9. Islands

1020. Number of Enclaves

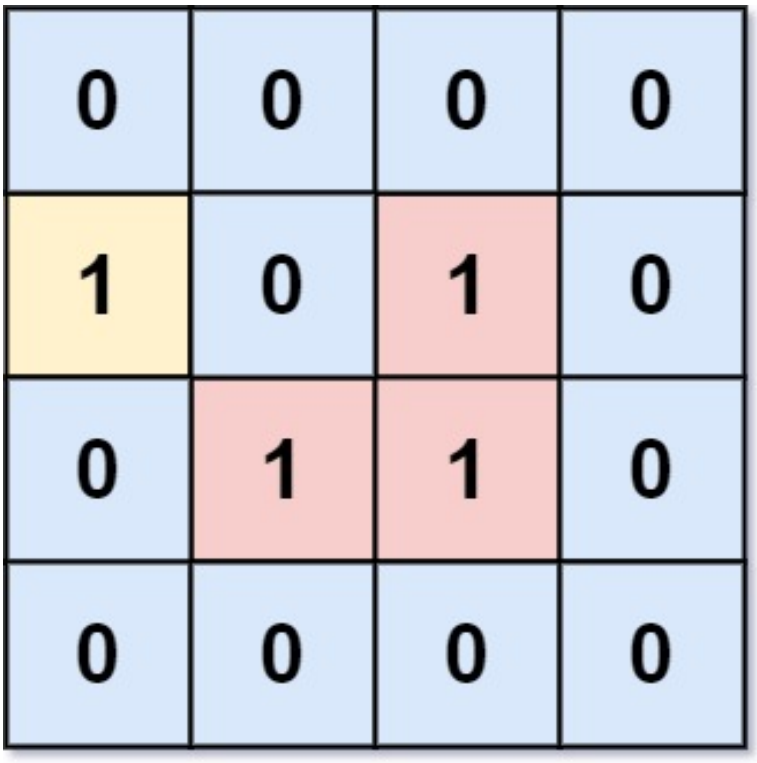
Created	@November 29, 2022 3:49 PM
Difficulty	Medium
LC Url	https://leetcode.com/problems/number-of-enclaves/
Importance	
Tag	DFS Island
Reference	https://labuladong.github.io/algo/4/31/107/

You are given an `m x n` binary matrix `grid`, where `0` represents a sea cell and `1` represents a land cell.

A **move** consists of walking from one land cell to another adjacent (**4-directionally**) land cell or walking off the boundary of the `grid`.

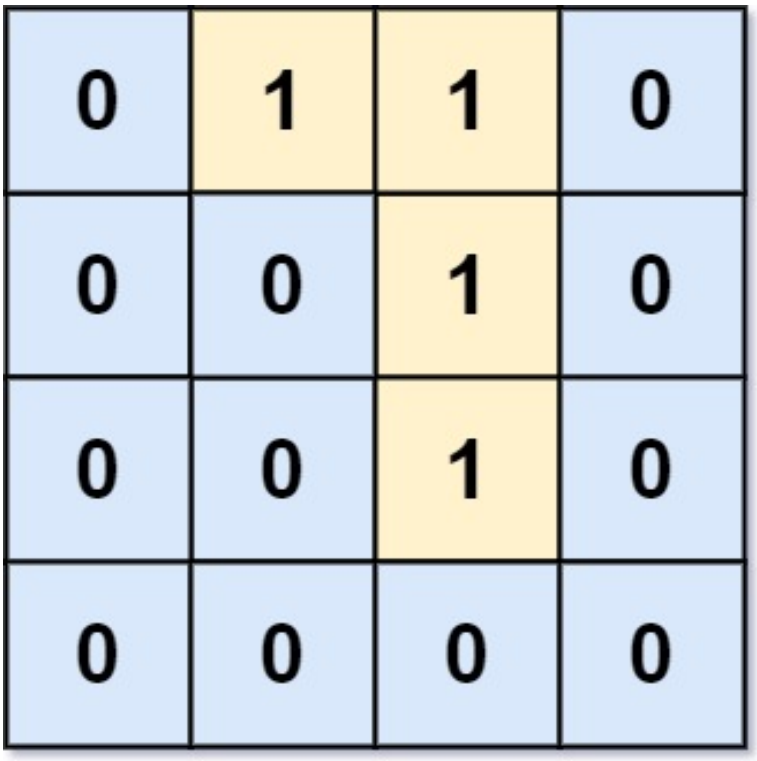
Return *the number of land cells in* `grid` *for which we cannot walk off the boundary of the grid in any number of* ***moves***.

