https://eventos.utfpr.edu.br//sicite/sicite2019

Crivo para números primos e teste de primalidade baseados em uma matriz de oito colunas

Sieve for prime numbers and primality test based on a eight column matrix

Gabriel Pastori Figueira
gabrielpastorifigueira@gmail.com
Universidade Tecnológica Federal do Paraná, Campo Mourão, Paraná, Brasil

Wellington José Corrêa
wcorrea@utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Campo Mourão, Paraná, Brasil

Fernando Cézar Gonçalves Manso
fmanso@utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Campo Mourão, Paraná, Brasil

Recebido: 19 ago. 2019.
Aprovado: 01 out. 2019.

RESUMO

A definição de um número primo é muito simples: um número é dito primo se for um inteiro maior do que 1 e se possuir como divisores somente o número 1 e ele mesmo. Apesar de ser simples definir, determinar a primalidade de um número grande, ou então gerar uma lista com os números primos dado um alto limite superior, pode se mostrar, mesmo computacionalmente, de difícil execução. Esse trabalho apresentará um crivo para números primos e um teste de primalidade, ambos desenvolvidos com base em uma matriz de oito colunas. O algoritmo desenvolvido para a geração de números primos possui vantagens quando comparado ao crivo de Eratóstenes, enquanto o algoritmo desenvolvido para o teste de primalidade possui vantagens quando comparado a um dos dois métodos levados em consideração.

PALAVRAS-CHAVE: Números primos. Algoritmos. Teoria dos números.

ABSTRACT

The definition of a prime number is very simple: a number is prime if it is an integer greater than 1 and if it has only 1 and itself as a divisor. Although it is simple to define, determining the primality of a large number, or generating a list of prime numbers given a high upper limit, even computationally, can be difficult to do. This paper will present a sieve for prime numbers and a primality test, both developed based on an eight column matrix. The algorithm developed for prime number generation has advantages when compared to the Eratosthenes sieve, and the algorithm developed for primality test has advantages when compared to one of the two methods taken into consideration.

KEYWORDS: Prime numbers. Algorithm. Number theory.

INTRODUÇÃO

Não se sabe com exatidão quando a humanidade teve seu primeiro contato com os misteriosos números primos. Entretanto, sabe-se que há mais de 2000 anos os matemáticos buscam compreender a vastidão desses. A dificuldade em determinar fatores primos de um número grande é tamanha, que alguns algoritmos de criptografia baseados em chave pública usam, por exemplo, o produto entre números primos para criptografar, fazendo com que somente os que tiverem os fatores primos do número resultante consigam ter acesso a mensagem (descriptografar). Existem teorias de que até mesmo algumas

espécies de cigarras utilizam-se do fato de que os números primos não possuem divisores além de 1 e o próprio número, a fim de evitar o encontro com parasitas, e assim, aumentar sua capacidade de reprodução.

Diversos são os algoritmos que permitem gerar uma lista com todos os números primos até um dado valor, também diversos são os que permitem testar a primalidade de um número natural qualquer. Esse trabalho apresentará um teste de primalidade determinístico e um crivo para números primos, ambos desenvolvidos com base em uma matriz de oito colunas.

MATERIAL E MÉTODOS

Os algoritmos para o teste de primalidade e para a geração de números primos foram implementados utilizando a linguagem de programação C++, sem o uso de *multi threads*. Para a aferição dos tempos de execução dos programas desenvolvidos, foi utilizado a biblioteca *ctime*, nativa da linguagem. Para melhor entendimento, foi também desenvolvido, utilizando ferramentas de desenvolvimento web, uma página que recebe o tempo entre eliminação de compostos e o número de linhas da matriz, gerando com isso uma animação da matriz de números primos desenvolvida.

O algoritmo do teste de primalidade por meio da matriz de oito colunas, teve o seu tempo aferido, e foi comparado com dois algoritmos convencionais, também implementados em C++. O algoritmo para a geração de números primos por meio da matriz de oito colunas teve o seu tempo aferido, e foi comparado com o algoritmo do crivo de Eratóstenes.

Todos os testes de desempenho (tempo) foram feitos em mesmas condições, ou seja, com o sistema em estado de inicialização e com mesmas configurações, tanto de software, tanto de hardware.

