湖南大學

数据结构

课程实验报告

题　　目：图 的 应 用

学生姓名　李晓彤

学生学号　201726010128

专业班级　软件 1701 班

完成日 期　2018.12.20

一、需求分析

0 问题描述

某国的军队由N个部门组成，为了提高安全性，部门之间建立了M条通路，每条通路只能单向传递信息，即一条从部门a到部门b的通路只能由a向b传递信息。信息可以通过中转的方式进传递，即如果a能将信息传递到b，b又能将信息传递到c，则a能将信息传递到c。一条信息可能通过多次中转最终到达目的地。

由于保密工作做得很好，并不是所有部门之间都互相知道彼此的存在。只有当两个部门之间可以直接或间接传递信息时，他们才彼此知道对方的存在。部门之间不会把自己知道哪些部门告诉其他部门。

现在请问，有多少个部门知道所有N个部门的存在。或者说，有多少个部门所知道的部门数量（包括自己）正好是N。

基本要求

1）图的ADT要用实验5实现的代码。

2）要基于图ADT来实现求解问题的代码

3）为降低难度，输入数据（测试样例）的规模选择原题中最小的一组范围。

1.问题分析

问题中的需求：

1. 使用实验5的ADT实现。
2. 每一条是单向通路，只有当两个部门之间可以直接或间接传递信息时，他们才彼此知道对方的存在。
3. 输出有多少个部门可以知道所有部门的存在。

功能：

1. 构造图
2. 输出邻接矩阵
3. 输出每个部门可以连通的部门。1表示连通，0表示不连通。
4. 判断有多少个部门知道所有的部门数量

2.输入数据

为降低难度，输入数据（测试样例）的规模选择原题中最小的一组范围。即：1 ≤ N ≤ 10，1 ≤ M ≤ 20

那么可知问题需要处理图的遍历。

数据输入格式为：输入的第一个数字为部门数量N，第二个数字为单向通路个数M，然后输入M个单向通路，两个部门中间用空格隔开。

例如输入：

```
4
4
1 2
1 3
2 4
3 4
```

则表示你输入的部门数量为4，单向通路个数为4，这些单向通路为：1到2、1到3、2到4、3到4。

3.输出数据

1. 先输出邻接矩阵。
2. 接下来输出每个点能知道的部门，为1是知道，0是不知道。
3. 接下来输出可以知道所有部门N的部门个数

例如输出：

接下来输出邻接矩阵：

0 1 1 0

0 0 0 1

0 0 0 1

0 0 0 0

接下来判断有多少个部门知道所有N个部门的存在：

下图是每个点能知道的部门，为1是知道，0是不知道：

1 1 1 1

1 1 0 1

1 0 1 1

1 1 1 1

那么可以会有2个部门知道所有N个部门的存在！

4.测试样例设计

测试样例一：（普通）

输入：

4

4

1 2

1 3

2 4

3 4

输出：

------欢迎来到由邻接矩阵实现的通信网络------

请输入部门数量N:

4

请输入单向通路的数量M:

4

接下来请输入4条单向通路：譬如说1 2有一条，则输入 1 2 ，顶点与顶点中间用空格隔开

1 2

1 3

2 4

3 4

接下来输出邻接矩阵：

0 1 1 0

0 0 0 1

0 0 0 1

0 0 0 0

接下来判断有多少个部门知道所有N个部门的存在：

下图是每个点能知道的部门，为1是知道，0是不知道：

1 1 1 1

1 1 0 1

1 0 1 1

1 1 1 1
那么可以会有2个部门知道所有N个部门的存在！
样例说明：该样例是用了ccf的示范样例，是部门1 2、1 3、2 4、3 4之间有单向通路。这里采用对每一个点DFS，然后判断遍历后能知道的部门矩阵来判断有多少个部门可以知道所有部门的存在。

