

Test Name:

Summary Timeline

Tasks summary

| Task | Time spent | Score |
|----------------------------------|------------|-------|
| TapeEquilibrium Java 8 | 52 min | 100% |

Total score

100%

Tasks Details

Easy

1. TapeEquilibrium
Minimize the value $|(A[0] + \dots + A[P-1]) - (A[P] + \dots + A[N-1])|$.

Task Score

100%

Correctness

100%

Performance

100%

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 = $|10 - 3| = 7$

Solution

Programming language used:

Java 8

Total time used:

52 minutes

?

Effective time used:

52 minutes

?

Notes:

not defined yet

Task timeline

22:47:38

23:38:54

Code: 23:38:54 UTC, java, final,
score: 100

[show code in pop-up](#)

1 // you can also use imports, for example:
2 // import java.util.*;
3

Write a function:

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

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].

```
4 // you can write to stdout for debugging purposes, e.g.
5 // System.out.println("this is a debug message");
6
7 class Solution {
8     public int solution(int[] A) {
9         // write your code in Java SE 8
10
11         int allSum = 0;
12         int leftArraySum = 0;
13         int rightArraySum = 0;
14         int minDiff = Integer.MAX_VALUE;
15
16         for(int i=0; i < A.length; i++){
17             allSum += A[i];
18         }
19
20         for(int i=0; i<A.length-1; i++){
21             leftArraySum += A[i];
22             rightArraySum = allSum - leftArraySum;
23             int currentDiff = Math.abs(rightArraySum -
24                                     leftArraySum);
25             minDiff = Math.min(currentDiff, minDiff);
26         }
27         return minDiff;
28     }
29 }
```

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 two elements | ✓ OK |
| ▶ simple_positive simple test with positive numbers, length = 5 | ✓ OK |
| ▶ simple_negative simple test with negative numbers, length = 5 | ✓ OK |
| ▶ simple_boundary only one element on one of the sides | ✓ OK |
| ▶ small_random random small, length = 100 | ✓ OK |
| ▶ small_range range sequence, length = ~1,000 | ✓ OK |
| ▶ small small elements | ✓ OK |
| expand all | Performance tests |
| ▶ medium_random1 random medium, numbers from 0 to 100, length = ~10,000 | ✓ OK |
| ▶ medium_random2 random medium, numbers from -1,000 to 50, length = ~10,000 | ✓ OK |
| ▶ | |

| | | |
|---|--|------|
| | large_ones | ✓ OK |
| | large sequence, numbers from -1 to 1, length = ~100,000 | |
| ▶ | large_random | ✓ OK |
| | random large, length = ~100,000 | |
| ▶ | large_sequence | ✓ OK |
| | large sequence, length = ~100,000 | |
| ▶ | large_extreme | ✓ OK |
| | large test with maximal and minimal values, length = ~100,000 | |

The 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.