Codility_

CodeCheck Report: training4F2BNF-33H

Test Name:

Check out Codility training tasks

Summary | Timeline

Tasks summary

Task	Time spent	Score
MaxSliceSum C#	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:

```
class Solution { public int solution(int[] A); }
```

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
```

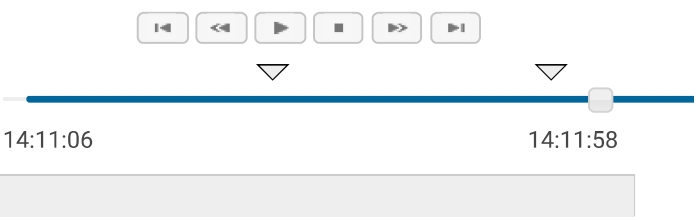
the function should return 5 because:

- (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,

Solution

Programming language used:	C#
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	*not defined yet*

Task timeline



- 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].

Code: 14:11:57 UTC, cs, final, score: **100** show code in pop-up

```
using System;

/**
 * 9.2 - Max Slice Sum
 * Paulo Santos
 * 15.Dec.2022
 */
class Solution {
    public int solution(int[] A) {

        /*
         * Check the input
         */
        if (A == null)
            throw new ArgumentNullException();

        int aMax = A[0];
        int lMax = A[0];
        int sum = 0;

        for (int i = 1; i < A.Length; i++) {
          sum = lMax + A[i];
          lMax = Math.Max(A[i], sum);
          aMax = Math.Max(aMax, lMax);
        }
        return aMax;
    }
}

```

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