Number Theory

Programming and Algorithms Group

What all we will be covering

- Prime Numbers
- Modular Arithmetic and Inverse Modulo
- Greatest Common Divisor and its properties

How to check whether a number is a prime?

```
bool isPrime(int n)

{

    if (n <= 1)  return false;
    if (n <= 3)  return true;
    if (n%2 == 0 || n%3 == 0) return false;
    for (int i=5; i*i<=n; i=i+6)
        if (n%i == 0 || n%(i+2) == 0)
           return false;
    return true;

}
```

Time complexity of this solution is O(√n)

How to calculate number of primes less than 10^6

Sieve of Eratosthenes

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

Prime numbers

```
void SieveOfEratosthenes(int n)
{

    bool prime[n+1];      // to have the index of last number
    memset(prime, true, sizeof(prime));

    for (int p=2; p*p<=n; p++)
    {
        if (prime[p])
        {
            for (int i=p*p; i<=n; i += p)
                prime[i] = false;
        }
    }

    for (int p=2; p<=n; p++)
       if (prime[p])
          cout << p << " ";
}
```

Time complexity of this algorithm is O(n(loglog(n)))

Questions related to Sieve of Eratosthenes

There are T test cases to a problem. Each case requires you to find the minimum prime factor of a given number n.

Constraints - 1<=T<=10^5
2<=N<=10^7

Modular Arithmetic

Basic Properties

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

$(a * b) \% m = ((a \% m) * (b \% m)) \% m$

$(a - b) \% m = ((a \% m) - (b \% m) + m) \% m$

$(a \wedge b) \% m = ((a \bmod m) \wedge b) \% m$

Modular Exponentiation (Finding (x^y) mod p)

```
long long int power(long long int x, long long int y, int p)
{
    long long int res = 1;
    x = x % p;

    while (y > 0)
    {
        if (y & 1)    // Bitwise And (Checks whether number is odd)
            res = (res*x) % p;

        y = y>>1;      // Bitwise Right Shift operator (returns y/2)
        x = (x*x) % p;

    }
    return res;
}
```

Time Complexity of above solution is O(Log y).

Fermat’s Little Theorem

Greatest Common Divisor

Euclid’s GCD Algorithm

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

Extended Euclid’s GCD Algorithm

```
int gcdExtended(int a, int b, int *x, int *y)
{

    if (a == 0)
    {
        *x = 0;
        *y = 1;
        return b;
    }

    int x1, y1;
    int gcd = gcdExtended(b%a, a, &x1, &y1);
    *x = y1 - (b/a) * x1;
    *y = x1;
    return gcd;
}
```

Modular Inverse

Lecture Material

Basics of Number Theory : https://crypto.stanford.edu/pbc/notes/numbertheory/

Primes, Modular Arithmetic and Fermat's Theorem

L1 : https://en.wikipedia.org/wiki/Modular_arithmetic

L2 : https://en.wikipedia.org/wiki/Modular_exponentiation

L3 : https://en.wikipedia.org/wiki/Fermat%27s_little_theorem

P1 : http://www.spoj.com/problems/ADST01/

P2 : https://erdos.sdslabs.co/problems/8

P3 : https://erdos.sdslabs.co/problems/19

P4 : https://projecteuler.net/problem=7

P5 : https://www.spoj.com/problems/APS/

P6 : https://www.spoj.com/problems/DIVFACT/

P7 : https://www.codechef.com/problems/BIPIN3

GCD and Extended GCD

L1 : https://en.wikipedia.org/wiki/Euclidean_algorithm

L2 : https://www.topcoder.com/community/competitive-programming/tutorials/mathematics-for-topcoders/

L3 : https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm

P1 : https://www.spoj.com/problems/MAY99_3/

P2 : https://www.spoj.com/problems/GCD2/

P3 : http://codeforces.com/problemset/problem/689/D

P4 : https://www.spoj.com/problems/MAIN74/

P5 : https://www.spoj.com/problems/ENIGMATH/

Additional Topics in Number Theory :

- **Euler Totient Function (ETF)**

  https://en.wikipedia.org/wiki/Euler's_totient_function

  https://www.topcoder.com/community/competitive-programming/tutorials/prime-numbers-factorization-and-euler-function/

- **Fibonacci Numbers (Matrix Exponentiation)**

  https://en.wikipedia.org/wiki/Fibonacci_number

- **Chinese Remainder Theorem**

  http://www.cut-the-knot.org/blue/chinese.shtml

  http://mathworld.wolfram.com/ChineseRemainderTheorem.html

  https://www.codechef.com/wiki/very-brief-tutorial-chinese-remainder-theorem