

Teoria dos Números

Notas de aula da disciplina
TE: Técnicas de Construção de Algoritmos

Fabiano de Souza Oliveira
(fabiano.oliveira@ime.uerj.br)

Paulo Eustáquio Duarte Pinto
(pauloedp@ime.uerj.br)

agosto/2020

Teoria dos Números: estudo de propriedades e algoritmos dos números inteiros, especialmente quanto à fatoração e divisibilidade.

Algoritmos básicos

Tabela de Números Primos/Fatoração

MDC - MMC - MDC Estendido

Aritmética Modular

Soma/Multiplicação modulares

Divisão modular

Equações modulares

Potenciação modular

Testes de Primalidade

Criptografia RSA

Fatoração de inteiros

Teorema Fundamental da Aritmética:

Dado um inteiro $n \geq 2$, existe um único modo de escrevê-lo na forma: $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$

Obs: O teorema não é verdadeiro para números complexos!!

Teste de primalidade, fatoração, divisores de inteiros

- a) $n \leq 10000$ - Força Bruta
- b) $10000 < n \leq 10^7$ - Crivo de Erastótenes
- c) $10^7 < n \leq 10^{15}$ - Crivo de Erastótenes + Tabela de Primos
- d) $10^{15} < n$ - Métodos Especiais
- e) Algoritmo AKS

I.a) Teste de primalidade para números pequenos (até 10.000)

Dado n , testa a divisibilidade por todos os números de 2 a $\text{Sqrt}(n)$.

```
função ÉPrimo(n):  
  para  $i \leftarrow 2$  até  $\lceil \sqrt{n} \rceil$  faça  
    se  $n \bmod i = 0$  então  
      retornar Falso  
  retornar Verdadeiro
```

I.b) Fatoração de números pequenos (até 10.000)

Dado n , testa a divisibilidade por todos os números de 2 a $\lceil \sqrt{n} \rceil$ e guarda os fatores no vetor F e a respectiva quantidade em nf .

Fatora(n , ref $F[]$, ref nf):

```
nf ← 0
para i ← 2 até  $\lceil \sqrt{n} \rceil$  faça
    enquanto  $n \bmod i = 0$  faça
        nf ← nf+1
         $F[nf]$  ← i
         $n$  ←  $n/i$ 
se  $n > 1$  então
    nf ← nf+1
     $F[nf]$  ← n
```

I.c) Divisores de números pequenos (até 10.000)

Dado n , testa a divisibilidade por todos os números de 2 a $\lceil \sqrt{n} \rceil$, guardando os divisores no vetor D e respectiva quantidade em nd .

```
Divisores( $n$ , ref  $D[]$ , ref  $nd$ ):  
   $nd \leftarrow 0$   
  para  $i \leftarrow 1$  até  $\lceil \sqrt{n} \rceil$  faça  
    se  $n \bmod i = 0$  então  
       $nd \leftarrow nd + 1$ ;  $D[nd] \leftarrow i$   
       $nd \leftarrow nd + 1$ ;  $D[nd] \leftarrow n/i$   
  se  $D[nd] = D[nd - 1]$  então  
     $nd \leftarrow nd - 1$ 
```

Crivo de Erastótenes

Serve para testar primalidade e fatorar números de 2 a n .

Usa um vetor de tamanho $n-1$ e marca, para cada elemento, o menor fator primo encontrado.

	2	3	4	5	6	7	8	9	10	..	15	16	..	25	..	77	..	121	..	143	144
2	2	3	2	5	2	7	2	9	2	..	15	2	..	25	..	77	..	121	..	143	2
3	2	3	2	5	2	7	2	3	2	..	3	2	..	25	..	77	..	121	..	143	2
5	2	3	2	5	2	7	2	3	2	..	3	2	..	5	..	77	..	121	..	143	2
7	2	3	2	5	2	7	2	3	2	..	3	2	..	5	..	7	..	121	..	143	2
11	2	3	2	5	2	7	2	3	2	..	3	2	..	5	..	7	..	11	..	11	2

Crivo de Erastótenes - Algoritmo de Geração:

Guarda, para cada número, seu menor fator.