RESULTADOS E DISCUSSÃO

Excluindo os múltiplos de 2, 3 e 5, é possível montar uma matriz A_{m8}, cuja lei de formação é:

$$a_{mn} = 30 \times m + a_{0n} \quad (1)$$

Onde $1 \leq n \in \mathbb{Z} \leq 8$ e $a_{01} = 7$, $a_{02} = 11$, $a_{03} = 13$, $a_{04} = 17$, $a_{05} = 19$, $a_{06} = 23$, $a_{07} = 29$, $a_{08} = 31$.

Todos os números não primos da matriz A podem ser obtidos por meio de multiplicações de elementos da mesma e, portanto, à partir da matriz A, pode-se encontrar todos os números primos com exceção de 2,3 e 5. Para se obter todos os números primos maiores do que 5 até um dado valor inteiro N, basta gerar a matriz A até a linha que contém o primeiro elemento com valor igual ou maior do que N, e eliminar, i.e, marcar como composto, os resultados dos produtos entre elementos da matriz k_1 e k_2, de forma que $(k_1 \times k_2) \leq N$.

A Figura 1 possui uma imagem com a representação da matriz A (FIGUEIRA, 2019).

Figura 1 – Matriz A.

Fonte: Autor.

É possível, à partir da matriz A, elaborar um teste de primalidade determinístico para um natural N qualquer. Tal teste constitui-se de congruências modulares do tipo:

$$L_i \equiv (30 * x^2 + (c_a + c_b) * x + \zeta_0)\ mod\ (30 * x + c_a) \quad (2)$$

$$L_i \equiv (30 * x^2 + (c_a + c_b) * x + \zeta_0)\ mod\ (30 * x + c_b) \quad (3)$$

Sendo L_i dado por:

$$L_i = \frac{N - L_0}{30} \quad (4)$$

E ζ_0 dado por:

$$\zeta_0 = \frac{c_a \times c_b - L_0}{30} \quad (5)$$

De forma que L_0 pode assumir qualquer valor da linha zero da matriz, e, se L_i não for inteiro, N não está na matriz, e portanto, é composto, a menos que seja 2, 3 ou 5.

As variáveis c_a e c_b representam os elementos da linha 0 e das colunas a e b, respectivamente. Sabe-se que o produto entre elementos das colunas a e b, independente da linha em que esses estejam, sempre irá resultar em números que estão numa mesma coluna da matriz, e portanto, é possível determinar previamente as colunas em que os produtos entre pares de colunas irão se situar. Assim, as colunas, e os pares de colunas que quando multiplicados levam a elas, respectivamente, são:

a) 1° coluna: $(1,8); (2,4); (3,5); (6,7);$
b) 2° coluna: $(1,6); (2,8); (3,4); (5,7);$
c) 3° coluna: $(1,5); (2,6); (3,8); (4,7);$

d) 4° coluna: $(1,2);(3,7);(4,8);(5,6);$
e) 5° coluna: $(1,1);(2,7);(3,3);(4,4);(5,8);(6,6);$
f) 6° coluna: $(1,7);(2,3);(4,5);(6,8);$
g) 7° coluna: $(1,4);(2,5);(3,6);(7,8);$
h) 8° coluna: $(1,3);(2,2);(4,6);(5,5);(7,7);(8,8);$

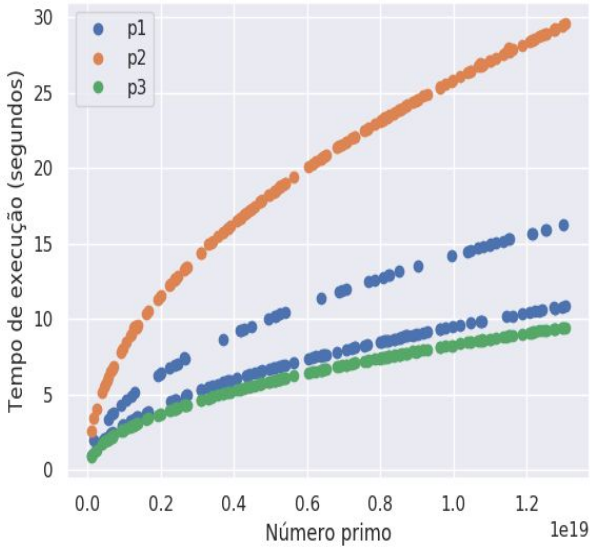
Os testes de congruência devem ocorrer enquanto $L_i \geq (30 * x^2 + (c_a + c_b) * x + \zeta_0)$, com $x \in \mathbb{N}$. Se para algum x, a Eq. (2) ou a Eq. (3) for verdadeira, N é composto, caso contrário, é primo.

