codility

Training center

Check out Codility training tasks

Congratulations

You have completed a Codility demo.

Return to proper test

Eugene Mukhanov

Session

ID: demoDN9672-JAQ
Time limit: 30 min.

Status: closed

Created on: 2016-06-11 22:12 UTC
Started on: 2016-06-11 22:12 UTC
Finished on: 2016-06-11 22:20 UTC

Tasks in test	Correctness	Performance	Task score
1 | Equi Submitted in: Java	100%	100%	100%

Test score

100%

100 out of 100 points

MEDIUM

1. Equi

Find an index in an array such that its prefix sum equals its suffix sum.

score: 100 of 100

Task description

This is a demo task. You can read about this task and its solutions in this blog post.

A zero-indexed array A consisting of N integers is given. An *equilibrium index* of this array is any integer P such that 0 ≤ P < N and the sum of elements of lower indices is equal to the sum of elements of higher indices, i.e.

A[0] + A[1] + ... + A[P−1] = A[P+1] + ... + A[N−2] + A[N−1].

Sum of zero elements is assumed to be equal to 0. This can happen if P = 0 or if P = N−1.

For example, consider the following array A consisting of N = 8 elements:

```
A[0] = -1
A[1] =  3
A[2] = -4
A[3] =  5
A[4] =  1
A[5] = -6
A[6] =  2
A[7] =  1
```

P = 1 is an equilibrium index of this array, because:

- A[0] = −1 = A[2] + A[3] + A[4] + A[5] + A[6] + A[7]

P = 3 is an equilibrium index of this array, because:

- A[0] + A[1] + A[2] = −2 = A[4] + A[5] + A[6] + A[7]

P = 7 is also an equilibrium index, because:

- A[0] + A[1] + A[2] + A[3] + A[4] + A[5] + A[6] = 0

Solution

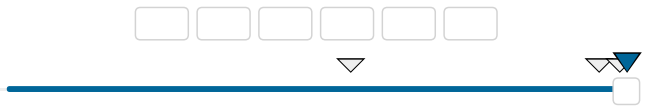
Programming language used: Java

Total time used: 8 minutes

Effective time used: 8 minutes

Notes: *not defined yet*

Task timeline



22:12:31 22:20:12

Code: 22:20:12 UTC, java, final, score: **100**

show code in pop-up

```
1  // you can also use imports, for example:
2  import java.math.BigInteger;
3
4  // you can write to stdout for debugging purposes, e.g.
5  // System.out.println("this is a debug message");
6
7  class Solution {
8    public int solution(int[] A) {
9      BigInteger sum = BigInteger.ZERO;
10     for (int i=0; i<A.length; i++) {
11       sum = sum.add(BigInteger.valueOf(A[i]));
12     }
13     BigInteger mus = BigInteger.ZERO;
14     for (int i=0; i<A.length; i++) {
15       sum = sum.subtract(BigInteger.valueOf(A[i]));
```

and there are no elements with indices greater than 7.

P = 8 is not an equilibrium index, because it does not fulfill the condition 0 ≤ P < N.

Write a function:

```
class Solution { public int solution(int[] A); }
```

that, given a zero-indexed array A consisting of N integers, returns any of its equilibrium indices. The function should return −1 if no equilibrium index exists.

For example, given array A shown above, the function may return 1, 3 or 7, as explained above.

Assume that:

- N is an integer within the range [0..100,000];
- each element of array A is an integer within the range [−2,147,483,648..2,147,483,647].

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.


```
16        if (sum.equals(mus)) {
17          return i;
18        }
19        mus = mus.add(BigInteger.valueOf(A[i]));
20      }
21      return -1;
22    }
23  }
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity:
O(N)

expand all	Example tests	
▶	example Test from the task description	✔ OK
expand all	Correctness tests	
▶	simple	✔ OK
▶	extreme_large_numbers Sequence with extremely large numbers testing arithmetic overflow.	✔ OK
▶	extreme_negative_numbers Sequence with extremely large numbers testing arithmetic overflow.	✔ OK
▶	overflow_tests1 arithmetic overflow tests	✔ OK
▶	overflow_tests2 arithmetic overflow tests	✔ OK
▶	one_large one large number at the end of the sequence	✔ OK
▶	sum_0 sequence with sum=0	✔ OK
▶	single_empty single number or empty array	✔ OK
▶	combinations_of_two multiple runs, all pairs of values: -1, 0 and 1	✔ OK
▶	combinations_of_three multiple runs, all triples of values -1, 0 and 1	✔ OK
▶	small_pyramid	✔ OK
expand all	Correctness/performance tests	
▶	extreme_max Maximal size test	✔ OK
expand all	Performance tests	
▶	large_long_sequence_of_ones	✔ OK
▶	large_long_sequence_of_minus_ones	✔ OK
▶	medium_pyramid	✔ OK
▶	large_pyramid Large performance test, O(n^2) solutions should fail.	✔ OK
▶	huge_pyramid Large performance test, O(n^2) solutions should fail.	✔ OK