Codility_

Candidate Report: trainingC7B6H8-FYS

Check out Codility training tasks

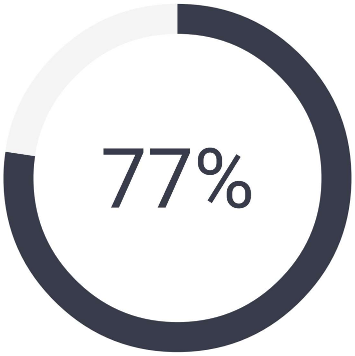
Test Name:

Summary Timeline Feedback

Tasks summary

Task	Time spent	Score
MaxProductOfThree C#	1 min	77%

Total score



Tasks Details

Easy

1. MaxProductOfThree

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

Task Score	Correctness	Performance
77%	100%	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 ≤ 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:

```
class Solution { public int solution(int[] A); }
```

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

Solution

Programming language used:	C#
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	*not defined yet*

Task timeline

21:23:53 21:24:37

Code: 21:24:37 UTC, cs, final, score: 77

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].

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Linq;
// you can also use other imports, for example:
// using System.Collections.Generic;

// you can write to stdout for debugging purposes, e.g.
// Console.WriteLine("this is a debug message");

class Solution {
           public int solution(int[] A)
        {
            var a = A.ToList().Where(x=>x>=0).OrderByDescen
            var b = A.ToList().Where(x=>x<0).OrderByDescend
            List<int> list2NegAnd1Positive = new List<int>(
            List<int> list3Positive = new List<int>();
            List<int> list3Negative = new List<int>();
            List<int> Maxes = new List<int>();
            if (a.Count() >= 3)
            {
                list3Positive = a.Take(3).ToList();
            }
            if (b.Count() >= 2)
            {
                list2NegAnd1Positive = b.Take(2).ToList();
                if (a.Count() >= 1)
                {
                    list2NegAnd1Positive.Add(a.First());
                }else
                {

                }
            }
            if (b.Count() >= 3)
            {
                list3Negative = b.Take(3).ToList();
            }

            if (list3Positive.Count >= 3)
                try
                {
                    Maxes.Add(list3Positive[0] * list3Posit
                }catch(Exception)
                { }
            if (list2NegAnd1Positive.Count >= 3)
                try
                {
                    Maxes.Add(list2NegAnd1Positive[0] * lis
                }catch(Exception)
                { }
            if (list3Negative.Count >= 3)
                try
                {
                    Maxes.Add(list3Negative[0] * list3Negat
                }
                catch (Exception)
                { }
            return Maxes.Max();




        }
}
```

Analysis summary

The following issues have been detected: wrong answers.

Analysis ?

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	✗ WRONG ANSWER got 997002000 expected 999000000
1. 0.032 s **WRONG ANSWER**, got 997002000 expected 999000000	
▶ medium_random random medium, length = ~10,000	✓ OK
▶ large_random random large, length = ~100,000	✓ OK
▼ large_range 2000 * (-10..10) + [-1000, 500, -1]	✗ WRONG ANSWER got 50000 expected 5000000
1. 0.116 s **WRONG ANSWER**, got 50000 expected 5000000	
▶ extreme_large (-2, .., -2, 1, .., 1) and (MAX_INT)..(MAX_INT), length = ~100,000	✓ 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.