Assignment #8: 图论：概念、遍历，及 树算

Updated 1919 GMT+8 Apr 8, 2024

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

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 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

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

耗时：10mins

思路：利用Vertex类，Graph类可以轻松实现

代码

```
#
class Vertex:# 定义节点类
    def __init__(self,key): # 初始化，设定节点的名称，建立节点邻居字典
        self.id=key
        self.connectedTo={}

    def addNeighbor(self,nbr,weight=0): # 给这个节点加邻居nbr，权重为weight
        # 邻居词典的key是邻居名字，value是weight
        self.connectedTo[nbr]=weight

    def __str__(self): #print这个节点的时候，这个方法会直接生效，打印节点名称和邻居
        return str(self.id)+'connectedTo:'+str([x.id for x in self.connectedTo])
```

```
    def getConnections(self): #返回所有邻居的名字列表
        return self.connectedTo.keys()

    def getId(self): # 获得这个节点的名字
        return self.id

    def getWeight(self,nbr): # 获得和nbr这个邻居的权重
        return self.connectedTo[nbr]

class Graph:
    def __init__(self): # 初始化，设定节点的词典，和现在几个节点
        self.vertList={}
        self.numVertices=0

    def addVertex(self,key): # 给图加一个节点key，节点词典的key是节点名称，value是节点对
象
        self.numVertices+=1
        newVertex=Vertex(key)
        self.vertList[key]=newVertex
        return newVertex

    def getVertex(self,n): # 根据节点名称，获取节点对象
        if n in self.vertList:
            return self.vertList[n]
        else:
            return None

    def __contains__(self,n): # 便捷判断某个节点在不在图中
        return n in self.vertList

    def addEdge(self,f,t,cost=0): # 给f加邻居t，权重为cost
        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],cost)

    def getVertices(self): # 获得图所有点的名称列表
        return self.vertList.keys()

    def __iter__(self): # 让图可以迭代，迭代对象是所有点的对象
        return iter(self.vertList.values())

def buildgraph(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)

    lplsmatrix=[]

    for vertex in graph:
```

```
        row=[0]*n
        row[vertex.getId()]=len(vertex.getConnections())
        for nbr in vertex.getConnections():
            row[nbr.getId()]=-1
        lplsmatrix.append(row)
    return lplsmatrix

n,m=map(int,input().split())
edges=[]
for i in range(m):
    a,b=map(int,input().split())
    edges.append((a,b))
lplsmatrix=buildgraph(n,edges)
for row in lplsmatrix:
    print(' '.join(map(str,row)))
```

代码运行截图 ==（至少包含有"Accepted"）==

#44634960提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```
class Vertex:# 定义节点类
    def __init__(self,key): # 初始化，设定节点的名称，建立节点邻居字典
        self.id=key
        self.connectedTo={}

    def addNeighbor(self,nbr,weight=0): # 给这个节点加邻居nbr，权重为weight
        # 邻居词典的key是邻居名字，value是weight
        self.connectedTo[nbr]=weight
```

基本信息

#: 44634960
题目: 19943
提交人: 尹柚鑫(2100015878)
内存: 3768kB
时间: 25ms
语言: Python3
提交时间: 2024-04-13 17:30:28

18160: 最大连通域面积

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

耗时：30mins

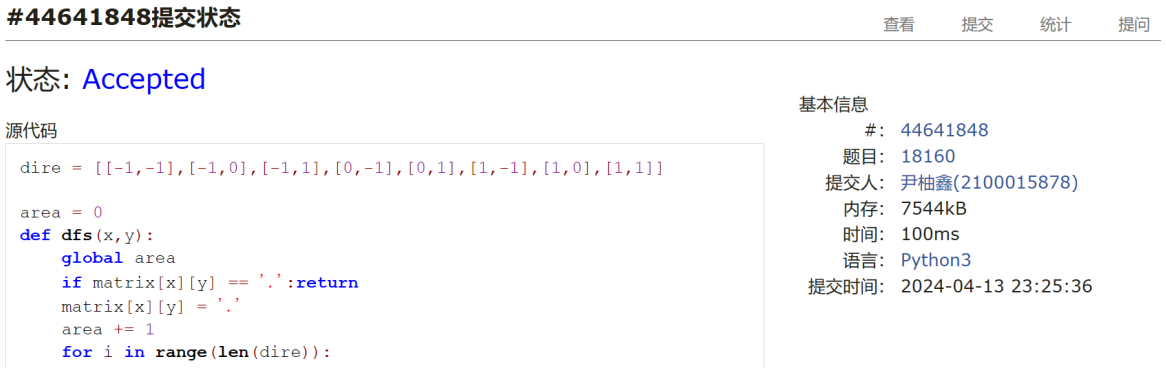
思路：对每一个格，递归遍历周围的格，同时做标记

代码