Example 1:




```
Input: grid = [[0,0,0,0],[1,0,1,0],[0,1,1,0],[0,0,0,0]]
Output: 3
Explanation: There are three 1s that are enclosed by 0s, and one 1 that is not enclosed be
cause its on the boundary.
```

Example 2:




```
Input: grid = [[0,1,1,0],[0,0,1,0],[0,0,1,0],[0,0,0,0]]
Output: 0
Explanation: All 1s are either on the boundary or can reach the boundary.
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 500`
- `grid[i][j]` is either `0` or `1`.

Solution

这道岛屿题目的解法稍微改改就可以解决力扣第 1020 题「飞地的数量」，这题不让你求封闭岛屿的数量，而是求封闭岛屿的面积总和。

其实思路都是一样的，先把靠边的陆地淹掉，然后去数剩下的陆地数量就行了，注意第 1020 题中 `1` 代表陆地，`0` 代表海水：

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

    def numEnclaves(self, grid: List[List[int]]) -> int:
        res = 0
        m = len(grid)
        if m == 0:
            return res
        n = len(grid[0])

        for i in range(m):
            # 把靠左边的岛屿淹掉
            self.dfs(grid, i, 0)
            # 把靠右边的岛屿淹掉
            self.dfs(grid, i, n - 1)

        for j in range(n):
            # 把靠上边的岛屿淹掉
            self.dfs(grid, 0, j)
            # 把靠下边的岛屿淹掉
            self.dfs(grid, m - 1, j)

        # 数一数剩下的陆地
        for i in range(m):
            for j in range(n):
                if grid[i][j] == 1:
                    res += 1

        return res

    def dfs(self, grid, i, j):
        """
        从 (i, j) 开始，将与之相邻的陆地都变成海水
        """
        if not self.is_valid(grid, i, j):
            return

        # 已经是海水了
        if grid[i][j] == 0:
            return

        # 将 (i, j) 变成海水
        grid[i][j] = 0

        # 淹没上下左右的陆地
        for direction in self.directions:
            cur_i, cur_j = i + direction[0], j + direction[1]
            self.dfs(grid, cur_i, cur_j)

    def is_valid(self, grid, i, j):
        """
        Check whether (i, j) is in the domain
```

```
        """
        m, n = len(grid), len(grid[0])
        if 0 <= i < m and 0 <= j < n:
            return True
        return False
```

1254. Number of Closed Islands

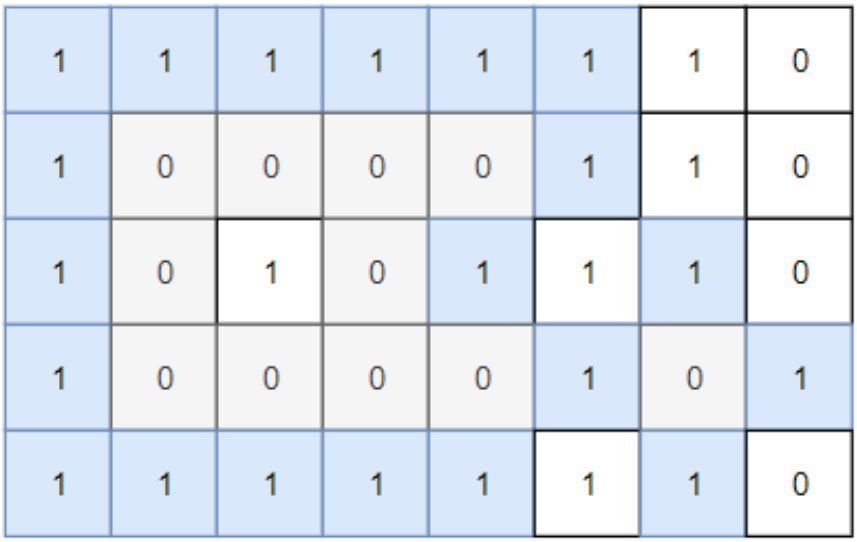
1254. Number of Closed Islands

Created	@November 29, 2022 3:25 PM
Difficulty	Medium
LC Url	https://leetcode.com/problems/number-of-closed-islands/
Importance	
Tag	DFS Island
Reference	

