Codility_

## Tasks Details

Check out Codility training tasks

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: | 6 minutes |
| Effective time used: | 6 minutes |
| Notes: | *not defined yet* |

### Task timeline

00:37:45 00:43:09

Code: 00:43:09 UTC, py, final, score: **100** show code in pop-up

```
1  # you can write to stdout for debugging purposes, e.g.
2  # print("this is a debug message")
3  import math
4  def solution(A):
5      # write your code in Python 3.6
6      s = 1
7      L = len(A)
8      i = 0
9      if L<=3:
10         while i<L:
11             s = s*A[i]
12             i = i + 1
13         return s
14     i = 0
15     s = 1
16     arr = sorted(A)
17     mx = -math.inf
18     positive = []
19     negative = []
20     zeros = []
21     while i<L:
22         v = arr[i]
23         if v>0:
24             positive.append(v)
25         elif v<0:
26             negative.append(v)
27         else:
28             zeros.append(v)
```

```
29          i = i + 1
30      #print(positive, negative, zeros)
31      L1 = len(positive)
32      L2 = len(negative)
33      L3 = len(zeros)
34      if L1>=3:
35          tmp = positive[-1]*positive[-2]*positive[-3]
36          #print(tmp)
37          if tmp>mx:
38              mx = tmp
39      if L2>=2 and L1>=1:
40          tmp = negative[0]*negative[1]*positive[-1]
41          #print(tmp)
42          if tmp>mx:
43              mx = tmp
44      elif L2>=3 and L1==0:
45          tmp = negative[-1]*negative[-2]*negative[-3]
46          if tmp>mx:
47              mx = tmp
48      if L3>0:
49          mx = max(mx, 0)
50      return mx
```

## Analysis summary

The solution obtained perfect score.

## Analysis

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

| expand all | Example tests | |
|---|---|---|
| ▶ | example<br>example test | ✓ OK |
| expand all | Correctness tests | |
| ▶ | one_triple<br>three elements | ✓ OK |
| ▶ | simple1<br>simple tests | ✓ OK |
| ▶ | simple2<br>simple tests | ✓ OK |
| ▶ | small_random<br>random small, length = 100 | ✓ OK |
| expand all | Performance tests | |
| ▶ | medium_range<br>-1000, -999, ... 1000, length = ~1,000 | ✓ OK |
| ▶ | medium_random<br>random medium, length = ~10,000 | ✓ OK |
| ▶ | large_random<br>random large, length = ~100,000 | ✓ OK |
| ▶ | large_range<br>2000 * (-10..10) + [-1000, 500, -1] | ✓ OK |
| ▶ | extreme_large<br>(-2, .., -2, 1, .., 1) and (MAX_INT).. | ✓ OK |

(MAX_INT), length = ~100,000

---