



Congratulations

You have completed a Codility training test.

Tweet this!

I scored 100% in #java on @Codility!
https://codility.com/demo/take-sample-test/perm_check/

Sign up for our newsletter!

Like us on Facebook!

Training ticket

Session

ID: trainingVUC8AJ-V2F
 Time limit: 120 min.

Status: closed

Created on: 2016-06-20 20:07 UTC
 Started on: 2016-06-20 20:07 UTC
 Finished on: 2016-06-20 20:08 UTC

Tasks in test

1 **PermCheck**
 Submitted in: Java

Correctness

100%

Performance

100%

Task score

100%

Test score

100%

100 out of 100 points

EASY

1. PermCheck

Check whether array A is a permutation.

score: 100 of 100



Task description

A non-empty zero-indexed 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 a zero-indexed 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
```

Solution

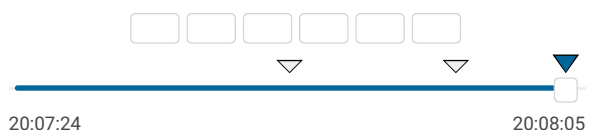
Programming language used: Java

Total time used: 1 minutes

Effective time used: 1 minutes

Notes: not defined yet

Task timeline



Code: 20:08:04 UTC, java, final,
 score: 100

[show code in pop-up](#)

```
1 import java.util.IntSummaryStatistics;
2 import java.util.stream.IntStream;
3
4 class Solution {
5     private static final int NOT_PERMUTATION = 0;
6     private static final int PERMUTATION = 1;
7
8     public int solution(int[] A) {
9         IntSummaryStatistics stats = IntStream.of(A).di
10         return containsAll(stats.getCount(), stats.get
11     }
12
13     private boolean isDistinct(long distinctElements, i
14         return distinctElements == inputElements;
15     }
16
17     private boolean containsAll(long distinctElements,
```

A[1] = 1
A[2] = 3
the function should return 0.

Assume that:

- 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].

Complexity:

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

Elements of input arrays can be modified.

Copyright 2009–2016 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```
18         return distinctElements == maxValue;  
19     }  
20 }
```

Analysis summary

The solution obtained perfect score.

Analysis



Detected time complexity:
O(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		
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_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		

Training center