

## José David Cadena Diaz

### Candidate

E-mail: [jdavidcd555@gmail.com](mailto:jdavidcd555@gmail.com)  
Last school attended: Universidad Distrital

### Session

ID: NDTQGX-8CA  
Time limit: 180 min.  
Report recipients:  
[catalina.montano@pragma.com.co](mailto:catalina.montano@pragma.com.co)  
[luisa.chica@pragma.com.co](mailto:luisa.chica@pragma.com.co)  
[valeria.moreno@pragma.com.co](mailto:valeria.moreno@pragma.com.co)  
Accessed from: 201.244.233.203,  
201.244.233.203  
Invited by: [catalina.montano@pragma.com.co](mailto:catalina.montano@pragma.com.co)

### Status: closed

Invitation: [sent](#)  
Created on: 2018-06-27 22:11 UTC  
Started on: 2018-07-01 23:58 UTC  
Finished on: 2018-07-02 02:42 UTC




### Notes:

N/A

### Similarity Check

Status: not found  
No similar solutions have been detected.

### Tasks in test

- 1 |  Lightbulbs  
Submitted in: Java
- 2 |  MaxSumDistance  
Submitted in: Java
- 3 |  LastBoundedElement  
Submitted in: Java

Correctness	Performance	Task score
100%	0%	50%
100%	0%	50%
80%	100%	90%

### Test score

# 63%

190 out of 300 points

Next step: online coding interview



Start CodeLive Interview

## TASKS DETAILS

EASY

## 1. Lightbulbs

Calculate the number of prefixes that are a permutation.

Task Score	Correctness	Performance
50	100	0

### Task description

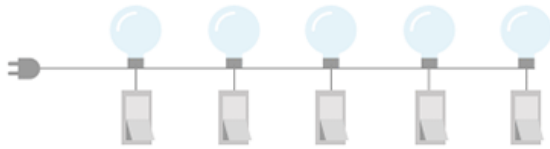
There are  $N$  bulbs, numbered from 1 to  $N$ , arranged in a row. The first bulb is plugged into the power socket and each successive bulb is connected to the previous one (the second bulb to the first, the third bulb to the second, etc.).

Initially, all the bulbs are turned off. At moment  $K$  (for  $K$  from 0 to  $N-1$ ), we turn on the  $A[K]$ -th bulb. A bulb shines if it is on and all the previous bulbs are turned on too.

Write a function `solution` that, given an array  $A$  of  $N$  different integers from 1 to  $N$ , returns the number of moments for which every turned on bulb shines.

#### Examples:

1. Given  $A=[2, 1, 3, 5, 4]$ , the function should return 3.



- At the 0th moment only the 2nd bulb is turned on, but it does not shine because the previous one is not on.
- At the 1st moment two bulbs are turned on (1st and 2nd) and both of them shine.
- At the 2nd moment three bulbs are turned on (1st, 2nd and 3rd) and all of them shine.
- At the 3rd moment four bulbs are turned on (1st, 2nd, 3rd and 5th), but the 5th bulb does not shine because the previous one is not turned on.
- At the 4th moment five bulbs are turned on (1st, 2nd, 3rd, 4th and 5th) and all five of them shine.

There are three moments (1st, 2nd and 4th) when every turned on bulb shines.

2. Given  $A=[2, 3, 4, 1, 5]$ , the function should return 2 (at the 3rd and 4th moment every turned on bulb shines).

3. Given  $A=[1, 3, 4, 2, 5]$ , the function should return 3 (at the 0th, 3rd and 4th moment every turned on bulb shines).

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

- $N$  is an integer within the range  $[1..100,000]$ ;
- the elements of  $A$  are all distinct;
- each element of array  $A$  is an integer within the range  $[1..N]$ .

Copyright 2009–2018 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

### Solution

[SEE LIVE VERSION](#)

