Codility_

CodeCheck Report: trainingE2QDW3-SX8

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
MaxProfit JavaScript ⚠	30 min	100%

Total score

100%

Tasks Details

1. MaxProfit

Easy

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

Task Score	Correctness	Performance
100%	100%	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 ≤ P ≤ Q < N, then the *profit* of such transaction is equal to A[Q] − A[P], provided that A[Q] ≥ 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 would occur because A[5] − A[4] = 21367 − 21013 = 354.

Solution

Programming language used:	JavaScript
Total time used:	30 minutes
Effective time used:	30 minutes
Notes:	*not defined yet*

Task timeline

04:38:05 05:07:09

Maximum possible profit was 356. It would occur if a share was bought on day 1 and sold on day 5.

Write a function,

```
function solution(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].

Code: 05:07:09 UTC, js, final, score: **100**

show code in pop-up

```
// you can write to stdout for debugging purposes,
// console.log('this is a debug message');
function solution(A) {
    var N = A.length;

    var max_profit = 0;
    var min = 200001;

    for (var i = 0; i < N; i++) {
        if (min > A[i]) {
            min = A[i];
        } else {
            max_profit = Math.max(max_profit, A[i]
        }
    }


    return max_profit;
}
```

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_before_min ✓ OK
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 chaotic sequence of 200K values from [100K..120K], then 200K values from [0..100K]	✓ OK
▶	large_3 chaotic sequence of 200K values from [1..200K]	✓ OK