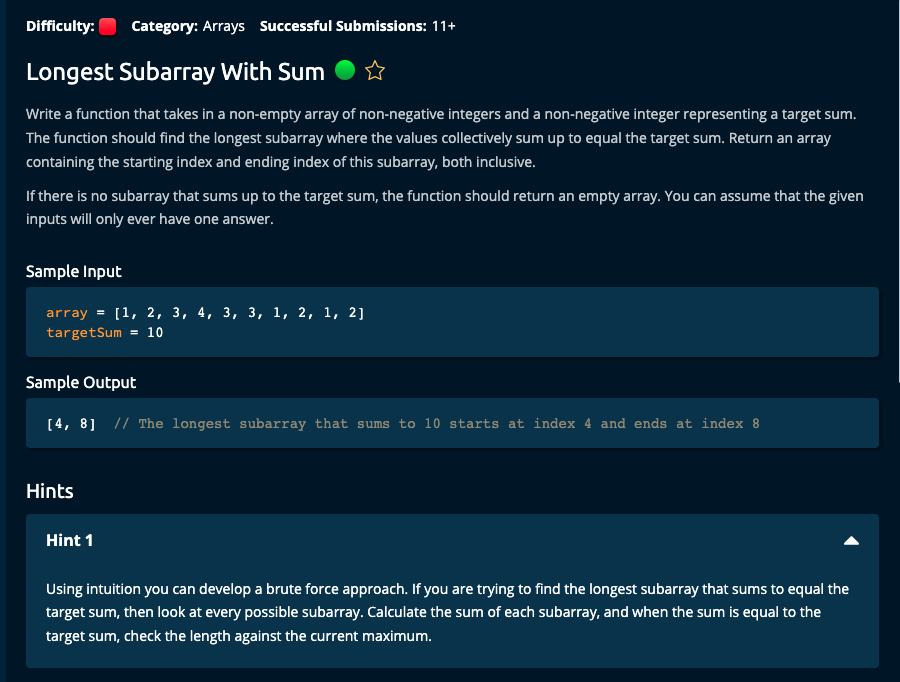
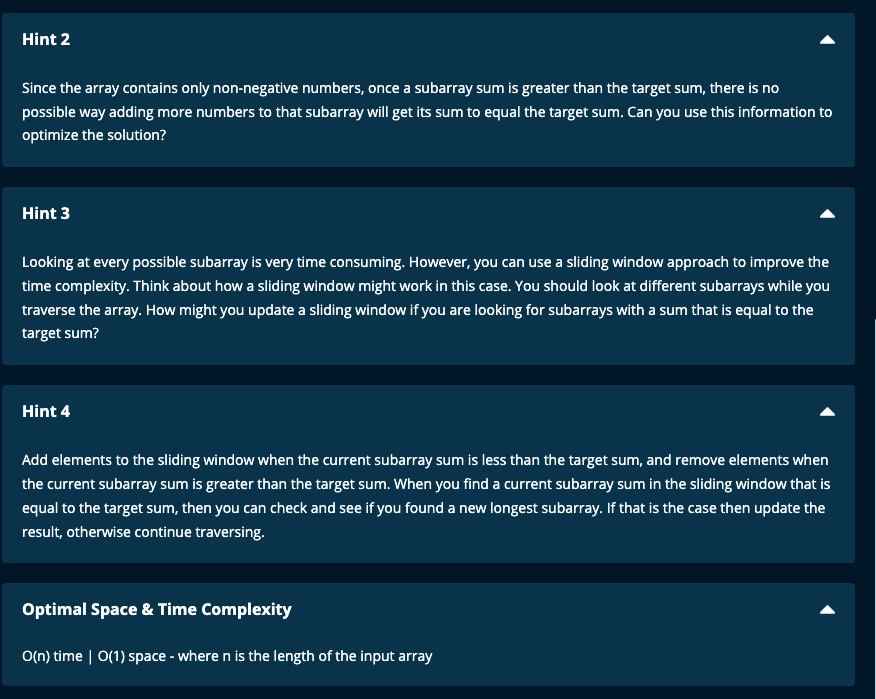
