Codility_

CodeCheck Report: training4W2BG3-Q34

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
MinAvgTwoSlice Python	4 min	100%

Total score

Tasks Details

1. MinAvgTwoSlice

Medium

Find the minimal average of any slice containing at least two elements.

Task Score	Correctness	Performance
100%	100%	100%

Task description

A non-empty array A consisting of N integers is given. A pair of integers (P, Q), such that 0 ≤ P < Q < N, is called a *slice* of array A (notice that the slice contains at least two elements). The *average* of a slice (P, Q) is the sum of A[P] + A[P + 1] + ... + A[Q] divided by the length of the slice. To be precise, the average equals (A[P] + A[P + 1] + ... + A[Q]) / (Q − P + 1).

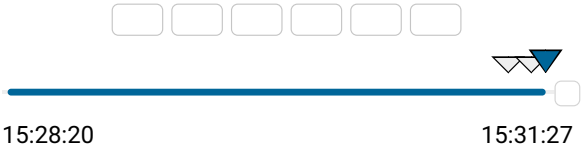
For example, array A such that:

```
A[0] = 4
A[1] = 2
A[2] = 2
A[3] = 5
A[4] = 1
A[5] = 5
A[6] = 8
```

Solution

Programming language used:	Python
Total time used:	4 minutes
Effective time used:	4 minutes
Notes:	*not defined yet*

Task timeline



contains the following example slices:

- slice (1, 2), whose average is (2 + 2) / 2 = 2;
- slice (3, 4), whose average is (5 + 1) / 2 = 3;
- slice (1, 4), whose average is (2 + 2 + 5 + 1) / 4 = 2.5.

The goal is to find the starting position of a slice whose average is minimal.

Write a function:

```
def solution(A)
```

that, given a non-empty array A consisting of N integers, returns the starting position of the slice with the minimal average. If there is more than one slice with a minimal average, you should return the smallest starting position of such a slice.

For example, given array A such that:

```
A[0] = 4
A[1] = 2
A[2] = 2
A[3] = 5
A[4] = 1
A[5] = 5
A[6] = 8
```

the function should return 1, as explained above.

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

- N is an integer within the range [2..100,000];
- each element of array A is an integer within the range [−10,000..10,000].

Code: 15:31:27 UTC, py, final, score: **100** show code in pop-up

```
1  # you can write to stdout for debugging pu
2  # print("this is a debug message")
3
4  def solution(A):
5      # Implement your solution here
6      # pass
7      N = len(A)
8      min_average = float('inf')
9      min_start_position = 0
10
11     for start in range(N - 1):
12         # Calculate the average of the sl
13         average2 = (A[start] + A[start + 
14         if average2 < min_average:
15             min_average = average2
16             min_start_position = start
17
18         if start < N - 2:
19             # Calculate the average of th
20             average3 = (A[start] + A[star
21             if average3 < min_average:
22                 min_average = average3
23                 min_start_position = star
24
25     return min_start_position
26
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all	**Example tests**	
▶	example example test	✔ OK
expand all	**Correctness tests**	
▼	double_quadruple two or four elements	✔ OK
1.	0.012 s	OK
2.	0.012 s	OK
3.	0.012 s	OK
4.	0.012 s	OK
▶	simple1 simple test, the best slice has length 3	✔ OK
▶	simple2	✔ OK

simple test, the best slice has length 3		
▶	small_random random, length = 100	✔ OK
▶	medium_range increasing, decreasing (legth = ~100) and small functional	✔ OK

expand all **Performance tests**

▶	medium_random random, N = ~700	✔ OK
▶	large_ones numbers from -1 to 1, N = ~100,000	✔ OK
▶	large_random random, N = ~100,000	✔ OK
▶	extreme_values all maximal values, N = ~100,000	✔ OK
▶	large_sequence many seqeneces, N = ~100,000	✔ OK