1254. 统计封闭岛屿的数目

ITCharge 大约 2 分钟

- 标签：深度优先搜索、广度优先搜索、并查集、数组、矩阵
- 难度：中等

题目链接

- 1254. 统计封闭岛屿的数目 - 力扣

题目大意

描述：给定一个二维矩阵 `grid` ，每个位置要么是陆地（记号为 `0` ）要么是水域（记号为 `1` ）。

我们从一块陆地出发，每次可以往上下左右 `4` 个方向相邻区域走，能走到的所有陆地区域，我们将其称为一座「岛屿」。

如果一座岛屿完全由水域包围，即陆　　缘上下左右所有相邻区域都是水域，那么我们将其称为「封闭岛屿」。

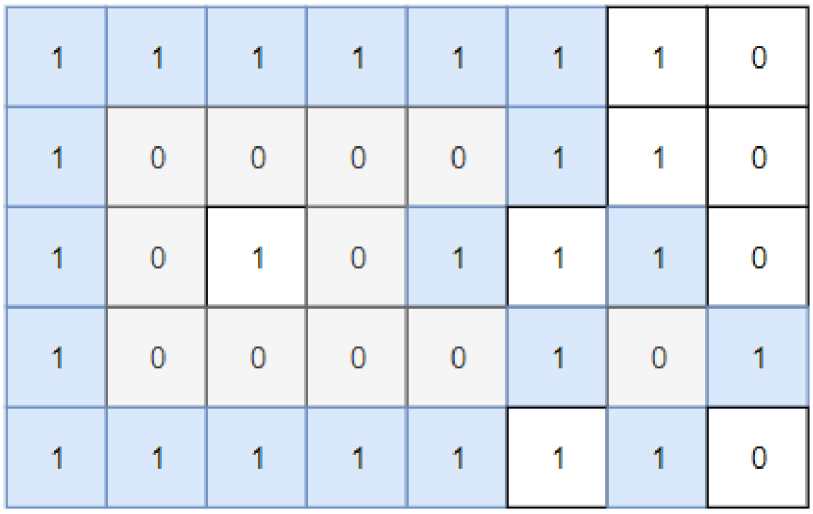
要求：返回封闭岛屿的数目。

说明：

- $1 \leq grid.length, grid[0].length \leq 100$。
- $0 \leq grid[i][j] \leq 1$。

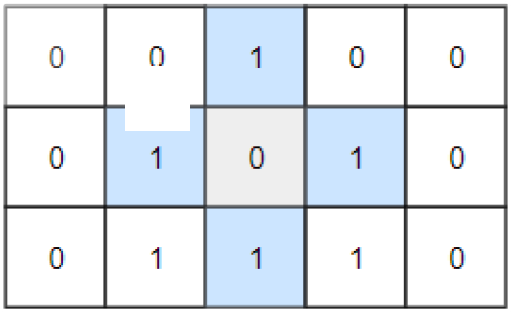
示例：

- 示例 1：



```py
输入：grid = [[1,1,1,1,1,1,1,0],[1,0,0,0,0,1,1,0],[1,0,1,0,1,1,1,0],
[1,0,0,0,0,1,0,1],[1,1,1,1,1,1,1,0]]
输出：2
解释：灰色区域的岛屿是封闭岛屿，因为这座岛屿完全被水域包围（即被 1 区域包围）。
```

- 示例 2：



```py
输入：grid = [[0,0,1,0,0],[0,1,0,1,0],[0,1,1,1,0]]
输出：1
```

解题思路

思路 1：深度优先搜索

1. 从 `grid[i][j] == 0` 的位置出发，使用深度优先搜索的方法遍历上下左右四个方向上相邻区域情况。
    1. 如果上下左右都是 `grid[i][j] == 1` ，则返回 `True` 。

    2. 如果有一个以上方向的 `grid[i][j] == 0` ，则返回 `False` 。
    3. 遍历之后将当前陆地位置置为 `1` ，表示该位置已经遍历过了。
2. 最后统计出上下左右都满足 `grid[i][j] == 1` 的情况数量，即为答案。

思路 1：代码

```py
class Solution:
    directs = [(0, 1), (0, -1), (1, 0), (-1, 0)]

    def dfs(self, grid, i, j):
        n, m = len(grid), len(grid[0])
        if i < 0 or i >= n or j < 0 or j >= m:
            return False
        if grid[i][j] == 1:
            return True
        grid[i][j] = 1

        res = True
        for direct in self.directs:
            new_i = i + direct[0]
            new_j = j + direct[1]
            if not self.dfs(grid, new_i, new_j):
                res = False
        return res

    def closedIsland(self, grid: List[List[int]]) -> int:
        res = 0
        for i in range(len(grid)):
            for j in range(len(grid[0])):
                if grid[i][j] == 0 and self.dfs(grid, i, j):
                    res += 1

        return res
```

思路 1：复杂度分析

- **时间复杂度**：$O(m \times n)$。其中 m 和 n 分别为行数和列数。
- **空间复杂度**：$O(m \times n)$。