Teoria dos números

- Fast Exponential:

```
ll fExp(ll a, ll b){
  ll ans = 1;
  while(b){
        if (b & 1) ans = ans * a % mod;
        a = a * a % mod;
        b >>= 1;
  }
  return ans;
}
```

- Matrix Exp:

```
const int m = 3; // size of matrix
struct Matrix{
  ll mat[m][m];
  Matrix operator * (const Matrix &p){
        Matrix ans;
        for (int i = 0; i < m; i++)
            for (int j = 0; j < m; j++)
                for (int k = ans.mat[i][j] = 0; k < m; k++)
                   ans.mat[i][j] = (ans.mat[i][j] + mat[i][k] * p.mat[k][j]) % mod;
        return ans;
  }
};
Matrix fExp(Matrix a, ll b){
  Matrix ans;
  for (int i = 0; i < m; i++) for (int j = 0; j < m; j++)
        ans.mat[i][j] = (i == j);
  while(b) {
        if (b & 1) ans = ans*a;
        a = a*a;
        b >>= 1;
  }
  return ans;
}
```

- Greatest common divisor(gcd/mdc):

```
ll gcd(ll a, ll b){
  while (b) a %= b, swap(a, b);
  return a;
}
```

- Least common multiple(lmm/mmc):

```
ll lcm(ll a, ll b){
  return a*b/gcd(a, b);
}
```

O(log(n))

- Count Divisors:

```
int contarDivisores(ll n){
  int c = 0;
  for (int i = 1; i <= sqrt(n); i++){
        if (n%i == 0){
              c++;
              if(n/i != i) c++;
        }
  }
  return c;
}
```

O(log(n))

- Euler Totient Function (phi):

```
int phi[n];
void PHI(){//sieve for phi
  for (int i = 1; i <= P; i++) phi[i] = i;
  for(int i = 2; i <= P; i++){
      if (phi[i]==i)
          for (int j = i; j <= P; j+=i)
              phi[j]=phi[j]/i*(i-1);
  }
}
void PHI(int n){//calculates one phi
  int ans = n;
  for (int i = 2; i*i <= n; i++){
      if (n%i==0){
          ans -= ans/i;
          while(n%i==0) n/=i;
      }
  }
  if (n>1) ans -= ans/n;
  return ans;
}
```

- Prime numbers (crivo):

```
vector<bool> crivo(20000001, true);
vector<ll> primos;

void fastPrime(ll lim){
  primos.push_back(2);
   for(ll i = 3 ; i < lim; i += 2){
      if(crivo[i]){
          primos.push_back(i);
          if(i > lim/i) continue;
          for(ll j = i*i ; j < lim; j += i+i){
              crivo[j] = false;
          }
      }
   }
}
```

- Equação Diofantina linear (A*X + B*Y = C):

```
ll gcd_ext(ll a, ll b, ll &x, ll &y){
  if (b==0){
      x = 1;
      y = 0;
      return a;
  }
  ll nx, ny;
  ll gc = gcd_ext(b, a%b, nx, ny);
  x = ny;
  y = nx - (a/b)*ny;

  return gc;
}
ll mdc = gcd(n1,n2);
//n1 /= mdc; n2 /= mdc; n /= mdc;
//Se gcd(a, b) nao divide C, entao nao tem solucao
if ((n%mdc != 0))
  printf("failed\n");
gcd_ext(n1, n2, x, y);
//Mult o x e o y por p/gcd(a,b)
x *= n/mdc;
y *= n/mdc;
printf("%lld %lld\n", x, y);
```

- Divisibilidade:

Definição: um inteiro A divide B se existe um inteiro K tal que A*K = B.
Notação: A | B == "A divide B".
Propriedades:
i) $a \mid b \Rightarrow a \mid bc$, para todo inteiro c.
ii) $a \mid b$ e $c \mid d \Rightarrow ac \mid bd$.
iii) $a \mid b$ e $b \mid c \Rightarrow a \mid c$.
iv) $a \mid b$ e $a \mid c \Rightarrow a \mid (bx + cy)$, para todos x, y inteiros.
v) $a \mid b$ e $b \mid a \Rightarrow a = b$ ou $a = -b$.
vi) $a \mid b$ com $a,b>0 \Rightarrow a <= b$.
vii) $a \mid b \Rightarrow ma \mid mb$, para todo m inteiro.