测试样例二：（特殊样例，所有部门都相互知道）
输入：
4
5
1 2
1 3
2 4
3 4
2 3

错误输出：
------欢迎来到由邻接矩阵实现的通信网络------
请输入部门数量N:
4
请输入单向通路的数量M：
5
接下来请输入5条单向通路：譬如说1 2有一条，则输入 1 2 ，顶点与顶点中间用空格隔开
1 2
1 3
2 4
3 4
2 3
接下来输出邻接矩阵：
0 1 1 0
0 0 1 1
0 0 0 1
0 0 0 0
接下来判断有多少个部门知道所有N个部门的存在：
下图是每个点能知道的部门，为1是知道，0是不知道：
1 1 1 0
1 1 1 1
1 1 1 1
0 1 1 1
那么可以会有2个部门知道所有N个部门的存在！

正确输出：
------欢迎来到由邻接矩阵实现的通信网络------
请输入部门数量N:

4
请输入单向通路的数量M:
5
接下来请输入5条单向通路：譬如说1 2有一条，则输入 1 2 ，顶点与顶点中间用空格隔开
1 2
1 3
2 4
3 4
2 3
接下来输出邻接矩阵：
0 1 1 0
0 0 1 1
0 0 0 1
0 0 0 0
接下来判断有多少个部门知道所有N个部门的存在：
下图是每个点能知道的部门，为1是知道，0是不知道：
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
那么可以会有4个部门知道所有N个部门的存在!
样例说明：该样例是所有部门均可知道全部部门的存在。与样例一不同的是：增加了2到3的单向通路，这样2和3也相互知道。这里错误输出是因为DFS没有做出相应改变，导致输出错误。

测试样例三：（特殊，图不连通）
输入：
5
4
1 2
1 3
2 4
3 4

输出：
------欢迎来到由邻接矩阵实现的通信网络------
请输入部门数量N:
5
请输入单向通路的数量M:
4
接下来请输入4条单向通路：譬如说1 2有一条，则输入 1 2 ，顶点与顶点中间用空格隔开
1 2
1 3

2 4
3 4
接下来输出邻接矩阵：
0 1 1 0 0
0 0 0 1 0
0 0 0 1 0
0 0 0 0 0
0 0 0 0 0
接下来判断有多少个部门知道所有N个部门的存在：
下图是每个点能知道的部门，为1是知道，0是不知道：
1 1 1 1 0
1 1 0 1 0
1 0 1 1 0
1 1 1 1 0
0 0 0 0 1
那么可以会有0个部门知道所有N个部门的存在！
样例说明：该样例是样例一的改进版，这里是增加了一个顶点5，因为这里的图不连通，所以并没有部门会知道所有N个部门的存在。

测试样例四：（特殊，有双向边）
输入：
4
4
1 2
1 3
2 4
4 2

输出：
------欢迎来到由邻接矩阵实现的通信网络------
请输入部门数量N:
4
请输入单向通路的数量M：
4
接下来请输入4条单向通路：譬如说1 2有一条，则输入 1 2 ，顶点与顶点中间用空格隔开
1 2
1 3
2 4
4 2
接下来输出邻接矩阵：
0 1 1 0
0 0 0 1
0 0 0 0
0 1 0 0

接下来判断有多少个部门知道所有N个部门的存在：
下图是每个点能知道的部门，为1是知道，0是不知道：
1 1 1 1
1 1 0 1
1 0 1 0
1 1 0 1
那么可以会有1个部门知道所有N个部门的存在！
样例说明：这是一个特殊样例。该样例是在样例一的基础上做了改进，将原来的3和4之间单向通路去除，变成了2和4的双向通路。这样的话由于只有1可以到达所有的顶点，所有只有1可以知道所有部门的存在。相比样例一，这里因为4不可以接收3的信息，所以4不可以知道所有N个部门的存在，而样例一是4可以接收3的信息，所以4可以知道所有N个部门的存在。

测试样例五（普通）
输入：
5
6
1 3
1 4
4 2
2 3
3 5
4 5

输出：
------欢迎来到由邻接矩阵实现的通信网络------
请输入部门数量N:
5
请输入单向通路的数量M：
6
接下来请输入6条单向通路：譬如说1 2有一条，则输入 1 2 ，顶点与顶点中间用空格隔开
1 3
1 4
4 2
2 3
3 5
4 5
接下来输出邻接矩阵：
0 0 1 1 0
0 0 1 0 0
0 0 0 0 1
0 1 0 0 1
0 0 0 0 0
接下来判断有多少个部门知道所有N个部门的存在：

