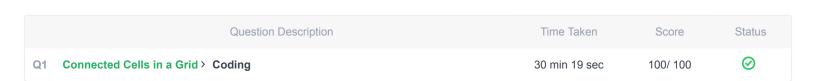
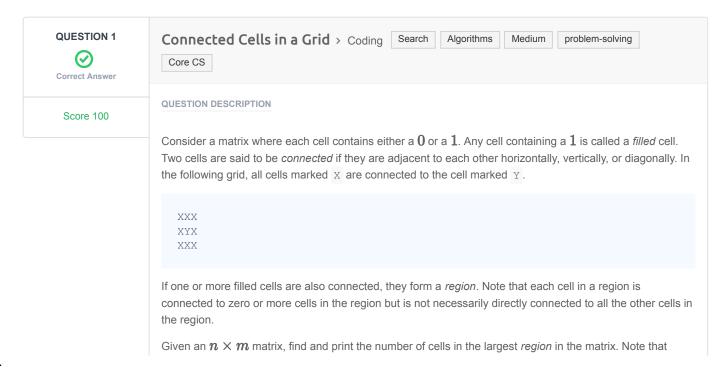


### **Recruiter/Team Comments:**

No Comments.





there may be more than one region in the matrix.

For example, there are two regions in the following  $3 \times 3$  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

# **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.

The larger region has  $\bf 5$  cells, marked  $\, {\tt X} \, .$ 

#### **CANDIDATE ANSWER**

Language used: Java 8

-1--- D---1+ (

```
4
        * Complete the 'connectedCell' function below.
        * The function is expected to return an INTEGER.
        * The function accepts 2D INTEGER ARRAY matrix as parameter.
 8
       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>>
41 matrix) {
     int result = 1;
          matrix.get(startR).set(startC, 2);
           //r+1
           if(startR+1 < matrix.size())</pre>
47
              if (matrix.get(startR+1).get(startC) == 1) result +=
49 goProcess(startR+1, startC, matrix);
            //r-1
           if(startR-1 >= 0)
              if(matrix.get(startR-1).get(startC) == 1) result +=
54 goProcess(startR-1, startC, matrix);
           //c+1
           if(startC+1 < matrix.get(startR).size())</pre>
               if(matrix.get(startR).get(startC+1) == 1) result +=
59 goProcess(startR, startC+1, matrix);
           //c-1
          if(startC-1 >= 0)
              if (matrix.get(startR).get(startC-1) == 1) result +=
64 goProcess(startR, startC-1, matrix);
```

Class Result {

```
//r+1 c+1
           if(startC+1 < matrix.get(startR).size())</pre>
               if(startR+1 < matrix.size())</pre>
                   if (matrix.get(startR+1).get(startC+1) == 1) result +=
70 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 +=
76 goProcess(startR-1, startC-1, matrix);
           //c+1 r-1
           if(startC+1 < matrix.get(startR).size())</pre>
              if(startR-1 >= 0)
                   if (matrix.get(startR-1).get(startC+1) == 1) result +=
82 goProcess(startR-1, startC+1, matrix);
           //c-1 r+1
          if(startC-1 >= 0)
               if(startR+1 < matrix.size())</pre>
                   if (matrix.get(startR+1).get(startC-1) == 1) result +=
  goProcess(startR+1, startC-1, matrix);
           return result;
       }
```

TESTCASE						
TESTUASE	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

PDF generated at: 6 Dec 2022 00:04:34 UTC