Given a 2D `grid` consists of `0s` (land) and `1s` (water). An *island* is a maximal 4-directionally connected group of `0` `s` and a *closed island* is an island **totally** (all left, top, right, bottom) surrounded by `1s.`

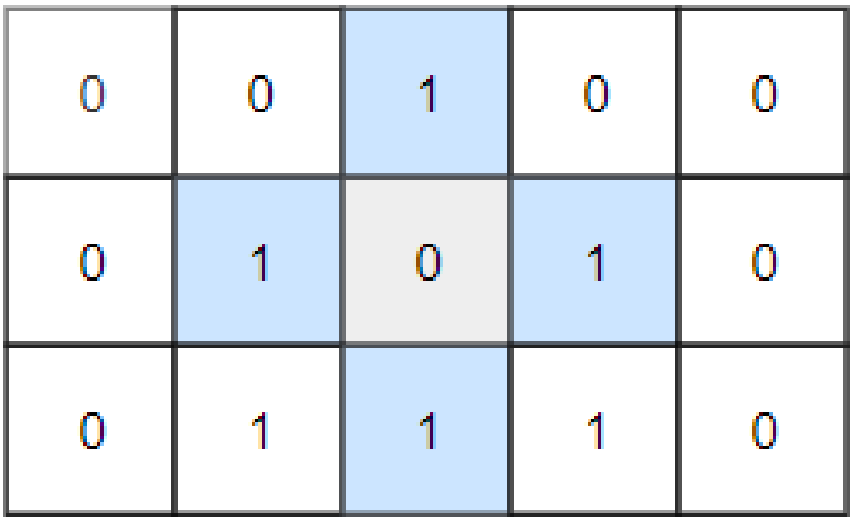
Return the number of *closed islands*.

Example 1:




```
Input: 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]]
Output: 2
Explanation:
Islands in gray are closed because they are completely surrounded by water (group of 1s).
```

Example 2:




```
Input: grid = [[0,0,1,0,0],[0,1,0,1,0],[0,1,1,1,0]]
Output: 1
```

Example 3:

```
Input: grid = [[1,1,1,1,1,1,1],
               [1,0,0,0,0,0,1],
               [1,0,1,1,1,0,1],
               [1,0,1,0,1,0,1],
               [1,0,1,1,1,0,1],
               [1,0,0,0,0,0,1],
               [1,1,1,1,1,1,1]]
Output: 2
```

Constraints:

- `1 <= grid.length, grid[0].length <= 100`
- `0 <= grid[i][j] <=1`

Solution

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

    def closedIsland(self, grid: List[List[int]]) -> int:
        res = 0
        m = len(grid)
        if m == 0:
            return res
        n = len(grid[0])

        for i in range(m):
            # 把靠左边的岛屿淹掉
            self.dfs(grid, i, 0)
            # 把靠右边的岛屿淹掉
            self.dfs(grid, i, n - 1)

        for j in range(n):
            # 把靠上边的岛屿淹掉
            self.dfs(grid, 0, j)
            # 把靠下边的岛屿淹掉
            self.dfs(grid, m - 1, j)

        # 遍历 grid，剩下的岛屿都是封闭岛屿
        for i in range(m):
            for j in range(n):
                if grid[i][j] == 0:
                    res += 1
                    self.dfs(grid, i, j)

        return res

    def dfs(self, grid, i, j):
        """
        从 (i, j) 开始，将与之相邻的陆地都变成海水
        """
        if not self.is_valid(grid, i, j):
            return