下图是每个点能知道的部门，为1是知道，0是不知道：

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

那么可以会有5个部门知道所有N个部门的存在！

样例说明：这是一个简单普通样例。相比样例二只是增加了部门数量和单向通路的数量，并没有什么本质区别，仅仅是一个每一个点都可以遍历所有点的图。

二、概要设计

1.抽象数据类型

因为题目要求用实验5的ADT实现，并且存储的数据满足：通过 for 循环依次存入顶点，按照存储的先后次序，满足线性特征，即<ai,ai+1>(0≤a＜n)。

又因为要用图的深度遍历功能，所以这里采用了邻接矩阵实现。

2.算法的基本思想

本题先输入部门个数N和部门之间的单向通路个数M，接下来输入这M个单向通路。这里利用实验5写好的init函数来构造一个有向图，同时构造好邻接矩阵。

接下来就是遍历：因为题目要求输出可以知道所有N个部门的部门个数，所以这里采用深度遍历改图。但是与深度遍历不同的是，我将深度遍历做了如下改进：

1.构造了一个二维数组来存储遍历结果。即如果1可以到2则将该数组中的第一行第二列置1，第二行第一列置1。

2.对每一个顶点深度遍历。

3.深度遍历中记录起始位置和当前位置，目的是方便之后递归遍历并且在二维数组中存储好遍历结果。

4.遍历完一个顶点后要对mark数组置0，进行下一个顶点的遍历。

3.程序的流程

程序由两个模块组成：

（1） 构造图模块：

这一模块的功能是：将输入进来的部门和单向通路构造一个有向图，这一步完成了构造图的功能。

（2） 遍历图模块：

这一模块的功能是：对已经构造好的有向图进行DFS。这里对DFS做了相应改进，原来只是改变mark数组的值来判断该点是否被遍历过，现在增加了一个二维数组来存储遍历结果。每一行对应一个顶点的遍历结果，1为可以遍历到，0为遍历不到。最后只需要对该二维数组遍历，如果这一行都是1则表明该部门可以知道所有N个部门，如果有0的出现，则说明该部门不可以知道所有N个部门，然后统计一行全是1的部门个数即可知道有多少个部门可以知道所有部门的存在。

三、详细设计

1.物理数据类型

数据是一串数字，根据这串数字来构建图，因为输入的单向通路使得部门与部门结构特性满足线性特性，并且之后对图进行遍历，所以选择邻接矩阵存储数据，这里的图继承了基本图类。

抽象数据类型的具体设计：

```
ADT   IntegerSet {
      数据对象：D = { ai | ai Î 整数，i = 1,2，…, n, n>=0}
```

数据关系：R = {< ai-1，ai> | ai-1，ai ÎD}

成员：

```
int numVertex,numEdge; //顶点数和边数

int vexs[MAX_VERTEX_NUM];//存储顶点信息

int** matrix;           //Pointer to adjacency matrix 指向邻接矩阵matrix

int* mark;              // Pointer to mark array 指向mark数组

int **connect;         //每个点和其他点的联系关系数组基本操作：
```

成员函数：

```
 // Initialize the graph初始化图
 void Init(int n);
 int n();
 int e();

 /**给某个顶点赋值**/
 void putVex(int v,int value);

 // Set edge (v1, v2) to"wt" 设置边(v1,v2)的权值为wt
 void setEdge(int v1, int v2, int wt);

 int weight(int v1, int v2);//设置边
 int getMark(int v);//获得顶点的遍历情况
 void setMark(int v, int val);//设置顶点的遍历情况
 void clearMark();//清空标志数组

 void printvexs(int n);//输出邻接矩阵

 void DFS(int start,int curr);//深度优先遍历
 void printcon(int n);//输出每个点的联系矩阵
 bool searchcon(int start);//检查这个点和其他顶点的联系情况
```

