LeetCode 图解 | 200. 岛屿数量

题目描述

给定一个由 `'1'`（陆地）和 `'0'`（水）组成的的二维网格，计算岛屿的数量。一个岛被水包围，并且它是通过水平方向或垂直方向上相邻的陆地连接而成的。你可以假设网格的四个边均被水包围。

示例 1:

```
输入：
11110
11010
11000
00000

输出：1
```

示例 2:

```
输入：
11000
11000
00100
00011

输出：3
```

题目解析

这道题的主要思路是深度优先搜索。每次走到一个是 1 的格子，就搜索整个岛屿。

网格可以看成是一个无向图的结构，每个格子和它上下左右的四个格子相邻。如果四个相邻的格子坐标合法，且是陆地，就可以继续搜索。

在深度优先搜索的时候要注意避免重复遍历。我们可以把已经遍历过的陆地改成 2，这样遇到 2 我们就知道已经遍历过这个格子了，不进行重复遍历。

每遇到一个陆地格子就进行深度优先搜索，最终搜索了几次就知道有几个岛屿。

动画理解

参考代码

```
class Solution {
    public int numIslands(char[][] grid) {
        if (grid.length == 0 || grid[0].length == 0) {
            return 0;
        }
```

```
        int count = 0;
        for (int r = 0; r < grid.length; r++) {
            for (int c = 0; c < grid[0].length; c++) {
                if (grid[r][c] == '1') {
                    dfs(grid, r, c);
                    count++;
                }
            }
        }
        return count;
    }

    void dfs(char[][] grid, int r, int c) {
        if (!(0 <= r && r < grid.length && 0 <= c && c < grid[0].length)) {
            return;
        }
        if (grid[r][c] != '1') {
            return;
        }
        grid[r][c] = '2';
        dfs(grid, r - 1, c);
        dfs(grid, r + 1, c);
        dfs(grid, r, c - 1);
        dfs(grid, r, c + 1);
    }
}
```

复杂度分析

设网格的边长为 n，则时间复杂度为 $O(n^2)$。