UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA

MESTRADO PROFISSIONAL EM MATEMÁTICA - PROFMAT

RICARDO PEREIRA DE ANDRADE

TESTES DE PRIMALIDADE: UMA ANÁLISE MATEMÁTICA DOS ALGORITMOS DETERMINÍSTICOS E PROBABILÍSTICOS

FORTALEZA

2017

RICARDO PEREIRA DE ANDRADE

TESTES DE PRIMALIDADE: UMA ANÁLISE MATEMÁTICA DOS ALGORITMOS DETERMINÍSTICOS E PROBABILÍSTICOS

Dissertação apresentada ao Curso de Mestrado Profissional em Matemática - PROFMAT do Programa de Pós-Graduação em Matemática do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Matemática. Área de Concentração: Matemática

Orientador: Prof. Dr. Marcelo Ferreira de Melo

FORTALEZA

2017

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A57t Andrade, Ricardo Pereira de.
Testes de primalidade : uma análise matemática dos algoritmos determinísticos e probabilísticos / Ricardo Pereira de Andrade. – 2017.
51 f.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Departamento de Matemática, Programa de Pós-Graduação em Matemática em Rede Nacional, Fortaleza, 2017.
Orientação: Prof. Dr. Marcelo Ferreira de Melo.

1. Algoritmo. 2. Teste de Primalidade. 3. Números Primos. I. Título.

CDD 510

RICARDO PEREIRA DE ANDRADE

TESTES DE PRIMALIDADE: UMA ANÁLISE MATEMÁTICA DOS ALGORITMOS DETERMINÍSTICOS E PROBABILÍSTICOS

Dissertação apresentada ao Curso de Mestrado Profissional em Matemática - PROFMAT do Programa de Pós-Graduação em Matemática do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Matemática. Área de Concentração: Matemática

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Marcelo Ferreira de Melo (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Antônio Caminha Muniz Neto
Universidade Federal do Ceará (UFC)

Prof. Dr. Ângelo Papa Neto
Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

Aos meus pais José e Neves, ao meu irmão Ronaldo, a minha esposa Gislene, a minha cunhada Gisleuda, a toda minha família e a Deus, por me renovar a cada manhã.

AGRADECIMENTOS

A todos os meus familiares que me incentivaram e sempre acreditaram em minha capacidade, em especial, a minha esposa Gislene.

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

Ao Prof. Dr. Marcelo Ferreira de Melo por me orientar neste trabalho.

Por fim, todos que, direta ou indiretamente, contribuiram para a realização e conclusão deste sonho o meu muito obrigado.

"O problema de distinguir os números primos dos números compostos e de exprimir estes últimos à custa de fatores primos deve ser considerado como um dos mais importantes e úteis em Aritmética."

(K. F. Gauss)

RESUMO

Este trabalho apresenta os principais testes de primalidade através de um estudo sobre a base matemática utilizada para garantir a validade de cada método, bem com uma analise da complexidade computacional dos mesmos.

Palavras-chave: Algoritmo. Teste de Primalidade. Números Primos.

ABSTRACT

This work presents the main tests of primality through a study on the mathematical basis used to guarantee the validity of each method, as well as an analysis of the computational complexity of the same.

Keywords: Algorithm. Primality test. Prime numbers.

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Os números primos desempenham um papel fundamental na Matemática, pois todo número natural maior que 1 ou é primo ou é formado pelo produto de fatores primos (chamado composto).

Por isso, descobrir um método eficaz para verificação da primalidade de um número é um problema que tem captado o interesse de matemáticos desde a antiguidade. O Crivo de Eratóstenes é o algoritmo mais antigo que se tem conhecimento; no entanto, sua complexidade de tempo o torna inviável para encontrar primos grandes (COUTINHO, 2004).

Nos últimos anos, com o advento de sistemas de criptografia, que são baseados na dificuldade da fatoração de números grandes, a necessidade de identificar números primos ganhou importância prática.

O sistema RSA, nome formado pelas iniciais de seus três idealizadores, Rivest, Shamir e Adleman, é um algoritmo que utiliza essa dificuldade de fatoração e o fato de que todo número natural pode ser escrito com o produto de fatores primos de forma única, a menos da ordem, para criar uma chave pública, que é formada pelo produto de dois números primos. Dessa forma, dificilmente a criptografia será revertida sem a chave privada, ou seja, os números primos usados na multiplicação.

Atualmente, existem diversos algoritmos que permitem determinar se um dado número é primo ou composto, os quais estão divididos em duas categorias: determinísticos – que retornam a primalidade com total confiabilidade – e probabilísticos – que estão corretos quando verificam que um número é composto, mas podem errar ao afirmar que um número é primo, sendo neste caso, possivelmente primos.

Em aplicações criptográficas, são utilizados testes de primalidade probabilísticos ou randomizados, uma vez que estes apresentam um desempenho superior aos testes determinísticos e têm um percentual de erro irrelevante para situações práticas, pois a tarefa de identificação dos fatores do produto resultante de números provavelmente primos continua sendo extremamente trabalhosa.

Sem a pretensão de ser uma revisão bibliográfica exaustiva concernindo a testes de primalidade, este trabalho se pautará no estudo dos principais algoritmos – determinísticos e probabilísticos - através de uma análise da eficiência e dos conceitos matemáticos utilizados para a validação da solução.

Sendo assim, iniciamos esta pesquisa apresentando, no capítulo 2, conceitos e

teoremas sobre números inteiros, os quais serão imprescindíveis para a compreensão dos capítulos posteriores. Para que possamos fazer a análise dos algoritmos mostrados, faz-se necessário entender como é determinada a complexidade de execução dos mesmos, assunto que trataremos no terceiro capítulo. No capítulo 4, abordaremos dos testes de primalidade determinísticos, desde métodos simples como o crivo de Eratóstenes e divisões sucessivas, a soluções mais sofisticadas tais como os testes AKS e de Lucas-Lehmer. Os algoritmos probabilísticos serão analisados no capítulo 5, dentre os quais os de Solovay-Strassen e Miller-Rabin (o mais utilizado atualmente). No capítulo seguinte, mostraremos o Teorema de Lucas, que funciona como uma espécie de "certificado de primalidade", pois se conseguirmos fazer o número passar pelo teste podemos afirmar que o mesmo é primo. Por fim, concluímos fazendo um estudo comparativo da complexidade de execução dos diversos algoritmos apresentados.

2 PRELIMINARES

Neste capítulo, apresentaremos os conceitos, propriedades e resultados sobre números inteiros que serão necessários no decorrer do texto.

2.1 Números Primos

Definição 2.1.1. *Sejam a e b números inteiros. Dizemos que a divide b, ou que a é um divisor de b ou, ainda b é um múltiplo de a e denotado por* $a \mid b$*, se e somente se existe um inteiro c tal que* $b = ac$*. Quando a não divide b representamos por* $a \nmid b$*.*

Notamos que c é único quando $a \neq 0$, pois se existisse d satisfazendo $b = ad$, teríamos: $0 = ac - ad = a(c-d) \Rightarrow c - d = 0 \Rightarrow c = d$.

Definição 2.1.2. *Sejam a e b números inteiros não simultaneamente nulos. Um natural d chama-se máximo divisor comum de a e b, denotado por* $mdc(a,b)$ *ou* (a,b) *se possuir as seguintes propriedades:*
(i) d é divisor comum de a e de b, e
(ii) d é um múltiplo de todo divisor comum de a e b.

Definição 2.1.3. *Um inteiro positivo* $p > 1$ *é um número primo, se e somente se, possui apenas dois divisores: 1 e p. Quando não é primo, p diz-se composto.*

Definição 2.1.4. *Sejam a e b números inteiros não nulos. Se* $(a,b) = 1$*, dizemos que a e b são primos entre si.*

Teorema 2.1.1 (Teorema Fundamental da Aritmética)**.** *Seja* $n \geq 2$ *um número natural. Podemos escrever n de uma única forma como um produto* $n = p_1.p_2...p_r$ *onde* $r \geq 1$ *é um natural e* $p_1 \leq p_2 \leq ... \leq p_r$ *são primos.*

Prova: Mostramos a existência da fatoração de n em primos por indução. Se n é primos escrevemos $r = 1$ e $p_1 = n$. Sendo n composto podemos escrever $n = ab$ com $a,b \in \mathbb{N}$ e $1 < a,b < n$. Por hipótese de indução, a e b se decompõem como o produto de primos. Juntando as fatorações de a e b (e reordenando os fatores) obtemos uma fatoração de n. Para provar a unicidade, vamos supor por absurdo que n possui duas fatorações diferentes $n = p_1.p_2...p_r = q_1.q_2...q_s$, com $p_1 \leq p_2 \leq ... \leq p_r$ e $q_1 \leq q_2 \leq ... \leq q_s$. Como $p_1 \mid q_1.q_2...q_s$, temos que $p_1 \mid q_i$ para algum valor de i tal que $1 \leq i \leq s$. Logo, como q_i é primo, então $p_1 = q_i$. Da mesma forma,

$q_1 = p_j$ para algum j tal que $1 \leq j \leq r$. Mas, como $p_1 = q_i \geq q_1 = p_j \geq p_1$, então $p_1 = q_1$. Assim, $\frac{n}{p_1} = p_2...p_r = q_2...q_s$ e a hipótese de indução garante que $p_i = q_i$ para todo i, o que contradiz o fato de n ter duas fatorações. □

Teorema 2.1.2 (Euclides)**.** *Existem infinitos números primos.*

Prova: Suponhamos, por absurdo, que $p_1, p_2, ..., p_m$ fossem todos os primos existentes. Assim, o número $p_1.p_2...p_m + 1$ não seria divisível por nenhum primo p_i, contrariando o Teorema 2.1.1. □

Proposição 2.1.3. *Sejam a e n números naturais maiores do que 1. Se $a^n - 1$ é primo, então $a = 2$ e n é primo.*

Prova: Se $a > 2$, então $a - 1 > 1$ e $(a-1) \mid (a^n - 1)$, portanto, $a^n - 1$ não é primo. Consequentemente, $a^n - 1$ primo implica $a = 2$. Por outro lado, suponha, que n não é primo, digamos, $n = rs$ com $r > 1$ e $s > 1$. Como $(2^r - 1) \mid ((2^r)^s - 1) = 2^n - 1$, segue que $2^n - 1$ não é primo. Logo, $a^n - 1$ primo implica n é primo. □

Os números da forma $M_p = 2^p - 1$ são denominados números de Mersenne; quando são primos, os chamamos de primos de Mersenne. Portanto, não necessariamente, $M_p = 2^p - 1$ é primo sempre que p for primo; por exemplo, $2^{11} - 1 = 2047 = 23 \times 89$.

2.2 Aritmética Modular

Definição 2.2.1. *Sejam a, b e m números inteiros, $m > 1$. Então, se diz que a é congruente a b módulo m, e escreve-se $a \equiv b \pmod{m}$, se, e somente se, $m \mid a - b$.*

Definição 2.2.2. *Denotaremos a classe de equivalência de a módulo m por $\bar{a} = \{r \in \mathbb{Z} \mid r \equiv a \pmod{m}\}$ e o conjunto das classes de equivalência módulo m por $\mathbb{Z}_m$. Assim, $\mathbb{Z}_m = \{\bar{0}, \bar{1}, ..., \bar{m-1}\}$.*

Definição 2.2.3. *Um conjunto $S = \{n_0, n_1, ..., n_{m-1}\} \subset \mathbb{Z}$ é um sistema completo de resíduos (SCR) módulo m se, e somente se,*
(i) para $i \neq j$ temos $n_i \not\equiv n_j \pmod{m}$,
(ii) dado $a \in \mathbb{Z}$ existe $n \in S$ tal que $a \equiv n \pmod{m}$.

Definição 2.2.4. *Seja S um sistema completo de resíduos módulo m. O subconjunto $S' = \{a \mid a \in S \text{ e } (a,m) = 1\}$ de S chama-se um sistema reduzido de resíduos (SRR) módulo m.*

Teorema 2.2.1 (Teorema Chinês dos Restos)**.** *Sejam $n_1, n_2, n_3, ..., n_k$ números inteiros tais que $(n_i, n_j) = 1$, para $i \neq j$. O sistema de congruências*

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ x &\equiv a_2 \pmod{n_2} \\ x &\equiv a_3 \pmod{n_3} \\ &\ldots \\ x &\equiv a_k \pmod{n_k} \end{aligned}$$

admite uma solução simultânea, que é única módulo o inteiro $n = n_1 n_2 n_3 ... n_k$.

Prova: Seja $n = n_1 n_2 n_3 ... n_k$. Para cada $r = 1, 2, 3, ..., k$, seja $N_r = \frac{n}{n_r} = n_1 n_2 ... n_{r-1} n_{r+1} ... n_k$, ou seja, N_r é o produto de todos os n_i, exceto o n_r. Como $(n_i, n_j) = 1$, a congruência $N_r x \equiv 1 \pmod{n_r}$ admite uma única solução, que chamaremos de x_r, pois $(N_r, n_r) = 1$, e divide 1. A solução do sistema será:

$\bar{x} = a_1 N_1 x_1 + a_2 N_2 x_2 + a_3 N_3 x_3 + ... + a_k N_k x_k$

De fato, basta observar que:

(a) $N_i \equiv 0 \pmod{n_r}$, para $i \neq r$, pois n_r divide N_i;

(b) $\bar{x} = a_1 N_1 x_1 + a_2 N_2 x_2 + a_3 N_3 x_3 + ... + a_k N_k x_k \equiv a_r N_r x_r \pmod{n_r}$;

(c) Como escolhemos x_r, satisfazendo $N_r x \equiv 1 \pmod{n_r}$, temos necessariamente, $\bar{x} \equiv a_r . 1 \equiv a_r \pmod{n_r}$.

