

TASKS DETAILS

MEDIUM

1. MinAvgTwoSlice

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

Task Score

100%

Correctness

100%

Performance

100%

Task description

A non-empty zero-indexed array A consisting of N integers is given. A pair of integers (P, Q), such that  $0 \leq P < Q < N$ , 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  $A[P] + A[P + 1] + \dots + A[Q]$  divided by the length of the slice. To be precise, the average equals  $(A[P] + A[P + 1] + \dots + A[Q]) / (Q - P + 1)$ .

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
```

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:

```
class Solution { public int solution(int[] A); }
```

that, given a non-empty zero-indexed 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.

Assume that:



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

Complexity:

- expected worst-case time complexity is  $O(N)$ ;
- expected worst-case space complexity is  $O(N)$ , beyond input storage (not counting the storage required for input arguments).

Copyright 2009–2018 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Solution

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

Task timeline

04:45:15

04:46:07

Code: 04:46:07 UTC, java, final, score: 100

[show code in pop-up](#)

```
1  class Solution {
2      public int solution(int[] A) {
3          int N = A.length;
4
5          if (N == 2) {
6              return 0;
7          }
8
9          int[] sumA= new int[N+1];
10         sumA[0] = 0;
11         for (int i=0; i < N; i++) {
12             sumA[i+1] = sumA[i] + A[i];
13         }
14
15         int currentInd = 0;
16         int minInd = 0;
17         double minAve = (double) sumA[2]/2;
18
19         while (currentInd < N-1) {
20             int i = 1;
21             while (currentInd+i<N && i<3) {
22                 double ave = (double) (sumA[currentInd+:
23                     if (ave < minAve) {
24                         minAve = ave;
25                         minInd = currentInd;
26                     }
27                     i++;
28                 }
29
30                 if (currentInd+4 < N && A[currentInd+4]==A[
31                     && A[currentInd+2]==A[currentInd+1] && i
32                     currentInd += 5;
33                     while (currentInd<N && A[currentInd] ==
34                         currentInd++;
35                     }
36                     if (currentInd < N) {
37                         currentInd -= 2;
38                     }
39                 }
40                 else {
41                     currentInd++;
42                 }
43             }
44
45             return minInd;
46         }
47     }
```

Analysis summary

The solution obtained perfect score.

Analysis?

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

collapse all

Example tests

▼ example

example test

OK

1. 0.004 s OK

collapse all

Correctness tests

▼ double\_quadruple

two or four elements

OK

1. 0.004 s OK

2. 0.004 s OK

3. 0.004 s OK

4. 0.008 s OK

▼ simple1

simple test, the best slice has length 3

OK

1. 0.004 s OK

2. 0.004 s OK

▼ simple2

simple test, the best slice has length 3

OK

1. 0.004 s OK

▼ small\_random

random, length = 100

OK

1. 0.004 s OK

▼ medium\_range

increasing, decreasing (legth = ~100) and small functional

OK

1. 0.004 s OK

2. 0.004 s OK

3. 0.004 s OK

collapse all

Performance tests

▼ medium\_random

random, N = ~700

OK

1. 0.008 s OK

▼ large\_ones

numbers from -1 to 1, N = ~100,000

OK

1. 0.168 s OK

2. 0.124 s OK

▼ large\_random

random, N = ~100,000

OK

1. 0.236 s OK

▼ extreme\_values

all maximal values, N = ~100,000

OK

1. 0.248 s OK

2. 0.256 s OK

3. 0.252 s OK

▼ large\_sequence

many sequeences, N = ~100,000

OK

1. 0.164 s OK

2. 0.128 s OK