

# Assignment #8: 图论：概念、遍历，及 树算

Updated 1919 GMT+8 Apr 8, 2024

2024 spring, Compiled by 同学的姓名、院系

## 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：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

### 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：

矩阵类或许可以通过numpy实现

定义顶点、图类；定义D、A、L方法

代码

```
class Vertex:
    def __init__(self, key):
        self.neighborlist={}
        self.key=key
        self.degree=0
    def addedge(self, tovertex, weight=1):
        self.neighborlist[tovertex]=weight
```

```

        self.degree+=1
    def isneighbor(self,tovortex):
        return tovortex in self.neighborlist.keys()

class Graph:
    def __init__(self):
        self.vertexlist={}
        self.vertexcount=0
    def addvertex(self,key):
        self.vertexcount+=1
        newvertex=Vertex(key)
        self.vertexlist[key]=newvertex
    def addedge(self,fromvertex,tovortex,weight=1):
        if fromvertex not in self.vertexlist.keys():
            self.addvertex(fromvertex)
        if tovortex not in self.vertexlist.keys():
            self.addvertex(tovortex)
        self.vertexlist[fromvertex].addedge(tovortex,weight)
        self.vertexlist[tovortex].addedge(fromvertex,weight)
    def getvertexs(self):
        vs=[]
        for i in self.vertexlist.keys():
            vs.append(i)
        vs.sort()
        return vs
    def Amatrix(self):
        m=[]
        for x in range(n):
            m.append(0)
        D=[]
        for x in range(n):
            D.append(m.copy())
        for i in self.getvertexs():
            for j in self.getvertexs():
                if self.vertexlist[i].isneighbor(j):
                    D[i][j]=1
                else:
                    D[i][j]=0
        return D
    def Dmatrix(self):
        m=[]
        for x in range(n):
            m.append(0)
        D=[]
        for x in range(n):
            D.append(m.copy())
        for i in self.getvertexs():
            D[i][i]=self.vertexlist[i].degree
        return D
    def Lmatrix(self):
        D=self.Dmatrix()
        A=self.Amatrix()
        m=[]
        for x in range(n):
            m.append(0)
        L=[]

```

```

        for x in range(n):
            L.append(m.copy())
        for i in range(n):
            for j in range(n):
                L[i][j]=D[i][j]-A[i][j]
        return L

nm1st=[int(x) for x in input().split()]
n,m=nm1st[0],nm1st[1]
graph=Graph()
for _ in range(m):
    ft1st=[int(x) for x in input().split()]
    f,t=ft1st[0],ft1st[1]
    graph.addedge(f,t)
laplacematrix=graph.Lmatrix()
for i in laplacematrix:
    prt=[]
    for j in i:
        prt.append(str(j))
    print(' '.join(prt))

```

代码运行截图 (至少包含有"Accepted")

#44585316提交状态

[查看](#) [提交](#)

状态: **Accepted**

源代码

```

class Vertex:
    def __init__(self, key):
        self.neighborlist={}
        self.key=key
        self.degree=0
    def addedge(self, tovertex, weight=1):
        self.neighborlist[tovertext]=weight
        self.degree+=1
    def isneighbor(self, tovertex):
        return tovertex in self.neighborlist.keys()

class Graph:
    def __init__(self):
        self.vertexlist={}
        self.vertexcount=0
    def addvertex(self, key):
        self.vertexcount+=1
        newvertex=Vertex(key)
        self.vertexlist[key]=newvertex
    def addedge(self, fromvertex, tovertex, weight=1):
        if fromvertex not in self.vertexlist.keys():
            self.addvertex(fromvertex)
        if tovertex not in self.vertexlist.keys():
            self.addvertex(tovertex)
        self.vertexlist[fromvertex].addedge(tovertex, weight)
        self.vertexlist[tovertex].addedge(fromvertex, weight)
    def getvertexs(self):
        vs=[]
        for i in self.vertexlist.keys():
            vs.append(i)
        vs.sort()
        return vs
    def Amatrix(self):
        m=[]
        for x in range(n):
            m.append(0)

```

基本信息

#: 44585316  
 题目: 19943  
 提交人: zvk  
 内存: 3816kB  
 时间: 28ms  
 语言: Python3  
 提交时间: 2024-04-09 17:

