**Pesquisando Números Primos**

**Mário Leite**

...

Os números primos sempre fascinaram a humanidade, desde a Antiguidade; em particular os matemáticos gregos. Na escola pitagórica do Século 5 a.C esses números eram conhecidos como *asynthetói aritmói*: “***aqueles que não podem ser gerados pelo produto de outros números além da unidade***”. Modernamente, números primos é um assunto da Matemática, na área da Teoria dos Números, e a definição de um número primo é bem concisa e fácil de entender: “n***úmero primo é um número natural que é divisível só por ele mesmo e pela unidade***”. Por outro lado, esta definição pode gerar alguma dúvida, e uma pergunta sempre ocorre nas aulas de Matemáticas do curso médio: “o número 1 é primo”? A resposta é NÃO! O número 1 não é primo; embora ele possa se encaixar, de alguma forma, na definição geral de número primo. Na verdade, o número 1 não é primo por uma simples razão: ele tem um único divisor: ele mesmo. Então, para ser primo um número natural tem que ter dois, e exatamente, dois divisores; nem mais e nem menos. E existem várias técnicas e algoritmos para detectar números primos. A mais simples é a da “Analise da Quantidade de Divisores” do número: se for apenas dois, é primo. Entretanto, uma das técnicas mais rápidas é através da “Radiciação do Número”, que se baseia na comparação dos restos de divisões de todos os menores ou iguais à sua raiz quadrada. Uma terceira opção é o “Método da Peneira” em que se vai testando números para ver se é primo, começando pelo 2, e elimina-se todos os múltiplos de 2; depois testa o 3 e elimina-se todos os múltiplos de 3 (o 4 e todos os pares são eliminados imediatamente por serem múltiplos de 2), e assim sucessivamente, até sobrar apenas os primos.

O maior número primo encontrado (até Dez/2017) tem incríveis 24.862.048 dígitos! Uma foto de parte desse “monstro” é mostrada na **figura 1**; tirada de um *site* mostrado no *link* abaixo.

<https://impa.br/noticias/descoberto-numero-primo-com-quase-25-milhoes-de-digitos/#:~:text=Matem%C3%A1ticos%20%E2%80%94%20profissionais%20e%20amadores%20%E2%80%94%20do,expresso%20como%20282%2C589%2C933%2D1>. (acesso em 16/10/2020 – 13:14).

Eu não tive a coragem de contar os dígitos desse número nessa figura, mas, por uma simples análise visual rápida, a matriz de dígitos não chega nem perto de vinte mil dígitos! Isto quer dizer que a página apresentada não daria para mostrar todos os dígitos desse número; a matriz é muito maior que a mostrada nessa figura. De qualquer forma, podemos concluir que além da curiosidade natural, detectar e mostrar números primos pode ser um bom exercício para testar um conjunto de *software/hardware* em qualquer método ou algoritmo. Para os curiosos que desejam ver este número primo gigante que ocupa um arquivo de 11Mb, podem baixá-lo da Internet. Para se ter uma noção do tamanho desse “bicho”, seus dígitos começam com 4673331833592310999883355 e terminam com 1136582730618069762179071. Isto é, entre os primeiros vinte e cinco dígitos e os últimos vinte e cinco existem 23.249.375 dígitos; uma loucura! E se não acreditar, abra o arquivo M77232917.txt (baixado) e confira!!!!

Desenvolvi dois programas para detectar e mostrar os números primos existentes num intervalo definido pelo usuário; ambos codificados em Python 3.7.1. A preferência por esta linguagem é o fato dela ter uma biblioteca matemática muito poderosa e trabalhar com números do tipo BigInt, podendo suportar números decimais bem grandes; o que não é comum em outras linguagens. O primeiro programa (“MostraPrimo”) é baseado no algoritmo de “Analise da Quantidade de Divisores”, contando a quantidade de divisores de cada número analisado: se ele tiver apenas dois divisores, então é computado como primo; caso contrário é descartado. O segundo programa (“MostraPrimo2”) é baseado no método da “Radiciação do Número”, extraindo a raiz quadrada do número e comparando com restos das divisões de seus antecessores.

