**Surround the 1's :-**

Easy Accuracy: 67.16% Submissions: 7K+ Points: 2

Given a matrix of order **n**x**m**, composed of only 0's and 1's, find the number of 1's in the matrix that are surrounded by an **even number (>0) of 0's**. The surrounding of a cell in the matrix is defined as the **elements above**, **below**, on **left**, on **right** as well as the**4 diagonal elements** around the cell of the matrix. Hence, the surrounding of any matrix elements is composed of 8 elements. Find the number of such 1's.

**Example 1:**

**Input:**matrix = {{1, 0, 0},   
 {1, 1, 0},

{0, 1, 0}}

**Output:**1

**Explanation:**1 that occurs in the 1st row and 1st column, has 3 surrounding elements 0,1 and 1. The occurrence of zero is odd.   
1 that occurs in 2nd row and 1st column has 5 surrounding elements 1,0,1,1 and 0. The occurrence of zero is even.   
1 that occurs in 2nd row and 2nd column has 8 surrounding elements. The occurrence of 0 is odd.   
Similarly, for the 1 that occurs in 3rd row and 2nd column, the occurrence of zero in it's 5 surrounding elements is odd.

Hence, the output is 1.

**Example 2:**

**Input:**matrix = {{1}}

**Output:**0

**Explanation:**There is only 1 element in the matrix. Hence, it has no surroundings, so it's count for even 0's is 0 for the whole matrix.   
0 is even but we want occurrence of a zero in the surrounding at least once.

Hence, output is 0.

**Your Task:**  
You don't need to read or print anything. Your task is to complete the function **Count()** which takes the matrix as input parameter and returns the number of 1's which are surrounded by even number of 0's.

**Expected Time Complexity:**O(n \* m)  
**Expected Space Complexity:**O(1)

**Constraints:**  
1 <= n, m <= 103

**Code :-**

//{ Driver Code Starts

#include<bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution {

public:

int Count(vector<vector<int> >& matrix) {

int row=matrix.size(), col=matrix[0].size(), ans=0;

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

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

// matrix cell value = 1

if(matrix[i][j]==1){

int count=0;

// up

if(i>0 and matrix[i-1][j]==0)

++count;

// down

if(i<row-1 and matrix[i+1][j]==0)

++count;

// left

if(j>0 and matrix[i][j-1]==0)

++count;

// right

if(j<col-1 and matrix[i][j+1]==0)

++count;

// left-up diagonal

if(i>0 and j>0 and matrix[i-1][j-1]==0)

++count;

// right-up diagonal

if(i>0 and j<col-1 and matrix[i-1][j+1]==0)

++count;

// left-down diagonal

if(i<row-1 and j>0 and matrix[i+1][j-1]==0)

++count;

// right-down diagonal

if(i<row-1 and j<col-1 and matrix[i+1][j+1]==0)

++count;

// final check for answer

if(count != 0 and count % 2 == 0)

++ans;

}

}

}

return ans;

}

};

//{ Driver Code Starts.

int main(){

int tc;

cin >> tc;

while(tc--){

int n, m;

cin >> n >> m;

vector<vector<int>> matrix(n, vector<int>(m,0));

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

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

cin >> matrix[i][j];

}

}

Solution ob;

int ans = ob.Count(matrix);

cout << ans <<"\n";

}

return 0;

}

// } Driver Code Ends

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

**S.C :- O(1)**