Codility_

CodeCheck Report: trainingCSSJQM-K75

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
MinAbsSumOfTwo Java 8 ⚠	1 min	100%

Total score

100%

Tasks Details

Medium

1. MinAbsSumOfTwo
Find the minimal absolute value of a sum of two elements.

Task Score	Correctness	Performance
100%	100%	100%

Task description

Let A be a non-empty array consisting of N integers.

The *abs sum of two* for a pair of indices (P, Q) is the absolute value |A[P] + A[Q]|, for 0 ≤ P ≤ Q < N.

For example, the following array A:

```
A[0] =  1
A[1] =  4
A[2] = -3
```

has pairs of indices (0, 0), (0, 1), (0, 2), (1, 1), (1, 2), (2, 2).
The abs sum of two for the pair (0, 0) is A[0] + A[0] = |1 + 1| = 2.
The abs sum of two for the pair (0, 1) is A[0] + A[1] = |1 + 4| = 5.
The abs sum of two for the pair (0, 2) is A[0] + A[2] = |1 + (−3)| = 2.
The abs sum of two for the pair (1, 1) is A[1] + A[1] = |4 + 4| = 8.
The abs sum of two for the pair (1, 2) is A[1] + A[2] = |4 + (−3)| = 1.
The abs sum of two for the pair (2, 2) is A[2] + A[2] = |(−3) + (−3)| = 6.

Write a function:

Solution

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

Task timeline

20:53:37 20:54:13

Code: 20:54:13 UTC, java, final, score: **100**

show code in pop-up

```
// you can also use imports, for example:
import java.util.*;
```

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

that, given a non-empty array A consisting of N integers, returns the minimal abs sum of two for any pair of indices in this array.

For example, given the following array A:

```
A[0] =  1
A[1] =  4
A[2] = -3
```

the function should return 1, as explained above.

Given array A:

```
A[0] = -8
A[1] =  4
A[2] =  5
A[3] =-10
A[4] =  3
```

the function should return |(−8) + 5| = 3.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..100,000];
- each element of array A is an integer within the range [−1,000,000,000..1,000,000,000].

```java

// you can write to stdout for debugging purpo
// System.out.println("this is a debug message

class Solution {
        public int solution(int[] A) {
                final int N = A.length;

                Arrays.sort(A);

                int left = 0;
                int right = N - 1;

                int minAbsSum = Math.abs(A[le
                while (left <= right) {
                        int sum = A[left] + A
                        minAbsSum = Math.min(m
                        if (sum <= 0) {
                                left++;
                        } else {
                                right--;
                        }
                }

                return minAbsSum;
        }
}
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(N * log(N))**

collapse all **Example tests**

▼ example1 ✔ OK
first example

1. 0.004 s **OK**

▼ example2 ✔ OK
second example

1. 0.004 s **OK**

collapse all **Correctness tests**

▼ extreme_single ✔ OK
sequences of 1 elements

1. 0.004 s **OK**
2. 0.004 s **OK**
3. 0.004 s **OK**

▼ extreme_double ✔ OK
sequences of 2 elements

1. 0.004 s **OK**
2. 0.008 s **OK**
3. 0.004 s **OK**

▼ positive_small ✔ OK
only positive numbers

1. 0.004 s **OK**

▼ negative_small ✔ OK
only negative numbers

1. 0.008 s **OK**

collapse all

Performance tests

▼ random_small ✔ OK
random sequence, length = ~1000

1. 0.008 s **OK**

▼ random_medium ✔ OK
random sequence, length = ~10,000

1. 0.052 s **OK**

▼ arithmetic_medium ✔ OK
arithemtic sequence, length = ~10,000

1. 0.124 s **OK**

▼ random_large ✔ OK
random sequence, length = ~100,000

1. 0.468 s **OK**

▼ extreme_large ✔ OK
sequence of MAX_INT, length = ~100,000

1. 0.452 s **OK**

▼ arithmetic_large ✔ OK
arithmetic sequence, length = ~100,000

1. 0.440 s **OK**

▼ constant_distance ✔ OK
constant distance between all elements, length = 100,000

1. 0.380 s **OK**