codility

Training center
Check out Codility training tasks

Congratulations

You have completed a Codility demo.

Tweet this!
I scored 76% in #objc on @Codility!
https://codility.com/demo/take-sample-test/

Sign up for our newsletter!

Like us on Facebook!

Demo ticket

Session

ID: demoKGWYSP-RBA
Time limit: 30 min.

Status: closed

Created on: 2015-12-30 21:14 UTC
Started on: 2015-12-30 21:14 UTC
Finished on: 2015-12-30 21:31 UTC

Tasks in test	Correctness	Performance	Task score
1 Equi Submitted in: Objective-C	95%	36%	76%

Test score

76%

76 out of 100 points

MEDIUM

1. Equi

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

score: 76 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:

Solution

Programming language used: Objective-C

Total time used: 18 minutes

Effective time used: 18 minutes

Notes: *not defined yet*

Task timeline

21:14:05 21:31:40

Code: 21:31:40 UTC, m, final, score: **76.00** show code in pop-up

```
1  // you can also use imports, for example:
2  // #import <Foundation/NSDictionary.h>
3
4  // you can write to stdout for debugging purposes, e.g.
5  // printf("this is a debug message\n");
6
7  int solution(NSMutableArray *A) {
```

- 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

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:

```
int solution(NSMutableArray *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.


```
 8      // write your code in Objective-C 2.0
 9          //NSLog(@"%@",A);
10      NSInteger arrayLength = A.count;
11      for(NSInteger i=0;i<arrayLength;i++){
12
13          NSMutableArray *rightArray = [NSMutableArray ne
14          NSMutableArray *leftArray = [NSMutableArray new
15
16          NSInteger leftsum=0;
17          NSInteger rightsum=0;
18
19          for (NSInteger j=0; j<i; j++) {
20              NSNumber *num =  A[j];
21              [leftArray addObject:num];
22
23              leftsum +=num.integerValue;
24          }
25
26
27          for (NSInteger k=i+1; k<arrayLength; k++) {
28               NSNumber *num =  A[k];
29               [rightArray addObject:num];
30
31              rightsum +=num.integerValue;
32          }
33
34          //NSLog(@"leftSum[%@] = %ld  -- rightsum:[%@] =
35          if (rightsum == leftsum) {
36              return (int)i;
37          }
38
39      }
40      return -1;
41  }
```

Analysis summary

The following issues have been detected: timeout errors.

Analysis

Detected time complexity:

O(N**2)

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	✘ TIMEOUT ERROR

	Maximal size test	running time: 4.68 sec., time limit: 0.33 sec.

expand all **Performance tests**

▶	large_long_sequence_of_ones	✘ **TIMEOUT ERROR** running time: 4.47 sec., time limit: 0.32 sec.
▶	large_long_sequence_of_minus_ones	✘ **TIMEOUT ERROR** running time: 4.47 sec., time limit: 0.33 sec.
▶	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.	✘ **TIMEOUT ERROR** running time: 3.21 sec., time limit: 0.36 sec.

Training center