



Signup and get free access to 100+ Tutorials and Practice Problems

Start Now

## ← Notes

### ▲ Number Theory - 1

197    Number Theory    Primes    CodeMonk

#### Introduction:

Problems in competitive programming which involve Mathematics are usually about number theory, or geometry. If you know number theory, that increases your ammo heavily in solving a lot of tougher problems, and helps you in getting a strong hold on a lot of other problems, too.

Problems in competitive programming require insight, so just knowing some topics is not enough at all. All of the problems requires more or less math tough. For instance, solving large systems of equations and approximating solutions to differential equations.

#### Modulo:

Modulo operation gives the remainder after division, when one number is divided by another. It is denoted by % sign.

Example:

We have two numbers 5 and 2, then  $5\%2$  is 1 as when 5 is divided by 2, it leaves 1 as remainder.

Properties:

Some of the properties of modulo are:

$$(a+b)\%c = (a\%c + b\%c)\%c.$$

$$(a * b)\%c = ((a\%c) * (b\%c))\%c.$$

Example:

Let's say  $a = 5$ ,  $b = 3$ ,  $c = 2$ .

Then:

$$1) (5+3)\%2 = 8\%2 = 0.$$

$$\text{Similarly } (5\%2 + 3\%2)\%2 = (1 + 1)\%2 = 0.$$

$$2) (5 * 3)\%2 = 15\%2 = 1.$$

$$\text{Similarly } ((5\%2) * (3\%2))\%2 = (1 * 1)\%2 = 1.$$

#### Greatest Common Divisor

Greatest Common Divisor (GCD) of two or more numbers is the largest positive number that divides all the numbers which are being taken into consideration.

?

For example: GCD of 6, 10 is 2 since 2 is the largest positive number that divides both 6 and 10.

### Naive Approach:

We can traverse over all the numbers from  $\min(A, B)$  to 1 and check if the current number divides both A and B or not. If it does, then it will be the GCD of A and B.

```
int GCD(int A, int B) {
    int m = min(A, B), gcd;
    for(int i = m; i > 0; --i)
        if(A % i == 0 && B % i == 0) {
            gcd = i;
            return gcd;
        }
}
```

**Time Complexity:** Time complexity of this function is  $O(\min(A, B))$ .

### Euclid's Algorithm:

Idea behind Euclid's Algorithm is  $\text{GCD}(A, B) = \text{GCD}(B, A \% B)$ . The algorithm will recurse until  $A \% B = 0$ .

```
int GCD(int A, int B) {
    if(B==0)
        return A;
    else
        return GCD(B, A % B);
}
```

Let us take an example.

Let  $A = 16$ ,  $B = 10$ .

$\text{GCD}(16, 10) = \text{GCD}(10, 16 \% 10) = \text{GCD}(10, 6)$

$\text{GCD}(10, 6) = \text{GCD}(6, 10 \% 6) = \text{GCD}(6, 4)$

$\text{GCD}(6, 4) = \text{GCD}(4, 6 \% 4) = \text{GCD}(4, 2)$

$\text{GCD}(4, 2) = \text{GCD}(2, 4 \% 2) = \text{GCD}(2, 0)$

Since  $B = 0$  so  $\text{GCD}(2, 0)$  will return 2.

**Time Complexity:** The time complexity of Euclid's Algorithm is  $O(\log(\max(A, B)))$ .

### Extended Euclid Algorithm:

This is the extended form of Euclid's Algorithm explained above.  $\text{GCD}(A, B)$  has a special property that it can always be represented in the form of an equation, i.e.,  $Ax + By = \text{GCD}(A, B)$ . ?

This algorithm gives us the coefficients (x and y) of this equation which will be later useful in finding the Modular Multiplicative Inverse. These coefficients can be zero or negative in their value. This algorithm takes two inputs as A and B and returns GCD(A, B) and coefficients of the above equations as output.

### Implementation:

```
#include < iostream >

int d, x, y;
void extendedEuclid(int A, int B) {
    if(B == 0) {
        d = A;
        x = 1;
        y = 0;
    }
    else {
        extendedEuclid(B, A%B);
        int temp = x;
        x = y;
        y = temp - (A/B)*y;
    }
}

int main( ) {
    extendedEuclid(16, 10);
    cout << "The GCD of 16 and 10 is " << d << endl;
    cout << "Coefficient x and y are: " << x << " and " << y << endl;
    return 0;
}
```

Output:

```
The GCD of 16 and 10 is 2
Coefficient x and y are: 2 and -3
```

Initially, Extended Euclid Algorithm will run as Euclid Algorithm until we get GCD(A, B) or until B gets 0 and then it will assign  $x = 1$  and  $y = 0$ . As  $B = 0$  and GCD(A, B) is A in the current condition, so equation  $Ax + By = \text{GCD}(A, B)$  will be changed to  $A * 1 + 0 * 0 = A$ .

So the values of d, x, y in the whole process of extendedEuclid( ) function are:

?

