Number Theory - 2

Important Properties

(a-b)%k = (x-y) Then (a-x)%k = (b-y)%k

Proof (Just for your understanding)

LHS =

(a-b) = N*k + (x-y);
(a-x) = (a-b)*k + (b-y); // After reorder.
//Take modulo with k on LHS and RHS
(a-x)%k = 0 + (b-y)%k;
(a-x)%k = (b-y)%k
= RHS

GCD(a,b)

Q. Write a C++ code to calculate GCD of two numbers.

GCD-> Greatest common divisors.

12, 16.

12-> 1 2 3 4 6 12 O(Sqrt(m))

16-> 1 2 4 8 16 O(sqrt(n));

GCD(12,16) = 4

First Solution->

1.Calculate all divisors of first number

2.Calculate all divisors of second number.

3. And just find the divisor which is common to both and have max value.

Time Complexity -> O(Sqrt(m))* O(sqrt(n))=O(sqrt(m*n))

Euclidean Algorithm for GCD

gcd(a,b) = a , if (b==0)

gcd(a,b) = gcd(b,a%b) , if(b!=0)

```
int GCD(int a,int b){
    if(b==0) return a;
    return GCD(b,a%b);
}
```

Time Complexity -> O(log n)

[Fast method]

LCM
(Loweset Common Multiple)

LCM(3,4) = 12
LCM(12,16) = 48
LCM(3,9) = 9

Def. The lowest number which is divisible by both a and b.

LCM(a,b) = a*b/gcd(a,b); = (a/gcd(a,b))*b
5,6 -> 30 = 5*6
max(a,b) to a*b
a*b = gcd*lcm
Lcm = (a*b)/gcd
A,b -> order of 10^10
Lcm = (a/gcd)*b

There is also an in-built function for GCD in C++, __gcd().

```
int a,b;
cin>>a>>b;
int gcd = __gcd(a,b);
int lcm = (a/gcd)*b;
int x = __gcd(a,__gcd(b,c));
```

N -> sqrt(n); 1<=n<=10^16

Q queries are given.
In each query, you are given 1 number x, you have to find whether x is prime or not.
1<=q<=1000,1<=x<=10^6

Naive solution -> q*sqrt(x)

1<=q<=10^6, 1<=x<10^6

[Naive solution is very slow, it will give TLE]
So, we use this method called **sieve of Erasthones:**

```
bool isPrime[1000001];
// isPrime[i] = 1 if i is prime
// isPrime[i] = 0 if i is not prime

// numbers=1 2 3 4 5 6 7 8 9 10 11 12 13 14
// isPrime  = 0 1 1 0 1 0 1 0 0 0  1   0   1   0
for(int i=0;i<=1000000;i++){
    isPrime[i]=1;
}
isPrime[1]=0;
isPrime[0]=0;
```

```
for(int i=2;i*i<=1000000;i++){
    if(isPrime[i]==1){
        for(int j=i*i;j<=1000000;j+=i){
            isPrime[j]=0;
        }
    }
}
```

Time complexity: n/2 + n/3 + n/5 + n/7…. = nlog(logn)
Multiple of 2 -> 4,6,8,10,12…
Multiple of 4 -> 8,12,16….
Multiple of 3 -> 6,9,12,15….
Multiple of 6 -> 12,18,24…
Jmin = i*2,i*3,i*4….i*i
jmax<=1000000
jmin<=jmax
i*i<=1000000
i<=1000 = sqrt(10^6)

Sieve of Eratosthenes

```
isPrime[1]=0;
isPrime[0]=0;
for(int i=2;i*i<=1000000;i++){
    if(isPrime[i]==1){
        for(int j=i*i;j<=1000000;j+=i){
            isPrime[j]=0;
        }
    }
}
```

Time Complexity -> n(log(log(sqrt(n)))
Space complexity -> O(n)
Time complexity - O(q + xlog(log(sqrt(x)))

Smallest Prime Factor(SPF)

spf[i] -> smallest prime number that divides i.
(3,6,8,10)
If z is a prime number, spf[z] = z

1<=q<=10^6,1<=x<=10^6

Find the spf[x] for each query?

```
for(int i=0;i<=1e6;i++){
        spf[i] = i;
    }
    for(int i=2;i*i<=1e6;i++){
        if(spf[i]==i){
            for(int j=i*i;j<=1e6;j+=i){
                if(spf[j]==j){
                    spf[j]=i;
                }
            }
        }
    }
int n;
cin>>n;
int a[n];
```

```
for(int i=0;i<n;i++){
    cin>>a[i];
}
```

Comparator function in Set

Q. Sort a vector of pair in reverse order using a set.