Resta-nos mostrar que a solução é única módulo $n = n_1 n_2 n_3 ... n_k$. Suponhamos que exista uma outra solução y, isto é, $\bar{x} \equiv a \pmod{n_r} \equiv y$, para $r = 1, 2, 3, ..., k$. Assim, n_r divide $\bar{x} - y$, para cada valor de r. Como $(n_i, n_j) = 1$, temos que $n = n_1 n_2 n_3 ... n_k$ divide $\bar{x} - y$. Portanto, $\bar{x} \equiv y \pmod{n}$. □

Proposição 2.2.2. *Se A é um subconjunto dos inteiros fechado para adição e subtração, então A é igual a $d\mathbb{Z}$, o conjunto dos múltiplos de d, para algum inteiro d.*

Prova: Se $A = \{0\}$ então $d = 0$ e está provado. Caso contrário, seja d o valor absoluto do menor elemento não nulo de A. O conjunto A contém todos os múltiplos de d, uma vez que contém $\{\pm d\}$ e é fechado para adição e subtração. Além disso, A não pode conter nenhum elemento x que não seja divisível por d, pois poderiamos subtrair o múltiplo mais próximo de d para obter um elemento diferente de zero com valor absoluto menor que d. □

Lema 2.2.1. *Sejam a e b números inteiros, então o conjunto* $a\mathbb{Z}+b\mathbb{Z}=\{ax+by \mid x,y\in\mathbb{Z}\}$ *é igual a* $d\mathbb{Z}$ *onde* $d=(a,b)$.

Prova: O conjunto $a\mathbb{Z}+b\mathbb{Z}$ é fechado para adição e subtração, então $a\mathbb{Z}+b\mathbb{Z}=c\mathbb{Z}$ para algum inteiro c. Se $d=(a,b)$ então todo elemento de $a\mathbb{Z}+b\mathbb{Z}$ é divisível por d, logo, $d \mid c$. Por outro lado, a e b são elementos de $c\mathbb{Z}$, ou seja, são divisíveis por c. Isso significa que c é um divisor comum de a e b, donde $c \mid d$. Portanto, $c=d$. □

Lema 2.2.2. *Se* $(a,n)=1$ *então existe um inteiro* a^{-1} *tal que* $a.a^{-1}\equiv 1 \pmod{n}$.

Prova: Pelo Lema 2.2.1, o conjunto $a\mathbb{Z}+n\mathbb{Z}$ é igual a $\mathbb{Z}$, o conjunto dos números inteiros. Em particular, isso significa que existem inteiros x e y tal que $ax+ny=1$. Assim, $a.x\equiv 1 \pmod{n}$, como desejado. □

Lema 2.2.3. *Sejam b, c e n inteiros positivos tal que* $(c,n)=1$ *e a congruência* $x^b\equiv c \pmod{n}$ *tem* $k>0$ *soluções, então a congruência* $x^b\equiv 1 \pmod{n}$ *também tem k soluções.*

Prova: Seja x_0 uma solução de $x^b\equiv c \pmod{n}$. Devemos ter $(x_0,n)=1$, uma vez que $({x_0}^b,n)=(c,n)$ podendo ser maior que 1, contrariando nossa hipótese. De acordo com o Lema 2.2.2 existe um número ${x_0}^{-1}$ tal que $x_0.{x_0}^{-1}\equiv 1 \pmod{n}$. Assim, uma correspondência entre o conjunto solução de $x^b\equiv c \pmod{n}$ e $x^b\equiv 1 \pmod{n}$ é dada pelo mapeamento $y\mapsto y.{x_0}^{-1}$. □

2.3 Teoremas de Fermat e Euler

Lema 2.3.1. *Seja p um número primo, então os números* $\binom{p}{k}$, *onde* $0<k<p$, *são todos divisíveis por p.*

Prova: Para $k=1$, temos $\binom{p}{1}=p$. Podemos, então, supor $1<k<p$, assim, $k! \mid p(p-1)...(p-k+1)$. Como $(k!,p)=1$, decorre que $k! \mid (p-1)...(p-k+1)$, e o resultado se segue, pois $\binom{p}{k}=p\frac{(p-1)...(p-k+1)}{k!}$. □

Teorema 2.3.1 (Pequeno Teorema de Fermat)**.** *Dado um número primo p, tem-se que p divide o número* a^p-a, *para todo* $a\in\mathbb{N}$.

Prova: Vamos provar por indução sobre a. O resultado vale para $a=1$, pois $p \mid 1^p-1=0$. Supondo que é válido para a, iremos prová-lo para $a+1$. Usando o binômio de Newton, temos: $(a+1)^p-(a+1)=a^p-a+\binom{p}{1}a^{p-1}+...+\binom{p}{p-1}a$. Pelo Lema 2.3.1 e a hipótese de indução o segundo membro da igualdade acima é divisível por p. Portanto, demonstra-se o resultado. □

Reescrevendo esse teorema obtemos um método para mostrar que um dado número n não é primo.

Teorema 2.3.2. *Seja n um inteiro positivo. Se existe a inteiro tal que n não divide $a^n - a$ então n não é primo. O número a é dito testemunha da não primalidade de n (testemunha de Fermat).*

Exemplo 2.3.1. *O número $n = 10$ não é primo, pois $2^{10} = 1024 \not\equiv 2 \pmod{10}$.*

Corolário 2.3.1. *Se p é um número primo e se a é um número inteiro não divisível por p, então p divide $a^{p-1} - 1$.*

Prova: Usando o Teorema 2.3.1, temos: $p \mid a^p - a = a(a^{p-1} - 1)$ e $(a, p) = 1$, segue-se, imediatamente, que p divide $a^{p-1} - 1$. □

Definição 2.3.1. *Um número composto n é dito pseudoprimo na base $a > 1$ se $a^{n-1} \equiv 1 \pmod{n}$. Denotaremos o número a por não-testemunha de Fermat de n.*

Exemplo 2.3.2. *O número $n = 341 = 11 \times 31$ é pseudoprimo na base 2, pois $2^{10} \equiv 1 \pmod{11} \Rightarrow 2^{340} \equiv 1 \pmod{11}$ e $2^5 \equiv 1 \pmod{31} \Rightarrow 2^{340} \equiv 1 \pmod{31}$, ou seja, $2^{340} \equiv 1 \mod 341$.*

Proposição 2.3.3. *Seja n um número composto, se existe uma testemunha de Fermat x tal que $(x, n) = 1$, então pelo menos metade dos elementos de $\{1, 2, ..., n-1\}$ são testemunha de Fermat de n.*

Prova: Se $(x, n) = 1$ e x é uma testemunha de Fermat de n, então $x^{x-1} \equiv c \pmod{n}$ para algum $c \neq 1$ satisfazendo $(c, n) = 1$. Assim, pelo Lema 2.2.3 mostramos que existem tantas testemunhas de Fermat quanto não-testemunhas. □

Definição 2.3.2. *Um número composto n é dito número de Carmichael se $a^n \equiv a \pmod{n}$ para todo inteiro a, ou equivalentemente, se n é pseudoprimo na base a para todo a.*

Lema 2.3.2. *Se p é primo, então para algum $k > 0$ o número de $x \in \{1, 2, ..., p-1\}$ satisfazendo $x^k \equiv 1 \pmod{p}$ é no máximo k.*

Prova: Vamos provar, de forma mais geral, que para um polinômio não nulo $P(x) = a_0 + a_1 x + ... + a_k x^k$, o número de $x \in \{1, 2, ..., p-1\}$ satisfazendo $P(x) \equiv 0 \pmod{p}$ é no máximo k. Usando indução em k, para $k = 0$ é trivial. Supondo a tal que $P(a) \equiv 0 \pmod{p}$, ou seja, $P(a) = (x - a)Q(x) + c$, onde $Q(x)$ é um polinômio de grau $k - 1$ com coeficientes inteiros. A congruência $P(a) \equiv 0 \pmod{p}$ implica que c é divisível por p. Se b satisfaz $P(b) \equiv 0 \pmod{p}$

mas $Q(b) \not\equiv 0 \pmod{p}$ então p é um divisor de $(b-a)Q(b)$ mas não é de $Q(b)$, logo, $b \equiv a \pmod{p}$. Segue que cada $b \in \{1,2,...,p-1\}$ satisfazendo $P(b) \equiv 0 \pmod{p}$ satisfaz $b = a$ ou $Q(b) \equiv 0 \pmod{p}$. Pela hipótese de indução, no máximo $k-1$ elementos de $\{1,2,...,p-1\}$ satisfazem a segunda congruência. □

Lema 2.3.3. *Se n é um número de Carmichael, então n tem, no mínimo, três fatores primos distintos e não é divisível pelo quadrado de nenhum primo.*

Prova: Supondo p primo e $p^2 \mid n$, escrevendo $n = p^k q$ onde $k > 1$ e $p \nmid q$. Usando o Teorema Chinês dos Restos, podemos encontrar um x tal que $x \equiv p+1 \pmod{p^k}$ e $x \equiv 1 \pmod{q}$. Como x não tem fator comum com p^k ou q, e $n = p^k q$ temos que $(x,n) = 1$. Para provar que $x^{n-1} \not\equiv 1 \pmod{n}$, basta mostrar que $x^{n-1} \not\equiv 1 \pmod{p^2}$ ou, equivalentemente, $x^n \not\equiv x \pmod{p^2}$. A fórmula $(p+1)^p = \sum_{k=0}^{p} \binom{p}{k} p^k$ implica $(p+1)^p \equiv 1 \pmod{p^2}$, donde temos $x^n \equiv 1 \pmod{p^2}$, ou seja, $x^n \not\equiv x \pmod{p^2}$. Portanto, x é uma testemunha de Fermat.
Se $n = pq$ com $p < q$, sem perda de generalidade, pelo Lema 2.3.2 temos $x \in \{1,2,...,q-1\}$ tal que a congruência $x^{p-1} \not\equiv 1 \pmod{q}$. Daí, $x^{n-1} = x^{pq-1} = x^{pq-1}x^{p-1} \equiv x^{p-1} \not\equiv 1 \pmod{q}$. Como q é um divisor de n, temos que $x^{n-1} \not\equiv 1 \pmod{n}$, ou seja, x é uma testemunha de Fermat, então n não é um número de Carmichael. □

Definição 2.3.3. *Seja $\varphi(m)$ o número de elementos de um sistema reduzido de resíduos módulo m, $m \geq 2$, que corresponde à quantidade de números naturais entre 0 e $m-1$ que são primos com m. Fazendo $\varphi(1) = 1$, está definida uma importante função $\varphi : \mathbb{N} \to \mathbb{N}$, conhecida como função Fi de Euler.*

Proposição 2.3.4. *Para todo $m \geq 2$ temos que $\varphi(m) \leq m-1$, onde $\varphi(m) = m-1$ se, e somente se, m é um número primo.*

Prova: De fato, se m é primo então o sistema reduzido módulo m é formado por $1,2,3,...,m-1 \Leftrightarrow \varphi(m) = m-1$. □

Teorema 2.3.5 (Teorema Euler-Fermat)**.** *Sejam $m,a \in \mathbb{Z}$, com $m > 1$ e $(a,m) = 1$ então $a^{\varphi(m)} \equiv 1 \pmod{m}$.*

Prova: Sejam $r_1, r_2, r_3, ..., r_{\varphi(m)}$ um sistema reduzido de resíduos módulo m, ou seja, $(r_i, m) = 1$ para todo i. Como $(a,m) = 1$ então $ar_1, ar_2, ar_3, ..., ar_{\varphi(m)}$ também forma um sistema reduzido módulo m. Assim, temos:

$a^{\varphi(m)} r_1.r_2.r_3...r_{\varphi(m)} \equiv ar_1.ar_2.ar_3...ar_{\varphi(m)} \equiv r_1.r_2.r_3...r_{\varphi(m)} \pmod{m}$. Logo, $a^{\varphi(m)} \equiv 1 \pmod{m}$. □

Definição 2.3.4. *Sejam $m, a \in \mathbb{N}^*$, com $m > 1$ e $(a,m) = 1$, a ordem k de a em relação a m é definida com sendo: $ord_m(a) = min\{k \in \mathbb{N}^* \mid a^k \equiv 1 \pmod{m}\}$*

Proposição 2.3.6. *Sejam $a, m \in \mathbb{N}$ e $(a,m) = 1$. Temos que $ord_m(a) \mid \varphi(m)$.*

Prova: Supondo que $ord_m(a) \nmid \varphi(m)$ então, pela divisão euclidiana, temos que $\varphi(m) = ord_m(a)q + r$, com $0 < r < ord_m(a)$. Assim, usando o Teorema 2.3.5, temos:
$1 \equiv a^{\varphi(m)} \equiv a^{ord_m(a)q+r} \equiv (a^{ord_m(a)})^q a^r \equiv a^r \pmod{m}$, o que é um absurdo, pois a $ord_m(a)$ é o menor expoente não nulo k tal que $a^k \equiv 1 \pmod{m}$. □

