Codility_

CodeCheck Report: trainingDZHT7E-QSB

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
MaxProfit Python	8 min	100%

Total score

100%

Tasks Details

Easy

1. MaxProfit

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

Write a function,

```
def 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].

Solution

Programming language used:	Python
Total time used:	8 minutes
Effective time used:	8 minutes
Notes:	*not defined yet*

Task timeline

01:34:39 01:42:01

Code: 01:42:00 UTC, py, final, score: **100** show code in pop-up

```python
# you can write to stdout for debugging purposes, e.g.
# print("this is a debug message")

def mp(B):
    if not B:
        return 0
    N = len(B)
    if N == 1:
        return 0
    if N == 2:
        if B[0] > B[1]:
            return 0
        else:
            return B[1]-B[0]

    min_ = min(B)
    idx_min = 0
    for i in range(N):
        if B[i] == min_:
            idx_min = i
            break
    max_ = max(B)
    idx_max = 0
    for i in range(N-1, -1, -1):
        if B[i] == max_:
            idx_max = i
            break
    max_profit = 0
    if idx_max > idx_min:
        max_profit = max(max_profit, B[idx_max]-B[idx_min])
    else:
        if B[idx_min+1:]:
            max_profit = max(max_profit, max(B[idx_min+1:])-B[
        if B[:idx_max]:
            max_profit = max(max_profit, B[idx_max]-max(B[:idx_
        max_profit = max(max_profit, mp(B[idx_max+1:idx_min]))
    if max_profit > 0:
        return max_profit
    else:
        return 0

def solution(A):
    # write your code in Python 3.6
    max_profit = mp(A)
    if max_profit > 0:
        return max_profit
    else:
        return 0
```

Analysis summary

The solution obtained perfect score.

Analysis

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

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 sequence, length=5	✔ OK
▶	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	✔ OK
▶	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