1. Graph Creation Time Complexity

Graph Creation Time complexity depends on two terminologies:

*E-> Number of **edges***
*V-> Number of **nodes/vertices***

→ T.C. = O(V+E) Why this time complexity?

Step1: *initialize the nodes list of size is available of nodes*

→ Note: Nodes are store uniquely in the map Always

So T.C. ⇒ O(V)

MAP

KEY	List
0	{1,2}
1	{0,3}
2	{1,0}
3	{2,7}

Nodes → KEY

AdjList → List

Total Node (V) = 4
Total Edges = 8

Step2: *store edges corresponding to each unique key(node) in the list*

V	{1,2}	{0,1}	{1,0}	{2,7}
	0	1	2	3

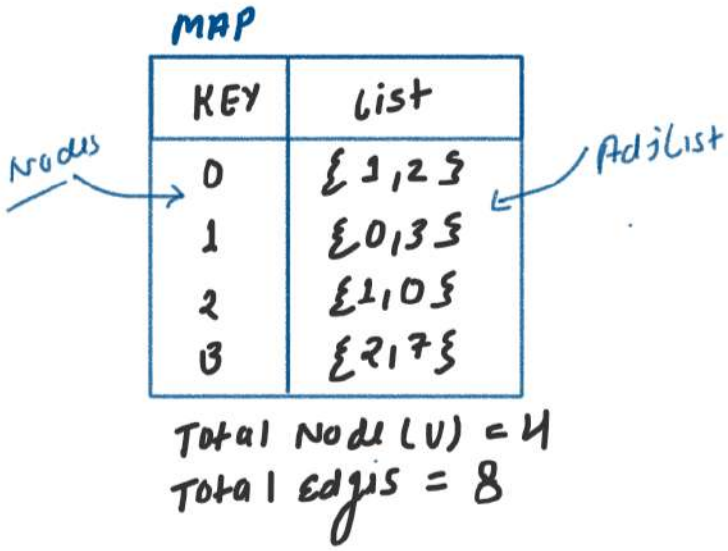
E = 8

Iska Mtlb: we have to call the addEdges function total number of Edges time.

So $T.C. = O(E)$

OVERAll $T.C. \Rightarrow O(V+E)$

2. Graph Creation Space Complexity



Space compluxity = O(V+E)

↳ why ?

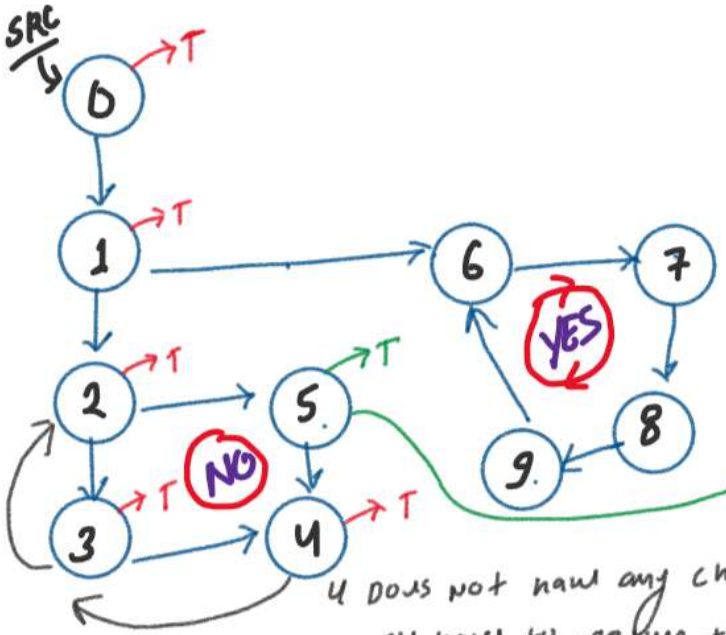
↳ total Boxes in map which are used to store the each Node(V) and Edges(E)

so Space Como = O(V+E)

3. Detect Cycle by DFS in Directed Graph

Question: why use of DFSTrack map ?