2.输入和输出的格式

输入流程及格式：输入的第一个数字为部门数量N，第二个数字为单向通路个数M，然后输入M个单向通路，两个部门中间用空格隔开。

输出流程：

先输出系统名称：------欢迎来到由邻接矩阵实现的通信网络------

然后输出提示信息和输入例子：

请输入部门数量N:

当用户输入部门数量N后，输出提示信息和输入例子：
请输入单向通路的数量M：
当用户输入完单向通路的数量M后，输出提示信息和输入例子：
接下来请输入4条单向通路：譬如说1 2有一条，则输入 1 2 ，顶点与顶点中间用空格隔开
当用户输入完单向通路后，输出构造好的邻接矩阵，以此来证明是否成功构建图。然后输出每个顶点进行深度遍历的结果矩阵。
最后输出可以知道所有N个部门的部门个数，然后结束程序。

2.算法的具体步骤

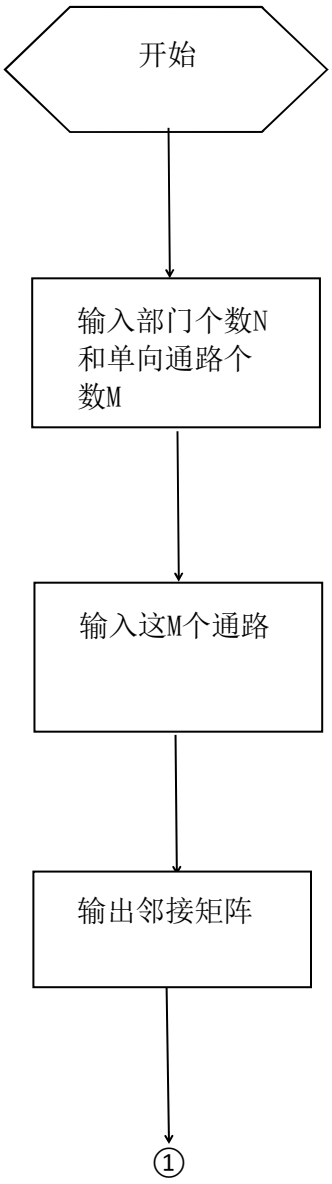
1.先输入结点个数来构建二叉检索树。
2.输入对应的结点值，并且利用自身定义的成员函数来完成二叉检索树的构造。
3.进行中序遍历，来检查是否构建正确的二叉检索树。
4.进行查找操作，利用 while 循环来让用户输出查找值，如果没找到则输出查找失败信息，如果找到了则输出查找成功的信息。这里是利用二叉检索树本身的成员函数来完成 查找操作。
5.输出查找需要的比较次数。
伪代码：

