

Introdução à Teoria dos Números

Laboratório de Programação Competitiva I

Pedro Henrique Paiola

Rene Pegoraro

Wilson M Yonezawa

BigInteger

- Certos problemas da Maratona de Programação recebem como entrada números inteiros que extrapolam o limite de variáveis do tipo **long long int**
- Tamanho de uma variável long long int: **8 bytes**
- Intervalo de números que podem ser armazenados em uma variável desse tipo:
 - -9.223.372.036.854.775.808 à 9.223.372.036.854.775.807
 - 0 à 18.446.744.073.709.551.615 (unsigned long long int)

BigInteger

- Certos problemas da Maratona de Programação recebem como entrada números inteiros que extrapolam o limite de variáveis do tipo **long long int**
- Exemplo: 2667 - Jogo de Boca
 - Entrada: N ($3 \leq N \leq 10^{100}$)

BigInteger

- **1ª Opção:** dependendo das operações necessárias de se fazer com o número, podemos ler o número como sendo uma **string** e trabalhar com essa string.
- Exemplos:
 - Operações simples com dígitos
 - Uso de Aritmética Modular

BigInteger

- **2ª Opção:** se precisarmos fazer operações com esse número **como soma, subtração, multiplicação e divisão**, o problema se torna mais complexo.
- Nesses casos, não recomendamos usar a linguagem C++. É possível trabalhar com BigInteger em C++ (a biblioteca do Thiago traz códigos para isso), porém a quantidade de código necessária é relativamente grande.
- Sugestões: **Java ou Python**

BigInteger

- Em Java podemos usar a classe [BigInteger](#) da biblioteca java.math

```
String Num;
BigInteger NumGrande;
Scanner S = new Scanner(System.in);
Num = S.nextLine();

NumGrande = new BigInteger(Num);
NumGrande = NumGrande.mod(new BigInteger("3"));

System.out.println(NumGrande);
```

BigInteger

- Em Python, não precisamos nos preocupar muito com o tamanho de um inteiro, a memória é alocada conforme o necessário para comportar o tamanho do número.

[Entrada e Saída em Python](#)

[Python em Programação Competitiva](#)

[Muita coisa sobre Python](#)

BigInteger

- Em Python, não precisamos nos preocupar muito com o tamanho de um inteiro, a memória é alocada conforme o necessário para comportar o tamanho do número.

```

U = int(input())
print(U % 3)
  
```


Teoria dos Números

- A Teoria dos Números é o ramo da matemática que se preocupa com as propriedades dos **números inteiros**.
- Existe uma coleção de algoritmos interessantes derivados de estudos da Teoria dos Números que solucionam problemas de forma inteligente e eficiente.
- Aqui faremos uma breve introdução à alguns tópicos relativos à Teoria dos Números.

Números primos

- Diversos problemas envolvem o uso de números primos.
- Dessa forma, precisamos, inicialmente, de uma forma de testar se um número é primo ou não.
- Recordando: números primos são **números naturais** que têm **apenas dois divisores**: 1 e ele mesmo.

Números primos

- Algoritmo ingênuo $O(n)$

```

bool ehPrimo(int n)
{
    for(int i = 2; i < n; i++)
        if (n % i == 0)
            return false;
    return true;
}
  
```

Números primos

- Porém, na verdade só precisamos testar até \sqrt{n}
- **Demonstração:**

Suponha que não, nesse caso existe n tal que o menor fator primo p de n é maior que \sqrt{n} .

Se p divide n , então $\frac{n}{p}$ também divide n , e $\frac{n}{p}$ deve ser maior que \sqrt{n} .

Mas se $p > \sqrt{n}$ e $\frac{n}{p} > \sqrt{n}$, então $p \cdot \frac{n}{p} > n$, o que é um absurdo! ■

Números primos

- Algoritmo $O(\sqrt{n})$

```

bool ehPrimo(int n)
{
    int raiz = sqrt(n);
    for(int i = 2; i <= raiz; i++)
        if (n % i == 0)
            return false;
    return true;
}
  
```

Números primos

- Algoritmo $O(\sqrt{n})$

```

bool ehPrimo(int n)
{
    for(int i = 2; i*i <= n; i++)
        if (n % i == 0)
            return false;
    return true;
}
  
```

Crivo de Eratóstenes

- O Crivo de Eratóstenes é um método de encontrar os números primos até um certo valor limite.
- Útil em casos que faremos vários testes de primalidade e na fatoração de números.
- Ideia geral: dado que um número p é primo, marcamos os múltiplos de p como não sendo números primos.

