Assignment #8: 图论：概念、遍历，及 树算

Updated 1919 GMT+8 Apr 8, 2024

2024 spring, Complied by 胡登科 生科 2200012286

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：感觉对于图的理解还不是很明白，这周期中季结束一定好好看看。

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"）

#44674016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:
            return None
```

18160: 最大连通域面积

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

思路：这个题可以用图实现，但是不是很会写。看看了大佬的代码，却发现没有用图。

代码

```
#
```

```
count = 0
def dfs(x, y):
    if M[x + 1][y + 1] == "W":
        global count
        count += 1
        M[x + 1][y + 1] = "."
        for i in range(8):
            dfs(x+d[i][0],y+d[i][1])

T = int(input())
d = [[-1, -1], [-1, 0], [-1, 1],\
     [0, -1], [0, 1],\
     [1, -1], [1, 0], [1, 1]]

for i in range(T):
    n, m = map(int, input().split())
    M = [["." for _ in range(m + 2)] for _ in range(n + 2)]
    for i in range(n):
        string = input()
        for j in range(m):
            M[i + 1][j + 1] = string[j]

    ans = 0
    for i in range(n):
        for j in range(m):
            if M[i + 1][j + 1] == "W":
                dfs(i, j)
                ans = max(ans, count)
            count = 0
    print(ans)
```

代码运行截图 （至少包含有"Accepted"）

#44674140提交状态

状态: Accepted

源代码

```
#

count = 0


def dfs(x, y):
    if M[x + 1][y + 1] == "W":
        global count
        count += 1
        M[x + 1][y + 1] = "."
        for i in range(8):
            dfs(x + d[i][0], y + d[i][1])


T = int(input())
d = [[-1, -1], [-1, 0], [-1, 1], \
     [0, -1], [0, 1], \
     [1, -1], [1, 0], [1, 1]]

for i in range(T):
    n, m = map(int, input().split())
    M = [["." for _ in range(m + 2)] for _ in range(n + 2)]
    for i in range(n):
        string = input()
        for j in range(m):
            M[i + 1][j + 1] = string[j]

    ans = 0
    for i in range(n):
        for j in range(m):
            if M[i + 1][j + 1] == "W":
                dfs(i, j)
                ans = max(ans, count)
            count = 0
    print(ans)
```

sy383: 最大权值连通块

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

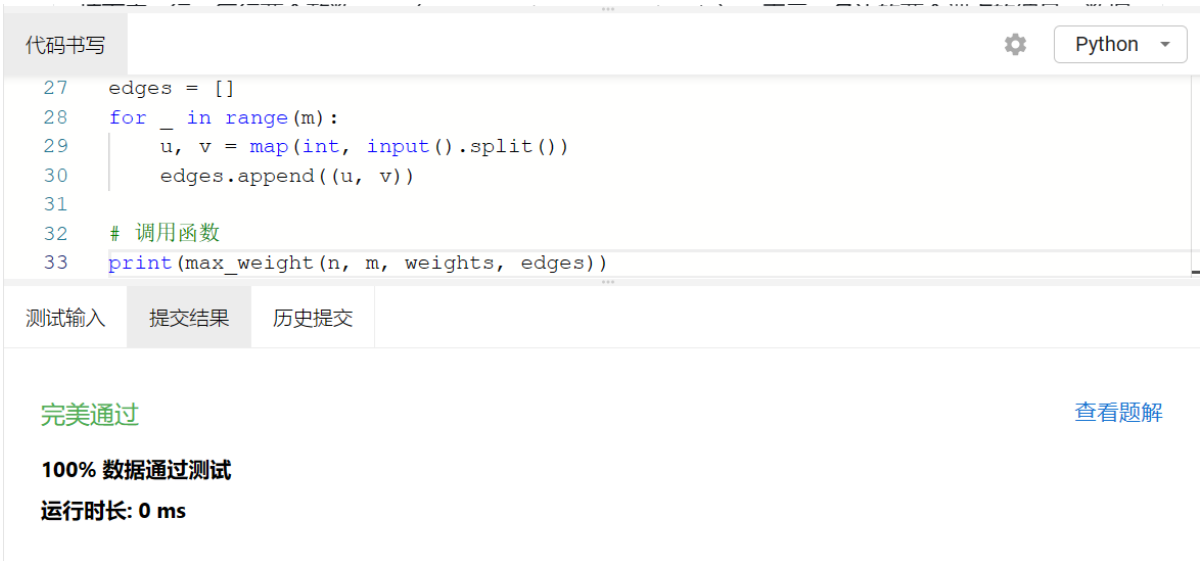
思路：对于dfs没有怎么掌握

代码

