

Test Name:

Summary Timeline

Tasks summary

Task	Time spent	Score
MaxProfit C#	4 min	100%

Total score

100%

Tasks Details

Easy

1. MaxProfit

Given a log of stock prices compute the maximum possible earning.

Task Score

100%

Correctness

100%

Performance

100%

Task description

An array A consisting of N integers is given. It contains daily prices of a stock share for a period of N consecutive days. If a single share was bought on day P and sold on day Q, where $0 \leq P \leq Q < N$, then the *profit* of such transaction is equal to $A[Q] - A[P]$, provided that $A[Q] \geq A[P]$. Otherwise, the transaction brings *loss* of $A[P] - A[Q]$.

For example, consider the following array A consisting of six elements such that:

A[0] = 23171
A[1] = 21011
A[2] = 21123
A[3] = 21366
A[4] = 21013
A[5] = 21367

If a share was bought on day 0 and sold on day 2, a loss of 2048 would occur because $A[2] - A[0] = 21123 - 23171 = -2048$. If a share was bought on day 4 and sold on day 5, a profit of 354

Solution

Programming language used: C#

Total time used:

4 minutes

?

Effective time used:

4 minutes

?

Notes:

not defined yet

Task timeline

13:45:35

13:48:49

would occur because $A[5] - A[4] = 21367 - 21013 = 354$.
Maximum possible profit was 356. It would occur if a share was bought on day 1 and sold on day 5.

Write a function,

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

that, given an array A consisting of N integers containing daily prices of a stock share for a period of N consecutive days, returns the maximum possible profit from one transaction during this period. The function should return 0 if it was impossible to gain any profit.

For example, given array A consisting of six elements such that:

```
A[0] = 23171
A[1] = 21011
A[2] = 21123
A[3] = 21366
A[4] = 21013
A[5] = 21367
```

the function should return 356, as explained above.

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

- N is an integer within the range [0..400,000];
- each element of array A is an integer within the range [0..200,000].

Copyright 2009–2022 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Code: 13:48:48 UTC, cs, final,
score: 100

[show code in pop-up](#)

```
1  using System;
2  using System.Linq;
3  using System.Collections.Generic;
4
5  /**
6   * 9.1 - Max Profit
7   * Paulo Santos
8   * 14.Dec.2022
9   */
10 class Solution {
11     public int solution(int[] A) {
12
13         /*
14          * Check the input
15          */
16         if (A == null)
17             throw new ArgumentNullException();
18         if (A.Length == 0)
19             return 0;
20
21         var min = A[0];
22         var difs = new List<int>();
23         var dif = 0;
24
25         for (var i = 1; i < A.Length; i++) {
26             dif = Math.Max((A[i] - min), dif);
27             if (min > A[i]) {
28                 difs.Add(dif);
29                 dif = 0;
30                 min = A[i];
31             }
32         }
33         difs.Add(dif);
34
35         return difs.Max();
36     }
37 }
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all

Example tests

▶ example ✓ OK
example, length=6

expand all

Correctness tests

▶ simple_1 ✓ OK
V-pattern sequence, length=7

▶ simple_desc ✓ OK
descending and ascending sequence,
length=5

▶ simple_empty ✓ OK
empty and [0,200000] sequence

▶	two_hills	✓ OK
	two increasing subsequences	
▶	max_profit_after_max_and_bef	✓ OK
	ore_min	
	max profit is after global maximum and before global minimum	
expand all		
Performance tests		
▶	medium_1	✓ OK
	large value (99) followed by short V-pattern (values from [1..5]) repeated 100 times	
▶	large_1	✓ OK
	large value (99) followed by short pattern (values from [1..6]) repeated 10K times	
▶	large_2	✓ OK
	chaotic sequence of 200K values from [100K..120K], then 200K values from [0..100K]	
▶	large_3	✓ OK
	chaotic sequence of 200K values from [1..200K]	