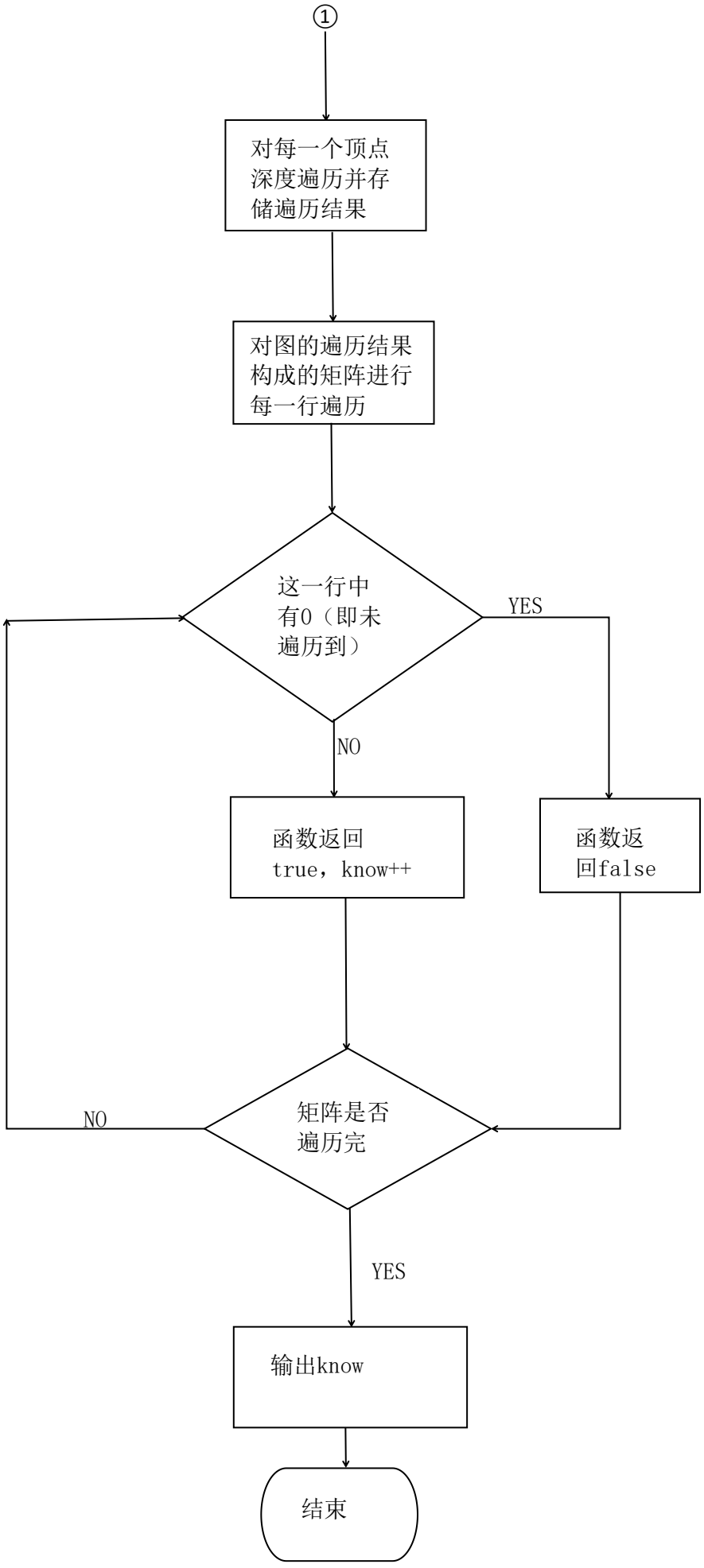
```
int main(int argc, char** argv) {
    cout<<"------欢迎来到由邻接矩阵实现的通信网络------"<<endl;
     int n=0;
     cout<<"请输入部门数量N: "<<endl;
     cin>>n;
     cout<<"请输入单向通路的数量M："<<endl;
     int num=0;
     cin>>num;
     Graphm G;
     G.Init(n);
     for(int i=0;i<n;i++)
     {
            G.putVex(i,i+1);
     }
     cout<<"接下来请输入"<<num<<"条单向通路：譬如说1 2有一条，则输入 1 2 ，
顶点与顶点中间用空格隔开"<<endl;
     int a,b;
     int weight=1;
     while(num!=0)
     {
            cin>>a>>b;
            G.setEdge(a,b,weight);//构造图
            num=num-1;
     }
     cout<<"接下来输出邻接矩阵： "<<endl;
     G.printvexs(n);
     cout<<"接下来判断有多少个部门知道所有N个部门的存在： "<<endl;

     //接下来是遍历模块
     int know=0;
     for(int i=0;i<n;i++)
```

```
        {
            G.DFS(i,i);//对每个点DFS
            G.clearMark(); //然后对遍历结果清空，进行下一次遍历
        }
        cout<<"下图是每个点能知道的部门，为1是知道，0是不知道："<<endl;
        G.printcon(n);//输出每一个点的遍历结果所构成的矩阵
        for(int i=0;i<n;i++)
        {
            if(G.searchcon(i)==true)//如果所有部门都可以遍历到
            know++;
        }
        cout<<"那么可以会有"<<know<<"个部门知道所有N个部门的存在！"<<endl;
        return 0;
}
```