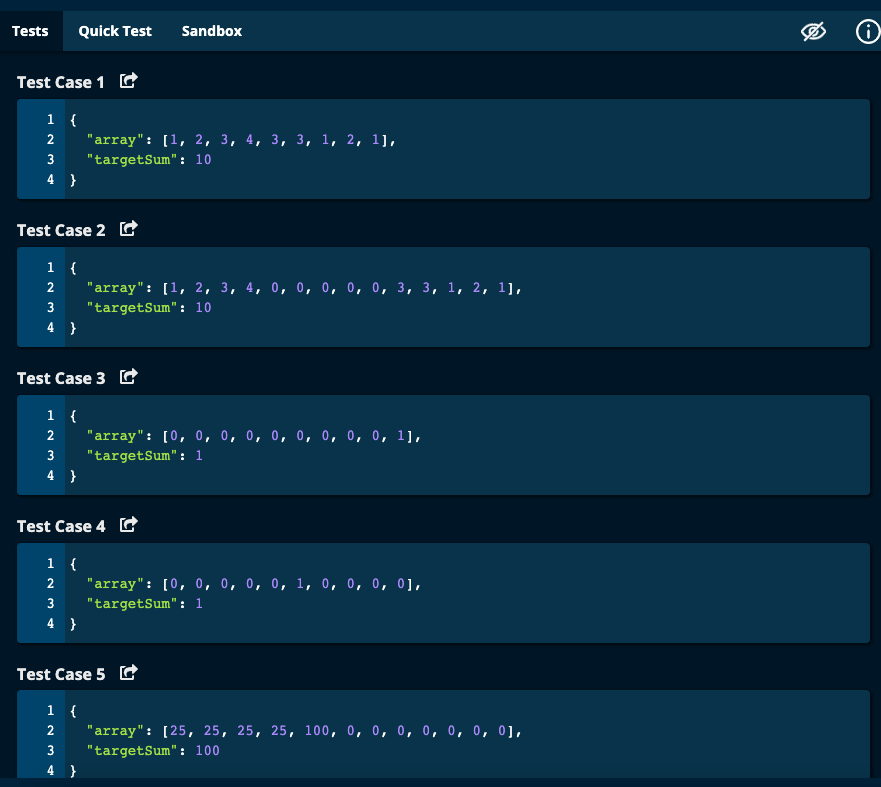
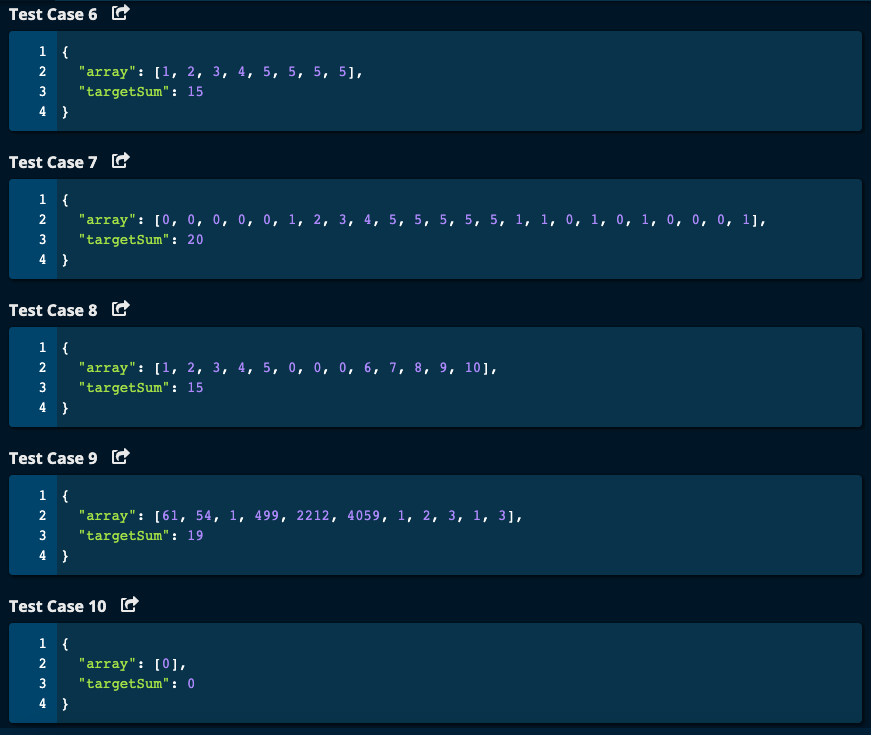
