

piglist=[]
```

基本信息

#:	44724395
题目:	22067
提交人:	尹柚鑫(2100015878)
内存:	8708kB
时间:	368ms
语言:	Python3
提交时间:	2024-04-20 17:50:38

04123: 马走日

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

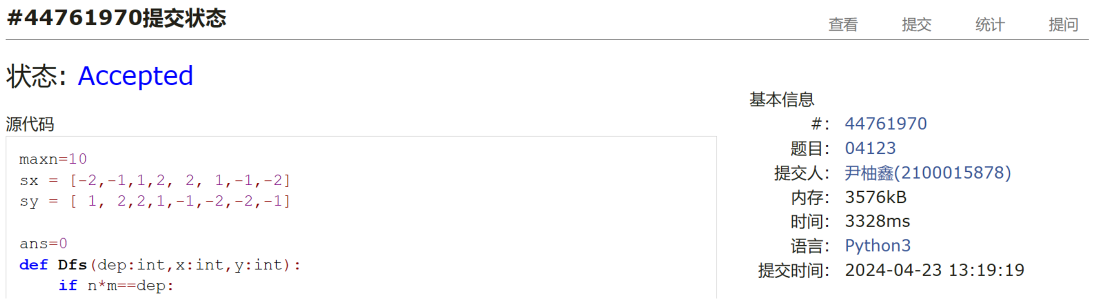
耗时：30mins

思路：DFS，从一个点出发，递归遍历每一个合法下一步，判断深度是否到达要求，同时要注意回溯

代码

```
#
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):
    if n*m==dep:
        global ans
        ans+=1
        return
    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

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)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==


#44761970提交状态

状态: Accepted

源代码

```
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):
    if n*m==dep:
```

基本信息

#:	44761970
题目:	04123
提交人:	尹柚鑫(2100015878)
内存:	3576kB
时间:	3328ms
语言:	Python3
提交时间:	2024-04-23 13:19:19

28046: 词梯

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

耗时：30mins

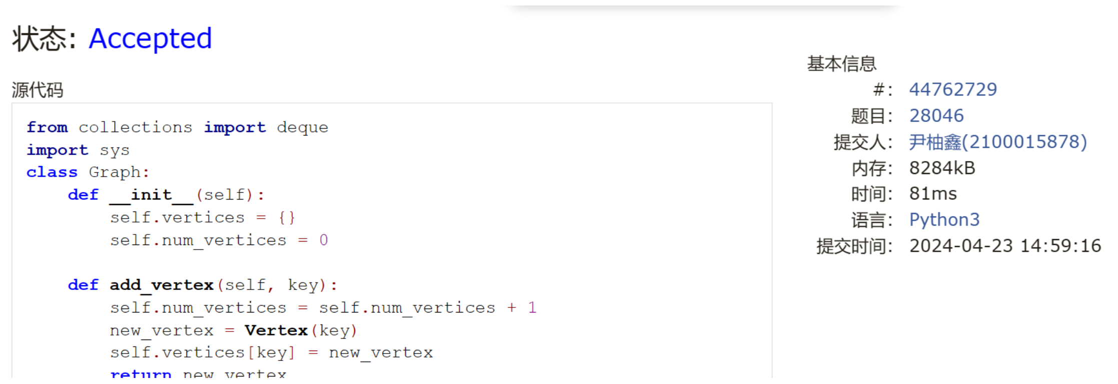
思路：词桶-建图-bfs-回溯

代码

```python
#
from collections import deque
import sys
class Graph:
    def __init__(self):
        self.vertices = {}
        self.num_vertices = 0

    def add_vertex(self, key):
        self.num_vertices = self.num_vertices + 1
        new_vertex = Vertex(key)
        self.vertices[key] = new_vertex
        return new_vertex

    def get_vertex(self, n):
        if n in self.vertices:
            return self.vertices[n]
        else:
            return None

    def __len__(self):
        return self.num_vertices

    def __contains__(self, n):
        return n in self.vertices

    def add_edge(self, f, t, cost=0):
        if f not in self.vertices:
            nv = self.add_vertex(f)
        if t not in self.vertices:
            nv = self.add_vertex(t)
        self.vertices[f].add_neighbor(self.vertices[t], cost)

    def get_vertices(self):
        return list(self.vertices.keys())

    def __iter__(self):
        return iter(self.vertices.values())


class Vertex:
    def __init__(self, num):
        self.key = num
        self.connectedTo = {}
        self.color = 'white'
        self.distance = sys.maxsize
        self.previous = None
        self.disc = 0
        self.fin = 0

    def add_neighbor(self, nbr, weight=0):
        self.connectedTo[nbr] = weight

    # def __lt__(self,o):
    #     return self.id < o.id

    # def setDiscovery(self, dtime):
    #     self.disc = dtime
    #
    # def setFinish(self, ftime):
    #     self.fin = ftime
    #
    # def getFinish(self):
    #     return self.fin
    #
    # def getDiscovery(self):
    #     return self.disc

    def get_neighbors(self):
        return self.connectedTo.keys()

    # def getWeight(self, nbr):
    #     return self.connectedTo[nbr]

    # def __str__(self):
    #     return str(self.key) + ":color " + self.color + ":disc " +
str(self.disc) + ":fin " + str(
    #         self.fin) + ":dist " + str(self.distance) + ":pred \n\t[" +
str(self.previous) + "]\n"




from collections import defaultdict
wordlist=[]
for _ in range(int(input())):
    wordlist.append(input())

a,b=input().split()
```

```python
buc_dict=defaultdict(list)
for i in wordlist:
    for j in range(4):
        name=i[0:j]+'_'+i[j+1:]
        buc_dict[name].append(i)

citi_graph=Graph()
for i in buc_dict.values():
    for x1 in i:
        for x2 in i:
            if x1!=x2:
                citi_graph.add_edge(x1,x2)
def citi(fromword):
    vert_deque=deque()
    fromword.distance=0
    fromword.previous=None
    vert_deque.append(fromword)

    while len(vert_deque)>0:
        current=vert_deque.popleft()
        for neighbor in current.get_neighbors():
            if neighbor.color=='white':
                neighbor.color='gray'
                neighbor.distance=current.distance+1
                neighbor.previous=current
                vert_deque.append(neighbor)
        current.color="black"

def traverse(toword):
    ans=[]
    current=toword
    while (current.previous):
        ans.append(current.key)
        current=current.previous
    ans.append(current.key)
    return ans

if a in citi_graph.get_vertices() and b in citi_graph.get_vertices():

    citi(citi_graph.get_vertex(a))
    ans=traverse(citi_graph.get_vertex(b))
    if len(ans)>1:
        for i in ans[::-1]:
            print(i,end=' ')
    else:
        print('NO')
else:
    print('NO')
```