DRY RUN with visited map only



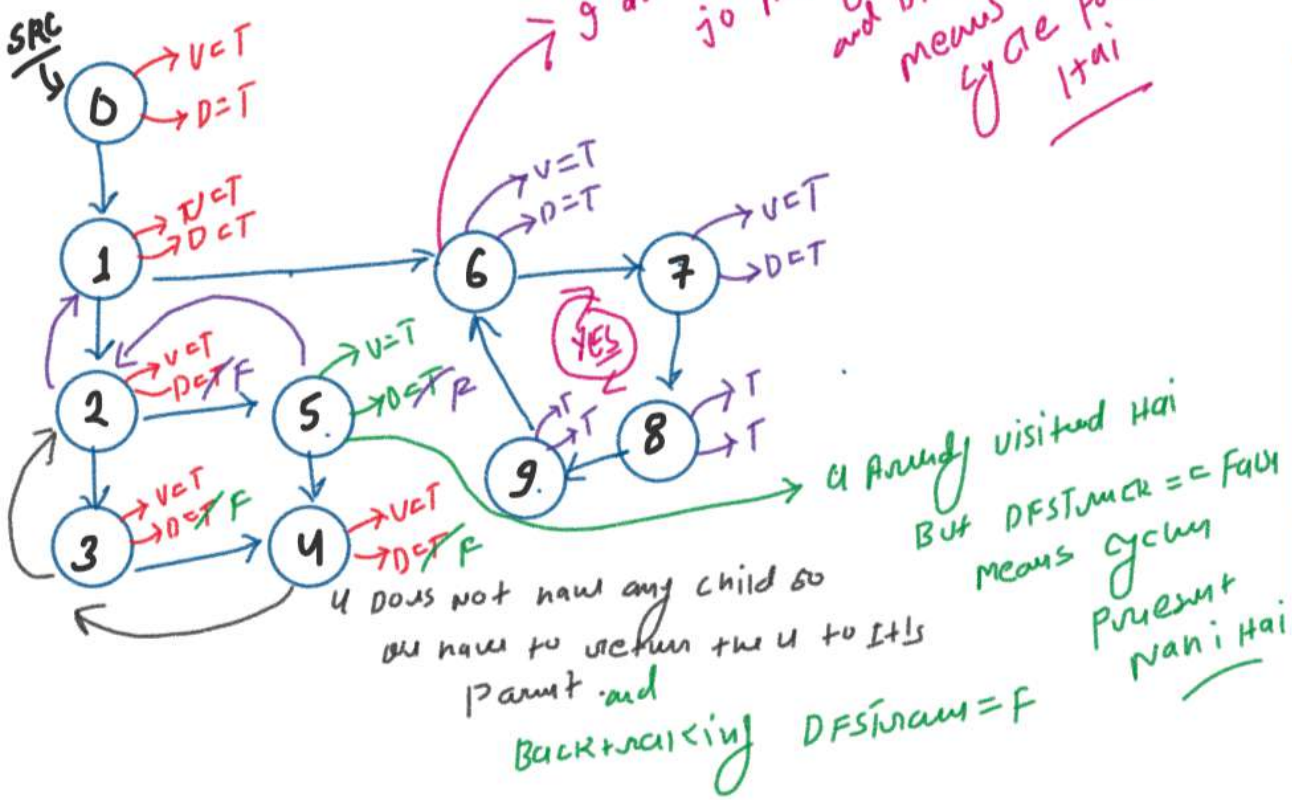
5 ka child 4 hai jo already visited hai Matlb Cycle present Ho jo ki Actual me yaha par cycle Print Hai Nahi

4 does not have any child so we have to return the 4 to its parent.

visited

Key	value
0	~~F~~ T
1	~~F~~ T
2	~~F~~ T
3	~~F~~ T
4	~~F~~ T
5	~~F~~ T
6	F
7	F

DRY RUN with visited map & DFSTrack



visited

Key	value
0	~~F~~ T
1	~~F~~ T
2	~~F~~ T
3	~~F~~ T
4	~~F~~ T
5	~~F~~ T
6	~~F~~ T
7	~~F~~ T
8	~~F~~ T
9	~~F~~ T

