1568. Minimum Number of Days to Disconnect Island

You are given an `m x n` binary grid `grid` where `1` represents land and `0` represents water.
An **island** is a maximal **4-directionally** (horizontal or vertical) connected group of `1`'s.

The grid is said to be **connected** if we have **exactly one island**, otherwise is said **disconnected**.

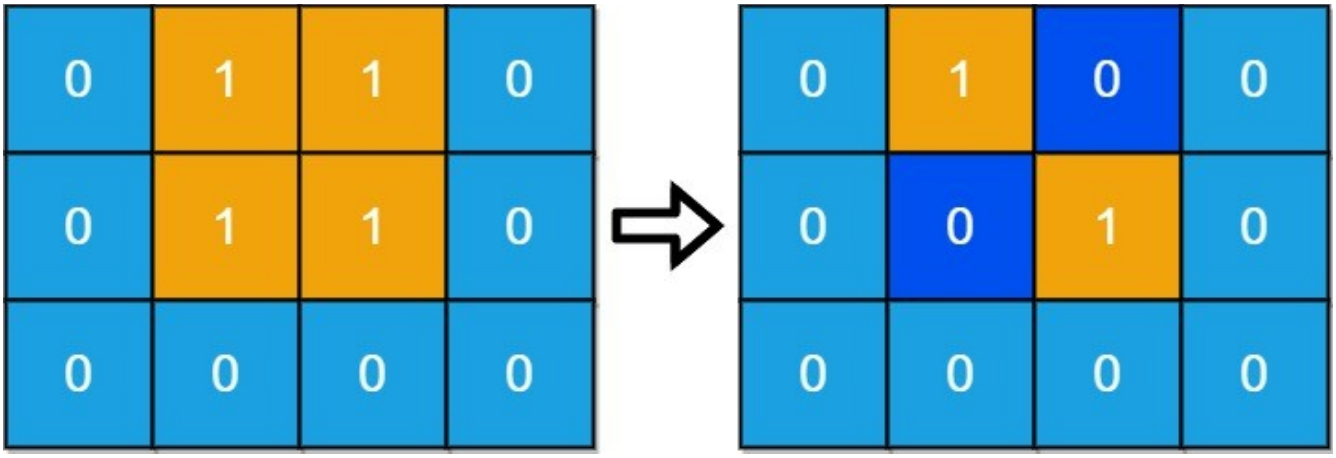
In one day, we are allowed to change **any** single land cell `(1)` into a water cell `(0)`.

Return *the minimum number of days to disconnect the grid*.

Example 1:

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

Output: 2
Explanation: We need at least 2 days to get a disconnected grid.
Change land grid[1][1] and grid[0][2] to water and get 2 disconnected island.
```



Example 2:

```
Input: grid = [[1,1]]
Output: 2
Explanation: Grid of full water is also disconnected ([[1,1]] -> [[0,0]]), 0
islands.
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 30`
- `grid[i][j]` is either `0` or `1`.

1568. Minimum Number of Days to Disconnect Island

```
/*
   Brute force approach
   Time complexity: O( (nm)^2)
   Space complexity: O((nm)^2)
*/
class Solution {
   public:
      int n;
      int m;
      pair<int,int> moves[4] = {
         {-1,0},
         {1,0},
         {0,-1},
         {0,1}
      };
   public:
       bool is_in_grid(int x,int y){
         return x>=0 && x<=m-1 && y>=0 && y<=n-1;
      }

      void DFS(vector<vector<int>>& grid,std::vector<vector<bool>>& visited,int x,int y){
         if(visited[x][y]) return;
         visited[x][y]=true;
         for(int i=0;i<4;++i){
            int move_x=x+moves[i].first;
            int move_y=y+moves[i].second;
            if(!is_in_grid(move_x,move_y)) continue;
            if(grid[move_x][move_y]==0) continue;
            DFS(grid,visited,move_x,move_y);
         }
      }
```

```
int count_islands(vector<vector<int>>& grid) {
        std::vector<vector<bool>> visited(m,std::vector<bool>(n,false));
        int cnt=0;
        for(int i=0;i<m;++i){
            for(int j=0;j<n;++j){
                if(grid[i][j]==0 || visited[i][j]) continue;
                cnt++;
                DFS(grid,visited,i,j);
            }
        }
        return cnt;
    }

    int minDays(vector<vector<int>>& grid) {
        m=grid.size();
        n=grid[0].size();

        int nb_inslands=count_islands(grid);
        if(nb_inslands!=1) return 0;


        for(int i=0;i<m;++i){
            for(int j=0;j<n;++j){
                if(grid[i][j]==1) {
                    grid[i][j]=0;
                    nb_inslands=count_islands(grid);
                    if(nb_inslands!=1) return 1;
                    grid[i][j]=1;
                }
            }
        }
        return 2;
    }
};
```