机器人活动区域

知识点深搜广搜
时间限制：1s 空间限制：256MB 限定语言：不限

题目描述：

现有一个机器人，可放置于 M × N的网格中任意位置，每个网格包含一个非负整数编号。当相邻网格的数字编号差值的绝对值小于等于 1 时，机器人可在网格间移动

问题：求机器人可活动的最大范围对应的网格点数目。

说明：

1）网格左上角坐标为 (0, 0)，右下角坐标为 (m-1, n-1)

2）机器人只能在相邻网格间上、下、左、右移动

示例1，输入如下网格

1	2	5	2
2	4	4	5
3	5	7	1
4	6	2	4

输出：6

说明：图中绿色区域，相邻网格差值绝对值都小于等于1，且为最大区域，对应网格点数目为6

示例 2，输入如下网格：

输出：1

说明：任意两个相邻网格的差值绝对值都大于1，机器人不能在网格间移动，只能在单个网格内活动，对应网格点数目为 1

输入描述：

第1行输入为M和N，M表示网格的行数，N表示网格的列数
之后M行表示网格数值，每行N个数值（数值大小用k表示），数值间用单个空格分隔，行首行尾无多余空格
M、N、k均为整数，且1<=M,N<=150，0<=k<=50

输出描述：

输出1行，包含1个数字，表示最大活动区域对应的网格点数目
行末无多余空格
补充说明：
无需验证输入格式和输入数据合法性

示例1

输入:

```
4 4
1 2 5 2
2 4 4 5
3 5 7 1
4 6 2 4
```

输出:

```
6
```

说明:

见描述中示例 1，最大区域对应网格点数目为 6

示例2

输入:

```
2 3
1 3 5
4 1 3
```

输出:

```
1
```

说明:

见前面描述中示例 2，最大区域对应网格点数目为 1

解题思路:

使用深度搜索求出所有机器人可活动区域大小，找出其中最大值。

```
public class Main{

    public static int[][] region;
    public static int M;
    public static int N;

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        M = sc.nextInt();
```

```
        N = sc.nextInt();

        region = new int[M][N];
        for(int i=0; i<M; i++){
            for(int j=0; j<N; j++){
                region[i][j] = sc.nextInt();
            }
        }

        int max = 0;        //最大活动区域
        for(int i=0; i<M; i++){
            for(int j=0; j<N; j++){
                if(region[i][j] != -1){
                    max = Math.max( max, move( i, j, region[i][j]));
                }
            }
        }

        System.out.println(max);
    }

    /**
     *
     * @param row      横坐标
     * @param col      纵坐标
     * @param num      上个网格的数字编号
     * @return         活动区域大小
     */
    public static int move(int row, int col, int num){

        if(row < 0 ||
           col < 0 ||
           row >= M ||
           col >= N
          ){
            return 0;     //越界了，返回 0
        }

        int currentNum = region[row][col];
        if(currentNum == -1 ||          //已经统计过的网格
           Math.abs(currentNum - num) > 1){        //不符合绝对差值小于等于 1
            return 0;
        }
```

```
            region[row][col] = -1;          //已经统计过的网格置为-1
            int count = 1;          //符合要求的网格统计 1
            count += move( row - 1, col, currentNum);      //向上
            count += move( row + 1, col, currentNum);      //向下
            count += move( row, col - 1, currentNum);      //向左
            count += move( row, col + 1, currentNum);      //向右

            return count;
        }

}
```