

## Demo ticket

### Session

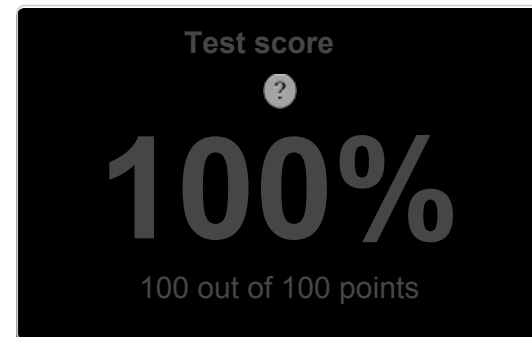
ID: demoPF977E-7NJ  
Time limit: 120 min.

### Status: closed

Created on: 2014-03-17 16:33 UTC  
Started on: 2014-03-17 16:33 UTC  
Finished on: 2014-03-17 17:14 UTC

### Tasks in test

### Task score



MEDIUM

#### 1. MaxProfit

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

score: 100 of 100



#### Task description

A zero-indexed 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 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,

```
int solution(const vector<int> &A);
```

that, given a zero-indexed 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
```

#### Solution

Programming language used: C++

Total time used: 42 minutes

Effective time used: 42 minutes

Notes: correct functionality and scalability

#### Task timeline



16:33:18

17:14:46

Code: 17:14:46 UTC, cpp, final, score: 100.00

```
01. // you can also use includes, for example:
02.
03. #include <algorithm>
04. #include <climits>
05. int solution(const vector<int> &A) {
06.     // set to max!
07.     long long min_price = LLONG_MAX;
08.     long long max_profit = 0;
09.
10.     for (int i = 0; i < (int)A.size(); i++) {
11.         max_profit = max(max_profit, (long long)A[i] -
12.             min_price);
13.         min_price = min(min_price, (long long)A[i]);
14.     }
15.     return max_profit;
16. }
```

#### Analysis

A[5] = 21567

the function should return 356, as explained above.  
Assume that:

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

Complexity:

- expected worst-case time complexity is  $O(N)$ ;
- expected worst-case space complexity is  $O(1)$ , beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

Copyright 2009–2014 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Detected time complexity:

$O(N)$

test	time	result
example example, length=6	0.020 s.	OK
simple_1 V-pattern sequence, length=7	0.020 s.	OK
simple_desc descending and ascending sequence, length=5	0.020 s.	OK
simple_empty empty and [0,200000] sequence	0.020 s.	OK
two_hills two increasing subsequences	0.020 s.	OK
max_profit_after_max_and_before_min max profit is after global maximum and before global minimum	0.020 s.	OK
medium_1 large value (99) followed by short V-pattern (values from [1..5]) repeated 100 times	0.020 s.	OK
large_1 large value (99) followed by short pattern (values from [1..6]) repeated 10K times	0.030 s.	OK
large_2 chaotic sequence of 200K values from [100K..120K], then 200K values from [0..100K]	0.120 s.	OK
large_3 chaotic sequence of 200K values from [1..200K]	0.070 s.	OK

Training center