Assignment #8: 图论：概念、遍历，及 树算

Updated 1919 GMT+8 Apr 8, 2024

2024 spring, Complied by ==狄晨阳 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

==（请改为同学的操作系统、编程环境等）==

操作系统：Windows11

Python编程环境：Spyder IDE 5.4.3

C/C++编程环境：无

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：定义两个类储存图，然后读取数据并转换为拉普拉斯矩阵

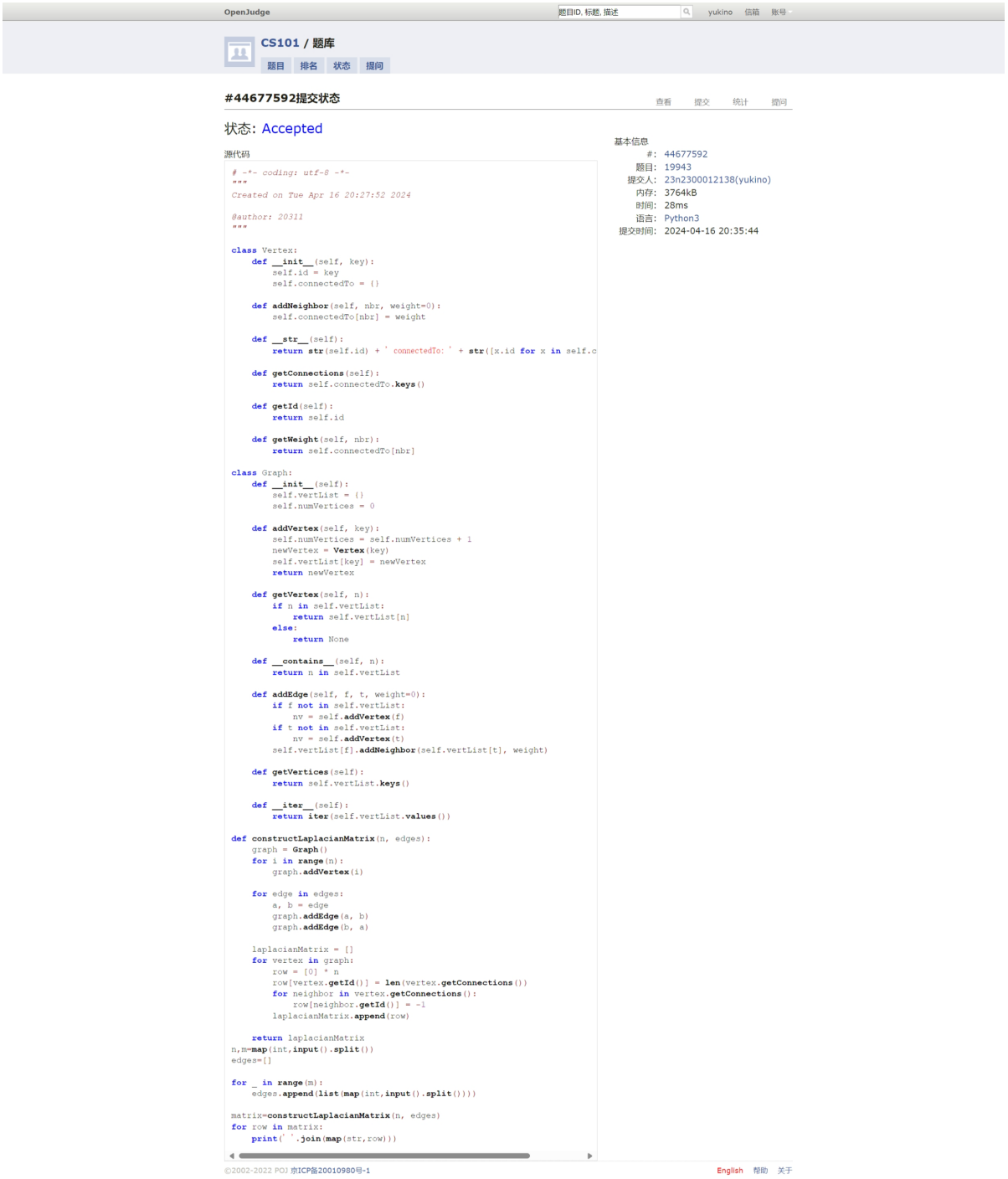
代码

```
# # -*- coding: utf-8 -*-
"""
Created on Tue Apr 16 20:27:52 2024

@author: 20311
"""

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 = 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())

def constructLaplacianMatrix(n, edges):
    graph = Graph()
    for i in range(n):
        graph.addVertex(i)
```

```
    for edge in edges:
        a, b = edge
        graph.addEdge(a, b)
        graph.addEdge(b, a)

    laplacianMatrix = []
    for vertex in graph:
        row = [0] * n
        row[vertex.getId()] = len(vertex.getConnections())
        for neighbor in vertex.getConnections():
            row[neighbor.getId()] = -1
        laplacianMatrix.append(row)

    return laplacianMatrix
n,m=map(int,input().split())
edges=[]

for _ in range(m):
    edges.append(list(map(int,input().split())))

matrix=constructLaplacianMatrix(n, edges)
for row in matrix:
    print(' '.join(map(str,row)))
```

