



Candidate Report: trainingEFZFRK-AFV

[Check out Codility training tasks](#)

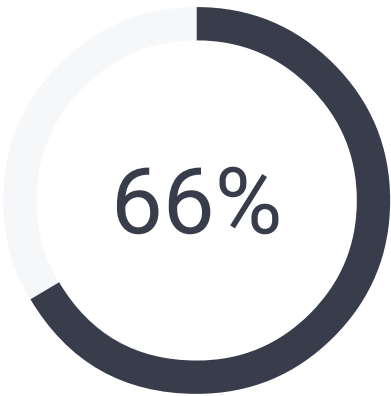
Test Name:

Summary Timeline

Tasks summary

Task	Time spent	Score
MaxProductOfThree C	66 min	66%

Total score



Tasks Details

Easy	1. MaxProductOfThree Maximize $A[P] * A[Q] * A[R]$ for any triplet (P, Q, R).	Task Score 66%	Correctness 75%	Performance 60%
------	--	-------------------	--------------------	--------------------

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
```

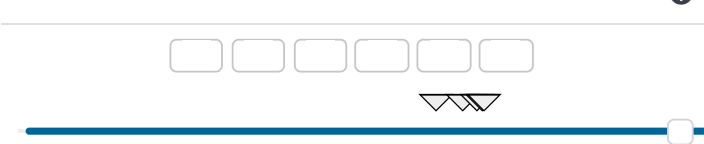
contains the following example triplets:

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

Solution

Programming language used:	C	
Total time used:	66 minutes	
Effective time used:	66 minutes	
Notes:	<i>not defined yet</i>	

Task timeline



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

07:53:23

08:58:37

Write a function:

```
int solution(int A[], int N);
```

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 2009–2020 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Code: 08:58:37 UTC, c, final,
score: 66

[show code in pop-up](#)

```
1 // you can write to stdout for debugging purposes, e
2 // printf("this is a debug message\n");
3 #define get_min_three(x,y,z) ((x<=y && x<=z)?0:((y<=
4 #define abs(x) (x<0? (-x):x)
5
6 int solution(int A[], int N) {
7     // write your code in C99 (gcc 6.2.0)
8     int i = 0;
9     int M[3], Mindex[3];
10    int tmp_index, tmp_val;
11    int neg_cnt = 0;
12    if (N==3) return A[0]*A[1]*A[2];
13
14    M[0] = A[0];
15    M[1] = A[1];
16    M[2] = A[2];
17
18    Mindex[0] = 0;
19    Mindex[1] = 1;
20    Mindex[2] = 2;
21
22    for (i = 3; i<N; i++){
23        tmp_index = get_min_three(abs(M[0]),abs(M[1]
24        if (A[i] > M[tmp_index]){
25            M[tmp_index] = A[i];
26            Mindex[tmp_index] = i;
27        }
28    }
29
30    neg_cnt = (M[0]<0)+(M[1]<0)+(M[2]<0);
31    //printf("nc%d: %d*%d*%d", neg_cnt,M[0],M[1],M[2]
32    tmp_val = -1000;
33    if (neg_cnt==3) {
34        for (i = 0; i<N; i++){
35            if (A[i] > tmp_val)
36                tmp_val = A[i];
37        }
38        tmp_index = get_min_three(abs(M[0]),abs(M[1]
39        M[tmp_index] = tmp_val;
40    }
41    else if (neg_cnt==1){
42        for (i = 0; i<N; i++){
43            if (A[i] > tmp_val && i^Mindex[0] && i^
44                tmp_val = A[i];
45        }
46        if (tmp_val>=0)
47            tmp_index = M[0]<0?0:(M[1]<0?1:2);
48        else
49            tmp_index = M[0]<0?(M[1]>=M[2]?2:1):(M[1]
50            M[tmp_index] = tmp_val;
51    }
52    return M[0]*M[1]*M[2];
53 }
```

Analysis summary

The following issues have been detected: wrong answers.

For example, for the input [-4, -6, 3, 4, 5] the solution returned a wrong answer (got 60 expected 120).

Analysis

collapse all		Example tests
▼	example example test	✓ OK
1. 0.001 s OK		
collapse all		Correctness tests
▼	one_triple three elements	✓ OK
1. 0.001 s OK		
2. 0.001 s OK		
3. 0.001 s OK		
▼	simple1 simple tests	✗ WRONG ANSWER got 84 expected 105
1. 0.001 s WRONG ANSWER, got 84 expected 105		
2. 0.001 s WRONG ANSWER, got 60 expected 120		
3. 0.001 s OK		
4. 0.001 s OK		
▼	simple2 simple tests	✓ OK
1. 0.001 s OK		
2. 0.001 s OK		
3. 0.001 s OK		
▼	small_random random small, length = 100	✓ OK
1. 0.001 s OK		
collapse all		Performance tests
▼	medium_range -1000, -999, ... 1000, length = ~1,000	✗ WRONG ANSWER got 997002000 expected 999000000
1. 0.001 s WRONG ANSWER, got 997002000 expected 999000000		
▼	medium_random random medium, length = ~10,000	✓ OK
1. 0.001 s OK		
▼	large_random random large, length = ~100,000	✓ OK
1. 0.004 s OK		

▼	large_range	✗ WRONG ANSWER
	2000 * (-10..10) + [-1000, 500, -1]	got 50000 expected 5000000
1. 0.001 s WRONG ANSWER , got 50000 expected 5000000		
▼	extreme_large	✓ OK
	(-2, .., -2, 1, .., 1) and (MAX_INT).. (MAX_INT), length = ~100,000	
1. 0.004 s OK		
2. 0.004 s OK		

The PDF version of this report that may be downloaded on top of this site may contain sensitive data including personal information. For security purposes, we recommend you remove it from your system once reviewed.