Crivo de Eratóstenes

- Algoritmo:
 - Cria-se uma lista de 2 a MAX , marcando todos como primos
 - Para cada número i de 2 até \sqrt{MAX}
 - Se i está marcado como primo
 - Marcar todos os números múltiplos de i a partir de $i.i$ como compostos (não primos)

Crivo de Eratóstenes

- Algoritmo:
 - Cria-se uma lista de 2 a MAX , marcando todos como primos
 - Para cada número i de 2 até \sqrt{MAX}
 - Se i está marcado como primo
 - Marcar todos os números múltiplos de i a partir de $i.i$ como compostos (não primos)

Por que podemos marcar só a partir de $i.i$?

Crivo de Eratóstenes

- Antes de $i.i$ temos: $i.2, i.3, i.4, \dots i.(i-1)$. Ou ainda, $i.j \mid 2 < j < i$
- Seja $x = i.j$, x é múltiplo de i e também é múltiplo de j
- Todo j ou é primo, ou é múltiplo de um número primo menor que i , ou seja, um primo já “descoberto” pelo algoritmo
- Se j é primo
 - Todos os seus múltiplos foram marcados como não primo, inclusive $i.j$
- Se j é múltiplo de um primo $p < i$
 - Então ele já foi marcado como composto, por ser múltiplo de p , assim como todos os seus múltiplos
- Logo, todos os números $i.j \mid 2 < j < i$ já foram marcados

Crivo de Eratóstenes

```

bool ehPrimo[MAX];
vector<int> primos;

void crivo(int n){
    memset(ehPrimo, true, sizeof(ehPrimo));
    for(int p = 2; p * p <= n; p++){
        if (ehPrimo[p]){
            primos.push_back(p); //Lista incompleta, primos até sqrt(n)
            for(int i = p*p; i <= n; i += p)
                ehPrimo[i] = false;
        }
    }
}

```

Crivo de Eratóstenes

	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

Números primos:

Crivo de Eratóstenes

	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

Números primos:
2

Crivo de Eratóstenes

	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

Números primos:
2 3

Crivo de Eratóstenes

	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

Números primos:

2 3 5

Crivo de Eratóstenes

	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

Números primos:

2 3 5 7

Crivo de Eratóstenes

	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

Números primos:

2	3	5	7	11
13	17	19	23	29
31	37	41	43	47
53	59	61	67	71
73	79	83	89	97
101	103	107	109	113

Crivo de Eratóstenes

- Este algoritmo possui complexidade $O(n \log \log n)$
- Esta demonstração não é muito simples. Caso queira conferir, veja o artigo do [CP-Algorithms](#).
- Com certas otimizações ainda é possível obter um [algoritmo de complexidade linear](#).

Fatoração

- Fatoração em $O(\sqrt{n})$

```

vector<int> fatorar(int n) {
    vector<int> fator;
    for (int i = 1; i*i <= n; i++){
        while (n % i == 0){
            fator.push_back(i);
            n /= i;
        }
    }
    if (n > 1)
        fator.push_back(n);
    return fator;
}
  
```

Fatoração

- Também é possível obter um algoritmo de fatoração com complexidade $O(\log n)$, baseando-se no Crivo de Eratóstenes.
- Primeiramente, ao invés de utilizarmos o crivo para descobrirmos todos os primos, faremos uma pequena alteração para computar para cada número o seu Menor Fator Primo (*Shortest Prime Factor* - SPF).