代码运行截图 ==（至少包含有"Accepted"）==




```
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 16 20:27:52 2024

@author: 20311
"""

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.c

    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 = 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())

def constructLaplacianMatrix(n, edges):
    graph = Graph()
    for i in range(n):
        graph.addVertex(i)

    for edge in edges:
        a, b = edge
        graph.addEdge(a, b)
        graph.addEdge(b, a)

    laplacianMatrix = []
    for vertex in graph:
        row = [0] * n
        row[vertex.getId()] = len(vertex.getConnections())
        for neighbor in vertex.getConnections():
            row[neighbor.getId()] = -1
        laplacianMatrix.append(row)

    return laplacianMatrix
n,m=map(int,input().split())
edges=[]

for _ in range(m):
    edges.append(list(map(int,input().split())))

matrix=constructLaplacianMatrix(n, edges)
for row in matrix:
    print(' '.join(map(str,row)))
```

18160: 最大连通域面积

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

思路：使用了dfs的搜索找出邻居是否为'w'并求出最大联通区域的面积

代码

```
#
t=int(input())
pace=[[-1,-1],[-1,0],[-1,1],[0,-1],[0,1],[1,-1],[1,0],[1,1]]
for _ in range(t):
    n,m=[int(x) for x in input().split()]
```

```
    matrix=[]
    for i in range(n):
        matrix.append(list(input()))

    maxx=0
    for i in range(n):
        for j in range(m):
            if matrix[i][j]=='W':
                c=0
                r=[[i,j]]
                matrix[i][j]='.'

                while len(r)>0:
                    c+=1
                    s=r.pop(0)
                    for p in pace:
                        x=s[0]+p[0]
                        y=s[1]+p[1]
                        if x>=0 and y>=0 and x<n and y<m:
                            if matrix[x][y]=='W':
                                r.append([x,y])
                                matrix[x][y]='.'

                maxx=max(maxx,c)

    print(maxx)
```

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

源代码

```
t=int(input())
pace=[[-1,-1],[-1,0],[-1,1],[0,-1],[0,1],[1,-1],[1,0],[1,1]]
for _ in range(t):
    n,m=[int(x) for x in input().split()]

    matrix=[]
    for i in range(n):
        matrix.append(list(input()))

    maxx=0
    for i in range(n):
        for j in range(m):
            if matrix[i][j]=='W':
                c=0
                r=[[i,j]]
                matrix[i][j]='.'

                while len(r)>0:
                    c+=1
                    s=r.pop(0)
                    for p in pace:
                        x=s[0]+p[0]
                        y=s[1]+p[1]
                        if x>=0 and y>=0 and x<n and y<m:
                            if matrix[x][y]=='W':
                                r.append([x,y])
                                matrix[x][y]='.'

                maxx=max(maxx,c)

    print(maxx)
```

基本信息

#:	44677634
题目:	18160
提交人:	23n2300012138(yukino)
内存:	3756kB
时间:	127ms
语言:	Python3
提交时间:	2024-04-16 20:38:58

sy383: 最大权值连通块

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

思路：与上一题思路基本相同，只需加入权重即可

代码

```
# # -*- coding: utf-8 -*-
"""
Created on Tue Apr 16 20:43:14 2024

@author: 20311
"""

n,m=map(int,input().split())
weights=list(map(int,input().split()))
vertexes=[True]*n
connections={x:[] for x in range(n)}
for _ in range(m):
    a,b=map(int,input().split())
    connections[a].append(b)
    connections[b].append(a)

maxx=0
for i in range(n):
    if vertexes[i]:
        vertexes[i]=False
        summ=0
        queue=[i]
        while queue:
            a=queue.pop(0)
            summ+=weights[a]
            vertexes[a]=False
            for nei in connections[a]:
                if vertexes[nei]:
                    queue.append(nei)
                    vertexes[nei]=False
        maxx=max(maxx,summ)
print(maxx)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

代码书写 Python

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 16 20:43:14 2024

@author: 20311
"""

n,m=map(int,input().split())
weights=list(map(int,input().split()))
vertexes=[True]*n
connections={x:[] for x in range(n)}
for _ in range(m):
    a,b=map(int,input().split())
    connections[a].append(b)
    connections[b].append(a)

