Codility_

CodeCheck Report: training2CMPCM-EAC

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
MaxProductOfThree ⚠ Python	3 min	100%

Total score

100%

Tasks Details

Easy

1. MaxProductOfThree
Maximize A[P] * A[Q] * A[R] for any triplet (P, Q, R).

Task Score	Correctness	Performance
100%	100%	100%

Task description

A non-empty array A consisting of N integers is given. The *product* of triplet (P, Q, R) equates to A[P] * A[Q] * A[R] (0 ≤ P < Q < R < N).

For example, array A such that:

```
A[0] = -3
A[1] = 1
A[2] = 2
A[3] = -2
A[4] = 5
A[5] = 6
```

contains the following example triplets:

- (0, 1, 2), product is −3 * 1 * 2 = −6
- (1, 2, 4), product is 1 * 2 * 5 = 10
- (2, 4, 5), product is 2 * 5 * 6 = 60

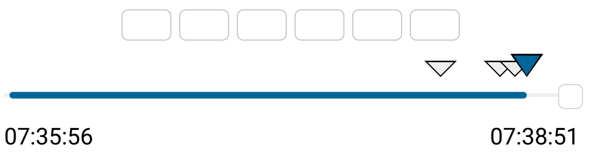
Your goal is to find the maximal product of any triplet.

Write a function:

Solution

Programming language used:	Python
Total time used:	3 minutes
Effective time used:	3 minutes
Notes:	*not defined yet*

Task timeline



Code: 07:38:50 UTC, py, final, score: **100**

show code in pop-up

```
def solution(A)
```

that, given a non-empty array A, returns the value of the maximal product of any triplet.

For example, given array A such that:

```
A[0] = -3
A[1] = 1
A[2] = 2
A[3] = -2
A[4] = 5
A[5] = 6
```

the function should return 60, as the product of triplet (2, 4, 5) is maximal.

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

- N is an integer within the range [3..100,000];
- each element of array A is an integer within the range [−1,000..1,000].

```
# you can write to stdout for debugging pur
# print("this is a debug message")

def solution(A):
    # Implement your solution here
    # pass
    sorted_A = sorted(A)
    product_max_1 = sorted_A[-1] * sorted_A
    product_max_2 = sorted_A[0] * sorted_A[

    return max(product_max_1, product_max_2
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all	Example tests	
▶	example example test	✔ OK

expand all	Correctness tests	
▶	one_triple three elements	✔ OK
▶	simple1 simple tests	✔ OK
▶	simple2 simple tests	✔ OK
▶	small_random random small, length = 100	✔ OK

expand all	Performance tests	
▶	medium_range -1000, -999, ... 1000, length = ~1,000	✔ OK
▶	medium_random random medium, length = ~10,000	✔ OK
▶	large_random random large, length = ~100,000	✔ OK
▶	large_range 2000 * (-10..10) + [-1000, 500, -1]	✔ OK
▶	extreme_large (-2, .., -2, 1, .., 1) and (MAX_INT)..(MAX_INT), length = ~100,000	✔ OK