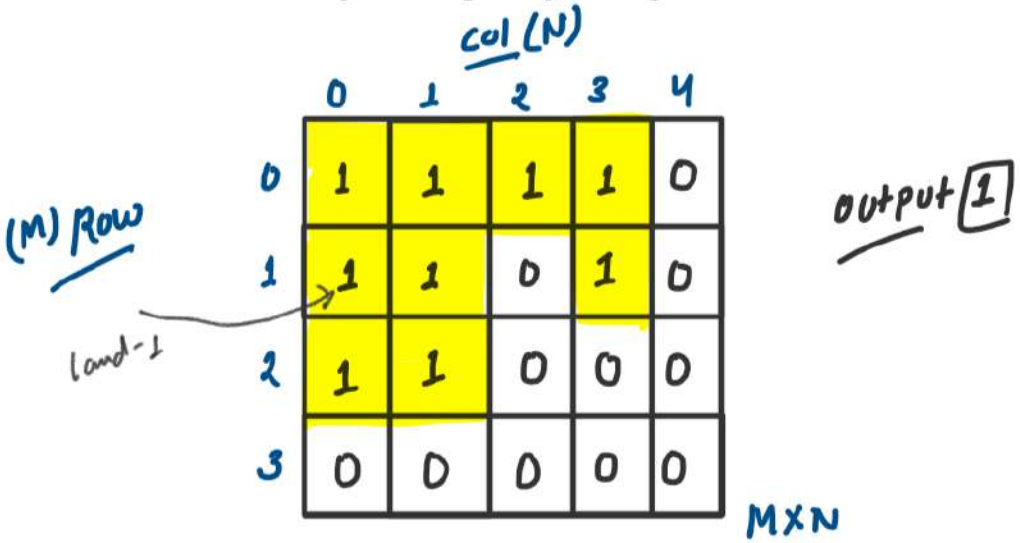
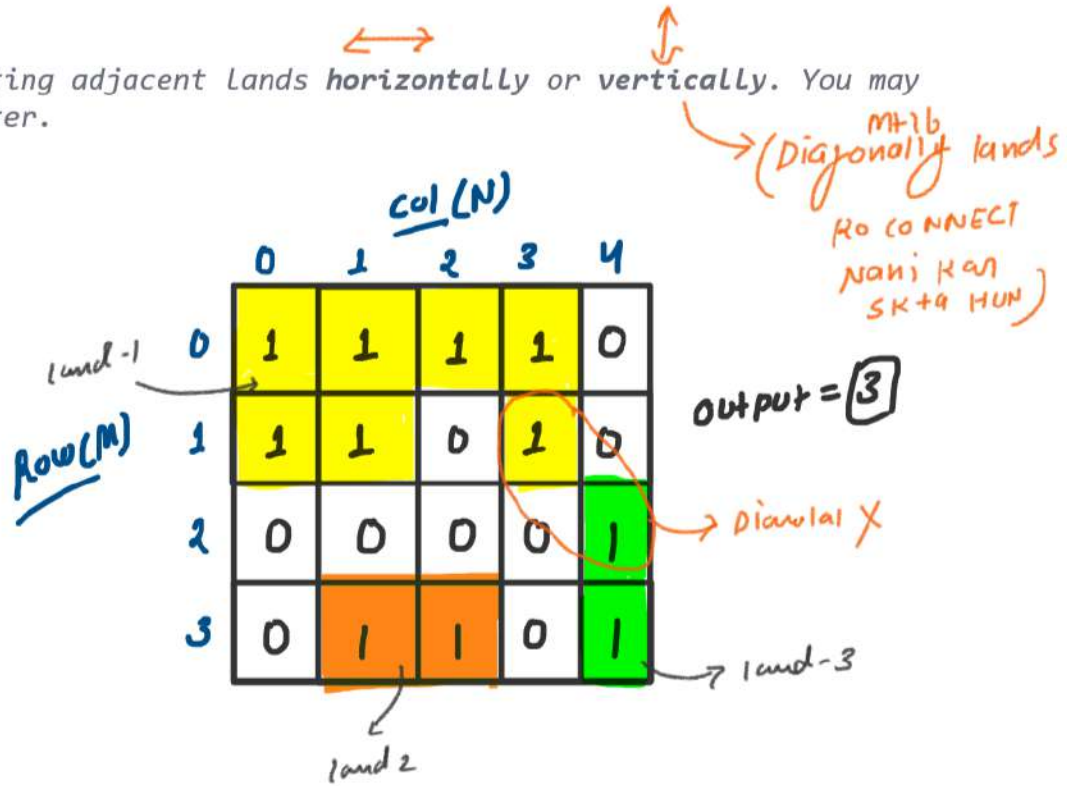
1. Number of Islands (Leetcode-200)

Problem Statement:
*Given an `m x n` 2D binary **grid** which represents a map of **'1's (land)** and **'0's (water)**, return the number of **islands**.*