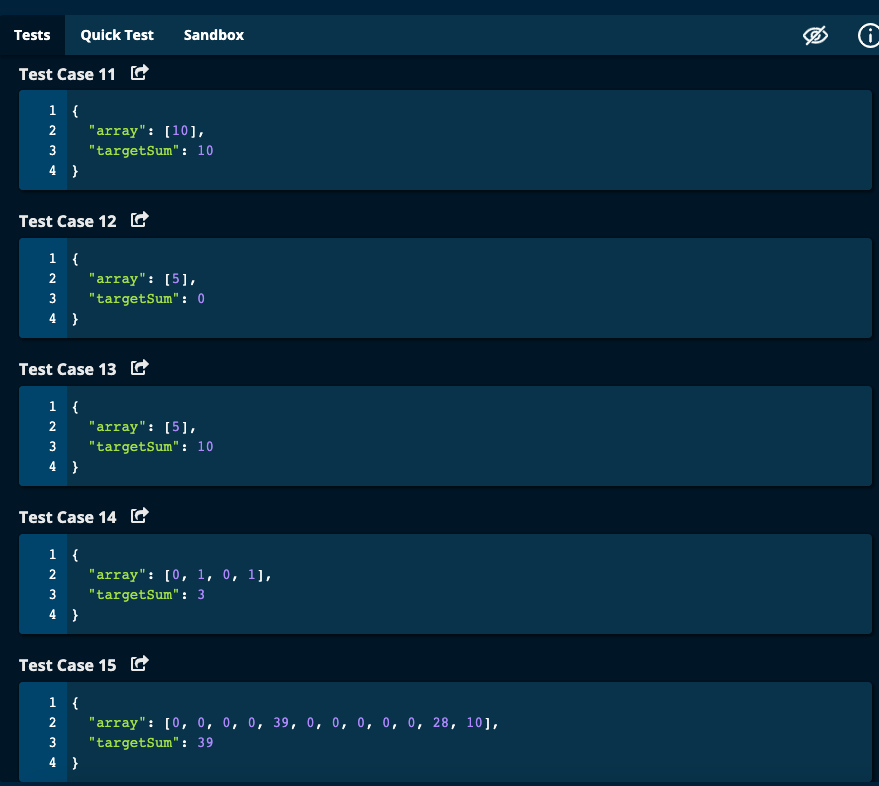
Longest Subarray with Sum (Hard)

