**1.Find two non-overlapping subarrays with sum equal target. Find answer where the sum of length of two- sub array is minimum.**

Input : arr=[3,2,2,4,3] Target=3

Output : 2

Explanation: Only two sub-arrays have sum=3([3] and [3]). The sum of their lengths is 2.

class Solution:

    def minSumOfLengths(self, arr: List[int], target: int) -> int:

        s,res,lsize = 0,float('inf'),float('inf')

        dic = {0:-1}

        for i in range(len(arr)):

            s+=arr[i]

            dic[s] = i

        s=0

        for i,val in enumerate(arr):

            s+=val

            if s-target in dic:

                lsize = min(i-dic[s-target],lsize)

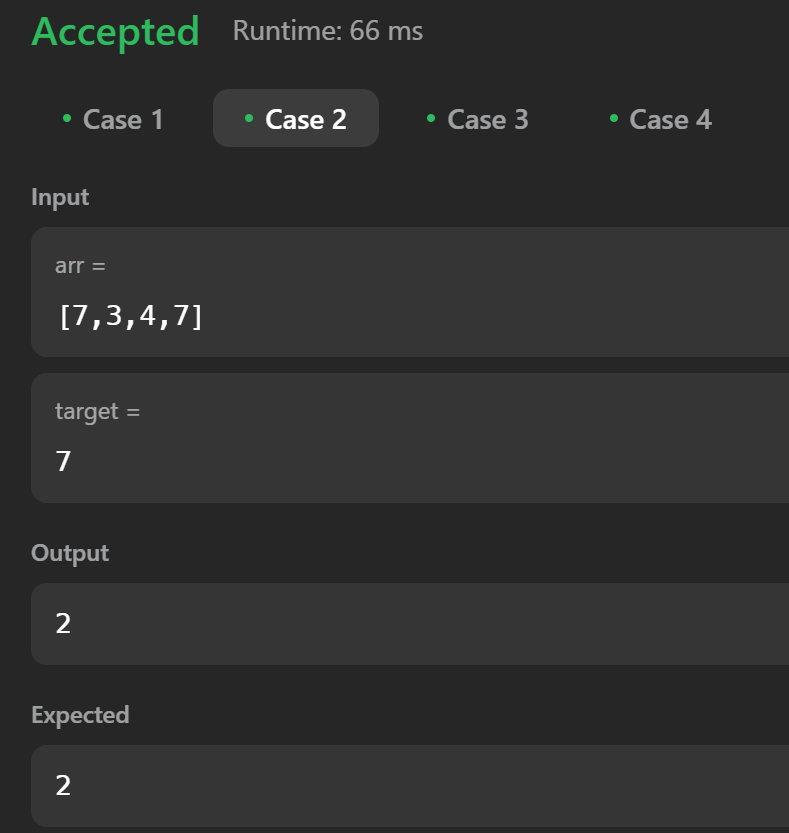
            if s+target in dic and lsize!=float('inf'):

                rsize = dic[s+target]-i

                res = min(res,lsize+rsize)

        return -1 if res==float('inf') else res

Output:



Explanation:  
Firstly a map with prefix sum of every length to the index of the sum is created. Then the array is traversed again to check if there is a subarray that matches the target sum. Then its length is used as minimum length. Then if lsize is not infinity and sum + target that is an yet to come value disjoint from left subset is present then it is rsize and then res is minimum of all such possible lsize and rsize.

**2. Concatenated Substring in s is a substring that contains all the strings of any permutation of words concatenated**

Input :s = "barfoothefoobarman", words = ["foo","bar"]

**Output:** [0,9]

**Explanation:** Since words.length == 2 and words[i].length == 3, the concatenated substring has to be of length 6. The substring starting at 0 is "barfoo". It is the concatenation of ["bar","foo"] which is a permutation of words. The substring starting at 9 is "foobar". It is the concatenation of ["foo","bar"] which is a permutation of words. The output order does not matter. Returning [9,0] is fine too.

# Solution:

def findSubstring(self, s: str, words: List[str]) -> List[int]:

    ans = []

    c = Counter(words)

    p = len(words) \* len(words[0])

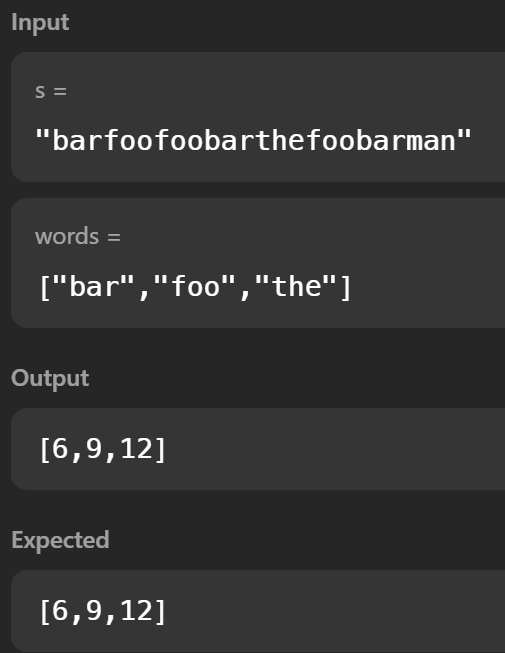
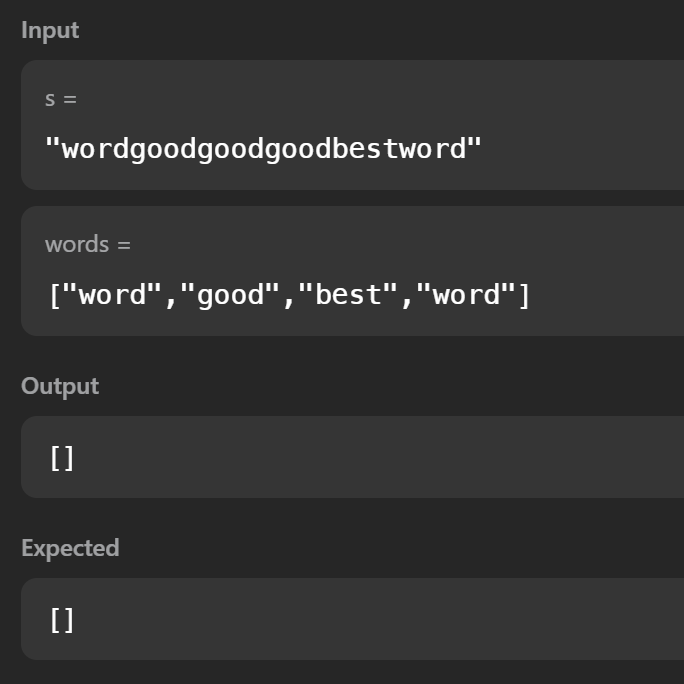
    for i in range(len(s)-p + 1):

        if Counter((map(''.join, zip(\*[iter(s[i:i+p])]\*len(words[0]))))) == c:

            ans.append(i)

    return ans

# Output:



# Explanation:

Use sliding window to check each substring of valid length and break the substring to segments of size that match the words and check frequency of words and the segments