代码运行截图 == （AC代码截图，至少包含有"Accepted"） ==


状态: Accepted

源代码

```
from collections import deque
import sys
class Graph:
    def __init__(self):
        self.vertices = {}
        self.num_vertices = 0

    def add_vertex(self, key):
        self.num_vertices = self.num_vertices + 1
        new_vertex = Vertex(key)
        self.vertices[key] = new_vertex
        return new_vertex
```

基本信息

#:	44762729
题目:	28046
提交人:	尹柚鑫(2100015878)
内存:	8284kB
时间:	81ms
语言:	Python3
提交时间:	2024-04-23 14:59:16

28050: 骑士周游

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

耗时：30mins

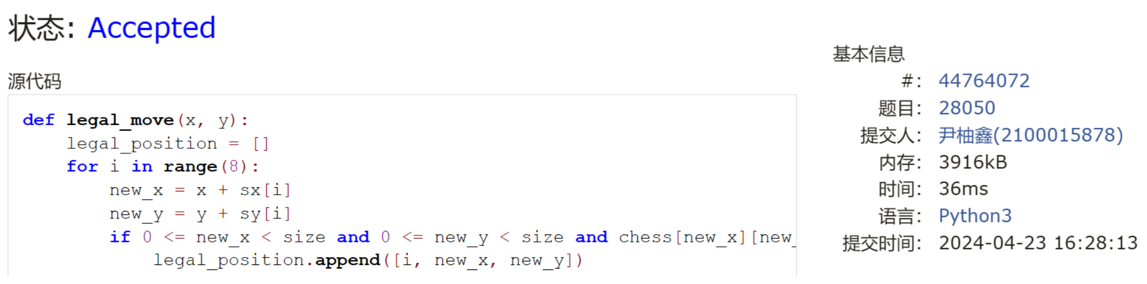
思路：能找到一个可行路径即可，需要使用教材上的选择靠边点的想法优化

代码

```
#
def legal_move(x, y):
    legal_position = []
    for i in range(8):
        new_x = x + sx[i]
        new_y = y + sy[i]
        if 0 <= new_x < size and 0 <= new_y < size and chess[new_x][new_y] ==
False:
            legal_position.append([i, new_x, new_y])
    for i in range(len(legal_position)):
        num = 0
        for j in range(8):
            new_x = legal_position[i][1] + sx[j]
            new_y = legal_position[i][2] + sy[j]
            if 0 <= new_x < size and 0 <= new_y < size and chess[new_x][new_y] ==
False:
                num += 1
        legal_position[i][0] = num
    legal_position.sort(key=lambda x: x[0])
    return legal_position

def dfs(dep, x, y):
    if dep == size**2:
        return True
    legal_position = legal_move(x, y)
    for i in legal_position:
        chess[i[1]][i[2]] = True
        if dfs(dep + 1, i[1], i[2]):
            return True
        chess[i[1]][i[2]] = False  # 回溯
    return False
```

```python
size = int(input())
ini_x, ini_y = map(int, input().split())
chess = [[False]*size for _ in range(size)]
chess[ini_x][ini_y] = True

sx = [-1, -2, -2, -1, 1, 2, 2, 1]
sy = [-2, -1, 1, 2, 2, 1, -1, -2]

if dfs(1, ini_x, ini_y):
    print('success')
else:
    print('fail')
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



状态: Accepted

源代码

```python
def legal_move(x, y):
    legal_position = []
    for i in range(8):
        new_x = x + sx[i]
        new_y = y + sy[i]
        if 0 <= new_x < size and 0 <= new_y < size and chess[new_x][new_
            legal_position.append([i, new_x, new_y])
```

基本信息

#:	44764072
题目:	28050
提交人:	尹柚鑫(2100015878)
内存:	3916kB
时间:	36ms
语言:	Python3
提交时间:	2024-04-23 16:28:13

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。==

1.BFS的实现方式，和树的层次遍历实现方式很像，都是用队列。想想也合理，树的层次遍历完全就是一个BFS，只不过BFS需要加一个回溯的步骤

2.这次的作业其实并没有很难，主要是对DFS的写法还不太熟练，导致用时较长