Codility_

CodeCheck Report: trainingBMV8TA-8NW

Test Name:

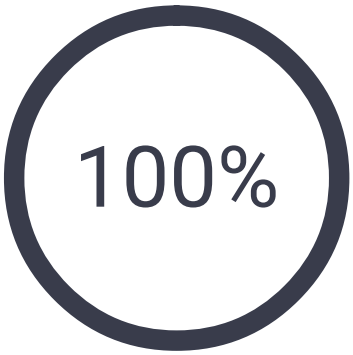
Check out Codility training tasks

Summary Timeline AI Assistant Transcript

Tasks summary

Task	Time spent	Score
MaxProductOfThree C++	3 min	100%

Total score



Tasks Details

1. MaxProductOfThree — Easy

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:

```
int solution(vector<int> &A);
```

Solution

Programming language used:	C++
Total time used:	3 minutes
Effective time used:	3 minutes
Notes:	*not defined yet*

Task timeline

04:28:07 04:31:07

Code: 04:31:06 UTC, cpp, final, score: **100** show code in pop-up

```
1  // you can use includes, for example:
2  #include <algorithm>
```

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].


```
3
4  // you can write to stdout for debugging purp
5  // cout << "this is a debug message" << endl;
6
7  int solution(vector<int> &A) {
8      // (0,1,2) -> A[0]*A[1]*A[2]
9      // find the maximal product of any triple
10     sort(A.begin(), A.end());
11     int size = A.size();
12
13     int result1 = A[size-1]*A[size-2]*A[size-3
14     int result2 = A[size-1]*A[0]*A[1];
15
16     return max(result1, result2);
17
18 }
```

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