Assignment #8: 图论：概念、遍历，及 树算

Updated 1919 GMT+8 Apr 8, 2024

2024 spring, Complied by 夏天 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 家庭版

Python编程环境：Spyder （python 3.11）

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：之前做这道题的时候是直接利用列表嵌套存了度数矩阵，邻接矩阵，然后两个矩阵对应元素相减得到拉普拉斯矩阵。这次要求定义类来写，算是正式开启图的系统学习的标志吧

代码

```
class Vertex:
    def __init__(self,value):
        self.value=value
        self.neighbor=[]
    def add_neighbor(self,other):
        self.neighbor.append(other)
class Graph:
    def __init__(self,n):
        self.graph=[Vertex(_)for _ in range(n)]
    def add_edge(self,a,b):
        self.graph[a].add_neighbor(b)
        self.graph[b].add_neighbor(a)
    def laplacian_matrix(self):
        n=len(self.graph)
        matrix=[[0]*n for _ in range(n)]
        for i in range(n):
            for j in range(n):
                if i==j:
                    matrix[i][j]=len(self.graph[i].neighbor)
                elif j in self.graph[i].neighbor:
                    matrix[i][j]=-1
        return matrix
n,m=map(int,input().split())
graph=Graph(n)
for _ in range(m):
    a,b=map(int,input().split())
    graph.add_edge(a,b)
matrix=graph.laplacian_matrix()
for _ in matrix:
    print(*_)
```

代码运行截图 （至少包含有"Accepted"）

18160: 最大连通域面积

matrix/dfs similar, http://cs101.

openjudge.cn/practice/18160

思路：上学期做这道题的时候第一次接触dfs，现在用递归写dfs已经很熟练了

代码

```
def dfs(a,b,graph,visited):
    if a<0 or b<0 or a>=N or b>=M:
        return 0
    if graph[a][b]!='W'or visited[a][b]:
        return 0
    visited[a][b]=True
    area=1
    for dx in [-1,0,1]:
        for dy in [-1,0,1]:
            if dx==dy==0:
                continue
            area+=dfs(a+dx,b+dy,graph,visited)
    return area
T=int(input())
for _ in range(T):
    N,M=map(int,input().split())
    graph=[input() for i in range(N)]
    visited=[[False]*M for i in range(N)]
    max_area=0
    for x in range(N):
        for y in range(M):
            area=dfs(x,y,graph,visited)
            max_area=max(area,max_area)
    print(max_area)
```

代码运行截图 （至少包含有"Accepted"）

sy383: 最大权值连通块

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

思路：感觉像前两道题的结合：用第一题的方法由数据建图，用第二题的方法进行图搜索

代码

```
class Vertex:
    def __init__(self,value,weight):
        self.value=value
        self.weight=weight
        self.neighbor=[]
    def add_neighbor(self,other):
        self.neighbor.append(other)
class Graph:
    def __init__(self,weights):
        self.graph=[Vertex(i,weights[i])for i in range(len(weights))]
    def add_edge(self,a,b):
        self.graph[a].add_neighbor(b)
        self.graph[b].add_neighbor(a)
def total_weight(i,graph,visited):
    if visited[i]:
        return 0
    total=weights[i]
    visited[i]=True
    if graph.graph[i].neighbor:
        for j in graph.graph[i].neighbor:
            total+=total_weight(j,graph,visited)
    return total
n,m=map(int,input().split())
weights=list(map(int,input().split()))
graph=Graph(weights)
visited=[False]*n
for _ in range(m):
    a,b=map(int,input().split())
    graph.add_edge(a,b)
max_total_weight=0
for i in range(n):
    total=total_weight(i,graph,visited)
    max_total_weight=max(total,max_total_weight)
print(max_total_weight)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

03441: 4 Values whose Sum is 0

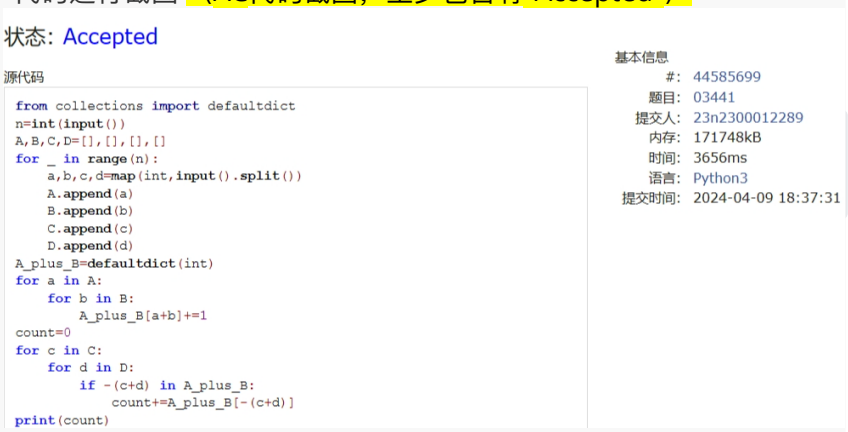
data structure/binary search, http://cs101.openjudge.cn/practice/03441

思路：最直接的想法是四重循环找所有满足条件的(a,b,c,d)，但时间复杂度是o(n^4),大概率会TLE.容易想到的一种优化方法是分成两组进行两次二重循环，这样时间复杂度就降低到o(n^2),同时利用哈希表存储a+b每个结果的出现次数，以提高查找效率

代码

