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==

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：

定义Vertex类，Graph类，然后实现

代码

```python
class Vertex:
    def __init__(self, key):
        self.id = key
        self.connected_to = {}
    def add_neighbor(self, nbr):
        self.connected_to[nbr] = 1

    def get_connections(self):
        return self.connected_to.keys()

    def degree(self):
        return(len(self.get_connections()))

class Graph:
    def __init__(self):
        self.vertex_list = {}

    def add_vertex(self, key):
        self.vertex_list[key] = Vertex(key)

    def get_vertices(self):
        return self.vertex_list.keys()

    def num_vertices(self):
        return len(self.get_vertices())

    def add_edge(self, i, j):
        if i not in self.vertex_list:
            self.add_vertex(i)
        if j not in self.vertex_list:
            self.add_vertex(j)
        self.vertex_list[i].add_neighbor(j)
        self.vertex_list[j].add_neighbor(i)

    def degree_matrix(self, i, j):
        return self.vertex_list[i].degree() * (i == j)

    def adjacent_matrix(self, i, j):
        return j in self.vertex_list[i].connected_to

    def laplacian_matrix(self, i, j):
        return self.degree_matrix(i, j) - self.adjacent_matrix(i, j)

n, m = map(int, input().split())
graph = Graph()
for i in range(n):
    graph.add_vertex(i)
for _ in range(m):
    graph.add_edge(*(int(x) for x in input().split()))

for i in range(n):
    print(*(graph.laplacian_matrix(i, j) for j in range(n)))
```

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

源代码

```python
#19943: 图的拉普拉斯矩阵

class Vertex:
    def __init__(self, key):
        self.id = key
        self.connected_to = {}

    def add_neighbor(self, nbr):
        self.connected_to[nbr] = 1

    def get_connections(self):
        return self.connected_to.keys()

    def degree(self):
        return(len(self.get_connections()))

class Graph:
    def __init__(self):
        self.vertex_list = {}

    def add_vertex(self, key):
        self.vertex_list[key] = Vertex(key)

    def get_vertices(self):
        return self.vertex_list.keys()

    def num_vertices(self):
        return len(self.get_vertices())

    def add_edge(self, i, j):
        if i not in self.vertex_list:
            self.add_vertex(i)
        if j not in self.vertex_list:
            self.add_vertex(j)
        self.vertex_list[i].add_neighbor(j)
        self.vertex_list[j].add_neighbor(i)

    def degree_matrix(self, i, j):
        return self.vertex_list[i].degree() * (i == j)

    def adjacent_matrix(self, i, j):
        return j in self.vertex_list[i].connected_to

    def laplacian_matrix(self, i, j):
        return self.degree_matrix(i, j) - self.adjacent_matrix(i, j)
```

18160: 最大连通域面积

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

思路：

dfs

代码

```
import itertools
T = int(input())
directions = [[-1, -1], [-1, 0], [-1, 1], [0, -1], [0, 1], [1, -1], [1, 0], [1, 1]]

def find_next(x, y):
    global board, result
    for dx, dy in directions:
        if board[x + dx][y + dy] == 1:
            board[x + dx][y + dy] = 2
            result += 1
            find_next(x + dx, y + dy)

for _ in range(T):
    N, M = map(int, input().split())
    board = [[0 for _ in range(M + 2)]] + [[0] + list(map(lambda x: 1 if x == 'W' else 0,
input())) + [0] for _ in range(N)] + [[0 for _ in range(M + 2)]]
    m = 0
    for i, j in itertools.product(range(1, N + 1), range(1, M + 1)):
        if board[i][j] == 1:
            result = 1
            board[i][j] = 2
            find_next(i, j)
            m = max(m, result)
    print(m)
```