GeraCrivo(n , ref $C[]$):

```
para  $i \leftarrow 1$  até  $n$  faça  
     $C[i] \leftarrow i$   
 $t \leftarrow 2$   
para  $i \leftarrow 1$  até  $n/2$  faça  
     $C[t] \leftarrow 2$ ;  $t \leftarrow t+2$   
para  $i \leftarrow 3$  até  $\lceil \sqrt{n} \rceil$  faça  
    se  $C[i] = i$  então  
         $t \leftarrow i*i$   
         $d \leftarrow i+i$   
        enquanto ( $t \leq n$ ) faça  
            se ( $C[t] = t$ ) então  
                 $C[t] \leftarrow i$   
             $t \leftarrow t+d$ 
```

II.a) Teste de primalidade para números até 10.000.000 com Crivo

ÉPrimo(n , C):

// C é o resultado da geração do Crivo até n

se $C[n] = n$ então

retornar Verdadeiro

senão

retornar Falso

II.b) Fatoração de números até 10.000.000 com Crivo

nf = número de fatores; F = vetor de fatores.

Fatora(n, C, ref F[], ref nf):

//C é o resultado da geração do Crivo até n

nf \leftarrow 0

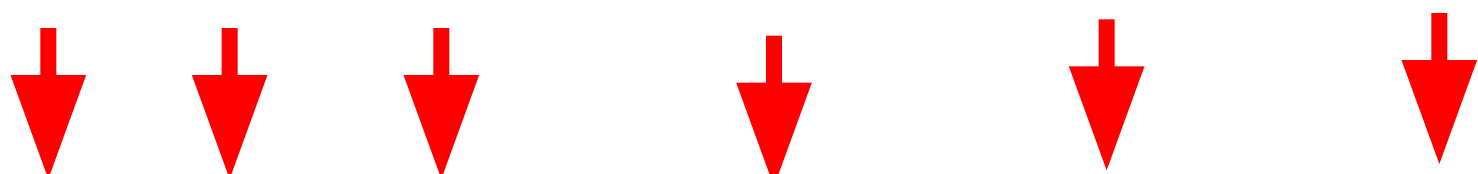
enquanto n \neq 1 faça

 nf \leftarrow nf + 1

 F[nf] \leftarrow C[n]

 n \leftarrow n/C[n]

Exemplo: fatoração de 468 = 2.2.3.3.13



q	1	13		39	...	117	...	234	...	468
v		13		3		3		2		2

II.c) Divisores de números até 10.000.000 com Crivo

$$n = 468 = 2 \cdot 2 \cdot 3 \cdot 3 \cdot 13$$

i	1	2	3	4	5
F	2	2	3	3	13
k	2	4	3	9	13

1																
---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

2

1	2															
---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

4

1	2	4														
---	---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--

3

1	2	4	3	6	12											
---	---	---	---	---	----	--	--	--	--	--	--	--	--	--	--	--

9

1	2	4	3	6	12	9	18	36								
---	---	---	---	---	----	---	----	----	--	--	--	--	--	--	--	--

13

1	2	4	3	6	12	9	18	36	13	26	52	39	78	156	117	234	468
---	---	---	---	---	----	---	----	----	----	----	----	----	----	-----	-----	-----	-----

II.c) Divisores de números até 10.000.000 com Crivo

Guarda nd divisores no vetor D . Usa sentinela 1 na pos. 0;
 Para n , a quantidade de primos é aproximadamente $O,1n$.

Divisores(n , F , nf , ref $D[]$, ref nd):

//(F , nf) é o resultado da geração de fatores de n

$F[0]$, nd , $D[1] \leftarrow 1, 1, 1$

para $i \leftarrow 1$ até nf faça

se $F[i] = F[i-1]$ então

$k \leftarrow k * F[i]$

senão

$k \leftarrow F[i]$

$nda \leftarrow nd$

para $j \leftarrow 1$ até nda faça

$nd \leftarrow nd + 1$

$D[nd] \leftarrow k * D[j]$

Crivo de Erastótenes - Tabela de Primos:

Guarda, no vetor P , os números primos até n , e a respectiva quantidade em np .