        # 已经是海水了
        if grid[i][j] == 1:
            return

        # 将 (i, j) 变成海水
        grid[i][j] = 1

        # 淹没上下左右的陆地
        for direction in self.directions:
            cur_i, cur_j = i + direction[0], j + direction[1]
            self.dfs(grid, cur_i, cur_j)

    def is_valid(self, grid, i, j):
        """
```

```
        Check whether (i, j) is in the domain
        """
        m, n = len(grid), len(grid[0])
        if 0 <= i < m and 0 <= j < n:
            return True
        return False
```

1020. Number of Enclaves

1905. Count Sub Islands

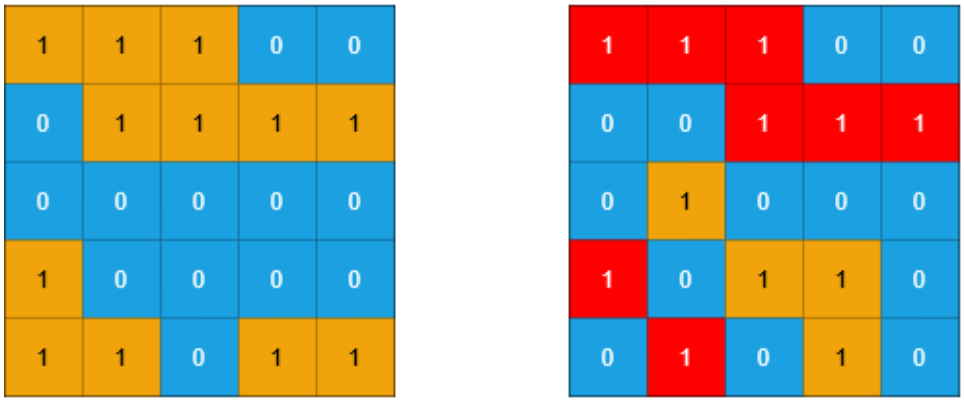
Created	@November 29, 2022 4:50 PM
Difficulty	Medium
LC Url	https://leetcode.com/problems/count-sub-islands/
Importance	
Tag	DFS Island
Reference	https://labuladong.github.io/algo/4/31/107/

You are given two `m x n` binary matrices `grid1` and `grid2` containing only `0`'s (representing water) and `1`'s (representing land). An **island** is a group of `1`'s connected **4-directionally** (horizontal or vertical). Any cells outside of the grid are considered water cells.

An island in `grid2` is considered a **sub-island** if there is an island in `grid1` that contains **all** the cells that make up **this** island in `grid2`.

Return the ***number*** *of islands in* `grid2` *that are considered* ***sub-islands***.

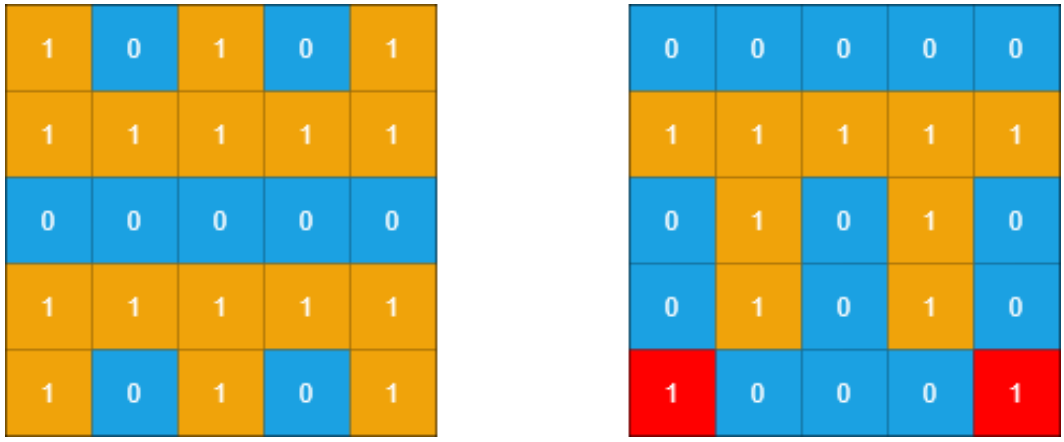
Example 1:




