

Tasks summary

Task	Time spent	Score
MaxProductOfThree Python	1 min	100%

Total score



Tasks Details

Easy

1. MaxProductOfThree

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

Task Score

100%

Correctness

100%

Performance

100%

Task description

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

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:

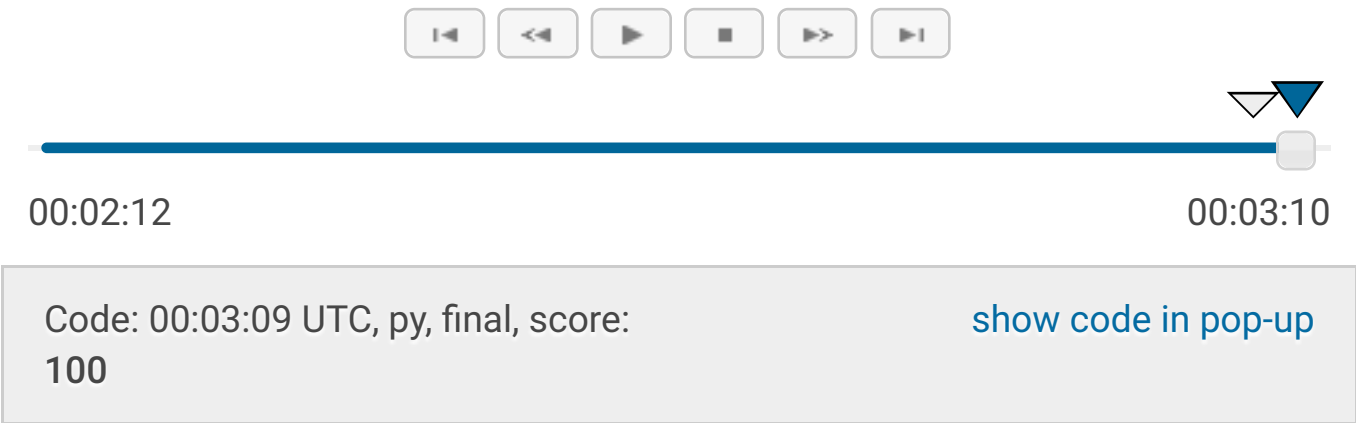
- $N$  is an integer within the range  $[3, 100,000]$ ;
- each element of array  $A$  is an integer within the range  $[-1,000, 1,000]$ .

Copyright 2009–2022 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Solution

Programming language used:	Python
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	<i>not defined yet</i>

Task timeline



```
Code: 00:03:09 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
4  def solution(A):
5      # write your code in Python 3.6
6      N = len(A)
7      if N == 3:
8          return A[0]*A[1]*A[2]
9
10     if N == 4:
11         arr = A.copy()
12         arr.sort()
13         if max(arr) <= 0:
14             return arr[1]*arr[2]*arr[3]
15         elif arr[3] >= 0 and arr[2] < 0:
16             return arr[0]*arr[1]*arr[3]
17         elif arr[3] >= 0 and arr[2] >= 0 and arr[1] < 0:
18             return arr[0]*arr[1]*arr[3]
19         else:
20             return arr[1]*arr[2]*arr[3]
21
22     are_all_negative = True
23     for el in A:
24         if el > 0:
25             are_all_negative = False
26
27     if are_all_negative == True:
28         max_ = -1000-1
29         first_largest_negative = max_
30         second_largest_negative = max_
31         third_largest_negative = max_
32         for el in A:
33             first_largest_negative = max(first_largest_negative, el)
34         i = 0
35         for el in A:
36             if el == first_largest_negative:
37                 i += 1
38             if i > 1:
39                 second_largest_negative = first_largest_negative
40         else:
41             for el in A:
42                 if el != first_largest_negative:
43                     second_largest_negative = max(second_largest_negative, el)
44         i = 0
45         j = 0
46         for el in A:
47             if el == first_largest_negative:
48                 i += 1
49             if el == second_largest_negative:
50                 j += 1
51             if i > 2:
52                 third_largest_negative = first_largest_negative
53         else:
54             if j > 1:
55                 third_largest_negative = second_largest_negative
56         else:
57             for el in A:
58                 if el != first_largest_negative and el != second_largest_negative:
59                     third_largest_negative = max(third_largest_negative, el)
60         return first_largest_negative*second_largest_negative*third_largest_negative
61
62     max_ = -1000-1
63     first_largest_positive = max_
64     second_largest_positive = max_
65     third_largest_positive = max_
66     for el in A:
67         first_largest_positive = max(first_largest_positive, el)
68     i = 0
69     for el in A:
70         if el == first_largest_positive:
71             i += 1
72             if i > 1:
73                 second_largest_positive = first_largest_positive
74         else:
75             for el in A:
76                 if el != first_largest_positive:
77                     second_largest_positive = max(second_largest_positive, el)
78     i = 0
79     j = 0
80     for el in A:
81         if el == first_largest_positive:
82             i += 1
83         if el == second_largest_positive:
84             j += 1
85             if i > 2:
86                 third_largest_positive = first_largest_positive
87     elif i == 2:
88         for el in A:
89             if el != first_largest_positive:
90                 third_largest_positive = max(third_largest_positive, el)
91     else:
92         if j > 1:
93             third_largest_positive = second_largest_positive
94         else:
95             for el in A:
96                 if el != first_largest_positive and el != second_largest_positive:
97                     third_largest_positive = max(third_largest_positive, el)
98     min_ = 1000+1
99     first_smallest_negative = min_
100    second_smallest_negative = min_
101    for el in A:
102        first_smallest_negative = min(first_smallest_negative, el)
103    i = 0
104    for el in A:
105        if el == first_smallest_negative:
106            i += 1
107            if i > 1:
108                second_smallest_negative = first_smallest_negative
109    else:
110        for el in A:
111            if el != first_smallest_negative:
112                second_smallest_negative = min(second_smallest_negative, el)
113
114    if first_largest_positive > 0 and second_largest_positive > 0 and third_largest_positive > 0:
115        if first_smallest_negative >= 0 and second_smallest_negative >= 0:
116            return first_largest_positive*second_largest_positive*third_largest_positive
117        elif second_smallest_negative >= 0 and second_smallest_negative < 0:
118            return first_largest_positive*second_largest_positive*second_smallest_negative
119        elif second_smallest_negative < 0 and second_smallest_negative < 0:
120            return max(first_largest_positive*second_largest_positive, first_smallest_negative*second_largest_positive)
121    elif first_largest_positive > 0 and second_largest_positive < 0:
122        return first_largest_positive*first_smallest_negative*second_largest_positive
123    else:
124        return first_largest_positive*first_smallest_negative*second_smallest_negative
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: <b>O(N * log(N))</b>	
expand all	Example tests
▶ example	OK
example test	
expand all	Correctness tests
▶ one_triple	OK
three elements	
▶ simple1	OK
simple tests	
▶ simple2	OK
simple tests	
▶ small_random	OK
random small, length = 100	
expand all	Performance tests
▶ medium_range	OK
-1000, -999, ..., 1000, length = ~1,000	
▶ medium_random	OK
random medium, length = ~10,000	
▶ large_random	OK
random large, length = ~100,000	
▶ large_range	OK
2000 * (-10, -10) + [-1000, 500, -1]	
▶ extreme_value	OK
(-2, ..., -2, 1, ..., 1) and (MAX_INT), (MAX_INT), length = ~100,000	