DAY-47

def nonDivisibleSubset(k, s):

# Count remainders

remainder\_count = [0] \* k

for num in s:

remainder\_count[num % k] += 1

# Start with at most one from remainder 0

result = min(remainder\_count[0], 1)

# Check pairs of remainders

for r in range(1, (k // 2) + 1):

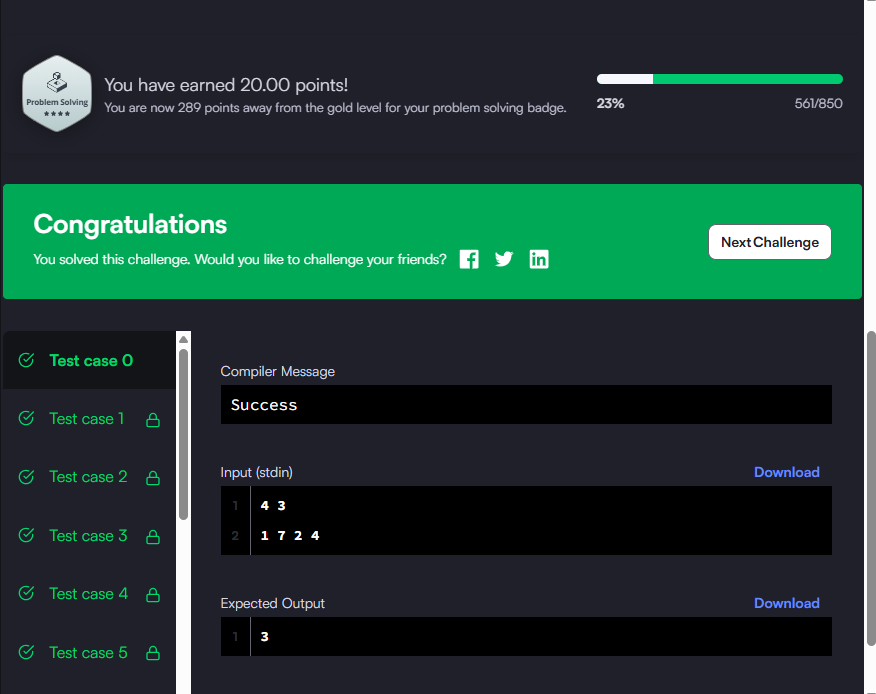
if r == k - r: # special case: when r and k-r are the same

result += 1

else:

result += max(remainder\_count[r], remainder\_count[k - r])

return result



class Solution(object):

def isMatch(self, s, p):

"""

:type s: str

:type p: str

:rtype: bool

"""

m, n = len(s), len(p)

dp = [[False] \* (n + 1) for \_ in range(m + 1)]

# Empty string matches empty pattern

dp[0][0] = True

# Handle patterns like a\*, a\*b\*, a\*b\*c\*

for j in range(2, n + 1):

if p[j-1] == '\*':

dp[0][j] = dp[0][j-2]

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

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

if p[j-1] == '.' or p[j-1] == s[i-1]:

dp[i][j] = dp[i-1][j-1]

elif p[j-1] == '\*':

# zero occurrence of preceding char

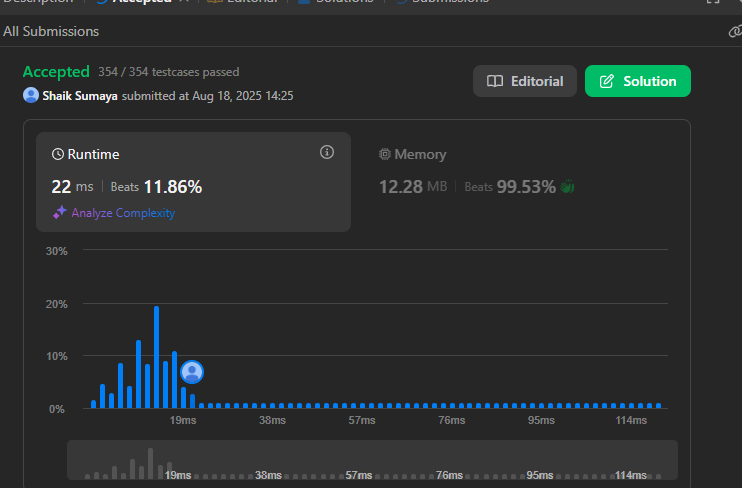
dp[i][j] = dp[i][j-2]

# one or more occurrences

if p[j-2] == '.' or p[j-2] == s[i-1]:

dp[i][j] |= dp[i-1][j]

return dp[m][n]



for \_ in range(int(input())):

vertices,sources,sinks=map(int,input().split())

common=sources+sinks-vertices

if common>0:

vertices=vertices-common

sources=sources-common

sinks=sinks-common

a=vertices\*(vertices-1)//2

b=sources\*(sources-1)//2

c=sinks\*(sinks-1)//2

print(a-b-c)