GeraPrimos(n , C , ref $P[]$, ref np):

// C é o resultado da geração do Crivo até n

$np \leftarrow 0$

para $i \leftarrow 2$ até n faça

se $C[i] = i$ então

$np \leftarrow np + 1$

$P[np] \leftarrow i$

III.a) Teste de primalidade para números entre 10^7 e 10^{15} usando Tabela de Primos

Testa primalidade de $10^7 \leq n \leq 10^{15}$

```
ÉPrimo(n, P, np):  
  //(P, np) é o resultado da geração de primos até  $\lceil \sqrt{n} \rceil$   
  para i  $\leftarrow$  1 até np faça  
    se  $n \bmod P[i] = 0$  então  
      se  $n \neq P[i]$  então  
        retornar Falso  
      retornar Verdadeiro  
  retornar Verdadeiro
```

III.b) Fatoração de números entre 10^7 e 10^{15} usando Tabela de Primos

Fatora $10^7 \leq n \leq 10^{15}$

Fatora(n , P , np , ref $F[]$, ref nf):

//(P , np) é o resultado da geração de primos até $\lceil \sqrt{n} \rceil$

$nf \leftarrow 0$

para $i \leftarrow 1$ até np faça

 enquanto $n \bmod P[i] = 0$ faça

$nf \leftarrow nf + 1$

$F[nf] \leftarrow P[i]$

$n \leftarrow n / P[i]$

 se $n = 1$ então

 sair-para

se $n \neq 1$ então

$nf \leftarrow nf + 1$

$F[nf] \leftarrow n$

III.c) Divisores de números entre 10^7 e 10^{15} usando Tabela de Primos.

Algoritmo idêntico ao do caso de números até 10.000.000

Divisores(n , F , nf , ref $D[]$, ref nd):

/(F , nf) é o resultado da geração de fatores de n

$F[0]$, nd , $D[1] \leftarrow 1, 1, 1$

para $i \leftarrow 1$ até nf faça

se $F[i] = F[i-1]$ então

$k \leftarrow k * F[i]$

senão

$nda \leftarrow nd$

$k \leftarrow F[i]$

para $j \leftarrow 1$ até nda faça

$nd \leftarrow nd + 1$

$D[nd] \leftarrow k * D[j]$

Questões relacionadas a primos

Teorema: Existem infinitos números primos.

Número de números primos $\pi(n)$ (Gauss):

$$\lim_{n \rightarrow \infty} \pi(n) / \ln n = 1$$

Conjectura de Goldbach:

Todo número par ≥ 4 é a soma de 2 primos

Teorema de Lagrange:

Todo número pode ser escrito como a soma de, no máximo, 4 quadrados.

Questões relacionadas a primos

Números de Mersenne: $M(n) = 2^n - 1$

Questão: $M(n)$ é primo quando n é primo?

R: Não. É primo p/ $n = 2, 3, 5, 7, 13, 17, 19, 31, 61, 89$

Números de Fermat: $F(n) = 2^{2^n} + 1$

Questão: $F(n)$ é primo?

R: Não. É primo p/ $n = 0, 1, 2, 3, 4$. Não é primo para n entre 5 e 32. Não são conhecidos outros primos de Fermat.

Questões relacionadas a primos

Seja $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ (fatoração por primos)

A quantidade $qd(n)$ de divisores de n é dada por:

$$qd(n) = (e_1 + 1)(e_2 + 1) \dots (e_k + 1)$$

Ex: $70 = 2 \cdot 5 \cdot 7$

$$\Rightarrow qd(70) = (1+1) \cdot (1+1) \cdot (1+1) = 2 \cdot 2 \cdot 2 = 8$$

Os divisores são: 1, 2, 5, 7, 10, 14, 35, 70

Questões relacionadas a primos

Seja $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ (fatoração por primos)

A função Totiente de Euler indica a quantidade de números relativamente primos a n e é dada por:

$$\text{grp}(n) = (p_1^{e_1} - p_1^{e_1-1}) \dots (p_k^{e_k} - p_k^{e_k-1})$$

Ex: $70 = 2^1 \cdot 5^1 \cdot 7^1$

$$\text{grp}(70) = (2^1 - 2^0) \cdot (5^1 - 5^0) \cdot (7^1 - 7^0) = (2-1)(5-1)(7-1) = 24$$

Números relativamente primos a 70:

1, 3, 9, 11, 13, 17, 19, 23, 27, 29, 31, 33, 37, 39,
41, 43, 47, 51, 53, 57, 59, 61, 67, 69

