1267. Count Servers that Communicate

My Submissions (/contest/weekly-contest-164/problems/count-servers-that-communicate/submissions/)

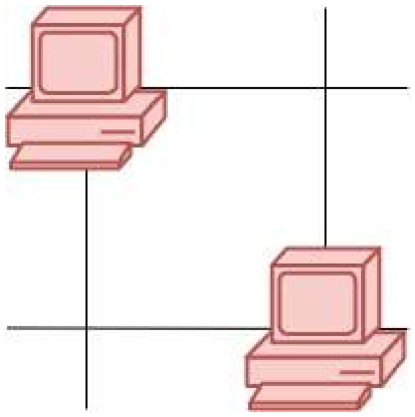
Back to Contest (/contest/weekly-contest-164/)

User Accepted:	2337
User Tried:	2642
Total Accepted:	2378
Total Submissions:	4289
Difficulty:	Medium

You are given a map of a server center, represented as a `m * n` integer matrix `grid`, where 1 means that on that cell there is a server and 0 means that it is no server. Two servers are said to communicate if they are on the same row or on the same column.

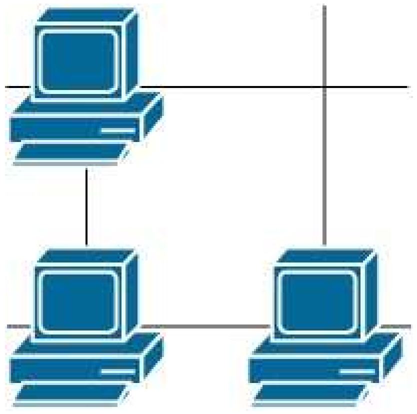
Return the number of servers that communicate with any other server.

Example 1:



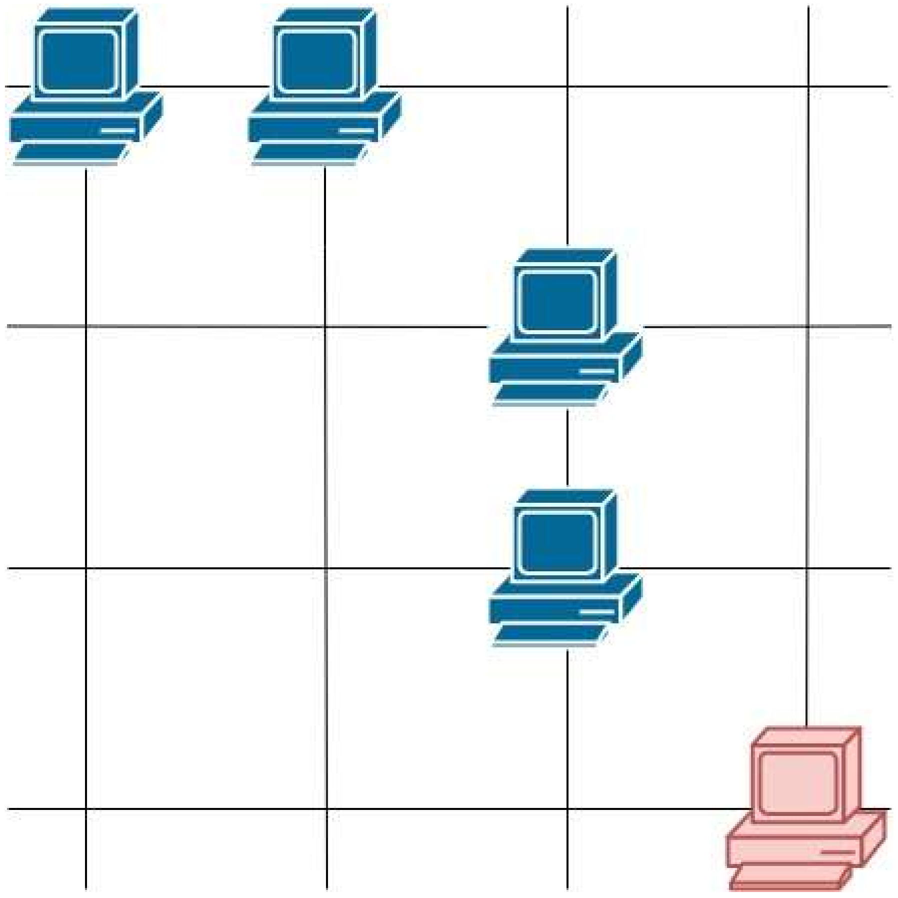
```
Input: grid = [[1,0],[0,1]]
Output: 0
Explanation: No servers can communicate with others.
```

Example 2:



```
Input: grid = [[1,0],[1,1]]
Output: 3
Explanation: All three servers can communicate with at least one other server.
```

Example 3:



```
Input: grid = [[1,1,0,0],[0,0,1,0],[0,0,1,0],[0,0,0,1]]
Output: 4
Explanation: The two servers in the first row can communicate with each other. The two servers
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m <= 250`
- `1 <= n <= 250`
- `grid[i][j] == 0 or 1`

Discuss (https://leetcode.com/problems/count-servers-that-communicate/discuss)

Java

```java
class Solution {
    public int countServers(int[][] grid) {
        int count = 0;
        if (grid == null || grid.length == 0 || grid[0].length == 0) {
            return count;
        }
        // Find the first point and insert into queue
        Queue<int[]> queue = new LinkedList<int[]>();
        outloop: for (int i = 0; i < grid.length; i++) {
            for (int j = 0; j < grid[0].length; j++) {
                if (grid[i][j] == 1 && isCommunicate(i, j, grid)) {
                    queue.offer(new int[] { i, j });
                    break outloop;
                }
            }
        }
        while (!queue.isEmpty()) {
            int[] head = queue.poll();
            int row = head[0];
            int col = head[1];
            grid[row][col] = 0;
            count++;
            for (int j = 0; j < grid[0].length; j++) {
                if (j != col && grid[row][j] == 1) {
                    queue.offer(new int[] { row, j });
                }
            }
            for (int i = 0; i < grid.length; i++) {
                if (i != row && grid[i][col] == 1) {
                    queue.offer(new int[] { i, col });
                }
            }
        }
        return count;
    }

    private boolean isCommunicate(int row, int col, int[][] grid) {
        boolean isServer = false;
        for (int j = 0; j < grid[0].length; j++) {
            if (j != col && grid[row][j] == 1) {
                isServer = true;
                break;
            }
        }
        for (int i = 0; i < grid.length; i++) {
            if (i != row && grid[i][col] == 1) {
                isServer = true;
                break;
            }
        }
        return isServer;
    }
}
```

Custom Testcase

Run Submit

Submission Result: Wrong Answer (/submissions/detail/281345676/)

More Details (/submissions/detail/281345676/) >_

```
Input:      [[1,1,0,0],[0,0,1,0],[0,0,1,0],[0,0,0,1]]

Output:     2
```

```
Expected:    4
```