1. Prime factorization, we determine root of that number frequently because we need the divisors of that changed number and we can gain it by running the loop till root of n.

And complexity is sum of all the powers. Cz the inner loop is running that times.

## 2.Spoj COMDIV - Number of common divisors

## Here I have to 1st calculate the gcd, and the find the number of divisors of gcd.

## Cz,

## We can think the gcd as largest square that can fill the a x b recatangle evenly, that is the largest common divisor, but the other smaller divisors? What are they? They are the divisors of that gcd, cz if we need another smaller square which will fill the a x b rectangle evenly, that square must fill the gcd square evenly too, or it will not be able to fill the main rectangle.

## Okay now if I want to pre calculate the number of divisors in a sieve,, then there a problem rises, that is when there a divisor greater than the sqrt of n , we can add it at the end of prime factorization by checking the condition if(n!=1) but for pre calculating I will create another array and keep the multiplication of the prime’s powers which will exclude that extra number, and at the end of that I will run a loop from 1 to 10^6 and check if b[i] != I then multiple by two.

## Topic : bitwise:

## Note 1: there is no problem for running the loop 10^8 with some optimization ,

## I use arrays size of 10^7 int a[10000009]; N=100000009; and ran the loop till 10^8

## But I have to exclude the even numbers that’s why u didn’t got tle, cz u brought down it into half,

## 1. I will use biwise operator , and num/32 == num>>5 not num>>32 -\_- bal khankir chawal -\_-

## Secondly, I don’t need to do the calculation for the even numbers, I will only do it for the odd ones.

## for(i=3; i<=10009; i+=2) for(i=3;i<=100000009;i+=2) no even

## but

## for (i = 4; i <= 100000009; i += 2) {

## a[i/32] |= (1 << (i %32));

## }

## And for(i=3;i<=100000009;i++) for even , thought theres no need.

## Use this optimization , though I am not clear enough about it,,,

## for( j = i\*i; j <= N; j += 2\*i )

## How the check and the set function worked is also important, it took the value stored in that index and inserted 1 in expected position, for setting we did OR and for checking we did AND. And we did it by 1<<pos;

## bool check(int N,int pos){return (bool)(N & (1<<pos));}

## int Set(int N,int pos){ return N=N | (1<<pos);}

## multiplication of 2 , 1<<pos,, divide ,,, num>>5;

**1220 - Mysterious Bacteria**

In this problem the input was said to be signed 32 bit, so it could be – INT\_MAX . and for neg numbers , I had to control the sign, I thought if I get even numbers then I will divide just once . but the division can also result to a even number.

\*\* when it is said that signed 32, we must take long long

**\*\*\*\*\*sieve use differently \*\*\*\*\***

## NFACTOR - N-Factorful

Here I had to find the numbers which had  it has exactly **n** distinct prime factors.

So , with others help, I run a loop from 2 to 10^6 of I where in sieve I use to 10^3, and increased by 1 for all the multiples of a definite prime, then I stored the values in a 2D array,

The rows are 1-10 since I need to use only 10 primes till 10^6, then I ran a loop from 1 – 10^6 for each row and checked the condition and stored the values,

void sieve()

{ int i,j;

for(i=2;i<=1000009;i++)

{

if(num\_div[i]==0){

for(j=1; i\*j<=1000009;j++)

{

num\_div[i\*j]++;

}

}

}

for(i=1;i<=10; i++)

{

for(j=2;j<=1000009;j++)

{

a[i][j]= a[i][j-1] + (bool)(num\_div[j]==i);

}

}

}

**1077 - How Many Points?**

**Here I have use the equation for given two points.   
that is ( y- y1)\*(x1-x2) / (y1-y2) . (y-y1) have to be divisible by the deno. And range will be y1-y1 to y1- y2. And it will happen for x. so I don’t need to calculate for x. cz there will be no case where x is integer and y is double, cz we are adding and multiplying real numbers. So , if I got the LHS fractional the RHS will also be frac and vice versa.**

**1197 - Help Hanzo**

**This problem is solve with segmented sieve,**

for(i=0; prime[i] <= root; i++)

{

j=(a/prime[i] + (bool)(a%prime[i]!=0) ) \* prime[i];

for( ; j<=b; j+= prime[i])

{

if(j==prime[i] || j==1 ) continue;

num2[j-a]=0;

}

}

Here I did num2[ j –a ] just transferred the segment to tha root. Nothing else.

**1215 - Finding LCM**

This was a good problem, I couldn’t solve it last night.

\*\* Here I ran prime factorization of two numbers together at a time, and it worked !! :D

\*\*1st I checked if the prime divides both the numbers , then cheked if I divides only my number, and at the end , when n!=1 I did the same thing again.

**1234 - Harmonic Number**

This was an very interesting and new problem. In this problem,

\*I had to run a loop till 10^8 and determine the harmonic series

\*\* I pre calculated the every 1000 numbers in an index.

\*\* when I need ans for n , total =a[ n/1000 -1] + a single loop from (n/1000)+1 to n;

This way I can store a huge amount of data