2.4 Resíduos Quadráticos

Definição 2.4.1. *Sejam $a \in \mathbb{Z}$, p primo e $p \nmid a$. Se a congruência $X^2 \equiv a \pmod{m}$ tem solução então dizemos que a é um resíduo quadrático módulo p, caso contrário, diremos que a é um resíduo não quadrático módulo p.*

Proposição 2.4.1. *Seja $p > 2$ um primo tal que $p \nmid a$. Então $p \mid a^{\frac{p-1}{2}} - 1$ ou $p \mid a^{\frac{p-1}{2}} + 1$, não ocorrendo as duas situações simultaneamente.*

Demonstração. Usando o Pequeno Teorema de Fermat temos:
(i) $(a^{\frac{p-1}{2}} - 1)(a^{\frac{p-1}{2}} + 1) = a^{p-1} - 1 \equiv 0 \pmod{m}$, ou seja, uma das situações ocorre.
(ii) $(a^{\frac{p-1}{2}} - 1) + (a^{\frac{p-1}{2}} + 1) = 2a^{\frac{p-1}{2}} = 2(a^{p-1})^{\frac{1}{2}} \equiv 2 \pmod{p}$, logo, não são simultaneas. □

Lema 2.4.1. *Sejam $a, p \in \mathbb{Z}$ em que $p > 2$ é primo e $p \nmid a$. Se $x_0 \in R^* = \{1, 2, ..., p-1\}$ é uma solução da congruência $X^2 \equiv a \pmod{p}$, então $p - x_0$ também é uma solução, não congruente com x_0.*

Prova: Supondo x_0 uma solução, ou seja, $x_0^2 \equiv a \pmod{p}$, temos: $(p - x_0)^2 = p^2 - 2px_0 + x_0^2 \equiv x_0^2 \equiv a \pmod{p}$. Além disso, se $x_1 \in R^*$ é tal que $x_1^2 \equiv a \pmod{p}$, então $x_0^2 \equiv x_1^2 \pmod{p} \Rightarrow p \mid x_0^2 - x_1^2 \Rightarrow p \mid x_0 - x_1$ ou $p \mid x_0 + x_1 \Rightarrow x_1 = x_0$ ou $x_1 = p - x_0$, pois $1 \leq x_0, x_1 \leq p - 1$.
Finalmente, se x_0 fosse congruente a x_1 teríamos:
$x_0 \equiv p - x_0 \pmod{p} \Rightarrow p \mid 2x_0 \Rightarrow p \mid x_0 \Rightarrow p \mid x_0^2 \Rightarrow p \mid a$, o que é um absurdo. □

Teorema 2.4.2 (Critério de Euler)**.** *Sejam $a, p \in \mathbb{Z}$ em que $p > 2$ é primo e $p \nmid a$. Então temos:*
(i) $p \mid a^{\frac{p-1}{2}} - 1 \Leftrightarrow a$ é um resíduo quadrático módulo p;
(ii) $p \mid a^{\frac{p-1}{2}} + 1 \Leftrightarrow a$ não é um resíduo quadrático módulo p.

Prova: Seja R^* um conjunto reduzido de resíduos módulo p. Se $r \in R^*$, temos que a congruência $rX \equiv a \pmod{p}$ possui uma única solução $s \in R^*$.
Supondo que a não é um resíduo quadrático módulo p, então a congruência $X^2 \equiv a \pmod{p}$ não tem solução. Assim, $s \neq r$. Agrupando aos pares os elementos de R^* e usando o Teorema de Wilson (demonstrado no capítulo 4) temos: $-1 \equiv (p-1)! \equiv a^{\frac{p-1}{2}} \pmod{p} \Rightarrow p \mid a^{\frac{p-1}{2}} + 1$.
Agora, suponhamos que a é um resíduo quadrático módulo p, então a congruência $X^2 \equiv a \pmod{p}$ tem uma solução e pelo Lema 2.4.1, temos duas soluções r e s, tais que $s = p - r$. Como $r^2 \equiv a \pmod{p}$, então $rs = r(p-r) = rp - r^2 \equiv -a \pmod{p}$. Agrupando os outros elementos de R^* aos pares, x e y tais que $xy \equiv a \pmod{p}$, temos:
$-1 \equiv (p-1)! \equiv -aa^{\frac{p-3}{2}} \equiv -a^{\frac{p-1}{2}} \pmod{p} \Rightarrow 1 \equiv a^{\frac{p-1}{2}} \pmod{p} \Rightarrow p \mid a^{\frac{p-1}{2}} - 1$.
Para concluir a demonstração vamos usar a Proposição 2.4.1. Assim, temos:
Se $p \mid a^{\frac{p-1}{2}} - 1 \Rightarrow p \nmid a^{\frac{p-1}{2}} + 1 \Rightarrow a$ tem que ser um resíduo quadrático módulo p.
Se $p \mid a^{\frac{p-1}{2}} + 1 \Rightarrow p \nmid a^{\frac{p-1}{2}} - 1 \Rightarrow a$ não pode ser um resíduo quadrático módulo p. □

Definição 2.4.2. *Seja $p > 2$ um número primo. Dado um número inteiro a define-se o Símbolo de Legendre, denotado por $\left(\frac{a}{p}\right)$, como sendo*

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{se } a \text{ é um resíduo quadrático módulo } p \\ 0 & \text{se } p \mid a \\ -1 & \text{se } a \text{ é um resíduo não quadrático módulo } p \end{cases}$$

Proposição 2.4.3. *Sejam $a, b, p \in \mathbb{N}$, com p primo ímpar e $(a,p) = (b,p) = 1$. Tem-se que:*
(i) $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}$
(ii) Se $a \equiv b \pmod{p}$, então $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$
(iii) $\left(\frac{a.b}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$

Prova: Observe que a propriedade (i) é o Critério de Euler, Teorema 2.4.2, utilizando-se o Símbolo de Legendre. Donde, temos:
(a) $\left(\frac{a}{p}\right) = 1 \Leftrightarrow p \mid a^{\frac{p-1}{2}} - 1 \Leftrightarrow 1 \equiv a^{\frac{p-1}{2}} \pmod{p} \Leftrightarrow \left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}$
(b) $\left(\frac{a}{p}\right) = 0 \Leftrightarrow p \mid a \Leftrightarrow p \mid a^{\frac{p-1}{2}} \Leftrightarrow 0 \equiv a^{\frac{p-1}{2}} \pmod{p} \Leftrightarrow \left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}$
(c) $\left(\frac{a}{p}\right) = -1 \Leftrightarrow p \mid a^{\frac{p-1}{2}} + 1 \Leftrightarrow -1 \equiv a^{\frac{p-1}{2}} \pmod{p} \Leftrightarrow \left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}$
A propriedade (ii) é uma consequência imediata do item anterior. Assim,

$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \equiv b^{\frac{p-1}{2}} \equiv \left(\frac{b}{p}\right) \pmod{p}$

Finalmente, para o item (iii), como $(ab)^{\frac{p-1}{2}} = a^{\frac{p-1}{2}}.b^{\frac{p-1}{2}}$, $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}$ e $\left(\frac{b}{p}\right) \equiv b^{\frac{p-1}{2}} \pmod{p}$ concluímos que

$\left(\frac{ab}{p}\right) \equiv (ab)^{\frac{p-1}{2}} = a^{\frac{p-1}{2}}.b^{\frac{p-1}{2}} \equiv \left(\frac{a}{p}\right)\left(\frac{b}{p}\right) \pmod{p}$. □

Definição 2.4.3. *Seja a um inteiro relativamente primo com um número ímpar $n = p_1^{\alpha_1} p_2^{\alpha_2} ... p_k^{\alpha_k}$ o Símbolo de Jacobi, denotado por $\left[\frac{a}{n}\right]$, é definido por:*

$\left[\frac{a}{n}\right] = \left[\frac{a}{p_1^{\alpha_1} p_2^{\alpha_2} ... p_k^{\alpha_k}}\right] = \left(\frac{a}{p_1}\right)^{\alpha_1}\left(\frac{a}{p_2}\right)^{\alpha_2} ... \left(\frac{a}{p_k}\right)^{\alpha_k}$*, onde os símbolos $\left(\frac{a}{p_i}\right)$ são símbolos de Legendre.*

3 ANÁLISE DE ALGORITMOS

Neste capítulo, apresentaremos os conceitos, exemplos e métodos utilizados para o estudo da complexidade de algoritmos.

3.1 Função de Complexidade

Segundo Cormen (2002), um algoritmo é uma sequencia de passos computacionais bem definidos que toma algum valor ou conjunto de valores como entrada e transforma em algum valor de saída (resultado).

Para comparar a eficiência dos algoritmos é usada uma medida chamada de complexidade computacional, que indica o custo ao se executar um algoritmo, sendo esse custo formado, basicamente, por consumo de memória (espaço utilizado) e tempo (duração de sua execução).

No presente estudo, desejamos medir o tempo de execução, por isso adotaremos um modelo matemático no qual as instruções são executadas uma após a outra. O tempo de execução de um algoritmo será representado por uma função f, onde $f(n)$ representa o tempo necessário para executar um algoritmo para uma entrada n, que será dado pelo número de operações executadas.

Objetivando simplificar o processo de contagem do número de instruções do algoritmo e facilitar a determinação da lei de formação da função f, as operações de atribuição $(\leftarrow)$ e retorno (retorna) não serão consideradas.

Exemplo 3.1.1. *O código abaixo identifica o menor elemento de uma lista com n elementos, no qual o trecho (4-6) é executado* $n-1$ *vezes, portanto, temos que* $f(n) = n-1$.

Algoritmo 3.1.1: Identifica o menor valor em uma lista de *n* elementos

```
1 início
2 |  menor ← lista[1];
3 |  para i de 2 até n faça
4 |  |  se (lista[i] < menor) então
5 |  |  |  menor ← lista[i];
6 |  |  fim
7 |  fim
8 |  retorna menor;
9 fim
```

No exemplo anterior, o tempo de execução é uniforme para qualquer tamanho de entrada *n*, mas existem algoritmos que dependem de outros fatores, como os de ordenação, nos quais o tempo gasto será menor se a lista de elementos estiver quase ordenada. Por isso, a função de complexidade pode ser obtida em três situações: pior caso, melhor caso e caso médio.

Exemplo 3.1.2. *Fazer uma busca sequencial por um determinado elemento em uma lista de n elementos. Portanto, a função $f(n)$, considerando os três casos, é dada por:*
- pior caso: $f(n) = n$, quando o elemento procurado é o último a ser consultado ou não está na lista;
- melhor caso: $f(n) = 1$, o elemento é o primeiro da lista;
- caso médio: $f(n) = \frac{n+1}{2}$, considera a probabilidade de $\frac{1}{n}$ para encontrar cada elemento.

Na análise de um algoritmo, devemos verificar o comportamento do mesmo no pior caso e para uma entrada *n* indefinidamente grande, pois, para valores de *n* pequenos, qualquer algoritmo gastará pouco tempo.

Na maioria dos casos, determinar a função $f(n)$ não é uma tarefa fácil, sendo assim, buscamos mostrar que a complexidade do algoritmo é limitada a uma função conhecida, categorizando-o em classes de complexidade (notação assintótica).

3.2 Notação Assintótica

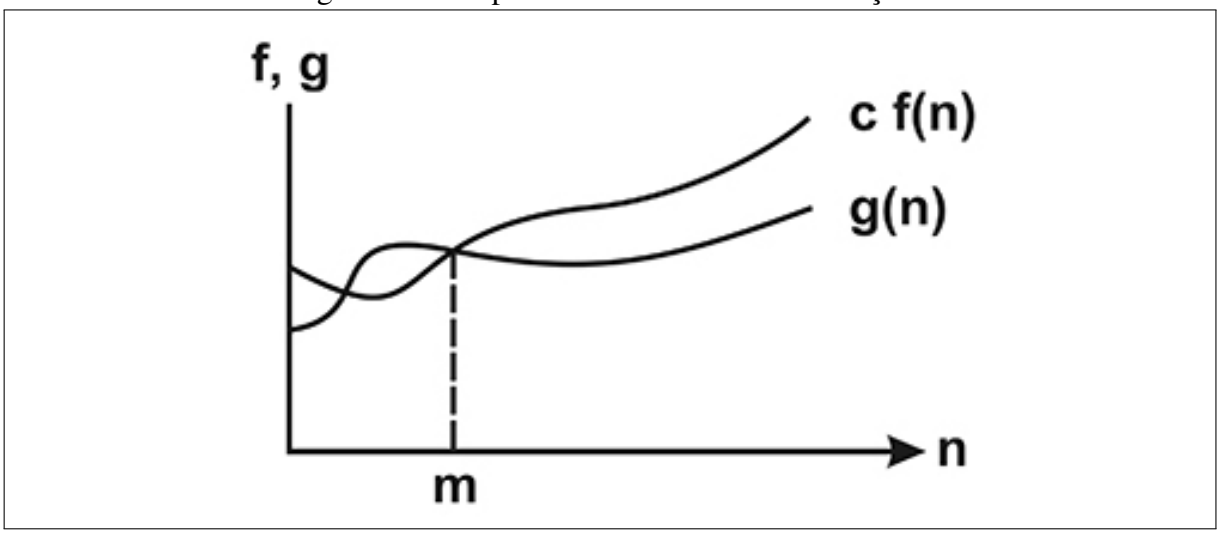
Definição 3.2.1. *Uma função $f(n)$ domina assintoticamente uma função $g(n)$ se existem duas constantes c e m tais que, para $n \geq m$, temos $|g(n)| \leq c.|f(n)|$, que pode ser representada por $g(n)$ é $O(f(n))$ ou $g(n) = O(f(n))$.*

Exemplo 3.2.1. *Mostre que $f(n) = 2n + 10$ é $O(n)$.*
Podemos realizar uma manipulação para encontrar c e m:
$2n + 10 \leq c.n \Rightarrow c.n - 2n \geq 10 \Rightarrow (c-2)n \geq 10 \Rightarrow n \geq \frac{10}{c-2}$
Assim, para que $f(n) \leq c.n$, tomando $c = 4$ temos que $m = 5$.

