Codility_

CodeCheck Report: training3STDT2-PXG

Test Name:

Check out Codility training tasks

Summary | Timeline

Tasks summary

Task	Time spent	Score
MaxProductOfThree Python	1 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:

```
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].

Solution

Programming language used:	Python
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	*not defined yet*

Task timeline

00:02:12 — 00:03:10

Code: 00:03:09 UTC, py, final, score: 100

show code in pop-up

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

def solution(A):
    # write your code in Python 3.6
    N = len(A)
    if N == 3:
        return A[0]*A[1]*A[2]

    if N == 4:
        arr = A.copy()
        arr.sort()
        if max(arr) <= 0:
            return arr[1]*arr[2]*arr[3]
        elif arr[3] >= 0 and arr[2] < 0:
            return arr[0]*arr[1]*arr[3]
        elif arr[3] >= 0 and arr[2] >= 0 and arr[1] < 0:
            return arr[0]*arr[1]*arr[3]
        else:
            return arr[1]*arr[2]*arr[3]

    are_all_negative = True
    for el in A:
        if el > 0:
            are_all_negative = False

    if are_all_negative == True:
        max_ = -1000-1
        first_largest_negative = max_
        second_largest_negative = max_
        third_largest_negative = max_
        for el in A:
            first_largest_negative = max(first_largest_negati
        i = 0
        for el in A:
            if el == first_largest_negative:
                i += 1
        if i > 1:
            second_largest_negative = first_largest_negative
        else:
            for el in A:
                if el != first_largest_negative:
                    second_largest_negative = max(second_larg
        i = 0
        j = 0
        for el in A:
            if el == first_largest_negative:
                i += 1
            if el == second_largest_negative:
                j += 1
        if i > 2:
            third_largest_negative = first_largest_negative
        else:
            if j > 1:
                third_largest_negative = second_largest_negat
            else:
                for el in A:
                    if el != first_largest_negative and el !=
                        third_largest_negative = max(third_la
        return first_largest_negative*second_largest_negative

    max_ = -1000-1
    first_largest_positive = max_
    second_largest_positive = max_
    third_largest_positive = max_
    for el in A:
        first_largest_positive = max(first_largest_positive, 
    i = 0
    for el in A:
        if el == first_largest_positive:
            i += 1
    if i > 1:
        second_largest_positive = first_largest_positive
    else:
        for el in A:
            if el != first_largest_positive:
                second_largest_positive = max(second_largest_
    i = 0
    j = 0
    for el in A:
        if el == first_largest_positive:
            i += 1
        if el == second_largest_positive:
            j += 1
    if i > 2:
        third_largest_positive = first_largest_positive
    elif i == 2:
        for el in A:
            if el != first_largest_positive:
                third_largest_positive = max(third_largest_po
    else:
        if j > 1:
            third_largest_positive = second_largest_positive
        else:
            for el in A:
                if el != first_largest_positive and el != sec
                    third_largest_positive = max(third_larges
    min_ = 1000+1
    first_smallest_negative = min_
    second_smallest_negative = min_
    for el in A:
        first_smallest_negative = min(first_smallest_negative
    i = 0
    for el in A:
        if el == first_smallest_negative:
            i += 1
    if i > 1:
        second_smallest_negative = first_smallest_negative
    else:
        for el in A:
            if el != first_smallest_negative:
                second_smallest_negative = min(second_smalles

    if first_largest_positive > 0 and second_largest_positive
        if first_smallest_negative >= 0 and second_smallest_n
            return first_largest_positive*second_largest_posi
        elif second_smallest_negative >= 0 and second_smalles
            return first_largest_positive*second_largest_posi
        elif second_smallest_negative < 0 and second_smallest_
            return max(first_largest_positive*second_largest_j
    elif first_largest_positive > 0 and second_largest_positi
        return first_largest_positive*first_smallest_negative
    else:
        return first_largest_positive*first_smallest_negative
```

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