- 1)  $d=2, x = 1, y = 0.$
- 2)  $d=2, x = 0, y = 1 - (4/2) * 0 = 1.$
- 3)  $d=2, x = 1, y = 0 - (6/4) * 1 = -1.$
- 4)  $d=2, x = -1, y = 1 - (10/6) * -1 = 2.$
- 5)  $d=2, x = 2, y = -1 - (16/10) * 2 = -3.$

**Time Complexity:** The time complexity of Extended Euclid's Algorithm is  $O(\log(\max(A, B)))$ .

## Prime Numbers

Prime numbers are the numbers greater than 1 that have only two factors, 1 and itself.

Composite numbers are the numbers greater than 1 that have at least one more divisor other than 1 and itself.

For example, 5 is prime number as 5 is divisible by 1 and 5 only. But, 6 is a composite number as 6 is divisible by 1, 2, 3 and 6.

There are different methods to check if the number is prime or not.

### Naive Approach:

We will traverse through all the numbers from 1 to N and count the number of divisors. If the number of divisors are equal to 2 then the given number is prime otherwise it is not.

```
void checkprime(int N){
    int count = 0;
    for( int i = 1; i <= N; ++i )
        if( N % i == 0 )
            count++;
    if(count == 2)
        cout << N << " is a prime number." << endl;
    else
        cout << N << " is not a prime number." << endl;
}
```

**Time Complexity:** Time complexity of this function is  $O(N)$  as we are traversing from 1 to N.

A slightly better approach:

Consider two positive numbers N and D such that N is divisible by D and D is less than the square root of N. Then,  $(N / D)$  must be greater than the square root of N. N is also divisible by  $(N / D)$ . So, if there is a divisor of N that is less than square root of N then there will be a divisor of N that is greater than square root of N. Therefore, we just need to traverse till the square root of N.

?

```

void checkprime(int N) {
    int count = 0;
    for( int i = 1; i * i <= N; ++i ) {
        if( N % i == 0 ) {
            if( i * i == N )
                count++;
            else
                // i < sqrt(N) and (N / i) > sqrt(N)
                count += 2;
        }
    }
    if(count == 2)
        cout << N << " is a prime number." << endl;
    else
        cout << N << " is not a prime number." << endl;
}

```

**Time complexity:** Time complexity of this function is  $O(\sqrt{N})$  as we are traversing from 1 to  $\sqrt{N}$ .

### Sieve of Eratosthenes:

The Sieve of Eratosthenes can be used to find all the prime numbers less than or equal to a given number  $N$ . It can also be used to find out if a number is prime or not, by just looking up at the sieve.

The basic idea behind the sieve of eratosthenes is that at each iteration, we pick one prime number and eliminate all the multiples of that prime number. After the elimination process ends, all the unmarked numbers which are left are prime!

Pseudo Code:

Mark all the numbers as prime numbers.

Traverse over all the prime numbers smaller than  $\sqrt{N}$ .

Mark the multiple of the prime numbers as composite numbers.

```

void sieve(int N) {
    bool isPrime[N+1];
    for(int i = 0; i <= N; ++i) {
        isPrime[i] = true;
    }
    isPrime[0] = false;
    isPrime[1] = false;
    for(int i = 2; i * i <= N; ++i) {

```

?

```

if(isPrime[i] == true) {
    // Mark all the multiples of i as composite numbers
    for(int j = i * i; j <= N ;j += i)
        isPrime[j] = false;
}
}
}

```

The above code will compute all the prime numbers that are smaller than or equal to N. Let us compute prime numbers where N = 10. First mark all the numbers as prime. Now mark 1 as composite. Then in each iteration we will select a prime and then mark its multiple as

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Mark 1 as composite

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Mark multiples of 2 as composite

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Mark multiples of 3 as composite

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Next prime is 5 which is greater than  $\sqrt{10}$  so the loop will break

composite.

So the prime numbers are 2, 3, 5 and 7.

#### Time Complexity:

The inner loop runs for each element.

if  $i = 2$ , inner loop runs  $N / 2$  times

if  $i = 3$ , inner loop runs  $N / 3$  times

if  $i = 5$ , inner loop runs  $N / 5$  times

.

.

?

**Total complexity:**  $N * (\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \dots) = O(N \log \log N)$

## Modular Exponentiation

Consider a problem in which you need to calculate  $a^b \% c$ , where  $\%$  is a modulo operator and  $b$  can be very large (i.e. in order of  $10^{18}$ ).

### Naive Approach:

$a^b \% c$  can be rewritten as  $a * a * a * a * \dots$  upto  $b$  times. So in this approach we will multiply  $a$ ,  $b$  times.

```
long long exponentiation(long long a, long long b, long long c) {
    long long ans = 1;
    for(int i = 1; i <= b; i++) {
        ans *= a;                                //multiplying a,
        ans %= c;                                b times.
    }
    return ans;
}
```

In each iteration the variable `ans` will get multiplied by `a`. Also, special care needs to be taken that this value never exceeds '`c`' in any iteration. Hence we take modulo of '`ans`' with '`c`' in each iteration. i.e., `ans = ans % c`.

