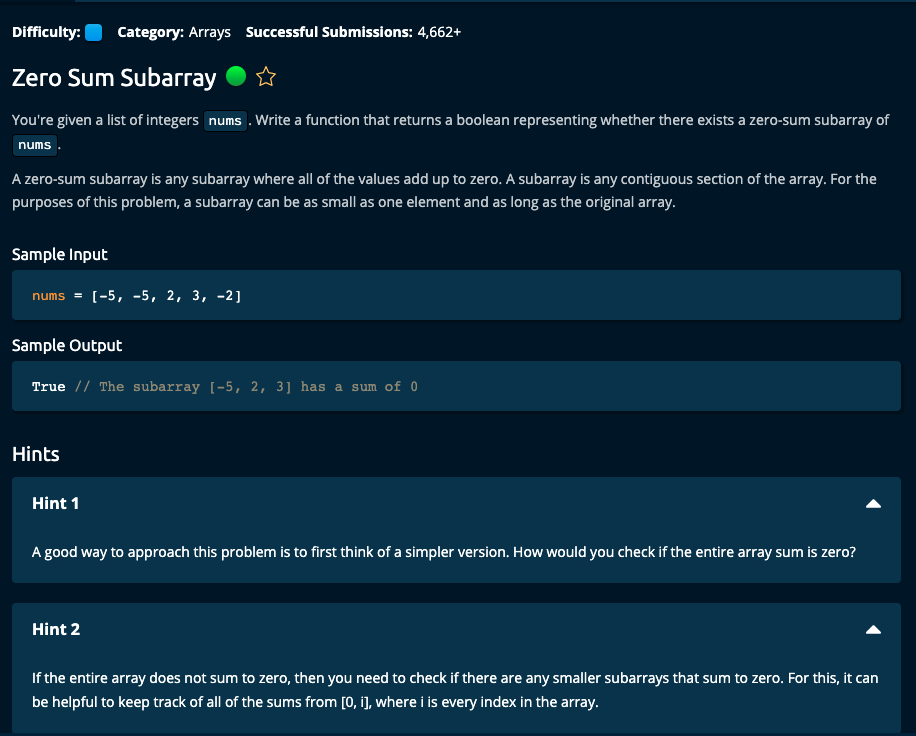
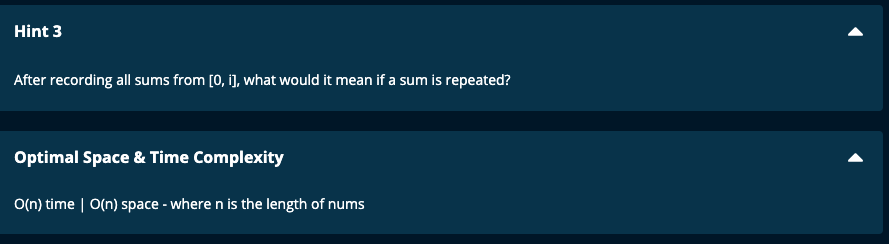
Zero Sum Subarray (Medium)





My Solution:

# My Solution using a set -- O(n) time | O(n) space

def zeroSumSubarray(nums):

if len(nums) == 0:

return False

if sum(nums) == 0:

return True

totals = set()

runningTotal = 0

for num in nums:

runningTotal += num

print("num = ", num, "runningTotal = ", runningTotal)

if runningTotal not in totals:

totals.add(runningTotal)

else:

return True

return False

JJ Notes:

1. If nums is an empty list then return False
2. If sum of all the elements in nums is 0, then return True
3. Otherwise, create an empty set to hold totals and initialize runningTotal to 0.
4. Then iterate though nums array. Add the current number to runningTotal.

If runningTotal is not in totals set, then add it. Otherwise return True.

1. After iterating through the array, return False since we have not found a repeat in the totals set.

The idea is if runningTotal is x and we add some more elements and the total still remains x, then the subarray after the runningTotal to the current element should have a sum of 0.

i.e. x + 0 = x

Time is O(n) since we iterate once through nums. Space is O(n) since we have to store the set of totals.

Algoexpert Solution:

# Algoexpert Solution -- O(n) time | O(n) space

def zeroSumSubarray(nums):

sums = set([0])

currentSum = 0

for num in nums:

currentSum += num

if currentSum in sums:

return True

sums.add(currentSum)

return False

Test cases

