Codility_

Candidate Report: Anonymous

Check out Codility training tasks

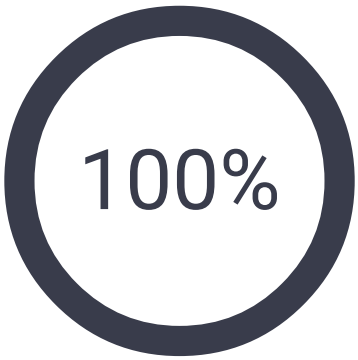
Test Name:

Summary Timeline Feedback

Tasks summary

Task	Time spent	Score
MaxSliceSum Swift 4	1 min	100%

Total score



Tasks Details

Easy

1. MaxSliceSum

Find a maximum sum of a compact subsequence of array elements.

Task Score	Correctness	Performance
100%	100%	100%

Task description

A non-empty array A consisting of N integers is given. A pair of integers (P, Q), such that 0 ≤ P ≤ Q < N, is called a *slice* of array A. The *sum* of a slice (P, Q) is the total of A[P] + A[P+1] + ... + A[Q].

Write a function:

```
public func solution(_ A : inout [Int]) -> Int
```

that, given an array A consisting of N integers, returns the maximum sum of any slice of A.

For example, given array A such that:

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

the function should return 5 because:

- (3, 4) is a slice of A that has sum 4,
- (2, 2) is a slice of A that has sum −6,
- (0, 1) is a slice of A that has sum 5,
- no other slice of A has sum greater than (0, 1).

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..1,000,000];
- each element of array A is an integer within the range [−1,000,000..1,000,000];
- the result will be an integer within the range [−2,147,483,648..2,147,483,647].

Solution

Programming language used:	Swift 4
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	*not defined yet*

Task timeline

19:00:17 19:00:39

Code: 19:00:38 UTC, swift4, final, score: **100**

show code in pop-up

```
1  import Foundation
2  import Glibc
3
4  public func solution(_ A : inout [Int]) -> Int {
5
6      var maxEnding = 0
```

```
7       var maxValue = Int.min
8       for item in A{
9           maxEnding = maxEnding + item
10          if maxValue < maxEnding {
11              maxValue = maxEnding
12          }
13          if maxEnding < 0 {
14              maxEnding = 0
15          }
16      }
17      return maxValue
18  }
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(N)**

expand all	Example tests	
▶	example	✔ OK
expand all	Correctness tests	
▶	one_element	✔ OK
▶	two_elements	✔ OK
▶	three_elements	✔ OK
▶	simple	✔ OK
▶	extreme_minimum	✔ OK
▶	fifty_random	✔ OK
▶	neg_const	✔ OK
▶	pos_const	✔ OK
expand all	Performance tests	
▶	high_low_1Kgarbage	✔ OK
▶	1Kgarbage_high_low	✔ OK
▶	growing_saw	✔ OK
▶	blocks	✔ OK
▶	growing_negative	✔ OK