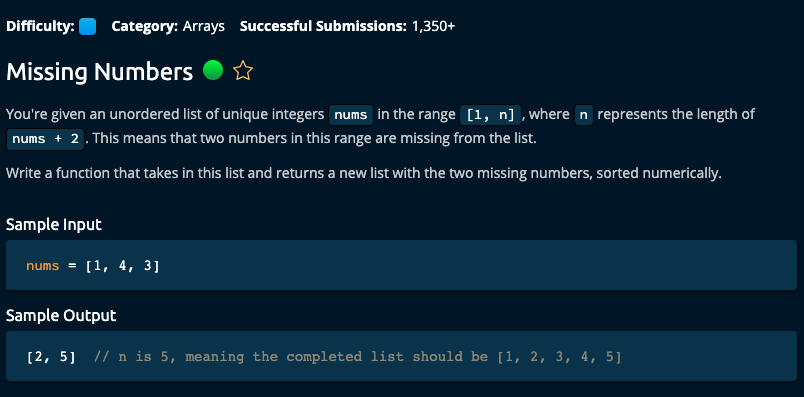
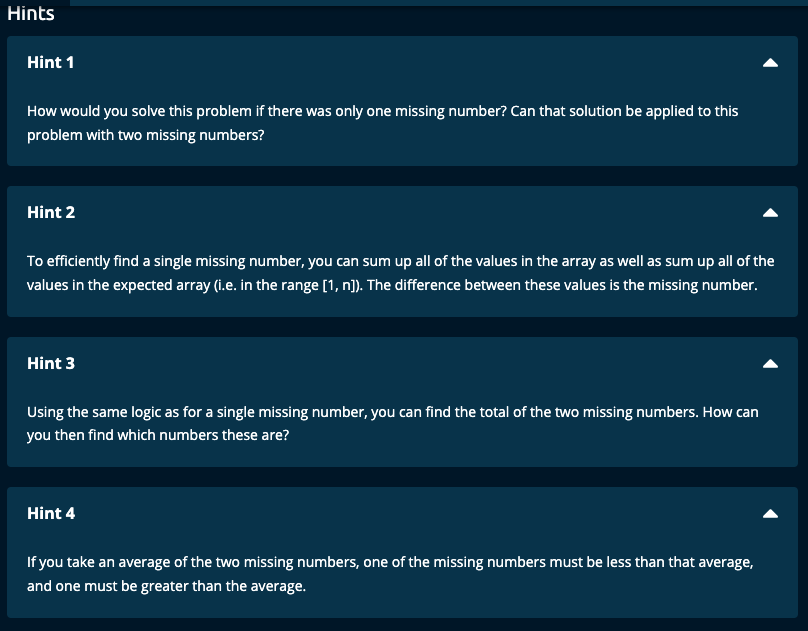
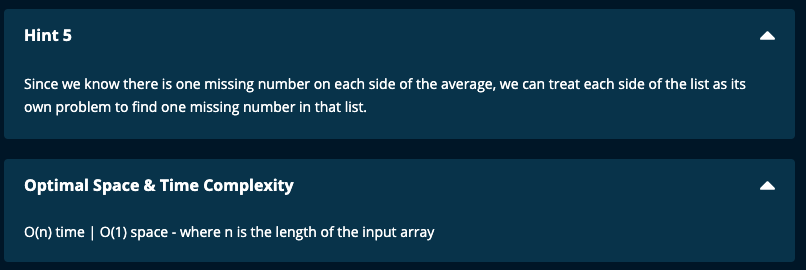
Missing Numbers (Medium)







My Solutions:

Solution 1:

# Find missing Sum, and then first missing number.

# O(n) time | O(n) space for numsSet

def missingNumbers(nums):

total = sum(nums)

n = len(nums) + 2

expectedTotal = n \* (n + 1)//2

# difference between total and expectedtotal is sum of missing numbers

missingSum = expectedTotal - total

numsSet = set(nums)

# Check whether expected array has the numbers from nums and report missing values

missingNum = -1

for i in range(1, n + 1):

if i not in numsSet:

missingNum = i

break

return [missingNum, missingSum - missingNum]

# algoexpert solution 1 -- O(n) time | O(n) space

def missingNumbers(nums):

solution = []

for num in range(1, len(nums) + 3):

if num not in nums:

solution.append(num)

return solution

Solution 2:

# Find missing Sum, and get the average of the 2 missing numbers.

# Divide the expected array to be <= average which is left total,

# and the rest of expected array to be right total

# Iterate through nums and find the leftSum and rightSum and get the missing numbers.

# O(n) time | O(1) space for numsSet

def missingNumbers(nums):

total = sum(nums)

n = len(nums) + 2

expectedTotal = n \* (n + 1)//2

# difference between total and expectedtotal is sum of missing numbers

missingSum = expectedTotal - total

avg = missingSum // 2

leftTotal = 0

rightTotal = 0

for i in range(len(nums)):

if nums[i] <= avg:

leftTotal += nums[i]

else:

rightTotal += nums[i]

expectedLeftTotal = avg \* (avg + 1)//2

expectedRightTotal = expectedTotal - expectedLeftTotal

return [expectedLeftTotal - leftTotal, expectedRightTotal - rightTotal]

# Algoexpert solution O(n) time | O(1) space

def missingNumbers(nums):

total = sum(range(len(nums) + 3))

for num in nums:

total -= num

averageMissingValue = total // 2

foundFirstHalf = 0

foundSecondHalf = 0

for num in nums:

if num <= averageMissingValue:

foundFirstHalf += num

else:

foundSecondHalf += num

expectedFirstHalf = sum(range(1, averageMissingValue + 1))

expectedSecondHalf = sum(range(averageMissingValue + 1, len(nums) + 3))

return [expectedFirstHalf - foundFirstHalf, expectedSecondHalf - foundSecondHalf]

Solution 3:

# My solution using sets and set difference.

def missingNumbers(nums):

numsSet = set(nums)

expectedSet = set(range(1, len(nums) + 3))

missingList = list(expectedSet - numsSet)

if missingList[0] > missingList[1]:

missingList[0], missingList[1] = missingList[1], missingList[0]

return missingList

Algoexpert Solution using XOR -- O(n) time | O(1) space

