# Number of Distinct Islands

Given a boolean 2D matrix grid of size n * m. You have to find the number of distinct islands where a group of connected 1s (horizontally or vertically) forms an island. Two islands are considered to be distinct if and only if one island is not equal to another (not rotated or reflected).

Example 1:

Input:
grid[][] = {{1, 1, 0, 0, 0},
{1, 1, 0, 0, 0},
{0, 0, 0, 1, 1},
{0, 0, 0, 1, 1}}
Output:
1
Explanation:
grid[][] = {{1, 1, 0, 0, 0},
{1, 1, 0, 0, 0},
{0, 0, 0, 1, 1},
{0, 0, 0, 1, 1}}
Same colored islands are equal.
We have 2 equal islands, so we
have only 1 distinct island.

Example 2:

Input:
grid[][] = {{1, 1, 0, 1, 1},
{1, 0, 0, 0, 0},
{0, 0, 0, 0, 1},
{1, 1, 0, 1, 1}}
Output:
3
Explanation:
grid[][] = {{1, 1, 0, 1, 1},
{1, 0, 0, 0, 0},
{0, 0, 0, 0, 1},
{1, 1, 0, 1, 1}}
grid[i][j] == 0 or grid[i][j] == 1

Same colored islands are equal.
We have 4 islands, but 2 of them
are equal, So we have 3 distinct islands.

Your Task:

You don't need to read or print anything. Your task is to complete the function countDistinctIslands() which takes the grid as an input parameter and returns the total number of distinct islands.

Expected Time Complexity: O(n * m)
Expected Space Complexity: O(n * m)

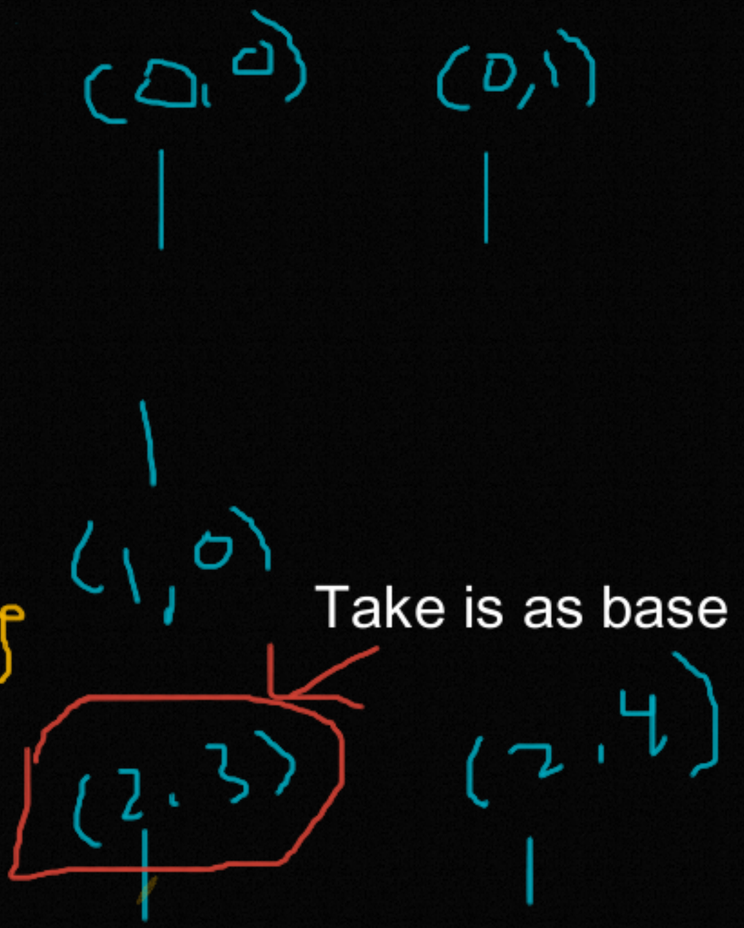
Constraints:
1 ≤ n, m ≤ 500

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 1 |
| 3 | 1 | 1 | 0 | 1 | 0 |

How to check this two island are equal

We store them in set

{(0,0),(0,1),(1,0)}




(2,3) - Base = (0,0)

(2,4) - Base = (0,1)

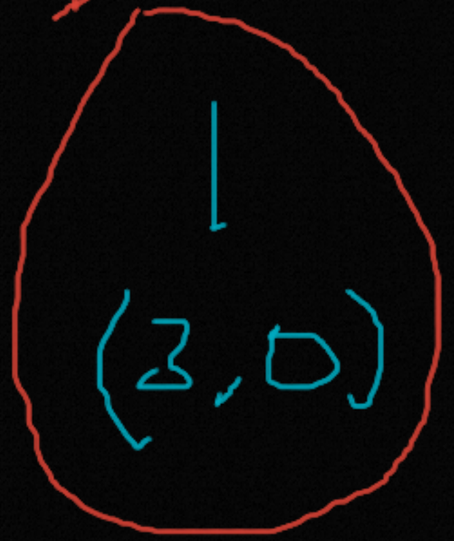
(3,3) - Base = (1,0)