流程图：




①
对每一个顶点深度遍历并存储遍历结果
对图的遍历结果构成的矩阵进行每一行遍历
这一行中有0（即未遍历到）
YES
NO
函数返回true，know++
函数返回false
矩阵是否遍历完
NO
YES
输出know
结束

4.算法的时空分析

（1）构造图模块：

这一模块主要是用while循环来构造图的邻接矩阵，while循环次数取决于单向通路M的个数，所以时间复杂度为θ（n）。由于是用邻接矩阵实现的图，所以邻接矩阵的空间代价为θ（|V|^2）。

（2）遍历图模块：

这一模块主要是利用邻接矩阵来对每一个顶点进行DFS操作，由于是邻接矩阵实现的图，并且是有向图，DFS对每一条边处理一次，故在DFS部分时间代价为θ（|V|）因为又是利用for循环对每一个顶点进行遍历，所以总时间代价为θ（n*|V|），若假设|V|=n，那么总时间代价为θ（n^3）。这里的空间代价主要取决于存储遍历结果的矩阵，因为存储结果的矩阵和邻接矩阵相似，也是一个n*n的大小矩阵，故该模块的空间代价为θ（n^2）。

四、调试分析

1.调试方案设计

调试目的：为了发现代码是否有语法错误、连接错误、逻辑错误和运算错误，有不严谨之处。

样例：

4

5

1 2

1 3

2 4

3 4

2 3

设置断点处：对每个顶点进行DFS操作的for循环处。

调试计划：留心观察是否可以成功存储每个顶点的遍历结果，仔细留心DFS中的当前位置curr和起始位置start以及DFS中的i的值。

2. 调试过程和结果，及分析

调试过程：现将遍历结果矩阵对应位置置1，即：

connect[curr][start]=connect[start][curr]=1。这个的意思是将connect矩阵的当前行起始列和起始行当前列置1，表示当前位置可以到起始位置，起始位置也可以到当前位置。然后将当前位置的点A标记为visited，即已经遍历过。然后对当前位置进行接下来的遍历，如果接下来的点B没有被访问，并且当前位置可以到该点，则递归调用DFS函数，而此时传的参数起始位置还是一开始传入的位置，但是当前位置变为B点。

递归结束的条件是整个图遍历结束，所有顶点都被访问。

问题：发现输出的遍历结果矩阵中顶点1到顶点4是0，即顶点1不可以到顶点4，这与事实相悖。

解决方法：改写DFS中的if判断条件，将原来的

```
if(mark[i] == 0 && matrix[start][i] !=0)
```

改写为：

```
if(mark[i] == 0 && matrix[curr][i] !=0)
```

原因：因为在调试的时候发现，如果顶点1和顶点4之间没有边就直接不会进入DFS了，尽管顶点2可以到顶点4，顶点1可以到顶点2，从而顶点1可以到顶点4。但是由于不满足matrix[start][i] !=0这个条件，所以不能进入DFS，就不能对顶点2遍历，所以相应的遍历结果矩阵就输出错误。

五、测试结果

测试样例一：（普通）

错误输出：

```
4
4
1 2
1 3
2 4
3 4
下图是每个点能知道的部门，为1是知道，0是不知道：
1 1 1 0
1 1 0 1
1 0 1 1
0 1 1 1
0
```

正确输出：

```
------欢迎来到由邻接矩阵实现的通信网络------
请输入部门数量N:
4
请输入单向通路的数量M:
4
接下来请输入4条单向通路：譬如说1 2有一条，则输入 1 2 ，顶点与顶点中间用空格隔开
1 2
1 3
2 4
3 4
接下来输出邻接矩阵：
0 1 1 0
0 0 0 1
0 0 0 1
0 0 0 0
接下来判断有多少个部门知道所有N个部门的存在：
下图是每个点能知道的部门，为1是知道，0是不知道：
1 1 1 1
1 1 0 1
1 0 1 1
1 1 1 1
那么可以会有2个部门知道所有N个部门的存在！
请按任意键继续. . .
```