```

测试输入 提交结果 历史提交

完美通过 查看题解

100% 数据通过测试

运行时长: 0 ms

03441: 4 Values whose Sum is 0

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

思路：使用一个列表储存所有可能的ab和及每一种的数目，再根据组合出的所有cd可能和的相反数确认其是否在列表中

代码

```
#
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 16 21:14:29 2024

@author: 20311
"""

n=int(input())
a=[0]*n
b=[0]*n
c=[0]*n
d=[0]*n
```

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

ab={}
for i in a:
    for j in b:
        if i+j in ab:
            ab[i+j]+=1
        else:
            ab[i+j]=1

ans=0
for i in c:
    for j in d:
        if -i-j in ab:
            ans+=ab[-i-j]
print(ans)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 16 21:14:29 2024

@author: 20311
"""

n=int(input())
a=[0]*n
b=[0]*n
c=[0]*n
d=[0]*n

for i in range(n):
    a[i],b[i],c[i],d[i]=map(int,input().split())

ab={}
for i in a:
    for j in b:
        if i+j in ab:
            ab[i+j]+=1
        else:
            ab[i+j]=1

ans=0
for i in c:
    for j in d:
        if -i-j in ab:
            ans+=ab[-i-j]
print(ans)
```

基本信息

#:	44678212
题目:	03441
提交人:	23n2300012138(yukino)
内存:	171728kB
时间:	4039ms
语言:	Python3
提交时间:	2024-04-16 21:25:34

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：用字符串储存电话号码，对其按字典序排序，然后只需检查下一项的开头是不是上一项即可

代码

```
# # -*- coding: utf-8 -*-
"""
Created on Tue Apr 16 21:33:53 2024

@author: 20311
"""

t=int(input())
for _ in range(t):
    n=int(input())
    dials=[]
    for z in range(n):
        dials.append(input())
    dials.sort()
    jg='YES'
    for i in range(n-1):
        l=len(dials[i])
        if dials[i] == dials[i+1][:l]:
            jg='NO'
            break
    print(jg)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 16 21:33:53 2024

@author: 20311
"""