Exemplo 3.2.2. *Seja $f(n) = n^2 + 2n + 1$, quando $c = 3$ e $m = 2$, temos que: $f(n) \leq 3n^2$, para $n \geq m = 2$, ou seja, $f(n) = O(n^2)$.*

Exemplo 3.2.3. *Para mostrar que $f(n) = 2^{n+2}$ é $O(2^n)$, basta tomar $c = 4$ para qualquer m, pois $2^{n+2} = 2^2.2^n = 4.2^n$.*

Figura 1 – Comportamento assintótico de funções



Fonte – Ziviani (1999)

De acordo com Ziviani (1999), os problemas possuem as seguintes funções de complexidade, que representam a razão de crescimento do tempo de execução de acordo com o tamanho da entrada n:

- Constante: $f(n) = O(1)$
- Logarítimica: $f(n) = O(\log_2 n)$
- Linear: $f(n) = O(n)$
- Linear-Logarítmica: $f(n) = O(n \log_2 n)$
- Quadrática: $f(n) = O(n^2)$
- Cúbica: $f(n) = O(n^3)$
- Exponencial: $f(n) = O(2^n)$.

O algoritmo para cálculo do fatorial de n tem como função de complexidade $f(n) = 3n$, pois a cada iteração (linhas 4 a 7) são executadas 3 (três) instruções (1 comparação, 1 multiplicação e 1 adição). Assim, usando a notação assintótica, temos que: $f(n) = O(n)$.

Algoritmo 3.2.1: Calcula o fatorial de um número n

```
1 início
2     resultado ← 1;
3     contador ← 1;
4     enquanto contador ≤ n faça
5         resultado ← resultado × contador;
6         contador ← contador + 1;
7     fim
8     retorna resultado;
9 fim
```

Observe no exemplo anterior, que contamos cada operação de multiplicação e adição como apenas uma única instrução, ou seja, consideramos que a complexidade dessas operações é $O(1)$. Porém, para valores de n relativamente grandes, tanto adição quanto multiplicação tem custos computacionais que influenciam no desempenho do algoritmo. Sendo assim, faz-se mister compreendermos a complexidade das operações matemáticas básicas, para que possamos obter uma análise algorítmica contemplando todos os fatores que determinam a eficiência de uma solução.

3.3 Operações Matemáticas

Proposição 3.3.1. *Seja n um número natural com k dígitos quando representado no sistema binário (k bits), então* $k = \lfloor \log_2 n \rfloor + 1$.

Prova: Sejam m e M os valores mínimo e máximo de possíveis para n, respectivamente. Assim, $m = \underbrace{100...0}_{\text{k dígitos}} = 1.2^{k-1} + 0.2^{k-2} + ... + 0.2^1 + 0.2^0 = 2^{k-1}$ e
$m = \underbrace{111...1}_{\text{k dígitos}} = 1.2^{k-1} + 1.2^{k-2} + ... + 1.2^1 + 1.2^0 = \frac{1.(2^k-1)}{2-1} = 2^k - 1.$
Daí, $2^{k-1} \leq n \leq 2^k - 1 \Leftrightarrow 2^{k-1} \leq n < 2^k \Leftrightarrow k-1 \leq \log_2 n < k \Leftrightarrow k-1 = \lfloor \log_2 n \rfloor$.
Logo, $k = \lfloor \log_2 n \rfloor + 1$. □

Proposição 3.3.2. *Dados dois números m e n com k dígitos na base 2, a adição (ou subtração) será realizada em* $O(k) = O(\log_2 n)$.

Prova: Sendo a adição (subtração) obtida operando-se dígito a dígito, teremos que executar k operações. Assim, $f(n) = f(m,n) = k$, ou seja, sua complexidade é $O(\log_2 n)$. □

Proposição 3.3.3. *Dados dois números m e n com k dígitos no sistema binário, a multiplicação será determinada em* $O(k^2)$.

Prova: Sabemos que cada bit de m é multiplicado por cada bit de n sendo realizadas k^2 multiplicações e, que posteriormente são efetuadas $k-1$ somas. Donde temos $f(m,n) = k^2 + k - 1$, que em notação assintótica é $O(k^2)$. □

Na multiplicação de números complexos, Gauss observou que o produto $(a+bi)(c+di)$ podia ser calculado usando 3 multiplicações e 5 adições em vez de 4 multiplicações e 4 adições (adição é mais rápida). Tomando os valores $x = ac$, $y = bd$ e $z = (a+b)(c+d)$ temos

que $(a+bi)(c+di) = (x-y)+(z-x-y)i$. Usando a mesma ideia para números inteiros, o algoritmo Karatsuba-Ofman reduz a complexidade da multiplicação para $O(k^{\log_2 3})$.

Utilizando-se método de Newton-Raphson, para o cálculo do inverso e transformando a divisão em uma multiplicação, verificou-se que a divisão inteira tem a mesma complexidade de tempo que a multiplicação. Da mesma forma, o custo de tempo para computar raiz quadrada (ou módulo) é proporcional ao da multiplicação.

Exemplo 3.3.1. *Sejam a, b e m números naturais, sendo a e m com k-bits e b com w-bits, o código abaixo mostra que* $a^b \pmod m$ *tem complexidade* $O(k^2w)$*, onde a representação binária de b é* $b_{w-1}b_{w-2}...b_1b_0$.

Algoritmo 3.3.1: Determina o valor da potência modular $a^b \pmod m$

```
1  início
2      resultado ← 1;
3      para i de w − 1 até 0 faça
4          resultado ← resultado × resultado (mod m);
5          se (b_i = 1) então
6              resultado ← resultado × a (mod m);
7          fim
8      fim
9      retorna resultado;
10 fim
```

Para cada uma das w iterações (linha 3 a 8) temos 2 produtos e 2 módulos, dando com função de custo $f(a,b,m) = w \times 4k^2$. Assim, assintoticamente, temos $O(k^2w)$.

Exemplo 3.3.2. *Dados a e b números inteiros, podemos calcular* (a,b)*, o máximo divisor comum de a e b, usando o algoritmo de Euclides.*

Algoritmo 3.3.2: Determina o máximo divisor comum de a e b

```
1  início
2      enquanto b ≠ 0 faça
3          r ← a (mod b);
4          a ← b;
5          b ← r;
6      fim
7      retorna a;
8  fim
```

Analisando o código acima, concluímos que em poucas iterações o valor de b será igual a 0(zero), portanto, a complexidade é determinada pelo módulo. Assim, fazendo $n = \max\{a,b\}$, teremos $O(\log_2^2 n)$ como função de crescimento do algoritmo.

4 TESTES DETERMINÍSTICOS

Neste capítulo apresentamos os principais testes de primalidade determinísticos, ou seja, algoritmos cujo resultado, seja ele primo ou composto, será incontestável.

4.1 Divisões Sucessivas

O mais simples dos testes de primalidade consiste em dividir um número n por todos os números inteiros que estiverem na faixa que vai de 2 até $n-1$. Se n for divisível por qualquer um deles, então n é um número composto. Caso contrário, é um número primo.

Proposição 4.1.1. *Seja n um número natural composto, então n tem um divisor primo p tal que* $p \leq \sqrt{n}$.

Prova: Seja p o menor divisor primo de n, então $n = pk$ para algum $k \in \mathbb{N}$. Como $p \leq k$ temos que $p^2 \leq pk \Rightarrow P^2 \leq n \Rightarrow p \leq \sqrt{n}$. □

Proposição 4.1.2. *Se p é um número primo diferente de 2 e 3, então p é da forma* $6k-1$ *ou* $6k+1$*, onde k é um inteiro positivo.*

Prova: Todo número ao ser dividido por 6 é de uma das formas: $6k$, $6k+1$, $6k+2$, $6k+3$, $6k+4$ ou $6k+5$.

- $n = 6k$, n é múltiplo de 6, não é primo.
- $n = 6k+1$, pode ser primo.
- $n = 6k+2$, $n = 2(3k+1) \Rightarrow n$ é múltiplo de 2, $6k+2$ não é primo.
- $n = 6k+3$, $n = 3(2k+1) \Rightarrow n$ é múltiplo de 3, $6k+3$ não é primo.
- $n = 6k+4$, $n = 2(3k+2) \Rightarrow n$ é múltiplo de 2, $6k+4$ não é primo.
- $n = 6k+5 \Rightarrow n = 6k+6-1 = 6(k+1)-1 = 6k^{'}-1$, pode ser primo.

Portanto, se n for primo ele não pode ser das formas $6k$, $6k+2$, $6k+3$ e $6k+4$. Assim, n é primo para $6k-1$ ou $6k+1$. □

Utilizando a Proposição 4.1.1, Eratóstenes criou um método, chamado Crivo de Eratóstenes, para listar os primos menores que um certo inteiro positivo $n > 1$, que consiste do seguinte:

(a) Escreva uma lista com todos os inteiros entre 2 e $n-1$;

(b) Para cada primo $p \leq \sqrt{n}$, elimina-se da lista todos os múltiplos pk de p, para $k \geq 2$;

(c) Os números que sobrarem são os primos menores que n.

Exemplo 4.1.1. *Determinar os números primos menores de 2 a 100.*

	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~~	~~60~~
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

De acordo com a proposiçoes 4.1.1 e 4.1.2, para verificarmos se um número é primo, usando o método das divisões sucessivas, devemos dividir o número por 2 e por 3 e, depois, faz-se as divisões por todos os inteiros na forma $6k \pm 1$ que sejam menores ou iguais à raiz quadrada do número testado.

Algoritmo 4.1.1: Divisões Sucessivas

```
1  início
2      se (n (mod 2) = 0) ou (n (mod 3) = 0) então
3          retorna COMPOSTO;
4      fim
5      k ← 1;
6      enquanto (6k − 1) × (6k − 1) ≤ n faça
7          se (n (mod 6k − 1) = 0) ou (n (mod 6k + 1) = 0) então
8              retorna COMPOSTO;
9          fim
10         k ← k + 1;
11     fim
12     retorna PRIMO;
13 fim
```

Exemplo 4.1.2. *Verifique se o número 323 é primo ou composto.*

Como $\sqrt{323} \simeq 17,9722$*, então vamos dividir 323 por 2, 3, 5, 7, 11, 13 e 17. Se nenhum destes*

números dividir 323, então ele será primo:

- *Dividindo por 2 e por 3*
  - $323 = 2 \times 161 + 1$
  - $323 = 3 \times 107 + 2$
- *Para* $k = 1$*, dividimos por* $6k-1 = 5$ *e* $6k+1 = 7$
  - $323 = 5 \times 64 + 3$
  - $323 = 7 \times 46 + 1$
- *Para* $k = 2$*, temos* $6k-1 = 11$ *e* $6k+1 = 13$
  - $323 = 11 \times 29 + 4$
  - $323 = 13 \times 24 + 11$
- *Para* $k = 3$*, dividimos por* $6k-1 = 17$ *e* $6k+1 = 19 > \sqrt{323}$
  - $323 = 17 \times 19 + 0$

Portanto, o número 323 não é primo porque é divisível por 17.

Exemplo 4.1.3. *Determine a primalidade do número 181.*

Observe que $\sqrt{181} \simeq 13,453$*, portanto devemos dividir 181 por 2, 3, 5, 7, 11 e 13. Se 181 não for divisível por nenhum destes números, então ele será primo:*

- *Dividindo por 2 e por 3*
  - $181 = 2 \times 90 + 1$
  - $181 = 3 \times 60 + 1$
- *Para* $k = 1$*, dividimos por* $6k-1 = 5$ *e* $6k+1 = 7$
  - $181 = 5 \times 36 + 1$
  - $181 = 7 \times 25 + 6$
- *Para* $k = 2$*, temos* $6k-1 = 11$ *e* $6k+1 = 13$
  - $181 = 11 \times 16 + 5$
  - $181 = 13 \times 13 + 12$

Assim, podemos afirmar que o número 181 é primo.

Neste algoritmo, a complexidade é determinada pelas $\frac{\sqrt{n}}{6}$ iterações (linhas 6 a 11) de custo $O(\log_2^2 n)$ - função módulo - requerendo $O(\sqrt{n}\log_2^2 n)$.

