# 实验 6 图的应用(通信网络)

## 计科 1903 201914020128 陈旭

## 一. 问题分析

### 1). 问题与功能描述

1. 需要处理的数据是一串成对的整型(int 型)数字，他们共同组成了图的元素(结点和边)。
2. 实现的功能有:
   - 读入两个整数，前者作为部门(结点个数)，后者作为结点之间的边数；
   - 循环。先循环将每个结点编号，然后再循环将结点之间的边的情况输入并插入图中，并由此得到邻接矩阵和初步的通信网络矩阵(一个和邻接矩阵类似的二维数组)
   - 将通信网络矩阵求解问题抽象为 warshall 算法求传递闭包的问题
   - 处理该树
   - 输出全都知道的部门个数
3. 使用标准输出即可。

### 2). 样例分析

1. **求解方法:**因为因信息传递而知道部门的存在是相互的，并且自己必然知道自己的存在，所以，在得到通信网络矩阵(就是将邻接矩阵进行拷贝，得到和其相同的矩阵)以后，需要先进行一个操作，将自身到自身的值赋值为 1。然后根据 warshall 算法[1]，构建起该题的通信网络矩阵。然后判断矩阵中每一行是否均为 1，若不是则直接

跳出本次循环，若全都是则中间变量 temp+1，最终返回中间变量，表示全都知道的部门个数，然后输出

**备注:** warshall 算法思想: 如果 a, b 两个点之间有边(即互相知道)， b, c 两点之间有边(即互相知道)，那么 a 和 c 互相知道存在，该算法可以用三重循环来构建。

**2. 样例求解:**

**【样例输入】** 4 4

1 2

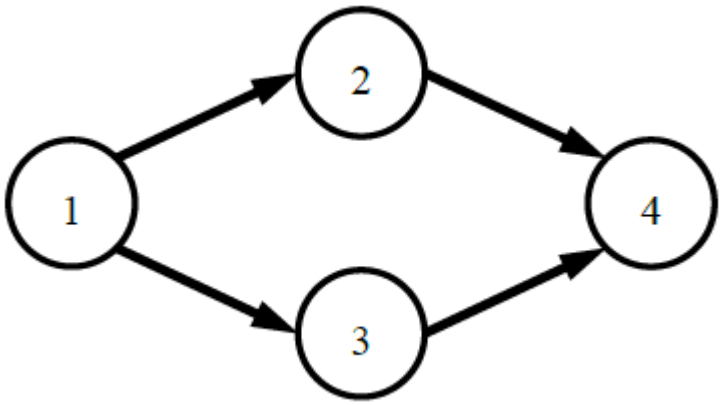
1 3

2 4

3 4

**【样例输出】** 2

**【求解过程】** 求解过程如下




1. 4 个结点, 4 条边，循环读入后插入图
2. 插入图后得到邻接矩阵和通信网络矩阵, 均为

   0 1 1 0

0 0 0 1

0 0 0 1

0 0 0 0

3. 将自身到自身的边均置为 1，得邻接矩阵和通信网络矩阵均为

   1 1 1 0

   0 1 0 1

   0 0 1 1

   0 0 0 1

4. 根据邻接矩阵和 warshall 算法，构建通信网络矩阵为

   1 1 1 1

   0 1 0 1

   0 0 1 1

   0 0 0 1

5. 根据知道存在得相互性，将上述矩阵中的 1 对称赋值给相应元素，得最终的通信网络矩阵

   1 1 1 1

   1 1 0 1

   1 0 1 1

   1 1 1 1

6. 统计每一行均为 1 的行数，为 2，则输出为 2

## 数据结构分析

**1）数据对象** 本题处理的数据为整型(int)数组。

**数据关系** 本题处理的数据为整型有限数据，题目每组数据之间均可能有关系。因此使用图作为基本数据结构进行操作。

**2）基本操作**

- **【功能描述】**设置图的类型(有向或者无向)