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

源代码

```
import itertools
T = int(input())
directions = [[-1, -1], [-1, 0], [-1, 1], [0, -1], [0, 1], [1, -1], [1,

def find_next(x, y):
    global board, result
    for dx, dy in directions:
        if board[x + dx][y + dy] == 1:
            board[x + dx][y + dy] = 2
            result += 1
            find_next(x + dx, y + dy)

for _ in range(T):
    N, M = map(int, input().split())
    board = [[0 for _ in range(M + 2)]] + [[0] + list(map(lambda x: 1 i
    m = 0
    for i, j in itertools.product(range(1, N + 1), range(1, M + 1)):
        if board[i][j] == 1:
            result = 1
            board[i][j] = 2
            find_next(i, j)
            m = max(m, result)
    print(m)
```

sy383: 最大权值连通块

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

思路:

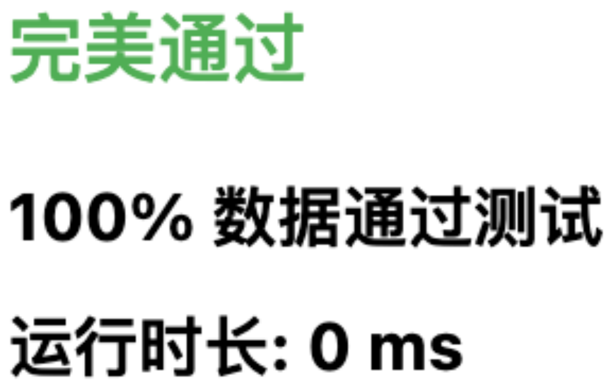
遍历

代码

```
class Vertex:
    def __init__(self, key, weight):
        self.id = key
        self.weight = weight
        self.connected_to = {}
        self.traversed = 0

    def add_neighbor(self, nbr):
```

```python
        self.connected_to[nbr] = 1

    def get_connections(self):
        return self.connected_to.keys()

    def degree(self):
        return(len(self.get_connections()))

class Graph:
    def __init__(self):
        self.vertex_list = {}

    def add_vertex(self, key, weight):
        self.vertex_list[key] = Vertex(key, weight)

    def get_vertices(self):
        return self.vertex_list.keys()

    def num_vertices(self):
        return len(self.get_vertices())

    def add_edge(self, i, j):
        self.vertex_list[i].add_neighbor(j)
        self.vertex_list[j].add_neighbor(i)

    def traverse(self, vert):
        if self.vertex_list[vert].traversed:
            return 0
        res = self.vertex_list[vert].weight
        self.vertex_list[vert].traversed = 1
        for nbr in self.vertex_list[vert].get_connections():
            res += self.traverse(nbr)
        return res

n, m = map(int, input().split())
graph = Graph()
for index, weight in enumerate(input().split()):
    graph.add_vertex(index, int(weight))
for _ in range(m):
    i, j = map(int, input().split())
    graph.add_edge(i, j)
print(max(map(graph.traverse, range(n))))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



完美通过

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：

双指针查找，注意边界条件

代码

```python
from itertools import product

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

for _ in range(n):
    i, j, k, l = map(int, input().split())
    a.append(i)
    b.append(j)
    c.append(k)
    d.append(l)
left = sorted(list(map(sum, product(a, b))))
right = sorted(list(map(lambda x: -sum(x), product(c, d))))

res = 0
lpointer = len(left) - 1
rpointer = len(right) - 1
same_flag = 0

while lpointer >= 0 and rpointer >= 0:
    if rpointer - same_flag < 0:
        if same_flag != 0:
            same_flag = 0
        lpointer -= 1
        continue
    if right[rpointer - same_flag] > left[lpointer]:
        rpointer -= 1
        continue
    if right[rpointer - same_flag] < left[lpointer]:
        if same_flag != 0:
            same_flag = 0
        lpointer -= 1
        continue
    if right[rpointer - same_flag] == left[lpointer]:
        same_flag += 1
        res += 1

print(res)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
#03441:4 Values whose Sum is 0
from itertools import product