```
Input: grid1 = [[1,1,1,0,0],[0,1,1,1,1],[0,0,0,0,0],[1,0,0,0,0],[1,1,0,1,1]], grid2 = [[1,
1,1,0,0],[0,0,1,1,1],[0,1,0,0,0],[1,0,1,1,0],[0,1,0,1,0]]
Output: 3
Explanation:In the picture above, the grid on the left is grid1 and the grid on the right
```

```
 is grid2.
The 1s colored red in grid2 are those considered to be part of a sub-island. There are thr
ee sub-islands.
```

Example 2:




```
Input: grid1 = [[1,0,1,0,1],[1,1,1,1,1],[0,0,0,0,0],[1,1,1,1,1],[1,0,1,0,1]], grid2 = [[0,
0,0,0,0],[1,1,1,1,1],[0,1,0,1,0],[0,1,0,1,0],[1,0,0,0,1]]
Output: 2
Explanation:In the picture above, the grid on the left is grid1 and the grid on the right
 is grid2.
The 1s colored red in grid2 are those considered to be part of a sub-island. There are two
sub-islands.
```

Constraints:

- `m == grid1.length == grid2.length`
- `n == grid1[i].length == grid2[i].length`
- `1 <= m, n <= 500`
- `grid1[i][j]` and `grid2[i][j]` are either `0` or `1`.

Solution

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

    def countSubIslands(self, grid1: List[List[int]], grid2: List[List[int]]) -> int:
```

```
        res = 0
        m = len(grid1)
        if m == 0:
            return res
        n = len(grid1[0])

        for i in range(m):
            for j in range(n):
                if grid1[i][j] == 0 and grid2[i][j] == 1:
                    # this island must not be a sub island
                    # so we change it to sea
                    self.dfs(grid2, i, j)

        # now, every islands is sub island
        # count the number of islands for grid2
        for i in range(m):
            for j in range(n):
                if grid2[i][j] == 1:
                    res += 1
                    self.dfs(grid2, i, j)

        return res

    def dfs(self, grid, i, j):
        if not self.is_valid(grid, i, j):
            return

        if grid[i][j] == 0:
            return

        grid[i][j] = 0

        for direction in self.directions:
            cur_i, cur_j = i + direction[0], j +direction[1]
            self.dfs(grid, cur_i, cur_j)

    def is_valid(self, grid, i, j):
        m, n = len(grid), len(grid[0])
        if 0 <= i < m and 0 <= j < n:
            return True
        return False
```

200. Number of Islands

Created	@November 28, 2022 11:27 PM
Difficulty	Medium
LC Url	https://leetcode.com/problems/number-of-islands/
Importance	
Tag	DFS Island
Reference	https://labuladong.github.io/algo/4/31/107/

Given an `m x n` 2D binary grid `grid` which represents a map of `'1'`s (land) and `'0'`s (water), return *the number of islands*.

An **island** is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

Example 1:

```
Input: grid = [
  ["1","1","1","1","0"],
  ["1","1","0","1","0"],
  ["1","1","0","0","0"],
  ["0","0","0","0","0"]
]
Output: 1
```

Example 2:

```
Input: grid = [
  ["1","1","0","0","0"],
  ["1","1","0","0","0"],
  ["0","0","1","0","0"],
  ["0","0","0","1","1"]
]
Output: 3
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 300`
- `grid[i][j]` is `'0'` or `'1'`.

Solution

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

    def numIslands(self, grid: List[List[str]]) -> int:
        res = 0
        m = len(grid)
        if m == 0:
            return res
        n = len(grid[0])

        # 遍历grid
        for i in range(m):
            for j in range(n):
                if grid[i][j] == '1':
                    # 每发现一个岛屿，岛屿总数加1
                    res += 1
                    # 用DFS将岛屿标记
                    self.dfs(grid, i, j)

        return res

    def dfs(self, grid, i, j):
        m = len(grid)
        n = len(grid[0])

        # 判断是否在区域内
        if i < 0 or j < 0 or i >= m or j >= n:
            return

        # 如果这个格子不是岛屿，直接返回
        if grid[i][j] != '1':
            return

        # 将格子标记为“已遍历过”
        grid[i][j] = '2'
        # 访问上下左右四个相邻节点
        for direction in self.directions:
```

```
            cur_i, cur_j = i + direction[0], j + direction[1]
            self.dfs(grid, cur_i, cur_j)
```

