

# Assignment #8: 图论：概念、遍历，及 树算

Updated 1919 GMT+8 Apr 8, 2024

2024 spring, Compiled by 陈奕好 工学院

## 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：macOS Sonoma 14.4 (23E214)

Python编程环境：PyCharm 2023.3.1 (Professional Edition)

## 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：先定义Vertex类存储点，这个点存储使用了buffer。再定义Graph类初始化每行。  
main还是初始化一个0矩阵，在零矩阵里操作更加便捷。

## 代码

```
1 class Vertex:
2     def __init__(self):
3         self.edges = {}
4
5     def add_edge(self, vertex):
6         # vertex buffer
7         if vertex in self.edges:
```

```

8         self.edges[vertex] += 1
9     else:
10         self.edges[vertex] = 1
11
12
13 class Graph:
14     def __init__(self, num_vertices):
15         self.vertices = {i: Vertex() for i in range(num_vertices)}
16
17     def add_edge(self, start, end):
18         self.vertices[start].add_edge(end)
19         self.vertices[end].add_edge(start)
20
21
22 n, m = map(int, input().split())
23 graph = Graph(n)
24
25 for i in range(m):
26     start, end = map(int, input().split())
27     graph.add_edge(start, end)
28
29 for vertex, data in graph.vertices.items():
30     line = [0]*n
31     for connected_vertex, weight in data.edges.items():
32         line[vertex] += weight
33         line[connected_vertex] -= weight
34     print(*line)
35
36

```

代码运行截图 (至少包含有"Accepted")

#44665345提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: **Accepted**

源代码

```

class Vertex:
    def __init__(self):
        self.edges = {}

    def add_edge(self, vertex):
        # vertex buffer
        if vertex in self.edges:
            self.edges[vertex] += 1
        else:
            self.edges[vertex] = 1

class Graph:
    def __init__(self, num_vertices):
        self.vertices = {i: Vertex() for i in range(num_vertices)}

    def add_edge(self, start, end):
        self.vertices[start].add_edge(end)
        self.vertices[end].add_edge(start)

```

基本信息

#: 44665345  
 题目: 19943  
 提交人: 23n2300011030(陈奕好)  
 内存: 3640kB  
 时间: 27ms  
 语言: Python3  
 提交时间: 2024-04-15 17:52:28

## 18160: 最大连通域面积

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

思路：N,M又看反了，而且python3.8卡我变量。

代码

```
1 size = 0
2 def dfs(x, y, matrix, neighbors, N, M):
3     global size
4     stack = [(x, y)]
5     while stack:
6         x, y = stack.pop()
7         if matrix[x][y] == "W":
8             size += 1
9             matrix[x][y] = "."
10            for dx, dy in neighbors:
11                nx, ny = x + dx, y + dy
12                if isvalid(nx, ny, N, M) and matrix[nx][ny] == "W":
13                    stack.append((nx, ny))
14
15
16 def isvalid(x, y, N, M):
17     return x >= 0 and x < N and y >= 0 and y < M
18
19
20 # Adjust the call to dfs in the solve function
21 def solve(N, M, matrix):
22     global size
23     largest_size = 0
24     neighbors = [(0, 1), (0, -1), (1, 0), (-1, 0), (1, 1), (-1, -1), (1, -1), (-1,
25 1)]
26     for i in range(N):
27         for j in range(M):
28             if matrix[i][j] == "W":
29                 dfs(i, j, matrix, neighbors, N, M)
30                 largest_size = max(largest_size, size)
31                 size = 0
32     return largest_size
33
34 T = int(input())
35 for turn in range(T):
36     size = 0
37     N, M = map(int, input().split())
38     graph = [list(map(str, input())) for _ in range(N)]
```

```

39     print(solve(N, M, graph))
40
41     """
42
43     temp = 0
44     def search(i,j):
45         global temp
46         temp += 1
47         matrix[i][j] = "."
48         for p in dfs:
49             if matrix[i+p[0]][j+p[1]] == "W":
50                 search(i+p[0],j+p[1])
51
52
53     dfs = [(-1,-1),(-1,0),(-1,1),(0,-1),(0,1),(1,-1),(1,0),(1,1)]
54
55     T = int(input())
56     for _ in range(T):
57         maxium = 0
58         N,M = map(int,input().split())
59         matrix = [ "."*(M+2) ]
60         for i in range(N):
61             matrix.append( "."+list(input())+"." )
62         matrix.append( "."*(M+2) )
63         #print(matrix)
64         for i in range(1,N+1):
65             for j in range(1,M+1):
66                 if matrix[i][j] == "W":
67                     temp = 0
68                     search(i,j)
69                     maxium = max(maxium,temp)
70         print(maxium)
71     """
72

```

代码运行截图 (至少包含有"Accepted")

