codility

Check out Codility training tasks

Candidate Report: Anonymous

Test Name:

Summary Timeline

Test Score

66 out of 100 points

66%

Tasks in Test

	Time Spent	Task Score
MaxProfit Submitted in: Java...	2 min	66%

TASKS DETAILS

1. MaxProfit

EASY

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

Task Score	Correctness	Performance
66%	100%	25%

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:

Solution

Programming language used:	JavaScript
Total time used:	2 minutes
Effective time used:	2 minutes
Notes:	*not defined yet*

```
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. 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:

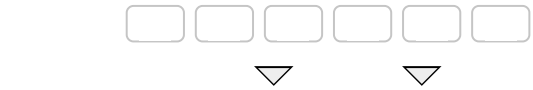
```
A[0] = 23171
A[1] = 21011
A[2] = 21123
A[3] = 21366
A[4] = 21013
A[5] = 21367
```

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].

Task timeline



17:25:02 17:26:40

Code: 17:26:40 UTC, js, final, score: **66** show code in pop-up

```
1  // you can write to stdout for debugging purp
2  // console.log('this is a debug message');
3
4  function solution(arr) {
5      let result = 0;
6
7      for (i = 0; i < arr.length - 1; i++) {
8          const firstElem = arr[i];
9          const rightPart = arr.slice(i+1);
10
11         maxValue = Math.max.apply(null, right
12         const subtractionResult = maxValue -
13
14         if (subtractionResult > 0 && subtract
15             result = subtractionResult;
16         }
17     }
18
19     return result;
20 }
```

Analysis summary

The following issues have been detected: timeout errors.

Analysis

Detected time complexity: **O(N**2)**

expand all	Example tests	
▶	example example, length=6	✓ OK
expand all	Correctness tests	
▶	simple_1 V-pattern sequence, length=7	✓ OK
▶	simple_desc descending and ascending	✓ OK

	sequence, length=5	
►	simple_empty empty and [0,200000] sequence	✓ OK
►	two_hills two increasing subsequences	✓ OK
►	max_profit_after_max_and_before_min max profit is after global maximum and before global minimum	✓ OK

expand all Performance tests

►	medium_1 large value (99) followed by short V-pattern (values from [1..5]) repeated 100 times	✓ OK
►	large_1 large value (99) followed by short pattern (values from [1..6]) repeated 10K times	✗ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
►	large_2 chaotic sequence of 200K values from [100K..120K], then 200K values from [0..100K]	✗ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
▼	large_3 chaotic sequence of 200K values from [1..200K]	✗ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.

1.	6.000 s	**TIMEOUT ERROR**, Killed. Hard limit reached: 6.000 sec.