Questões relacionadas a primos

Seja $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ (fatoração por primos)

A soma dos divisores $sd(n)$ de n é dada por:

$$sd(n) = ((p_1^{e_1+1} - 1)/(p_1 - 1)) \cdot \dots \cdot ((p_k^{e_k+1} - 1)/(p_k - 1))$$

Ex: $70 = 2^1 \cdot 5^1 \cdot 7^1$

$$\begin{aligned} sd(70) &= ((2^2 - 1)/(2 - 1)) \cdot ((5^2 - 1)/(5 - 1)) \cdot ((7^2 - 1)/(7 - 1)) = \\ &= (3/1) \cdot (24/4) \cdot (48/6) = 3 \cdot 6 \cdot 8 = 144 \end{aligned}$$

Soma dos divisores de 70: $1 + 2 + 5 + 7 + 10 + 14 + 35 + 70 = 144$

Fatoração de inteiros

Exercícios:

- a) Calcular a quantidade de divisores de 36 ou 50.
- b) Calcular a quantidade de números relativamente primos a 36 ou 50.

Máximo Divisor Comum

MDC(a,b) - Algoritmo de Euclides

```
inteiro MDC (a, b):  
  enquanto (b  $\neq$  0):  
     $f \leftarrow a \bmod b$   
     $a \leftarrow b$   
     $b \leftarrow f$   
  retornar a;
```

a	b
99	78
78	21
21	15
15	6
6	3
3	0

Mínimo Múltiplo Comum

$MMC(a,b)$ - Baseado em MDC

$$MMC(a,b) = a*b/(MDC(a,b))$$

inteiro $MMC(a, b)$:

$d \leftarrow MDC(a, b)$

$m \leftarrow (a*b)/d$

retornar m

a	b
99	78
d	3
m	$99*78/3 =$ 2574

Máximo Divisor Comum

MDCE(a,b,x,y) - Algoritmo de Euclides Estendido

Objetivo: dados a e b , encontrar $d = \text{MDC}(a, b)$ e também x e y , tais que $a.x + b.y = d$

Exemplo:

$\text{MDCE}(8,6,x,y) = 2$ e $x = 1, y = -1$, pois

$\text{MDC}(8, 6) = 2$ e $8.1 + 6.(-1) = 2$

MDCE(a,b) - Algoritmo Estendido

Objetivo: dados a e b, encontrar $d = \text{MDC}(a, b)$ e também x e y, tais que $a.x + b.y = d$

Exemplo: Calcular MDCE para 99 e 78.

a	b	d	x	y	Equação
99	78	3	-11	14	$99.(-11) + 78.14 = 3$
78	21	3	3	-11	$78.3 + 21.(-11) = 3$
21	15	3	-2	3	$21.(-2) + 15.3 = 3$
15	6	3	1	-2	$15.1 + 6.(-2) = 3$
6	3	3	0	1	$6.0 + 3.1 = 3$
3	0	3	1	0	$3.1 + 0.0 = 3$

MDCE(a,b) - Algoritmo de Euclides Estendido (Versão não recursiva)

```
(inteiro, inteiro, inteiro) MDCE (a, b):  
  t ← 0;  x ← 1;  y ← 0;  
  enquanto (b ≠ 0):  
    t ← t+1  
    Pa[t] ← a;    Pb[t] ← b;  
    f ← a mod b;  a ← b;          b ← f;  
  para i ← t..1 incl.:  
    f ← y  
    y ← x - [Pa[i]/Pb[i]]*y  
    x ← f  
  retornar (a, x, y)
```

MDCE(a,b) - Algoritmo Estendido

Objetivo: dados a e b , encontrar $d = \text{MDC}(a, b)$ e também x e y , tais que $a.x + b.y = d$

Exemplo: Calcular MDCE para 99 e 78.

a	b	d	x	y	Equação
99	78	3	-11	14	$99.(-11) + 78.14 = 3$
78	21	3	3	-11	$78.3 + 21.(-11) = 3$
21	15	3	-2	3	$21.(-2) + 15.3 = 3$
15	6	3	1	-2	$15.1 + 6.(-2) = 3$
6	3	3	0	1	$6.0 + 3.1 = 3$
3	0	3	1	0	$3.1 + 0.0 = 3$