4.2 Teorema de Wilson

Proposição 4.2.1. *Sejam $a, m \in \mathbb{Z}$, com $m > 1$. A congruência $aX \equiv 1 \pmod{m}$ possui solução se, e somente se, $(a,m) = 1$. Além disso, se $x_0 \in \mathbb{Z}$ é uma solução, então x é uma solução da congruência se, e somente se, $x \equiv x_0 \pmod{m}$.*

Prova: Seja x_0 uma solução, então $m \mid ax_0 - 1 \Leftrightarrow aX - mY = 1$, portanto, $(a,m) = 1$.
Por outro lado, se x_0 e x são soluções da congruência $aX \equiv 1 \pmod{m}$, então $ax \equiv ax_0 \equiv 1 \pmod{m}$, donde temos, $x \equiv x_0 \pmod{m}$. □

Lema 4.2.1. *Seja p um número primo. Os únicos elementos do conjunto $R^* = \{1, 2, ..., p-1\}$ que satisfazem a equação $x^2 \equiv 1 \pmod{p}$ são 1 e $p-1$.*

Prova: Tomando $a \in R^*$ de modo que $a^2 \equiv \pmod{p}$, isto é, $p \mid a^2 - 1$. Como $a \in R^*$ e $p \mid a+1$ ou $p \mid a-1$ temos:
$1 \leq a \leq p-1 \Rightarrow 2 \leq a+1 \leq p \Rightarrow p = a+1 \Rightarrow a = p-1$.
Analogamente, considerando que $p \mid a-1$ e $a-1 < p$, para todo $a \in R^*$, concluímos que $a - 1 = 0 \Rightarrow a = 1$. □

Teorema 4.2.2 (Teorema de Wilson)**.** *Um número inteiro $n > 1$ é primo se, e somente se, $(n-1)! \equiv -1 \pmod{n}$.*

Prova: Suponha n primo, usando a Proposição 4.2.1 e o Lema 4.2.1, temos que cada fator de $(n-2)! = 2.3.4...(n-3)(n-2)$ possui seu próprio inverso (módulo n) entre os outros fatores. Assim,
$(n-2)! = 2.3.4...(n-3)(n-2) \equiv 1 \pmod{n} \Leftrightarrow (n-1)! \equiv n-1 \equiv -1 \pmod{n}$.
A recíproca será provada por contradição. Supondo que $(n-1)! \equiv -1 \pmod{n} \Leftrightarrow n \mid (n-1)!+1$ e que n não seja primo, ou seja, $n = rs$ e $1 < r, s < n$. Nestas condições $r \mid (n-1)!+1$ e, sendo r um divisor de n, $r \mid (n-1)!$. Portanto, r divide a diferença $(n-1)!+1-(n-1)! = 1$, absurdo, uma vez que $r > 1$. Assim, n satisfazendo $(n-1)! \equiv -1 \pmod{n}$ deve ser primo. □

Analisando o algoritmo abaixo, concluímos que este teste não é prático, pois são necessárias $n-1$ multiplicações módulo n para calcular $(n-1)!$, fazendo com que sua complexidade seja $O(n \log_2^2 n)$.

Algoritmo 4.2.1: Teorema de Wilson

```
1  início
2      resultado ← 1;
3      numero ← 1;
4      enquanto numero ≤ n − 1 faça
5          resultado ← (resultado × numero) (mod n);
6          numero ← numero + 1;
7      fim
8      se (resultado ≡ −1 (mod n)) então
9          retorna PRIMO;
10     fim
11     retorna COMPOSTO;
12 fim
```

Exemplo 4.2.1. *Verifique se o número 6 é primo ou composto.*

Sendo $(6-1)! = 5! = 5 \times 4 \times \times 3 \times 2 \times 1 = 120 = 6 \times 20 \equiv 0 \pmod 6$.

Portanto, 6 não é um número primo.

Exemplo 4.2.2. *Determine a primalidade do número 11.*

Como $(11-1)! = 10! = 10 \times 9 \times ... \times 2 \times 1$, *agrupando os termos inversos (módulo 11) temos:*

$10! = 1 \times (2 \times 6) \times (3 \times 4) \times (5 \times 9) \times (7 \times 8) \times 10 \equiv -1 \pmod{11}$.

Assim, o número 11 é primo.

4.3 AKS

Em 2002, no artigo chamado "PRIMES is in P", os indianos Manindra Agrawal, Neeraj Kayal e Nitin Saxena apresentaram um teste determinístico, denominado AKS, capaz de verificar a primalidade de um número em tempo polinomial.

O algoritmo AKS é composto pelos seis passos descritos abaixo. Dado um número inteiro $n > 1$, temos:

1. Se $n = a^b$ para $a \in \mathbb{N}$ e $b > 1$, retorna COMPOSTO.

2. Encontre o menor r tal que $ord_r(n) > \log_2^2 n$.

3. Se $1 < (a,n) < n$ para algum $a \leq r$, retorna COMPOSTO.

4. Se $n \leq r$, retorna PRIMO.

5. Para $a = 1$ até $\lfloor \sqrt{\varphi(r)} \log_2 n \rfloor$ faça: Se $(x+a)^n \not\equiv (x^n + a) \pmod{(x^r - 1, n)}$ então retorna COMPOSTO.

6. Retorna PRIMO.

O algoritmo baseia-se na equação de congruência $(x+a)^p \equiv (x^p+a) \pmod{p} \Leftrightarrow p$ é primo, que é uma generalização do Pequeno Teorema de Fermat.

Podemos verificar esta propriedade no Triângulo de Pascal, onde cada linha representa os coeficientes da expansão binomial. Assim, se todos os números da linha, excetuando-se os extremos, forem múltiplos da linha (número da linha) então o número que representa a linha será primo.

Exemplo 4.3.1. *Observe Triângulo de Pascal abaixo que 2, 3, 5 e 7 são números primos.*

```
0:    1
1:    1     1
2:    1     2     1
3:    1     3     3     1
4:    1     4     6     4     1
5:    1     5    10    10     5     1
6:    1     6    15    20    15     6     1
7:    1     7    21    35    35    21     7     1
8:    1     8    28    56    70    56    28     8     1
9:    1     9    36    84   126   126    84    36     9     1
10:   1    10    45   120   210   252   210   120    45    10     1
```

Teorema 4.3.1. *Sejam $a \in \mathbb{Z}$, $n \in \mathbb{N}$, $n \geq 2$ e $(a,n) = 1$. Então n é primo se, e somente se,* $(x+a)^n \equiv (x^n+a) \pmod{n}$

Prova: Para $0 < k < n$, o coeficiente de x^k em $(x+a)^n - (x^n+a)$ é $\binom{n}{k}a^{n-k}$. Supondo que n é primo, pelo Lema 2.3.1, temos $\binom{n}{k} = 0 \pmod{n}$, ou seja, todos os coeficientes são 0 (zero). Por outro lado, suponhamos que n é composto. Seja o primo p um fator de n e p^q a maior potência de p que divide n. Assim, p^q não divide $\binom{n}{p}$ e $(p^q, a^{n-p}) = 1$, logo, o coeficiente de x^p não é 0 (zero). Portanto, $n \nmid (x+a)^n - (x^n+a)$. □

A congruência acima tem um tempo de execução exponencial. No entanto, para um número r razoavelmente pequeno e adequadamente escolhido, podemos utilizar módulo de $x^r - 1$ para reduzir o número de testes necessários, pois teremos que examinar apenas o resto da divisão de $(x+a)^n$ por $x^r - 1$. Dessa forma, a congruência utilizada é a seguinte:

$$(x+a)^n \equiv (x^n+a) \pmod{(x^r-1, n)}$$

onde $\pmod{(x^r-1,n)}$ significa aplicar $\pmod{x^r-1}$ e ao resultado a congruência $\pmod{n}$.

Na análise deste algoritmo omitiremos a demonstração da equivalência acima e dos demais resultados utilizados. Portanto, para uma explanação detalhada, consulte o artigo citado no inicio desta seção (ou COUTINHO, 2004).

Neste trabalho, mostraremos, de forma resumida, os pontos que determinam a complexidade do algoritmo, por isso vamos verificar o custo computacional envolvido em cada de seus passos.

Tabela 4.3.1 – Complexidade das etapas do AKS

Etapa	Descrição	Complexidade
1	Identifica se n é uma potência própria	$O(\log_2^3 n)$
2	Encontra um r tal que $ord_r(n) > \log_2^2 n$	$O(\log_2^7 n)$
3	Calcula o $mdc(a,n)$ para $a \leq r$	$O(\log_2^6 n)$
4	Compara os valores de n e r	$O(\log_2 n)$
5	Verifica $(x+a)^n \equiv (x^n+a) \pmod{(x^r-1,n)}$	$O(\log_2^{21/2} n)$

Fonte – Elaborada pelo autor

Algoritmo 4.3.1: Verifica se um número n é uma potência própria

```
1  início
2      baseFinal ← n;
3      para expoente de 2 até log2 n faça
4          baseInicial ← 2;
5          enquanto baseFinal − baseInicial > 1 faça
6              baseMedia ← (baseInicial + baseFinal)/2;
7              resultado ← potencia(baseMedia, expoente);
8              se (resultado < n) então
9                  baseInicial ← baseMedia;
10             fim
11             senão se (resultado > n) então
12                 baseFinal ← baseMedia;
13             fim
14             senão
15                 retorna VERDADEIRO;
16             fim
17         fim
18         resultado ← potencia(baseMedia, expoente);
19         se (resultado = n) então
20             retorna VERDADEIRO;
21         fim
22     fim
23     retorna FALSO;
24 fim
```

O método acima verifica a existência de potência própria, ou seja, se existem inteiros positivos $a > 1$ e $b > 1$ de modo que $n = a^b$. Observe que $2 \leq b \leq \log_2 n$, pois o valor máximo de b é obtido quando a for mínimo ($a = 2$).

Para encontrar o valor de r que satisfaz o item 2, basta determinar o menor r tal que não exista $k = ord_r(n)$ onde $1 \leq k \leq \log_2^2 n$. Por outro lado, do Teorema 2.3.5 e das Proposições 2.3.4 e 2.3.6 temos que $\log_2^2 n < ord_r(n) \leq \varphi(r) \leq r - 1 < r$. Assim, o valor de r pode ser dado pelo algoritmo abaixo.

Algoritmo 4.3.2: Encontra o menor r tal que $ord_r(n) > \log_2^2 n$

```
1  início
2      r ← log₂² n;
3      encontrado ← FALSO;
4      enquanto encontrado = FALSO faça
5          encontrado ← VERDADEIRO;
6          k ← 1;
7          enquanto k < log₂² n faça
8              resultado ← n^k (mod r);
9              se (resultado = 1) ou resultado = r − 1 então
10                 encontrado ← FALSO;
11             fim
12             k ← k + 1;
13         fim
14         r ← r + 1;
15     fim
16     retorna r;
17 fim
```

Na execução da etapa 3, devemos verificar se existe $a \leq r$ tal que $1 < (a,n) < n$, o que pode ser feito usando o seguinte procedimento:

Algoritmo 4.3.3: Calcula o mdc(a, n) para $a \leq r$

```
1  início
2      para a de 1 até r faça
3          mdc ← (a,n);
4          se (mdc ≠ 1) e (mdc ≠ n) então
5              retorna COMPOSTO;
6          fim
7      fim
8  fim
```

O quarto passo consiste em uma simples comparação entre os valores de r e n, não

sendo necessária a construção de nenhum algoritmo.

O teste da congruência consiste na verificação de $\lfloor\sqrt{\varphi(r)}\log_2 n\rfloor$ equações, que necessitam de $O(\log_2 n)$ multiplicações de r coeficientes de tamanho $O(\log_2 n)$. Portanto, cada equação requer um custo de processamento de $O(r\log_2^2 n)$, logo, a complexidade desta etapa é dada por: $O(r\sqrt{\varphi(r)}\log_2^3 n) \cong O(r^{\frac{3}{2}}\log_2^3 n) \cong O(\log_2^{21/2} n)$.

Como dito anteriormente, $\log_2^2 n < ord_r(n) \leq \varphi(r) < r$ donde concluímos que $\log_2 n < \sqrt{r}$ e $\varphi(r) < r$. Assim, a expressão $\lfloor\sqrt{\varphi(r)}\log_2 n\rfloor < \sqrt{r}\sqrt{r} = r$ pode ser substituída no algoritmo.

Algoritmo 4.3.4: Verifica a congruência $(x+a)^n \equiv (x^n+a) \pmod{(x^r-1,n)}$

```
1 início
2     para a de 1 até r faça
3         lado1 ← (x+a)^n (mod n);
4         lado2 ← x^n + a (mod n);
5         se (lado1 ≠ lado2) então
6             retorna COMPOSTO;
7         fim
8     fim
9 fim
```

Proposição 4.3.2. *Sejam $n, r \in \mathbb{N}$ então $x^n + a \equiv x^{n \bmod r} + a \pmod{(x^r-1)}$.*

Prova: Inicialmente, mostraremos que $x^r - 1 \mid x^{rq} - 1$. Por indução temos:
Sendo verdade para $q = 1$, vamos provar que vale para $q+1$ supondo que $x^r - 1 \mid x^{rq} - 1$. Assim, $x^{r(q+1)} - 1 = x^{rq}x^r - 1 = x^{rq}x^r - x^r + x^r - 1 = x^r(x^{rq} - 1) + x^r - 1$, logo, $x^r - 1 \mid x^{rq} - 1$.
Tomando $n = rq + l$ onde $l < r$ temos: $x^n + a = x^{rq+l} + a = x^{rq}x^l + a = x^l(x^{rq} - 1) + x^l + a$.
Portanto, $x^n + a \equiv x^l + a \equiv x^{n \bmod r} + a \pmod{(x^r-1)}$. □

Dentre todos os passos, este é o que apresenta o maior custo computacional. Usando a proposição anterior, podemos facilmente calcular $x^n + a$, mas ainda não temos mecanismo semelhante para simplificar o termo $(x+a)^n$.