状态: Accepted

源代码

```
size = 0

def dfs(x, y, matrix, neighbors, N, M):
    global size
    stack = [(x, y)]
    while stack:
        x, y = stack.pop()
        if matrix[x][y] == "W":
            size += 1
            matrix[x][y] = "."
            for dx, dy in neighbors:
                nx, ny = x + dx, y + dy
                if isvalid(nx, ny, N, M) and matrix[nx][ny] == "W":
                    stack.append((nx, ny))

def isvalid(x, y, N, M):
    return x >= 0 and x < N and y >= 0 and y < M

# Adjust the call to dfs in the solve function
```

基本信息

#: 44665883  
题目: 18160  
提交人: 23n2300011030(陈奕好)  
内存: 3728kB  
时间: 109ms  
语言: Python3  
提交时间: 2024-04-15 18:57:05

## sy383: 最大权值连通块

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

思路: disjset的运用

代码

```
1 class DisjSet:
2     def __init__(self, n):
3         # Constructor to create and
4         # initialize sets of n items
5         self.rank = [1] * n
6         self.parent = [i for i in range(n)]
7         self.weights = [0] * n
8
9     def find(self, x):
10        # Find the root of the set in which element x belongs
11        if self.parent[x] != x:
12            # Path compression: Make the parent of x the root of its set
13            self.parent[x] = self.find(self.parent[x])
14        return self.parent[x]
15
16    def union(self, x, y):
17        # Perform union of two sets
18        x_root, y_root = self.find(x), self.find(y)
```

```

19
20     if x_root == y_root:
21         return
22     # Attach smaller rank tree under root of higher rank tree
23     if self.rank[x_root] < self.rank[y_root]:
24         self.parent[x_root] = y_root
25         self.weights[y_root] += self.weights[x_root]
26     else:
27         self.parent[y_root] = x_root
28         self.weights[x_root] += self.weights[y_root]
29         if self.rank[x_root] == self.rank[y_root]:
30             self.rank[x_root] += 1
31
32
33 n, m = map(int, input().split())
34 ds = DisjSet(n)
35 weights = list(map(int, input().split()))
36 ds.weights = weights.copy()
37 for i in range(m):
38     u, v = map(int, input().split())
39     ds.union(u, v)
40
41 max_weight = max(ds.weights)
42 print(max_weight)
43

```

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：这里思路比较淳朴，就是字典。

代码

```
1 n = int(input())
2 A, B, C, D = [], [], [], []
3 for i in range(n):
4     a, b, c, d = map(int, input().split())
5     A.append(a)
6     B.append(b)
7     C.append(c)
8     D.append(d)
9 AB = {}
10 for a in A:
11     for b in B:
12         if a + b not in AB:
13             AB[a + b] = 1
14         else:
15             AB[a + b] += 1
16
17 count = 0
18 for c in C:
19     for d in D:
20         if -c - d in AB:
21             count += AB[-c - d]
22 print(count)
23
```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

#44669650提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: Accepted

源代码

```
n = int(input())
A, B, C, D = [], [], [], []
for i in range(n):
    a, b, c, d = map(int, input().split())
    A.append(a)
    B.append(b)
    C.append(c)
    D.append(d)
AB = {}
for a in A:
    for b in B:
        if a + b not in AB:
            AB[a + b] = 1
        else:
            AB[a + b] += 1

count = 0
for c in C:
    for d in D:
        if -c - d in AB:
            count += AB[-c - d]
print(count)
```

基本信息

#: 44669650  
题目: 03441  
提交人: 23n2300011030(陈奕好)  
内存: 171772kB  
时间: 4019ms  
语言: Python3  
提交时间: 2024-04-15 23:39:14

## 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：

**字典树（前缀树，Trie）**：字典树是一种树形数据结构，用于高效地存储和检索字符串数据集中的键。如果你使用嵌套的字典来表示字典树，其中每个字典代表一个节点，键表示路径上的字符，而值表示子节点，那么就构成了字典树。例如：

```
1 trie = {
2     'a': {
3         'p': {
4             'p': {
5                 'l': {
6                     'e': {'is_end': True}
7                 }
8             }
9         }
10    },
11    'b': {
12        'a': {
13            'l': {
14                'l': {'is_end': True}
15            }
16        }
17    },
18    'c': {
19        'a': {
20            't': {'is_end': True}
21        }
22    }
23 }
```

