codility

Training center

Check out Codility training tasks

TASKS DETAILS

EASY

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

Task Score	Correctness	Performance
50%	100%	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:

- 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
```

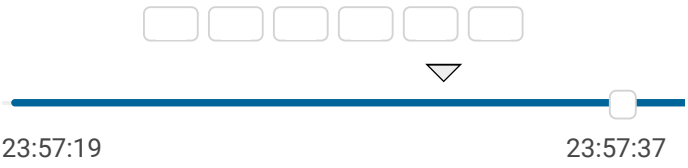
the function should return 1, as explained above.

Assume that:

Solution

Programming language used:	Python
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	*not defined yet*

Task timeline



23:57:19 23:57:37

Code: 23:57:36 UTC, py, final, score: **50**

show code in pop-up

```
1  # Solution 1
2  def solution(A):
3      length = len(A)
4      minimal = None
5      for i in range(1, length):
6          distance = abs(sum(A[0:i]) - sum(A[i-
7          if minimal is None or distance < minir
8              minimal = distance
9      return minimal
```

Analysis summary

The following issues have been detected: timeout errors.

Analysis

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

- 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) (not counting the storage required for input arguments).

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
▶ 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: 1.18 sec., time limit: 0.21 sec.
▶ medium_random2 random medium, numbers from -1,000 to 50, length = ~10,000	✘ TIMEOUT ERROR running time: 1.45 sec., time limit: 0.21 sec.
▶ large_ones large sequence, numbers from -1 to 1, length = ~100,000	✘ TIMEOUT ERROR running time: >6.00 sec., time limit: 0.67 sec.
▶ large_random random large, length = ~100,000	✘ TIMEOUT ERROR running time: >6.00 sec., time limit: 0.72 sec.
▶ large_sequence large sequence, length = ~100,000	✘ TIMEOUT ERROR running time: >6.00 sec., time limit: 0.43 sec.
▶ large_extreme large test with maximal and minimal values, length = ~100,000	✘ TIMEOUT ERROR running time: >6.00 sec., time limit: 0.74 sec.