分析结论：因为这里遍历结果矩阵中第一行和第四行是全为1的，那么以为着只有顶点1和顶点4可以知道所有N个部门的存在，故输出2。

测试样例二：（特殊样例，所有部门都相互知道）

错误输出：

```
4
5
1 2
1 3
2 4
3 4
2 3
下图是每个点能知道的部门，为1是知道，0是不知道：
1 1 1 0
1 1 1 1
1 1 1 1
0 1 1 1
2
```

正确输出：

```
------欢迎来到由邻接矩阵实现的通信网络------
请输入部门数量N:
4
请输入单向通路的数量M:
5
接下来请输入5条单向通路：譬如说1 2有一条，则输入 1 2 ，顶点与顶点中间用空格隔开
1 2
1 3
2 4
3 4
2 3
接下来输出邻接矩阵：
0 1 1 0
0 0 1 1
0 0 0 1
0 0 0 0
接下来判断有多少个部门知道所有N个部门的存在：
下图是每个点能知道的部门，为1是知道，0是不知道：
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
那么可以会有4个部门知道所有N个部门的存在！
请按任意键继续. . .
```

分析结论：相比测试样例一，这里增加了顶点2和顶点3之间的通路，这样顶点2和顶点3也可以知道所有N个部门的存在，故这里输出4.

测试样例三：（特殊，图不连通）

输出：

```
------欢迎来到由邻接矩阵实现的通信网络------
请输入部门数量N:
5
请输入单向通路的数量M:
4
接下来请输入4条单向通路：譬如说1 2有一条，则输入 1 2 ，顶点与顶点中间用空格隔开
1 2
1 3
2 4
3 4
接下来输出邻接矩阵：
0 1 1 0 0
0 0 0 1 0
0 0 0 1 0
0 0 0 0 0
0 0 0 0 0
接下来判断有多少个部门知道所有N个部门的存在：
下图是每个点能知道的部门，为1是知道，0是不知道：
1 1 1 1 0
1 1 0 1 0
1 0 1 1 0
1 1 1 1 0
0 0 0 0 1
那么可以会有0个部门知道所有N个部门的存在！
请按任意键继续. . .
```

分析结论：测试样例三相比测试样例一多增加了一个顶点5，因为顶点5是一个孤立的点，所以这个图并不连通。所以没有部门会知道所以部门的存在。

测试样例四：（特殊，有双向边）

输出：

```
------欢迎来到由邻接矩阵实现的通信网络------
请输入部门数量N:
4
请输入单向通路的数量M:
4
接下来请输入4条单向通路：譬如说1 2有一条，则输入 1 2 ，顶点与顶点中间用空格隔开
1 2
1 3
2 4
4 2
接下来输出邻接矩阵：
0 1 1 0
0 0 0 1
0 0 0 0
0 1 0 0
接下来判断有多少个部门知道所有N个部门的存在：
下图是每个点能知道的部门，为1是知道，0是不知道：
1 1 1 1
1 1 0 1
1 0 1 0
1 1 0 1
那么可以会有1个部门知道所有N个部门的存在！
请按任意键继续. . .
```

分析结论：这是一个特殊样例。该样例是在样例一的基础上做了改进，将原来的3和4之间单向通路去除，变成了2和4的双向通路。这样的话由于只有1可以到达所有的顶点，所有只有1可以知道所有部门的存在。相比样例一，这里因为4不可以接收3的信息，所以4不可以知道所有N个部门的存在。这里从遍历矩阵也可以看出结果，只有第一行是全为1的，所以只有顶点1知道所有部门的存在，故输出1.

