Assignment #8: 图论：概念、遍历，及 树算

Updated 1919 GMT+8 Apr 8, 2024

2024 spring, Complied by 余汶青 生命科学学院

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 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

操作系统：版本　Windows 11 家庭中文版
版本　22H2
安装日期 2023/7/18
操作系统版本 22621.2283
序列号　5CD323PJKL
体验　Windows Feature Experience Pack 1000.22662.1000.0

Python编程环境：* Spyder version: 5.4.3 (conda)

- Python version: 3.11.4 64-bit
- Qt version: 5.15.2
- PyQt5 version: 5.15.7
- Operating System: Windows 10

1. 题目

19943: 图的拉普拉斯矩阵

matrices, http://cs101.openjudge.cn/practice/19943/

请定义Vertex类，Graph类，然后实现

思路：

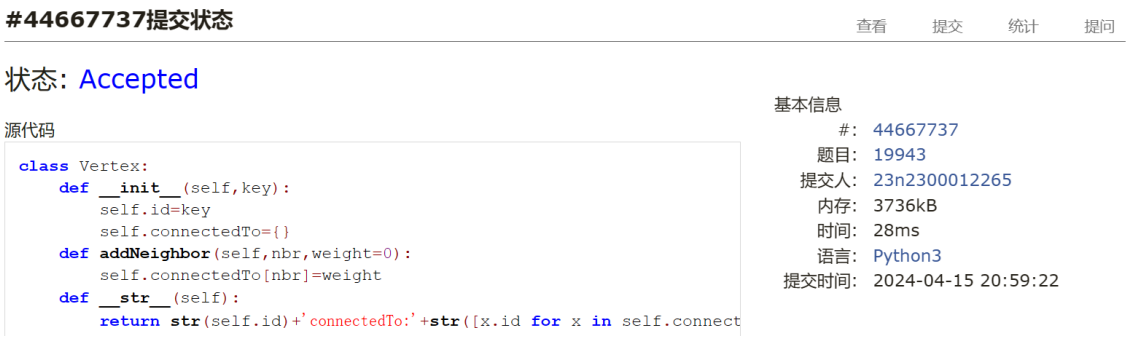
代码

```
class Vertex:
```

```
    def __init__(self,key):
        self.id=key
        self.connectedTo={}
    def addNeighbor(self,nbr,weight=0):
        self.connectedTo[nbr]=weight
    def __str__(self):
        return str(self.id)+'connectedTo:'+str([x.id for x in self.connectedTo])
    def getConnections(self):
        return self.connectedTo.keys()
    def getId(self):
        return self.id
    def getWeight(self,nbr):
        return self.connectedTo[nbr]
class Graph:
    def __init__(self):
        self.vertList={}
        self.numVertices=0
    def addVertex(self,key):
        self.numVertices+=1
        newVertex=Vertex(key)
        self.vertList[key]=newVertex
        return newVertex
    def getVertex(self,n):
        if n in self.vertList:
            return self.vertList[n]
        else:
            return None
    def __contains__(self,n):
        return n in self.vertList
    def addEdge(self,f,t,weight=0):
        if f not in self.vertList:
            nv=self.addVertex(f)
        if t not in self.vertList:
            nv=self.addVertex(t)
        self.vertList[f].addNeighbor(self.vertList[t],weight)
    def getVertices(self):
        return self.vertList.keys()
    def __iter__(self):
        return iter(self.vertList.values())

graph=Graph()
n,m=map(int,input().split())
for i in range(n):
    graph.addVertex(i)
for _ in range(m):
    a,b=map(int,input().split())
    graph.addEdge(a,b)
    graph.addEdge(b,a)
for i in range(n):
    a=[0]*n
    vertex=graph.vertList[i]
    b=vertex.getConnections()
    a[i]+=len(b)
    for j in b:
        a[j.id]-=1
```

```
    print(*a,sep=' ')
```