(0,-4)

(0 -3) - (0-3) = (0, 0)

(0 -4) - (0 -3) = (0, 1)

BASE




(3,1)

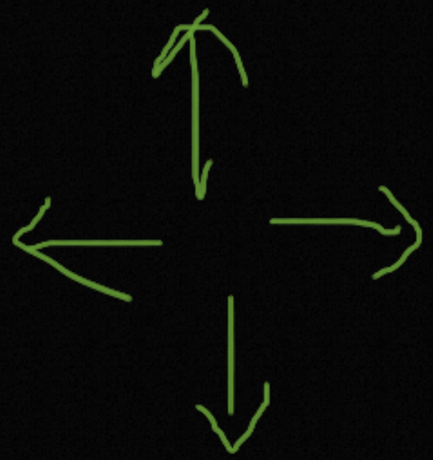
(3,0) - (3,0) = (0,0)

(3-1) - (3,0) = (0,1)

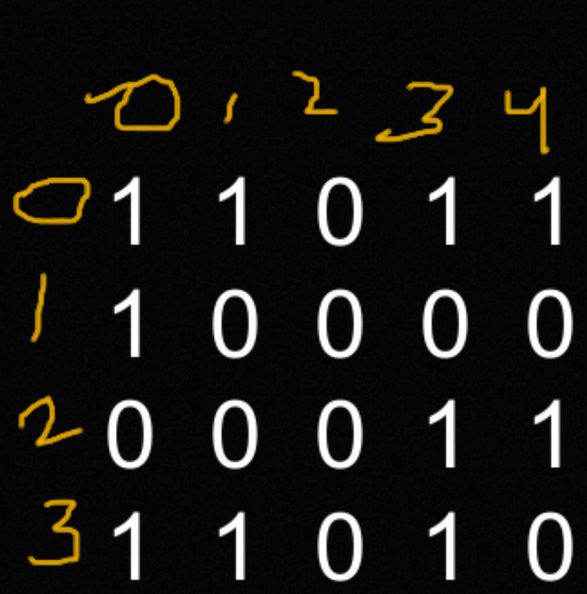
Follow the same order thoughtout

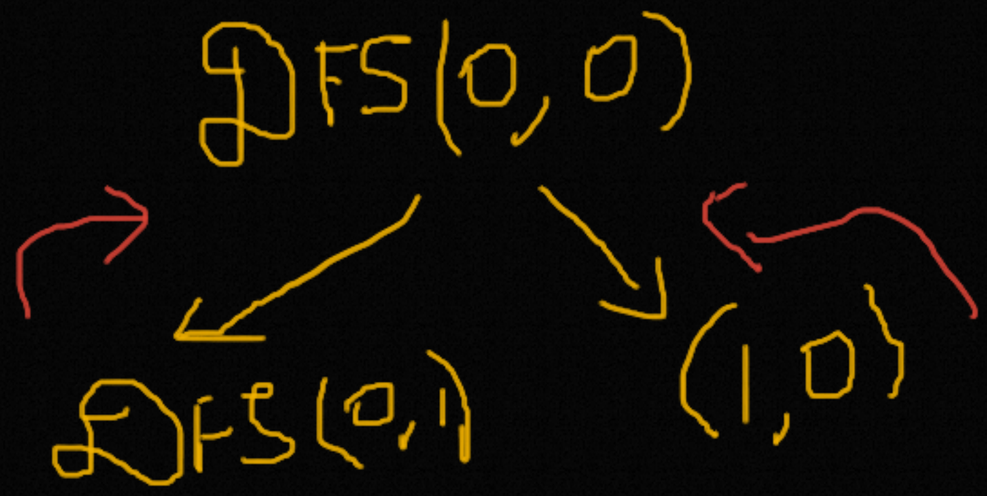
One of the  order

Right -> down -> left -> up

