Codility_

Candidate Report: Anonymous

Check out Codility training tasks

Test Name:

Summary Timeline Feedback

Tasks summary

Task	Time spent	Score
MaxDoubleSliceSum Swift 4	1 min	100%

Total score

100%

Tasks Details

Medium

1. **MaxDoubleSliceSum**
Find the maximal sum of any double slice.

Task Score	Correctness	Performance
100%	100%	100%

Task description

A non-empty array A consisting of N integers is given.

A triplet (X, Y, Z), such that 0 ≤ X < Y < Z < N, is called a *double slice*.

The *sum* of double slice (X, Y, Z) is the total of A[X + 1] + A[X + 2] + ... + A[Y − 1] + A[Y + 1] + A[Y + 2] + ... + A[Z − 1].

For example, array A such that:

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

contains the following example double slices:

- double slice (0, 3, 6), sum is 2 + 6 + 4 + 5 = 17,
- double slice (0, 3, 7), sum is 2 + 6 + 4 + 5 − 1 = 16,
- double slice (3, 4, 5), sum is 0.

The goal is to find the maximal sum of any double slice.

Write a function:

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

that, given a non-empty array A consisting of N integers, returns the maximal sum of any double slice.

For example, given:

Solution

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

Task timeline

21:35:29 21:35:55

Code: 21:35:55 UTC, swift4, final, score: **100**

show code in pop-up

```
import Foundation
import Glibc

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

    let n = A.count
    if n == 3 { return 0 }
    var maxEndingUntilIndex = Array(repeating: 0, count
    var maxEndingFromIndex = Array(repeating: 0, count
```

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

the function should return 17, because no double slice of array A has a sum of greater than 17.

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

- N is an integer within the range [3..100,000];
- each element of array A is an integer within the range [−10,000..10,000].


```
    var maxResult = 0

    for i in (1...n-2) {
        maxEndingUntilIndex[i] = max(0, maxEndingUntil
    }

    for i in (1...n-2).reversed() {
        maxEndingFromIndex[i] = max(0, maxEndingFromIn
    }

    var sum = 0
    for i in 0...n-3 {
        sum = maxEndingUntilIndex[i] + maxEndingFromIn
        maxResult = max(maxResult, sum)
    }

    return maxResult
}
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all	Example tests	
▶	example example test	✔ OK
expand all	Correctness tests	
▶	simple1 first simple test	✔ OK
▶	simple2 second simple test	✔ OK
▶	simple3 third simple test	✔ OK
▶	negative all negative numbers	✔ OK
▶	positive all positive numbers	✔ OK
▶	extreme_triplet three elements	✔ OK
expand all	Performance tests	
▶	small_random1 random, numbers form -10**4 to 10**4, length = 70	✔ OK
▶	small_random2 random, numbers from -30 to 30, length = 300	✔ OK
▶	medium_range -1000, ..., 1000	✔ OK
▶	large_ones random numbers from -1 to 1, length = ~100,000	✔ OK
▶	large_random random, length = ~100,000	✔ OK
▶	extreme_maximal all maximal values, length = ~100,000	✔ OK
▶	large_sequence many the same small sequences, length = ~100,000	✔ OK