## 18160: 最大连通域面积

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

思路:

dfs, 因为要找最大所以对每个节点都dfs一遍

代码

```
def dfs(i,j,visited,matrix):
    cnt=0
    if [i,j] not in visited:
        # print('new vertex')
        # print(matrix[i][j])
        visited.append([i,j])
        if matrix[i][j]=='w':
            # print(f'{i+1},{j+1}is w')
            cnt+=1
            if i<len(matrix)-1:
                # print('go down')
                cnt+=dfs(i+1,j,visited,matrix)
            if j<len(matrix[0])-1:
                # print('go right')
                cnt+=dfs(i,j+1,visited,matrix)
            if i<len(matrix)-1 and j<len(matrix[0])-1:
                # print('go down right')
                cnt+=dfs(i+1,j+1,visited,matrix)
            if i>0:
                # print('go up')
                cnt+=dfs(i-1,j, visited, matrix)
            if j>0:
                # print('go left')
                cnt+=dfs(i,j-1,visited,matrix)
            if i>0 and j>0:
                # print('go left up')
                cnt+=dfs(i-1,j-1,visited,matrix)
            if i>0 and j<len(matrix[0])-1:
                # print('go up right')
                cnt+=dfs(i-1,j+1,visited,matrix)
            if i<len(matrix)-1 and j>0:
                # print('go left down')
                cnt+=dfs(i+1,j-1,visited,matrix)
        return cnt

T=int(input())
matrixes=[]
for _ in range(T):
    N,M=map(int,input().split())
    matrix=[]
    for i in range(N):
        irow=input()
        adrow=[]
        for j in irow:
```

```

        adrow.append(j)
        matrix.append(adrow)
    matrixes.append(matrix)
for matrix in matrixes:
    probans=[]
    visited=[]
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            probans.append(dfs(i,j,visited,matrix))
    print(max(probans))

```

代码运行截图 (至少包含有"Accepted")

#44598360提交状态

[查看](#)
[提交](#)
[统计](#)
[提问](#)

状态: Accepted

源代码

```

def dfs(i,j,visited,matrix):
    cnt=0
    if [i,j] not in visited:
        # print('new vertex')
        # print(matrix[i][j])
        visited.append([i,j])
        if matrix[i][j]!='W':
            # print(f'{i+1},{j+1}is W')
            cnt+=1
            if i<len(matrix)-1:
                # print('go down')
                cnt+=dfs(i+1,j,visited,matrix)
            if j<len(matrix[0])-1:
                # print('go right')
                cnt+=dfs(i,j+1,visited,matrix)
            if i<len(matrix)-1 and j<len(matrix[0])-1:
                # print('go down right')
                cnt+=dfs(i+1,j+1,visited,matrix)
            if i>0:
                # print('go up')
                cnt+=dfs(i-1,j, visited, matrix)
            if j>0:
                # print('go left')
                cnt+=dfs(i,j-1,visited,matrix)
            if i>0 and j>0:
                # print('go left up')
                cnt+=dfs(i-1,j-1,visited,matrix)
            if i>0 and j<len(matrix[0])-1:
                # print('go up right')
                cnt+=dfs(i-1,j+1,visited,matrix)
            if i<len(matrix)-1 and j>0:
                # print('go left down')
                cnt+=dfs(i+1,j-1,visited,matrix)
        return cnt

T=int(input())
matrixes=[]
for _ in range(T):
    N,M=map(int,input().split())
    matrix=[]
    for i in range(N):
        irow=input()
        adrow=[]
        for j in irow:
            adrow.append(j)
        matrix.append(adrow)
    matrixes.append(matrix)
for matrix in matrixes:
    probans=[]
    visited=[]
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            probans.append(dfs(i,j,visited,matrix))
    print(max(probans))

```

基本信息

#: 44598360

题目: 18160

提交人: zxx

内存: 5128kB

时间: 2464ms

语言: Python3

提交时间: 2024-04-10 20:12:20

©2002-2022 POJ 京ICP备20010980号-1

[English](#)
[帮助](#)
[关于](#)

## sy383: 最大权值连通块

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

思路:

