

#!/bin/python3

import math

import os

import random

import re

import sys

#

# Complete the 'equalStacks' function below.

#

# The function is expected to return an INTEGER.

# The function accepts following parameters:

#  1. INTEGER\_ARRAY h1

#  2. INTEGER\_ARRAY h2

#  3. INTEGER\_ARRAY h3

#

def equalStacks(h1, h2, h3):

    def cumulative\_heights(stack):

        cum\_heights = []

        total = 0

        for h in reversed(stack):

            total += h

            cum\_heights.append(total)

        return set(cum\_heights)

    # Get all possible heights of the three stacks

    s1 = cumulative\_heights(h1)

    s2 = cumulative\_heights(h2)

    s3 = cumulative\_heights(h3)

    # Find common heights and return the max one

    common = s1 & s2 & s3

    return max(common) if common else 0

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

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

    first\_multiple\_input = input().rstrip().split()

    n1 = int(first\_multiple\_input[0])

    n2 = int(first\_multiple\_input[1])

    n3 = int(first\_multiple\_input[2])

    h1 = list(map(int, input().rstrip().split()))

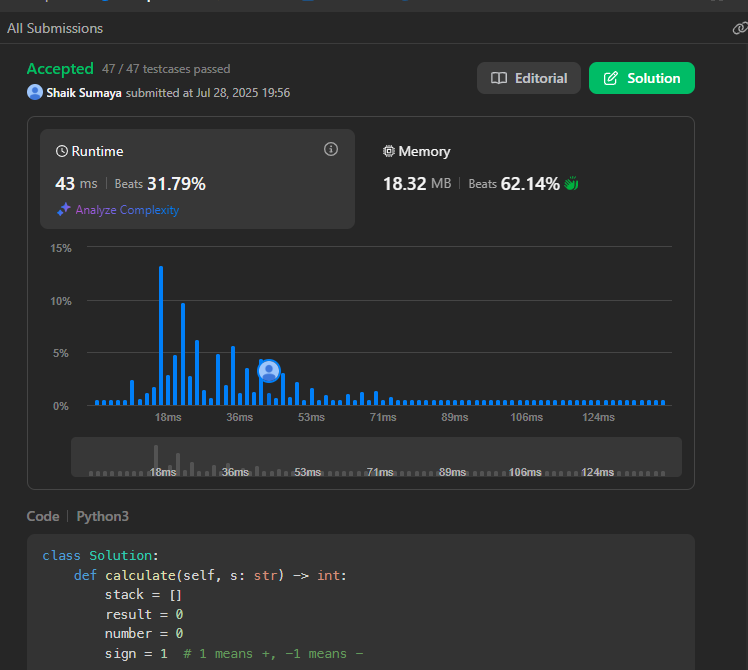
    h2 = list(map(int, input().rstrip().split()))

    h3 = list(map(int, input().rstrip().split()))

    result = equalStacks(h1, h2, h3)

    fptr.write(str(result) + '\n')

    fptr.close()



class Solution:

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

stack = []

result = 0

number = 0

sign = 1 # 1 means +, -1 means -

i = 0

while i < len(s):

char = s[i]

if char.isdigit():

number = 0

while i < len(s) and s[i].isdigit():

number = number \* 10 + int(s[i])

i += 1

result += sign \* number

continue # skip i += 1 since we already moved

elif char == '+':

sign = 1

elif char == '-':

sign = -1

elif char == '(':

# Push the current result and sign onto the stack

stack.append(result)

stack.append(sign)

result = 0

sign = 1

elif char == ')':

# Pop sign and previous result from the stack

prev\_sign = stack.pop()

prev\_result = stack.pop()

result = prev\_result + prev\_sign \* result

# Ignore spaces

i += 1

return result



def chef\_set\_problem(Q, S1, A, B):

MOD = 2\*\*32

s = S1

seen = set()

total = 0

for \_ in range(Q):

x = s >> 1 # x = s // 2

if s & 1: # Odd = insert

if x not in seen:

seen.add(x)

total += x

else: # Even = erase

if x in seen:

seen.remove(x)

total -= x

s = (A \* s + B) % MOD

print(total)

Q, S1, A, B = map(int, input().split())

chef\_set\_problem(Q, S1, A, B)