MDCE(a,b) - Algoritmo de Euclides Estendido (Versão recursiva)

```
(inteiro, inteiro, inteiro) MDCE (a, b):  
  se (b = 0):  
    retornar (a, 1, 0)  
  senão:  
     $(d', x', y') \leftarrow \text{MDCE}(b, a \bmod b)$   
     $(d, x, y) \leftarrow (d', y', x' - \lfloor a/b \rfloor * y')$   
    #escrever (a, b, d, x, y):  
  retornar (d, x, y)
```

MDCE(a,b) - Algoritmo de Euclides Estendido

P: **Porque funciona?**

R: **Porque temos uma propriedade invariante.**

Sejam os parâmetros de duas chamadas recursivas sucessivas: (a_1, b_1, x_1, y_1) e (a_2, b_2, x_2, y_2) , com

$$\begin{aligned} a_2 &= b_1, & b_2 &= a_1 \bmod b_1, \\ x_1 &= y_2, & y_1 &= x_2 - \lfloor a_1/b_1 \rfloor \cdot y_2 \\ a_1 \bmod b_1 &= (a_1 - \lfloor a_1/b_1 \rfloor \cdot b_1) \end{aligned}$$

Se tivermos $a_2 x_2 + b_2 y_2 = d$, então
 $b_1 x_2 + (a_1 \bmod b_1) y_2 = d$ e

$$\begin{aligned} a_1 x_1 + b_1 y_1 &= a_1 y_2 + b_1 (x_2 - \lfloor a_1/b_1 \rfloor \cdot y_2) = \\ b_1 x_2 + (a_1 - \lfloor a_1/b_1 \rfloor \cdot b_1) y_2 &= b_1 x_2 + (a_1 \bmod b_1) y_2 = d \end{aligned}$$

MDCE(a,b) - Algoritmo Estendido

Objetivo: dados a e b , encontrar $d = \text{MDC}(a, b)$ e também x e y , tais que $a.x + b.y = d$

Exercício: Calcular MDCE para 1104 e 828.

MDCE(a,b,x,y) - Algoritmo Estendido - Aplicações

a) soluções inteiras para equações

Ex: Problema (10090) - Marbles

dadas 43 bolinhas e dois tipos de caixas, com custos e capacidades (1, 3) e (3, 4), qual a maneira mais barata de guardar as bolinhas?

Solução: temos que encontrar x e y tal que
 $3x + 4y = 43$ e $x + 3y$ seja mínimo

b) ferramenta da aritmética modular

MDCE(a,b) - Algoritmo Estendido

Exemplo: Quais as soluções inteiras para
 $99x + 78y = 30$.

Solução básica: Como $99 \cdot (-11) + 78 \cdot 14 = 3 \Rightarrow$
 $99 \cdot (-110) + 78 \cdot (140) = 30$

Outras soluções:

$$99(-110 + a) + 78(140 + b) = 30.$$

Devemos ter:

$$99a + 78b = 0 \Rightarrow 99a = -78b \Rightarrow a = -26b/33.$$

Portanto para qualquer b múltiplo de 33
temos uma solução distinta.

Aritmética modular

Está relacionada a fenômenos cíclicos. Por exemplo a aritmética das horas é modular: $13 + 18 = 7$.

Relações de equivalência: Relações em um conjunto onde valem as propriedades reflexiva, simétrica e transitiva. O conjunto é particionado em classes de equivalência.

Aritmética modular

Congruência: Relação definida em \mathbb{Z} .

Dizemos que a e b são congruentes módulo n quando n divide $(a-b)$, o que é a mesma coisa que dizer que $a \bmod n = b \bmod n$.

Uma congruência em \mathbb{Z} é uma relação de equivalência em \mathbb{Z} e, portanto, particiona \mathbb{Z} em classes de equivalência.