Os algoritmos levados em consideração, para a comparação do desempenho do teste de primalidade desenvolvido, são os seguintes:

a) p1: algoritmo baseado na matriz de oito colunas;
b) p2: algoritmo que verifica se um número natural N qualquer é divisível por algum número entre 2 e $\sqrt{N}$ (utilizando iteração igual a uma unidade), se for, N é composto, caso contrário, é primo;
c) p3: algoritmo que usa do fato de que todo número primo pode ser escrito da forma $6 * k \pm 1$ (com exceção de 2 e 3) com $k \in \mathbb{N}^*$. Verifica se um número natural N qualquer é divisível por 2 ou 3, e então, verifica se é divisível por algum número menor que $\sqrt{N}$, da forma $6 * k \pm 1$, se for, N é composto, caso contrário, é primo.

Na Figura 2, há um gráfico com a comparação entre os três algoritmos (p1, p2 e p3) para o teste de primalidade.

Figura 2 – Gráfico contendo os tempos de execução dos três algoritmos para o teste de primalidade



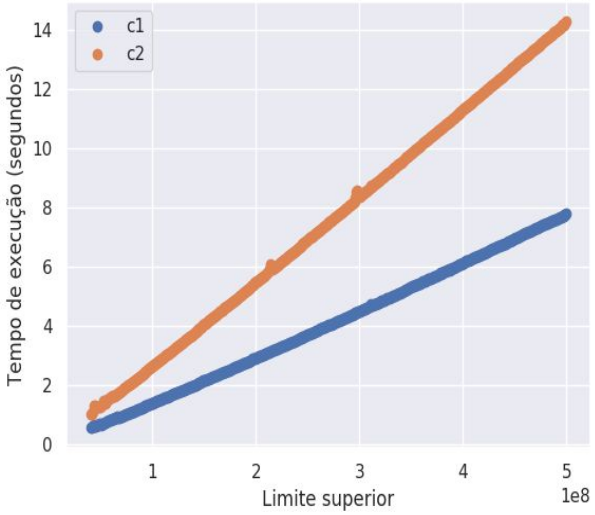
Fonte: Autor.

Já para a geração de números primos até um limite superior N, foram utilizados os seguintes algoritmos:

a) c1: algoritmo baseado na matriz de 8 colunas;
b) c2: crivo de Eratóstenes, nesse se elimina, i.e, marca-se como composto, os múltiplos até N, de todos os primos até $\sqrt{N}$ (WEISSTEN, 2019).

Na Figura 3, há um gráfico com a comparação entre os dois algoritmos (c1 e c2) para geração de números primos.

Figura 3 – Gráfico contendo os tempos de execução dos dois algoritmos para a geração de números primos



Fonte: Autor.

CONCLUSÃO

Observa-se na Figura 2, que o algoritmo desenvolvido no decorrer da pesquisa (p1) possui desempenho similar ao algoritmo p3 para os números que não estão nas colunas 5 e 8, sendo o desempenho consideravelmente pior para os números que estão nessas colunas. Em relação ao algoritmo p2, o algoritmo p1 se mostra em grande vantagem, em todos os casos avaliados.

Observa-se na Figura 3, que o algoritmo desenvolvido no decorrer da pesquisa (c1) possui desempenho consideravelmente superior, quando comparado ao crivo de Eratóstenes (c2).

AGRADECIMENTOS

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico, pela bolsa de estudos e auxílio financeiro. Aos meus orientadores Wellington José Corrêa e Fernando Cézar Gonçalves Manso. Ao meu irmão Victor José Figueira e ao meu amigo Vinícius Guimarães de Oliveira. À Universidade Tecnológica Federal do Paraná, pela estrutura fornecida.

REFERÊNCIAS

Weissten, E. W. Sieve of Eratosthenes. Disponível em: http://mathworld.wolfram.com/SieveofEratosthenes.html. Acesso em: 14 ago. 2019.

Figueira, G. P. Matriz de oito colunas. Disponível em: https://gabrielpastori.github.io/prime-sieve/. Acesso em: 18 ago. 2019.