codility

Training center

Check out Codility training tasks

TASKS DETAILS

EASY

1. **PermCheck**
Check whether array A is a permutation.

Task Score	Correctness	Performance
83%	83%	83%

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

06:00:44 06:01:01

Code: 06:01:01 UTC, py, final, score: **83**

show code in pop-up

```
1  # Solution 1
2  def solution(A):
3      permutation = sorted(A)
4      for i in range(1, permutation[-1] + 1):
5          if i != permutation[i-1]:
6              return 0
7      return 1
```

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).

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).

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	✔ OK
▶	permutations_of_ranges permutations of sets like [2..100] for which the anwsers 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
▶	large_range sequence 1, 2, ..., N, N = ~100,000	✔ OK
▶	extreme_values all the same values, N = ~100,000	✘ WRONG ANSWER got 1 expected 0
▶	various_permutations all sequences are permutations	✔ OK