

CodeCheck Report: trainingG9D2FH-952

Test Name:

[Check out Codility training tasks](#)

Summary

Timeline

Tasks summary

Task		Time spent	Score
TapeEquilibrium	⚠	7 min	100%
Python			

Total score

100%

Tasks Details

1.	TapeEquilibrium				
Easy	Minimize the value $ (A[0] + \dots + A[P-1]) - (A[P] + \dots + A[N-1]) $.	Task Score	Correctness	Performance	
		100%	100%	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:

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

Solution

Programming language used: Python

Total time used: 7 minutes ?

Effective time used: 7 minutes ?

Notes: not defined yet

Task timeline ?



$A[4] = 3$

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:

```
def solution(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 $[-1,000..1,000]$.

Copyright 2009–2023 by Codility Limited. All Rights Reserved.

Unauthorized copying, publication or disclosure prohibited.



10:56:52

11:03:37

Code: 11:03:37 UTC, py,

[show code in pop-up](#)

final, score: 100

```
1 # you can write to stdout for debugging p
2 # print("this is a debug message")
3
4 def solution(A):
5     # Implement your solution here
6     # pass
7     length_array = len(A)
8     total_sum = sum(A)
9     left_sum = 0
10    minimum = float('inf')
11
12    for i in range(length_array - 1):
13        left_sum += A[i]
14        right_sum = total_sum - left_sum
15        difference = abs(left_sum - right_sum)
16        minimum = min(minimum, difference)
17
18    return minimum
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all	Example tests
▶ example	✓ OK
example test	
expand all	Correctness tests
▶ double	✓ OK
two elements	
▶ simple_positive	✓ OK
simple test with positive numbers, length = 5	
▶ simple_negative	✓ OK
simple test with negative numbers, length = 5	
▶ simple_boundary	✓ OK
only one element on one of the sides	
▶ small_random	✓ OK
random small, length = 100	
▶ small_range	✓ OK
range sequence, length = ~1,000	
▶ small	✓ OK
small elements	

Performance tests	
▶ medium_random1	✓ OK
random medium, numbers from 0 to 100, length = ~10,000	
▶ medium_random2	✓ OK
random medium, numbers from -1,000 to 50, length = ~10,000	
▶ 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	