A non-empty array A consisting of N integers is given. A pair of integers (P, Q), such that 0 ≤ P < Q < N, is called a *slice* of array A (notice that the slice contains at least two elements). The *average* of a slice (P, Q) is the sum of A[P] + A[P + 1] + ... + A[Q] divided by the length of the slice. To be precise, the average equals (A[P] + A[P + 1] + ... + A[Q]) / (Q − P + 1).

For example, array A such that:

A[0] = 4 A[1] = 2 A[2] = 2 A[3] = 5 A[4] = 1 A[5] = 5 A[6] = 8

contains the following example slices:

* slice (1, 2), whose average is (2 + 2) / 2 = 2;
* slice (3, 4), whose average is (5 + 1) / 2 = 3;
* slice (1, 4), whose average is (2 + 2 + 5 + 1) / 4 = 2.5.

The goal is to find the starting position of a slice whose average is minimal.

Write a function:

def solution(A)

that, given a non-empty array A consisting of N integers, returns the starting position of the slice with the minimal average. If there is more than one slice with a minimal average, you should return the smallest starting position of such a slice.

For example, given array A such that:

A[0] = 4 A[1] = 2 A[2] = 2 A[3] = 5 A[4] = 1 A[5] = 5 A[6] = 8

the function should return 1, as explained above.

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

* N is an integer within the range [2..100,000];
* each element of array A is an integer within the range [−10,000..10,000].

def solution(A):

min\_index = 0

min\_value = float("inf")

for index1 in range(0, len(A)-1):

if (A[index1] + A[index1+1])/2.0 < min\_value:

min\_index = index1

min\_value = ( A[index1] + A[index1+1] )/2.0

if ( index1 < len(A)-2 ) and ( (A[index1] + A[index1+1] + A[index1+2])/3.0 < min\_value ) :

min\_index = index1

min\_value = (A[index1] + A[index1+1] + A[index1+2])/3.0

return min\_index

A = [4, 2, 2, 5, 1, 5, 8]

car1 = solution(A)

print(car1)