MaratonaCIn

Seletiva 2020

Teoria dos Números

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.

Como contar os divisores?

MaratonaCIn

Percorrer k de 1 a N, checar se n%k==0 ?

Dá pra fazer melhor:

Suponha que N = ab, com a <= b

Assim temos que aa <= N

a <= sqrt(N)

```
int contarDivisores(int n){
    int c=0;
    for(int i=1;i<=sqrt(n);i++){
        if(n%i==0){//i e n/i sao divisores de n
            c++; //i eh divisor
            if(n/i != i) c++;//  n/i eh outro
        }
    }
    return c;
}
```

crivo+gdc+lcm

Crivo

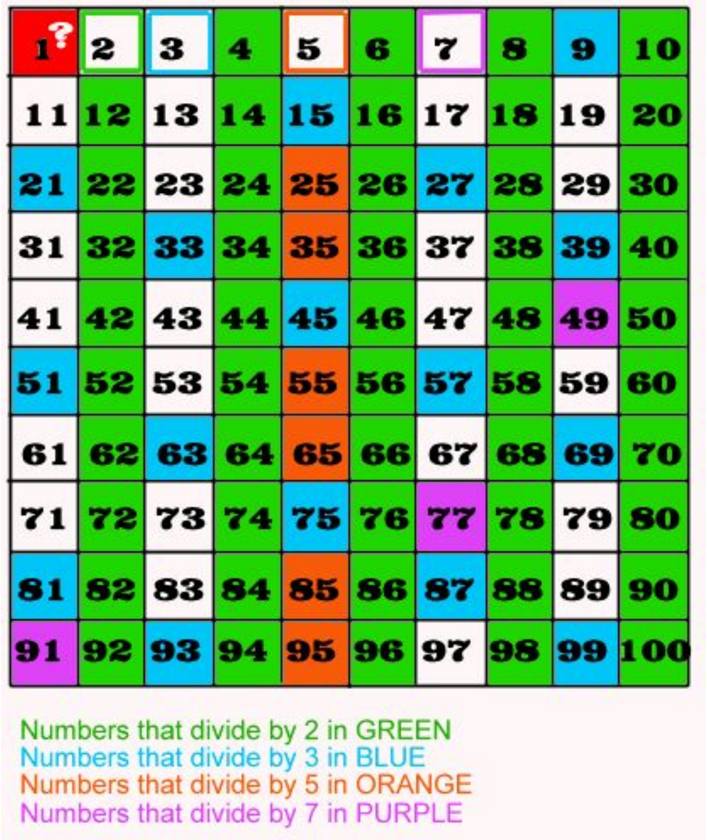
Princípio básico:

- Se ‘a’ é primo, então podemos ter certeza que todos os múltiplos de ‘a’ nao sao primos

- Logo podemos armazenar um array de boolean indicando todos os numeros que nao sao primos

- Assim conseguiremos gerar um vetor com todos os números primos até um dado limite

- O(nlogn)



Crivo

MaratonaCIn

```
bool sieve[ms];
vector<int> fastPrime(int lim) {
    memset(sieve, 1, sizeof sieve);
    vector<int> resp;
    resp.push_back(2);
    for(int i=3; i<lim; i+=2){
        if(sieve[i]){
            resp.push_back(i);
            if(i>lim/i) continue;
            for(int j=i*i; j<lim; j+=i+i){
                sieve[j]=0;
            }
        }
    }
    return resp;
}
```

GCD

Algoritmo de euclides

- Algoritmo para encontrar o gcd(Greatest common divisor)
- gcd(a,b) == b==0? a : gcd(b, a%b)
- Ja implementado no std: *__gcd(a, b)*
- O(logn)

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

LCM

- Least common multiple

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

Aritmetica Modular

MaratonaCIn

Propriedades básicas de congruências lineares:

i) $a \equiv b \pmod{n}$ é equivalente a dizer:

$$a = n*q + b \text{ ou}$$
$$b = n*q' + a$$

ii) Se $a \equiv b \pmod{n}$ e $b \equiv c \pmod{n}$, então $a \equiv c \pmod{n}$.

iii) Se $a \equiv b \pmod{n}$ e $c \equiv d \pmod{n}$, então $a + c \equiv b + d \pmod{n}$ e $ac \equiv bd \pmod{n}$.