Exemplo 4.3.2. *Detemine a primalidade do número 341.*
Analisando cada passo do algoritmo AKS temos:
(1) 341 não é uma potência;
(2) Para $r = 77$ temos $ord_{77}(341) > \log_2^2 341$;
(3) Para $a = 11$ temos $1 < (11, 341) = 11 < 341$, portanto, 341 é um número composto.

Exemplo 4.3.3. *Verifique se o número 31 é primo ou composto.*

Executando o teste AKS obtemos:

(1) 31 não é uma potência;

(2) Para $r = 29$ *temos* $ord_{29}(31) > \log_2^2 31$*;*

(3) Não existe $a \leq 29$ *tal que* $1 < (a, 31) < 31$*;*

(4) $31 > 29$*, nada podemos afirmar;*

(5) A congruência é verdadeira para todo $1 \leq a \leq 29$*;*

(6) Logo, o número 31 é primo.

4.4 Lucas-Lehmer

O teste de Lucas-Lehmer é um teste de primalidade, para números de Mersenne, originalmente desenvolvido por Edouard Lucas e posteriormente melhorado por Derrick Henry Lehmer.

Teorema 4.4.1. *Seja* S_k *a sequência definida por* $S_k = (2+\sqrt{3})^{2^k} + (2-\sqrt{3})^{2^k}$ *para* $k \in \mathbb{N}$*. Seja* $n > 2$*,* $M_n = 2^n - 1$ *é primo se, e somente se,* S_{n-2} *é múltiplo de* M_n*.*

Prova: Suponha, por absurdo, que $M_n \mid (2+\sqrt{3})^{2^{n-2}} + (2-\sqrt{3})^{2^{n-2}}$ e M_n seja composto, com um fator primo p tal que $p^2 \leq M_n$. Assim, teremos:

$(2+\sqrt{3})^{2^{n-2}} + (2-\sqrt{3})^{2^{n-2}} \equiv 0 \pmod{p} \Leftrightarrow (2+\sqrt{3})^{2^{n-2}} \equiv -(2-\sqrt{3})^{2^{n-2}} \pmod{p}$

Como $(2+\sqrt{3})(2-\sqrt{3}) = 1$, ou seja, $2-\sqrt{3} = (2+\sqrt{3})^{-1}$ então reescrevendo a equação $(2+\sqrt{3})^{2^{n-2}} \equiv -\frac{1}{(2+\sqrt{3})^{2^{n-2}}} \pmod{p} \Leftrightarrow (2+\sqrt{3})^{2^{n-1}} \equiv -1 \pmod{p}$, o que significa que a ordem de $2+\sqrt{3}$ é igual a 2^n. Absurdo, pois o valor máximo de $\varphi(M_n) = p^2 - 1 < 2^n$, logo, se S_{n-2} é múltiplo de M_n então M_n é primo.

Supondo M_n primo, pelo Teorema 4.3.1 temos:

$(1+\sqrt{3})^{M_n} \equiv 1 + (\sqrt{3})^{M_n} \equiv 1 + 3^{\frac{M_n-1}{2}}\sqrt{3} \equiv 1\left(\frac{3}{M_n}\right)\sqrt{3} \equiv 1 - \sqrt{3} \pmod{M_n}$, já que pela reciprocidade quadrática $\left(\frac{3}{M_n}\right) = -\left(\frac{M_n}{3}\right) = -\left(\frac{-2}{3}\right) = -1$.

Note ainda que $(1+\sqrt{3})^2 = 2(2+\sqrt{3})$. Assim, temos:

$[(1+\sqrt{3})^2]^{2^{n-1}} = 2^{2^{n-1}}(2+\sqrt{3})^{2^{n-1}} \Leftrightarrow (1+\sqrt{3})^{2^n} = 2^{2^{n-1}}(2+\sqrt{3})^{2^{n-1}}$

$(1+\sqrt{3})^{M_n+1} = 2.2^{\frac{M_n-1}{2}}(2+\sqrt{3})^{2^{n-1}} \Leftrightarrow -2 \equiv 2\left(\frac{2}{M_2}\right)(2+\sqrt{3})^{2^{n-1}} \pmod{M_n}$

$-1 \equiv \left(\frac{2}{M_n}\right)(2+\sqrt{3})^{2^{n-1}} \pmod{M_n} \Leftrightarrow -1 \equiv (2+\sqrt{3})^{2^{n-1}} \pmod{M_n}$,

uma vez que $\left(\frac{2}{M_n}\right) = (-1)^{\frac{M_n^2-1}{8}} = 1$.

Daí, usando novamente a igualdade $(2+\sqrt{3})(2-\sqrt{3}) = 1$ concluímos que:

$(2+\sqrt{3})^{2^{n-1}}.(2-\sqrt{3})^{2^{n-2}} \equiv -(2-\sqrt{3})^{2^{n-2}} \pmod{M_n}$
$(2+\sqrt{3})^{2^{n-1}}.\frac{1}{(2+\sqrt{3})^{2^{n-2}}} \equiv -(2-\sqrt{3})^{2^{n-2}} \pmod{M_n}$
$(2+\sqrt{3})^{2^{n-2}} \equiv -(2-\sqrt{3})^{2^{n-2}} \pmod{M_n}$
$(2+\sqrt{3})^{2^{n-2}} + (2-\sqrt{3})^{2^{n-2}} \equiv 0 \pmod{M_n}$
Portanto, $M_n \mid (2+\sqrt{3})^{2^{n-2}} + (2-\sqrt{3})^{2^{n-2}}$, ou seja, S_{n-2} é múltiplo de M_n. □

Proposição 4.4.2. *Seja a sequência $S_k = (2+\sqrt{3})^{2^k} + (2-\sqrt{3})^{2^k}$, S_k de forma recursiva é dada por: $S_0 = 4$ e $S_{k+1} = S_k^2 - 2$.*

Prova: Para $k = 0$, teremos $S_0 = (2+\sqrt{3})^{2^0} + (2-\sqrt{3})^{2^0} \Leftrightarrow S_0 = 2+\sqrt{3}+2-\sqrt{3} \Leftrightarrow S_0 = 4$.
De modo análogo, vamos escrever S_{k+1} em função de S_k
$S_{k+1} = (2+\sqrt{3})^{2^{k+1}} + (2-\sqrt{3})^{2^{k+1}} = (2+\sqrt{3})^{2^k.2} + (2-\sqrt{3})^{2^k.2}$
$= [(2+\sqrt{3})^{2^k}]^2 + [(2-\sqrt{3})^{2^k}]^2 + 2(2+\sqrt{3})^{2^k}(2-\sqrt{3})^{2^k} - 2(2+\sqrt{3})^{2^k}(2-\sqrt{3})^{2^k}$
$= [(2+\sqrt{3})^{2^k} + (2-\sqrt{3})^{2^k}]^2 - 2[(2+\sqrt{3})(2-\sqrt{3})]^{2^k}$
$= [(2+\sqrt{3})^{2^k} + (2-\sqrt{3})^{2^k}]^2 - 2.1^{2^k} = [(2+\sqrt{3})^{2^k} + (2-\sqrt{3})^{2^k}]^2 - 2 = S_k^2 - 2$ □

Algoritmo 4.4.1: Lucas-Lehmer

```
1  início
2      M ← 2^n − 1;
3      S ← 4;
4      para i de 1 até n − 2 faça
5          S ← (S × S − 2) (mod M);
6      fim
7      se (S = 0) então
8          retorna PRIMO;
9      fim
10     retorna COMPOSTO;
11 fim
```

O código acima mostra que existem duas operações que determinam a sua complexidade: a multiplicação de $S \times S$ e o módulo de M. O cálculo do módulo torna-se mais eficiente utilizando, no sistema binário, a propriedade $x \equiv [x \pmod{2^n}] + \lfloor x/2^n \rfloor \pmod{2^n - 1}$, fazendo o custo de cada repetição (linha 4 a 6) ser $O(n \log_2 n)$. Portanto, como são necessárias $n-2$ iterações teremos uma complexidade $O(n^2 \log_2 n)$ para este algoritmo.

Exemplo 4.4.1. *Verifique se $M_5 = 2^5 - 1 = 31$ é um número primo.*
Tomando $S = 4$ vamos atualizar o valor da expressão $S \leftarrow (S \times S - 2) \pmod{M_5}$ 3 vezes:

- $S \leftarrow (4 \times 4 - 2) \pmod{31} = 14$

- $S \leftarrow (14 \times 14 - 2) \pmod{31} = 8$
- $S \leftarrow (8 \times 8 - 2) \pmod{31} = 0$

Portanto, o número 31 é primo.

Exemplo 4.4.2. *Determine a primalidade do número* $M_{11} = 2^{11} - 1 = 2047$.

Atualizando $11 - 2 = 9$ *vezes a expressão* $S \leftarrow (S \times S - 2) \pmod{M_{11}}$ *temos:*

- $S \leftarrow (4 \times 4 - 2) \pmod{2047} = 14$
- $S \leftarrow (14 \times 14 - 2) \pmod{2047} = 194$
- $S \leftarrow (194 \times 194 - 2) \pmod{2047} = 788$
- $S \leftarrow (788 \times 788 - 2) \pmod{2047} = 701$
- $S \leftarrow (701 \times 701 - 2) \pmod{2047} = 119$
- $S \leftarrow (119 \times 119 - 2) \pmod{2047} = 1877$
- $S \leftarrow (1877 \times 1877 - 2) \pmod{2047} = 240$
- $S \leftarrow (240 \times 240 - 2) \pmod{2047} = 282$
- $S \leftarrow (282 \times 282 - 2) \pmod{2047} = 1736$

Portanto, o número 2047 não é primo, pois o valor final de S é diferente de 0 (zero).

Note que, todo número $M_n = 2^n - 1$ quando representado no sistema binário é da forma $\underbrace{111...11}_{\text{n dígitos}}$. Essa característica juntamente com o baixo custo computacional do teste de Lucas-Lehmer fazem com que os maiores números primos encontrados sejam todos primos de Mersenne.

Atualmente, o maior primo conhecido é $M_{74207281} = 2^{74207281} - 1$, com 22338618 dígitos, descoberto em 2016 por Curtis Cooper, através do GIMPS (Great Internet Mersenne Prime Search), um projeto de computação distibuída para identificar primos de Mersenne.

5 TESTES PROBABILÍSTICOS

Os testes probabilísticos são aqueles que retornam uma resposta com um percentual de confiabilidade e utilizam números gerados aleatoriamente para a sua execução. Neste capítulo iremos analisar os dois principais algoritmos probabilísticos para determinar a primalidade de um número: Solovay-Strassen e Miller-Rabin.

5.1 Solovay-Strassen

Desenvolvido por Robert Martin Solovay e Volker Strassen, este algoritmo permite que dado um número ímpar possamos determinar se o mesmo é um número composto ou provavelmente primo utilizando o Critério de Euler e o Símbolo de Jacobi.

Teorema 5.1.1. *Seja n um número inteiro ímpar composto, existe um inteiro a tal que* $(a,n) = 1$ *e* $a^{\frac{n-1}{2}} \not\equiv \left[\frac{a}{n}\right] \pmod{n}$.

