

Candidate Report: Anonymous

Test Name:

Summary Timeline

Test Score

83 out of 100 points

83%

Tasks in Test

	Time Spent ⓘ	Task Score
PermCheck Submitted in: Java 8	16 min	83%

TASKS DETAILS

EASY	1. PermCheck	Task Score	Correctness	Performance
	Check whether array A is a permutation.	83%	66%	100%

Task description

A non-empty array A consisting of N integers is given.

A *permutation* is a sequence containing each element from 1 to N once, and only once.

For example, array A such that:

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

is a permutation, but array A such that:

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

is not a permutation, because value 2 is missing.

The goal is to check whether array A is a permutation.

Write a function:

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

that, given an array A, returns 1 if array A is a permutation and 0 if it is not.

For example, given array A such that:

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

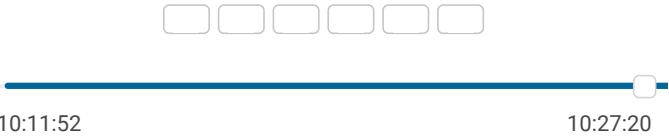
the function should return 1.

Given array A such that:

Solution

Programming language used:	Java 8	
Total time used:	16 minutes	?
Effective time used:	16 minutes	?
Notes:	not defined yet	

Task timeline ?



Code: 10:27:20 UTC, java, final, [show code in pop-up](#)
score: 83

```
1 // you can also use imports, for example:
2 // import java.util.*;
3
4 // you can write to stdout for debugging purposes, e.g.
5 // System.out.println("this is a debug message");
6
7 class Solution {
8     public static int max = 1000000;
9     public static boolean[] mem;
10
```

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

the function should return 0.

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..1,000,000,000].
- Copyright 2009–2020 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Test results - Codility

```
11 public static int solution(int[] a) {  
12     mem = new boolean[max + 1];  
13  
14     for (int i = 0; i < a.length; i++) {  
15         if (a[i] > a.length)  
16             return 0;  
17         mem[a[i]] = true;  
18     }  
19  
20     for (int i = 1; i < a.length; i++) {  
21         if (!mem[i])  
22             return 0;  
23     }  
24     return 1;  
25 }  
26  
27 }
```

Analysis summary

The following issues have been detected: wrong answers.
For example, for the input [1, 1] the solution returned a wrong answer (got 1 expected 0).

Analysis ?

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

expand all Example tests	
▶ example1 the first example test	✓ OK
▶ example2 the second example test	✓ OK
expand all Correctness tests	
▶ extreme_min_max single element with minimal/maximal value	✓ OK
▶ single single element	✓ OK
▶ double two elements	✗ WRONG ANSWER got 1 expected 0
▶ antiSum1 total sum is correct, but it is not a permutation, N <= 10	✓ OK
▶ small_permutation permutation + one element occurs twice, N = ~100	✗ WRONG ANSWER got 1 expected 0
▶ permutations_of_ranges permutations of sets like [2..100] for which the answers should be false	✓ OK
expand all Performance tests	
▶ medium_permutation permutation + few elements occur twice, N = ~10,000	✓ OK
▶ antiSum2 total sum is correct, but it is not a permutation, N = ~100,000	✓ OK
▶ large_not_permutation permutation + one element occurs three times, N = ~100,000	✓ OK

Test results - Codility

▶ large_range	✓ OK
sequence 1, 2, ..., N, N = ~100,000	
▶ extreme_values	✓ OK
all the same values, N = ~100,000	
▶ various_permutations	✓ OK
all sequences are permutations	

PDF version of this report that may be downloaded on top of this site may contain sensitive data including personal information. For security purposes, we recommend you remove it from your system once reviewed.