code warrior

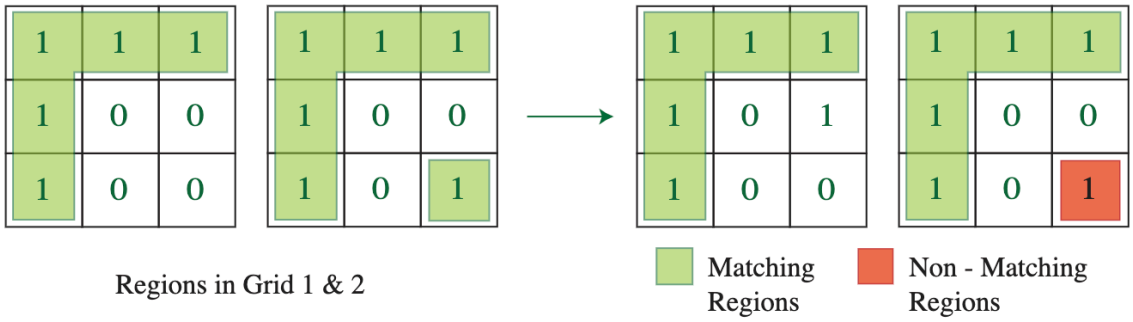
Infrastructure CodeWarrior Challenge #1

Problem #1: Image Matching
Score points: 100

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*.

There are two grids where each cell of the grids contains either *0* or *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 by row or column 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 the total number of such matched regions in the second grid.

Example
given two 3x3 grids *1* and *2*:
grid1= [111, 100, 100]
grid2 = [111, 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.

countMatches has the following parameter(s):

string grid1[n]: an array of bit strings representing the rows of image 1

string grid2[n]: an array of bit strings representing the rows of image 2

Returns:

int: number of matching regions.

Constraints

- *1 ≤ n ≤ 100*
- *1 ≤ lengths of grid1[i] and grid2[i] ≤ 100*
- *grid cells contain only 0 or 1*

Sample Cases 0:

Sample Input 0

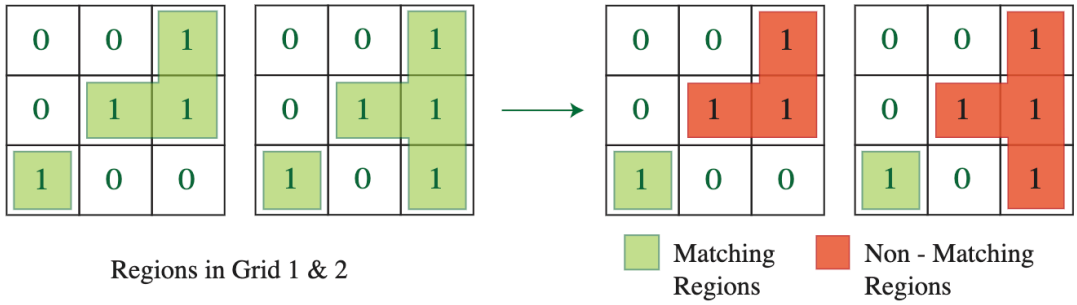
```
STDIN    Function
-----    --------
3     →  grid1[] size n = 3
001   →  grid1 = ['001', '011', '100']
```

```
011
100
3     → grid2[] size n = 3
001   → grid2 = ['001', '011', '101']
011
101
```

Sample Output 0

```
1
```

Explanation 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

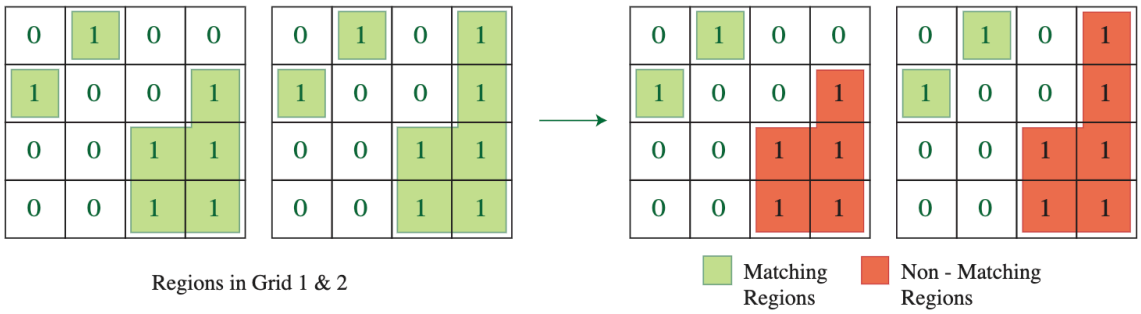
Sample Input 1

```
STDIN       Function
-----       --------
4       →   grid1[] size n = 4
0100    →   grid1 = ['0100', '1001', '0011', '0011']
1001
0011
0011
4       →   grid2[] size n = 4
0101    →   grid2 = ['0101', '1001', '0011', '0011']
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

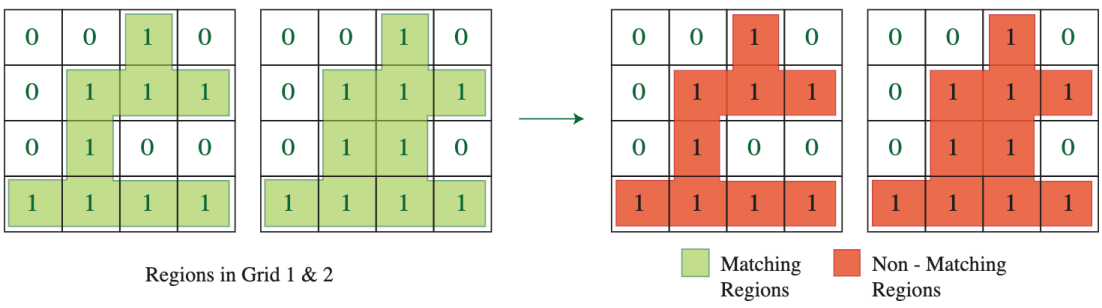
Sample Input 2

```
STDIN       Function
-----       --------
4       →   grid1[] size n = 4
0010    →   grid1 = ['0010', '0111', '0100', '1111']
0111
0100
1111
4       →   grid2[] size n = 4
0010    →   grid2 = ['0010', '0111', '0110', '1111']
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) }
No regions match.

Problem #2: System Reporting
Score points: 50

In production environment, developers are restricted to SSH into servers without VP approval. Therefore, it is critical that our team provide tools to allow developers to debug problems using our monitoring tools.

For example, when we get an alert that a disk is getting full, you would want to know what files are using up all of the space.

Write a program in a language of your choice which will take a mount point as an argument and return a list of all the files on the mountpoint and their disk usage in bytes in json format.

Sample Case 1

getdiskusage.py /tmp

{
"files":

[
{"/tmp/foo", 1000}, {"/tmp/bar", 1000000}, {"/tmp/buzzz", 42},

], }