```
#
dire = [[-1,-1],[-1,0],[-1,1],[0,-1],[0,1],[1,-1],[1,0],[1,1]]

area = 0
def dfs(x,y):
    global area
    if matrix[x][y] == '.':return
    matrix[x][y] = '.'
    area += 1
    for i in range(len(dire)):
        dfs(x+dire[i][0], y+dire[i][1])
```

```
for _ in range(int(input())):
    n,m = map(int,input().split())

    matrix = [['.' for _ in range(m+2)] for _ in range(n+2)]
    for i in range(1,n+1):
        matrix[i][1:-1] = input()

    sur = 0
    for i in range(1, n+1):
        for j in range(1, m+1):
            if matrix[i][j] == 'W':
                area = 0
                dfs(i, j)
                sur = max(sur, area)
    print(sur)
```

代码运行截图 ==（至少包含有"Accepted"）==



sy383: 最大权值连通块

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

耗时：15mins

思路：和上一题求最大连通面积很像，对每一个节点，递归遍历邻居，同时做标记，再算最大值

代码

```
#
n,m=map(int,input().split())
# n个点，m条边
weightslist=[*map(int,input().split())]#储存点的权重

graph = [[] for _ in range(n)]

for i in range(m):
    a,b=map(int,input().split())
    graph[a].append(b)
    graph[b].append(a)
```

```
visited=[0]*n

result_end=0
result_tmp=0

def calsumweight(index_node):
    global result_tmp

    if visited[index_node]==1:
        return

    result_tmp+=weightslist[index_node]
    visited[index_node]=1

    for i in graph[index_node]:
        calsumweight(i)
    return result_tmp

for i in range(n):
    visited=[0]*n
    result_tmp=0
    calsumweight(i)
    result_end=max(result_end,result_tmp)
print(result_end)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

提交时间	结果	时长 (ms)	语言	
2024-04-14 17:13:58	完美通过	0	Python	查看

03441: 4 Values whose Sum is 0

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

耗时：15mins

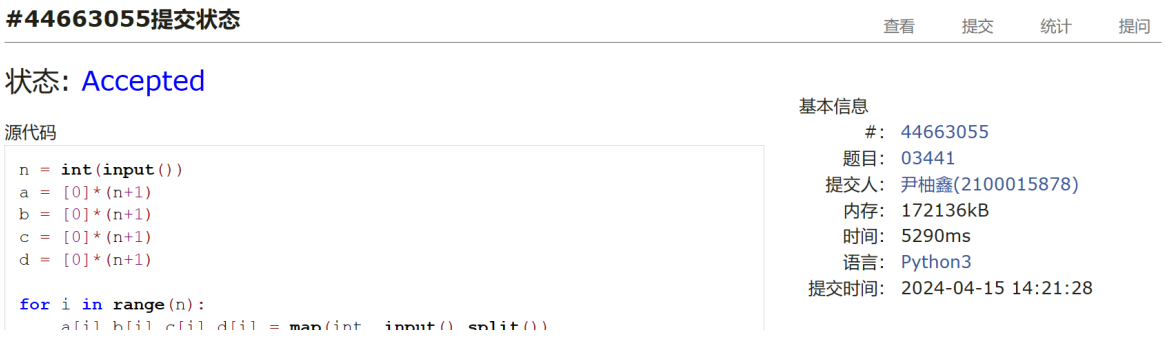
思路：比较简单粗暴的思路，分别两个列表求和，用词典储存结果，可以略微提高速度，然后比对两组数据的和是不是0

代码

```
#
n = int(input())
a = [0]*(n+1)
b = [0]*(n+1)
```

```
c = [0]*(n+1)
d = [0]*(n+1)

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

dict1 = {}
for i in range(n):
    for j in range(n):
        if not a[i]+b[j] in dict1:
            dict1[a[i] + b[j]] = 0
        dict1[a[i] + b[j]] += 1

ans = 0
for i in range(n):
    for j in range(n):
        if -(c[i]+d[j]) in dict1:
            ans += dict1[-(c[i]+d[j])]

print(ans)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



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

```
#
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

04082: 树的镜面映射

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

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

==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.graph=[[]]*n会导致所有子列表引用同一个对象，修改其中一个，所有的子列表都会变化，最好使用graph = [[] for _ in range(n)]

2.在函数内修改函数外的列表，是可以的，此时外部列表会被修改。

在函数内直接修改函数外的变量，比如a=1，是不行的，会报错。此时有两种情况

第一种是，global a，这个时候，函数中对a的修改，会同步到函数外

第二种是，在函数内定义a=1，再修改a，此时的修改只在函数内生效，函数外部的a其实还是没有改变

这其实是因为列表是可变的，变量不是可变的

3.第一次接触图这一数据类型，看定义的时候，感觉很像是自由度更高的树，写了几个题目，发现题目做法不太一样。一大区别在于，对于很多树的题目，都需要定义树这个类，但是对于图的题目，有时定义图这个类做题反而会更麻烦。原因或许是在于，树的很多题目都像是为树量身打造的，对数据的输出顺序的要求和树很契合。但是图的题目，就更自由更五花八门，做题中会有图的思想，但是做法可以多样