这样的表示方式使得我们可以非常高效地搜索和插入字符串，特别是在大型数据集上。

代码

```
1 class TrieNode:
2     def __init__(self):
3         self.children = {}
4         self.end_of_word = False
5
6 class Trie:
7     def __init__(self):
```



```

8         self.root = TrieNode()
9
10    def insert(self, word):
11        node = self.root
12        for char in word:
13            if char not in node.children:
14                node.children[char] = TrieNode()
15            node = node.children[char]
16        node.end_of_word = True
17
18    def search(self, word):
19        node = self.root
20        for char in word:
21            if char not in node.children:
22                return False
23            node = node.children[char]
24        return node.end_of_word
25
26    def is_prefix(self, word):
27        node = self.root
28        for char in word:
29            if char not in node.children:
30                return False
31            elif node.children[char].end_of_word:
32                return True
33            node = node.children[char]
34        return False
35
36
37    t = int(input())
38    for _ in range(t):
39        n = int(input())
40        phone_numbers = [input() for _ in range(n)]
41        phone_numbers.sort()
42        trie = Trie()
43        consistent = True
44
45        for phone in phone_numbers:
46            if trie.is_prefix(phone):
47                consistent = False
48                break
49        trie.insert(phone)
50    if consistent:
51        print("YES")
52    else:
53        print("NO")

```

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.children = {}
        self.end_of_word = False

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

    def insert(self, word):
        node = self.root
        for char in word:
            if char not in node.children:
                node.children[char] = TrieNode()
            node = node.children[char]
        node.end_of_word = True

    def search(self, word):
        node = self.root
        for char in word:
            if char not in node.children:
                return False
            node = node.children[char]
        return node.end_of_word
```

基本信息

#: 44670693  
题目: 04089  
提交人: 23n2300011030(陈奕好)  
内存: 24744kB  
时间: 300ms  
语言: Python3  
提交时间: 2024-04-16 08:51:50

## 04082: 树的镜面映射

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

思路：做了很久，表示不出来。呼之欲出，求之不得。

代码

```
1  from collections import deque
2
3
4  class TreeNode:
5      def __init__(self, x):
6          self.x = x # 节点值
7          self.children = [] # 子节点
8
9
10 def create_node(): # 创建节点
11     return TreeNode('')
12
13
14 def build_tree(tempList, index): # 构建多叉树 index为当前节点在tempList中的索引
15     node = create_node() # 创建节点
16     node.x = tempList[index][0] # 节点值
17     if tempList[index][1] == '0' and node.x != '$': # 如果节点值不为'$'且有子节点
```

```

18         index += 1
19         child, index = build_tree(tempList, index) # 递归构建子节点
20         node.children.append(child) # 添加子节点
21         index += 1
22         child, index = build_tree(tempList, index) # 递归构建子节点
23         node.children.append(child) # 添加子节点
24     return node, index # 返回当前节点及下一个节点的索引
25
26
27 def print_tree(p): # 宽度优先遍历并打印镜像映射序列
28     Q = deque() # 队列Q
29     s = deque() # 栈s
30
31     # 遍历右子节点并将非虚节点加入栈s
32     while p is not None:
33         if p.x != '$':
34             s.append(p)
35         p = p.children[1] if len(p.children) > 1 else None # 右子节点
36
37     # 将栈s中的节点逆序放入队列Q
38     while s:
39         Q.append(s.pop())
40
41     # 宽度优先遍历队列Q并打印节点值
42     while Q:
43         p = Q.popleft()
44         print(p.x, end=' ')
45
46     # 如果节点有左子节点，将左子节点及其右子节点加入栈s
47     if p.children:
48         p = p.children[0]
49         while p is not None:
50             if p.x != '$':
51                 s.append(p)
52             p = p.children[1] if len(p.children) > 1 else None
53
54     # 将栈s中的节点逆序放入队列Q
55     while s:
56         Q.append(s.pop())
57
58     # 读取输入
59     n = int(input())
60     tempList = input().split(' ')
61
62     # 构建二叉树
63     root, _ = build_tree(tempList, 0)
64
65     # 执行宽度优先遍历并打印镜像映射序列
66     print_tree(root)
67

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

#44673079提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: Accepted

源代码

```
from collections import deque

class TreeNode:
    def __init__(self, x):
        self.x = x # 节点值
        self.children = [] # 子节点

def create_node(): # 创建节点
    return TreeNode('')

def build_tree(tempList, index): # 构建二叉树 index为当前节点在tempList中的
    node = create_node() # 创建节点
    node.x = tempList[index][0] # 节点值
    if tempList[index][1] == '0' and node.x != '$': # 如果节点值不为'$'且
        index += 1
        child, index = build_tree(tempList, index) # 递归构建子节点
        node.children.append(child) # 添加子节点
        index += 1
        child, index = build_tree(tempList, index) # 递归构建子节点
        node.children.append(child) # 添加子节点
    return node, index # 返回当前节点及下一个节点的索引

def print_tree(p): # 宽度优先遍历并打印镜像映射序列
    q = deque() # 队列q
```

基本信息

#: 44673079  
题目: 04082  
提交人: 23n2300011030(陈奕好)  
内存: 6472kB  
时间: 30ms  
语言: Python3  
提交时间: 2024-04-16 15:07:03

## 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。

期中寄也结束了, 回归刷题模式 \(\text{O}u\text{O}\)/