A non-empty array A consisting of N integers is given. Array A represents numbers on a tape.

Any integer P, such that 0 < P < N, splits this tape into two non-empty parts: A[0], A[1], ..., A[P − 1] and A[P], A[P + 1], ..., A[N − 1].

The *difference* between the two parts is the value of: |(A[0] + A[1] + ... + A[P − 1]) − (A[P] + A[P + 1] + ... + A[N − 1])|

In other words, it is the absolute difference between the sum of the first part and the sum of the second part.

For example, consider array A such that:

A[0] = 3 A[1] = 1 A[2] = 2 A[3] = 4 A[4] = 3

We can split this tape in four places:

* P = 1, difference = |3 − 10| = 7
* P = 2, difference = |4 − 9| = 5
* P = 3, difference = |6 − 7| = 1
* P = 4, difference = |10 − 3| = 7

Write a function:

def solution(A)

that, given a non-empty array A of N integers, returns the minimal difference that can be achieved.

For example, given:

A[0] = 3 A[1] = 1 A[2] = 2 A[3] = 4 A[4] = 3

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 [−1,000..1,000].

def solution(A):

if len(A) == 2:

return abs(A[0] - A[1])

minimum = float (“inf”)

total = sum(A)

leftSum = 0

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

leftSum += A[i]

rightSum = total - leftSum

dif = abs(rightSum - leftSum)

minimum = min(minimum, dif)

return minimum

deger = [2, 3, 1, 5]

car\_1 = solution(deger)

print(car\_1)

or

def solution(A):

P = 1

N = len(A)

L = sum(A[:P])

R = sum(A[P:])

minimal\_difference = abs(L - R)

for P in range(2, N):

current = A[P - 1]

L += current

R -= current

minimal\_difference = min(minimal\_difference, abs(L - R))

return minimal\_difference