

Candidate Report: Anonymous

Test Name:

Summary

Timeline

Test Score

Tasks in Test

0 out of 100 points

0%

TapeEquilibrium
Submitted in: Java 8

Time Spent ⓘ

Task Score

29 min

0%

TASKS DETAILS

EASY	1. TapeEquilibrium Minimize the value $ (A[0] + \dots + A[P-1]) - (A[P] + \dots + A[N-1]) $.	Task Score 0%	Correctness 0%	Performance ⓘ 0%
------	---	------------------	-------------------	---------------------

Task description

A non-empty array A consisting of N integers is given. Array A represents numbers on a tape.

Any integer P, such that $0 < P < N$, splits this tape into two non-empty parts: $A[0], A[1], \dots, A[P - 1]$ and $A[P], A[P + 1], \dots, A[N - 1]$.

The *difference* between the two parts is the value of: $| (A[0] + A[1] + \dots + A[P - 1]) - (A[P] + A[P + 1] + \dots + A[N - 1]) |$

In other words, it is the absolute difference between the sum of the first part and the sum of the second part.

For example, consider array A such that:

A[0] = 3
A[1] = 1
A[2] = 2
A[3] = 4
A[4] = 3

We can split this tape in four places:

- P = 1, difference = $| 3 - 10 | = 7$
- P = 2, difference = $| 4 - 9 | = 5$
- P = 3, difference = $| 6 - 7 | = 1$
- P = 4, difference = $| 10 - 3 | = 7$

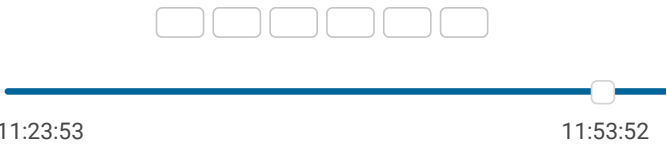
Write a function:

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

Solution

Programming language used:	Java 8	
Total time used:	29 minutes	?
Effective time used:	29 minutes	?
Notes:	not defined yet	

Task timeline



Code: 11:52:02 UTC, java, final, [show code in pop-up](#)
score: 0

```
1 // you can also use imports, for example:  
2 // import java.util.*;  
3  
4 // you can write to stdout for debugging purposes, e.g.  
5 // System.out.println("this is a debug message");  
6
```

that, given a non-empty array A of N integers, returns the minimal difference that can be achieved.

For example, given:

A[0] = 3
A[1] = 1
A[2] = 2
A[3] = 4
A[4] = 3

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 [−1,000..1,000].

Copyright 2009–2020 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```
7  class Solution {
8      public static int solution(int[] a) {
9
10         int s = 0;
11         for (int i = 0; i < a.length; i++)
12             s = s + a[i];
13
14         System.out.println(s);
15
16         int m = Integer.MAX_VALUE;
17         int l = 0;
18         int r = 1;
19         for (int p = 1; p < a.length; p++) {
20             l = l + a[p];
21             System.out.println(l);
22
23             int d = Math.abs(2 * l - s);
24             System.out.println(d);
25
26             if (d < m) {
27                 m = d;
28                 r = p;
29             }
30         }
31
32         return r;
33     }
34 }
```

Analysis summary

The following issues have been detected: wrong answers, timeout errors.

For example, for the input [3, 1, 2, 4, 3] the solution returned a wrong answer (got 3 expected 1).

Analysis ?

expand all		Example tests
▶	example example test	✗ WRONG ANSWER got 3 expected 1
expand all		Correctness tests
▶	double two elements	✗ WRONG ANSWER got 1 expected 2000
▶	simple_positive simple test with positive numbers, length = 5	✗ WRONG ANSWER got 2 expected 4
▶	simple_negative simple test with negative numbers, length = 5	✗ WRONG ANSWER got 1 expected 0
▶	simple_boundary only one element on one of the sides	✗ WRONG ANSWER got 2 expected 1
▶	small_random random small, length = 100	✗ WRONG ANSWER got 69 expected 39
▶	small_range range sequence, length = ~1,000	✗ WRONG ANSWER got 706 expected 56
▶	small small elements	✗ WRONG ANSWER got 4 expected 20
expand all		Performance tests
▶	medium_random1 random medium, numbers from 0 to 100, length = ~10,000	✗ TIMEOUT ERROR running time: 0.548 sec., time limit: 0.100 sec.
▶	medium_random2 random medium, numbers from -1,000 to	✗ TIMEOUT ERROR running time: 0.536 sec.,

Test results - Codility

50, length = ~10,000		time limit: 0.112 sec.
▶ large_ones	large sequence, numbers from -1 to 1, length = ~100,000	✗ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
▶ large_random	random large, length = ~100,000	✗ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
▶ large_sequence	large sequence, length = ~100,000	✗ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
▶ large_extreme	large test with maximal and minimal values, length = ~100,000	✗ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.

PDF version of this report that may be downloaded on top of this site may contain sensitive data including personal information. For security purposes, we recommend you remove it from your system once reviewed.