The basic property that we exploit in this is:  $(x*y) \bmod n = ((x \bmod n) * (y \bmod n)) \bmod n$ .

So in the above code we have to calculate  $(ans * a) \% c$  which is equal to  $((ans \% c) * (a \% c)) \% c$ .

**Complexity:**  $O(b)$

### Modular exponentiation:

We can now clearly see that this approach is very inefficient, and we need to come up with something better. We can take care of this problem in  $O(\log_2 b)$  by using a technique called exponentiation by squaring. This uses only  $O(\log_2 b)$  squarings and  $O(\log_2 b)$  multiplications. This is a major improvement over the most naive method.

The process we would follow is:

```
ans=1                                //Final answer which will be
displayed
while(b != 0 ) {
    /*Finding the right most digit of 'b' in binary form, if it is 1
    , then multiply the current value of a
```

```

        in ans. */
        if(b%2 == 1) {           //as if b%2 == 1, means last
/rightmost digit of b in binary form is 1.
            ans = ans*a ;
            ans = ans%c;         //at each iteration if value of ans
exceeds then reduce it to modulo c.
        }

        a = a*a;                 // This is explained below
        a %= c;                  //at each iteration if value of a
exceeds then reduce it to modulo c.
        b /= 2;                  //Trim the right-most digit of b in
binary form.
    }

```

To understand the above code : Let's say  $b = 45$ , then express 'b' in binary form. Note the positions of '1' in 'b' .

As in binary form b changes to  $\{101101\}_2$

$$a^b = a^{\{101101\}_2}$$

As if we split the binary representation of b ,

$$b = 1 * 2^5 + 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 .$$

$$\text{Then } a^b = ( 1 * (a^{2^5}) * ( 0 * (a^{2^4}) ) * ( 1 * (a^{2^3}) ) * ( 1 * (a^{2^2}) ) * ( 0 * (a^{2^1}) ) * ( 1 * (a^{2^0}) ) )$$

Now this can be computed easily in 5 iterations.

In every iteration the value of 'a' is made equal to 'a \* a'.

As,

$$a * (2^0) = a$$

$$a * (2^1) = a^2 = a * a$$

$$a * (2^2) = a^4 = ( a * a ) * ( a * a )$$

and so on..

So, at 1st iteration a changes to  $a * a$ , then  $a * a$  changes to  $(a * a) * (a * a)$  and so on. And also in each iteration, we multiply current value of a in ans, whenever we encounter a digit of b in binary form, which is equal to 1.

Processing:

1. Calculate  $(5^{59})\%19$ , where '%' stands for modulo operator.

express 59 in binary =  $\{111011\}_2$

$$5^{59} = 5^{\{111011\}_2} = 5^{2^5} * 5^{2^4} * 5^{2^3} * 5^{2^2} * 5^{2^1} * 5^{2^0}$$

Let's compute its value in 6 iterations:

?



Let  $ans=1$ ,  $a=5$ ,  $b=59$ ,  $c=19$ ;

Iteration 1:

Since the rightmost digit of 'b'(111011) is 1:

$$ans=(ans * a)\%c = (1 * 5)\%19 = 5$$

$$a=(a * a)\%c = (5 * 5)\%19 = 6$$

$$b /= 2 \text{ (} b = 11101 \text{)}$$

Iteration 2:

Since the rightmost digit of 'b'(11101) is 1:

$$ans=(ans * a)\%c = (5 * 6)\%19 = 11$$

$$a=(a * a)\%c = (6 * 6)\%19 = 17$$

$$b /= 2 \text{ (} b = 1110 \text{)}$$

Iteration 3:

Since the rightmost digit of 'b'(1110) is 0:

We won't multiply anything to ans

$$a=(a * a)\%c = (17 * 17)\%19 = 4$$

$$b /= 2 \text{ (} b = 111 \text{)}$$

Iteration 4:

Since the rightmost digit of 'b'(111) is 1:

$$ans=(ans * a)\%c = (11 * 4)\%19 = 6$$

$$a=(a * a)\%c = (4 * 4)\%19 = 16$$

$$b /= 2 \text{ (} b = 11 \text{)}$$

Iteration 5:

Since rightmost digit of 'b'(11) is 1:

$$ans=(ans * a)\%c = (6 * 16)\%19 = 1$$

$$a=(a * a)\%c = (16 * 16)\%19 = 9$$

$$b /= 2 \text{ (} b = 1 \text{)}$$

Iteration 6:

Since rightmost digit of 'b'(1) is 1:

$$ans=(ans * a)\%c = (1 * 9)\%19 = 9$$

$$a=(a * a)\%c = (9 * 9)\%19 = 5$$

$$b /= 2 \text{ (} b = 0 \text{)} // \text{break}$$

Hence, the final answer is  $ans = 9$ ,

therefore  $(5^{59})\%19 = 9$

**Time limit:**

$$O(\log_2(b))$$

Memory limit:

$O(1)$

Complexity:  $O(\log_2(b))$  (number of digits present in binary notation of number 'b')

Implementation:

```
#include < cstdio >
#include < iostream >
using namespace std;

int modpowIter(int a, int b, int c) {
    int ans=1;
    while(b != 0) {
        if(b%2 == 1)
            ans=(ans*a)%c;

        a=(a*a)%c;
        b /= 2;
    }
    return ans;
}

int main() {
    int a=5,b=59,c=19,ans,ans1;
    ans = modpowIter(a,b,c);
    cout << ans << endl;
}
```

Output:

9

You can use another recursive approach, to find  $(a^b)\%c$ .

