1267. Count Server That Coummunicate.

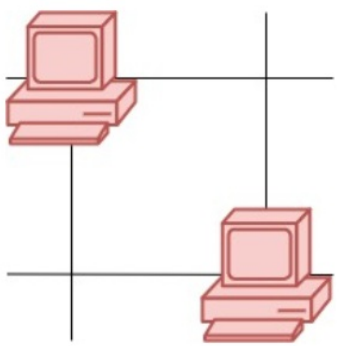
1267. Count Servers that Communicate

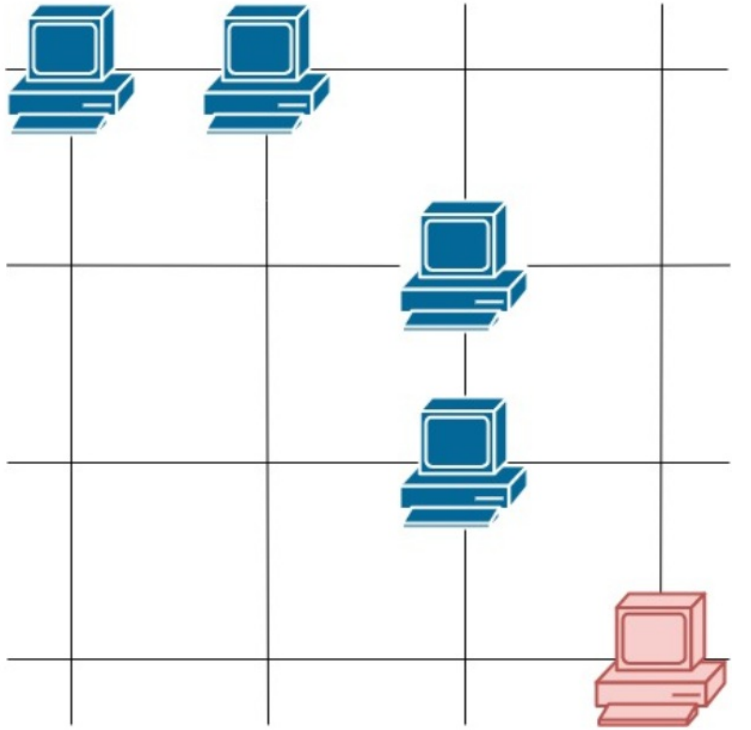
Medium 142 13 Add to List Share

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.

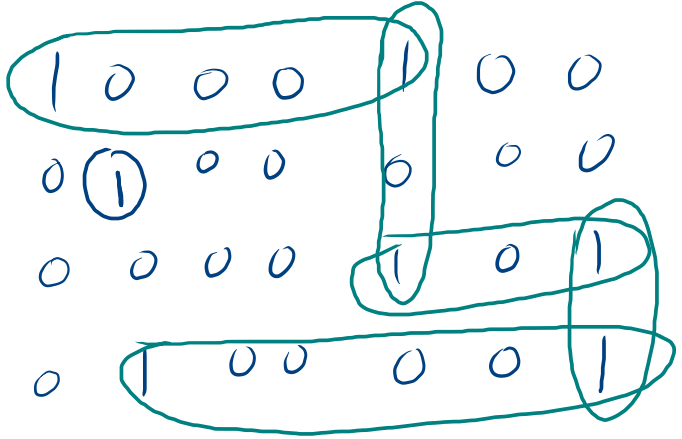
Example 1:





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

Ex-



1. all server can communicate except 1
2. simple apply flood fill algo of radius method.

→

dir: +, and $1 \le rad \le max(n, m)$

```
int dir[4][2] = {{1, 0}, {-1, 0}, {0, -1}, {0, 1}};   // four dir

int dfs(int x, int y, int& len,vector<vector<int>> &grid)
{
    grid[x][y] = 2;
    int count = 0;
    for (int i = 0; i < 4; i++)
    {
        for(int rad=1;rad<=len;rad++){
        int u = x + rad*dir[i][0];
        int v = y + rad*dir[i][1];

        if(u>=0 && v>=0 && u<grid.size()&& v<grid[0].size()){
            if(grid[u][v]==1)
            count+=dfs(u,v,len,grid);
        }else break;                    // to prevent unnecessary call
    }
    }

    return count+1;
}
```

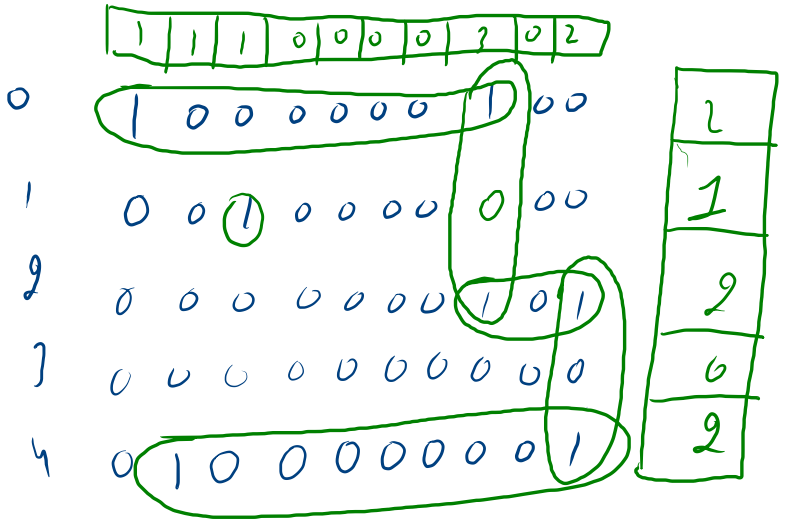
radius method.

```
int countServers(vector<vector<int>> &grid)
{
    std::ios_base::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int ans = 0;
    int len=max(grid.size(),grid[0].size());
    for (int u = 0; u < grid.size(); u++)
        for (int v = 0; v < grid[0].size(); v++)
            if (grid[u][v]==1)
            {
                int count = dfs(u, v,len, grid);
                ans += (count == 1 ? 0 : count);   // if only one server then no communication can happen.
            }

    return ans;
}
```

Fastio input output.

method2 :



for (1,2) → row[1] * col[2] * grid[1][2] == 1
server count--;

// For each server, the row or the column must have another server except the current one.
// We can simply keep a count of servers in each row and column and use this information to get the result while traversing the grid.

```
int method02(vector<vector<int>> &grid)
{
    std::ios_base::sync_with_stdio(false);
    std::cin.tie(nullptr);

    int n = grid.size();
    int m = grid[0].size();
    vector<int> row(n, 0);
    vector<int> col(m, 0);

    int total_server = 0;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            row[i] += grid[i][j];
            col[j] += grid[i][j];
            total_server+=grid[i][j];
        }
    }

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            if (grid[i][j] == 1 && row[i] * col[j] * grid[i][j] == 1)
                total_server--;
        }
    }

    return total_server;
}
```

1. We simply take count of server in each row and column
2. total possible server is += grid[i][j].

If there is only one server in row and there respective column then server count should be miun 1.