Codility_

CodeCheck Report: trainingYX63G5-JFS

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
MaxDoubleSliceSum C# ⚠	75 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,

Solution

Programming language used:	C#
Total time used:	75 minutes
Effective time used:	75 minutes
Notes:	*not defined yet*

Task timeline

14:29:01 15:43:09

- 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:

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

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

For example, given:

```
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].

Code: 15:43:08 UTC, cs, final, score: **100**

show code in pop-up

```
using System;

/**
 * 9.3 - Max Double Slice
 * Paulo Santos
 * 15.Dec.2022
 */
class Solution {
    public int solution(int[] A) {
        var len = A.Length;
        var s1Max = new int[len];
        var s2Max = new int[len];

        for(var i = 1; i < A.Length - 1; i++) {
            s1Max[i] = Math.Max(s1Max[i - 1] + A[i]
        }

        for(int i = A.Length - 2; i > 0; i--) {
            s2Max[i] = Math.Max(s2Max[i + 1] + A[i]
        }

        int max = 0;

        for(int i = 1; i < A.Length - 1; i++) {
            max = Math.Max(max, s1Max[i - 1] + s2Ma
        }

        return max;
    }
}
```

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 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