closed

Demo ticket

Session
ID: demoA237Q3-NJC
Time limit: 120 min.

Status: closed
Started on: 2014-01-05 06:56 UTC

Score:

100

of 100

1. TapeEquilibrium

score: 100 of 100

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

Task description

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], ..., 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:

```
class Solution { public int solution(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
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.

Solution

Programming language used: C#

Total time used: 2 minutes (?)

Effective time used: 1 minutes (?)

Notes: correct functionality and scalability

Task timeline What is it?

06:56:43 06:58:01

Code: 06:58:01 UTC, cs, final, score: 100.00

```
using System;
// you can also use other imports, for example:
// using System.Collections.Generic;
class Solution {
        public int solution(int[] A)
        {
            if (A == null) throw new ArgumentNullException();
            if (A.Length < 2 || A.Length > 100000) throw new ArgumentOutOfRangeException();
            // write your code in C# with .NET 2.0
            long sum = 0L;
            for(var index = 0; index < A.Length; index++) {
             sum += A[index];
            }

            long minimumDifference = long.MaxValue;
            long rightSum = sum;
            long leftSum = 0L;

            for (var index = 0; index < A.Length - 1; index++)
            {
             leftSum += A[index];
             rightSum -= A[index];
             long difference = rightSum - leftSum;
             difference = difference > 0 ? difference : -difference;

```

```
                if (minimumDifference >
                    difference)
                    minimumDifference =
                    difference;
            }
            return (int)minimumDifference;
        }

}
```

Analysis

Detected time complexity:

O(N)

test	time	result
example example test	0.080 s.	**OK**
double two elements	0.080 s.	**OK**
simple_positive simple test with positive numbers, length = 5	0.080 s.	**OK**
simple_negative simple test with negative numbers, length = 5	0.080 s.	**OK**
small_random random small, length = 100	0.080 s.	**OK**
small_range range sequence, length = ~1,000	0.080 s.	**OK**
small small elements	0.080 s.	**OK**
medium_random1 random medium, numbers from 0 to 100, length = ~10,000	0.080 s.	**OK**
medium_random2 random medium, numbers from -1,000 to 50, length = ~10,000	0.080 s.	**OK**
large_ones large sequence, numbers from -1 to 1, length = ~100,000	0.100 s.	**OK**
large_random random large, length = ~100,000	0.100 s.	**OK**
large_sequence large sequence, length = ~100,000	0.090 s.	**OK**
large_extreme large test with maximal and minimal values, length = ~100,000	0.100 s.	**OK**

Get acco