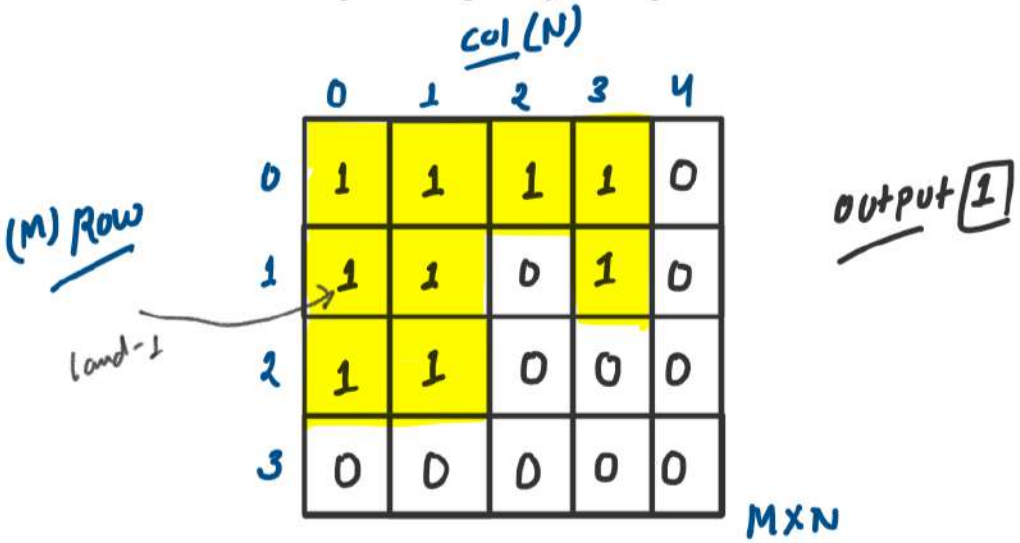
DFSTrack

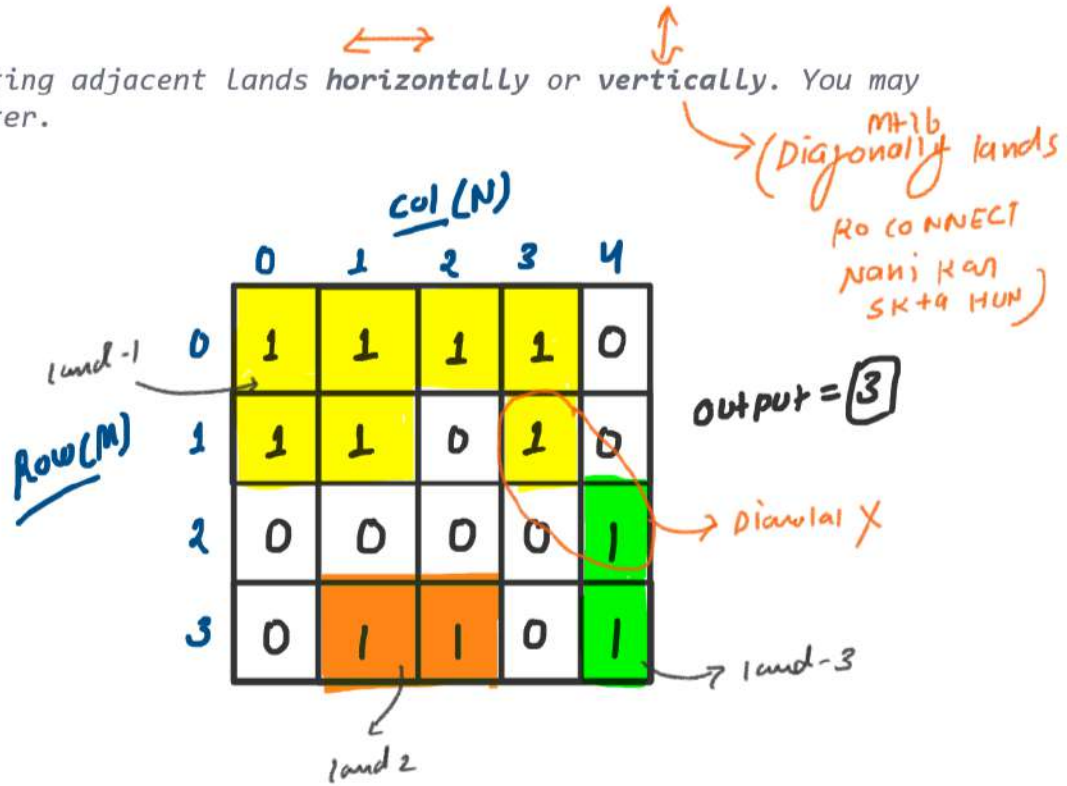
Key	value
0	~~F~~ T
1	~~F~~ T
2	~~F~~ ~~T~~ F
3	~~F~~ ~~T~~ F
4	~~F~~ ~~T~~ F
5	~~F~~ ~~T~~ F
6	~~F~~ T
7	~~F~~ T
8	~~F~~ T
9	~~F~~ T

