Full Name:	GUILHERME FERNANDES
Email:	contato@guifr.com.br
Test Name:	**Mock Test**
Taken On:	6 Dec 2022 05:02:08 IST
Time Taken:	30 min 30 sec/ 40 min
Invited by:	Ankush
Invited on:	6 Dec 2022 05:02:00 IST
Skills Score:	
Tags Score:	Algorithms 100/100 Core CS 100/100 Medium 100/100 Search 100/100 problem-solving 100/100

100%

100/100

scored in **Mock Test** in 30 min 30 sec on 6 Dec 2022 05:02:08 IST

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Connected Cells in a Grid > **Coding**	30 min 19 sec	100/ 100	✓

QUESTION 1

Correct Answer

Score 100

Connected Cells in a Grid › Coding

Search | Algorithms | Medium | problem-solving | Core CS

QUESTION DESCRIPTION

Consider a matrix where each cell contains either a 0 or a 1. Any cell containing a 1 is called a *filled* cell. Two cells are said to be *connected* if they are adjacent to each other horizontally, vertically, or diagonally. In the following grid, all cells marked `X` are connected to the cell marked `Y`.

```
XXX
XYX
XXX
```

If one or more filled cells are also connected, they form a *region*. Note that each cell in a region is connected to zero or more cells in the region but is not necessarily directly connected to all the other cells in the region.

Given an $n \times m$ matrix, find and print the number of cells in the largest *region* in the matrix. Note that

there may be more than one region in the matrix.

For example, there are two regions in the following 3×3 matrix. The larger region at the top left contains 3 cells. The smaller one at the bottom right contains 1.

```
110
100
001
```

Function Description

Complete the *connectedCell* function in the editor below.

connectedCell has the following parameter(s):
- *int matrix[n][m]*: $matrix[i]$ represents the i^{th} row of the matrix

Returns
- *int:* the area of the largest region

Input Format

The first line contains an integer n, the number of rows in the matrix.
The second line contains an integer m, the number of columns in the matrix.
Each of the next n lines contains m space-separated integers $matrix[i][j]$.

Constraints

- $0 < n, m < 10$

Sample Input

```
STDIN       Function
-----       --------
4           n = 4
4           m = 4
1 1 0 0     grid = [[1, 1, 1, 0], [0, 1, 1, 0], [0, 0, 1, 0], [1, 0, 0,
0]]
0 1 1 0
0 0 1 0
1 0 0 0
```

Sample Output

```
5
```

Explanation

The diagram below depicts two regions of the matrix. Connected regions are filled with X or Y. Zeros are replaced with dots for clarity.

```
X X . .
. X X .
. . X .
Y . . .
```

The larger region has 5 cells, marked `X`.

CANDIDATE ANSWER

Language used: **Java 8**

```
class Result {
```

```
class Result {

    /*
     * Complete the 'connectedCell' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts 2D_INTEGER_ARRAY matrix as parameter.
     */

    public static int connectedCell(List<List<Integer>> matrix) {

        print(matrix);

        int result = 0;

        for(int r = 0; r < matrix.size(); r++){
            for(int c = 0; c < matrix.get(r).size();c++){
                if(matrix.get(r).get(c) == 1)
                result = Math.max(result,goProcess(r,c,matrix));
                //print(matrix);
            }
        }


        return result;

    }

    public static void print(List<List<Integer>> matrix){

        for(List<Integer> rows : matrix){
            for(int element : rows){
                System.out.print(element);
            }
            System.out.println();
        }
        System.out.println("#######");
    }

    public static int goProcess(int startR, int startC, List<List<Integer>>
matrix){
        int result = 1;

        matrix.get(startR).set(startC, 2);

        //r+1
        if(startR+1 < matrix.size())
            if(matrix.get(startR+1).get(startC) == 1) result +=
goProcess(startR+1, startC, matrix);

         //r-1
        if(startR-1 >= 0)
            if(matrix.get(startR-1).get(startC) == 1) result +=
goProcess(startR-1, startC, matrix);

        //c+1
        if(startC+1 < matrix.get(startR).size())
            if(matrix.get(startR).get(startC+1) == 1) result +=
goProcess(startR, startC+1, matrix);

        //c-1
        if(startC-1 >= 0)
            if(matrix.get(startR).get(startC-1) == 1) result +=
goProcess(startR, startC-1, matrix);
```

```
        //r+1 c+1
        if(startC+1 < matrix.get(startR).size())
            if(startR+1 < matrix.size())
                if(matrix.get(startR+1).get(startC+1) == 1) result +=
goProcess(startR+1, startC+1, matrix);

        //r-1 c-1
        if(startC-1 >= 0)
            if(startR-1 >= 0)
                if(matrix.get(startR-1).get(startC-1) == 1) result +=
goProcess(startR-1, startC-1, matrix);

        //c+1 r-1
        if(startC+1 < matrix.get(startR).size())
            if(startR-1 >= 0)
                if(matrix.get(startR-1).get(startC+1) == 1) result +=
goProcess(startR-1, startC+1, matrix);

        //c-1 r+1
        if(startC-1 >= 0)
            if(startR+1 < matrix.size())
                if(matrix.get(startR+1).get(startC-1) == 1) result +=
goProcess(startR+1, startC-1, matrix);

        return result;
    }

}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	Success	0	0.2398 sec	29.5 KB
Testcase 2	Easy	Hidden case	Success	20	0.1587 sec	30.1 KB
Testcase 3	Easy	Sample case	Success	0	0.132 sec	30.2 KB
Testcase 4	Easy	Hidden case	Success	20	0.1471 sec	30 KB
Testcase 5	Easy	Hidden case	Success	20	0.1826 sec	29.7 KB
Testcase 6	Easy	Hidden case	Success	20	0.2386 sec	29.9 KB
Testcase 7	Easy	Hidden case	Success	20	0.1102 sec	30.1 KB

No Comments