

Assignment #8: 图论：概念、遍历，及 树算

Updated 1919 GMT+8 Apr 8, 2024

2024 spring, Complied by 田济维 物理学院

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

(python)

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：

代码

```
1  #
2  class Vertex:
3      def __init__(self, key):
4          self.id = key
5          self.connectedto = {}
6
7      def addNeighbor(self, nbr, weight=0):
```

```

8         self.connectedto[nbr]=weight
9
10    class Graph:
11        def __init__(self):
12            self.vertList = {}
13            self.vertnum = 0
14
15        def addVertex(self,key):
16            self.vertnum+=1
17            newVertex = Vertex(key)
18            self.vertList[key]=newVertex
19            return newVertex
20
21        def addEdge(self,f,t,weight = 0):
22            if f not in self.vertList:
23                self.addVertex(f)
24            if t not in self.vertList:
25                self.addVertex(t)
26            self.vertList[f].addNeighbor(t,weight)
27            self.vertList[t].addNeighbor(f,weight)
28
29    n,m = map(int,input().split())
30    graph = Graph()
31    for i in range(n):
32        graph.addVertex(i)
33    for i in range(m):
34        f,t = map(int,input().split())
35        graph.addEdge(f,t)
36    Laplace = [[0]*n for i in range(n)]
37    for i in range(n):
38        for j in graph.vertList[i].connectedto:
39            Laplace[i][j]-=1
40            Laplace[i][i]+=1
41
42    for i in range(n):
43        print(" ".join(map(str,Laplace[i])))
44
45
46

```

代码运行截图 (至少包含有"Accepted")

状态: Accepted

基本

源代码

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

    def addNeighbor(self, nbr, weight=0):
        self.connectedto[nbr]=weight

class Graph:
    def __init__(self):
        self.vertList = {}
        self.vertnum = 0

    def addVertex(self, key):
        self.vertnum+=1
        newVertex = Vertex(key)
        self.vertList[key]=newVertex
        return newVertex

    def addEdge(self, f, t, weight = 0):
```

提

提交

18160: 最大连通域面积

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

思路:

代码

```
1  #
2  from collections import deque
3  tx = [1,0,0,-1,1,1,-1,-1]
4  ty = [0,1,-1,0,1,-1,1,-1]
5  def bfs(m,n):
6      cnt = 0
7      queue = deque([(m,n)])
8      while queue:
9          sx,sy = queue.popleft()
10         cnt+=1
11         for i in range(8):
12             if Map[sx+tx[i]][sy+ty[i]] == "W" :
13
14                 queue.append((sx+tx[i],sy+ty[i]))
15                 Map[sx + tx[i]][sy + ty[i]] = "."
16         return cnt
17
18
19
20
21 T = int(input())
```

```

22 for _ in range(T):
23     N,M = map(int,input().split())
24     Map = [ "."*(M+2) for i in range(N+2)]
25     for i in range(N):
26         Map[i+1][1:-1]=input()
27     maxn = 0
28     for i in range(1,N+1):
29         for j in range(1,M+1):
30             if Map[i][j]=="w":
31                 Map[i][j]="."
32                 maxn = max(maxn,bfs(i,j))
33     print(maxn)
34
35
36

```

代码运行截图 (至少包含有"Accepted")

状态: Accepted

源代码

```

from collections import deque
tx = [1,0,0,-1,1,1,-1,-1]
ty = [0,1,-1,0,1,-1,1,-1]
def bfs(m,n):
    cnt = 0
    queue = deque([(m,n)])
    while queue:
        sx,sy = queue.popleft()
        cnt+=1
        for i in range(8):
            if Map[sx+tx[i]][sy+ty[i]] == "W" :

                queue.append((sx+tx[i],sy+ty[i]))
                Map[sx + tx[i]][sy + ty[i]] = "."
    return cnt

```

sy383: 最大权值连通块

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

思路:

代码

```

1  #
2  graph = {}
3  n,m = map(int,input().split())
4  weight = list(map(int,input().split()))
5  for i in range(n):
6      graph[i]=[]
7  for i in range(m):
8      f,t = map(int,input().split())
9      graph[f].append(t)
10     graph[t].append(f)
11
12     visited = [False for i in range(n)]
13     def dfs(vert):
14         cnt = 0
15         visited[vert]=True
16         pstack = [vert]
17         while pstack:
18             s = pstack.pop()
19             cnt +=weight[s]
20             for x in graph[s]:
21                 if visited[x]==False:
22                     visited[x]=True
23                     pstack.append(x)
24         return cnt
25     maxn = 0
26     for i in range(n):
27         if visited[i]==False:
28             maxn = max(maxn,dfs(i))
29     print(maxn)
30
31

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

```

1 graph = {}
2 n,m = map(int,input().split())
3 weight = list(map(int,input().split()))
4 for i in range(n):
5     graph[i]=[]
6 for i in range(m):
7     f,t = map(int,input().split())
8     graph[f].append(t)
9     graph[t].append(f)
10
11 visited = [False for i in range(n)]
12 def dfs(vert):
13     cnt = 0
14     visited[vert]=True
15     pstack = [vert]
16     while pstack:
17         s = pstack.pop()

