codility

Check out Codility training tasks

Candidate Report: Anonymous

Test Name:

Summary Timeline

Test Score

66 out of 100 points

66%

Tasks in Test

	Time Spent	Task Score
MaxProductOfThree Submitted in: Java 8	1 min	66%

TASKS DETAILS

EASY

1. MaxProductOfThree

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

Task Score	Correctness	Performance
66%	50%	80%

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.

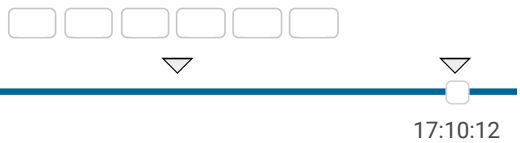
For example, given array A such that:

```
A[0] = -3
A[1] = 1
A[2] = 2
A[3] = -2
```

Solution

Programming language used:	Java 8
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	*not defined yet*

Task timeline



17:09:54 17:10:12

Code: 17:10:11 UTC, java, final, score: **66** show code in pop-up

```
1  // you can also use imports, for example:
2  // import java.util.*;
3
4  // you can write to stdout for debugging purposes, e.g.
5  // System.out.println("this is a debug message");
6
```

```
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].


```java
import java.util.Arrays;

public class Solution {

	public static int solution(int[] a) {
		class Wrap implements Comparable<Wrap> {
			boolean neg;
			int v;

			@Override
			public int compareTo(Wrap o) {
				return this.v - o.v;
			}

		}

		boolean allneg = true;
		for (int s = 0; s < a.length; s++) {
			if (a[s] >= 0) {
				allneg = false;
			}
		}

		if (allneg) {
			Arrays.sort(a);
			int r = 1;
			for (int i = a.length - 1; i > a
				r = r * a[i];

			return r;
		}

		Wrap[] w = new Wrap[a.length];

		for (int s = 0; s < a.length; s++) {
			w[s] = new Wrap();
			w[s].neg = a[s] < 0;
			w[s].v = Math.abs(a[s]);
		}

		Arrays.sort(w);

		int r = 1;
		int m = 1;
		for (int i = w.length - 1; i >= 0; i--)
			m = m * (w[i].neg ? -1 : 1);
			r = r * w[i].v;
			// System.out.println("m=" + m +

			if (m == -1 && i <= w.length - 3
				if (i == 0)
					return m * r;

				// System.out.println(i)
				m = 1;
				r = r / w[i].v;

			} else if (i <= w.length - 3) {
				return r;
			}
		}

		return r;
	}

}
```

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 90 expected 120).

Analysis

expand all	Example tests	
▶ example example test		✓ OK
expand all	Correctness tests	
▶ one_triple three elements		✓ OK
▶ simple1 simple tests		✗ WRONG ANSWER got 90 expected 120
▶ simple2 simple tests		✗ WRONG ANSWER got 200000 expected 600
▶ small_random random small, length = 100		✓ OK
expand all	Performance tests	
▶ medium_range -1000, -999, ... 1000, length = ~1,000		✗ WRONG ANSWER got 998000000 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]		✓ OK
▶ extreme_large (-2, .., -2, 1, .., 1) and (MAX_INT).. (MAX_INT), length = ~100,000		✓ OK