We know that  $a^b$  can be written as:

$a^b = a^{2^{(b/2)}}$  - If b is even, and  $b > 0$ .

$a^b = (a) * a^{2^{(b-1/2)}}$  - If b is odd.

$a^b = 1$  - If b is 0.

```
int modRecursion(int a, int b, int c)
{
```

?

```

    if(b == 0)
        return 1;
    if(b == 1)
        return a%c;
    else if( b%2 == 0)
        //if b is even
        {
            return modRecursion((a *a)%c,b/2,c);
        }
    else
        // if b is odd.
        {
            return (a*modRecursion((a*a%c),b/2,c))%c;
        }
}

```

Let's take  $a = 2, b = 5, c = 5$ , so  $\text{modRecursion}(2, 5, 5)$ .

1) As  $b$  is odd, it will be  $2 * \text{modRecursion}(2 * 2, 2, 5) = \text{modRecursion}(4, 2, 5)$ .

2) Now in  $\text{modRecursion}(4, 2, 5)$  as  $b$  is even, it will move to  $\text{modRecursion}(16\%5, 1, 5) = \text{modRecursion}(1, 1, 5)$ .

3) Now as in  $\text{modRecursion}(1, 1, 5)$ ,  $B = 1$ , therefore it will return 1.

4) Therefore  $\text{modRecursion}(4, 2, 5)$  will return 1 and as in first statement  $\text{modRecursion}(2, 5, 5)$  will return  $2 * \text{modRecursion}(4, 2, 5) = 2 * 1 = 2$ .

So final answer will be 2 here.

**Complexity:**  $O(\log_2(b))$

**//Change the data types according to the problem statement**

In many problems ' $c$ ' might be given as  $10^9+7$ . Hence, data types of all variables must be kept "long long int".

[Solve Problems](#)

Like 33

Tweet



COMMENTS (81)

SORT BY: **Relevance** ▼

Login/Signup to Comment



**Prerak Atolia** 2 years ago

The below expression in Modular Exponentiation is wrong :

$b = 1 * 25 + 0 * 24 + 1 * 23 + 1 * 22 + 0 * 21 + 1 * 20$  .

Then  $a^b = (1 * (a^{2^5}) * (0 * (a^{2^4})) * (1 * (a^{2^3})) * (1 * (a^{2^2})) * (0 * (a^{2^1})) * (1 * (a^{2^0})))$  ?

$\ast (1 \ast (a^{2^0}))$

This makes  $a^b = 0$ .

The correct expression should be

Then  $a^b = (a^{(1 \ast 2^5)}) \ast (a^{(0 \ast 2^4)}) \ast (a^{(1 \ast 2^3)}) \ast (a^{(1 \ast 2^2)}) \ast (a^{(0 \ast 2^1)}) \ast (a^{(1 \ast 2^0)})$

▲ 9 votes ● Reply ● Message ● Permalink



**Om AnirudhChellapilla** 2 years ago

Yup!

▲ 0 votes ● Reply ● Message ● Permalink



**Arjit Srivastava** 3 years ago

This is a well written post! :)

▲ 2 votes ● Reply ● Message ● Permalink



**Nichit Goel** 2 years ago

isn't there a typo in Sieve of Eratosthenes code initial value of j should be  $j=i+i$  or  $j=2*i$ ?

▲ 0 votes ● Reply ● Message ● Permalink



**Atul Kumar** Edited 2 years ago

Nope it's correct, as the loop progress you can see that first multiple of current j is  $i*i$ . Consider an example the first multiple of 3 after all the multiple of 2 which are already marked is 9 ( $3*3$ ) because 6 has been already marked false by 2.

▲ 2 votes ● Reply ● Message ● Permalink



**Vishal Anand** a year ago

Is there any relationship for expression  $(a - b) \% c = [a \% c - b \% c] \% c$ ? I know this is wrong but is there a correct equation that would give  $(a - b) \% c$  just like  $(a+b) \% c = (a \% c + b \% c) \% c$ .

▲ 0 votes ● Reply ● Message ● Permalink



**Sachin Chauhan** 7 months ago

It is right if  $a > b$

▲ 0 votes ● Reply ● Message ● Permalink



**Umang Panchal** 7 months ago

if  $a=10, b=8$  and  $c=5$  then it will go wrong ,....

