Matemática

Números prímos

Descrição

- sieve: Crivo de Eristótenes, armazena no vetor primes os primos até **n** e no bitset bs se o o numero é ou não primo. em complexidade **O(N log log N)**;
- is_prime_sieve: Calcula um numero primo caso **sievesize** seja maior ou igual que √N em complexidade **O(√N/log N)**
- is_prime: Calcula se um número é ou não primo de maneira rápida sem depender do crivo em complexidade **O(√N)**

```
using ll = long long;
const long long MAX = 10000009;
ll sievesize;
bitset<MAX> bs;
vector<ll> primes;

void sieve(ll n){
    sievesize = n+1;
    bs.set();
    bs[0]=bs[1]=0;
    for(ll i=2; i<=sievesize; ++i){
        if(bs[i]){
            for(ll j=i*i; j<=sievesize; j+=i)
                bs[j]=0;
            primes.push_back(i);
        }
    }
}

bool is_prime_sieve(ll n){
    if(n<=(ll)sievesize) return bs[n];
    for(size_t i=0; i<primes.size() and primes[i]*primes[i]<=n; ++i)
        if(n%primes[i] == 0) return false;
    return true;
}

bool is_prime(ll n){
    if(n<0) n=-n;
    if(n<5 or n%2==0 or n%3==0)
        return (n==2 or n==3);
    ll maxP = sqrt(n)+2;
    for(ll p=5; p<maxP; p+=6){
        if( p<n and n%p==0 ) return false;
        if( p+2<n and n%(p+2)==0 ) return false;
    }
    return true;
}
```

Maior Divisor Comum

Dados dois inteiros a e b, o maior divisor comum (MDC) de a e b (notamos d = (a, b) é o inteiro não-negativo d tal que d divide a e d divide b; se c divide a e c divide b, então c divide d.

```
long long gdc(long long a, long long b){
    long long rest;
    do{
        rest = a%b;
        a=b;
        b=rest;
    }while(rest!=0);
    return a;
}
```

Complexidade O(log a +log b)

Menor Múltiplo Comum

Sejam a e b dois inteiros. O menor múltiplo comum (MMC) de a e b (notamos m = [a,b]) é o inteiro m tal que a divide m e b divide m;

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

Complexidade O(log a + log b)

Veja que, na implementação acima, a divisão é feita antes do produto: esta ordem pode evitar overflow em alguns casos.

Número de Divisores

A fatoração de um número n também permite computar o número de divisores deste número: basta fazer o produto de todos os expoentes da fatoração, somados cada um de uma unidade. Veja o código abaixo.

```
long long number_of_divisors(long long n)
{
```

```
    long long res = 0;

    for (long long i=1; i*i <= n; ++i)
    {
        if (n%i == 0)
            res += (i == n/i ? 1 : 2);
    }

    return res;
}
```

Complexidade O(sqrt n)