Important Line:
*An `island` is surrounded by water and is formed by connecting adjacent lands **horizontally** or **vertically**. You may assume all four edges of the grid are all surrounded by water.*





Observation

0 → water
1 → land
[GRID] Row x col

[1] ↔ [1] Horizontal
and
[1]
↕
[1] vertical

Hints

☺

- Dissconnected Graph mein jitne no. of components hote hai utne hi number of lands mane jayingi

↳ MEANS DFS Algorithm ka use kar sakte hai to explore all possible ways

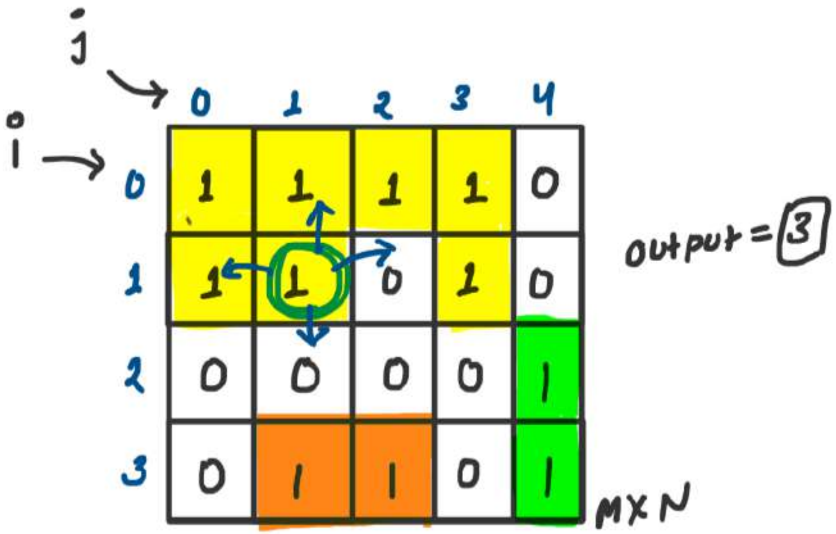
	0	1	2	3	4
0	1	1	1	1	0
1	1	1	0	1	0
2	0	0	0	0	1
3	0	1	1	0	1

output = [3]

M x N

let ☺ I am here Grid[1][1]

↳ I have four side to move but kab move kar skta hun yeh me Base cases se maloom karunga.



Base case (1) visited cell ke liye me 'X' fill kar loonga jisse me wapas dobara move na kar paun

grid[i][j] == 'X'

→ return 0

Base cases (2)

grid[i][j] == '0'

→ return 0

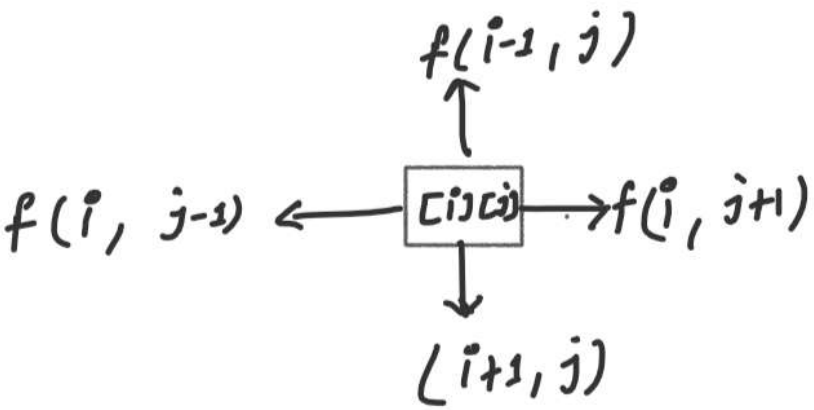
Base cases (3)(4)(5)(6)

i<0 || j<0 || i>=M || j>=N

→ return 0

We have 4 move from each cell of grid jaha par hum stand kar rhe hai

1. TopMove
2. Bottom Move
3. Rijmt Move
4. Left Move



```cpp
// 1. Number of Islands (Leetcode-200)

class Solution {
public:
    void dfs(vector<vector<char>>& grid, int i, int j){
        int m = grid.size();
        int n = grid[0].size();
        // Base case
        if(i < 0 || j < 0 || i >= m ||  j >= n || grid[i][j] =='0' || grid[i][j] == 'x'){
            return;
        }

        // 1 case hum solve kar lege:
        // x represents the current cell of grid is visited now
        grid[i][j] = 'x';

        // We have four move from each cell janaha par hum khade hue hai
        // TopMove
        dfs(grid, i-1, j);
        // BottomMove
        dfs(grid, i+1, j);
        // RightMove
        dfs(grid, i, j+1);
        // LeftMove
        dfs(grid, i, j-1);
    }

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

        for(int i=0; i<m; i++){
            for(int j=0; j<n; j++){
                if(grid[i][j] !='0' && grid[i][j] != 'x'){
                    dfs(grid, i, j);
                    ans++;
                }
            }
        }
        return ans;
    }
};
```