▲ 1 vote ● Reply ● Message ● Permalink



**Sachin Chauhan** 7 months ago

No it's not.

See..

$(10-8) \% 5 = 2 \% 5 = 2$

and

$((10 \% 5) - (8 \% 5)) \% 5 = (0 - 3) \% 5 = -3 \% 5 = 2$  (We should treat -3 and 2 same in modulo 5 arithmetic)

SO...  $(a - b) \% c = [a \% c - b \% c] \% c$  is right provided we treat  $-b = -b + c$ . :)

▲ 1 vote ● Reply ● Message ● Permalink



**Gaston Fontenla Nuñez** 3 years ago

I have a few improvements over your prime sieve. Here's the code:

1)Declaring the bool array outside the function allow make queries to the array from anywhere.

2)Making it dynamic is a good choice, because it can be bigger (see code)

3)Use memset to set the array to 1. It's faster.

bool \*isPrime; //Can be bigger, and support queries

void sieve(int N)

{

?

```
isPrime = new(bool[N+1]); //Dynamic is better
memset(isPrime, 1, sizeof(bool)*(N+1)); //Save time (use <string.h>)
isPrime[0] = 0, isPrime[1] = 0;
for(int i=2; i*i<=N; i++)
if(isPrime[i])
for(int j=i*i; j<=N; j+=i) // Mark all the multiples of i as composite numbers
isPrime[j] = 0;
}
```

▲ 1 vote ● Reply ● Message ● Permalink

**Deepanshu Thakur** 3 years ago

You have a typo in example of Modular exponentiation Iteration 3:  
Since the rightmost digit of 'b'(1110) is 1:  
it should be  
Iteration 3:

Since the rightmost digit of 'b'(1110) is 0:

▲ 0 votes ● Reply ● Message ● Permalink



**Prateek Garg** ⚡ Author 3 years ago

Thank You for the information Deepanshu, will fix it soon :)

▲ 0 votes ● Reply ● Message ● Permalink



**Manoj Kumar Regar** 3 years ago

can you provide a link for more explained proof of time complexity for sieve of Eratosthenes ?

▲ 0 votes ● Reply ● Message ● Permalink



**Prateek Garg** ⚡ Author 3 years ago

[https://en.wikipedia.org/wiki/Sieve\\_of\\_Eratosthenes](https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes)

You can visit this link for more information on complexity of this algorithm .

▲ 0 votes ● Reply ● Message ● Permalink



**D Rajiv Lochan Patra** 3 years ago

how the else block worked in recursion and value of A and B are assigned in else block while computing the recursion?

▲ 0 votes ● Reply ● Message ● Permalink



**Prateek Garg** ⚡ Author 3 years ago

recursion in which algorithm?

▲ 0 votes ● Reply ● Message ● Permalink



**D Rajiv Lochan Patra** 3 years ago

Extended Euclid algorithm

▲ 0 votes ● Reply ● Message ● Permalink



**Prateek Garg** ⚡ Author 3 years ago

Try to understand this using Euclid Algorithm. In Extended Euclid, the values of A and B are computed as were in Euclid Algorithm. You can see in the Example explained above for Euclid Algorithm :

Let A = 16, B = 10.

$GCD(16, 10) = GCD(10, 16 \% 10) = GCD(10, 6)$

$GCD(10, 6) = GCD(6, 10 \% 6) = GCD(6, 4)$

$GCD(6, 4) = GCD(4, 6 \% 4) = GCD(4, 2)$

$GCD(4, 2) = GCD(2, 4 \% 2) = GCD(2, 0)$

Here the values changes from {16,10} -> {10, 6} -> {6, 4} -> {4, 2} -> {2, 0}.

Now in Extended Algorithm, once you got GCD , then in each level of recursion, these values are used in reverse order. For this you can refer Example ?

explained for Extended Euclid.  
I hope it will help you more :) .  
▲ 0 votes ● Reply ● Message ● Permalink



**D Rajiv Lochan Patra** 3 years ago  
This comment has been deleted.  
● Reply ● Message ● Permalink



**D Rajiv Lochan Patra** 3 years ago  
and how the reverse order is working?  
▲ 0 votes ● Reply ● Message ● Permalink



**Prateek Garg** ⚡ Author 3 years ago  
Recursion always gives the result by computing values in reverse order. Try to recurse on example manually. It will make this more clear to you.  
▲ 0 votes ● Reply ● Message ● Permalink



**Karan Kapoor** 3 years ago  
hey @prateek it would be great if you could include a bit about CRT and ncr modulo m.. I always get confused with them :(  
▲ 0 votes ● Reply ● Message ● Permalink



**Prateek Garg** ⚡ Author 3 years ago  
Hi Karan, as this is Part I, so I have only included basic topics. The topics like CRT and ncr modulo m, will surely be included in further parts of Number Theory. Stay tuned :)  
▲ 0 votes ● Reply ● Message ● Permalink