t=int(input())
for _ in range(t):
    n=int(input())
    dials=[]
    for z in range(n):
        dials.append(input())
    dials.sort()
    jg='YES'
    for i in range(n-1):
        l=len(dials[i])
        if dials[i] == dials[i+1][:l]:
            jg='NO'
            break
    print(jg)
```

基本信息

#:	44678846
题目:	04089
提交人:	23n2300012138(yukino)
内存:	4328kB
时间:	87ms
语言:	Python3
提交时间:	2024-04-16 22:06:26

04082: 树的镜面映射

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

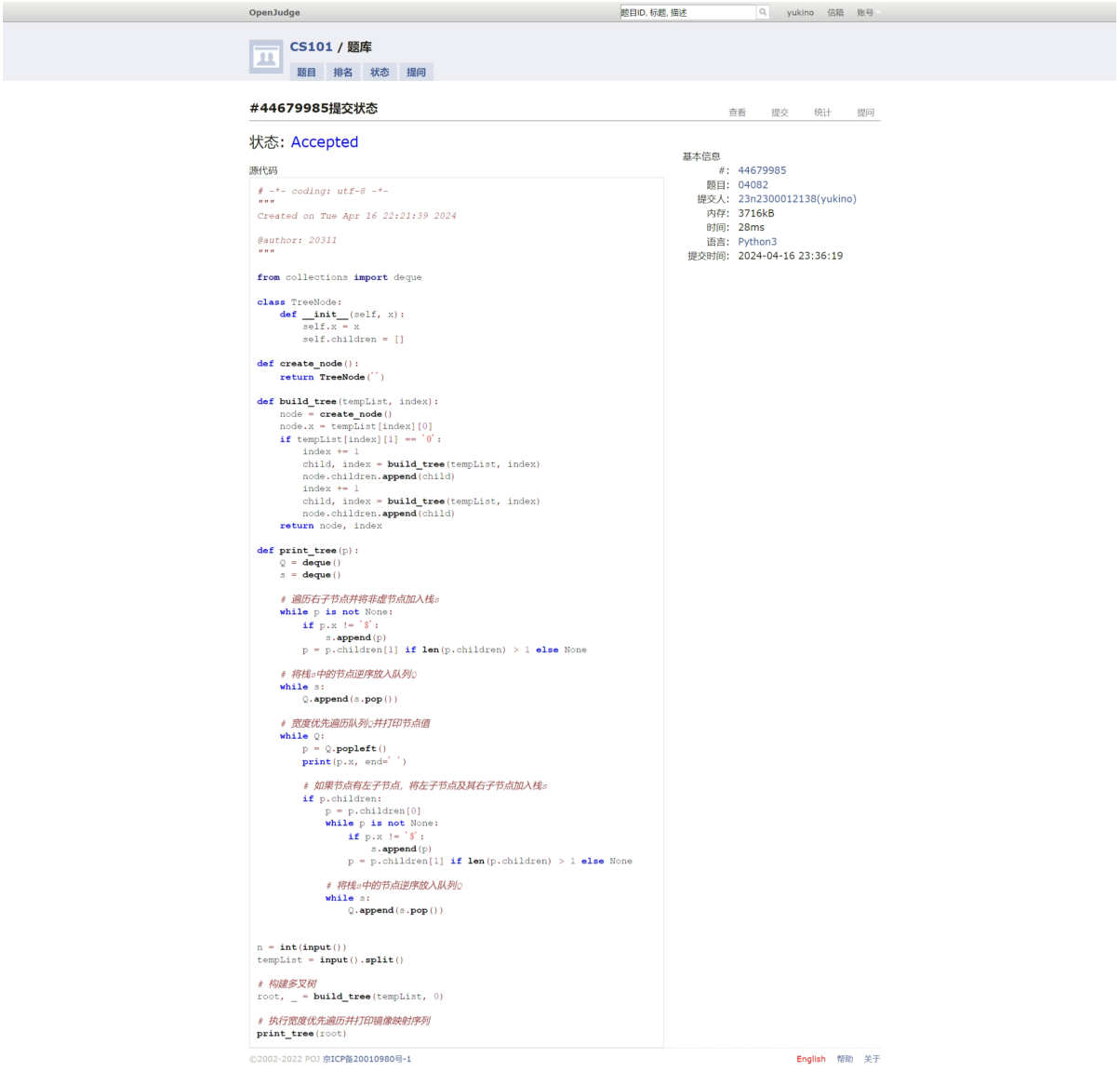
思路：整体想法就是读取完了之后对每层逆序输出，但是卡在了建树的地方，便看了题解

代码

```
# from collections import deque

class TreeNode:
    def __init__(self, x):
        self.x = x
        self.children = []

def create_node():
    return TreeNode('')

def build_tree(tempList, index):
    node = create_node()
    node.x = tempList[index][0]
    if tempList[index][1] == '0':
        index += 1
        child, index = build_tree(tempList, index)
        node.children.append(child)
        index += 1
        child, index = build_tree(tempList, index)
        node.children.append(child)
    return node, index

def print_tree(p):
    Q = deque()
    s = deque()

    # 遍历右子节点并将非虚节点加入栈s
    while p is not None:
        if p.x != '$':
            s.append(p)
        p = p.children[1] if len(p.children) > 1 else None

    # 将栈s中的节点逆序放入队列Q
    while s:
        Q.append(s.pop())

    # 宽度优先遍历队列Q并打印节点值
    while Q:
        p = Q.popleft()
        print(p.x, end=' ')

        # 如果节点有左子节点，将左子节点及其右子节点加入栈s
        if p.children:
            p = p.children[0]
```

```
            while p is not None:
                if p.x != '$':
                    s.append(p)
                p = p.children[1] if len(p.children) > 1 else None

            # 将栈s中的节点逆序放入队列Q
            while s:
                Q.append(s.pop())


n = int(input())
tempList = input().split()

# 构建多叉树
root, _ = build_tree(tempList, 0)

# 执行宽度优先遍历并打印镜像映射序列
print_tree(root)
```

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

==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。==

有

本次作业中复习了bfs和dfs以及一些数据类型，有几道题还是比较困难的，仍需多加练习来加快解题速度，但最近期中考试花的时间较少，考完了要补上来