Codility_

CodeCheck Report: training5H4PK9-WG4

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
CountriesCount Python	1 min	100%

Total score

Tasks Details

Hard

1. **CountriesCount**

Count the number of different countries that a map contains.

Task Score	Correctness	Performance
100%	100%	100%

Task description

A rectangular map consisting of N rows and M columns of square areas is given. Each area is painted with a certain color.

Two areas on the map *belong to the same country* if the following conditions are met:

- they have the same color;
- it is possible to travel from one area to the other orthogonally (that is, by moving only north, south, west or east) without moving over areas of a different color.

The map can be described by a zero-indexed matrix A consisting of N rows and M columns of integers. The color of each area is described by the corresponding element of the matrix. Two areas have the same color if and only if

Solution

Programming language used:	Python
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	*not defined yet*

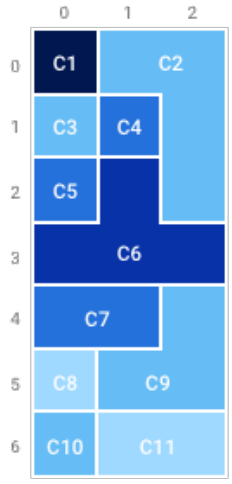
Task timeline

their corresponding matrix elements have the same value.

For example, consider the following matrix A consisting of seven rows and three columns:

```
A[0][0] = 5    A[0][1] = 4    A[0][2] = 4
A[1][0] = 4    A[1][1] = 3    A[1][2] = 4
A[2][0] = 3    A[2][1] = 2    A[2][2] = 4
A[3][0] = 2    A[3][1] = 2    A[3][2] = 2
A[4][0] = 3    A[4][1] = 3    A[4][2] = 4
A[5][0] = 1    A[5][1] = 4    A[5][2] = 4
A[6][0] = 4    A[6][1] = 1    A[6][2] = 1
```

Matrix A describes a map that is colored with five colors. The areas on the map belong to eleven different countries (C1–C11), as shown in the following figure:



Write a function

```
def solution(A)
```

that, given a zero-indexed matrix A consisting of N rows and M columns of integers, returns the number of different countries to which the areas of the map described by matrix A belong.

For example, given matrix A consisting of seven rows and three columns corresponding to the example above, the function should return 11.

Write an **efficient** algorithm for the following assumptions:

- N and M are integers within the range [1..300,000];
- the number of elements in matrix A is within the range [1..300,000];
- each element of matrix A is an integer within the range [−1,000,000,000..1,000,000,000].

07:41:45 07:42:16

Code: 07:42:16 UTC, py, final, score: **100** show code in pop-up

```
1  # you can write to stdout for debugging p
2  # print("this is a debug message")
3
4  def solution(A):
5      # Implement your solution here
6      n = len(A)
7      if n == 0:
8          return 0
9      m = len(A[0])
10     visited = [[False]*m for _ in range(n
11     countries = 0
12     directions = [(0,1),(1,0),(0,-1),(-1,
13
14     for i in range(n):
15         for j in range(m):
16             if not visited[i][j]:
17                 countries += 1
18                 color = A[i][j]
19                 stack = [(i, j)]
20                 while stack:
21                     x, y = stack.pop()
22                     if x<0 or x>=n or y<0
23                         continue
24                     visited[x][y] = True
25                     for dx, dy in directi
26                         stack.append((x+d
27
28     return countries
29
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(N*M)**

expand all	Example tests	
▶	example	✔ OK
expand all	**Correctness tests**	
▶	small_1x1	✔ OK
▶	positive_negative_zeros	✔ OK
▶	small_2x2	✔ OK
▶	small_3x3	✔ OK
▶	matrix_12x10	✔ OK
▶	matrix_10x10_labyrinth	✔ OK
▶	matrix_wide	✔ OK
▶	large_numbers	✔ OK
▶	anti_heuristics	✔ OK

expand all	Performance tests	
▶	medium_matrix	✔ OK
▶	medium_square_matrix	✔ OK
▶	medium_horizontal_matrix	✔ OK
▶	large_square_one_country	✔ OK
▶	max_matrix	✔ OK
▶	max_one_country	✔ OK