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].

def solution(A):

return int(set(A) == set(range(1, len(A)+1)))

deger\_1 = [4,1,3,2]

car\_1 = solution(deger\_1)

print(car\_1)

or

def solution(A):

seen = [False] \* len(A)

for value in A:

if 0 <= value > len(A):

return 0

if seen[value-1] == True:

return 0

seen[value-1] = True

return 1

or

def solution(A):

total1 = 1

total2 = 1

for index,value in enumerate(A):

total1 \*= value

total2 = total2 \* (index+1)

if total1 == total2:

return int(1)

else:

return int(0)

deger\_1 = [4,1,3,2]

car\_1 = solution(deger\_1)

print(car\_1)