**Karan Kapoor** 3 years ago  
@prateek..Thanks :) i was trying to solve a problem using extended euclid algorithm on spoj but was stuck.. Can you help me out a bit ?  
▲ 0 votes ● Reply ● Message ● Permalink



**Prateek Garg** ⚡ Author 3 years ago  
Sure, share the link !  
▲ 0 votes ● Reply ● Message ● Permalink



**Karan Kapoor** 3 years ago  
This was the problem. <http://www.spoj.com/problems/POUR1/> this is my solution.. <http://ideone.com/fntnEP> i have written the problem i am facing in comments in the code.  
I am writing it again here as well..  
i considered the problem similar to the solving the diophantine equation by extended euclid algorithm  
and in case the value of c is equal to gcd(a,b) we can simply return the value of  $\text{abs}(x) + \text{abs}(y)$  where x and y are the coefficients calculated by extended gcd however in the other case.. when c is a multiple of gcd(a,b)  
one of the solution will be  $x \cdot (c/\text{gcd}(a,b))$  and  $y \cdot (c/\text{gcd}(a,b))$  however it may not be the minimum no of moves.  
How do we minimize the number of moves. ???  
I am stuck at this point..  
How can we solve this problem by using extended gcd algorithm in this case when c is not gcd but multiple of it..??  
▲ 0 votes ● Reply ● Message ● Permalink



**Pradyumna Bang** 3 years ago

?

You guys must use LaTeX for writing mathematical formulae.

▲ 0 votes ● Reply ● Message ● Permalink



**Nikhil Kumar Singh** 3 years ago

I support you for this

▲ 0 votes ● Reply ● Message ● Permalink



**Dharmesh Jogadia** 3 years ago

Awesome post :D

▲ 0 votes ● Reply ● Message ● Permalink



**Anirud Samala** 3 years ago

Awesome post!! Can you also suggest some Practice Problems too? Thanks

▲ 0 votes ● Reply ● Message ● Permalink



**Sandeep KD** 3 years ago

Just Awesome coding tutorial with example ... courage to learn Many many thanks for share and making such a great site . Thank again.....

▲ 0 votes ● Reply ● Message ● Permalink



**Pushkar** 3 years ago

In the explanation of modular exponentiation Binary of  $b=55$  is wrongly calculated as 101101(binary of 45). It should be 110111.

▲ 0 votes ● Reply ● Message ● Permalink



**sathiyaseelan** 3 years ago

Yes. Confused for long time.And now verified manually. Change  $b=45$  or binary form of 55 = 110111

▲ 0 votes ● Reply ● Message ● Permalink



**Prateek Garg** ⚡ Author 3 years ago

Yes, it was a typo. It is 45 not 55. It is fixed now.

▲ 0 votes ● Reply ● Message ● Permalink



**Anurag Sharma** 3 years ago

@Prateek please tell me if i want to print almost 3000 prime numbers by Sieve of Eratosthenes algorithm..then what should be the length of array ?

▲ 0 votes ● Reply ● Message ● Permalink



**Prateek Garg** ⚡ Author 3 years ago

As you are using Sieve of Eratosthenes , you do not need an extra array. You can use `isPrime[ ]` array to print prime numbers. Like :

```
if (isPrime[ i ] == true)
    cout << i << endl;
```

And if you want to print exactly 3000 primes, then you can use counter for that :) .

▲ 0 votes ● Reply ● Message ● Permalink



**Anurag Sharma** 3 years ago

i tried but couldn't get 3000 primes...as  $j=i*i$  then even  $n=1000000$  is not working for my code.... please help

▲ 0 votes ● Reply ● Message ● Permalink



**sathiyaseelan** 3 years ago

Is there any specific reason for counting the divisors in `checkPrime`?

We can return immediately after finding the first divisor,right?

?

```
boolean checkPrime(int N){
    for (i=2; i*i<=N;i++){
        if(N%i==0)
            return false;
    }
    return true;
}
```

▲ 0 votes ● Reply ● Message ● Permalink



**Prateek Garg** ⚡ Author 3 years ago

Yes for checking prime number, you can use this code also. The complexity of this algorithm will also be  $O(\sqrt{N})$ . I have used algorithm which is explained in article, because by using that, user will also learn, how to calculate the total number of divisors of any number.

▲ 0 votes ● Reply ● Message ● Permalink



**sathiyaseelan** 3 years ago

ok cool :)

▲ 0 votes ● Reply ● Message ● Permalink



**Jeffry Cpps** 3 years ago

In Sieve algorithm,  
-The inner loop starts from  $i*i$ ?  
Then how come  $i=3$ , will select 6 (shown in the example)?  
please correct me if I am wrong.

▲ 0 votes ● Reply ● Message ● Permalink



**sathiyaseelan** 3 years ago

6 would be already covered by 2.

▲ 0 votes ● Reply ● Message ● Permalink



**Jeffry Cpps** 3 years ago

Exactly. You are right. But it was colored in orange which means it was affected by 3. So I got confused.

