


Tasks summary

Task	Time spent	Score
PermCheck  Python	6 min	100%

Total score



Tasks Details

Easy

1. PermCheck

Check whether array A is a permutation.

Task Score

100%

Correctness

100%

Performance

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
```



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].

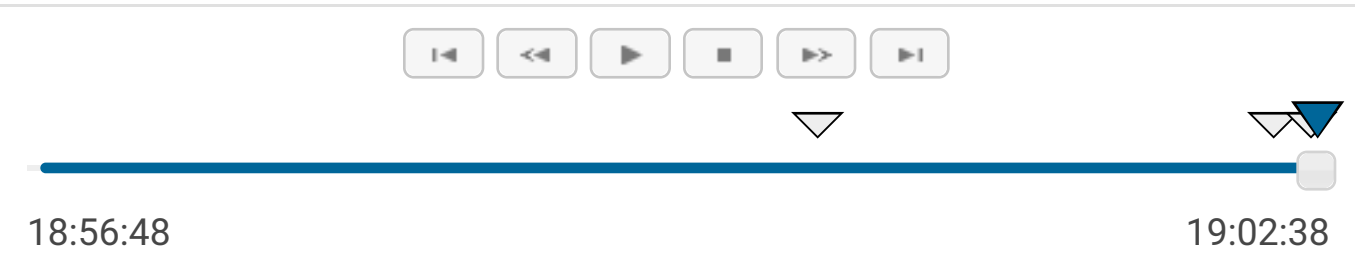
Solution

Programming language used:	Python
----------------------------	--------

Total time used:	6 minutes	
Effective time used:	6 minutes	

Notes: *not defined yet*

Task timeline



Code: 19:02:38 UTC, py, final, score: 100 [show code in pop-up](#)

```
1 # you can write to stdout for debugging purposes, e.g.
2 # print("this is a debug message")
3
4 def solution(A):
5     # write your code in Python 3.6
6     if len(A) == 1:
7         if A[0] == 1:
8             return 1
9         else:
10            return 0
11     elif max(A) > len(A):
12         return 0
13     else:
14         set_ = set(A)
15         B = [i+1 for i in range(len(A))]
16         set1 = set(B)
17         if set_ == set1:
18             return 1
19         else:
20             return 0
```

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	