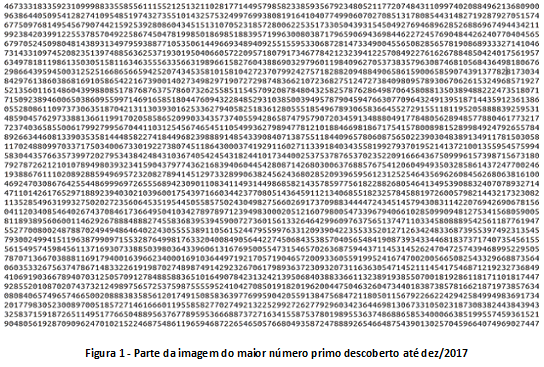
A **figura 2** mostra o código do programa **“MostraPrimo”** e a **figura 3** o código do programa **“MostraPrimo2”**. Na simulação foi feito um teste para detectar os números primos que existem no intervalo de 1 a um milhão, com um computador Intel Dual Core, com 4Gb de RAM e *clock* de 2 Ghz.

Os resultados são apresentados nas **figuras 4** e **5**, respectivamente; e como pode ser observado, a **figura 4** mostra que o tempo total gasto para exibir os 78.498 números primos existentes no intervalo de 1-1000000 com o programa **“MostraPrimo”** foi de **51.158,04** segundos ou, aproximadamente, 14h12m38s; o que é um tempo extraordinariamente grande no contexto da computação. A **figura 5** mostra que no segundo caso, com o programa “MostraPrimo2”, o tempo foi ínfimo: de apenas **9,62** segundos para detectar e exibir os mesmos 78.498 primos no referido intervalo. Isto quer dizer que, considerando o mesmo conjunto de *hardware/software*, o tempo gasto para detectar primos pela “Radiação de Números” foi mais de cinco mil vezes menor que o tempo gasto utilizando a “Análise da Quantidade de Divisores”. No programa “MostraPrimo” (utilizando a “Análise da Quantidade de Divisores”) foi observado que nos primeiros dez mil números analisados o processamento até que desenvolveu com boa velocidade; entretanto, a partir daí a exibição de cada primo encontrado começou a ficar muito lenta. A explicação é simples: a cada novo número a ser analisado a quantidade de divisores aumenta, exigindo um maior esforço computacional, tornando a exibição dos números cada vez mais morosa. Está ai, portanto, a prova da baixa eficiência desse tipo de algoritmo, ainda muito utilizada pelos iniciantes em programação por ser o mais fácil e que segue à risca a definição de números primos.

A conclusão é que, além de um *hardware* bem potente e uma linguagem de codificação bem eficiente, o algoritmo empregado para criar o programa é o fator mais importante pois, a Programação é que define a qualidade do programa, e não a Codificação. É o algoritmo que dá a solução do problema; portanto, é ele que vai definir a qualidade do produto final, e não a linguagem de programação que o implementa! Pense BEM nisto, antes de optar por uma linguagem só porque está na “moda”, ou porque disseram para você que é a melhor!

---------------------------------------------------------------------------------

*“Que a Lógica esteja com Vocês...”*



*#MostraPrimo.py  
#Mostrar os números primos num intervalo dado pelo usuário, pelo  
#algoritmo de “Análise da Quantidade de Divisores”.  
#Autor: Mário Leite  
#-----------------------------------------------------------------------  
#Início programa***import** time  
endwhile = **"endwhile"**endfor = **"endfor"**endif = **"endif"**Lim1 = 1  
Lim2 = 0  
Cond = (Lim1>Lim2) **or** ((Lim1<1) **or** (Lim2<2))  
**while**(Cond):  
 print(**"\n"**)  
 Lim1 = int(input(**"Entre com o limite inferior: "**))  
 Lim2 = int(input(**"Entre com o limite inferior: "**))  
 Cond = (Lim1>Lim2) **or** ((Lim1<1) **or** (Lim2<2))  
**endwhile**  
print(**"\n"**)  
Cont = 0  
inicio = time.time() *#liga o cronômetro***for** j **in** range(Lim1,(Lim2+1)):  
 Num = j  
 QteDiv = 0  
 **for** j **in** range(1,(Num+1)):  
 **if**(Num % j == 0):  
 QteDiv = QteDiv + 1  
 **endif**  
 **endfor**  
 **if**(QteDiv==2):

