Codility_

Candidate Report: trainingXRA2AY-ZNW

Check out Codility training tasks

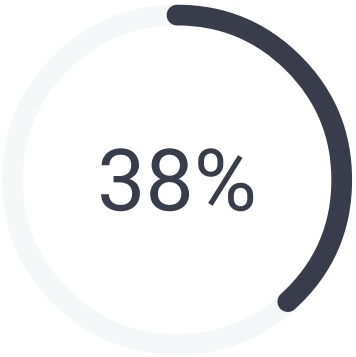
Test Name:

Summary Timeline

Tasks summary

Task	Time spent	Score
TapeEquilibrium Python	9 min	38%

Total score



Tasks Details

Easy

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

Task Score	Correctness	Performance
38%	71%	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], ..., A[P − 1] and A[P], A[P + 1], ..., A[N − 1].

The *difference* between the two parts is the value of: |(A[0] + A[1] + ... + A[P − 1]) − (A[P] + A[P + 1] + ... + 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:

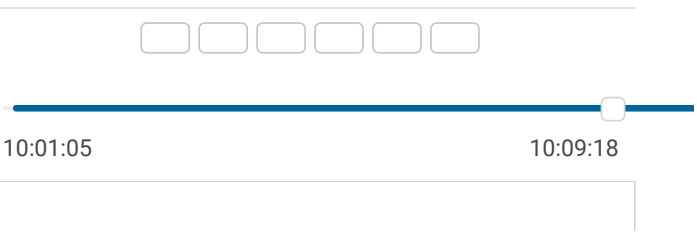
```
A[0] = 3
A[1] = 1
A[2] = 2
A[3] = 4
A[4] = 3
```

We can split this tape in four places:

Solution

Programming language used:	Python
Total time used:	9 minutes
Effective time used:	9 minutes
Notes:	*not defined yet*

Task timeline



- 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].

Code: 10:09:18 UTC, py, final, score: **38** show code in pop-up

```
1  # you can write to stdout for debugging purposes, e.g.
2  # print("this is a debug message")
3  import math
4  def solution(A):
5      # write your code in Python 3.6
6      L = len(A)
7      mi = math.inf
8      i = 1
9      while i<L-1:
10         first = A[:i]
11         second = A[i:]
12         s1 = sum(first)
13         s2 = sum(second)
14         dif = abs(s1-s2)
15         if dif<mi:
16             mi = dif
17         i = i + 1
18     return mi
```

Analysis summary

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

For example, for the input `[1, 1]` the solution terminated unexpectedly.

Analysis

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

expand all	Example tests	
▶	example example test	✓ OK
expand all	Correctness tests	
▶	double two elements	✗ RUNTIME ERROR tested program terminated with exit code 1
▶	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	✗ WRONG ANSWER got 3 expected 1
▶	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	✗ **TIMEOUT ERROR** running time: 1.208 sec., time limit: 0.208 sec.
▶	medium_random2 random medium, numbers from -1,000 to 50, length = ~10,000	✗ **TIMEOUT ERROR** running time: 1.492 sec., time limit: 0.208 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.