

#!/bin/python3

import math

import os

import random

import re

import sys

class SinglyLinkedListNode:

    def \_\_init\_\_(self, node\_data):

        self.data = node\_data

        self.next = None

class SinglyLinkedList:

    def \_\_init\_\_(self):

        self.head = None

        self.tail = None

    def insert\_node(self, node\_data):

        node = SinglyLinkedListNode(node\_data)

        if not self.head:

            self.head = node

        else:

            self.tail.next = node

        self.tail = node

def print\_singly\_linked\_list(node, sep, fptr):

    while node:

        fptr.write(str(node.data))

        node = node.next

        if node:

            fptr.write(sep)

# Complete the mergeLists function below.

#

# For your reference:

#

# SinglyLinkedListNode:

#     int data

#     SinglyLinkedListNode next

#

#

def mergeLists(head1, head2):

    # Dummy node to start merged list

    dummy = SinglyLinkedListNode(0)

    current = dummy

    # Traverse both lists

    while head1 and head2:

        if head1.data <= head2.data:

            current.next = head1

            head1 = head1.next

        else:

            current.next = head2

            head2 = head2.next

        current = current.next

    # Append the remaining elements of the non-empty list

    if head1:

        current.next = head1

    elif head2:

        current.next = head2

    return dummy.next  # Return head of merged list

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    tests = int(input())

    for tests\_itr in range(tests):

        llist1\_count = int(input())

        llist1 = SinglyLinkedList()

        for \_ in range(llist1\_count):

            llist1\_item = int(input())

            llist1.insert\_node(llist1\_item)

        llist2\_count = int(input())

        llist2 = SinglyLinkedList()

        for \_ in range(llist2\_count):

            llist2\_item = int(input())

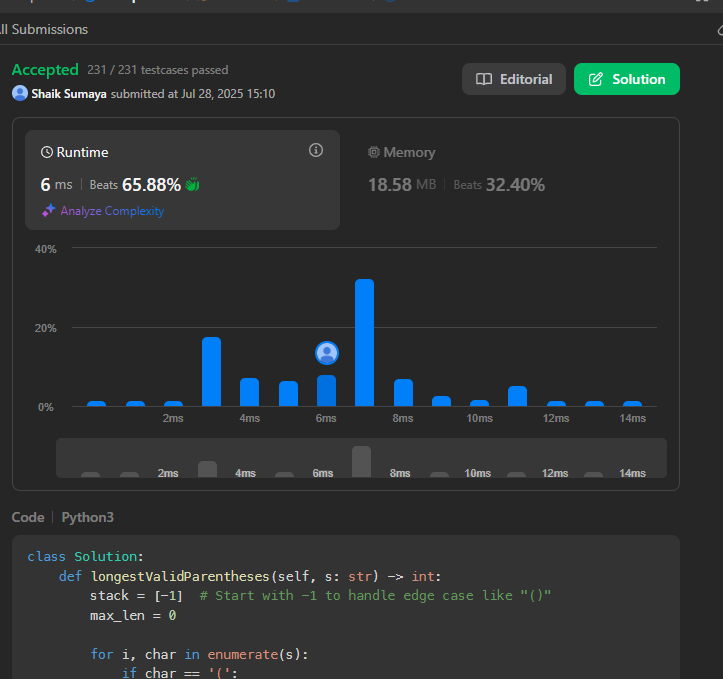
            llist2.insert\_node(llist2\_item)

        llist3 = mergeLists(llist1.head, llist2.head)

        print\_singly\_linked\_list(llist3, ' ', fptr)

        fptr.write('\n')

    fptr.close()



class Solution:

def longestValidParentheses(self, s: str) -> int:

stack = [-1] # Start with -1 to handle edge case like "()"

max\_len = 0

for i, char in enumerate(s):

if char == '(':

stack.append(i)

else: # char == ')'

stack.pop()

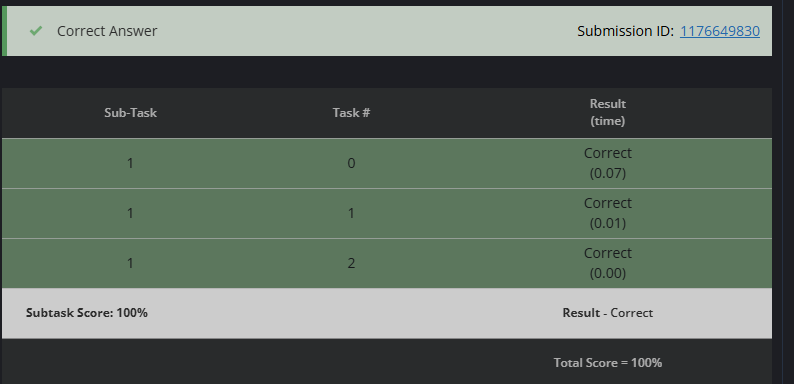
if not stack:

stack.append(i) # Reset base index

else:

max\_len = max(max\_len, i - stack[-1])

return max\_len



# cook your dish here

def count\_ways\_to\_split(N, A):

total\_sum = sum(A)

if total\_sum % 3 != 0:

return 0

part\_sum = total\_sum // 3

prefix\_sum = 0

count\_part\_sum = 0

ways = 0

for i in range(N - 1):

prefix\_sum += A[i]

if prefix\_sum == 2 \* part\_sum:

ways += count\_part\_sum

if prefix\_sum == part\_sum:

count\_part\_sum += 1

return ways

# Input Reading

N = int(input())

A = list(map(int, input().split()))

print(count\_ways\_to\_split(N, A))