695 Max Area of Island

2018年5月3日 18:53

Given a non-empty 2D array grid of 0's and 1's, 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.
Find the maximum area of an island in the given 2D array. (If there is no island, the maximum area is 0.)
Example 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]]
Given the above grid, return 6. Note the answer is not 11, because the island must be connected 4-directionally.
Example 2:
[[0,0,0,0,0,0,0,0]]
Given the above grid, return 0.
Note: The length of each dimension in the given grid does not exceed 50.

来自 <https://leetcode.com/problems/max-area-of-island/description/>

给定一个包含了一些 0 和 1的非空二维数组 grid , 一个 **岛屿** 是由四个方向 (水平或垂直) 的 1 (代表土地) 构成的组合。你可以假设二维矩阵的四个边缘都被水包围着。
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。

Solution for Python3:

```python
class Solution1:
    def maxAreaOfIsland(self, grid):
        """
        :type grid: List[List[int]]
        :rtype: int
        """
        seen = set()
        def area(r, c):
            if not (0 <= r < len(grid) and 0 <= c < len(grid[0]) and (r,c) not in seen and grid[r][c]):
                return 0
            seen.add((r,c))
            return (1 + area(r+1, c) + area(r-1, c) + area(r, c-1) + area(r, c+1))

        return max(area(r, c) for r in range(len(grid)) for c in range(len(grid[0])))

class Solution2:
    def maxAreaOfIsland(self, grid):
        """
        :type grid: List[List[int]]
        :rtype: int
        """
        m, n = len(grid), len(grid[0])
        maximum = 0
        def dfs(i, j):
            if 0 <= i < m and 0 <= j < n and grid[i][j]:
                grid[i][j] = 0
                return 1 + dfs(i-1, j) + dfs(i+1, j) + dfs(i, j-1) + dfs(i, j+1)
            return 0
```

```
29
30          for i in range(m):
31             for j in range(n):
32                 if grid[i][j]:
33                     maximum = max(maximum, dfs(i,j))
34          return maximum
```

Solution for C++:

```
1 class Solution {
2 public:
3     int maxAreaOfIsland(vector<vector<int>>& grid) {
4         int max_area = 0;
5         for (int i = 0; i < grid.size(); i++) {
6             for (int j = 0; j < grid[0].size(); j++) {
7                 if (grid[i][j])
8                     max_area = max(max_area, AreaOfIsland(grid, i, j));
9             }
10        }
11        return max_area;
12    }
13    int AreaOfIsland(vector<vector<int>>& grid, int i, int j) {
14        if (i >= 0 && i < grid.size() && j >= 0 && j < grid[0].size() && grid[i][j]) {
15            grid[i][j] = 0;
16            return 1 + AreaOfIsland(grid, i+1, j) + AreaOfIsland(grid, i-1,j) + AreaOfIsland(grid, i, j+1) +
17 AreaOfIsland(grid, i, j-1);
18        }
19        return 0;
20    }
  };
```