codility

Training center

y training tasks

Congratulations

You have completed a Codility training test.

Sign up for our newsletter!

Like us on Facebook!

Training ticket

Session

ID: trainingGCY56M-H49
Time limit: 120 min.

Status: closed

Created on: 2016-06-05 08:06 UTC
Started on: 2016-06-05 08:06 UTC
Finished on: 2016-06-05 08:06 UTC

Tasks in test	Correctness	Performance	Task score
1 PermCheck Submitted in: Java	100%	100%	100%

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:

Solution

Programming language used: Java

Total time used: 1 minutes

Effective time used: 1 minutes

Notes: *not defined yet*

Task timeline

08:06:21 08:06:56

Code: 08:06:56 UTC, java, final, score: **100**

show code in pop-up

```
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 int solution(int[] A) {
9
10     boolean[] used = new boolean[A.length];
11     for (int i=0; i<A.length; i++) {
12       if (A[i] < 1 || A[i] > A.length || used[A[i]-1])
13         return 0;
14       }
15       used[A[i]-1] = true;
```

```
A[0] = 4
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.


```
16      }
17      return 1;
18
19      }
20  }
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity:
O(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	✔ OK
▶	antiSum1 total sum is correct, but it is not a permutation, N <= 10	✔ OK
▶	small_permutation permutation + one element occurs twice, N = ~100	✔ 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_permutation permutation + one element occurs three times, N = ~100,000	✔ OK
▶	large_range sequence 1, 2, ..., N, N = ~100,000	✔ OK
▶	extreme_values all the same values, N = ~100,000	✔ OK