**Boundary traversal of matrix: -**

**Easy** Accuracy: **35.32%** Submissions: **71K+** Points: **2**

You are given a matrix of dimensions **n x m**. The task is to perform boundary traversal on the matrix in a clockwise manner.  
  
**Example 1:**

**Input**:

n = 4, m = 4

matrix[][] = {{1, 2, 3, 4},

  {5, 6, 7, 8},

  {9, 10, 11, 12},

  {13, 14, 15,16}}

**Output**: 1 2 3 4 8 12 16 15 14 13 9 5

**Explanation**:

The matrix is:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

The boundary traversal is:

1 2 3 4 8 12 16 15 14 13 9 5

**Example 2:**

**Input**:

n = 3, m = 4

matrrix[][] = {{12, 11, 10, 9},

  {8, 7, 6, 5},

  {4, 3, 2, 1}}

**Output**: 12 11 10 9 5 1 2 3 4 8

**Your Task:**  
Complete the function **boundaryTraversal()**that takes matrix, n and m as input parameters and returns the list of integers that formthe boundary traversal of the matrix in a clockwise manner.  
  
**Expected Time Complexity:** O(N + M)  
**Expected Auxiliary Space:** O(N+M)  
  
**Constraints:**  
1 <= n, m<= 1000  
0 <= matrixi <= 1000

**Code: -**

//{ Driver Code Starts

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution

{

public:

//Function to return list of integers that form the boundary

//traversal of the matrix in a clockwise manner.

vector<int> boundaryTraversal(vector<vector<int> > matrix, int n, int m){

vector<int> ans;

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

ans.push\_back(matrix[0][j]);

for(int i=1; i<n; ++i)

ans.push\_back(matrix[i][m-1]);

if(n > 1){

for(int j=m-2; j>=0; --j)

ans.push\_back(matrix[n-1][j]);

}

if(m > 1){

for(int i=n-2; i>=1; --i)

ans.push\_back(matrix[i][0]);

}

return ans;

}

};

//{ Driver Code Starts.

int main() {

int t;

cin>>t;

while(t--)

{

int n,m;

cin>>n>>m;

vector<vector<int> > matrix(n);

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

{

matrix[i].assign(m, 0);

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

{

cin>>matrix[i][j];

}

}

Solution ob;

vector<int> result = ob.boundaryTraversal(matrix, n, m);

for (int i = 0; i < result.size(); ++i)

cout<<result[i]<<" ";

cout<<endl;

}

return 0;

}

// } Driver Code Ends

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

**S.C: - O(1) excluding answer storage**