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.*





Obsenbation

0 → water
1 → land
[GRID] Row x col

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

Hints

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

↳ MENS DFS Algoritm ka use kar sakte hai to explore all possible ways

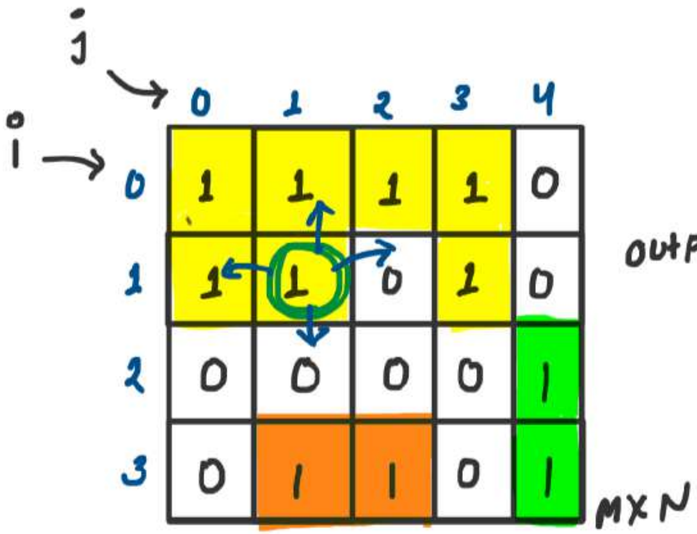
	0	1	2	3	4
0	1	1	1	1	0
1	1	1	0	1	0
2	0	0	0	0	1
3	0	1	1	0	1

M x N

output = [3]

Let I am Hurl 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 wapis dobara move na kar paun

grid[i][j] == 'x'

→ return 0

Base cases (2)

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

→ return 0

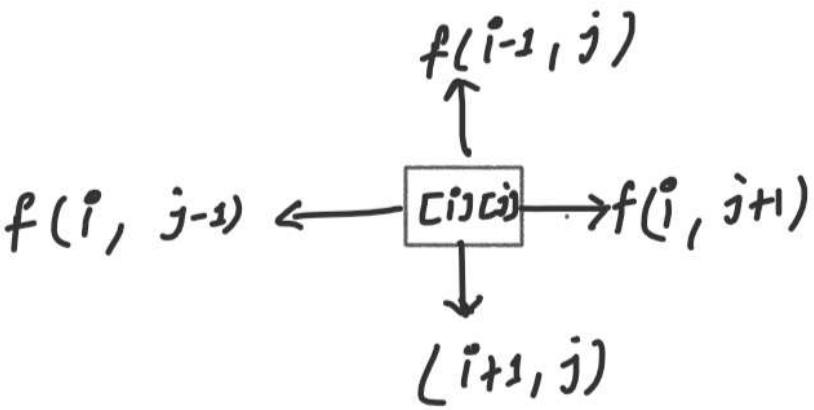
Base cases (3)(4)(5)(6)

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;
    }
};
```