

题目描述：

为了达到新冠疫情精准防控的需要，为了避免全员核酸检测带来的浪费，需要精准圈定可能被感染的人群。

现在根据传染病流调以及大数据分析，得到了每个人之间在时间、空间上是否存在轨迹的交叉。

现在给定一组确诊人员编号 ($X_1, X_2, X_3, \dots, X_n$)，在所有人当中，找出哪些人需要进行核酸检测，输出需要进行核酸检测的人数。（注意：确诊病例自身不需要再做核酸检测）需要进行核酸检测的人，是病毒传播链条上的所有人员，即有可能通过确诊病例所能传播到的所有人。

例如： A 是确诊病例， A 和 B 有接触、 B 和 C 有接触、 C 和 D 有接触、 D 和 E 有接触，那么 $B \setminus C \setminus D \setminus E$ 都是需要进行核酸检测的人。

输入描述：

第一行为总人数 N

第二行为确诊病例人员编号（确诊病例人员数量 $< N$ ），用逗号分割

第三行开始，为一个 $N \times N$ 的矩阵，表示每个人员之间是否有接触， 0 表示没有接触， 1 表示有接触。

输出描述：

整数：需要做核酸检测的人数

补充说明：

人员编号从 0 开始

$0 < N < 100$

示例 1

输入：

```
5
1,2
1,1,0,1,0
1,1,0,0,0
0,0,1,0,1
1,0,0,1,0
0,0,1,0,1
```

输出：

3

说明：

编号为 1、2 号的人员，为确诊病例。

1 号与 0 号有接触，0 号与 3 号有接触。

2 号与 4 号有接触。

所以，需要做核酸检测的人是 0 号、3 号、4 号，总计 3 人需要进行核酸检测。

```
import java.util.*;
```

```
// 注意类名必须为 Main, 不要有任何 package xxx 信息
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        int N = scanner.nextInt();  
        scanner.nextLine();  
        String[] infectedStr = scanner.nextLine().split(",");  
        Set<Integer> infected = new HashSet<Integer>();  
        for(String str : infectedStr){  
            infected.add(Integer.parseInt(str));  
        }  
  
        int[][] contactMatrix = new int[N][N];  
  
        for(int i = 0; i < N; i++){  
            String[] row = scanner.nextLine().split(",");  
            for(int j = 0; j < N; j++){  
  
                contactMatrix[i][j] = Integer.parseInt(row[j]);  
            }  
        }  
  
        int result = getResult(N, infected, contactMatrix);  
        System.out.println(result);  
    }  
}
```

```

private static int getResult(int N ,Set<Integer> infecter, int[][] contactMatrix ) {
    Set<Integer> testing=new HashSet<Integer>(infecter);
    Queue<Integer> queue = new LinkedList<Integer>(infecter);

    while(!queue.isEmpty()){
        int person = queue.poll();
        for(int i = 0;i<N;i++){
            if(contactMatrix[person][i] == 1 && !testing.contains(i)){
                testing.add(i);
                queue.add(i);
            }
        }
    }
    return testing.size() - infecter.size();
}

}

```