Programming language used: Java

Total time used: 164 minutes

Effective time used: 80 minutes

Notes: *not defined yet*

### Source code

Code: 02:42:07 UTC, java, final,  
score: 50

```

1 // you can also use imports, for example:
2 // import java.util.*;
3
4 // you can write to stdout for debugging purposes,
5 e.g.
6 // System.out.println("this is a debug message");
7
8 class Solution {
9     public int solution(int[] A) {
10         int momentos=0;
11         int total = A.length;
12         String orden = "";
13
14         for(int i=0; i< total; i++){
15             int numero = A[i];
16             orden = orden+numero+"";
17             String arreglo[] = orden.split("");
18             arreglo = numeroOrdenado(arreglo);
19             int j= 0;
20             while (j< arreglo.length){
21                 if(Integer.parseInt(arreglo[j])==
22                 (j+1)){
23                     j++;
24                 }
25                 else{
26                     j= arreglo.length+1;
27                 }
28                 if(j == arreglo.length){
29                     momentos++;
30                 }
31             }
32         }
33         return momentos;
34     }
35
36     public String[] numeroOrdenado(String []B){
37         String auxiliar;
38         for(int k=0; k< B.length; k++){
39             for(int l=0; l<k; l++){
40                 if(Integer.parseInt(B[k])<
41                 Integer.parseInt(B[l])){
42                     auxiliar = B[l];
43                     B[l] = B[k];
44                     B[k] = auxiliar;
45                 }
46             }
47         }
48     }
49 }

```

```

46 |         }
47 |         return B;
48 |     }
49 | }

```

## Analysis summary

The following issues have been detected: timeout errors.

## Analysis

Detected time complexity:  **$O(N^{**2})$**

Example tests	
example_1 First example.	✓ OK
example_2 Second example.	✓ OK
example_3 Third example.	✓ OK
Correctness tests	
single single element	✓ OK
double two elements	✓ OK
small_functional small functional tests	✓ OK
small_sorted sorted sequence, N = 12	✓ OK
small_random small random permutation, N = 100	✓ OK
Performance tests	
medium_random medium random permutation, N = 10,000	✗ TIMEOUT ERROR running time: >6.00 sec., time limit: 0.11 sec.
medium_sorted sorted sequence, N = 10,000	✗ TIMEOUT ERROR running time: >6.00 sec., time limit: 0.11 sec.
large_random1 large random permutation, N = 100,000	✗ TIMEOUT ERROR running time: >6.00 sec., time limit: 0.99 sec.
large_random2 large random permutation, N = ~100,000	✗ TIMEOUT ERROR running time: >6.00 sec., time limit: 0.99 sec.
large_sorted sorted sequence, N = 100,000	✗ TIMEOUT ERROR running time: >7.00 sec., time limit: 1.01 sec.

MEDIUM

## 2. MaxSumDistance

Maximize the value of  $A[P] + A[Q] + (Q - P)$ .

Task Score	Correctness	Performance
50	100	0

### Task description

Let  $A$  be a non-empty array consisting of  $N$  integers. A *sum-distance* of a pair of indices  $(P, Q)$ , for  $0 \leq P \leq Q < N$ , is the value  $A[P] + A[Q] + (Q - P)$ .

For example, for the following array  $A$ :

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

there are the following pairs of indices:  $(0, 0)$ ,  $(1, 1)$ ,  $(2, 2)$ ,  $(0, 1)$ ,  $(1, 2)$ ,  $(0, 2)$ , for each of which the sum-distance is defined as follows:

- for  $(0, 0)$  it is  $A[0] + A[0] + (0 - 0) = 1 + 1 + 0 = 2$ ,
- for  $(1, 1)$  it is  $A[1] + A[1] + (1 - 1) = 3 + 3 + 0 = 6$ ,
- for  $(2, 2)$  it is  $A[2] + A[2] + (2 - 2) = (-3) + (-3) + 0 = -6$ ,
- for  $(0, 1)$  it is  $A[0] + A[1] + (1 - 0) = 1 + 3 + 1 = 5$ ,
- for  $(1, 2)$  it is  $A[1] + A[2] + (2 - 1) = 3 + (-3) + 1 = 1$ ,
- for  $(0, 2)$  it is  $A[0] + A[2] + (2 - 0) = 1 + (-3) + 2 = 0$ .

Write a function:

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

that, given an array  $A$  consisting of  $N$  integers, returns the maximal sum-distance value for this array.

For example, given the following array  $A$ :

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

the function should return 6, as explained above.

Given the following array  $A$ :

```
A[0] = -8
A[1] = 4
A[2] = 0
A[3] = 5
A[4] = -3
A[5] = 6
```

the function should return  $4 + 6 + (5 - 1) = 14$ .

