Codility_

Candidate Report: trainingYW3K3A-HHY

Check out Codility training tasks

Test Name:

Summary Timeline

Tasks summary

Task	Time spent	Score
MaxProductOfThree Python	27 min	100%

Total score

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:

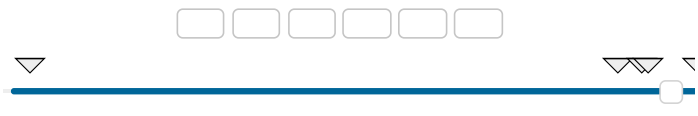
```
def solution(A)
```

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

Solution

Programming language used:	Python
Total time used:	27 minutes
Effective time used:	27 minutes
Notes:	*not defined yet*

Task timeline



06:50:04 07:17:04

Code: 07:17:04 UTC, py, final, score: **100** show code in pop-up

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].


```python
# 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
    if len(A)==3:
        return A[0]*A[1]*A[2]
    A.sort(reverse=True)

    ret1 = A[0]*A[1]*A[2]
    ret2 = A[0]*A[-1]*A[-2]

    return ret1 if ret1>=ret2 else ret2


```

Analysis summary

The solution obtained perfect score.

Analysis

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

collapse all — Example tests

▼ example example test		✓ OK
1. 0.036 s	OK	

collapse all — Correctness tests

▼ one_triple three elements		✓ OK
1. 0.036 s	OK	
2. 0.036 s	OK	
3. 0.036 s	OK	
▼ simple1 simple tests		✓ OK
1. 0.036 s	OK	
2. 0.036 s	OK	
3. 0.036 s	OK	
4. 0.036 s	OK	
▼ simple2 simple tests		✓ OK
1. 0.036 s	OK	
2. 0.036 s	OK	
3. 0.036 s	OK	
▼ small_random random small, length = 100		✓ OK
1. 0.036 s	OK	

collapse all — Performance tests

▼ medium_range -1000, -999, ... 1000, length = ~1,000	✓ OK
1. 0.040 s OK	
▼ medium_random random medium, length = ~10,000	✓ OK
1. 0.048 s OK	
▼ large_random random large, length = ~100,000	✓ OK
1. 0.164 s OK	
▼ large_range 2000 * (-10..10) + [-1000, 500, -1]	✓ OK
1. 0.084 s OK	
▼ extreme_large (-2, .., -2, 1, .., 1) and (MAX_INT).. (MAX_INT), length = ~100,000	✓ OK
1. 0.128 s OK	
2. 0.132 s OK	