Prova: Suponha que $n = p_1 p_2 ... p_r$ com $r \geq 2$ e os p_i's primos ímpares distintos. Como metade dos números $\pmod{p_1}$ não são resíduos quadráticos, então existe $b \in \mathbb{Z}$ tal que $\left(\frac{b}{p_1}\right) = -1$. Pelo Teorema Chinês do Resto, algum $a \in \mathbb{Z}$ satisfaz $a \equiv b \pmod{p_1}$, donde, $a \equiv 1 \pmod{p_2 ... p_r}$. Assim, a é relativamente primo com p_1 e $p_2 ... p_r$, logo, $(a,n) = 1$. Sendo $\left(\frac{a}{p_1}\right) = \left(\frac{b}{p_1}\right) = -1$ e $\left(\frac{a}{p_i}\right) = \left(\frac{1}{p_1}\right) = 1$ para $i > 1$, $\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)\left(\frac{a}{p_2}\right) ... \left(\frac{a}{p_r}\right) = \left(\frac{a}{p_1}\right) = -1$.
Por absurdo, tomemos $a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \pmod{n}$, ou seja $a^{\frac{n-1}{2}} \equiv -1 \pmod{n}$ que pode ser reduzida para módulo p_2, pois p_2 divide n. Portanto,
$1 = \left(\frac{a}{p_2}\right) \equiv a^{\frac{n-1}{2}} \equiv -1 \pmod{p_2} \Leftrightarrow 1 \equiv -1 \pmod{p_2} \Leftrightarrow 2 \equiv 0 \pmod{p_2}$.
Isto é uma contradição, pois o módulo p_2 é maior que 2.
Agora, supondo que n possui um fator primo p repetido, então $n = p^k m$ sendo $k \geq 2$ e $(p,m) = 1$. Novamente do Teorema Chinês do Resto, existe um $a \in \mathbb{Z}$ satisfazendo a congruência $a \equiv 1 + p \pmod{p^2}$, então $a \equiv 1 \pmod{m}$, ou seja, a não é divisível por p e $(a,m) = 1$, logo, $(a,n) = 1$. Se $a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \pmod{n}$, elevando ao quadrado teremos a congruência $a^{n-1} \equiv 1 \pmod{n}$, que é impossível. Pois reduzindo para módulo p^2 (fator de n) obtemos $a^{n-1} \equiv 1 \pmod{p^2}$. Assim, $a \equiv 1 + p \pmod{p^2} \Leftrightarrow (1+p)^{n-1} \equiv a^{n-1} \pmod{p^2} \Leftrightarrow (1+p)^{n-1} \equiv 1 \pmod{p^2}$.
Usando o binomial $(1+p)^{n-1} \equiv 1 + (n-1)p \pmod{p^2}$ teremos:
$(1+p)^{n-1} \equiv 1 \pmod{p^2} \Leftrightarrow 1 + (n-1)p \equiv 1 \pmod{p^2} \Leftrightarrow (n-1)p \equiv 0 \pmod{p^2}$.
Portanto, $n - 1 \equiv 0 \pmod{p}$, absurdo, pois n é múltiplo de p. □

Vejamos o funcionamento deste método no código abaixo:

Algoritmo 5.1.1: Solovay-Strassen

```
1  início
2      para i de 1 até k faça
3          a ← RANDOMICO({2,3,...,n−1});
4          j ← Jacobi(a,n);
5          se (j = 0) ou (a^((n−1)/2) ≢ j (mod n)) então
6              retorna COMPOSTO;
7          fim
8      fim
9      retorna PRIMO;
10 fim
```

Para calcular o Símbolo de Jacobi, indicado pela função $Jacobi(a,n)$ (linha 4), utilizamos o seguinte algoritmo:

Algoritmo 5.1.2: Calcula o Símbolo de Jacobi

```
1  início
2      j ← 1;
3      enquanto a ≠ 0 faça
4          enquanto a (mod 2) ≡ 0 faça
5              a ← a/2;
6              se (n ≡ 3 (mod 8)) ou (n ≡ 5 (mod 8)) então
7                  j ← −j;
8              fim
9          fim
10         c ← n;
11         n ← a;
12         a ← c;
13         se (a ≡ 3 (mod 4)) ou (n ≡ 3 (mod 4)) então
14             j ← −j;
15         fim
16     fim
17     se (n = 1) então
18         retorna j;
19     fim
20     retorna 0;
21 fim
```

O custo computacional da funcão $Jacobi(a,n)$ é de $O(\log_2^2 n)$ uma vez que o execução só é finalizada quando $a = 0$ (linha 3) e a cada iteração a e n trocam de valor.

Proposição 5.1.2. *Seja $n > 1$ um inteiro ímpar. A quantidade de inteiros a tal que $(a,n) = 1$, $1 \leq a \leq n-1$ e $a^{\frac{n-1}{2}} \equiv \left[\frac{a}{n}\right] \pmod{n}$ será:*

(i) igual a $n-1$, para n primo;

(ii) menor que $\frac{n-1}{2}$, para n composto.

Prova: O item (i) é valido pelo Critério de Euler e o Símbolo de Legendre, conforme a Proposição 2.4.3. Para provar (ii), sabemos que $\left(\frac{a}{n}\right) = \pm 1$, se $(a,n) = 1$ e $\left(\frac{a}{n}\right) = 0$, se $(a,n) > 1$. Assim, para os números de 1 até $n-1$, temos os conjuntos disjuntos:

$A = \{1 \leq a \leq n-1 \mid (a,n) = 1 \text{ e } a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \pmod{n}\}$,

$B = \{1 \leq a \leq n-1 \mid (a,n) = 1 \text{ e } a^{\frac{n-1}{2}} \not\equiv \left(\frac{a}{n}\right) \pmod{n}\}$ e

$C = \{1 \leq a \leq n-1 \mid (a,n) > 1\}$.

O conjunto A não é vazio, pois $1 \in A$, assim como C quando n é composto. Pelo Teorema 5.1.1, também B provavelmente não é vazio. Tomando $a \in A$ e $b \in B$ temos que $(ab,n) = 1$ e portanto, $(ab)^{\frac{n-1}{2}} \equiv a^{\frac{n-1}{2}} b^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) b^{\frac{n-1}{2}} \pmod{n}$. Logo, $ab \pmod{n} \in A$ ou B.

Se $ab \pmod{n} \in A$, $(ab)^{\frac{n-1}{2}} \equiv \left(\frac{ab}{n}\right) \equiv \left(\frac{a}{n}\right)\left(\frac{b}{n}\right) \pmod{n} \Rightarrow \left(\frac{a}{n}\right)\left(\frac{b}{n}\right) \equiv \left(\frac{a}{n}\right) b^{\frac{n-1}{2}} \pmod{n}$. Como para $(a,n) = 1$ temos $\left(\frac{a}{n}\right) = \pm 1$, podemos cancelar $\left(\frac{a}{n}\right)$ em ambos os membros da congruência. Daí, $\left(\frac{b}{n}\right) \equiv b^{\frac{n-1}{2}} \pmod{n}$, ou seja, $b \in A$. Absurso, portanto $ab \pmod{n} \in B$.

Sejam $x,y \in A$, se $xb \equiv yb \pmod{n} \Rightarrow x \equiv y \pmod{n} \Rightarrow x = y$, pois os números em A estão estritamente entre 0 e n. Como o número de elementos de $Ab = |A|$ e $Ab \subset B$ temos que $|A| = |Ab| \leq |B|$. Assim, $n-1 = |A|+|B|+|C| \geq |A|+|A|+1 \Rightarrow n-1 > 2|A| \Rightarrow |A| < \frac{n-1}{2}$. □

Exemplo 5.1.1. *Verificando a proposição acima para $n = 15$.*

Note que, $a = 3,5,6,9,10$ e 12 o resultado será COMPOSTO, pois $\left[\frac{a}{n}\right] = 0$. Para outros valores de a temos:

- $a = 2 \Rightarrow \left[\frac{2}{15}\right] = 1$ *e* $2^7 \equiv 8 \pmod{15}$
- $a = 4 \Rightarrow \left[\frac{4}{15}\right] = 1$ *e* $4^7 \equiv 4 \pmod{15}$
- $a = 7 \Rightarrow \left[\frac{7}{15}\right] = -1$ *e* $7^7 \equiv 13 \pmod{15}$
- $a = 8 \Rightarrow \left[\frac{8}{15}\right] = 1$ *e* $8^7 \equiv 2 \pmod{15}$
- $a = 11 \Rightarrow \left[\frac{11}{15}\right] = -1$ *e* $11^7 \equiv 11 \pmod{15}$
- $a = 13 \Rightarrow \left[\frac{13}{15}\right] = -1$ *e* $13^7 \equiv 7 \pmod{15}$
- $a = 14 \Rightarrow \left[\frac{14}{15}\right] = -1$ *e* $14^7 \equiv 14 \pmod{15}$

Assim, o algoritmo retorna PRIMO apenas para $a = 14$, ou seja, a probabilidade de identificar o número 15 como COMPOSTO é $\frac{12}{13} \simeq 92,3\%$.

O ponto principal do teste é $a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \pmod{n}$ falha para algum a relativamente primo com n. Pela Proposição 5.1.2, temos que a probabilidade de erro é 0% se n é um número primo e menos de 50% se n é um número composto. Assim, quando o algoritmo retorna *COMPOSTO* o número é realmente composto, mas quando tem como resposta *PRIMO* existe a possibilidade de ser composto.

Dessa forma, para que tenhamos um maior percentual de acerto devemos executar o teste um k número de vezes. Sendo assim, quando o algoritmo retornar *PRIMO* a probabilidade de veracidade será de pelo menos $1 - \frac{1}{2^k}$.

Apesar de ser bastante eficiente este teste sempre fornece uma primalidade relativa, mesmo que a probabilidade seja alta (aproximadamente $99,9\%$ para $k = 10$), ou seja, há casos em que um número composto é declarado primo.

5.2 Miller-Rabin

Desenvolvido por Gary Lee Miller e Michael Oser Rabin utilizando os conceitos de congruência e o Pequeno Teorema de Fermat. Originalmente era um teste determinístico, supondo verdadeira a Hipótese de Riemann Estendida, sendo posteriormente transformado em um algoritmo probabilístico.

Teorema 5.2.1. *Seja n um primo ímpar tal que $n-1 = 2^w m$, onde 2^w é a maior potência de 2 em $n-1$. Dado a um inteiro tal que $(a,n) = 1$, então $a^m \equiv 1 \pmod{n}$ ou existe $r \in \{0,1,2,...,w-1\}$ tal que $a^{2^r m} \equiv -1 \pmod{n}$.*

Prova: Considere a sequência de potências (módulo n): $a^m, a^{2m}, a^{2^2 m}, ..., a^{2^{w-1} m}, a^{2^w m}$. Se n é um número primo, usando o Corolário 2.3.1 temos $a^{2^w m} = a^{n-} \equiv 1 \pmod{n}$, ou seja, pelo menos uma das potências acima é congruente a 1 módulo n.
Seja $s \geq 1$ o menor expoente tal que $a^{2^s m} \equiv 1 \pmod{n}$, temos:
$a^{2^s m} - 1 = (a^{2^{s-1} m} - 1)(a^{2^{s-1} m} + 1) \Rightarrow n \mid a^{2^{s-1} m} - 1$ ou $a^{2^{s-1} m} + 1$
Sendo s o menor expoente tal que $a^{2^s m} - 1$ é divisível n, então $a^{2^{s-1} m} - 1$ não é divisível por n.
Portanto, $n \mid a^{2^{s-1} m} + 1 \Rightarrow a^{2^{s-1} m} \equiv -1 \mod n \Rightarrow a^{2^r m} \equiv -1 \mod n$. □

Proposição 5.2.2. *Seja n um inteiro ímpar composto, o conjunto $\{1,2,...,n-1\}$ tem no máximo $\frac{n-1}{4}$ números que são primos com n, mas não identificam n como composto no Teorema 5.2.1.*

Prova: Sendo $n-1=2^w m$ e particionando $\{1,2,...,n-1\}$ nos conjuntos $X,Y,Z_1,Z_2,...,Z_w$ definidos da seguinte forma:

- $x \in X$, se $(x,n) \neq 1$;
- $x \in Y$, se $x^m \equiv 1 \pmod{n}$;
- $x \in Z_j$, se $x^{2^j m} \equiv 1 \pmod{n}$ e $x^{2^{j-1}m} \not\equiv 1 \pmod{n}$ onde $1 \leq j \leq w$;

Supondo n um número de Carmichael e $n = p_1 p_2 ... p_k$ a fatoração em primos de n, pelo Lema 2.3.3 temos $k \geq 3$ e $p_1 p_2 ... p_k$ são todos distintos. De acordo com o Teorema Chinês dos Restos, escolher um número $x \in \{1,2,...,n-1\} - X$ equivale a escolher $x_i \in \{1,2,...,p_i-1\}$ para cada $i = 1,2,...,k$. O número x pertence a Y se, e somente, $x_i^m \equiv 1 \pmod{p_i}$ para todo i. Sendo A_i o conjunto de todos os inteiros u tal que para $y \in \{1,2,...,p_i-1\}$ temos $y^u \equiv 1 \pmod{p_i}$. Este conjunto é fechado para adição e subtração, consequentemente é igual a $d\mathbb{Z}$ para algum d. Além disso, pelo Pequeno Teorema de Fermat temos que $p_i - 1 \in A_i$ e o Lema 2.3.2 assegura que A_i não contém nenhum número entre 0 e $p_i - 1$, entao $A_i = \{p_i - 1\} \Rightarrow m \notin A_i$. Assim, pelo menos um $y \in \{1,2,...,p_i-1\}$ satisfaz $y^m \not\equiv 1 \pmod{p_i}$ e no máximo metade dos elementos de $\{1,2,...,p_i-1\}$ verifica $y^m \equiv 1 \pmod{p_i}$ (Lema 2.2.3). Quando escolhemos ao acaso, para cada $i = 1,2,...,k$, a probabilidade de cada um destes números x_i satisfazerem $x_i^m \equiv \pmod{p_i}$ é no máximo $\left(\frac{1}{2}\right)^k$, que é menor ou igual a $\frac{1}{8}$, pois $k \leq 3$.

Assim, aprobabilidade de que um aleatório $x \in \{1,2,...,n-1\} - X$ pertença a Y é no máximo $\frac{1}{8}$, logo $|Y| \leq \frac{n-1}{8}$.

