



Training center

Check out Codility training tasks

## Congratulations

You have completed a Codility demo.

## Tweet this!

I scored 100% in #objc on @Codility!  
<https://codility.com/demo/take-sample-test/>

Sign up for our newsletter!

Like us on Facebook!

## Demo ticket

## Session

ID: demoKC38YM-P5J  
Time limit: 30 min.

## Status: closed

Created on: 2015-12-30 22:59 UTC  
Started on: 2015-12-30 22:59 UTC  
Finished on: 2015-12-30 23:24 UTC

## Tasks in test

1 Equi  
Submitted in: Objective-C

## Correctness

100%

## Performance

100%

## Task score

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  $0 \leq P < N$  and the sum of elements of lower indices is equal to the sum of elements of higher indices, i.e.

$$A[0] + A[1] + \dots + A[P-1] = A[P+1] + \dots + A[N-2] + A[N-1].$$

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

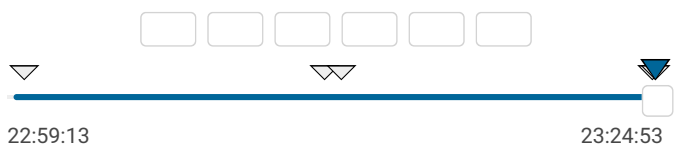
Programming language used: Objective-C

Total time used: 26 minutes

Effective time used: 26 minutes

Notes: not defined yet

## Task timeline

Code: 23:24:52 UTC, m, final,  
score: 100.00[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(NSArray *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  $0 \leq P < N$ .

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.

Copyright 2009–2015 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```

8 // write your code in Objective-C 2.0
9
10 if (A.count==0) {
11     return -1;
12 }
13 NSNumber *sum = [NSNumber numberWithInt:0];
14 for (NSInteger i=0; i<A.count; i++) {
15     sum = @(sum.integerValue + ((NSNumber *)A[i]).i
16 }
17
18 NSNumber *leftSum = [NSNumber numberWithInt:0];
19 NSNumber *rightSum = [NSNumber numberWithInt:0];
20
21 for (NSInteger i=0; i<A.count; i++) {
22     rightSum = @(sum.integerValue -leftSum.integerV
23     if ([rightSum isEqualToNumber:leftSum]) {
24         return (int)i;
25     }
26     leftSum = @(leftSum.integerValue +((NSNumber *)
27 }
28
29
30 return -1;
31
32
33 }
```

### Analysis summary

The solution obtained perfect score.

### Analysis

Detected time complexity:

**$O(N)$**

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

Large performance test,  $O(n^2)$  solutions  
should fail.



huge\_pyramid

✓ OK

Large performance test,  $O(n^2)$  solutions  
should fail.

Training center