Fatoração

- Crivo para Fatoração

```

int spf[MAXN];

void crivo(){
    for(int i=2; i < MAXN; i++){
        if(spf[i] == 0){
            spf[i] = i;
            for(int j=i*i; j<MAXN; j+=i){
                if(spf[j] == 0) spf[j] = i;
            }
        }
    }
}
  
```

Fatoração

- A partir do vetor SPF pré-calculado, podemos realizar a fatoração de um número qualquer seguindo o seguinte algoritmo:
 - `fatores = []`
 - enquanto `n > 1`
 - Inserir `spf[n]` em `fatores`
 - `n = n/spf[n]`

Fatoração

- Fatoração em $O(\log n)$

```

vector<int> fatorar(int n){
    vector<int> fator;
    while(n > 1){
        fator.push_back(spfn[n]);
        n /= spfn[n];
    }
    return fator;
}
  
```

Look-up tables

- Existem casos onde podemos gerar um **vetor ou matriz de consulta** manualmente (ou previamente por outro programa), e inseri-los prontos no nosso código. Dessa forma, economiza-se o tempo de gerar tal vetor/matriz.
- Por exemplo, se para resolver um problema precisamos de todos os primos até N , podemos embutir um vetor de primos já dentro do código.

```
int primos[] = {2, 3, 5, 7, 11, 13, ... }
```

- Isso também pode ser gerado por um programa auxiliar.

Look-up tables

“The judge can’t look into your heart or your program to see your intentions - it only checks the results.”
(Skiena & Revilla, 2003; p. 129)

Máximo Divisor Comum

- Problema: encontrar o maior divisor comum de um par de números.
- Algoritmo de Euclides
- OBS: se $mdc(x, y) = 1$, então dizemos que x e y são coprimos ou primos entre si.

Máximo Divisor Comum

O processo das divisões sucessivas

$$\begin{array}{r|l} a & b \\ \hline r_1 & q_1 \end{array} \quad
 \begin{array}{r|l} b & r_1 \\ \hline r_2 & q_2 \end{array} \quad
 \begin{array}{r|l} r_1 & r_2 \\ \hline r_3 & q_3 \end{array} \quad
 \dots \quad
 \begin{array}{r|l} r_{n-2} & r_{n-1} \\ \hline r_n & q_n \end{array} \quad
 \begin{array}{r|l} r_{n-1} & r_n \\ \hline 0 & q_{n+1} \end{array}$$

nos garante que:

$$\text{mdc}(a, b) = \text{mdc}(b, r_1) = \text{mdc}(r_1, r_2) = \dots = \text{mdc}(r_{n-2}, r_{n-1}) = \text{mdc}(r_{n-1}, r_n) = \text{mdc}(r_n, 0) = r_n$$

Esse processo pode ser efetuado usando-se o seguinte dispositivo prático:

	q_1	q_2	q_3			q_n	q_{n+1}
a	b	r_1	r_2	\dots	r_{n-2}	r_{n-1}	r_n
r_1	r_2	r_3			r_n	0	

Observe que o $\text{mdc}(a, b)$ é o **último resto não nulo** do processo das divisões sucessivas.

Máximo Divisor Comum

- MDC/GCD em $O(\log(a + b))$

```

int gcd(int a, int b){
    if (a == 0)
        return b;
    return gcd(b % a, a);
}
  
```

Mínimo Múltiplo Comum

- Problema: encontrar o menor múltiplo comum entre um par de inteiros.
- Para encontrar o $\text{mmc}(x, y)$, podemos calcular o $\text{mdc}(x, y)$ e utilizar a seguinte fórmula:

$$\text{mmc}(x, y) * \text{mdc}(x, y) = x * y$$

Ou seja:

$$\text{mmc}(x, y) = x * y / \text{mdc}(x, y)$$

Mínimo Múltiplo Comum

- MMC/LCM em $O(\log(a + b))$

```

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

Equações diofantinas

- Podemos definir uma equação diofantina linear como uma equação da forma

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = c$$

sendo a_1, \dots, a_n coeficientes inteiros não nulos, x_1, \dots, x_n as **variáveis inteiras** a serem determinadas e c uma constante inteira.

- Diversos problemas podem ser modelados usando equações diofantinas. Em especial, vamos nos preocupar com equações diofantinas de duas variáveis

$$ax + by = c$$

Equações diofantinas - Fantastic Beasts

- Exemplo de Problema: [Fantastic Beasts](#) (Final da Maratona SBC de Programação - 2018)
- Resumindo: considere um **grafo direcionado** em que os vértices representam zoológicos, e cada zoológico aponta para apenas para um outro zoológico (grau de saída = 1). Temos animais espalhados por esses zoológicos, e a **cada unidade de tempo todos os animais avançam para o próximo zoológico**.
- Objetivo: determinar **onde e quando TODOS os animais se encontrarão**, no mesmo zoológico ao mesmo tempo (se isso puder ocorrer em diversos momentos e locais, determinar o primeiro deles)

Equações diofantinas - Fantastic Beasts

- Supondo que já estamos em uma fase um pouco mais avançada no problema, onde conseguimos modelar para cada **zoológico** **z** uma **equação** que determina os momentos em que um animal *a* passa por lá (os animais vão acabar presos em ciclos).

$$t = t_0 + k.i$$

Equações diofantinas - Fantastic Beasts

- Se quisermos saber quando que os animais a_1 e a_2 se encontram no zoológico z , temos:

Para animal a_1 : $t_1 = t_0^1 + k_1 i$

Para animal a_2 : $t_2 = t_0^2 + k_2 j$

Como queremos saber o momento de encontro, $t_1 = t_2$

$$t_0^1 + k_1 i = t_0^2 + k_2 j$$

$$k_1 i - k_2 j = t_0^2 - t_0^1$$

Equação diofantina com $a = k_1, b = -k_2, c = t_0^2 - t_0^1$ e com variáveis i, j

Equações diofantinas

- Proposição 1: $ax + by = c$ admite solução sse $\gcd(a, b) \mid c$

\Rightarrow

- Sendo (x_0, y_0) uma solução da equação
- Seja $d = \gcd(a, b)$, então $d \mid a$ e $d \mid b$. Logo podemos reescrever $a = Ad$ e $b = Bd$

$$\begin{aligned}
 c &= ax_0 + by_0 = (Ad)x_0 + (Bd)y_0 = \\
 &= d(Ax_0 + By_0)
 \end{aligned}$$

$$\text{Denotando } q = Ax_0 + By_0$$

$$c = dq$$

$$\text{Portanto, } d \mid c$$

Equações diofantinas

- Proposição 1: $ax + by = c$ admite solução sse $\gcd(a, b) \mid c$

\Leftarrow

- Seja $d = \gcd(a, b)$
- Pelo Teorema de Bézout, existe solução (x_0, y_0) para $ax + by = d$
- Por hipótese, $d \mid c \Rightarrow \exists t / c = dt$

$$\begin{aligned}
 c &= dt \\
 c &= (ax_0 + by_0)t \\
 c &= a(x_0t) + b(y_0t)
 \end{aligned}$$

Portanto, se $d \mid c$, então a equação $ax + by = c$ admite solução



Equações diofantinas

- Como determinar uma solução?
 1. Obter uma solução (x_0, y_0) para $ax + by = \gcd(a, b)$
 2. Para $ax + by = c$:
 - a) $t = c/d$ em que $d = \gcd(a, b)$
 - b) $x = x_0 t$
 - c) $y = y_0 t$
 3. Se uma equação diofantina tem uma solução, então ela tem infinitas:

$$S = \left\{ \left(x + \frac{b}{d}k, y - \frac{a}{d}k \right), k \in \mathbb{Z} \right\}$$

Equações diofantinas

- Solução para $ax + by = \gcd(a, b)$

1. Caso base ($a = 0$):

- Se $a = 0$ então temos $by = \gcd(0, b)$
- Sabemos que $\gcd(0, b) = b$
- Então $by = b$, logo $y = 1$
- Nesse caso x pode assumir qualquer valor. Como queremos uma solução qualquer, por motivos de simplificação, faremos $x = 0$
- Solução base: $(0, 1)$

Equações diofantinas

- Solução para $ax + by = \gcd(a, b)$

2. Passo da indução:

- Temos $ax + by = \gcd(a, b)$
- Pelo Algoritmo de Euclides, sabemos que

$$\gcd(a, b) = \gcd(b \% a, a) = d$$
- Logo

$$ax + by = d = (b \% a)x_1 + ay_1 \quad (*)$$

Equações diofantinas

- Solução para $ax + by = \gcd(a, b)$
 - Considerando o resultado da divisão inteira, podemos dizer que:

$$b = \frac{b}{a}a + b \% a$$

$$b \% a = b - \frac{b}{a}a$$

- Substituindo em (*)

$$\left(b - \frac{b}{a}a\right)x_1 + ay_1 = d$$

$$bx_1 - \frac{b}{a}ax_1 + ay_1 = d$$

$$a\left(y_1 - \frac{b}{a}x_1\right) + bx_1 = d$$

Equações diofantinas

- Solução para $ax + by = \gcd(a, b)$
 - Considerando o resultado da divisão inteira, podemos dizer que:

$$b = \frac{b}{a}a + b \% a$$

$$b \% a = b - \frac{b}{a}a$$

- Substituindo em (*)

$$\left(b - \frac{b}{a}a\right)x_1 + ay_1 = d$$

$$bx_1 - \frac{b}{a}ax_1 + ay_1 = d$$

$$a\left(y_1 - \frac{b}{a}x_1\right) + bx_1 = d$$

$$\underbrace{ax}_{ax} + \underbrace{by}_{by} = d$$



Equações diofantinas

- Implementação