▲ 0 votes ● Reply ● Message ● Permalink



**sathiyaseelan** 3 years ago

ok :)

▲ 0 votes ● Reply ● Message ● Permalink



**shubham rajput** 2 years ago

But it must be starting from  $i*2$  instead of  $i*i$ , because in case of 3 it will be starting from 9 ( $3*3$ ). According to your algorithm, with  $i=3$ , the inner loop will run only twice but you are stating that time complexity is  $N/3$  which should be 3 as  $\text{int}(10/3)=3$ .

▲ 0 votes ● Reply ● Message ● Permalink



**RISHABH TRIPATHI** 3 years ago

I think Binary GCD algo is better .

▲ 0 votes ● Reply ● Message ● Permalink



**Sunil Yadav** 3 years ago

good post ! I request to add a link to download these notes as a pdf .It will be of great help to take print and read any time . Thanks .

▲ 0 votes ● Reply ● Message ● Permalink



**Ashutosh Mangalekar** 3 years ago

▲ 0 votes ● Reply ● Message ● Permalink

?



**Gaurav Shankar** 3 years ago

The time complexity of both Euclid's and extended Euclid's algorithm is same. Any advantages of one over the other?

▲ 0 votes ● Reply ● Message ● Permalink

**Prateek Garg** ⚡ Author 3 years ago

Yes, extended euclid's also gives two variables of the above equation. They are very helpful in various cases, which will be discussed in future articles of number theory. Stay tuned :)

▲ 0 votes ● Reply ● Message ● Permalink

**Abdullah Hashem** 3 years ago

You can implement the modular exponentiation using binary operation to make the code run faster:

Code:

```
ans = 1;
while(b)
{
  if(b & 1) ans = (ans * a) % c; // b & 1 return 1 if the number is odd (most right bit is 1)
  otherwise 0;
  a = (a * a) % c;
  b >>= 1; // integer division by 2;
}
return ans;
```

▲ 0 votes ● Reply ● Message ● Permalink

**Sharvin Jondhale** 2 years ago

Compilers are optimized to do that automatically, using them explicitly increases the chances of bugs.

▲ 0 votes ● Reply ● Message ● Permalink

**Mohit Singh** 3 years ago

Hii [developer:ptk23] after that post i am become your fan dear.... and this is request.. can you please write a post on "RECURSION WITH MEMOIZATION"..??

▲ 0 votes ● Reply ● Message ● Permalink

**Shivasurya S** 3 years ago

Really gud article !

▲ 0 votes ● Reply ● Message ● Permalink

**Ishpreet Singh** 3 years ago

Great post .... Keep writing such post .....

▲ 0 votes ● Reply ● Message ● Permalink

**Jyotman Singh** 3 years ago

a = 999999, b=999999 gives negative answer.

▲ 0 votes ● Reply ● Message ● Permalink

**Sreenidhi** 3 years ago

Thank you so much! brilliant explanation

▲ 0 votes ● Reply ● Message ● Permalink

**Shubham Aggarwal** 3 years ago

can you put binary exponentiation algorithm?

▲ 0 votes ● Reply ● Message ● Permalink

**Renganatha** 3 years ago

?

Good Easy to understand :)

▲ 0 votes ● Reply ● Message ● Permalink



**Abhishek Kumar** 3 years ago

In every iteration the value of 'a' is made equal to 'a \* a'.

As,

$$a * (2^0) = a$$

$$a * (2^1) = a^2 = a * a$$

$$a * (2^2) = a^4 = (a * a) * (a * a)$$

and so on..

in the above phrase, how powers of two are being multiplied with a to get powers of a. a must be with powers of two's not with multiple of two's.

▲ 0 votes ● Reply ● Message ● Permalink



**Akash Garg** 3 years ago

In extended euclid's algorithm, its okay how we get d, but for x and y, x and y get their values assigned as 0 or 1 only after we get b=0.

and after that, there is no further looping, so how does the values of x and y start changing? please elaborate.

▲ 0 votes ● Reply ● Message ● Permalink



**Abhishek Inamdar** 3 years ago

sir, please can you provide some notes of basic mathematics for computer science and also a guidance to built a logic some mathematical programs and object oriented programs..:)

▲ 0 votes ● Reply ● Message ● Permalink



**Bhishma Raj** 3 years ago

Can you please write the tutorials in LATEX

▲ 0 votes ● Reply ● Message ● Permalink



**Samyak Jain** 3 years ago

There is a mistake in the Sieve of Eratosthenes.

In the inner for loop to mark all multiples of the prime numbers as composite

Your code:

```
for(int j = i * i; j <= N ;j += i)
```

```
isPrime[j] = false;
```

Correction:

```
for(int j = 2 * i; j <= N ;j += i)
```

```
isPrime[j] = false;
```

▲ 0 votes ● Reply ● Message ● Permalink



**Atul Kumar Gupta** Edited 3 years ago

No it is not. j should start from i\*i only.