构建邻接表图, 节点的id是序号, 所以如果需要找到权重直接构建一个列表按顺序表示其权值就好, 之后建图, 但是对于这种有环无向图, 建立邻接表只能做到链接, dfs还必须建立双向的连接, 不然其邻居的表示不全

代码

```
#顶点类
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]

#图类
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, item):
        return item 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 dfs(vertex,visited,graph):
    wei=0
    if vertex not in visited:
        visited.append(vertex)
        wei+=weight_list[vertex]
        for nei in g.vertList[vertex].getConnections():
            wei+=dfs(nei.id,visited,graph)
    return wei

n,m=map(int,input().split())
weight_list=[int(x) for x in input().split()]
g=Graph()
for j in range(m):
    u,v=map(int,input().split())
    g.addEdge(u,v)
    g.addEdge(v,u)
visited=[]
weis=[]
for ver in g.getVertices():
    weis.append(dfs(ver,visited,g))
print(max(weis))
```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

The screenshot shows a coding competition interface. On the left, the problem statement is in Chinese, describing a graph with vertices and edges, and asking for the maximum sum of weights in a connected component. Below the problem statement is a sample input and output. The sample input is a 6x5 grid of numbers, and the sample output is the number 8. Below the sample input/output is a diagram of the graph. The graph has 6 vertices labeled 0 to 5. Vertex 0 is connected to 1 and 3. Vertex 1 is connected to 0 and 5. Vertex 3 is connected to 0 and 4. Vertex 4 is connected to 3 and 2. Vertex 2 is connected to 4 and 5. Vertex 5 is connected to 1 and 2. The weights of the vertices are: 0: 2, 1: 3, 2: 4, 3: 2, 4: 3, 5: 1. The maximum sum of weights in a connected component is 8 (vertices 0, 1, 3, 5).

On the right, the code editor shows a Python solution using a Depth-First Search (DFS) algorithm to find the maximum sum of weights in a connected component. The code is as follows:

```
def dfs(vertex,visited,graph):
    wei=0
    if vertex not in visited:
        visited.append(vertex)
        wei+=weight_list[vertex]
        for nei in g.vertList[vertex].getConnections():
            wei+=dfs(nei.id,visited,graph)
    return wei

n,m=map(int,input().split())
weight_list=[int(x) for x in input().split()]
g=Graph()
for j in range(m):
    u,v=map(int,input().split())
    g.addEdge(u,v)
    g.addEdge(v,u)
visited=[]
weis=[]
for ver in g.getVertices():
    weis.append(dfs(ver,visited,g))
print(max(weis))
```

The code editor also shows the test input, submission results, and history. The test input is the same as the sample input. The submission results show "完美通过" (Perfectly Passed) and "100% 数据通过测试" (100% Data Passed Test). The runtime is 0 ms.

## 03441: 4 Values whose Sum is 0

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

思路:

先穷举左边两列的和，构建一个字典记录每个和值所对应的可能的组数，然后穷举右边两列的和，在左边的字典里加对应的组数

代码

```
n=int(input())
martrix=[]
for _ in range(n):
    ipt_lst=[int(x) for x in input().split()]
    martrix.append(ipt_lst)
AB_dct={}
for i in range(n):
    for j in range(n):
        if martrix[i][0]+martrix[j][1] not in AB_dct.keys():
            AB_dct[martrix[i][0]+martrix[j][1]]=1
        else:
            AB_dct[martrix[i][0] + martrix[j][1]]+=1
cnt=0
for x in range(n):
    for y in range(n):
        sumCD=-martrix[x][2]-martrix[y][3]
        if sumCD in AB_dct.keys():
            cnt+=AB_dct[sumCD]
print(cnt)
```

代码运行截图 (AC代码截图，至少包含有"Accepted")

#44600887提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: Accepted

源代码

```
n=int(input())
martrix=[]
for _ in range(n):
    ipt_lst=[int(x) for x in input().split()]
    martrix.append(ipt_lst)
AB_dct={}
for i in range(n):
    for j in range(n):
        if martrix[i][0]+martrix[j][1] not in AB_dct.keys():
            AB_dct[martrix[i][0]+martrix[j][1]]=1
        else:
            AB_dct[martrix[i][0] + martrix[j][1]]+=1
cnt=0
for x in range(n):
    for y in range(n):
        sumCD=-martrix[x][2]-martrix[y][3]
        if sumCD in AB_dct.keys():
            cnt+=AB_dct[sumCD]
print(cnt)
```

