463. Island Perimeter

Easy 6.1K 304

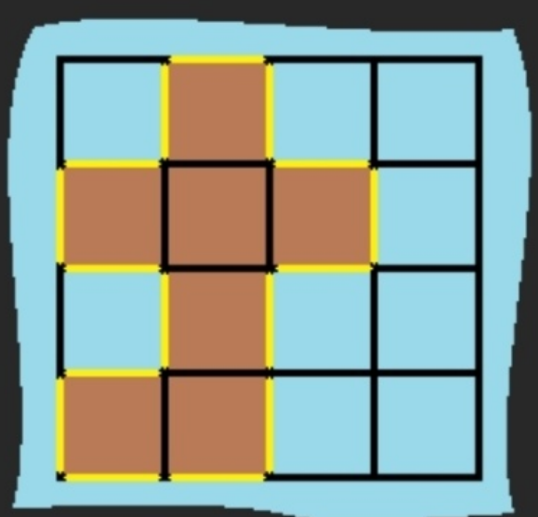
Companies

You are given `row x col` `grid` representing a map where `grid[i][j] = 1` represents land and `grid[i][j] = 0` represents water.

Grid cells are connected **horizontally/vertically** (not diagonally). The `grid` is completely surrounded by water, and there is exactly one island (i.e., one or more connected land cells).

The island doesn't have "lakes", meaning the water inside isn't connected to the water around the island. One cell is a square with side length 1. The grid is rectangular, width and height don't exceed 100. Determine the perimeter of the island.

Example 1:



> **Input:** grid = [[0,1,0,0],[1,1,1,0],[0,1,0,0],[1,1,0,0]]
> **Output:** 16
> **Explanation:** The perimeter is the 16 yellow stripes in the image above.

Example 2:

> **Input:** grid = [[1]]
> **Output:** 4

Example 3:

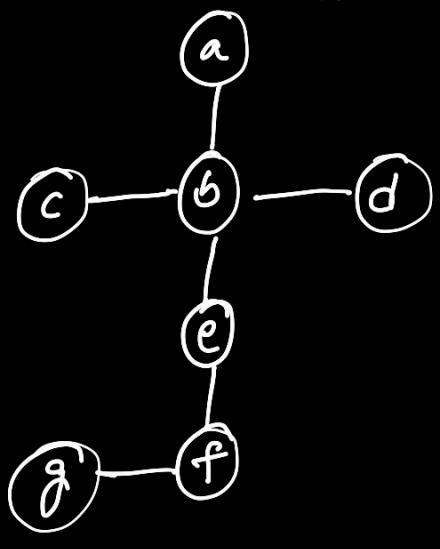
> **Input:** grid = [[1,0]]
> **Output:** 4

Constraints:

- `row == grid.length`
- `col == grid[i].length`
- `1 <= row, col <= 100`
- `grid[i][j]` is `0` or `1`.
- There is exactly one island in `grid`.

Accepted 467.4K | Submissions 668K | Acceptance Rate 70.0%

If we consider each land grid as one vertex,



If a vertex has only 1 adjacent vertex
then perimeter = 3 i.e. 4 - 1
If a vertex has 2 adjacent vertices
then perimeter = 2 4 - 2
Similarly if it has 3 adjacent vertices
then perimeter = 1 4 - 3
if it has 4 adjacent vertices
then perimeter = 0 4 - 4

```
class Solution {
public:
    int islandPerimeter(vector<vector<int>>& grid) {
        int ans=0;
```

```
        for(int i=0;i<grid.size();i++){
            for(int j=0;j<grid[0].size();j++){
                if(grid[i][j] == 1){
                    int count=0;
                    if(i>0 && grid[i-1][j] == 1) count++;
                    if(j>0 && grid[i][j-1] == 1) count++;
                    if(i<grid.size()-1 && grid[i+1][j] == 1) count++;
                    if(j<grid[0].size()-1 && grid[i][j+1] == 1) count++;

                    ans=ans+(4-count);
                }
            }
        }

        return ans;
    }
};
```

Top
left
bottom
right

T(n) : O(m x n)

S(n) : O(1)