2658. Maximum Number of Fish in a Grid

My Submissions (/contest/biweekly-contest-103/problems/maximum-number-of-fish-in-a-grid/submissions/) Back to Contest (/contest/biweekly-contest-103/)

User Accepted:	6063
User Tried:	6818
Total Accepted:	6310
Total Submissions:	11635
Difficulty:	Medium

You are given a **0-indexed** 2D matrix `grid` of size `m x n`, where `(r, c)` represents:

- A **land** cell if `grid[r][c] = 0`, or
- A **water** cell containing `grid[r][c]` fish, if `grid[r][c] > 0`.

A fisher can start at any **water** cell `(r, c)` and can do the following operations any number of times:

- Catch all the fish at cell `(r, c)`, or
- Move to any adjacent **water** cell.

Return *the **maximum** number of fish the fisher can catch if he chooses his starting cell optimally, or* `0` if no water cell exists.

An **adjacent** cell of the cell `(r, c)`, is one of the cells `(r, c + 1)`, `(r, c - 1)`, `(r + 1, c)` or `(r - 1, c)` if it exists.

Example 1:

0	2	1	0
4	0	0	3
1	0	0	4
0	3	2	0

```
Input: grid = [[0,2,1,0],[4,0,0,3],[1,0,0,4],[0,3,2,0]]
Output: 7
Explanation: The fisher can start at cell (1,3) and collect 3 fish, then move to cell (2,3) and collect 4 fish.
```

Example 2:

1	0	0	0
0	0	0	0
0	0	0	0
0	0	0	1

```
Input: grid = [[1,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,1]]
Output: 1
Explanation: The fisher can start at cells (0,0) or (3,3) and collect a single fish.
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 10`
- `0 <= grid[i][j] <= 10`

Discuss (https://leetcode.com/problems/maximum-number-of-fish-in-a-grid/discuss)

JavaScript

```
const deepCopy2DArray = (g) => { let d = []; for (const a of g) d.push([...a]); return d; };

const findMaxFish = (g) => {
    let areas = getAllAreasCoordinates(deepCopy2DArray(g)), res = 0;
    for (const area of areas) {
        let sum = 0;
        for (const [x, y] of area) sum += g[x][y];
        res = Math.max(res, sum);
    }
    return res;
};

const dx = [1, -1, 0, 0], dy = [0, 0, 1, -1];
const getAllAreasCoordinates = (g) => {
    const forbid = 0, floodFillMakeConnected = '*';
    let n = g.length, m = g[0].length, res = [];
    for (let i = 0; i < n; i++) {
        for (let j = 0; j < m; j++) {
            if (g[i][j] != forbid) {
                let q = [[i, j]], area = [];
                while (q.length) {
                    let [x, y] = q.shift();
                    for (let k = 0; k < 4; k++) {
                        let nx = x + dx[k], ny = y + dy[k];
                        if (nx < 0 || nx >= n || ny < 0 || ny >= m || g[nx][ny] == forbid || g[nx][ny] ==
floodFillMakeConnected) continue;
                        g[nx][ny] = floodFillMakeConnected;
                        area.push([nx, ny])
                        q.push([nx, ny]);
                    }
                }
                if (area.length == 0 && g[i][j] != floodFillMakeConnected) area.push([i, j]);
                res.push(area);
            }
        }
    }
    return res;
};
```

☐ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/948616311/) More Details (/submissions/detail/948616311/)

Share your acceptance!

3