基本信息

#: 44600887  
题目: 03441  
提交人: zxk  
内存: 172276kB  
时间: 6156ms  
语言: Python3  
提交时间: 2024-04-11 00:03:00

©2002-2022 POJ 京ICP备20010980号-1

[English](#) [帮助](#) [关于](#)

## 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：

用字典嵌套建树，按照每一位数字，如果是子节点就在子节点继续添加，如果不是就新建子节点，在此添加，最后记录end；遍历的时候按深度优先，如果end和其他节点同时存在，就是NO

代码

```
def trav(adct,i):
    ans=[]
    kys=[k for k in adct.keys()]
    if 'end' in kys and len(kys)>1:
        ans.append('NO')
    for key in adct.keys():
        # print(' '*i+key)
        if adct[key]:
            ans.extend(trav(adct[key],i+1))
    return ans
def insert(alst,adct):
    p=alst.pop(0)
    if p not in adct.keys():
        adct[p]={}
    if alst:
        adct[p]=insert(alst,adct[p])
    else :
        adct[p].update({'end':None })
    return adct
t=int(input())
anslst=[]
for _ in range(t):
    n=int(input())
    tree={}
    ans='YES'
    num_set=set()
    for i in range(n):
        ipt=str(input())
        if ipt in num_set:
            ans='NO'
        num_set.add(ipt)
        ipt_num_lst=[x for x in ipt]
        tree=insert(ipt_num_lst,tree)
    if 'NO' in trav(tree,0):
        ans='NO'
    anslst.append(ans)
for i in anslst:
    print(i)
```

代码运行截图 (AC代码截图，至少包含有"Accepted")

CS101 / 题库

题目 排名 状态 提问

#44602179提交状态

查看 提交

状态: Accepted

源代码

```
def trav(adct,i):
    ans=[]
    kys=[k for k in adct.keys()]
    if 'end' in kys and len(kys)>1:
        ans.append('NO')
    for key in adct.keys():
        # print(' '*i+key)
        if adct[key]:
            ans.extend(trav(adct[key],i+1))
    return ans
def insert(alst,adct):
    p=alst.pop(0)
    if p not in adct.keys():
        adct[p]={}
    if alst:
        adct[p]=insert(alst,adct[p])
    else:
        adct[p].update({'end':None })
    return adct
t=int(input())
anslst=[]
for _ in range(t):
    n=int(input())
    tree={}
    ans='YES'
    num_set=set()
    for i in range(n):
        ipt=str(input())
        if ipt in num_set:
            ans='NO'
        num_set.add(ipt)
        ipt_num_lst=[x for x in ipt]
        tree=insert(ipt_num_lst,tree)
    if 'NO' in trav(tree,0):
        ans='NO'
    anslst.append(ans)
for i in anslst:
    print(i)
```

基本信息

#: 44602179

题目: 04089

提交人: zxk

内存: 18368kB

时间: 460ms

语言: Python3

提交时间: 2024-04-11 11:00

©2002-2022 POJ 京ICP备20010980号-1

English

## 04082: 树的镜面映射

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

思路:

镜面映射简单，每层倒着输出就行；关键是这个树怎么构建，其实是斜着来表示树的层级，具体实现有点类似dfs，遇到顶就回溯

构建的数据结构很有意思，其实就是一个双层列表，我给他起了个名字叫梯子

代码

```
class ladder:
    def __init__(self):
        self.height=-1
        self.matrix=[]
```

```

def add_level(self):
    self.height+=1
    self.matrix.append([])

def insert_i_level(self,node,level):
    if self.height<level:
        self.add_level()
    self.matrix[level].append(node)

def traversal(Ladder):
    ans=[]
    for i in range(Ladder.height+1):
        stack=[]
        for j in Ladder.matrix[i]:
            stack.insert(0,j)
        ans.extend(stack)
    print(' '.join(ans))

n=int(input())
data=[str(x) for x in input().split()]
L=ladder()
level=0
while data:
    # print(f'current level is {level}')
    p=data.pop(0)
    if p == '$1':
        level-=1
    # print('turn right')
    else:
        L.insert_i_level(p[0],level)
    # print(f'add {p[0]} into level {level}')
    if p[-1]!='1':
        level+=1
    else:
        level-=1

# for i in range(L.height+1):
#     print(L.matrix[i])
traversal(L)

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

状态: Accepted

源代码

