题目地址(695. 岛屿的最大面积)

https://leetcode-cn.com/problems/max-area-of-island/

入选理由

1. BFS 和 DFS 大家都可以试试，一般都是通的

标签

- BFS
- DFS

难度

- 中等

题目描述

```
给定一个包含了一些 0 和 1 的非空二维数组 grid 。
一个 岛屿 是由一些相邻的 1 (代表土地) 构成的组合，
这里的「相邻」要求两个 1 必须在水平或者竖直方向上相邻。
你可以假设 grid 的四个边缘都被 0（代表水）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。
```

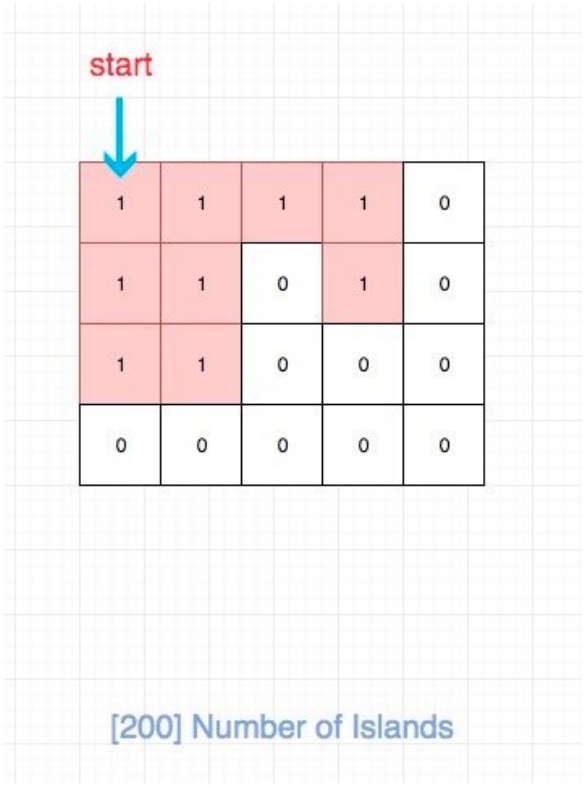
注意：给定的矩阵 grid 的长度和宽度都不超过 50

公司

- 字节跳动

思路

和 **200. 岛屿数** 思路一样， 只不过 200 是求小岛个数，这个是求小岛最大面积，这也就是多一个变量记录一下的事情。



这道题目仍然可以采用原位修改的方式避免记录 visited 的开销。我们的做法是将 grid[i][j] = 0，需要注意的是，我们无需重新将 grid[i][j] = 1, 因为题目没有这个要求。另外如果你这么做的话，也会产生 bug， 比如：

1**1**1
111

上面加粗的 1，如果在遍历了上下左右邻居之后，将 0，重新变成 1。那么就会被重复计算。如下，粗体上方的 1 就会被计算多次

111
10**1**

代码

- 语言支持：Python，CPP, Java, JS

Python Code：

```python
class Solution:
    def maxAreaOfIsland(self, grid: List[List[int]]) -> int:
        m = len(grid)
        if m == 0: return 0
        n = len(grid[0])
        ans = 0
        def dfs(i, j):
            if i < 0 or i >= m or j < 0 or j >= n: return 0
            if grid[i][j] == 0: return 0
            grid[i][j] = 0
            top = dfs(i + 1, j)
            bottom = dfs(i - 1, j)
            left = dfs(i, j - 1)
            right = dfs(i, j + 1)
            return 1 + sum([top, bottom, left, right])
        for i in range(m):
            for j in range(n):
                ans = max(ans, dfs(i, j))
        return ans
```

CPP Code:

```cpp
class Solution {
public int maxAreaOfIsland(int[][] grid) {
    int max = 0;
    for (int i = 0; i < grid.length; i++) {
    for (int j = 0; j < grid[0].length; j++) {
        max = Math.max(max, getIslandArea(i, j, grid));
    }
    return max;
}

private int getIslandArea(int i, int j, int[][] grid) {
    if (i < 0 || i >= grid.length || j < 0 || j >= grid[0].length || grid[i][j] == 0) {
        return 0;
    }
    grid[i][j] = 0;
    return 1 + getIslandArea(i - 1, j, grid) + getIslandArea(i + 1, j, grid) + getIslandArea(i, j - 1, grid)
            + getIslandArea(i, j + 1, grid);
}

}
```

Java Code:

```java
class Solution {
    public int maxAreaOfIsland(int[][] grid) {
        int maxArea = 0;
        for (int r = 0; r < grid.length; r++) {
            for (int c = 0; c < grid[0].length; c++) {
                if (grid[r][c] != 0) {
                    int[] curArea = new int[1];
                    dfs(grid, r, c, curArea);
                    maxArea = Math.max(curArea[0], maxArea);
                }
            }
        }
        return maxArea;
    }

    private void dfs(int[][] grid, int curRow, int curCol, int[] curArea) {
        if (curRow < 0 || curRow >= grid.length || curCol < 0 || curCol >= grid[0].length || grid[curRow][curCol] == 0)
            return;
        }
        // mark
        grid[curRow][curCol] = 0;
        curArea[0] += 1;

        dfs(grid, curRow - 1, curCol, curArea);
        dfs(grid, curRow + 1, curCol, curArea);
        dfs(grid, curRow, curCol - 1, curArea);
        dfs(grid, curRow, curCol + 1, curArea);
    }
}
```

JS Code：

```javascript
var maxAreaOfIsland = function (grid) {
  let x = grid[0].length,
    y = grid.length;
  let res = 0;
  function dfs(i, j) {
    // i 是 y ,j 是 x
    if (i >= y || j >= x || j < 0 || i < 0 || grid[i][j] != 1) return 0;
    grid[i][j] = -1;
    return 1 + dfs(i + 1, j) + dfs(i - 1, j) + dfs(i, j - 1) + dfs(i, j + 1);
  }
  for (let i = 0; i < y; i++) {
    // i 是 y ,j 是 x
    for (let j = 0; j < x; j++) {
```

```
      if (grid[i][j]) {
        res = Math.max(res, dfs(i, j));
      }
    }
  }
  return res;
};
```

复杂度分析

- 时间复杂度：$O(m * n)$
- 空间复杂度：$O(m * n)$

大家对此有何看法，欢迎给我留言，我有时间都会一一查看回答。更多算法套路可以访问我的 LeetCode 题解仓库：https://github.com/azl397985856/leetcode 。 目前已经 37K star 啦。 大家也可以关注我的公众号《力扣加加》带你啃下算法这块硬骨头。