```

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:

这题内存空间卡的太死了，一开始用的defaultdict就刚好过不了，用了dict加判断内存空间就恰好过了

代码

```

1 #
2
3 n=int(input())
4 num = [[0]*n for i in range(4)]
5 for i in range(n):
6     s = list(map(int,input().split()))
7     for j in range(4):
8         num[j][i]=s[j]
9
10 CD={}
11 for i in range(n):
12     for j in range(n):
13         if num[2][i]+num[3][j] not in CD:
14             CD[num[2][i] + num[3][j]] =1
15         else:

```

```

16         CD[num[2][i]+num[3][j]]+=1
17     cnt =0
18     for i in range(n):
19         for j in range(n):
20             if -(num[0][i]+num[1][j]) in CD:
21                 cnt+=CD[-(num[0][i]+num[1][j])]
22     print(cnt)

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

状态: **Accepted**

源代码

```

n =int(input())
num = [[0]*n for i in range(4)]
for i in range(n):
    s = list(map(int,input().split()))
    for j in range(4):
        num[j][i]=s[j]

CD={}
for i in range(n):

```

基本信息

#: 44663687
 题目: 03441
 提交人: 23n2300011503
 内存: 171624kB
 时间: 5265ms
 语言: Python3
 提交时间: 2024-04-15 15:53:01

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

```

1  #
2  class TrieNode:
3      def __init__(self):
4          self.nodes = {}
5          self.is_leaf = False
6          self.flag = True
7
8
9      def insert(self,word):
10         cur = self
11         n = len(word)
12         for i in range(n):
13             c = word[i]
14             if c not in cur.nodes:
15                 cur.nodes[c]=TrieNode()
16             elif c in cur.nodes:
17                 if cur.nodes[c].is_leaf == True or i == n-1:
18                     self.flag = False
19
20

```

```

21
22
23         cur = cur.nodes[c]
24         cur.is_leaf=True
25 t = int(input())
26 for _ in range(t):
27     n = int(input())
28     dial = TrieNode()
29     for i in range(n):
30         dial.insert(input())
31     #深度优先搜索
32     if dial.flag:
33         print("YES")
34     else:
35         print("NO")
36

```

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.nodes = {}
        self.is_leaf = False
        self.flag = True

    def insert(self, word):
        cur = self
        n = len(word)
        for i in range(n):
            c = word[i]
            if c not in cur.nodes:
                cur.nodes[c] = TrieNode()
            elif c in cur.nodes:
                if cur.nodes[c].is_leaf == True or i == n-1:
                    self.flag = False

```

基本信息

#: 44664716
 题目: 04089
 提交人: 23n230001
 内存: 24260kB
 时间: 438ms
 语言: Python3
 提交时间: 2024-04-15

04082: 树的镜面映射

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

思路:

代码

```

1 #
2 from collections import deque
3 class TreeNode:
4     def __init__(self, x):
5         self.x = x

```



```

6         self.children = []
7         #此函数要达到的效果是构建好此节点开始的子树并且给出结束的index
8         def buildTree(string,index):
9             node = TreeNode(string[index][0])
10            if string[index][1]=="0":
11                #先考虑此节点的左子树
12                index +=1
13                nd,index = buildTree(string,index)
14                node.children.append(nd)
15                #再构造此节点的右子树
16                index+=1
17                nd,index = buildTree(string,index)
18                node.children.append(nd)
19            #说明此点无子树，直接返回即可
20            return node,index
21
22        def traverse(tree):
23            #由于左儿子右兄弟，先朝右走到底
24            queue = deque([tree])
25            temp = deque([])
26            while queue:
27                s = queue.popleft()
28                print(s.x,end = " ")
29                if len(s.children)>1:
30                    p = s.children[0]
31                    while p:
32                        if p.x != "$":
33                            temp.append(p)
34                            if len(p.children)>1:
35                                p = p.children[1]
36                            else:
37                                p = None
38                    while temp:
39                        queue.append(temp.pop())
40
41        n = int(input())
42        string = input().split()
43        tree,i = buildTree(string,0)
44        traverse(tree)
45

```

代码运行截图 (AC代码截图，至少包含有"Accepted")

状态: Accepted

害

源代码

```
from collections import deque
class TreeNode:
    def __init__(self, x):
        self.x = x
        self.children = []
    #此函数要达到的效果是构建好此节点开始的子树并且给出结束的index
def buildTree(string, index):
    node = TreeNode(string[index][0])
    if string[index][1]!="0":
        #先考虑此节点的左子树
        index +=1
        nd, index = buildTree(string, index)
        node.children.append(nd)
        #再构造此节点的右子树
        index+=1
        nd, index = buildTree(string, index)
        node.children.append(nd)
    #说明此点无子树, 直接返回即可
    return node, index

def traverse(tree):
```

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。

收获1: 学习了字典树

收获2: 最后一题没想到返回两个量从而实现迭代, 最后一题好恶心