

TASKS DETAILS

| | | | | | |
|------|---|------------|-------------|-------------|--|
| EASY | 1. | | | | |
| | PermCheck | | | | |
| | 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:

```
def solution(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
```

Solution

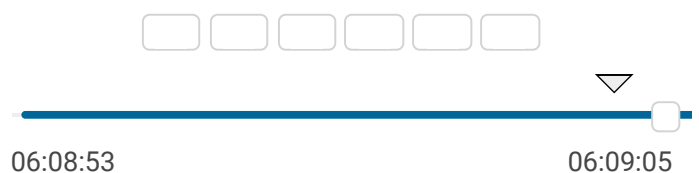
Programming language used: Python

Total time used: 1 minutes ?

Effective time used: 1 minutes ?

Notes: not defined yet

Task timeline ?



Code: 06:09:05 UTC, py, [show code in pop-up](#)
final, score: 100

```
1 # Solution 2
2 def solution(A):
3     permutation = sorted(A)
4     for idx, value in enumerate(permutation):
5         if idx + 1 != value:
6             return 0
7     return 1
```

Analysis summary

The solution obtained perfect score.

Analysis ?

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 $O(N)$ (not counting the storage required for input arguments).

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

Detected time
complexity:

$$O(N) \text{ or } O(N * \log(N))$$

| Example tests | |
|--|------|
| ▶ example1 | ✓ OK |
| the first example test | |
| ▶ example2 | ✓ OK |
| the second example test | |
| 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 | |
| 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 | |