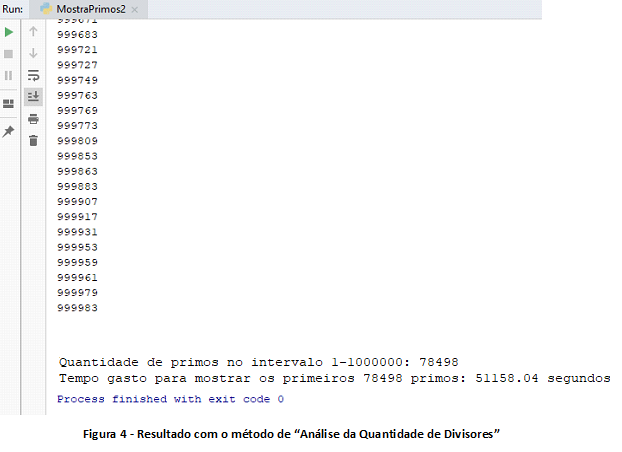
Cont = Cont + 1  
 print(Num) *#número é primo* **endif**  
**endfor**  
fim = time.time() *#desliga o cronômetro*tempo = fim - inicio  
tempo = int(tempo\*100+0.50)/100  
print(**""**)  
print(**f"Quantidade de primos no intervalo {**Lim1**}-{**Lim2**}: {**Cont**}"**)  
print(**f"Tempo gasto para mostrar os primeiros {**Cont**} primos: {**tempo**} segundos"**)  
*#FimPrograma------------------------------------------------------------*

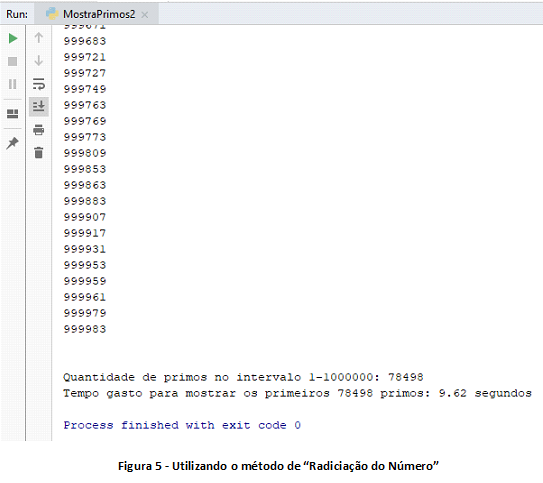
**Figura 2 - Código do programa “MostraPrimos.py”**

*#MostraPrimo2.py  
#Mostrar os números primos num intervalo dado pelo usuário, pelo*

*#”Método da Radiciação do Número”.  
#Autor: Mário Leite  
#-----------------------------------------------------------------------  
#Início programa***import** time  
**import** math  
endwhile = **"endwhile"**endfor = **"endfor"**endif = **"endif"**Lim1 = 1  
Lim2 = 0  
Cond = (Lim1>Lim2) **or** ((Lim1<1) **or** (Lim2<2))  
**while**(Cond):  
 print(**"\n"**)  
 Lim1 = int(input(**"Entre com o limite inferior: "**))  
 Lim2 = int(input(**"Entre com o limite inferior: "**))  
 Cond = (Lim1>Lim2) **or** ((Lim1<1) **or** (Lim2<2))  
**endwhile**  
print(**"\n"**)  
Cont = 0  
inicio = time.time() *#liga o cronômetro***for** n **in** range (Lim1,(Lim2+1)):  
 ProxNum = n + 1 *#pega o primeiro número após n  
 #Verifica se* ***ProxNum*** *é primo* IntRaiz = int(math.sqrt(ProxNum))  
 TemDiv = **False  
 for** k **in** range(2,(IntRaiz+1)): *#faz as divisões* RDiv = (ProxNum % k)  
 **if**(RDiv==0):  
 TemDiv = **True  
 break** *#abandona incondicionalmente o loop (não é primo)* **endif**  
 **endfor**  
 **if**(TemDiv==**False**):  
 Cont += 1  
 print(ProxNum)  
 **endif**  
**endfor**  
fim = time.time() *#desliga o cronômetro*tempo = fim - inicio  
tempo = int(tempo\*100+0.50)/100  
print(**"\n"**)  
print(**f"Quantidade de primos no intervalo {**Lim1**}-{**Lim2**}: {**Cont**}"**)  
print(**f"Tempo gasto para mostrar os primeiros {**Cont**} primos: {**tempo**} segundos"**)  
*#FimPrograma------------------------------------------------------------*

**Figura 3 - Código do programa para detectar números primos pela “Radiciação do Número”**



****