**Number Of Enclaves: -**

**Medium** Accuracy: **50.93%** Submissions: **33K+** Points: **4**

You are given an **n x m** binary matrix **grid**, where **0** represents a sea cell and **1** represents a land cell.

A move consists of walking from one land cell to another adjacent (4-directionally) land cell or walking off the boundary of the grid.

Find the number of land cells in **grid** for which we cannot walk off the boundary of the grid in any number of moves.

**Example 1:**

**Input:**

grid[][] = {{0, 0, 0, 0},

{1, 0, 1, 0},

{0, 1, 1, 0},

{0, 0, 0, 0}}

**Output:**

3

**Explanation:**

0 0 0 0

1 0 **1** 0

0 **1** **1** 0

0 0 0 0

The highlighted cells represents the land cells.

**Example 2:**

**Input:**

grid[][] = {{0, 0, 0, 1},

{0, 1, 1, 0},

{0, 1, 1, 0},

{0, 0, 0, 1},

{0, 1, 1, 0}}

**Output:**

4

**Explanation:**

0 0 0 1

0 **1** **1** 0

0 **1** **1** 0

0 0 0 1

0 1 1 0

The highlighted cells represents the land cells.

**Your Task:**

You don't need to print or input anything. Complete the function **numberOfEnclaves()**which takes a 2D integer matrix **grid**as the input parameter and returns an integer, denoting the number of land cells.

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

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

**Constraints:**

* 1 <= n, m <= 500
* grid[i][j] == 0 or 1

**Code: -**

//{ Driver Code Starts

// Initial Template for C++

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

// User function Template for C++

int dx[] = {0, -1, 0, 1};

int dy[] = {-1, 0, 1, 0};

class Solution {

public:

void mark(vector<vector<int>> &grid, int row, int col, vector<vector<bool>> &vis){

int n = grid.size(), m = grid[0].size();

// base case

// recursive case

// traversing all 4 directions

for(int i=0; i<4; ++i){

int ni = row + dx[i];

int nj = col + dy[i];

if(0<=ni and ni<n and 0<=nj and nj<m and grid[ni][nj]==1 and !vis[ni][nj]){

vis[ni][nj] = true;

mark(grid, ni, nj, vis);

}

}

return;

}

int numberOfEnclaves(vector<vector<int>> &grid) {

// Code here

int n = grid.size(), m = grid[0].size();

vector<vector<bool>> vis(n, vector<bool>(m, false));

for(int i=0; i<n; ++i){

for(int j=0; j<m; ++j){

if((i==0 or i==n-1) and grid[i][j]==1 and !vis[i][j]){

vis[i][j] = true;

mark(grid, i, j, vis);

}

else if((j==0 or j==m-1) and grid[i][j]==1 and !vis[i][j]){

vis[i][j] = true;

mark(grid, i, j, vis);

}

}

}

int ans = 0;

for(int i = 0; i<n; ++i){

for(int j=0; j<m; ++j){

if(grid[i][j]==1 and !vis[i][j])

++ans;

}

}

return ans;

}

};

//{ Driver Code Starts.

int main() {

int t;

cin >> t;

while (t--) {

int n, m;

cin >> n >> m;

vector<vector<int>> grid(n, vector<int>(m));

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

cin >> grid[i][j];

}

}

Solution obj;

cout << obj.numberOfEnclaves(grid) << endl;

}

}

// } Driver Code Ends

**T.C: - O(N \* M)**

**S.C: - O(N \* M)**