

## Demo ticket

### Session

ID: demoKJ8RBQ-4WA  
Time limit: 120 min.

### Status: closed

Created on: 2014-03-16 01:01 UTC  
Started on: 2014-03-16 01:02 UTC  
Finished on: 2014-03-16 02:02 UTC

## Tasks in test

## Task score

## Test score

100%

100 out of 100 points

EASY

### 1. TapeEquilibrium

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

score: 100 of 100



#### 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:  $A[0], A[1], \dots, A[P-1]$  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
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:

```
int solution(vector<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:

#### Solution

Programming language used: C++

Total time used: 60 minutes

Effective time used: 60 minutes

Notes: correct functionality and scalability

#### Task timeline



01:02:46

02:02:40

Code: 02:02:40 UTC, cpp, final, score: 100.00

```
01. #include <cstdlib>
02.
03. int solution(vector<int> &A) {
04.     long sumleft = 0;
05.     long sumright = 0;
06.     long localmin;
07.     long ans;
08.
09.     // start from 2nd element
10.     for (int i = 1; i < (int)A.size(); i++) {
11.         sumright += A[i];
12.     }
13.
14.     sumleft = A[0];
15.     ans = abs(sumright - sumleft);
16.
17.     for (int i = 1; i < (int)(A.size() - 1); i++) {
18.         sumleft += A[i];
19.         sumright -= A[i];
20.         localmin = abs(sumleft - sumright);
```

- 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.

Copyright 2009–2014 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```
21.         if (localmin < ans) {
22.             ans = localmin;
23.         }
24.     }
25.
26.     return ans;
27. }
```

Analysis



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

test	time	result
example example test	0.020 s.	OK
double two elements	0.020 s.	OK
simple_positive simple test with positive numbers, length = 5	0.020 s.	OK
simple_negative simple test with negative numbers, length = 5	0.020 s.	OK
small_random random small, length = 100	0.020 s.	OK
small_range range sequence, length = ~1,000	0.020 s.	OK
small small elements	0.020 s.	OK
medium_random1 random medium, numbers from 0 to 100, length = ~10,000	0.020 s.	OK
medium_random2 random medium, numbers from -1,000 to 50, length = ~10,000	0.020 s.	OK
large_ones large sequence, numbers from -1 to 1, length = ~100,000	0.030 s.	OK
large_random random large, length = ~100,000	0.040 s.	OK
large_sequence large sequence, length = ~100,000	0.030 s.	OK
large_extreme large test with maximal and minimal values, length = ~100,000	0.030 s.	OK

Training center