1/11/14 Codility

Get account with free plan

See how Codility works from recruiter's point of view.

closed

Demo ticket

Session ID: demoVF7CFF-RMA Time limit: 120 min.

Status: closed

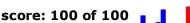
Started on: 2014-01-10 15:50 UTC

Score:

of 100

1. MaxProductOfThree

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



Task description

A non-empty zero-indexed 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] = 1A[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 zero-indexed 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

the function should return 60, as the product of triplet (2, 4, 5) is maximal. Assume that:

- 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].

Complexity:

- expected worst-case time complexity is O(N*log(N));
- expected worst-case space complexity is O(1), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

Copyright 2009–2014 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Solution

Programming language used: C#

Total time used: 1 minutes

Effective time used: 1 minutes

Notes: correct functionality and scalability

Task timeline



Code: 15:51:57 UTC, cs, final, score: 100.00

```
using System.Collections.Generic;
     using System.Linq;
03.
     class Solution {
            public int solution(int[] A)
04.
05.
                 // write your code in C# with .NET 2.0
06.
97.
                 var sorted =
                    Solution.sort(A.ToList<int>
                    ());
08.
                 var count = sorted.Count;
                09.
                sorted[count - 1];
var leftMaximum = sorted[0] *
10.
                    sorted[1] * sorted[count -
1];  // In the event first
                    2 elements are negative
11.
                 return (rightMaximum >=
12.
                    leftMaximum ? rightMaximum :
                    leftMaximum);
13.
             }
14.
15.
             // Merge sort not really required
                as the Sort function provided by
                .NET performs the same
                functionality but just for
                knowledge
16.
             public static List<int>
                sort(List<int> value)
17.
18.
                 if (value.Count == 1)
19.
                    return value;
20.
                 var midIndex = (int)
                (value.Count / 2);
var left = value.GetRange(0,
21.
                   midIndex);
22.
                 var right =
                    value.GetRange(midIndex,
                    value.Count - midIndex);
23.
24.
                left = sort(left);
25.
                 right = sort(right);
26.
                 var leftIndex = 0;
```

1/11/14 C

```
Codility
    28.
                     var rightIndex = 0;
                     var result = new List<int>();
    29.
    30.
                     for (var count = 0; count <</pre>
                        left.Count + right.Count;
                        count++)
    31.
                         if (leftIndex ==
    32.
                            left.Count)
    33.
    34.
                             // If no more elements
                                in the left index
                                append all sorted
                                elements from the
                                right array to the
                                result then exit
    35.
                             result.AddRange(right.GetRange(right
                                right.Count
                                rightIndex));
    36.
                            break;
    37.
                         else if (rightIndex ==
  right.Count)
    38.
    39.
                             // If no more elements
    40.
                                in the right index
                                append all sorted
                                elements from the
                                left array to the
                                result then exit
    41.
                             result.AddRange(left.GetRange(leftIr
                                left.Count
                                leftIndex));
    42.
                            break;
    43.
                         else if (left[leftIndex] >
  right[rightIndex])
    44.
    45.
                            result.Add(right[rightIndek]);
    46.
    47.
                             rightIndex++;
    48.
                         else
    49.
    50.
                         {
                             result.Add(left[leftIndex]);
    51.
    52.
                             leftIndex++;
    53.
    54.
    55.
                     return result;
    56.
                 }
    57. }
   Analysis
```

O(N * log(N))

	test	time	result
Get acco	example example test	0.080 s.	ок
	one_triple three elements	0.080 s.	ок
	simple1 simple tests	0.080 s.	ок
	simple2 simple tests	0.080 s.	ок
	small_random random small, length = 100	0.090 s.	ок
	medium_range -1000, -999, 1000, length = ~1,000	0.090 s.	ок
	medium_random random medium, length = ~10,000	0.110 s.	ок
	large_random random large, length = ~100,000	0.330 s.	ок
	large_range 2000 * (-1010) + [-1000, 500, -1]	0.200 s.	ок
	extreme_large (-2,, -2, 1,, 1) and (MAX_INT) (MAX_INT), length = ~100,000	0.320 s.	ок