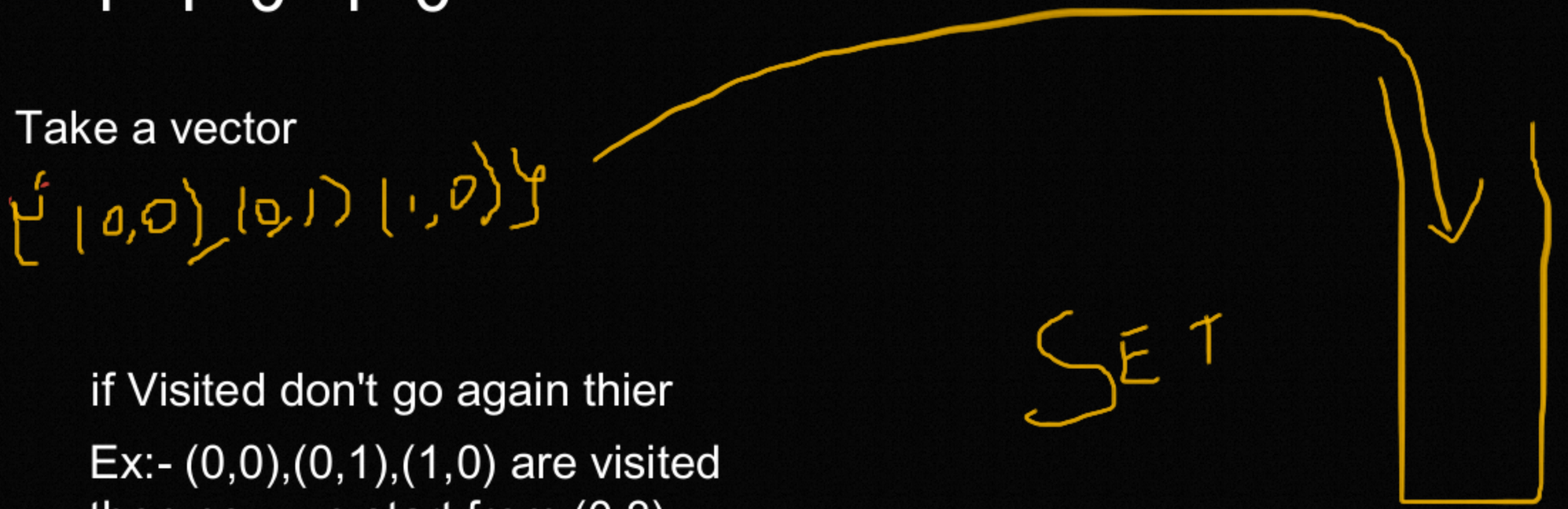


If right 1 go right
else go down
similarly for other



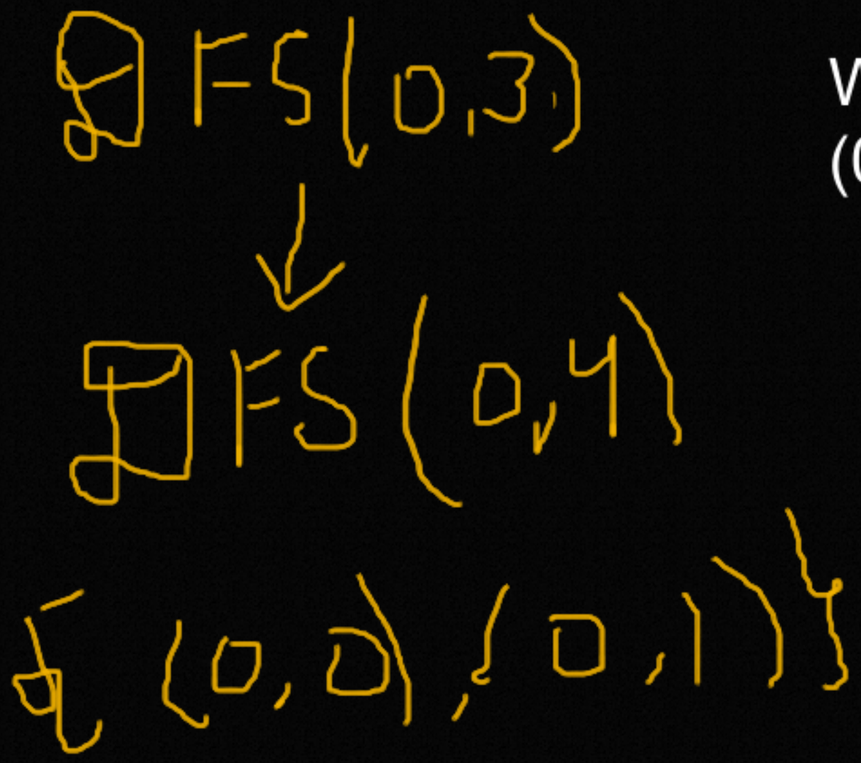




Take a vector




if Visited don't go again thier

Ex:- (0,0),(0,1),(1,0) are visited
then now we start from (0,3)




We start from (0,3) because (0,2) is 0 and (0,1) is visited

By substracting base

Similary do for Other