代码运行截图



18160: 最大连通域面积

matrix/dfs similar, http://cs101.openjudge.cn/practice/18160

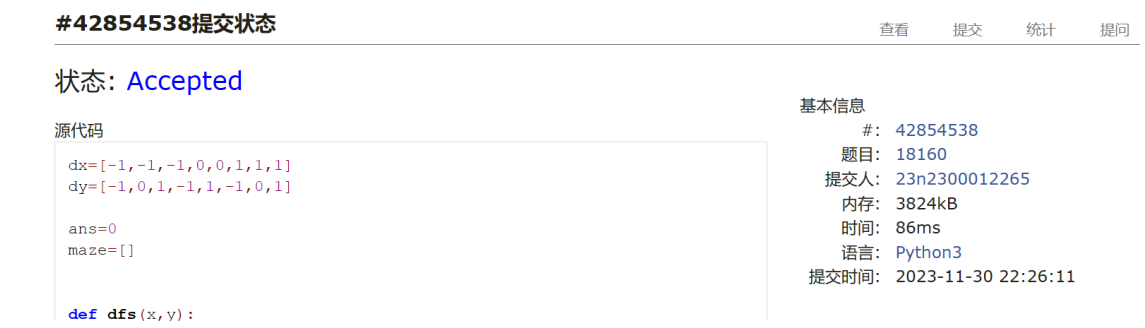
思路:

代码

```
dx=[-1,-1,-1,0,0,1,1,1]
dy=[-1,0,1,-1,1,-1,0,1]

ans=0
maze=[]


def dfs(x,y):
    global ans
    ans+=1
    maze[x][y]='.'

    for i in range(8):
        nx=x+dx[i]
        ny=y+dy[i]
        if maze[nx][ny]=='W':
            #print(x,y,vmaze)
            dfs(nx,ny)


t=int(input())
for _ in range(t):

    n,m=map(int,input().split())
```

```
    maze=[]
    maze.append(['.' for i in range(m+2)])
    for i in range(n):
        s=['.']+[j for j in input()]+['.']
        maze.append(s)
    maze.append(['.' for i in range(m+2)])

    #print(vmaze)

    area=[0]
    for i in range(1,n+1):
        for j in range(1,m+1):
            #print(vmaze[i][j])

            if maze[i][j]=='W':
                #print(213)
                ans=0
                dfs(i,j)
                area.append(ans)
                #print(i,j,vmaze,maze)

    print(max(area))
```

代码运行截图



sy383: 最大权值连通块

https://sunnywhy.com/sfbj/10/3/383

思路:

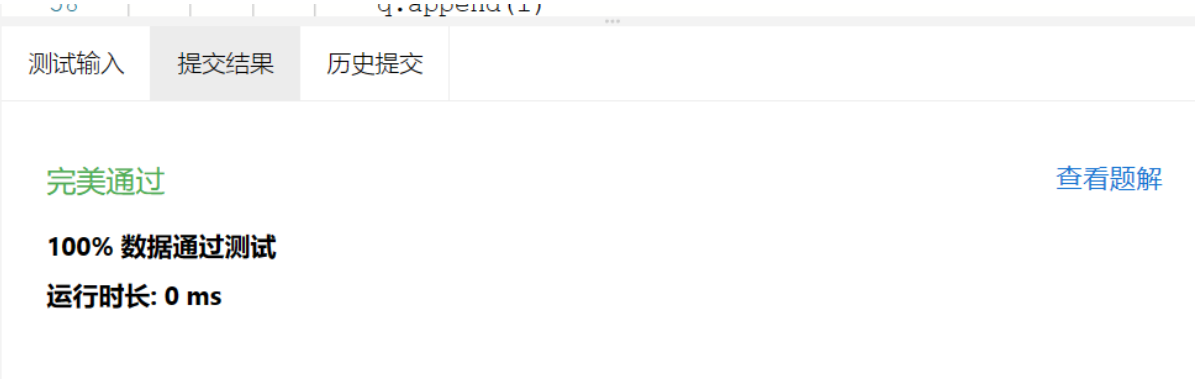
代码

```
import sys
from collections import deque

class Vertex:
```

```
    def __init__(self,key,weight=0):
        self.id=key
        self.weight=weight
        self.connectedTo={}
        self.visit=0
    def addNeighbor(self,nbr,weight=0):
        self.connectedTo[nbr]=weight
    def __str__(self):
        return str(self.id)+'connectedTo:'+str([x.id for x in self.connectedTo])
    def getConnections(self):
        return self.connectedTo.keys()
    def getId(self):
        return self.id
    def getWeight(self,nbr):
        return self.connectedTo[nbr]
class Graph:
    def __init__(self):
        self.vertList={}
        self.numVertices=0
    def addVertex(self,key,weight=0):
        self.numVertices+=1
        newVertex=Vertex(key,weight)
        self.vertList[key]=newVertex
        return newVertex
    def getVertex(self,n):
        if n in self.vertList:
            return self.vertList[n]
        else:
            return None
    def __contains__(self,n):
        return n in self.vertList
    def addEdge(self,f,t,weight=0):
        if f not in self.vertList:
            nv=self.addVertex(f)
        if t not in self.vertList:
            nv=self.addVertex(t)
        self.vertList[f].addNeighbor(self.vertList[t],weight)
    def getVertices(self):
        return self.vertList.keys()
    def __iter__(self):
        return iter(self.vertList.values())

def bfs(seed):
    ans=0
    q=deque()
    q.append(seed)
    ans+=seed.weight

    seed.visit=1
    while q:
        a=q.popleft()
        for i in a.getConnections():
            if i.visit==0:
                q.append(i)
                ans+=i.weight
                i.visit=1
```

```
    return ans


graph=Graph()
n,m=map(int,input().split())
weight=[int(i) for i in input().split()]
for i in range(n):
    graph.addVertex(i,weight[i])
for i in range(m):
    a,b=map(int,input().split())
    graph.addEdge(a, b)
    graph.addEdge(b, a)
ans=0
for i in graph.getVertices():
    vex=graph.getVertex(i)
    if vex.visit==0:
        ans=max(ans,bfs(vex))
print(ans)
```

代码运行截图



03441: 4 Values whose Sum is 0

data structure/binary search, http://cs101.openjudge.cn/practice/03441

思路:

代码

```
n = int(input())
A, B, C, D = [], [], [], []
for i in range(n):
    a, b, c, d = map(int, input().split())
    A.append(a)
    B.append(b)
    C.append(c)
```

```
    D.append(d)
ab = {}
for i in range(n):
    for j in range(n):
        if A[i]+B[j] in ab:
            ab[A[i]+B[j]]+=1
        else:
            ab[A[i]+B[j]]=1

ans=0
for i in C:
    for j in D:
        #print(i,j)
        if (-i-j) in ab:
            ans+=ab[-i-j]
print(ans)
```

代码运行截图

04089: 电话号码

trie, http://cs101.openjudge.cn/practice/04089/

Trie 数据结构可能需要自学下。

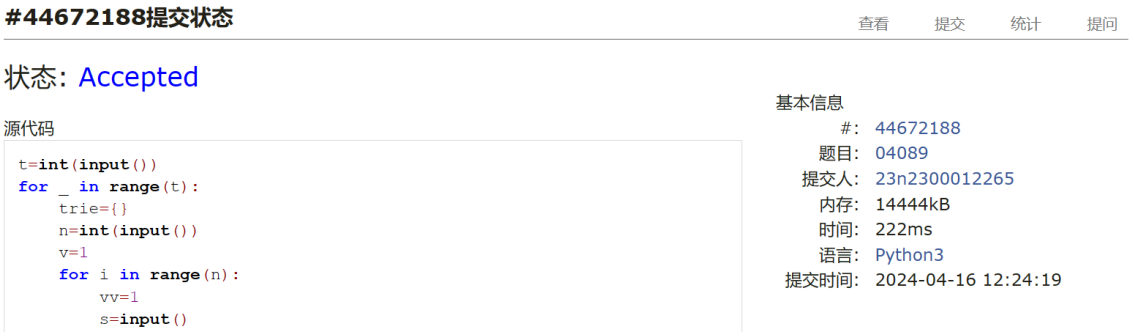
思路:

