Codility_

CodeCheck Report: trainingNEA2BK-4JD

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
MinAbsSum Java 8	18 min	100%

Total score

100%

Tasks Details

Hard

1. MinAbsSum

Given array of integers, find the lowest absolute sum of elements.

Task Score	Correctness	Performance
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:

val(A, S) = |**sum**{ A[i]*S[i] 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.

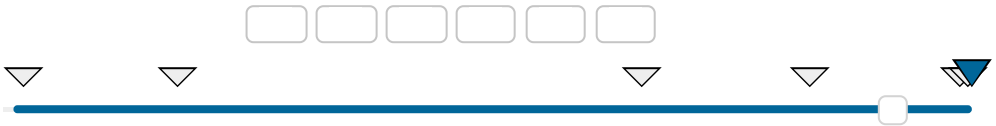
Write an **efficient** algorithm for the following assumptions:

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

Solution

Programming language used:	Java 8
Total time used:	18 minutes
Effective time used:	18 minutes
Notes:	*not defined yet*

Task timeline



18:45:19 19:02:56

Code: 19:02:56 UTC, java, final, score: **100** show code in pop-up

```
import java.util.Arrays;

class Solution {
        /**
         * Solution - 100% (correctness: 100%, performanc
         *
         * Solution was inspired by
         * https://codility.com/media/train/solution-min-
         *
         * Detected time complexity: O(N * max(abs(A))**2
         *
         *
         * @param A
```

```
 * @return
 *
 * @see https://app.codility.com/demo/results/tra
 */
public int solution(int[] A) {
        int N = A.length;

        //
        // The main idea is to find all achievabl
        // Then for each sum = s1, we get s2 = ma
        //

        //
        // This task is equivalent to the task ho
        // two persons so that the difference bet
        //

        // 1. replace A[i] with its absolute valu
        // 2. find max value
        // 3. find max sum
        int maxValue = 0;
        int maxSum = 0;

        for (int i = 0; i < N; i++) {
                A[i] = Math.abs(A[i]);
                maxValue = Math.max(maxValue, A[i
                maxSum += A[i];
        }

        // count of value A[i]
        int[] count = new int[maxValue + 1];
        for (int val : A) {
                count[val]++;
        }

        // this array marks whether we can achiev
        // the count array
        // dp[sum] = -1 if this sum is unachievab
        // dp[sum] >= 0 if this sum is achievable
        // "val" remained.

        // note: we consider only sum from 0 to "
        int[] dp = new int[maxSum / 2 + 1];

```

```
                // At the beginning,
        // set dp[sum] = -1 if sum > 0
                // set dp[sum] = 0 if sum = 0
                Arrays.fill(dp, 1, dp.length, -1);

                // for each set of value "val", we check
                for (int val = 0; val <= maxValue; val++)
                        if (count[val] > 0) {
                                for (int sum = 0; sum < d
                                        if (dp[sum] >= 0)
                                                // we don
                                                // so the
                                                dp[sum] =
                                        } else if (sum >=
                                                // "dp[su
                                                // "sum >
                                                // (sum -

                                                // Theref
                                                dp[sum] =
                                        }
                                }
                        }
                }

                // find the available "sum" which is most
                for (int sum = dp.length - 1; sum >= 0; s
                        if (dp[sum] >= 0) {
                                // Math.abs(right - left)
                                // sum
                                return maxSum - 2 * sum;
                        }
                }

                return maxSum;
        }
}
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(N * max(abs(A))**2)**

collapse all — Example tests

▼ example1 — ✔ OK
example test

1. 0.004 s OK

collapse all — Correctness tests

▼ simple1 — ✔ OK
simple 1

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

▼ simple2 — ✔ OK
simple 2

1. 0.004 s OK
2. 0.004 s OK

▼ simple3 — ✔ OK
simple 3

1. 0.004 s OK
2. 0.004 s OK

3. 0.004 s OK

▼ range ✔ OK

range 2..20

1. 0.004 s OK
2. 0.004 s OK

▼ extreme ✔ OK

empty and single element

1. 0.004 s OK
2. 0.004 s OK

▼ functional ✔ OK

small functional test

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

collapse all

Performance tests

▼ medium1 ✔ OK

medium random

1. 0.012 s OK
2. 0.004 s OK

▼ medium2 ✔ OK

multiples of 10 + 5

1. 0.004 s OK
2. 0.004 s OK

▼ big1 ✔ OK

multiples of 5 + 42

1. 0.068 s OK
2. 0.008 s OK

▼ big3 ✔ OK

all 4s and one 3

1. 0.020 s OK

▼ big4 ✔ OK

multiples of 10

1. 0.116 s OK
2. 0.004 s OK