  **【名字】** setType

  **【输入】** 整数 n,1 代表有向，0 代表无向

  **【输出】** 无

- **【功能描述】**设置结点

  **【名字】** setVex

  **【输入】** 两个整数，分别代表结点编号和结点值

  **【输出】** 无

- **【功能描述】**设置图的边

  **【名字】** setEdge

  **【输入】** 三个整数，分别代表出度，入度，权值(为处理方便本题直接统一赋值为 0)

  **【输出】** 无

- **【功能描述】**设置通信网络矩阵

  **【名字】** setComnetwork

  **【输入】** 无

  **【输出】** 无

- **【功能描述】**求解通信网络矩阵

  **【名字】** delComnetwork()

  **【输入】** 无

  **【输出】** 无

- **【功能描述】**返回都知道的个数

  **【名字】** allKnow

  **【输入】** 无

  **【输出】** 一个整数，表示知道所有部门存在情况的个数

**3) 物理实现**

```
void setType(int flag){
    if (flag==1){
        isDirected=true;
    }
    else{
        isDirected=false;
    }
} //给结点赋值

void setVex(int v, E value){
    assert(v<numVertex);
    vertex[v]=value;
}//给结点赋值


void setEdge(int v1, int v2, int wght){
    assert(v1<numVertex&&v2<numVertex);
    if (v1==v2){
        Comnetwork[v1][v2]=1;
        matrix[v1][v2]=1;
    }
    if (matrix[v1][v2]==0){
        numEdge++;
        matrix[v1][v2]=1;
```

```
        Comnetwork[v1][v2]=1;
    }
    weight[v1][v2]=wght;
    if(!isDirected){
        matrix[v2][v1]=1;
        weight[v2][v1]=wght;
        Comnetwork[v2][v1]=1;
    }
}//建立一个边，也就是把权值和坐标赋值
void setComnetwork(){
    for (int i=0; i<numVertex; i++){
        for (int j=0; j<numVertex; j++){
            if(matrix[j][i]){
             for (int k=0; k<numVertex; k++){
                    if(matrix[j][k]==0){
                       if(matrix[i][k]){
                          Comnetwork[j][k]=1;
                        }
                    }
                }
            }
        }
    }
} //设置通信网络


void delComnetWork(){
    for (int i=0; i<numVertex; i++){
        for (int j=0; j<numVertex; j++){
            if (Comnetwork[i][j]){
                Comnetwork[j][i]=1;
            }
        }
    }
} //求解通信网络矩阵


int allKnow(){
    int result=0;
    for (int i=0; i<numVertex; i++){
        int ju=0;
        for (int j=0; j<numVertex; j++){
            if (Comnetwork[i][j]!=1){
                ju=1;
```

```
                    break;
                }
            }
            if (ju==0){
                result++;
            }
        }
        return result;
    } //返回有几个都知道的部门
```

## 二. 算法分析

- **算法思想：**因为因信息传递而知道部门的存在是相互的，并且自己必然知道自己的存在，所以，在得到通信网络矩阵(就是将邻接矩阵进行拷贝，得到和其相同的矩阵)以后，需要先进行一个操作，将自身到自身的值赋值为 1。然后根据 warshall 算法[1]，构建起该题的通信网络矩阵。然后判断矩阵中每一行是否均为 1，若不是则直接跳出本次循环，若全都是则中间变量 temp+1，最终返回中间变量，表示全都知道的部门个数，然后输出。

  **备注：**warshall 算法思想：如果 a, b 两个点之间有边(即互相知道)， b, c 两点之间有边(即互相知道)，那么 a 和 c 互相知道存在，该算法可以用三重循环来构建。

- **性能分析**

【空间复杂度】原题代码可以分为三个部分，开辟了两个一维动态数组，三个二维动态数组，除此之外还开辟了一些单独的变量，三者空间复杂度为 O(n), $O(n^2)$,O(1);取最大值得到空间复

杂度为 $O(n^2)$

【时间复杂度】setComnetwork 函数时间复杂度为 $O(n^3)$，建图，allknow, delComnetwork 函数时间复杂度为 $O(n^2)$，其余均为 O(1),取最大值得该程序时间复杂度为 $O(n^3)$