An array A consisting of N integers is given. The *dominator* of array A is the value that occurs in more than half of the elements of A.

For example, consider array A such that

A[0] = 3 A[1] = 4 A[2] = 3 A[3] = 2 A[4] = 3 A[5] = -1 A[6] = 3 A[7] = 3

The dominator of A is 3 because it occurs in 5 out of 8 elements of A (namely in those with indices 0, 2, 4, 6 and 7) and 5 is more than a half of 8.

Write a function

def solution(A)

that, given an array A consisting of N integers, returns index of any element of array A in which the dominator of A occurs. The function should return −1 if array A does not have a dominator.

For example, given array A such that

A[0] = 3 A[1] = 4 A[2] = 3 A[3] = 2 A[4] = 3 A[5] = -1 A[6] = 3 A[7] = 3

the function may return 0, 2, 4, 6 or 7, as explained above.

Write an **efficient** algorithm for the following assumptions:

* N is an integer within the range [0..100,000];
* each element of array A is an integer within the range [−2,147,483,648..2,147,483,647].

from collections import Counter

def solution(A):

if len(A) ==0:

return -1

if len(A) ==1:

return 0

count1 = Counter(A)

max\_occ = count1.most\_common(1)

if max\_occ[0][1] > len(A) / 2:

if max\_occ != []:

return A.index(max\_occ[0][0])

else:

return -1

else:

return -1

A = [3, 4, 3, 2, 3, -1, 3, 3]

car1 = solution(A)

print(car1)