iv) Se $a \equiv b \pmod{n}$ e $d \mid n$, $d > 0$, então $a \equiv b \pmod{d}$.

v) Se $a \equiv b \pmod{n}$, então $ac \equiv bc \pmod{nc}$ para $c > 0$.

vi) Se $a \equiv b \pmod{n}$, então $a^k \equiv b^k \pmod{n}$, para qualquer inteiro positivo k. CUIDADO!

vii) Se $ca \equiv cb \pmod{n}$, então $a \equiv b \pmod{n/d}$, em que $d \triangleq \text{MDC}(c, n)$.

viii) $a \equiv b \pmod{pq}$ se e só se $a \equiv b \pmod{p}$ e $a \equiv b \pmod{q}$. (UTIL PRA CRT, que não será cobrado)

equacao diofantina + algoritmo de euclides estendido

Equação diofantina linear

MaratonaCIn

- Encontre algum par de inteiros (X, Y), tal que A * X + B * Y = C, com A, B, C constantes.

- E se nós só quisermos saber se existe resposta ?

Euclides estendido

- Podemos descobrir um par (X, Y), tal que A * X + B * Y = gcd(A, B)

```
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;
}
```

Equação diofantina linear

- Sabendo uma solução para A * X + B * Y = gcd(A, B), podemos descobrir uma solução para a equação inicial, se existir resposta.

- É só fazer X = X * C / gcd(A, B); Y = Y * C / gcd(A, B);

- E se quisermos descobrir todas as soluções ?

- E se quisermos encontrar uma solução para uma equação com N variáveis ?

Pequeno Teorema de Fermat

Se P é primo e P não divide a, então:

$$a^{p-1} \equiv 1 \pmod{p}$$

Prova: Considere os p-1 múltiplos de a,

$$a, 2a, 3a, \ldots\ldots\ldots, (p\text{-}1)a,$$

esses múltiplos são incongruentes módulo p, pois se $ra \equiv sa \pmod{p}$, em que $1 \leq r, s \leq p\text{-}1$, então, como MDC($a$, p) =1, o fator comum a pode ser cancelado, resultando em $r \equiv s \pmod{p}$, o que só é possível se $r = s$. Dessa forma os múltiplos de a são congruentes, em alguma ordem, aos inteiros 1, 2, 3, ..., p-1, ou seja,

$$a.2a.3a....(p-1)a \equiv 1.2.3....(p-1) \pmod{p},$$
$$a^{p-1}(p-1)! \equiv (p-1)! \pmod{p};$$

$$a^{p-1} \equiv 1 \pmod{p}$$

MaratonaCIn

Pequeno Teorema de Fermat

Utilidades:

- Exponenciação modular
- Achar inverso multiplicativo

Mas e se o módulo não for primo??

Funcao de Euler

Definição: Quantidade de números k entre 1 e N tal que MDC(k,N) = 1

Propriedades:

i) $\phi(p) = p - 1.$

ii) $\phi(p^k) = p^{k-1}(p-1)$

iii) A função de Euler é multiplicativa

iv) Se a fatoração canônica de n é $n = p_1^{k_1} p_2^{k_2} \dots p_r^{k_r}$, então $\phi(n) = n \prod_{i=1}^{r} \frac{(p_i - 1)}{p_i}$

Teorema de Euler: Se os inteiros a e n são relativamente primos, então

$$a^{\phi(n)} \equiv 1 (mod\ n).$$

$$a^{p-1} \equiv 1 \pmod{p}$$

Funcao de Euler

MaratonaCIn

```
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)//n is prime
      for(int j=i;j<=P;j+=i)
        //i is a prime factor of j
        phi[j]=phi[j]/i*(i-1);
}
```

```
int PHI(int n){//calculates one phi
  int ans = n;
  for(int i=2;i*i<=n;i++){
    if(n%i==0){//i is a prime factor
      ans-=ans/i;
      while(n%i==0)n/=i;
    }
  }
  if(n>1){//n is prime
    ans-=ans/n;
  }
  return ans;
}
```

exponenciação rápida

exponenciação rápida

exponenciação rápida

- Algoritmo usado para calcularmos uma exponenciação em O(logn)
- Calcula A^B

```
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;
}
```

Exponenciação de matrizes

MaratonaCIn

- Seja A uma matriz, queremos calcular A^k

```
const int m = 2; // 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;
  }
};
```

Exponenciação de matrizes

```
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;
}
```