```
# def max_weight(n, m, weights, edges):
    graph = [[] for _ in range(n)]
    for u, v in edges:
        graph[u].append(v)
        graph[v].append(u)
```

```
    visited = [False] * n
    max_weight = 0

    def dfs(node):
        visited[node] = True
        total_weight = weights[node]
        for neighbor in graph[node]:
            if not visited[neighbor]:
                total_weight += dfs(neighbor)
        return total_weight

    for i in range(n):
        if not visited[i]:
            max_weight = max(max_weight, dfs(i))

    return max_weight

# 接收数据
n, m = map(int, input().split())
weights = list(map(int, input().split()))
edges = []
for _ in range(m):
    u, v = map(int, input().split())
    edges.append((u, v))

# 调用函数
print(max_weight(n, m, weights, edges))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



03441: 4 Values whose Sum is 0

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

思路:

第一次看见这么引入函数的。虽然很方便，但是时间复杂度比较高，不知道能不能用图的方法化简。

代码

```
#
from collections import Counter
from itertools import product

A, B, C, D = [], [], [], []

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

ab_sum_counter = Counter(map(sum, product(A, B)))
cn = 0
for cd_sum in map(sum, product(C, D)):
    cn += ab_sum_counter.get(-cd_sum, 0)

print(cn)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

状态: Accepted

源代码

```
#
from collections import Counter
from itertools import product

A, B, C, D = [], [], [], []

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

ab_sum_counter = Counter(map(sum, product(A, B)))
cn = 0
for cd_sum in map(sum, product(C, D)):
    cn += ab_sum_counter.get(-cd_sum, 0)

print(cn)
```


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:

理解了一遍Trie树

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"）

#44674329提交状态

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
```

04082: 树的镜面映射

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

思路:

终于找到一个可以理解的题目了 还是觉得树比较直观

代码

```
# from collections import deque


class Node:
    def __init__(self, name):
        self.name = name
        self.children = []
```

```
def create_node():
    return Node('')


def build_tree(line, index):
    node = create_node()
    fullname = line[index]
    node.name = fullname[0]
    if fullname[1] == '0' and node.name != '$':
        index += 1
        child, index = build_tree(line, index)
        node.children += child,
        index += 1
        child, index = build_tree(line, index)
        node.children += child,
    return node, index


def print_tree(root):
    queue, stack = deque(), deque()

    while root is not None:
        if root.name != '$':
            stack += root,
        root = root.children[1] if len(root.children) > 1 else None

    while stack:
        queue.append(stack.pop())

    while queue:
        root = queue.popleft()
        print(root.name, end=' ')

        if root.children:
            root = root.children[0]
            while root is not None:
                if root.name != '$':
                    stack += root,
                root = root.children[1] if len(root.children) > 1 else None

            while stack:
                queue.append(stack.pop())


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

root, _ = build_tree(line, 0)

print_tree(root)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

状态：Accepted

源代码

```
from collections import deque


class Node:
    def __init__(self, name):
        self.name = name
        self.children = []


def create_node():
    return Node('')


def build_tree(line, index):
    node = create_node()
    fullname = line[index]
    node.name = fullname[0]
    if fullname[1] == '0' and node.name != '$':
        index += 1
        child, index = build_tree(line, index)
        node.children += child,
        index += 1
        child, index = build_tree(line, index)
        node.children += child,
    return node, index


def print_tree(root):
    queue, stack = deque(), deque()

    while root is not None:
        if root.name != '$':
            stack += root,
        root = root.children[1] if len(root.children) > 1 else None
```

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。

这周是魔鬼期中季了，很难抽出时间打代码了。下周我会逐渐提升自己打代码的时间的。这些知识也会更加系统的学习的。相信自己，加油。