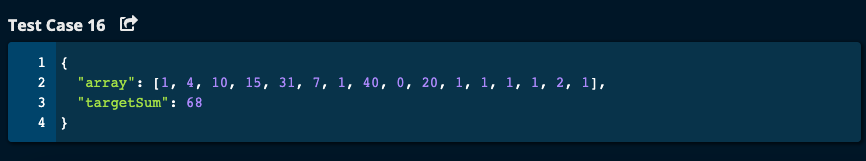












My Solution:

# My Solution -- O(n) time | O(1) space

def longestSubarrayWithSum(array, targetSum):

if len(array) == 1 and array[0] == targetSum:

return [0, 0]

startIdx = 0

endIdx = 0

total = array[0]

maxLength = 0

result = []

while startIdx <= len(array) - 2 and endIdx <= len(array) - 2:

print("While Loop begin : startIdx = ", startIdx, "endIdx = ", endIdx, "total = ", total, "maxLength = ", maxLength, "result = ", result)

while total < targetSum and endIdx <= len(array) - 2:

endIdx += 1

total += array[endIdx]

print("While Loop after endIdx move: startIdx = ", startIdx, "endIdx = ", endIdx, "total = ", total, "maxLength = ",

maxLength, "result = ", result)

if total == targetSum:

while endIdx <= len(array) -2 and array[endIdx + 1] == 0:

endIdx += 1

subarrayLength = endIdx - startIdx + 1

if subarrayLength > maxLength:

maxLength = subarrayLength

result = [startIdx, endIdx]

print("While Loop at total = targetSum : startIdx = ", startIdx, "endIdx = ", endIdx, "total = ", total, "maxLength = ", maxLength, "result = ", result)

total -= array[startIdx]

startIdx += 1

print("While Loop end: ", "startIdx = ", startIdx, "endIdx = ", endIdx, "total = ", total, "maxLength = ", maxLength, "result = ", result)

print("----------")

return result

JJ Notes:

1. If the length of array is 1 and the first element in the array is equal to targetSum, then return [0, 0].
2. Initialize startIdx, endIdx and maxLength to 0. Initialize total to array[0] and result to an empty list.
3. While startIdx and endIdx are less than or equal to length of array -2, do the following:
4. While total is less than targetSum and endIdx is less than or equal to length of array – 2, increment endIdx by 1 and total by array at the new endIdx
5. If total is equal to targetSum, while array length permits and if the next element in the array is 0, then increment endIdx by 1. When we get out of the while loop, calculate subarrayLength to be endIdx – startIdx + 1 (since we want both indexes to be included in the count).
6. If subarrayLength is greater than the maxLength then update maxLength with subarrayLength and result with a list consisting of startIdx and endIdx.
7. From running total subtract the value of array at startIdx, and move the startIdx to the next position (i.e. increment by 1).
8. Return result consisting of updated list with startIdx and endIdx.

# Algoexpert Solution: O(n) Time | O(1) Space

def longestSubarrayWithSum(array, targetSum):

result = []

startIdx = 0

endIdx = 0

total = 0

while endIdx < len(array):

total += array[endIdx]

if startIdx < endIdx and total > targetSum:

total -= array[startIdx]

startIdx += 1

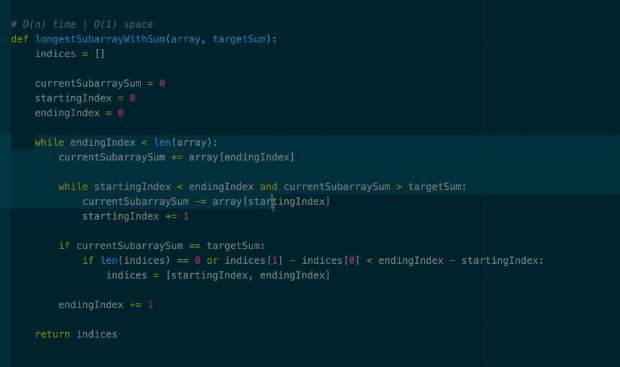
if total == targetSum:

if len(result) == 0 or result[1] - result[0] < endIdx - startIdx:

result = [startIdx, endIdx]

endIdx += 1

return result



JJ Notes:

1. Initialize result to an empty array. Initialize startIdx, endIdx, and total (i.e. running total) to 0.
2. While endIdx is less than the length of the array, do the following:
3. Update total with the value of the array at index endIdx.
4. If startIdx is less than endIdx and total > targetSum, then we deduct from total the array value at startIdx, and we increment startIdx by 1. We are moving the starting point to the next index in the array.
5. If total is equal to targetSum, then if length of result is either 0 or difference between result at index 1 and result at index 0 is less thanthe difference between endIdx and startIdx, then we update result with startIdx and endIdx.
6. Increment the endIdx by 1.
7. Finally return the result.
8. This algorithm O(n) Time | O(1) Space