Java-数组递归-存在一个 m*n 的二维数组
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)。
示例 1
输入：
4 4
0 0 0 0
0 2 2 2
0 2 0 0
0 2 0 0
输出：
9
说明：
输入数字前两个数字是矩阵大小。后面的是数字是矩阵内容。
这个矩阵的内容如下：
{
  0,0,0,0
  0,2,2,2
  0,2,0,0
  0,2,0,0
}
起始位置 (0,0)被修改为 1 后，最终只能同化矩阵为：
{
  1,1,1,1
  1,2,2,2
  1,2,0,0
  1,2,0,0
}
所以矩阵中非 1 的元素个数为 9。

```
import java.util.*;

public class Main {
    public static int[][] directions = {{0, 1}, {0, -1}, {1, 0}, {-1, 0}};
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
```

```
            int m = in.nextInt();
            int n = in.nextInt();

            int[][] arr = new int[m][n];
            for (int i = 0; i < m; i++) {
                  for (int j = 0; j < n; j++) {
                        arr[i][j] = in.nextInt();
                  }
            }

            LinkedList<Integer[]> list = new LinkedList<>();
            list.add(new Integer[] {0, 0});
            arr[0][0] = 1;
            int count = 1;
            while (list.size() > 0) {
                  Integer[] item = list.removeFirst();
                  int x = item[0];
                  int y = item[1];

                  for (int[] direction : directions) {
                        int newX = x + direction[0];
                        int newY = y + direction[1];
                        if (newX >= 0 && newX < m && newY >= 0 && newY < n && arr[newX][newY]
== 0) {
                              arr[newX][newY] = 1;
                              list.add(new Integer[] {newX, newY});
                              count++;
                        }
                  }
            }

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


}
```