Demo ticket

demoGPJC9H-6AJ
Started on: 2013-01-04 21:02 UTC
Status: closed
Time limit: 30 min.

Score:
100
of 100

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 = 7 elements:

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

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

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

P = 6 is also an equilibrium index, because:

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

and there are no elements with indices greater than 6.
P = 7 is not an equilibrium index, because it does not fulfill the condition 0 ≤ P < N.
Write a function

```
class Solution { public int equi(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.
Assume that:

- N is an integer within the range [0..10,000,000];
- each element of array A is an integer within the range [−2,147,483,648..2,147,483,647].

For example, given array A such that

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

the function may return 3 or 6, as explained above.
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

Time used: 1 min.

Notes: correct functionality and scalability

Task timeline

Use the controls below to see how the code changed during the test.

21:02:43 — 21:02:54

Code: 21:02:54 UTC, java, final, score: 100.00

```
01. public class Solution
02. {
03.     public int equi( int[] A ) {
04.         long sum = 0;
05.         long leftSum = 0;
06.
07.         for (int i=0;i<A.length;i++){
08.         sum += A[i]; //get sum of entire
               array
09.         }
10.         for (int i=0;i<A.length;i++){
11.         if (leftSum==sum-A[i]) return i;
12.         leftSum+=A[i];
13.         sum-=A[i];
14.         }
15.         return -1;
16.
17.         }
18. }
19.
```

Analysis

Detected time complexity:
O(N)

test	time	result
example Test from the task description	0.260 s.	**OK**
simple	0.240 s.	**OK**
extreme_large_numbers Sequence with extremly large numbers testing arithmetic overflow.	0.240 s.	**OK**
overflow_tests	0.240 s.	**OK**
one_large one large number at the end of the sequence	0.240 s.	**OK**
sum_0 sequence with sum=0	0.240 s.	**OK**

single single number	0.230 s.	**OK**
empty Empty array	0.270 s.	**OK**
combinations_of_two multiple runs, all combinations of {-1,0,1}^2	0.260 s.	**OK**
combinations_of_three multiple runs, all combinations of {-1,0,1}^3	0.240 s.	**OK**
small_pyramid	0.240 s.	**OK**
large_long_sequence_of_ones	0.270 s.	**OK**
large_long_sequence_of_minus_ones	0.270 s.	**OK**
medium_pyramid	0.280 s.	**OK**
large_pyramid Large performance test, O(n^2) solutions should fail.	0.300 s.	**OK**