数值同化

题目描述：

存在一个 m*n 的二维数组，其成员取值范围为 0,1,2。其中值为 1 的元素具备同化特性，每经过 1S，将上下左右值为 0 的元素同化为 1。而值为 2 的元素，免疫同化。将数组所有成员随机初始化为 0 或 2，再将矩阵的[0,0]元素修改成 1，在经过足够长的时间后，求矩阵中有多少个元素是 0 或 2（即 0 和 2 数量之和）。

输入描述：

输入的前两个数字是矩阵大小。后面的数字是矩阵内容。

输出描述：

返回矩阵中非 1 的元素个数

补充说明：

m 和 n 不会超过 30(含 30)。

题目描述：

存在一个 m*n 的二维数组，其成员取值范围为 0,1,2。其中值为 1 的元素具备同化特性，每经过 1S，将上下左右值为 0 的元素同化为 1。而值为 2 的元素，免疫同化。将数组所有成员随机初始化为 0 或 2，再将矩阵的[0,0]元素修改成 1，在经过足够长的时间后，求矩阵中有多少个元素是 0 或 2（即 0 和 2 数量之和）。

输入描述：

输入的前两个数字是矩阵大小。后面的数字是矩阵内容。

输出描述：

返回矩阵中非 1 的元素个数

补充说明：

m 和 n 不会超过 30(含 30)。

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

public class Main {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		int m = sc.nextInt();
		int n = sc.nextInt();
		int[][] arr = new int[m][n];
		for (int i = 0; i < m; i++) {
			for (int j = 0; j < n; j++) {
				arr[i][j] = sc.nextInt();
			}
```

```
}
arr[0][0] = 1;

int count = 0;
Queue<int[]> queue = new LinkedList<>();
for (int i = 0; i < m; i++) {
	for (int j = 0; j < n; j++) {
		if (arr[i][j] == 1){
			queue.add(new int[] {i,j});
		}
	}
}
while (!queue.isEmpty()){
	int size = queue.size();
	for (int i = 0; i < size; i++) {
		int[] temp = queue.poll();
		int r = temp[0];
		int c = temp[1];
		if (r-1>=0 && arr[r-1][c] == 0 ){
			arr[r-1][c] = 1;
			queue.add(new int[] {r-1,c});
		}
		if (r+1<m && arr[r+1][c] == 0){
			arr[r+1][c] = 1;
			queue.add(new int[] {r+1,c});
		}
		if (c-1>=0 && arr[r][c-1]==0){
			arr[r][c-1] = 1;
			queue.add(new int[] {r,c-1});
		}
		if (c+1<n && arr[r][c+1]==0){
			arr[r][c+1] = 1;
			queue.add(new int[] {r,c+1});
		}

	}
}
for (int i = 0; i < m; i++) {
	for (int j = 0; j < n; j++) {
		if (arr[i][j] == 1){
			count++;
		}
	}
}
```

```
            System.out.println(m*n - count);;

        }
    }
```