测试样例五（普通）

输出：

```
------欢迎来到由邻接矩阵实现的通信网络------
请输入部门数量N:
5
请输入单向通路的数量M:
6
接下来请输入6条单向通路：譬如说1 2有一条，则输入 1 2 ，顶点与顶点中间用空格隔开
1 3
1 4
4 2
2 3
3 5
4 5
接下来输出邻接矩阵：
0 0 1 1 0
0 0 1 0 0
0 0 0 0 1
0 1 0 0 1
0 0 0 0 0
接下来判断有多少个部门知道所有N个部门的存在：
下图是每个点能知道的部门，为1是知道，0是不知道：
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
那么可以会有5个部门知道所有N个部门的存在！
请按任意键继续. . .
```

分析结论：这是一个简单普通样例。相比样例二只是增加了部门数量和单向通路的数量，并没有什么本质区别，仅仅是一个每一个点都可以遍历所有点的图。从遍历结果矩阵可以看出，所有都为1，说明所有顶点都可以知道所有N个部门的存在。故输出5。

六、实验日志

2018/12/10

内容：实验六上机

在三道ccf题中抉择，确定解决的问题是通信网络这道题。

2018/12/15

内容：确定如何遍历所有顶点

1. DFS 函数：

初想法：刚开始的想法是利用DFS遍历有向图，然后每次遍历后对每个顶点的遍历情况清空，再进行下一次遍历。如果这个mark数组全为1则该顶点可以访问所有顶点，则将know++。

遇到的问题：对于第二个测试样例就无法通过。因为第二个测试样例只是增加了顶点2和顶点3之间的通路，如果按照样例一的说明，则顶点2和顶点3也可以彼此知道对方的存在，这样所有的点都可以知道全部部门的存在。但是利用这种方法输出结果仍为2。经过调试后发现，因为这个是有向图，所以有向图遍历的时候顶点2就不能到顶点1，自然顶点1就不能为visited。但按照题意来讲，因为顶点1可以传输信息至顶点2，所以顶点1与顶点2彼此知道对方的存在。所以这个方法有缺陷。

修改后的想法：不采用DFS，采用自己写的find函数。这个find函数是直接判断顶点1和顶点4之间可不可以联系，这里的可以通过别的部门联系也可以直接联系，如果可以则返回true，不可以则返回false。

2018/12/16

内容：确定find函数

最开始确定的find函数如下图所示：

```
bool Graphm::find(int start,int end)
{
    if(matrix[start][end]==1)//有直达的
    return true;
    else{//没有直达的
        for(int i=0;i<numVertex;i++)//开始找间接的
        {
            if(matrix[start][i]==1)
            {
                if(find(i,end))
                break;
                else continue;
            }
        }
        return false;//没找到
    }

}
```

这里的start是开始的顶点，end是要判断要联系到的顶点，i是当前顶点位置。这个思路就是：如果邻接矩阵中，要判断的开始顶点到结束顶点中有直达的通路，则返回true，如果没有则找间接的通路，即通过其他部门可以联系上的。如果没找到则返回false，找到了返回true。

2018/12/17
内容：修改find函数
遇到的问题：
这里利用的是样例一进行调试：
利用find函数进行递归判断的时候，当判断1可以到4之后并不能停止递归操作，而是返回上一级再进行接下来的操作，所以就算是1可以通过2和3到达4，但是之后也不会停止递归操作，而是继续对2进行递归操作，最后的结果也是返回false。原因未知，最后调试把自己调试晕了。
解决方法：
舍弃这种利用find函数来找的思路。上网搜大牛的代码，发现他们的思路主要是存储了一个遍历结果的矩阵，最后只需遍历这个矩阵，看全为1的一行有多少个则有多少个部门可以知道所有部门N。然后就将自己的DFS函数做了改进，加了一个二维数组存储遍历结果，然后对每次遍历都记录结果，1是可以相互知道彼此，0是不知道。最后利用for循环来对这个二维数组结果遍历，一行全为1的则是可以知道所有部门，最后输出个数即可。

经验和体会：这次实验六的代码上交为30分，也就是说满足了实验六的需求，但是还是不太明白思路哪里有缺陷，也不明白为什么不是100分。在这次写实验六的代码同样也遇到了指针越界问题，还有类不能实例化的问题。但这次的主要问题还是不会解决递归，这让我认识到了自己的代码水平还有待提高，但自己对于图的遍历问题也有了更深入的了解，自己平时也要多练题啊。