day 7 two pointers & mathematics

(i) sieve of eratho cenes.

0 and -1 are not prime numbers ← prime Number : number with only 2 factors, 1 and the number itself

factors. number which gives modulo 0, when it divides X.

↓

2 is a factor of 6 since

6 % 2 == 0.

* finding whether a number is prime or not

↓

loop from 2 to $\sqrt{n}$, if any number divides n completely, then n is not prime.

↓

since for a factor 'A', where N = A * B, if $A < \sqrt{N}$, then $B > \sqrt{N}$. if A is checked, therefore corresponding B is also checked.

* sieve : array ranging from 0 to N, all prime number indices are marked by true. and composite number indices are marked by false.

↓

so we initiate a bool array of req. size. and mark all values from 2 to the range-1 as true. Now looping from 2 to range-1. if the current index is true, we check if it is prime &

then mark its multiple as false and so on

∴ time complexity

no. of operations performed here :

(i) $n \log n$

* (ii) $n \log \log n$ (tighter bound)

$$\frac{N}{2} + \frac{N}{3} + \frac{N}{5} + \ldots + \frac{N}{P_i} \leq \frac{N}{1} + \frac{N}{2} + \frac{N}{3} + \frac{N}{4} + \ldots$$

$$\leq N \sum_{i}^{N} \frac{1}{i}$$ → replacing summation by integration

$$\leq N \log N$$

(ii) $n \log \log N$ → Merten's theorem tighter bound

$i = 2$ to $\sqrt{N}$

$$\frac{N}{2} + \frac{N}{3} + \frac{N}{5} + \frac{N}{7} + \frac{N}{11} + \ldots$$

$$= N\left(\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \ldots\right)$$

↓

$\sqrt{N}$ terms

according to Merten's theorem, the sum of first N terms of such a series is

$\log \log N$.

↓

Here $N = \sqrt{N}$, ∴ $\log \log N^{1/2}$

$\frac{1}{2} \log \log N$

↓

$O(N \log \log N)$

Q. Given i & j, find the sum of factors of k where k are all numbers between i & j.

↓

$\sum_{k=i}^{j}$ num-factors(k). $O(n^2)$

↓

for each term rather than marking false, we increase the count at each multiple of them it is done at all numbers & not only prime.

↓

then construct a prefix sum array. to make the query of the order of $O(1)$

↓

prefix sum : cdf for i, then subtract A[i] from A[j], to get the sum of elements from i to j.

↓

Another method : multiplication principle.

↓

The number of factors for any number is ~~sum of~~ multiplication of $(1+n_i)$ where n_i is the ith prime factor's power.

↓

Q create a sieve with the max prime factor for each number.

(ii) segmented sieve

↓

when the given range of R is $> 10^6$. then creating a conventional sieve won't be possible and there comes the concept of segmented sieve., when we are given a range between which the primes are supposed to be generated, where the R-L for the range $\leq 10^6$.

↓

looping from 2 to $\sqrt{R}$ and marking all multiples in range L to R as false.

↓

a sieve from range 2 to $\sqrt{R}$ to find primes. and for all primes in this range, mark multiples in the given range as false.

↓

(This will return us the required array. giving us all prime in the given range.)

Scanned by CamScanner

* rank of permutation of letters (all letters are unique)

abc

↓

n! permutations

↓

abc
acb
bac
bca
cab
cba

given any permutation of the above string, we need to tell or return their rank

↓

(solving recursively)

rank of bca → 4

↓

rank of 1st letter as compared to other letters of the string.

(rank-1) * (n-1)! + (rank of ca)

↓

for bca, rank = 2,

1 * 2 + (rank of ca) 1 * 2 + 2 = 4

↓

for ca, rank = 2

1 * 1 + (rank of a) = 1 = 2

↓

for ac, rank = 1 1 * 2 + 1 = 3

0 * 1 + (rank of c) = 1 = 1

abcd 1
abdc 2
acbd 3
acdb 4
adbc 5
adcb 6
bacd 7
badc 8
bcad 9
bcda 10
bdac 11
bdca 12
cabd 13
cadb 14
cbad 15 *
cbda (16)

* cbda

rank of c = 3

2 * 3! + rank of bda

4 ↓

1 * 2! + rank of da

2 ↓

1 * 1! + rank of a

↓

1

3! = 6

2 * 6 | 4

12 | 4 = 16

4 * 3 * 2

24

= (16)

when there are no duplicates, then for any given string with a given starting character $(n-1)!$ permutations are given.

↓

So once the rank of the starting element is calculated, we can find the permutations before that using

$(rank-1) * (n-1)!$

↓

elements before it and their corresponding permutation sum.

(iii) two pointes concept

↓

(Window in the array sort of questions)

Q. Given a sorted array, you need to find if there are two elements in the given array whose sum is equal to a given number k.

i.e. $A[i] + A[j] = k$, where $i \neq j$.

approach 1 ↓

one ptr at start i, one ptr at end j

↓

maintain sum of no.s at the two addresses, if sum < k, increment i else decrement j. Do this till $i <= j$.

$O(n)$ ← $O(1)$

approach 2 : for each element find $k - A[i]$ element in the array.

↓

$O(n \log n)$, $O(1)$

Q. Now A[i] - A[j] = K.

• put both ptrs at 0, and 1 and check diff.
if diff is less than k, then move i, else
move j, till j < N, if i >= j at
any point, then make j = i+1.

↓

* if the same approach as the last question
then ambiguity arised how to make
a decision to move in certain direction,
since movement from both direction leads to
reduction in difference.