```

int gcd(int a, int b, int &x, int &y){
    if (b == 0){
        x = 1;
        y = 0;
        return a;
    }
    int x1, y1;
    int d = gcd(b, a % b, x1, y1);
    x = y1;
    y = x1 - y1 * (a/b);
    return d;
}
  
```

Equações diofantinas

- Implementação

```

bool solve(int a, int b, int c, int &x0, int &y0, int &g) {
    g = gcd(abs(a), abs(b), x0, y0);
    if (c % g) {
        return false;
    }

    x0 *= c / g;
    y0 *= c / g;
    if (a < 0) x0 = -x0;
    if (b < 0) y0 = -y0;
    return true;
}
  
```

Aritmética Modular

- Em vários problemas precisamos operar com os restos de divisões de inteiros.
- A aritmética modular permite fazer cálculos com restos de divisões de modo eficiente, e é especialmente útil quando estamos trabalhando com números grandes (BigInteger).
- Na verdade, a Aritmética Modular pode nos ajudar a evitar ter que trabalhar com números muito grandes.

Aritmética Modular

- A aritmética modular se baseia nas seguintes propriedades:

$$(x + y) \% n = ((x \% n) + (y \% n)) \% n$$

$$(x - y) \% n = ((x \% n) - (y \% n)) \% n$$

$$(x * y) \% n = ((x \% n) * (y \% n)) \% n$$

$$(x ^ y) \% n = ((x \% n) ^ y) \% n$$

Aritmética Modular

- [UVa 374 - Big Mod](#)
 - Calcule $R = B^P \bmod M$
 - $0 \leq B, P \leq 2147483647$ e $1 \leq M \leq 46340$

Aritmética Modular

- Parte da solução do problema UVA 374 - Big Mod

```
long long pow(long long x, long long y, long long mod) {
    if (y == 0)
        return 1;

    long long p = pow(x, y/2, mod);
    if (y % 2 == 0)
        return (p * p) % mod;
    else
        return (((p * p) % mod) * (x % mod)) % mod;
}
```

Referências

Biblioteca de códigos de Thiago Alexandre Domingues de Souza.

Matemática Discreta e Suas Aplicações. Kenneth H. Rosen.

Programming Challenges: The Programming Contest Training Manual. Stevem S. Skiena e Miguel A. Revilla.

<https://www.geeksforgeeks.org/sieve-of-eratosthenes/>

<http://www.lcad.icmc.usp.br/~jbatista/scc210/AulaTeoriadosNumeros1.pdf>

<http://www.lcad.icmc.usp.br/~jbatista/scc210/AulaTeoriadosNumeros2.pdf>

https://www.ufsj.edu.br/portal2-repositorio/File/comat/tcc_Ricardo.pdf

<https://cp-algorithms.com/algebra/linear-diophantine-equation.html>

<https://noic.com.br/materiais-informatica/curso/math-02/>

<https://noic.com.br/materiais-informatica/curso/math-03/>