

CodeCheck Report: trainingTMCRVH-47A

Test Name:

[Check out Codility training tasks](#)

Summary

Timeline

Tasks summary

Task	Time spent	Score
MinAbsSum Java 8	62 min	100%

Total score

100%

Tasks Details

Hard	1. <b>MinAbsSum</b>	Task Score	Correctness	Performance
	Given array of integers, find the lowest absolute sum of elements.			
		100%	100%	100%

Task description

For a given array A of N integers and a sequence S of N integers from the set {−1, 1}, we define val(A, S) as follows:

$$\text{val}(A, S) = |\sum\{ A[i]*S[i] \text{ for } i = 0..N-1 \}|$$

(Assume that the sum of zero elements equals zero.)

For a given array A, we are looking for such a sequence S that minimizes val(A,S).

Write a function:

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

that, given an array A of N integers, computes the minimum value of val(A,S) from all possible values of val(A,S) for all possible sequences S of N integers from the set {−1, 1}.

For example, given array:

```
A[0] = 1
A[1] = 5
```

Solution

Programming language used:	Java 8
Total time used:	62 minutes
Effective time used:	62 minutes
Notes:	not defined yet

Task timeline



A[2] = 2  
A[3] = -2

your function should return 0, since for S = [-1, 1, -1, 1], val(A, S) = 0, which is the minimum possible value.

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

- N is an integer within the range [0..20,000];
- each element of array A is an integer within the range [-100..100].

Copyright 2009–2021 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Code: 09:04:46 UTC, java,  
final, score: 100

[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,
5 // System.out.println("this is a debug message");
6
7 class Solution {
8     public int solution(int[] a){
9         if (a.length == 0) return 0;
10        if (a.length == 1) return a[0];
11        int sum = 0;
12        for (int i=0;i<a.length;i++){
13            sum += Math.abs(a[i]);
14        }
15        int[] indices = new int[a.length];
16        indices[0] = 0;
17        int half = sum/2;
18        int localSum = Math.abs(a[0]);
19        int minLocalSum = Integer.MAX_VALUE;
20        int placeIndex = 1;
21        for (int i=1;i<a.length;i++){
22            if (localSum<half){
23                if (Math.abs(2*minLocalSum-sum) > Math.
24                    minLocalSum = localSum;
25                localSum += Math.abs(a[i]);
26                indices[placeIndex++] = i;
27            }else{
28                if (localSum == half)
29                    return Math.abs(2*half - sum);
30
31                if (Math.abs(2*minLocalSum-sum) > Math.
32                    minLocalSum = localSum;
33                if (placeIndex > 1) {
34                    localSum -= Math.abs(a[indices[plac
35                    i = indices[placeIndex];
36                }
37            }
38        }
39        return (Math.abs(2*minLocalSum - sum));
40    }
41 }
42 }
```

## Analysis summary

The solution obtained perfect score.

## Analysis

Detected time complexity:  $O(N * \max(\text{abs}(A))^2)$

expand all

### Example tests



example1  
example test

✓ OK

expand all

### Correctness tests



simple1	✓ OK
simple 1	
▶ simple2	✓ OK
simple 2	
▶ simple3	✓ OK
simple 3	
▶ range	✓ OK
range 2..20	
▶ extreme	✓ OK
empty and single element	
▶ functional	✓ OK
small functional test	
expand all	Performance tests
▶ medium1	✓ OK
medium random	
▶ medium2	✓ OK
multiples of 10 + 5	
▶ big1	✓ OK
multiples of 5 + 42	
▶ big3	✓ OK
all 4s and one 3	
▶ big4	✓ OK
multiples of 10	