Notação: $a \equiv b \bmod n$

$$\mathbb{Z}_n = \{\underline{0}, \underline{1}, \dots, \underline{n-1}\}$$

$$\underline{a} = \{a + kn : k \in \mathbb{Z}\}$$

Aritmética modular - Operações +, - e *

Teorema: Se $a \equiv b \pmod{n}$ então:

$$1. a + c \equiv b + c \pmod{n} \text{ ou } \underline{(a+c)} = \underline{(b+c)}$$

$$2. a - c \equiv b - c \pmod{n} \text{ ou } \underline{(a-c)} = \underline{(b-c)}$$

$$3. ac \equiv bc \pmod{n} \quad \text{ou} \quad \underline{ac} = \underline{bc}$$

Aritmética modular - Operações +, - e *

Exemplos:

Como $16 \equiv 3 \pmod{13}$, temos

$$16 + 11 \equiv 3 + 11 \pmod{13} \equiv 1 \pmod{13}.$$

$$16 - 11 \equiv 3 - 11 \pmod{13} \equiv -8 \pmod{13} \\ \equiv 5 \pmod{13}.$$

$$16 \cdot 11 \equiv 3 \cdot 11 \pmod{13} \equiv 33 \pmod{13} \\ \equiv 7 \pmod{13}.$$

$$(16 + 11) \pmod{13} =$$

$$(16 \pmod{13} + 11 \pmod{13}) \pmod{13} =$$

$$(3 + 11) \pmod{13} = 14 \pmod{13} =$$

$$1 \pmod{13}$$

Exemplo de uso computacional da Aritmética modular

Problema: Dado $p = d_{1000}d_{999}\dots d_1d_0$ (1001 dígitos),
 p é divisível por $n = 13$?

Solução:

p pode ser escrito como

$$p = (\dots(d_{1000}).10 + d_{999}).10 + \dots + d_1).10 + d_0$$

o que sugere o seguinte algoritmo:

$q \leftarrow 0$;

Para i decrescendo de 1000 a 0:

$q \leftarrow (q \cdot 10 + d_i) \bmod n$;

Fp;

Se $(q = 0)$ Então Escrever "n divide p"

Senão Escrever "n não divide p";

Problemas computacionais da Aritmética modular

Problema 1: As operações $(a+c) \bmod b$ ou $(a*c) \bmod b$ podem apresentar overflow!

Solução: Se houver perigo de overflow, usar:

$$\begin{aligned} & ((a \bmod b) + (c \bmod b)) \bmod b \text{ ou} \\ & ((a \bmod b) * (c \bmod b)) \bmod b \end{aligned}$$

Exemplo: $(802507 * 354197) \bmod 7$ retorna 5,

$((802507 \bmod 7) * (354197 \bmod 7)) \bmod 7$
retorna 2, quando os números são
representados em 32 bits;

Problemas computacionais da Aritmética modular

Problema 2: A operação $a \bmod b$ pode retornar valores negativos!

Solução: Quando a puder ser negativo, usar

$$((a \bmod b) + b) \bmod b$$

Exemplo:

$-5 \bmod 7$ retorna -5 ,

$((-5 \bmod 7) + 7) \bmod 7$ retorna 2 ;

Testes de Primalidade para grandes números - Pequeno Teorema de Fermat

Teorema: Seja p primo e a inteiro. Então:
 $a^p \equiv a \pmod{p}$, ou: $a^{p-1} \equiv 1 \pmod{p}$.

Exemplos:

$$3^2 = 9 \equiv 1 \pmod{2};$$

$$3^{2-1} = 3 \equiv 1 \pmod{2};$$

$$(-7)^5 = -16807 \equiv -7 \pmod{5}$$

$$(-7)^{5-1} = 2401 \equiv 1 \pmod{5}$$

Testes de Primalidade para grandes números - Pequeno Teorema de Fermat

Outros exemplos:

$$26^{11} = 3670344486987776 = \\ = 333667680635252 \cdot 11 + 4 \equiv 4 \pmod{11}$$

$$26^{11-1} = 141167095653376 = \\ = 12833372332125 \cdot 11 + 1 \equiv 1 \pmod{11}$$

$$2^{5432974} \pmod{13} = 2^{452747 \cdot 12 + 10} \pmod{13} = \\ (2^{452747 \cdot 12} \pmod{13} \cdot 2^{10} \pmod{13}) \pmod{13} \\ = 2^{10} \pmod{13} = 1024 \pmod{13} = 10$$

Testes de Primalidade para grandes números

Primeiro teste: dado n , usar uma ou mais bases e verificar o pequeno teorema de Fermat. O teste é conclusivo para números compostos mas pode falhar para números primos.

