Open Feedback Dialog

codility

Training center

Check out Codility training tasks

Demo ticket

Session

ID: demo69MYQS-5TH
Time limit: 120 min.

Status: closed

Created on: 2014-03-15 03:25 UTC
Started on: 2014-03-15 03:25 UTC
Finished on: 2014-03-15 03:37 UTC

Tasks in test

Task score

Test score

100%

100 out of 100 points

EASY

1. TapeEquilibrium

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

Solution

Programming language used: Python

Total time used: 12 minutes

Effective time used: 12 minutes

Notes: correct functionality and scalability

Task timeline

03:25:35 03:37:32

Code: 03:37:32 UTC, py, final, score: **100.00**

```
01. def solution(A):
02.     # write your code in Python 2.6
03.     size = len(A)
04.     front = [0]*size
05.     for i in xrange(size):
06.         if i == 0:
07.             front[i] = A[i]
08.         else:
09.             front[i] = front[i-1] + A[i]
10.
11.     back = [0]*size
12.     for i in xrange(size-1, -1, -1):
13.         if i == size-1:
14.             back[i] = A[i]
15.         else:
16.             back[i] = back[i+1] + A[i]
17.
18.     min_diff = 1 << 32 # large number
19.     for i in xrange(1, size):
20.         min_diff = min(min_diff, abs(back[i] - front[i-1]))
21.     return min_diff
```

- 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.

Analysis

Detected time complexity:
O(N)

test	time	result
example example test	0.050 s.	**OK**
double two elements	0.050 s.	**OK**
simple_positive simple test with positive numbers, length = 5	0.050 s.	**OK**
simple_negative simple test with negative numbers, length = 5	0.050 s.	**OK**
small_random random small, length = 100	0.050 s.	**OK**
small_range range sequence, length = ~1,000	0.050 s.	**OK**
small small elements	0.050 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	0.310 s.	**OK**
large_random random large, length = ~100,000	0.320 s.	**OK**
large_sequence large sequence, length = ~100,000	0.180 s.	**OK**
large_extreme large test with maximal and minimal values, length = ~100,000	0.270 s.	**OK**

Training center