# [岛屿的最大面积](https://leetcode-cn.com/explore/interview/card/bytedance/243/array-and-sorting/1034/)

**头条重点**

## 题目

给定一个包含了一些 0 和 1的非空二维数组 grid , 一个 岛屿 是由四个方向 (水平或垂直) 的 1 (代表土地) 构成的组合。你可以假设二维矩阵的四个边缘都被水包围着。

找到给定的二维数组中最大的岛屿面积。(如果没有岛屿，则返回面积为0。)

示例 1:

[[0,0,1,0,0,0,0,1,0,0,0,0,0],
 [0,0,0,0,0,0,0,1,1,1,0,0,0],
 [0,1,1,0,1,0,0,0,0,0,0,0,0],
 [0,1,0,0,1,1,0,0,1,0,1,0,0],
 [0,1,0,0,1,1,0,0,1,1,1,0,0],
 [0,0,0,0,0,0,0,0,0,0,1,0,0],
 [0,0,0,0,0,0,0,1,1,1,0,0,0],
 [0,0,0,0,0,0,0,1,1,0,0,0,0]]
对于上面这个给定矩阵应返回 6。注意答案不应该是11，因为岛屿只能包含水平或垂直的四个方向的‘1’。

示例 2:

[[0,0,0,0,0,0,0,0]]
对于上面这个给定的矩阵, 返回 0。

注意: 给定的矩阵 grid 的长度和宽度都不超过 50。

## 解题思路

1. 通过循环遍历，找到 1
2. 再通过递归遍历该 1 临近的所有 1，并计算总面积

private static int[][] steps = new int[][]{{1, 0}, {0, 1}, {-1, 0}, {0, -1}};

/\*\*
 \* 上学时做过，属于图的 DFS
 \* @param grid
 \* @return
 \*/
public static int maxAreaOfIsland(int[][] grid) {
 if (grid.length == 0) {
 return 0;
 }

 int[][] marks = new int[grid.length][grid[0].length];
 for (int[] mark : marks) {
 Arrays.fill(mark, 0);
 }

 Wrapper<Integer> maxArea = new Wrapper<>(0);

 for (int i = 0; i < grid.length; i++) {
 for (int j = 0; j < grid[i].length; j++) {
 p(grid, marks, i, j, new Wrapper<>(0), maxArea);
 }
 }
 return maxArea.v;
}

private static void p(int[][] grid, int[][] mark,
 int i, int j, Wrapper<Integer> curArea,
 Wrapper<Integer> maxArea) {
 if (i < 0 || j < 0 || i >= grid.length || j >= grid[0].length) {
 return;
 }

 if (grid[i][j] == 1 && mark[i][j] == 0) {
 curArea.v++;
 maxArea.v = Math.max(maxArea.v, curArea.v);
 } else {
 return;
 }

 for (int[] step : steps) {
 mark[i][j] = 1;
 p(grid, mark, i + step[0], j + step[1], curArea, maxArea);
// mark[i][j] = 0;
 }

}

private static final class Wrapper<V> {
 V v;

 public Wrapper(V v) {
 this.v = v;
 }
}