Image Matching

Images are stored in the form of a grid. Image recognition is possible by comparing grids of two images and checking if they have any matching regions.

You are given two grids where each cell of the grids contains either a 0 or a 1. If two cells share a side then they are adjacent. Cells that contain 1 form a connected region if any cell of that region can be reached by moving through the adjacent cells that contain 1. Overlay the first grid onto the second and if a region of the first grid completely matches a region of the second grid, the regions are matched. Count total number of such matched regions in the second grid.

For example, given two 3x3 grids 1 and 2:

111 111 → 111 111
100 100 → 100 100
100 101 → 100 101

There are 2 regions in the second grid: { (0,0),(0,1),(0,2),(1,0),(2,0) } and { (2,2) }.
Regions in grid 1 cover the first region of grid 2, but not the second region. There is 1 matching region.

Making a slight alteration to the above example:

111 111
101 100
100 101

There are no matching regions. From the first graph, the 1 at position (1,2) is not matched in the second grid's larger region. The second grid position (2,2) is not matched in grid 1.

Function Description
Complete the function countMatches in the editor below. The function must return the number of matching regions.

countMatches has the following parameter(s):
grid1[grid1[0],...grid1[n-1]]: an array of bit strings representing the rows of image 1
grid2[grid2[0],...grid2[n-1]]: an array of bit strings representing the rows of image 2

Constraints

- 1 ≤ n ≤ 100
- 1 ≤ |grid1[i]|, |grid2[i]| ≤ 100
- grid cells contain only 0 or 1

Input Format For Custom Testing

Sample Case 0

Sample Input 0

```
3
001
011
100
3
001
011
101
```

Sample Output 0

The first grid forms 2 regions. They are { (0,2), (1,1), (1,2) } and { (2,0) }
The second grid forms 2 regions. They are { (0,2), (1,1) , (1,2), (2,2)} and { (2,0) }
So, only one region matches.

Sample Case 1

Sample Input 1

```
4
0100
1001
0011
0011
4
0101
1001
0011
0011
```

Sample Output 1

```
2
```

Explanation 1
The first grid forms 3 regions. They are { (0,1) }, { (1,0) } and { (1,3), (2,2), (2,3), (3,2), (3,3) }
The second grid forms 3 regions. They are { (0,1) } , { (1,0) } and { (0,3), (1,3), (2,2), (2,3), (3,2), (3,3) }
So, two regions match.

Sample Case 2

Sample Input 2

```
4
0010
0111
0100
1111
4
0010
0111
0110
1111
```

Sample Output 2

```
0
```

Explanation 2
The first grid forms 1 region. It is { (0,2) , (1,1), (1,2), (1,3) , (2,1), (3,0), (3,1), (3,2), (3,3) }
The second grid forms 1 region. It is { (0,2) , (1,1), (1,2), (1,3) , (2,1), (2,2), (3,0), (3,1), (3,2), (3,3) }
So, no regions match.

YOUR ANSWER

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour. Start tour ✖

Draft saved 08:04 am Original Code Java 8

```
     */
    private static int cnt = 0;
    private static Set<Integer> set = new HashSet<>();
    private static int[][] dirs = { { 1, 0 }, { -1, 0 }, { 0, 1 }, { 0, -1 } };
    public static int countMatches(List<String> grid1, List<String> grid2) {
```

Twitter 2018-2019 University Recruiting Coding C... 6d 06h to test end 2/3 Attempted Amanda Xu

- Section 1 -

1

2

- Section 2 -

3

```
            if (grid1.get(i).length() != grid2.get(i).length()) return cnt;
        }
        int m = grid1.size();
        int n = grid1.get(0).length();
        //initilize
        List<int[]> delete = new ArrayList<>();
        int[] roots = new int[m * n];
        Arrays.fill(roots , -1);
```

Line: 30 Col: 17

Test against custom input

Run Code Submit code & Continue

(You can submit any number of times)

Download sample test cases The input/output files have Unix line endings. Do not use Notepad to edit them on windows.

Status: Compiled successfully. 4/13 test cases passed.

Test case 3 ✕

Test case 4 ✕

Test case 5 ✕

Test case 6 ✕

Test case 7 ✕

Test case 9 ✕

Test case 10 ✕

Test case 11 ✕

Test case 12 ✕

Compiler Message

```
Wrong Answer
```

Your Output (stdout)

```
Output hidden
```

About Privacy Policy Terms of Service