Representando $x \in Z_j$ como uma k-tupla de números $x_i \in \{1,2,...,p_i-1\}$ temos $x_i^{2^j m} \equiv 1 \pmod{p_i}$ que implica $x_i^{2^{j-1}m} \equiv \pm 1 \pmod{p_i}$, então podemos associar a cada $x \in Z_j$ uma sequência de k sinais (+/-), onde i-ésimo será + ou - de acordo com $x_i^{2^{j-1}m} \equiv +1$ ou $-1 \pmod{p_i}$. Um elemento $x \in Z_j$ não é testemunha de Miller-Rabin se, e somente se, a sequência de sinais é $(-,-,...,-)$, isto implica que cada uma das congruências $y^{2^{j-1}m} \equiv -1 \pmod{p_i} (1 \leq i \leq k)$ tem uma solução y_i. Seja v_i um elemento de $\{1,2,...,n-1\}$ tal que $v_i \equiv y_i \pmod{p_i}$ e $v_i \equiv \pmod{p_{i'}}$ para todo $i' \neq i$. Se s e s' são duas sequências de sinais cuja diferença é apenas o i-ésimo sinal, então podemos definir $2^k - 1$ possibilidade de sequências que são subconjuntos de Z_j que tem cardinalidade de $|Z_j|/2^k - 1$. Em particular, a sequência $(-,-,...,-)$ tem cardinalidade $|Z_j|/2^k - 1 \leq |Z_j|/7$.

Note que, X não contém não testemunha de Miller-Rabin, $|Y| \leq \frac{n-1}{8}$ e cada conjunto Z_j tem $\frac{|Z_j|}{7}$ não testemunhas. Assim,

$$|Y| + \frac{n-1-|Y|}{7} = \frac{6|Y|}{7} + \frac{n-1}{7} \leq \frac{6\times(n-1)}{7} + \frac{n-1}{7} = \frac{n-1}{4} \quad .$$

□

Algoritmo 5.2.1: Miller-Rabin

```
1  início
2      m ← n − 1;
3      w ← 1;
4      enquanto m (mod 2) ≡ 0 faça
5          m ← m/2;
6          w ← w + 1;
7      fim
8      para i de 1 até k faça
9          a ← RANDOMICO({2,3,...,n − 1});
10         x ← a^m (mod n);
11         se (x ≠ 1) e (x ≠ n − 1) então
12             contador ← 0;
13             para j de 1 até w − 1 faça
14                 x ← x^2 (mod n);
15                 se (x = 1) então
16                     retorna COMPOSTO;
17                 fim
18                 se (x = n − 1) então
19                     contador ← contador + 1;
20                 fim
21             fim
22             se (contador = 0) então
23                 retorna COMPOSTO;
24             fim
25         fim
26     fim
27     retorna PRIMO;
28 fim
```

Exemplo 5.2.1. *Mostre que $n = 1729$ é um número composto.*

- $n - 1 = 1728 = 2^6 \times 27$, *então* $w = 6$ *e* $m = 27$*;*
- *Tomando* $a = 671$ *temos:*

$$671^{27} \equiv 1084 \pmod{1729}$$

$$671^{2.27} \equiv 1084^2 \equiv 1065 \pmod{1729}$$

$$671^{2^2.27} \equiv 1065^2 \equiv 1 \pmod{1729}$$

Assim, n é declarado composto e o teste finaliza.

Exemplo 5.2.2. *Verifique a primalidade de* $n = 104513$.

- $n-1 = 104512 = 2^6 \times 1633$*, então* $w = 6$ *e* $m = 1633$*;*
- *Tomando* $a = 3$ *vamos analisar as congruências* $\pmod{n}$*:*

$$3^{1633} \equiv 88958 \pmod{104513}$$
$$3^{2.1633} \equiv 88958^2 \equiv 10430 \pmod{104513}$$
$$3^{2^2.1633} \equiv 10430^2 \equiv 91380 \pmod{104513}$$
$$3^{2^3.1633} \equiv 91380^2 \equiv 29239 \pmod{104513}$$
$$3^{2^4.1633} \equiv 29239^2 \equiv 2781 \pmod{104513}$$
$$3^{2^5.1633} \equiv 2781^2 \equiv -1 \pmod{104513}$$

Portanto, conclui-se que n é provavelmente primo.

Observe que, se um número passa pelo teste k vezes é provável que seja primo (com pelo menos $1 - \frac{1}{4^k}$ de probabilidade. Caso o algoritmo retorne PRIMO temos uma probabilidade de acerto de 75% na primeira verificação. Aplicando novamente, esse percentual vai para $93,75\%$ chegando a $99,9\%$ na quinta iteração. No entanto, não podemos garantir que o número é realmente primo.

Analisando o tempo de execução deste algoritmo vemos que são necessárias $\log_2 n$ divisões por para encontrar os valores de w e m. Usando a representação binária de n essa divisão pode ser acelerada deslocando-se para a esquerda em vez de realizar cada operação. No entanto, o cálculo do módulo da exponencial (linha 10) tem custo de $O(\log_2^3 n)$, então para k iterações o algoritmo terá complexidade $O(k \log_2^3 n)$.

Por apresentar um custo computacional relativamente baixo e sua probabilidade de acerto aproximar-se de 100% mais rápido que o teste de Solovay-Strassen o método de Miller-Rabin é o mais utilizado para a determinação de números primos.

6 CERTIFICADO DE PRIMALIDADE

Neste capítulo, discutiremos o teste proposto por Édouard Lucas para garantir que um dado número n é realmente primo. Apesar de ser baseado no Pequeno Teorema de Fermat, a primalidade é garantida utilizando-se a fatoração de $n-1$ que, em geral, é um problema de difícil solução. Sendo assim, consideraremos esse mecanismo um "certificado"para números primos, em vez de um teste determinístico.

6.1 Teste de Lucas

Teorema 6.1.1. *Sejam n e a números inteiros com $n > 1$ e $2 \leq a \leq n-1$. Se $a^{n-1} \equiv 1 \pmod{n}$ e $a^{\frac{n-1}{p}} \not\equiv 1 \pmod{n}$ para cada fator p de $n-1$, então n é primo.*

Prova: Da primeira condição temos que $ord_n(a) \mid n-1$, pois $ord_n(a)$ é o menor número k tal que $a^k \equiv 1 \pmod{n}$. Ademais, a segunda condição mostra que $ord_n(a)$ não é divisor próprio de $n-1$, logo, $ord_n(a) = n-1$. Pela Proposição 2.3.6 temos que $ord_n(a) \mid \varphi(n)$, donde $\varphi(n) \geq n-1$ e usando a Proposição 2.3.4, $\varphi(n) < n-1$ quando n é composto. Assim, n é um número primo. □

O Teste de Lucas pode ser descrito pelos seguintes passos:

1. Selecionamos uma base a para realizar o teste;
2. Verificamos se $a^{n-1} \equiv 1 \pmod{n}$ é válida;
3. Analisamos se $a^{\frac{n-1}{p}} \not\equiv 1 \pmod{n}$ para cada fator primo p de $n-1$.

Exemplo 6.1.1. *Demonstre que o número 7 é primo.*

Seguindo os passos do algoritmo temos:

- *Escolhendo a base: $a = 2$;*
- $a^{n-1} \equiv 1 \pmod{n}$*: $2^6 = (2^3)^2 \equiv 1 \pmod{7}$, pois $8 \equiv 1 \pmod{7}$;*
- $a^{\frac{n-1}{p}} \not\equiv 1 \pmod{n}$*: Fatorando $n-1 = 6 = 2 \times 3$. Vamos verificar os 2 fatores primos:*

  - $p = 2$: $2^{\frac{6}{2}} = 2^3 = 8 \equiv 1 \pmod{7}$, a condição falhou.
- *Note que, o objetivo do teste é encontrar uma base que valide os itens do algoritmo. Portanto, vamos escolher uma nova base.*
- *Escolhendo a base: $a = 3$;*
- $a^{n-1} \equiv 1 \pmod{n}$*: $3^6 = (3^3)^2 \equiv 1 \pmod{7}$, pois $27 \equiv -1 \pmod{7}$;*
- *Testando $a^{\frac{n-1}{p}} \not\equiv 1 \pmod{n}$ para cada fator:*

- $p = 2$: $3^{\frac{6}{2}} = 3^3 = 27 \equiv -1 \not\equiv 1 \pmod 7$

- $p = 3$: $3^{\frac{6}{3}} = 3^2 = 9 \equiv 2 \not\equiv 1 \pmod 7$

Assim, podemos afirmar que 7 é um número primo, pois todas etapas do teste foram verificadas.

Exemplo 6.1.2. *Verifique que 41 é um número primo.*

Fatorando $n - 1 = 40 = 2^3 \times 5$ *devemos encontrar* $2 \leq a \leq 40$ *tal que:*

- $a^{40} \equiv 1 \pmod{41}$
- $a^{20} \not\equiv 1 \pmod{41}$
- $a^8 \not\equiv 1 \pmod{41}$

Mas $2^{20} \equiv 1 \pmod{41}$, $3^8 \equiv 1 \pmod{41}$, $4^{20} \equiv 1 \pmod{41}$ *e* $5^{20} \equiv 1 \pmod{41}$. *Assim, uma base possível é* $a = 6$.

Uma grande vantagem do teste em questão é a possibilidade de verificar a validade do caráter primo de um número retornado por um algoritmo.

7 CONCLUSÃO

Para que possamos compreender as vantagens e desvantagens de um teste em relação a outro, vamos realizar uma análise comparativa entre os diversos algoritmos de identificação de números primos apresentados nesta pesquisa. Nesta tarefa, usaremos apenas primos de Mersenne para que sejam aplicados todos os métodos discutidos.

Vamos supor um computador que execute 10^9 operações por segundo (uma operação a cada nanossegundo) para que possamos determinar o tempo dispendido por cada teste.

Ademais, para que tenhamos probabilidades semelhantes nos algoritmos Solovay-Strassen e Miller-Rabin usaremos $k = 10$ e $k = 5$, respectivamente.

Tabela 7.0.1 – Tempo gasto por cada algoritmo no teste de primalidade

ALGORITMO	COMPLEXIDADE	$n = 2^5 - 1 = 31$	$n = 2^{13} - 1 = 8191$	$n = 2^{31} - 1 = 2147483647$
Divisão Sucessiva	$O(\sqrt{n}\log_2^2 n)$	$136,65$	$15294,78$	$44533652,94$
Teorema de Wilson	$O(n\log_2^2 n)$	$760,86$	$1384241,49$	$2063731784677,55$
AKS	$O(\log_2^{21/2} n)$	$19825252,21$	$496985158695,9$	$4563497165980708,95$
Lucas-Lehmer	$O(p^2 \log_2 p)$	$58,04$	$625,37$	$4760,98$
Solovay-Strassen	$O(k\log_2^3 n)$	$1215,96$	$21969,1$	$297909,99$
Miller-Rabin	$O(k\log_2^3 n)$	$607,98$	$10984,55$	$148954,99$

Fonte – Elaborada pelo autor

Pelos valores da tabela acima, podemos observar o bom desempenho dos algoritmos Lucas-Lehmer, usado apenas para números de Mersenne e, Miller-Rabin, para números naturais ímpares, uma vez que 2 (dois) é o único primo par.

Testes de primalidade e outras formas de determinar números primos são essenciais para a Criptografia, que utiliza esses números para o seu correto funcionamento. Portanto, o estudo deste assunto contribui para o desenvolvimento de áreas como a Criptografia e a Teoria dos Números, inclusive da própria Matemática.

REFERÊNCIAS

AGRAWAL, Manindra, KAYAL, Neeraj e SAXENA, Nitin. **Primes is in P**. Disponível em: <https://www.cse.iitk.ac.in/users/manindra/algebra/primality_v6.pdf>. Acesso em: 20 mar. 2017.

ALMEIDA, Felipe Duerno do Couto. **Implementação e análise de algoritmos acerca de números primos para uso em maratonas de programação**. Monografia de Graduação. Brasília: UnB, 2016.

ASCENCIO, Ana Fernanda Gomes. **Estruturas de dados**: algoritmos, análise da complexidade e implementações em JAVA e C/C++. São Paulo: Pearson Prentice Hall, 2010.

CORMEN, Thomas H. **Algoritmos**: teoria e prática. 2ª Edição. Rio de Janeiro: Elsevier, 2002.

COUTINHO, S. C. **Números Inteiros e Criptografia RSA**. Instituto Nacional de Matemática Pura e Aplicada - IMPA, 2009.

COUTINHO, S. C. **Primalidade em Tempo Polinomial**: uma introdução ao algoritmo AKS. Universidade Federal do Rio de Janeiro, 2004.

HEFEZ, Abramo. **Elementos de Aritmética**. Textos Universitários, SBM 2006.

LEMOS, Manuel. **Criptografia, Números Primos e Algoritmos**. Universidade Federal de Pernambuco, 2010.

MOREIRA, Carlos Gustavo e SALDANHA, Nicolau. **Primos de Mersenne** (e outros primos muito grandes). 3ª Edição. Rio de Janeiro: IMPA, 1999

MARTINEZ, Fabio Brochero; et al. **Teoria dos Números**: um passeio com primos e outros números familiares pelo mundo inteiro. 2ª Edição. Rio de Janeiro: IMPA, 2011.

ZIVIANI, Nivio. **Projeto de algoritmos com implementações em Pascal e C**. 4ª Edição. São Paulo: Pioneira, 1999.