```
from collections import defaultdict
n=int(input())
A,B,C,D=[],[],[],[]
for _ in range(n):
    a,b,c,d=map(int,input().split())
    A.append(a)
    B.append(b)
    C.append(c)
    D.append(d)
A_plus_B=defaultdict(int)
for a in A:
    for b in B:
        A_plus_B[a+b]+=1
count=0
for c in C:
    for d in D:
        if -(c+d) in A_plus_B:
            count+=A_plus_B[-(c+d)]
print(count)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



04089: 电话号码

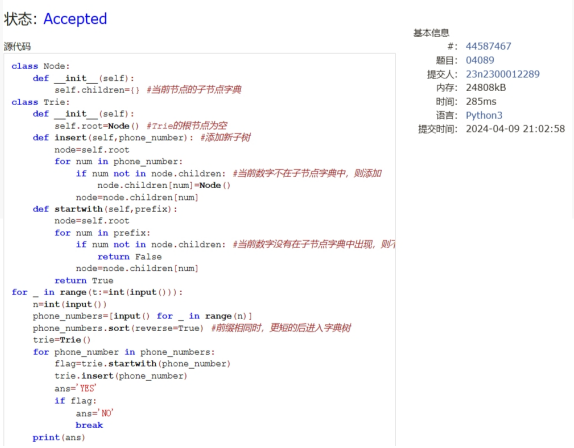
trie, http://cs101.openjudge.cn/practice/04089/ Trie 数据结构可能需要自学下。

思路：见注释，排序是保证输出正确的关键一步

代码

```
class Node:
    def __init__(self):
        self.children={} #当前节点的子节点字典
class Trie:
    def __init__(self):
        self.root=Node() #Trie的根节点为空
    def insert(self,phone_number): #添加新子树
        node=self.root
        for num in phone_number:
            if num not in node.children: #当前数字不在子节点字典中，则添加
                node.children[num]=Node()
            node=node.children[num]
    def startwith(self,prefix):
        node=self.root
        for num in prefix:
            if num not in node.children: #当前数字没有在子节点字典中出现，则不是前缀
                return False
            node=node.children[num]
        return True
for _ in range(t:=int(input())):
    n=int(input())
    phone_numbers=[input() for _ in range(n)]
    phone_numbers.sort(reverse=True) #前缀相同时，更短的后进入字典树
    trie=Trie()
    for phone_number in phone_numbers:
        flag=trie.startwith(phone_number)
        trie.insert(phone_number)
        ans='YES'
        if flag:
            ans='NO'
            break
    print(ans)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



04082: 树的镜面映射

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

思路：根据先序遍历生成伪满二叉树，然后遍历（注意遍历顺序）

代码

