

AVAILABLE
LESSONS:*Lesson 1*
Iterations*Lesson 2*
Arrays*Lesson 3*
**Time
Complexity***Lesson 4*
Counting
Elements*Lesson 5*
Prefix Sums*Lesson 6*
Sorting*Lesson 7*
Stacks and
Queues*Lesson 8*
Leader*Lesson 9*
Maximum slice
problem*Lesson 10*
Prime and
composite
numbers

PAINLESS

TapeEquilibrium

START

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

Programming language: C++ ▼

A non-empty zero-indexed 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:

$$\begin{aligned} A[0] &= 3 \\ A[1] &= 1 \\ A[2] &= 2 \\ A[3] &= 4 \\ A[4] &= 3 \end{aligned}$$

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:

```
int solution(vector<int> &A);
```

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

For example, given:

$$A[0] = 3$$

Lesson 11

Sieve of
Eratosthenes

Lesson 12

Euclidean
algorithm

Lesson 13

Fibonacci
numbers

Lesson 14

Binary search
algorithm

Lesson 15

Caterpillar
method

Lesson 16

Greedy
algorithms

Lesson 17

Dynamic
programming

Lesson 90

Tasks from
Indeed Prime
2016 challenge

Lesson 99

Future training

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

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 [-1,000..1,000].

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).

Elements of input arrays can be modified.

Copyright 2009–2016 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

**For
programmers**

Lessons Challenges
Terms FAQ

For companies

Tour About
Pricing us
Blog Jobs
Privacy Terms
API Cookies

**Sign up for our
newsletter:**

Information about
upcoming challenges,
solutions and lessons
directly in your inbox.

Social:

[f](#) [t](#) [in](#)

Contact us:

For customer support
queries:

UK +44 (0) 208 970
78 68

US 1-415-466-8085
support@codility.com

For sales queries:

UK +44 (0) 208 970
78 67

US 1-415-466-8085
sales@codility.com

© 2009–2016 Codility Ltd., registered in England and Wales (No. 7048726). VAT ID
GB981191408. Registered office: 107 Cheapside, London EC2V 6DN