Assume that:

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

Complexity:

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

Copyright 2009–2018 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

### Solution

[SEE LIVE VERSION](#)

Programming language used: Java

Total time used: 54 minutes

Effective time used: 54 minutes

Notes: *not defined yet*

### Source code

Code: 01:00:48 UTC, java, final, score: 50

```
1 // you can also use imports, for example:
2 // import java.util.*;
3
4 // you can write to stdout for debugging purposes,
5 e.g.
6 // System.out.println("this is a debug message");
7
8 class Solution {
9     public int solution(int[] A) {
10         String sumas = "";
11
12         for(int i=0; i< A.length; i++){
13             for(int j=0; j< A.length; j++){
14                 if(i<=j){
15                     int provincial = A[i]+ A[j]+
16                     Math.abs (i-j);
17                     sumas =
18                     sumas+String.valueOf(provincial)+"";
19                 }
20             }
21
22             String aSumas[] = sumas.split("");
23             int numeroMayor =
24             Integer.parseInt(aSumas[0]);
25
26             for(int k = 1; k < aSumas.length; k++){
27                 if(Integer.parseInt(aSumas[k])>numeroMayor){
28                     numeroMayor =
29                     Integer.parseInt(aSumas[k]);
30                 }
31             }
32             return numeroMayor;
33         }
34     }
```

### Analysis summary

The following issues have been detected: timeout errors.

## Analysis

Detected time complexity:  **$O(N^2)$**

Example tests	
example1 example from the problem statement	✓ OK
example2 example from the problem statement	✓ OK
Correctness tests	
simple1 small correctness test	✓ OK
simple2 small correctness tests	✓ OK
simple small correctness tests	✓ OK
extreme_min_max_value corner-cases with minimal and maximal values	✓ OK
extreme_single corner-cases with N = 1	✓ OK
Performance tests	
medium1 medium random test N = 3,006	✗ TIMEOUT ERROR running time: >6.00 sec., time limit: 0.10 sec.
medium2 medium random tests N = 5,000	✗ TIMEOUT ERROR running time: >6.00 sec., time limit: 0.10 sec.
large1 large random tests N > 90,000	✗ TIMEOUT ERROR running time: >7.00 sec., time limit: 1.41 sec.
large2 large random tests N > 90,000	✗ TIMEOUT ERROR running time: >7.00 sec., time limit: 1.41 sec.
large_wave alternate ascending/descending slices, N ~= 100,000	✗ TIMEOUT ERROR running time: >6.00 sec., time limit: 0.99 sec.

EASY

### 3. LastBoundedElement

Find the last element of a quadratic sequence below a threshold.

Task Score	Correctness	Performance
90	80	100

## Task description

Function  $F(K)$  is defined for non-negative integers as follows:

- $F(K) = 0$  when  $K = 0$
- $F(K) = F(K-1) + K$  when  $K > 0$

Write a function:

```
class Solution { public int solution(int N); }
```

that, given a non-negative integer  $N$ , returns the largest non-negative integer  $L$  such that  $F(L) \leq N$ .

For example, given  $N = 17$  the function should return 5, because  $F(5) = 15$ , and  $F(K) > 17$  for all integers  $K$  greater than 5.

Assume that:

- $N$  is an integer within the range  $[0..1,000,000,000]$ .

Complexity:

- expected worst-case time complexity is  $O(\sqrt{N})$ ;
- expected worst-case space complexity is  $O(1)$ .

Copyright 2009–2018 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

## Solution

[SEE LIVE VERSION](#)

Programming language used: Java

Total time used: 22 minutes

Effective time used: 22 minutes

Notes: *not defined yet*

## Source code

Code: 01:22:49 UTC, java, final,  
score: 90

```
1 // you can also use imports, for example:
2 // import java.util.*;
3
4 // you can write to stdout for debugging purposes,
5 e.g.
6 // System.out.println("this is a debug message");
7
8 class Solution {
9     public int solution(int N) {
10         int numeroMayor =0;
11         int acumulado =0;
12
13         for(int i=1; i<= N; i++){
14             if(N == 0){
15                 numeroMayor =0;
16             }
17             else{
18                 acumulado = acumulado +i;
19                 if(acumulado > N){
20                     numeroMayor = i-1;
21                     break;
22                 }
23             }
24         }
25         return numeroMayor;
26     }
```

## Analysis summary

The following issues have been detected: wrong answers.

For example, for the input 1 the solution returned a wrong answer (got 0 expected 1).

## Analysis

Detected time complexity:  **$O(\sqrt{N})$**

Example tests

example	✓ OK
Correctness tests	
simple1	✓ OK
simple2	✓ OK
simple3	✓ OK
extreme_n0	✓ OK
extreme_n1	✗ WRONG ANSWER got 0 expected 1
Performance tests	
medium1	✓ OK
medium2	✓ OK
medium3	✓ OK
big1	✓ OK
big2	✓ OK
big3	✓ OK