Assignment #8: 图论：概念、遍历，及 树算

Updated 1919 GMT+8 Apr 8, 2024

2024 spring, Complied by ==同学的姓名、院系==

罗瑞哲 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：Windows 10

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：

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 = 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):  # 添加顶点
        graph.addVertex(i)

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

```
    laplacianMatrix = []    # 构建拉普拉斯矩阵
    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())    # 解析输入
edges = []
for i in range(m):
    a, b = map(int, input().split())
    edges.append((a, b))

laplacianMatrix = constructLaplacianMatrix(n, edges)    # 构建拉普拉斯矩阵

for row in laplacianMatrix: # 输出结果
    print(' '.join(map(str, row)))
```

代码运行截图 ==（至少包含有"Accepted"）==

#44675387提交状态

状态: Accepted

源代码

```
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.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:
```

18160: 最大连通域面积

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

思路:

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"）==

#44675502提交状态

状态: Accepted

源代码

```
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)
```

sy383: 最大权值连通块

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

思路:

代码

```
#
class Node():
    def __init__(self, value, weight, visit):
        self.value = value
        self.weight = weight
        self.children = []
        self.visit = visit

def dfs(node):
```

```
    source = [node]
    answer = 0
    while source:
        subject = source.pop()
        if not subject.visit:
            subject.visit = True
            answer += subject.weight
            source += subject.children[::-1]
    return answer

n, m = map(int, input().split())
node_list = [Node(i, 0, False) for i in range(n)]
weight_list = list(map(int, input().split()))

for i in range(n):
    node_list[i].weight = weight_list[i]

for _ in range(m):
    a, b = map(int, input().split())
    node_list[a].children += node_list[b],
    node_list[b].children += node_list[a],

max_mass = 0
for node in node_list:
    if not node.visit:
        max_mass = max(max_mass, dfs(node))

print(max_mass)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

代码书写 Python

```
class Node():
    def __init__(self, value, weight, visit):
        self.value = value
        self.weight = weight
        self.children = []
        self.visit = visit

def dfs(node):
    source = [node]
    answer = 0
    while source:
        subject = source.pop()
        if not subject.visit:
            subject.visit = True
            answer += subject.weight
            source += subject.children[::-1]
```

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:

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"）==

#44675484提交状态

状态: Accepted

源代码

```
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)
```

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

```
#
class TrieNode:
    def __init__(self):
        self.child={}


class Trie:
    def __init__(self):
        self.root = TrieNode()

    def insert(self, nums):
        curnode = self.root
        for x in nums:
            if x not in curnode.child:
                curnode.child[x] = TrieNode()
            curnode=curnode.child[x]

    def search(self, num):
        curnode = self.root
        for x in num:
            if x not in curnode.child:
                return 0
            curnode = curnode.child[x]
        return 1


t = int(input())
p = []
for _ in range(t):
    n = int(input())
    nums = []
    for _ in range(n):
        nums.append(str(input()))
    nums.sort(reverse=True)
    s = 0
    trie = Trie()
    for num in nums:
        s += trie.search(num)
        trie.insert(num)
    if s > 0:
        print('NO')
    else:
        print('YES')
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
class TrieNode:
    def __init__(self):
        self.child={}


class Trie:
    def __init__(self):
        self.root = TrieNode()

    def insert(self, nums):
        curnode = self.root
        for x in nums:
            if x not in curnode.child:
                curnode.child[x] = TrieNode()
            curnode=curnode.child[x]

    def search(self, num):
        curnode = self.root
        for x in num:
            if x not in curnode.child:
                return 0
            curnode = curnode.child[x]
        return 1


t = int(input())
p = []
for _ in range(t):
    n = int(input())
    nums = []
    for _ in range(n):
        nums.append(str(input()))
    nums.sort(reverse=True)
    s = 0
    trie = Trie()
    for num in nums:
        s += trie.search(num)
        trie.insert(num)
    if s > 0:
        print('NO')
    else:
        print('YES')
```

04082: 树的镜面映射

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

思路:

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"）==

#44675548提交状态

状态：Accepted

源代码

```
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()
```

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。==

对于图这个概念及其应用，我还需要再努力去理解和掌握。