Codility_

Screen Report: Anonymous

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Effective time spent	Score
TapeEquilibrium ⚠ C++	4 min	100%

Total score

100%

Tasks Details

Easy

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

Solution

Programming language used:	C++
Time spent on task:	4 minutes
Notes:	*not defined yet*

Task timeline

09:27:50 09:30:57

Code: 09:30:56 UTC, cpp, final, score: **100**

show code in pop-up

```cpp
#include <numeric>

using namespace std;

int solution(vector<int> &A) {
    // Implement your solution here

    // we need two for loops to get the full solution
    // First one to get the full sum
```

```
int solution(vector<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].

Copyright 2009–2024 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```
    long long int totalSum{accumulate(A.begin(), A.end(),
    long long int rightSum{A[0]};
    long long int minDiff{abs(totalSum - (2 * rightSum))};

    for (unsigned int i = 1; i < A.size()-1; i++)
    {
        rightSum += A[i];
        minDiff = min(minDiff, abs(totalSum - (2 * rightSu
    }

    return minDiff;
}
```

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