**Replace O's with X's: -**

Medium Accuracy: 34.0% Submissions: 67K+ Points: 4

Given a matrix **mat** of size **N x M** where every element is either 'O' or 'X'. Replace all 'O' with 'X' that are surrounded by 'X'.

A 'O' (or a set of 'O') is considered to be surrounded by 'X' if there are 'X' at locations just below, just above, just left and just right of it.

**Example 1:**

**Input:**   
n = 5, m = 4

mat = {{'X', 'X', 'X', 'X'},

  {'X', 'O', 'X', 'X'},

  {'X', 'O', 'O', 'X'},

  {'X', 'O', 'X', 'X'},

  {'X', 'X', 'O', 'O'}}

**Output:**   
ans = {{'X', 'X', 'X', 'X'},

  {'X', 'X', 'X', 'X'},

  {'X', 'X', 'X', 'X'},

  {'X', 'X', 'X', 'X'},

  {'X', 'X', 'O', 'O'}}

**Explanation:**   
Following the rule the above matrix is the resultant matrix.

**Example 2:**

**Input:**   
n = 5, m = 4

mat = {{'X', 'O', 'X', 'X'},

  {'X', 'O', 'X', 'X'},

  {'X', 'O', 'O', 'X'},

  {'X', 'O', 'X', 'X'},

  {'X', 'X', 'O', 'O'}}

**Output:**   
ans = {{'X', 'O', 'X', 'X'},

  {'X', 'O', 'X', 'X'},

  {'X', 'O', 'O', 'X'},

  {'X', 'O', 'X', 'X'},

  {'X', 'X', 'O', 'O'}}

**Explanation:**   
Following the rule the above matrix is the resultant matrix.

**Your Task:**  
You do not need to read input or print anything. Your task is to complete the function **fill()** which takes **N**, **M** and **mat** as input parameters ad returns a 2D array representing the resultant matrix.

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

**Constraints:**  
1 ≤ n, m ≤ 500

**Code: -**

//{ Driver Code Starts

// Initial Template for C++

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

// User function Template for C++

class Solution{

public:

void bfs(vector<vector<char>> &mat, vector<vector<bool>> &vis, int x, int y){

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

vis[x][y] = true;

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

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

queue<pair<int,int>> q;

q.push({x,y});

while(q.size()){

int count = q.size();

while(count--){

int frontx = q.front().first;

int fronty = q.front().second;

q.pop();

// moving all 4 directions

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

if(0<=frontx+dx[i] and frontx+dx[i]<n and 0<=fronty+dy[i] and fronty+dy[i]<m

and !vis[frontx+dx[i]][fronty+dy[i]] and mat[frontx+dx[i]][fronty+dy[i]]=='O'){

q.push({frontx+dx[i], fronty+dy[i]});

vis[frontx+dx[i]][fronty+dy[i]] = true;

}

}

}

}

return;

}

vector<vector<char>> fill(int n, int m, vector<vector<char>> &mat){

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

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

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

// 1st row and last row

if((i==0 or i==n-1) and mat[i][j]=='O' and !vis[i][j])

bfs(mat, vis, i, j);

else if((j==0 or j==m-1) and mat[i][j]=='O' and !vis[i][j])

bfs(mat, vis, i, j);

}

}

// for final answer

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

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

if(!vis[i][j] and mat[i][j]=='O')

mat[i][j] = 'X';

}

}

return mat;

}

};

//{ Driver Code Starts.

int main(){

int t;

cin>>t;

while(t--){

int n, m;

cin>>n>>m;

vector<vector<char>> mat(n, vector<char>(m, '.'));

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

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

cin>>mat[i][j];

Solution ob;

vector<vector<char>> ans = ob.fill(n, m, mat);

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

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

cout<<ans[i][j]<<" ";

}

cout<<"\n";

}

}

return 0;

}

// } Driver Code Ends

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

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