# 题目

给你一个由 '1'（陆地）和 '0'（水）组成的的二维网格，请你计算网格中岛屿的数量。

岛屿总是被水包围，并且每座岛屿只能由水平方向和/或竖直方向上相邻的陆地连接形成。

此外，你可以假设该网格的四条边均被水包围。

示例 1：

输入：grid = [

['1','1','1','1','0'],

['1','1','0','1','0'],

['1','1','0','0','0'],

['0','0','0','0','0']

]

输出：1

示例 2：

输入：grid = [

['1','1','0','0','0'],

['1','1','0','0','0'],

['0','0','1','0','0'],

['0','0','0','1','1']

]

输出：3

提示：

m == grid.length

n == grid[i].length

1 <= m, n <= 300

grid[i][j] 的值为 '0' 或 '1'

# 分析

要解决计算网格中岛屿数量的问题，我们可以使用深度优先搜索（DFS）或广度优先搜索（BFS）来遍历每个岛屿，通过标记已访问的陆地避免重复计数。核心思路是找到所有相互连接的陆地区域（岛屿），并统计其数量。

解题思路

1、问题分析：

岛屿由相邻的'1'（水平或竖直方向）组成，且被'0'（水）包围。网格的四条边也被视为水。因此，我们需要遍历整个网格，每当遇到未访问的'1'时，就启动一次搜索（DFS或BFS），探索整个岛屿并标记所有相关的'1'为已访问（例如改为'0'），每完成一次搜索，岛屿数量加1。

2、算法选择：

- DFS：通过递归遍历当前陆地的上下左右四个方向，将所有相连的'1'标记为'0'（已访问），直到所有相连的陆地都被处理。

- BFS：使用队列存储当前陆地的坐标，依次出队并检查其四个方向的相邻单元格，将相连的'1'入队并标记为'0'，直到队列为空。

两种方法均可高效解决问题，此处以DFS为例实现。

3、步骤拆解：

- 遍历网格的每个单元格。

- 若单元格为'1'，则启动DFS，将所有相连的'1'标记为'0'（避免重复计数），并将岛屿数量加1。

- 遍历结束后，返回岛屿数量。

代码实现（DFS，含详细注释）：

class Solution {

public:

int numIslands(vector<vector<char>>& grid) {

if (grid.empty()) return 0; // 空网格，岛屿数量为0

int m = grid.size(); // 网格行数

int n = grid[0].size(); // 网格列数

int count = 0; // 岛屿数量计数器

// 遍历网格的每个单元格

for (int i = 0; i < m; ++i) {

for (int j = 0; j < n; ++j) {

if (grid[i][j] == '1') {

// 发现未访问的陆地，启动DFS探索整个岛屿

dfs(grid, i, j, m, n);

count++; // 岛屿数量加1

}

}

}

return count;

}

private:

// DFS：将当前陆地(i,j)及所有相连的陆地标记为'0'（已访问）

void dfs(vector<vector<char>>& grid, int i, int j, int m, int n) {

// 边界检查：超出网格范围或当前单元格不是陆地（'1'），直接返回

if (i < 0 || i >= m || j < 0 || j >= n || grid[i][j] != '1') {

return;

}

// 标记当前陆地为已访问（改为'0'）

grid[i][j] = '0';

// 递归探索上下左右四个方向的相邻单元格

dfs(grid, i - 1, j, m, n); // 上

dfs(grid, i + 1, j, m, n); // 下

dfs(grid, i, j - 1, m, n); // 左

dfs(grid, i, j + 1, m, n); // 右

}

};

代码解释

1、网格遍历：外层双重循环遍历网格的每个单元格，当发现'1'时，说明找到一个新岛屿，触发DFS。

2、DFS标记：DFS函数递归访问当前单元格的上下左右四个方向，将所有相连的'1'改为'0'，确保这些陆地不会被重复计数。

3、边界检查：在DFS中，首先检查当前坐标是否超出网格范围或是否为非陆地（'0'），若满足则终止递归，避免越界或无效访问。

复杂度分析

- 时间复杂度：O(m\*n)，其中m和n分别为网格的行数和列数。每个单元格最多被访问一次（访问后标记为'0'），因此总时间与网格大小成正比。

- 空间复杂度：O(m\*n)，最坏情况下（网格全为'1'），DFS的递归栈深度可达m\*n（例如一条链状的陆地）。

该方法通过DFS高效标记并计数所有岛屿，逻辑清晰且适合网格规模在300x300以内的问题约束。