

Number Theory - 1

Properties of Modulus

```
int x = -8%5; // x = -3  
-> -3,-8,-13,2  
x+=m
```

Modulus:

Non-negative remainder when x is divided by m.

It is also said as “x mod m”.

Range of “x mod m” : $[0, m-1]$

$x \% m$ -> modulus

If x is negative -> modulus = $(x \% m + m) \% m$

Q. N is a given number

Find the factorial $N \% m$

Eg. $n=5$ $m = 20$

$5! = 120$

$120 \% m = 0;$

```
long long a=1e18,b=2e18;  
int m = 1e9+7; // -> 1000000007  
int x = (a+b)%m;  
cout<<x;
```

$a > 0, b > 0$

Properties of Modulus:

1. $(a+b) \% m = ((a \% m) + (b \% m)) \% m$

2. $(a-b)\%m = ((a\%m)-(b\%m)+m)\%m$
3. $(a*b)\%m = ((a\%m)*(b\%m))\%m$
4. $(a/b)\%m = ((a\%m)*(b^{-1}\%m))\%m$

$m=5, a=9, b=14;$
 $a\%m = 4, b\%m = 4$
 $m=5, a=8, b=4;$
 $(a-b)\%m = 4$
 $a\%m = 3$
 $b\%m = 4$
 $-1\%m = -1$

Q. Find $(N!)\%m$. Where $(N! = 1*2*3*4*5*.....*n)$

N -> input

M = $1e9+7$;

$(n!\%m)?$

$N!\%m = (n*(n-1)*(n-2)*(n-3)....1)\%m = ((n\%m)*((n-1)*(n-2)...1)\%m)\%m$
 $1 \leq n \leq 1000$

```

int ans=1;
for(int i=1;i<=n;i++){
    ans = (ans%m)*(i%m);
    ans%=m;
}
return ans;

```

Q. Find (x^n) . (x raised to the power n or x^n)

Method 1: (Using a for loop)

```

int ans=1;
for(int i=1;i<=n;i++){

```

```
    ans*=x;
}
```

Method 2: (Using a recursive function)

```
int fun(int x,int n) //fun(x,n) -> x^n;
{
    if(n==0){
        return 1;
    }
    return x*fun(x,n-1);
}
```

fun(2,5) -> fun(2,4)-> fun(2,3)-> fun(2,2)->fun(2,1)-> fun(2,0)

Time complexity of both methods is $O(n)$

Binary Exponentiation

$2^8 = (2^2)^4 = 4^4 = (4^2)^2 = 16^2 = (16^2)^1 = 256 \rightarrow \log n$

$x^n \rightarrow \log n$

$2^5 = 2 * 2^4$

x^n

Q. Find $(x^n) \bmod m$

Method 1: Using a recursive function

```
int binaryExponentiation(int x,int n,int m) // x^n O(logn)
{
    if(n==0){
        return 1;
    }
    if(n%2==0){
        return binaryExponentiation(((x%m)*(x%m))%m,n/2,m);
    }
}
```

```

    return
    ((x%m)*binaryExponentiation(((x%m)*(x%m))%m,(n-1)/2,m)%m)%m;
}

```

Iterative (Using a loop) :-

```

int binaryExponentiation(int x,int n,int m)    // O(logn)
{
    int res=1;
    while(n!=0){
        if(n%2==1){
            res = ((res%m)*(x%m))%m;
        }
        x = ((x%m)*(x%m))%m;
        n/=2;
    }
    return res;
}

```

Time complexity of both methods is $O(\log N)$ here, which works very fast.

Even, for large numbers like $N=10^{20}$, $\log N$ has a very small value.

Prime Number

Definition

Prime numbers are those numbers that are divisible by only 1 and the number itself. i.e the number of divisors should be 2.

Q. 2,3,4,6,7,8,9
P P N N P N N

Q. Write a C++ code to check whether the given number is prime or not.

```
int n;  
cin>>n;  
int divisors=0;  
for(int i=1;i<=n;i++){ O(N)  
    if(n%i==0) {  
        // n is divisible by i  
        divisors=divisors+1;  
    }  
}  
if(divisors==2) cout<<"The given number is a prime  
number"<<endl;  
else cout<<"The given number is not a prime number"<<endl;
```

Time Complexity - $O(n)$

[This was a slow method]

Important Key Point

Consider a natural number N

If i is a divisor of N .

Then, (N/i) is also a divisor of N .

e.g

$N=6$;

2 is divisor of N bcz $(6\%2==0)$;

$6/2 = 3$ is also divisor of N bcz $(6\%3==0)$;

$N \leq 10^{10}$ [Worst case of (N)]

i is divisor of N then (N/i) is also a divisor of N.

N=12 -> 1 2 3 4 6 12
1 to $\leq \sqrt{N}$

**Property: If you have got a divisor $> \sqrt{N}$,
Then there must be a divisor that is less than
 \sqrt{N}**

$N=a*b$;
 $a \leq \sqrt{N}$;
 $b \geq \sqrt{N}$;

Fast method to check if n is prime

```
int n;  
cin>>n;  
int divisors=0;  
for(int i=1;i<=sqrt(n);i++){  
    if(n%i==0) {  
        // n is divisible by i  
        int first_Divisor=i;  
        int second_Divisor=(N/i);  
        if(first_Divisor!=second_Divisor) divisors+=2;  
        else divisors++;  
    }  
}  
if(divisors==2) cout<<"The given number is a prime  
number"<<endl;  
else cout<<"The given number is not a prime number"<<endl;
```

Time complexity of this method: $O(\sqrt{N})$

[Faster than previous method]

$\sqrt{16}=4$

16 \rightarrow 1 2 4 8 16

$i=4 \ i*i=16$

$10^6 \rightarrow n/2 == 5*10^5$

$\sqrt{N} == 10^3$

Some important in-built functions

1. $\text{pow}(n, x)$ \Rightarrow Finds n^x in $O(\log n)$

2. \sqrt{n} \Rightarrow Finds square root on n .

Caution: If the numbers are small then only use $\text{pow}()$ function, otherwise use the Binary Exponentiation method to calculate power.