

Tasks summary

Task	Time spent	Score
PermCheck Java 8	14 min	100%

Total score



Tasks Details

Easy	1. <b>PermCheck</b> Check whether array A is a permutation.	Task Score	Correctness	Performance
		100%	100%	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
```

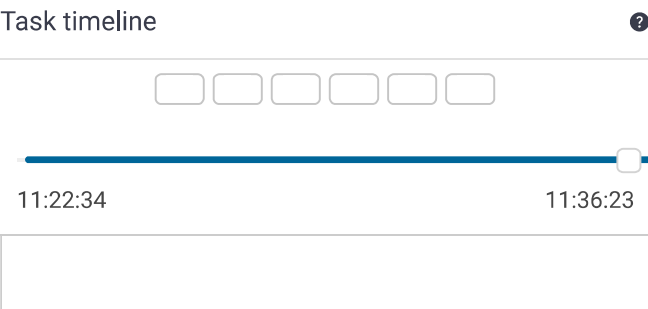
is not a permutation, because value 2 is missing.

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

Write a function:

Solution

Programming language used:	Java 8	
Total time used:	14 minutes	
Effective time used:	14 minutes	
Notes:	<i>not defined yet</i>	



```
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:

```
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 2009–2024 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Code: 11:36:23 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,
5 // System.out.println("this is a debug message");
6 import java.util.*;
7 class Solution {
8     public int solution(int[] A) {
9         int N = A.length;
10        Set<Integer> set = new HashSet<>();
11        for (int i:A){
12            if(i<1 || i > N || set.contains(i)){
13                return 0;
14            }
15            set.add(i);
16        }
17        return set.size() ==N?1 :0;
18    }
19 }
```

### Analysis summary

The solution obtained perfect score.

### Analysis

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

expand all	Example tests
▶ example1	✓ OK
the first example test	
▶ example2	✓ OK
the second example test	
expand all	Correctness tests
▶ extreme_min_max	✓ OK
single element with minimal/maximal value	
▶ single	✓ OK
single element	
▶ double	✓ OK
two elements	
▶ antiSum1	✓ OK
total sum is correct, but it is not a permutation, N <= 10	
▶ small_permutation	✓ OK
permutation + one element occurs twice, N = ~100	
▶ permutations_of_ranges	✓ OK
permutations of sets like [2..100] for	

which the answers should be false

expand all

### Performance tests

▶ medium_permutation	✓ OK
permutation + few elements occur twice, N = ~10,000	
▶ antiSum2	✓ OK
total sum is correct, but it is not a permutation, N = ~100,000	
▶ large_not_permutation	✓ OK
permutation + one element occurs three times, N = ~100,000	
▶ 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	