

Candidate Report: Anonymous

[Check out Codility training tasks](#)

Test Name:

SummaryTimelineFeedback

Tasks summary

Total score

Task	Time spent	Score
MaxDoubleSliceSum Swift 4	1 min	100%



Tasks Details

Medium	1. MaxDoubleSliceSum Find the maximal sum of any double slice.	Task Score	Correctness	Performance
			100%	100%

Task description

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

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

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

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.

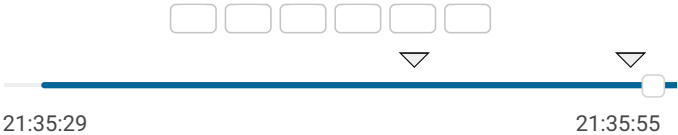
For example, given:

<https://app.codility.com/demo/results/trainingFZ9UVT-6S8/>

Solution

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

Task timeline



Code: 21:35:55 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     let n = A.count
7     if n == 3 { return 0 }
8     var maxEndingUntilIndex = Array(repeating: 0, count: n)
9     var maxEndingFromIndex = Array(repeating: 0, count: n)
```

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].

Copyright 2009–2020 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Test results - Codility

```
10     var maxResult = 0
11
12     for i in (1..n-2) {
13         maxEndingUntilIndex[i] = max(0, maxEndingUntilIndex[i-1] + A[i])
14     }
15
16     for i in (1..n-2).reversed() {
17         maxEndingFromIndex[i] = max(0, maxEndingFromIndex[i+1] + A[i])
18     }
19
20     var sum = 0
21     for i in 0..n-3 {
22         sum = maxEndingUntilIndex[i] + maxEndingFromIndex[i+2]
23         maxResult = max(maxResult, sum)
24     }
25
26     return maxResult
27 }
```

Analysis summary

The solution obtained perfect score.

Analysis ?

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

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

The PDF version of this report that may be downloaded on top of this site may contain sensitive data including personal information. For security purposes, we recommend you remove it from your system once reviewed.