https://course.acciojob.com/idle?question=b21cba45-2a97-4492-82f7-5e23ed20ac00

- EASY
- Max Score: 30 Points
-
-
-
-
-

Rotting Oranges

You are given an m x n grid where each cell can have one of three values:

- 0 representing an empty cell,
- 1 representing a fresh orange, or
- 2 representing a rotten orange.

Every minute, any fresh orange that is 4-directionally adjacent to a rotten orange becomes rotten.

Return the minimum number of minutes that must elapse until no cell has a fresh orange. If this is impossible, return -1.

Input Format

The first line contains two integers r and c.

The next r lines contains c spaced integers , elements of matrix.

Output Format

Print the minimum number of minutes.

Example 1

Input

```
3 3
2 1 1
1 1 0
0 1 1
```

Output

```
4
```

Explanation

After 4 min all the fresh oranges becomes rotten.

Example 2

Input

```
3 3
2 1 1
0 1 1
1 0 1
```

Output:

```
-1
```

Explanation

The orange in the bottom left corner (row 2, column 0) is never rotten, because rotting only happens 4-directionally.

Constraints

m == grid.length

n == grid[i].length

1 <= m, n <= 10

grid[i][j] is 0, 1, or 2.

Topic Tags

- **BFS**

My code

```
// n java
import java.util.*;
import java.lang.*;
import java.io.*;

public class Main {

          public static void main (String[] args) throws IOException {
                  Scanner sc = new Scanner(System.in);
                int n = sc.nextInt();
                int m = sc.nextInt();
                int [][] arr= new int[n][m];
                for(int i=0;i<n;i++){
                    for(int j=0;j<m;j++){
                        arr[i][j]=sc.nextInt();
                    }
                }
                Solution obj= new Solution();

                System.out.println(obj.orangesRotting(arr));
                }
}
```

```
//
class Solution{
	public static void bfs(int[][] grid, int[][] vis, LinkedList<int[]> q) {
		int m = grid.length;
		int n = grid[0].length;

		int[] di = {-1, 1, 0, 0};
		int[] dj = {0, 0, -1, 1};

		while(q.size() > 0) {
			int[] f = q.poll();

			for(int k = 0; k < 4; k++) {
				int i = f[0] + di[k], j = f[1] + dj[k];

				if(i < 0 || i >= m || j < 0 || j >= n || grid[i][j] == 0)
continue;

				if(vis[i][j] == -1) {
					vis[i][j] = vis[f[0]][f[1]] + 1;
					int[] p = {i, j};
					q.add(p);
				}
			}
		}
	}
	 public static int orangesRotting(int[][] grid) {

//your code
		 int m = grid.length;
}

int n = grid[0].length;

 int[][] vis = new int[m][n];

 LinkedList<int[]> q = new LinkedList<>();

 for(int i = 0; i < m; i++) {
         for(int j = 0; j < n; j++) {
                 vis[i][j] = -1;
                 if(grid[i][j] == 2) {
                         int[] p = {i, j};
                         q.add(p);

                         vis[i][j] = 0;
                 }
         }
 }

 bfs(grid, vis, q);

 int ans = 0;

 for(int i = 0; i < m; i++) {
         for(int j = 0; j < n; j++) {
                 if(vis[i][j] == -1 && grid[i][j] == 1) return -1;

                 ans = Math.max(ans, vis[i][j]);
         }
 }
```

```
                return ans;

        }
}
```