```
class ladder:
    def __init__(self):
        self.height=-1
        self.matrix=[]

    def add_level(self):
        self.height+=1
        self.matrix.append([])

    def insert_i_level(self,node,level):
        if self.height<level:
            self.add_level()
            self.matrix[level].append(node)

def traversal(Ladder):
    ans=[]
    for i in range(Ladder.height+1):
        stack=[]
        for j in Ladder.matrix[i]:
            stack.insert(0,j)
        ans.extend(stack)
    print(' '.join(ans))

n=int(input())
data=[str(x) for x in input().split()]
L=ladder()
level=0
while data:
    # print(f'current level is {level}')
    p=data.pop(0)
    if p=='1':
        level-=1
    # print('turn right')
    else:
        L.insert_i_level(p[0],level)
    # print(f'add {p[0]} into level {level}')
    if p[-1]!='1':
        level+=1
    else:
        level-=1

#for i in range(L.height+1):
#    print(L.matrix[i])
traversal(L)
```

基本信息

#: 44677899

题目: 04082

提交人: zxx

内存: 3688kB

时间: 31ms

语言: Python3

提交时间: 2024-04-16 21:

## 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。

- Laplace矩阵：第一次手搓顶点和图的类，卡住的地方在于有的顶点没有边.....导致在做类的时候是按照输入哪个顶点记录哪个顶点的，后来改的时候就很不乱，最后终于改过了；其实Laplace矩阵的D和A是无关的.....可以直接当一个算
- 最大连通面积：第一次接触DFS，之前看群里有很多人发了各种DFS，感觉好厉害，现在看了貌似还挺简单的...就和走迷宫遍历一样，顺着一条路走，走不通了就返回，在上一个岔路口走另一条路，直到走通，最后可能也没都走一遍就return了，这个题我设计的是当W无路可走了就返回，但对于不同的W岛没办法处理，所以干脆对所有Vertex都dfs一遍，反正有visited做记录，充其量会多个O(n)，有没有什么更简单的办法我还没想到；卡住的地方在于W是大写而不是小写（😭）

·最大权值连通块：这个是定义了vertex和graph类，采用邻接表表示，这个题的问题出在了邻接表表示有环无向表的时候要添加两次边，不然信息构建不全，然后就是dfs即可

·和为0：看到数据规模被吓了一跳，这么大规模的数据不可能都遍历一遍，但是貌似数据又没什么特点，只能遍历；问了一下gpt有什么思路，说是dfs，其实我没有搞懂为啥是dfs。。。其实我对于动规也不熟悉，但是受到了启发，遂尝试用字典表示每个可能的和所对应的组数，甚至感觉有点像并查集啊。。。就是把不用的信息都滤掉，并查集是滤掉路径，这个题是不关心具体是哪两个数和为某个数，只关心有几组，用dict可以大大缩减时间

·电话号码：尝试了很长时间，从开始的时候就知道要建树，先是定义节点，但是感觉用list做子节点会大大增加时间复杂度，如果用dict做子节点的话。。。何不直接用字典建树呢；用字典建树卡在了递归的语法上边，尝试了很多次才把树建好，遍历的时候又出现了很多语法问题，最后终于通过dfsAC了；有一个收获就是当不知道怎么递归错了的时候，可以手动可视化，遍历一个看得见的结构出来，可能就知道哪一步有问题了，遍历看得见的结构就比较像之前做的文件结构图的题

·镜面映射：没看题解，自己写的，感觉还是比较好理解的吧，只是数据结构上可能需要好好斟酌一下；自己编了一个数据结构叫“ladder”，是一个嵌套列表，有添层和给指定层添数的功能（貌似不class也可以实现，只是用类更能体现数算的精髓（doge）忘了叫什么名字了，反正就是类似于有人研发一辆车，但是具体怎么开，先踩什么后踩什么不管，有人开车，知道怎么给油怎么刹车，但是不知道怎么实现的，中间有个接口）