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

for _ in range(n):
    i, j, k, l = map(int, input().split())
    a.append(i)
    b.append(j)
    c.append(k)
    d.append(l)
left = sorted(list(map(sum, product(a, b))))
right = sorted(list(map(lambda x: -sum(x), product(c, d))))

res = 0
lpointer = len(left) - 1
rpointer = len(right) - 1
same_flag = 0

while lpointer >= 0 and rpointer >= 0:
    if rpointer - same_flag < 0:
        if same_flag != 0:
            same_flag = 0
        lpointer -= 1
        continue
    if right[rpointer - same_flag] > left[lpointer]:
        rpointer -= 1
        continue
    if right[rpointer - same_flag] < left[lpointer]:
        if same_flag != 0:
            same_flag = 0
        lpointer -= 1
        continue
    if right[rpointer - same_flag] == left[lpointer]:
        same_flag += 1
        res += 1

print(res)
```

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：

注意考虑到输入完全相同的情况。

代码

```python
# 04089:电话号码

class Node:
    def __init__(self, is_root=False, key=None):
        if not is_root:
            self.key = key
        self.children = {}
        self.is_end_of_word = 0

    def add_char(self, char):
        if char not in self.children:
            self.children[char] = Node(key=char)
        return self.children[char]

    def end_of_word(self):
        self.is_end_of_word += 1

class Trie:
    def __init__(self):
        self.root_node = Node(is_root=True)
        self.words = {}

    def add_word(self, word):
        node_now = self.root_node
        for char in word:
            node_now = node_now.add_char(char)
        node_now.end_of_word()
        self.words[word] = node_now

    def is_consistent(self):
        for word in self.words.values():
            if (word.is_end_of_word == 1 and word.children != {}) or word.is_end_of_word >
1:
                return False
        return True

t = int(input())
for _ in range(t):
    trie = Trie()
    n = int(input())
    for _ in range(n):
        trie.add_word(input())
    print("YES" if trie.is_consistent() else "NO")
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
# 04089:电话号码

class Node:
    def __init__(self, is_root=False, key=None):
        if not is_root:
            self.key = key
        self.children = {}
        self.is_end_of_word = 0

    def add_char(self, char):
        if char not in self.children:
            self.children[char] = Node(key=char)
        return self.children[char]

    def end_of_word(self):
        self.is_end_of_word += 1

class Trie:
    def __init__(self):
        self.root_node = Node(is_root=True)
        self.words = {}

    def add_word(self, word):
        node_now = self.root_node
        for char in word:
            node_now = node_now.add_char(char)
        node_now.end_of_word()
        self.words[word] = node_now

    def is_consistent(self):
        for word in self.words.values():
            if (word.is_end_of_word == 1 and word.children != {}) or wo
                return False
        return True

t = int(input())
for _ in range(t):
    trie = Trie()
    n = int(input())
    for _ in range(n):
        trie.add_word(input())
    print("YES" if trie.is_consistent() else "NO")
```

04082: 树的镜面映射

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

思路：

右儿子节点为同一层。左儿子节点为下一层，反向输出即可。

代码

```python
class Node:
    _ID=0
    id:int
    name:str
    sub:list

    def __init__(self, name, sub):
        self.id = self.__class__._ID
        self.__class__._ID += 1
        self.name = name
        self.sub = sub
    def isFake(self):
        return self.name == '$'

def levelOrder(root: Node):
    l = [[]]
    def pseudoLeverParser(node: Node, l: list, level: int):
        if node is None or node.isFake(): return
        try:
            l[level].append(node)
        except IndexError:
            l.append([node])
        pseudoLeverParser(node.sub[0], l, level + 1)
        pseudoLeverParser(node.sub[1], l, level)
    pseudoLeverParser(root, l, 0)
    return ' '.join(''.join([''.join([y.name for y in reversed(x)]) for x in l]))

n = int(input())
l = input().split()
stack = []
root = None
for x in l[::-1]:
    name, nodetype = x[0], int(x[1])
    if nodetype:
        stack.append(Node(name, [None, None]))
    else:
        node1 = stack.pop()
        node2 = stack.pop()
        neonode = Node(name, [node1, node2])
        stack.append(neonode)
        root = neonode
```

```
ans = levelOrder(root)
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

```
class Node:
    _ID=0
    id:int
    name:str
```

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。==

复习了计概时学习到的图的方法，并且这次用更加面向对象的方法实现了。