Exemplos:

P: 343 é primo?

R: Não, pois $2^{343} \equiv 324 \pmod{343}$

P: 341 é primo?

R: Não, pois $3^{341} \equiv 168 \pmod{341}$

P: 3523721 é primo?

R: Não, pois $2^{3523720} \equiv 1926315 \pmod{3523721}$

Testes de Primalidade - Números de Carmichael

Números de Charmichael são números n onde o teste de Fermat é inconclusivo para todas as bases b , $1 < b < (n-1)$, na 1ª versão do Teorema.

Exemplos:

P: 341 é primo?

R: Não, pois $2^{341} \equiv 2 \pmod{341}$, mas
 $3^{341} \equiv 168 \pmod{341}$,

P: 561 é primo?

R: Não sei, pois $a^{561} \equiv a \pmod{561}$, para
 $1 < a < 561$. (Não é primo pois $561 = 3 \cdot 187$)
561 é o menor número de Carmichael!

Outros números : 1105, 1729, 2465, 2821, 6601, 8911.

Potenciação modular - Versão não recursiva

Para calcular $a^b \bmod n$, onde b e n são inteiros positivos, a idéia é usar a representação binária de b $\langle b_k, b_{k-1}, \dots, b_1, b_0 \rangle$ e usar multiplicação e soma modulares.

Exemplo: calcular $3^6 \bmod 7$

$$6_{10} = 110_2 = (((1).2+1).2+0)_{10}$$

$$3^1 \bmod 7 = 3$$

$$3^2 \bmod 7 = 2$$

$$2.3^1 \bmod 7 = 6$$

$$6^2 \bmod 7 = 1$$

$$1.3^0 \bmod 7 = 1$$

$$\begin{aligned} \text{E, } 3^6 \bmod 7 &= 729 \bmod 7 = \\ &= (104.7 + 1) \bmod 7 = 1 \end{aligned}$$

Potenciação modular - Versão não recursiva

Para calcular $a^b \bmod n$, onde b e n são inteiros positivos, a idéia é usar a representação binária de b $\langle b_k, b_{k-1}, \dots, b_1, b_0 \rangle$ e usar multiplicação.

inteiro PotenciaModular(a , b , n):

$d \leftarrow 1$

$c \leftarrow 0$

para $i \leftarrow k..0$ incl.:

$d \leftarrow (d * d) \bmod n$

$c \leftarrow 2 * c$

se $(b_i = 1)$:

$d \leftarrow (d * a) \bmod n$

$c \leftarrow c + 1$

retornar d ;

Aritmética modular - Potenciação modular

Exemplo: $a = 5$, $b = 230 = \langle 11100110 \rangle$,
 $n = 337$;

i	7	6	5	4	3	2	1	0
b_i	1	1	1	0	0	1	1	0
c	1	3	7	14	28	57	115	230
d	5	125	278	111	189	332	125	123

Potenciação modular - Versão recursiva

Recorrência: Seja $P_m(a, b, n) = a^b \bmod n$.

$P_m(a, b, n) = P_m(a, b/2, n)^2 \bmod n$, $b > 1$, par

$P_m(a, b, n) = (a \cdot P_m(a, \lfloor b/2 \rfloor, n)^2) \bmod n$, $b > 1$, ímpar

$P_m(a, 0, n) = 1$;

inteiro PotenciaModular(a, b, n);

se (b = 0):

retornar 1

senão:

$x \leftarrow \text{PotenciaModular}(a, \lfloor b/2 \rfloor, n)$

se (b é par):

retornar $(x * x) \bmod n$

senão:

retornar $(a * x * x) \bmod n$

Aritmética modular - Potenciação modular

Exercício: Calcular $34^{11} \bmod 24$

Criptografia RSA - é um sistema de chave pública baseado na Aritmética modular.

Para entender o método é necessário se entender, primeiro, a divisão modular e os inversos modulares., que são baseadas no Algoritmo de Euclides Estendido.

Aritmética modular - Divisão modular

A operação é estendida mesmo quando p/q não é inteiro. Só pode ser feita quando

$$\text{MDC}(q, n) = 1.$$

Neste caso, dado q , existe a classe inversa de q , q' , tal que $q \cdot q' = 1$.