▲ 0 votes ● Reply ● Message ● Permalink



**shubham rajput** 2 years ago

Exactly, I was also thinking the same, it is wrong as it doesn't match with the given time complexity

▲ 0 votes ● Reply ● Message ● Permalink



**Atul Kumar** 2 years ago

Yeah, but it's little improvement . i\*i is right

▲ 0 votes ● Reply ● Message ● Permalink



**Gopish Monga** 2 years ago

In the above given Euclid's Algorithm, variable A must always be greater than or equal to B. If otherwise then  $A \% B$  will become equal to zero(0), which is not desired and algorithm fails.

?

Also the time complexity should be  $O(\log(A))$  as A must be the larger element.

▲ 0 votes ● Reply ● Message ● Permalink



**Himanshu Gupta** 2 years ago

$\gcd(10,16) = \gcd(16, 10\%16) = \gcd(16,10) = \dots$

So, it really does not matter if  $a > b$  or not. It is automatically taken care of.

▲ 0 votes ● Reply ● Message ● Permalink



**Ankur Shah** Edited 2 years ago

If you have explained for GCD, you should have also written for LCM too! But why didn't you do that?

▲ 0 votes ● Reply ● Message ● Permalink



**Sachin Chauhan** 7 months ago

LCM(A,B) is simply  $((A*B)/\text{GCD}(A,B))$ .

▲ 0 votes ● Reply ● Message ● Permalink



**Krushika Gada** 2 years ago

In the Extended Euclid implementation, shouldn't the variables x and y be initialized to 1 and 0 respectively? Considering that the given value of B will not be 0, we enter the else part of the function whereby the values of x and y are needed but we don't have them in the first go.

▲ 0 votes ● Reply ● Message ● Permalink



**Rezoan Shakil** 2 years ago

thanks a lot to post this

i want more post in mathematics for competitive programming ...

▲ 0 votes ● Reply ● Message ● Permalink



**Rishabh Agarwal** 2 years ago

Great Post !

Just wanted more explanation for extended euclidean theorem as it is important.

▲ 0 votes ● Reply ● Message ● Permalink



**Dhruvil** 2 years ago

how to multiply 2 10 digit numbers using modular arithmetic

▲ 0 votes ● Reply ● Message ● Permalink



**Atul Kumar** 2 years ago

Can anyone explain to me why we are using this expression in modular exponentiation " $a = (a * a) \% c$ ", it can be kept simple by counting the bit's traversed in some counter k , then calculating each  $(a^k) \% c$  !

▲ 0 votes ● Reply ● Message ● Permalink



**kaushik raj** 2 years ago

extended euclid gives wrong answer for

$a=2$ ;

$b=1000000007$ ;

the correct answer should be 500000004

▲ 0 votes ● Reply ● Message ● Permalink



**Sharvin Jondhale** 2 years ago

You are correct, the problem with the code is.... it requires  $a > b$  at input stage too, as the recursions proceed with that assumption.. swap the variables i.e.

$a=1e9+7$

$b=2$

It will give correct answer, I had made this error when I had written it :p

▲ 0 votes ● Reply ● Message ● Permalink

?

**Suraj Kumar Patel** 7 months ago

thanks hackerearth...

▲ 0 votes ● Reply ● Message ● Permalink

**Tanvir Hossain Yuv** 7 months ago

Please update the article in Modular Exponentiation. There are some mistakes making confused.

1.  $ab = (1 * (a^{2^5}) * (0 * (a^{2^4})) * (1 * (a^{2^3})) * (1 * (a^{2^2})) * (0 * (a^{2^1})) * (1 * (a^{2^0})))$ 

This statement is wrong.

2.  $559 = 5^{\{111011\}_2} = 5^{2^5} * 5^{2^4} * 5^{2^3} * 5^{2^3} * 5^{2^1} * 5^{2^0}$ This statement is also wrong. Here,  $5^{2^3}$  occurs twice which should not be.

Please update.

▲ 0 votes ● Reply ● Message ● Permalink

**saksham agarwal** a month agocomplexity of euclid theorem will  $\log(\min(A,B))$ .

▲ 0 votes ● Reply ● Message ● Permalink

## AUTHOR

**Prateek Garg**

SDE-1 at Flipkart

BENGALURU

7 notes

## TRENDING NOTES

[Python Diaries Chapter 3 Map | Filter | For-else | List Comprehension](#)

written by Divyanshu Bansal

[Bokeh | Interactive Visualization Library | Use Graph with Django Template](#)

written by Prateek Kumar

[Bokeh | Interactive Visualization Library | Graph Plotting](#)

written by Prateek Kumar

[Python Diaries chapter 2](#)

written by Divyanshu Bansal

[Python Diaries chapter 1](#)

written by Divyanshu Bansal

[more ...](#)

[About Us](#)[Innovation Management](#)[Technical Recruitment](#)[University Program](#)[Developers Wiki](#)[Blog](#)[Press](#)[Careers](#)[Reach Us](#)

Site Language: [English](#) ▼ | [Terms and Conditions](#) | [Privacy](#) | © 2018 HackerEarth