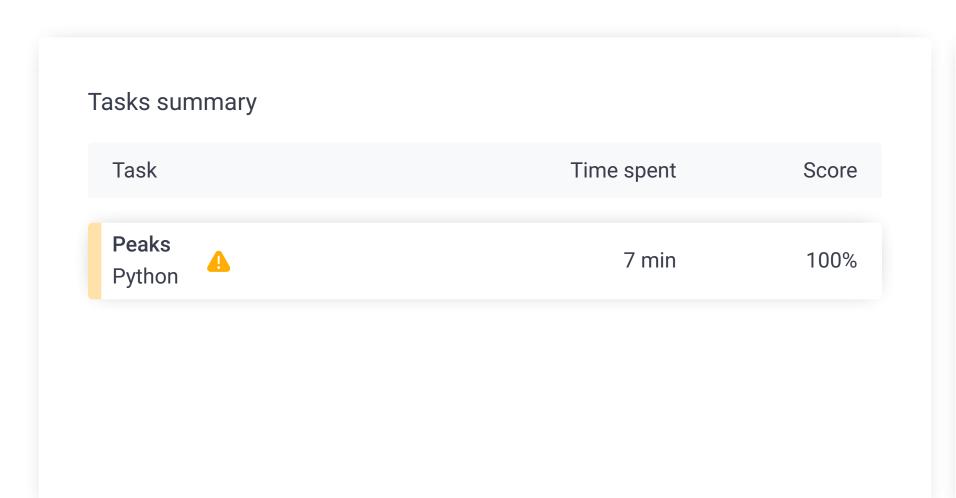
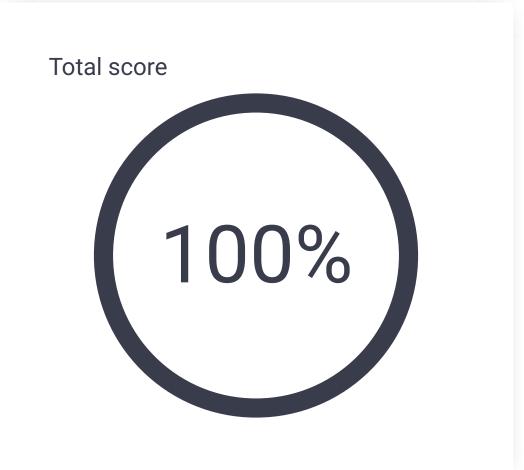
3

CodeCheck Report: trainingGDVQND-F5W

Test Name:

Timeline Summary





Tasks Details

1. Peaks

Divide an array into the maximum number of same-sized blocks, each of which should contain an index P such that A[P - 1] < A[P] > A[P + 1].

Task Score Performance Correctness 100% 100% 100%

Task description

A non-empty array A consisting of N integers is given.

A peak is an array element which is larger than its neighbors. More precisely, it is an index P such that 0 < P < N - 1, A[P - 1] < A[P] and A[P] > A[P + 1].

For example, the following array A:

- A[0] = 1
- A[1] = 2
- A[2] = 3A[3] = 4
- A[4] = 3
- A[5] = 4A[6] = 1
- A[7] = 2
- A[8] = 3
- A[9] = 4
- A[10] = 6
- A[11] = 2

has exactly three peaks: 3, 5, 10.

We want to divide this array into blocks containing the same number of elements. More precisely, we want to choose a number K that will yield the following blocks:

- A[0], A[1], ..., A[K 1],
- A[K], A[K + 1], ..., A[2K 1],
- A[N K], A[N K + 1], ..., A[N 1].

What's more, every block should contain at least one peak. Notice that extreme elements of the blocks (for example A[K - 1] or A[K]) can also be peaks, but only if they have both neighbors (including one in an adjacent blocks).

The goal is to find the maximum number of blocks into which the array A can be divided.

Array A can be divided into blocks as follows:

- one block (1, 2, 3, 4, 3, 4, 1, 2, 3, 4, 6, 2). This block contains three peaks.
- two blocks (1, 2, 3, 4, 3, 4) and (1, 2, 3, 4, 6, 2). Every block has a peak.
- three blocks (1, 2, 3, 4), (3, 4, 1, 2), (3, 4, 6, 2). Every block has a peak. Notice in particular that the first block (1, 2, 3, 4) has a peak at A[3], because A[2] < A[3] > A[4], even though A[4] is in the adjacent block.

However, array A cannot be divided into four blocks, (1, 2, 3), (4, 3, 4), (1, 2, 3) and (4, 6, 2), because the (1, 2, 3) blocks do not contain a peak. Notice in particular that the (4, 3, 4) block contains two peaks: A[3] and A[5].

The maximum number of blocks that array A can be divided into is three.

Write a function:

def solution(A)

that, given a non-empty array A consisting of N integers, returns the maximum number of blocks into which A can be divided.

If A cannot be divided into some number of blocks, the function should return 0. For example, given:

- A[0] = 1
- A[1] = 2A[2] = 3
- A[3] = 4
- A[4] = 3A[5] = 4
- A[6] = 1
- A[7] = 2
- A[8] = 3A[9] = 4
- A[10] = 6A[11] = 2

the function should return 3, as explained above.

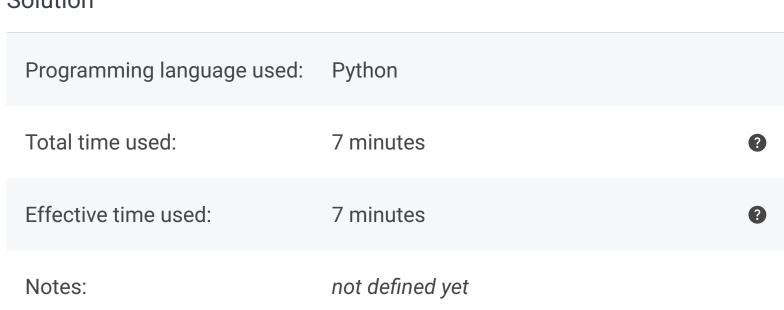
Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..100,000];
- each element of array A is an integer within the range [0..1,000,000,000].

Copyright 2009–2022 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Solution

Task timeline





19:46:2	19:46:29		
Code: 100	19:53:22 UTC, py, final, score:	show code in pop-up	
2 #	<pre>k = j * allowed_ while True: if peaks[k]: break if k == (j+1) bool_ = break k += 1 if bool_: break if not bool_:</pre>	<pre>N) .append(N//i) == 1: wed_block_sizes)-1, -1, llowed_block_sizes[i]): _block_sizes[i] : 1) * allowed_block_size</pre>	

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all	Example tests
example example test	✓ OK
expand all	Correctness tests
extreme_min extreme min test	✓ OK
extreme_without_pe test without peaks	aks V OK
prime_length test with prime sequence	✓ OK e length
anti_bin_search anti bin_search test	∠ OK
simple test	∠ OK
second simple test	∠ OK
expand all	Performance tests
medium_random chaotic medium sequence	
medium_anti_slow medium test anti slow so	✓ OK blutions
► large_random	✓ OK

✓ OK

✓ OK

chaotic large sequences, length = ~50,000

► large_anti_slow

extreme_max

extreme max test

large test anti slow solutions