⇒ O problema é transformado em multiplicação modular.

Aritmética modular - Divisão modular

Exemplo:

Como $\text{MDC}(3, 32) = 1$, podemos verificar que a classe inversa de 3 é 11, pois $(3 \cdot 11 + 32 \cdot (-1) = 1)$

$$(3 \cdot 11 = 33 \equiv 1 \pmod{32}).$$

Então: $(813/3) \bmod 32 = (813 \cdot 11) \bmod 32 = 8943 \bmod 32 = 15.$

Aritmética modular - Divisão modular

O inverso modular é calculado usando-se o MDC estendido, pois $a \cdot a' \equiv 1 \pmod{n}$ equivale a achar x e y tal que $a \cdot x + y \cdot n = 1$, onde $x = a'$

Exemplo:

Para achar $\underline{3'}$ em Z_{32} , aplicamos o MCDE a $(3, 32)$ e obtemos:

$$3 \cdot 11 + 32 \cdot (-1) = 1$$

Logo, $\underline{3'} = \underline{11}$

a	b	x	y
3	32	11	-1
32	3	-1	11
3	2	1	-1
2	1	0	1
1	0	1	0

Aritmética modular - Divisão modular

Dado \mathbb{Z}_n , nem todos seus elementos têm inverso modular.

Chamamos U_n o conjunto das classes que têm inverso.
 $U_n = \{ \underline{a} \in \mathbb{Z}_n \mid \text{MDC}(a, n) = 1 \}.$

Se n é primo, $U_n = \mathbb{Z}_n \setminus \underline{0}.$

Exemplos: $U_4 = \{ \underline{1}, \underline{3} \},$

$$U_7 = \{ \underline{1}, \underline{2}, \underline{3}, \underline{4}, \underline{5}, \underline{6} \}$$

$$U_8 = \{ \underline{1}, \underline{3}, \underline{5}, \underline{7} \}$$

Criptografia RSA - é um sistema de chave pública baseado na Aritmética modular.

Resumo do método:

- a) Selecionar primos p , q e calcular $n = p \cdot q$ e $t = (p-1)(q-1)$.
- b) Selecionar f pequeno, relativamente primo a t .
- c) Calcular g = inv. modular de f relativo a t .
- d) Chave pública é o par (f, n) , secreta, (g, n) .
- e) A codificação da mensagem M é $C = M^f \bmod n$.
- f) A decodificação é $M = C^g \bmod n$.

Criptografia RSA - Exemplo com núm pequenos.

- a) Sejam $p = 31$, $q = 43$, $\Rightarrow n = 1333$, $t = 1260$.
- b) Podemos fazer $f = 11$, relativ. primo a 1260 .
- c) O inverso modular de 11, relativamente a 1260 é $g = -229$, que mudamos para $g = 1260 - 229 = 1031$.
- d) Chave pública: $(11, 1333)$, secreta: $(1031, 1333)$.
- e) Codificação de 1241: $1241^{11} \bmod 1333 = 652$.
- f) Decodificação de 652: $652^{1031} \bmod 1333 = 1241$.

Criptografia RSA - Porque funciona?

- a) Seja $0 \leq b < n$. Devemos ter $b^{e \cdot d} \bmod n = b$.
- b) Mas $e \cdot d = 1 + k \cdot t$, pois e é o inv. modular de e .
- c) Mas $b^{k \cdot t} \equiv 1 \bmod n$. (Teorema de Euler).
- d) Portanto $b^{e \cdot d} \equiv b \bmod n$.
- e) Como $0 \leq b < n \Rightarrow b^{e \cdot d} \bmod n = b$.

Criptografia RSA - Qual a segurança?

- a) A segurança do método é a DIFICULDADE de fatoração de n . Não são conhecidos métodos eficientes de fatoração.
- b) Suponhamos que se possa obter t facilmente a partir da chave pública (n e e).
- c) Então temos: $t = (p-1)(q-1) = n - (p+q) + 1$. \Rightarrow podemos calcular $p+q$.
- d) Como $(p-q)^2 = (p+q)^2 - 4n$, podemos calcular $p-q$, e, como consequência, p e q .
- b) Logo, a dificuldade de obter t é, no mínimo, igual à de fatorar n .

FIM.