力扣

https://leetcode.cn/problems/number-of-islands/solution/dao-yu-lei-wen-ti-de-tong-yong-jie-fa-dfs-bian-li-/

695. Max Area of Island

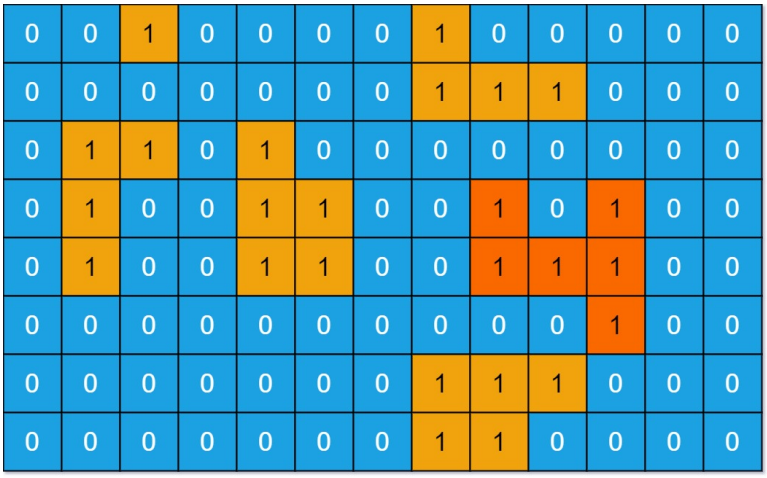
Created	@November 29, 2022 4:21 PM
Difficulty	Medium
LC Url	https://leetcode.com/problems/max-area-of-island/
Importance	
Tag	DFS Island
Reference	https://labuladong.github.io/algo/4/31/107/

You are given an `m x n` binary matrix `grid`. An island is a group of `1`'s (representing land) connected **4-directionally** (horizontal or vertical.) You may assume all four edges of the grid are surrounded by water.

The **area** of an island is the number of cells with a value `1` in the island.

Return *the maximum **area** of an island in* `grid`. If there is no island, return `0`.

Example 1:




```
Input: grid = [[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]]
Output: 6
Explanation: The answer is not 11, because the island must be connected 4-directionally.
```

Example 2:

```
Input: grid = [[0,0,0,0,0,0,0,0]]
Output: 0
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 50`
- `grid[i][j]` is either `0` or `1`.

Solution

这题的大体思路和之前完全一样，只不过 `dfs` 函数淹没岛屿的同时，还应该想办法记录这个岛屿的面积。

我们可以给 `dfs` 函数设置返回值，记录每次淹没的陆地的个数，

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

    def maxAreaOfIsland(self, grid: List[List[int]]) -> int:
        # 记录岛屿的最大面积
        res = 0
        m = len(grid)
        if m == 0:
            return res
        n = len(grid[0])

        for i in range(m):
            for j in range(n):
                if grid[i][j] == 1:
                    # 淹没岛屿，并更新最大岛屿面积
                    res = max(res, self.dfs(grid, i, j))

        return res

    def is_valid(self, grid, i, j):
        """
        判断是否超出索引边界
```

```
        """
        m, n = len(grid), len(grid[0])
        if 0 <= i < m and 0 <= j < n:
            return True
        return False

    def dfs(self, grid, i, j):
        """
        淹没与 (i, j) 相邻的陆地，并返回淹没的陆地面积
        """
        if not self.is_valid(grid, i, j):
            # 超出边界
            return 0

        if grid[i][j] == 0:
            # 已经是海水了
            return 0

        # 将 (i, j) 变成海水
        grid[i][j] = 0

        cnt = 1
        for direction in self.directions:
            cur_i, cur_j = i + direction[0], j + direction[1]
            cnt += self.dfs(grid, cur_i, cur_j)

        return cnt
```