```
from collections import deque
class Treenode:
    def __init__(self,value,type_):
        self.value=value
        self.left=None
        self.right=None
        self.type=type_
def build_tree(n,prefix):
    stack=[] #存储还没有孩子子节点的双亲节点
    root=Treenode(prefix[0][0],prefix[0][1])
    stack.append(root)
    for i in range(1,n):
        node=Treenode(prefix[i][0],prefix[i][1])
        if stack[-1].left is None: #栈顶双亲节点没有左子节点
            stack[-1].left=node
        else:
            while stack[-1].right is not None: #若栈顶双亲节点已有左右两个子节点，则出栈
                stack.pop()
            stack[-1].right=node
        if node.type=='0': #内部节点有子节点，需入栈
            stack.append(node)
    return root
def traversal(root):
    queue=deque()
    stack=deque()
    while root and root.value!='$':
        stack.append(root)
        root=root.right
    while stack:
        queue.append(stack.pop())
    while queue:
        root=queue.popleft()
        print(root.value,end=' ')
        if root.left and root.left.value!='$':
            root=root.left
            while root and root.value!='$':
                stack.append(root)
                root=root.right
            while stack:
                queue.append(stack.pop())
n=int(input())
prefix=input().split()
root=build_tree(n,prefix)
traversal(root)
```

状态: Accepted

源代码

```
from collections import deque
class Treenode:
    def __init__(self,value,type_):
        self.value=value
        self.left=None
        self.right=None
        self.type=type_
def build_tree(n,prefix):
    stack=[] #存储还没有孩子子节点的双亲节点
    root=Treenode(prefix[0][0],prefix[0][1])
    stack.append(root)
    for i in range(1,n):
        node=Treenode(prefix[i][0],prefix[i][1])
        if stack[-1].left is None: #栈顶双亲节点没有左子节点
            stack[-1].left=node
        else:
            while stack[-1].right is not None: #若栈顶双亲节点已有左右两个子
                stack.pop()
            stack[-1].right=node
        if node.type=='0': #内部节点有子节点，需入栈
            stack.append(node)
    return root
def traversal(root):
    queue=deque()
    stack=deque()
    while root and root.value!='$':
        stack.append(root)
        root=root.right
    while stack:
        queue.append(stack.pop())
    while queue:
        root=queue.popleft()
        print(root.value,end=' ')
        if root.left and root.left.value!='$':
            root=root.left
            while root and root.value!='$':
                stack.append(root)
                root=root.right
            while stack:
                queue.append(stack.pop())
```

基本信息

#:	44592907
题目:	04082
提交人:	23n2300012289
内存:	3728kB
时间:	28ms
语言:	Python3
提交时间:	2024-04-10 14:21:09

代码运行截图 （AC代码截图，至少包含有"Accepted"）

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。

与树的镜面映射“大战了三百回合”，现在精神状态很美好(

一开始按部就班地根据前序表达式转化为伪满二叉树，并按照根→右子树（需要逆序输出）→左子树的步骤写，结果不知道为什么返回的列表是空的，debug了一会儿发现根返回错了，修改之后样例数据和自己编的一些较简单的数据得到的输出是正确的，但交上去之后还是WA，百思不得其解，遂向群中大佬求助，被建议构造更复杂的数据，尝试后发现在某些情况下自己的代码会导致节点入栈/队列的顺序出错，然而不知道该怎么改，最终放弃挣扎看了题解，发现自己平常的习惯是将答案一次性输出，从而导致本题出现入队/栈顺序错误的情况，题解更类似于先把该层的节点全部输出后，再进行入队/栈的操作，代码可读性更强，也不容易出错。

此外，越来越发觉细节的重要性，如最大权值连通块一个没注意就把变量名起的和函数名一样，电话号码没加sort（）导致自己是自己的前缀从而输出是清一色的‘NO’，也希望自己能从中吸取教训，在其他课的期中考试时也能细心一点

一刻也没有为写不出来树的镜面映射而感到悲伤，立刻到达战场上的是期中考试！