代码

```
t=int(input())
for _ in range(t):
    trie={}
    n=int(input())
    v=1
    for i in range(n):
        vv=1
        s=input()
```

```
        a=trie
        for j in range(len(s)):
            if v==0:
                break
            if s[j] in a:
                a=a[s[j]]
                if a==-1:
                    v=0
            else:
                vv=0
                if j==len(s)-1:
                    a[s[-1]]=-1
                else:
                    a[s[j]]={}
                    a=a[s[j]]
        if vv:
            v=0
    if v:
        print("YES")
    else:
        print("NO")
```

代码运行截图



04082: 树的镜面映射

http://cs101.openjudge.cn/practice/04082/

思路:

先把伪满二叉树建了，然后转换成普通的树，最后遍历输出

建二叉树的时候，根据1/0来判断是该跳到父节点还是继续从该节点往下建

转化树的时候，如果是左节点就加到父节点的儿子序列中，然后是右节点就加到父节点的父节点的儿子序列中

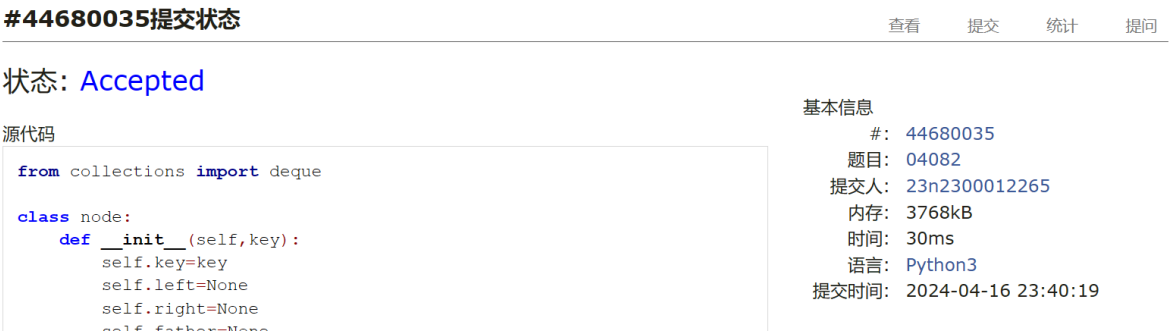
遍历就是普通的bfs

代码

```
from collections import deque

class node:
    def __init__(self,key):
        self.key=key
        self.left=None
        self.right=None
        self.father=None
        self.new=None
class newnode:
    def __init__(self,key):
        self.old=key
        self.key=key.key
        self.child=[]
        self.father=None
def build(node):
    if node.key=='$':
        return
    new=newnode(node)
    node.new=new
    if node.father!=None:
        newfa=node.father.new
        if node.father.left==node:
            newfa.child.append(new)
            new.father=newfa
        if node.father.right==node:
            newfa.father.child.append(new)
            new.father=newfa.father
    if node.left!=None:
        build(node.left)
    if node.right!=None:
        build(node.right)
    return new
def bfs(node):
    queue=deque()
    queue.append(node)
    while queue:
        now=queue.popleft()
        print(now.key,end=' ')
        for i in range(len(now.child)-1,-1,-1):
            queue.append(now.child[i])

n=int(input())
s=input().split()
root=node(s[0][0])
a=root
for i in s[1:]:
    nodei=node(i[0])
    v=0
    while 1:
        if a.left==None:
            v=1
            nodei.father=a
            a.left=nodei
            break
        if a.right==None:
```

```
            v=2
            nodei.father=a
            a.right=nodei
            break
        a=a.father
    if i[1]=='1' and v==2:
        a=a.father
    if i[1]=='1' and v==1:
        pass
    if i[1]=='0':
        a=nodei
newroot=build(root)
bfs(newroot)
```

代码运行截图



2. 学习总结和收获

学到了图的建立与表达，学会了graph和vertex类的写法

然后第一次，完全不依靠参考代码，自己写出来了一棵树的题（电话号码和树的镜面映射）！！！还一遍就AC了！非常开心

对字典的各种灵活应用产生了深刻的印象