**VALID SUBSEQUENCE:**

The problem statement says there are 2^N subsequence of a given sequence of length n, first we will wort the array there are 3 cases

1. All the k elements are different
2. The elements before the maximum element in k size seq. are duplicate
3. The maximum element of the k size seq. is duplicate

For the first case output would be 1 as no other combination is possible.

For second case also output is 1 as elements could be shuffled within but no other elements could replace it

For third case there are 3 possibilities:

1. The maximum element let’s say M is duplicate and there are no elements outside the k size sequence with value M.
2. There is only 1 M in k size seq. and ‘m’ number of M’s outside k.
3. There are ‘n’ M’s in k and ‘m’ elements outside k.

For first still output is 1 because still elements could be shuffled inside the k size

For second the output is m as m M’s could be selected.

For third output is m+nCn .

Code:

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

n,k = [int(x) for x in input().split()]

ar = [int(x) for x in input().split()]

ar.sort()

le=ar[k-1]

lec=0

lec\_in\_k=0

for i in range(k-1,-1,-1):

if ar[i]!=le:

break

lec+=1

lec\_in\_k=lec

for i in range(k,n):

if ar[i]!=le:

break

lec+=1

import math

n=math.factorial(lec)

d=math.factorial(lec\_in\_k)\*math.factorial(lec - lec\_in\_k)

print(n//d)

**Non dividible subsets:**

There are possibilities of k even and odd

The trick here is to do mod of all elements (mod k), after that values available would be 0 to k-1

Let’s say k=5

Rem = 0,1,2,3,4 taking 2 pointers i,j

i starts from 1 and j from k-1 here 4

when you select 1 and 4 if uh take both elements then you will end up with sum=5 so uh need to select maximum count from 1 and 4

Similarly, with 2 and 3.

Lastly only one element with mod k=0 could be selected

Now lets assume and even k =6

Rem = 0,1,2,3,4,5 here also similar procedure 1 and 5, 2 and 4 , 1 element of modk=0 but what about 3? When you take 3 more than once it becomes in valid as 3+3=6 and it is divisible by 6. So only 1 element could be considered i.e. if k is even then elements with rem=k/2 could be considered only once.

Code:

from collections import Counter

n,k=[int(x) for x in input().split()]

a=[int(x)%k for x in input().split()]

dct=dict(Counter(a))

for i in range(k):

if i not in dct:

dct[i]=0

c=0

if k%2:

i=1

j=k-1

while i<=j:

c=c+ max(dct[i], dct[j])

i+=1

j-=1

if dct[0]:

c+=1

else:

i=1

j=k-1

while i<j:

c=c+ max(dct[i], dct[j])

i+=1

j-=1

if dct[0]:

c+=1

if dct[k//2]:

c+=1

if c:

print(c)

else:

print(1)

An edge case is where only one element is given so irrespective of k answer is as no two elements could be added.

**addition\_a\_headache:**

When I faced this question I didn’t had idea of how online compilers read input. Unlike offline compilers installed in pc the online compilers read from files, so the stop condition becomes that when input is reading file but there is no text left it raises EOFError.

Code:

while True:

try:

a,b=[int(x) for x in input().split()]

print(a+b)

except EOFError:

break

**4th power:**

A little math problem. PS is whether a number could be expressed as difference of 2 numbers to the power of 4 c = a4 -b4, using indices it could be converted to c = (a-b)\*(a+b)\*(a2 +b2 ).

Again odd even game a ,b could be odd or even.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a b | a-b | a+b | a2+b2 | c |
| O O | E  Divisible by 2  Lets say 2x | E  Divisible by 2  Lets say 2y | E  Divisible by 2  Lets say 2z | E  Divisible by 8  As 2x\*2y\*2z=8xyz |
| O E | O | O | O | O |
| E O | O | O | O | O |
| E E | E  Divisible by 2  Lets say 2x | E  Divisible by 2  Lets say 2y | E  Divisible by 2  Lets say 2z | E  Divisible by 8  As 2x\*2y\*2z=8xyz |

From above table it is clear that for a number to be expressed as a4 -b4 either it should be odd or divisible by 8.

Code:

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

n=int(input())

if n%8==0:

print("YES")

elif n%2:

print("YES")

else:

print("NO")

**Maximum Draw:**

Easy one! If there are n pairs of socks n+1 trails will give a pair.

Code:

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

print(int(input())+1)

**Repeated String:**

Here a string given whose chars are repeated n times and you need to find count of ‘a’.

Brute Force would be repeating the string and then counting ‘a’, eventually result TLE.

Now when the chars are repeated ‘n’ times if count of ‘a’ in given string is m then in length of string repeated for n-1 times would be m\*(n-1).

The last string may or may not be an entire string so number of char last string would be n%len(given\_string)

So, you need to find count of ‘a’ last string.

Code:

s=input()

ls=len(s)

n=int(input())

i=0

c=0

while i<ls:

if s[i]=="a":

c+=1

i+=1

tn=n%ls

n1=n//ls

c=c\*n1

if tn:

for i in range(tn):

if s[i]=="a":

c+=1

print(c)