Candidate Report: Anonymous

Test Name:

Summary | Timeline

Test Score

100 out of 100 points

100%

Tasks in Test

	Time Spent	Task Score
MinAvgTwoSlice Submitted in: Java	116 min	100%

TASKS DETAILS

MEDIUM

1. MinAvgTwoSlice

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
```

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 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].

Solution

Programming language used:	Java
Total time used:	116 minutes
Effective time used:	116 minutes
Notes:	*not defined yet*

Task timeline

22:00:32 — 23:56:14

Code: 23:56:14 UTC, java, final, score: 100

show code in pop-up

```java
import java.util.*;

class Solution {
    public int solution(int[] A) {
        double minAvg = (A[0] + A[1]) / 2;
        int minInd = 0;
        double curAvg;

        for (int i = 0; i < A.length - 2; i++) {

            curAvg = (A[i]+A[i+1])/2.0;
            if (minAvg > curAvg) {
                minAvg = curAvg;
                minInd = i;
            }

            curAvg = (A[i]+A[i+1]+A[i+2])/3.0;
            if (minAvg > curAvg) {
                minAvg = curAvg;
                minInd = i;
            }
        }

        curAvg = ( A[A.length - 1] + A[A.length -2] ) / 2.0;
        if (minAvg > curAvg) {
            minAvg = curAvg;
            minInd = A.length - 2;
        }

        return minInd;
    }
}
```

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
▶	simple1 simple test, the best slice has length 3	✔ OK
▶	simple2 simple test, the best slice has length 3	✔ OK
▶	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 segeneces, N = ~100,000	✔ OK