



CodeCheck Report: trainingDJDZEK-QYX

[Check out Codility training tasks](#)

Test Name:

Summary Timeline

Tasks summary

Task	Time spent	Score
MinAbsSum Java 8	30 min	100%

Total score

100%

Tasks Details

Hard	1. MinAbsSum	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:

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

(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
A[2] = 2
A[3] = -2
```

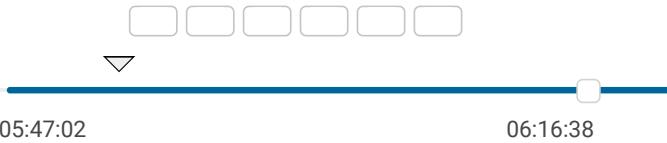
your function should return 0, since for S = [−1, 1, −1, 1], val(A, S) = 0, which is the minimum possible value.

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

Solution

Programming language used:	Java 8	
Total time used:	30 minutes	
Effective time used:	30 minutes	
Notes:	<i>not defined yet</i>	

Task timeline



Code: 06:16:37 UTC, java, [show code in pop-up](#)
final, score: 100

```
1 // you can also use imports, for example:
2 // import java.util.*;
3
4 // you can write to stdout for debugging purposes,
```

- 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.

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

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity:

O(N *
max(abs(A))**2)

expand all	Example tests
▶ example1 example test	✓ OK
expand all	Correctness tests
▶ simple1 simple 1	✓ OK
▶ simple2 simple 2	✓ OK
▶ simple3 simple 3	✓ OK
▶ range range 2..20	✓ OK
▶ extreme empty and single element	✓ OK
▶	

functional		✓ OK
small functional test		
expand all	Performance tests	
▶	medium1 medium random	✓ OK
▶	medium2 multiples of 10 + 5	✓ OK
▶	big1 multiples of 5 + 42	✓ OK
▶	big3 all 4s and one 3	✓ OK
▶	big4 multiples of 10	✓ OK