Given an n x m grid, where each cell has the following values :

2 - represents a rotten orange

1 - represents a Fresh orange

0 - represents an Empty Cell

Every minute, if a fresh orange is adjacent to a rotten orange in **4-direction** ( upward, downwards, right, and left ) it becomes rotten.

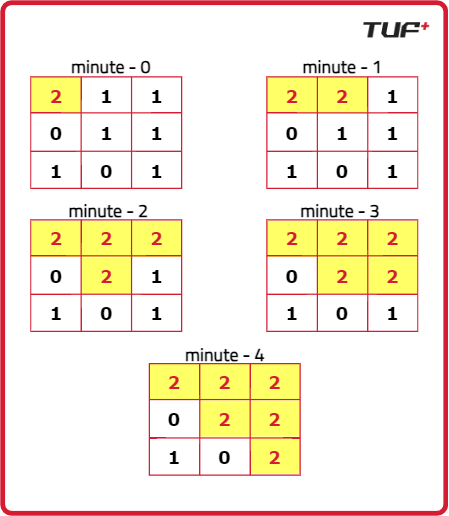
Return the **minimum** number of **minutes** required such that none of the cells has a Fresh Orange. If it's not possible, return -1.

**Examples:**

**Input:** grid = [ [2, 1, 1] , [0, 1, 1] , [1, 0, 1] ]

**Output:** -1

**Explanation:** Orange at (3,0) cannot be rotten.



Solution :

class Solution {

public:

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

int n = grid.size();

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

int tm = 0;

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

queue<pair<pair<int, int>, int>> q;

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

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

if (grid[i][j] == 2) {

q.push({{i, j}, 0});

vis[i][j] = 2;

}

}

}

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

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

while (!q.empty()) {

int row = q.front().first.first;

int col = q.front().first.second;

int time = q.front().second;

tm = max(tm, time);

q.pop();

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

int nrow = row + delrow[i];

int ncol = col + delcol[i];

if (nrow >= 0 && nrow < n && ncol >= 0 && ncol < m &&

grid[nrow][ncol] == 1 && vis[nrow][ncol] != 2) {

q.push({{nrow, ncol}, time + 1});

vis[nrow][ncol] = 2;

}

}

}

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

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

if (grid[i][j] == 1 && vis[i][j] != 2) {

return -1;

}

}

}

return tm;

}

};

TC : O(4\*n\*m)

SC :O(n\*m)