

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: **“número 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 “Análise 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 “Análise 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 aí, 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...”

```
4673331833592310999883355855611155212513211028177144957985823385935679234805211772074843110997402088496213680900
9638644050954128274109548519743273551014325753249976993808191641040774990607027085131780854431482719287927051574
6775097681495456790744215925392808604345151310705231857280062253517330504393154504927694689628526886967494434211
9923842039912255378570492258674504781998501869851883957199630800387179659069436984462272457690484426240770404565
679705245098048143893134795938877105350614496693489409255155953306872814733490045565082856578190868933327141046
7314331097452082351397488563625371930195040660572209571807917346778421232394122570849227616267884850424017561957
634978181198613503051581163463556335663198966158276043886903297960119840962705373835796308746810568436498180676
2986643959450031252516686566594252074345358101581042723707992427571828820948849065861590065859074391377828173034
8429761386038681691058654221673900140273498297190727298748366210723682751247273840980957893067062615324968571927
521356011614860439988085178768763757860732625585115457092087848043258257876286498706458088135038948822473518071
7150923894600650386095599714691658518044760943228485293103850039495787904594766307709643249139518714435912361386
0552808611097373063518704213113039301625336279405825183612805551854967893065836645527291551181195205888839259531
4859045762973388136611991702058586520990334357374055942865874795790720345913488804917784805628948577880461773217
7237403658550061799279956704411031254567465451105499362798947781210188466981867175415780089815289984924792655784
8926634460813390353581444858227418449682398889148543390840713875511844096578060875650223903048389134911781503058
1170248809970337175034006733019227380745118643000374192911602711339184034355819927937019521413721001355954575994
5830443576635739972027953438424843103674054245431824410173440025375378765370235220916664367509996157398715673180
7927872621210107894980393234159043797743621683940604454280871426803006376885767541206049493503285861437247700246
1938867611102089288594969572320827894145129733289906382456243680285209396595612312525464356962608456268063816100
4692470308676425544869996972655508942309011083411493144986858214357859775618228826805464134953908832407078932714
4711014261765297188923940302103960017543971660344237708051436459112134060551823257845881972600579821443217323082
1135285496319932750202723560645351954450558575024304982756602691737098834444724345145794308311422076942690678156
0411203408546402674370846173664950410342789789712394983000205121607980054733967940661028509909481275341568059005
8118938956060011462926788848882745583683953945900727360156133264642996097637565137471103345808889542561187761947
5527700800248788702494948646402243055553891105615244795599763312093904223553352012712634248336873955397492313545
79300249941511963879909715532876499817633204008490564422745086435385704095654841908739343344681837371740735456155
5615495745984561137169307338850398036433960613167695005547315465702636875944371145314526247047257439468995229505
7870713667038881169179400163966234000169103644971921705719046572009336055919952416747002005665082543329668873564
06035326756374786714833226191987027489874914292326706179893637230932073116363054714521145417546871219232736849
4106919036678940703125057091278488588365101649078423132421395068403883366113238919385507001819286118171101817447
9285520108702074373212498975657253759875555952410427085019182019620044750463260473440183873857816621871973857634
0808406574965746650020888383585612017491508583639776995904205591384756847211805011567922662242942584994983691734
2017798305230089700518572714616660119558582770274921322529927262779296034236446981306733105023187308382443843943
3258371591872651149517766504889563767789595366688737271634155875378019895536374868805853400066385199557459361521
9048856192870909624701021522468754861196594687226546505766804935872478889265466487543901302570459664074969027447
```

Figura 1 - Parte da imagem do maior número primo descoberto até dez/2017

```

#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-----

```

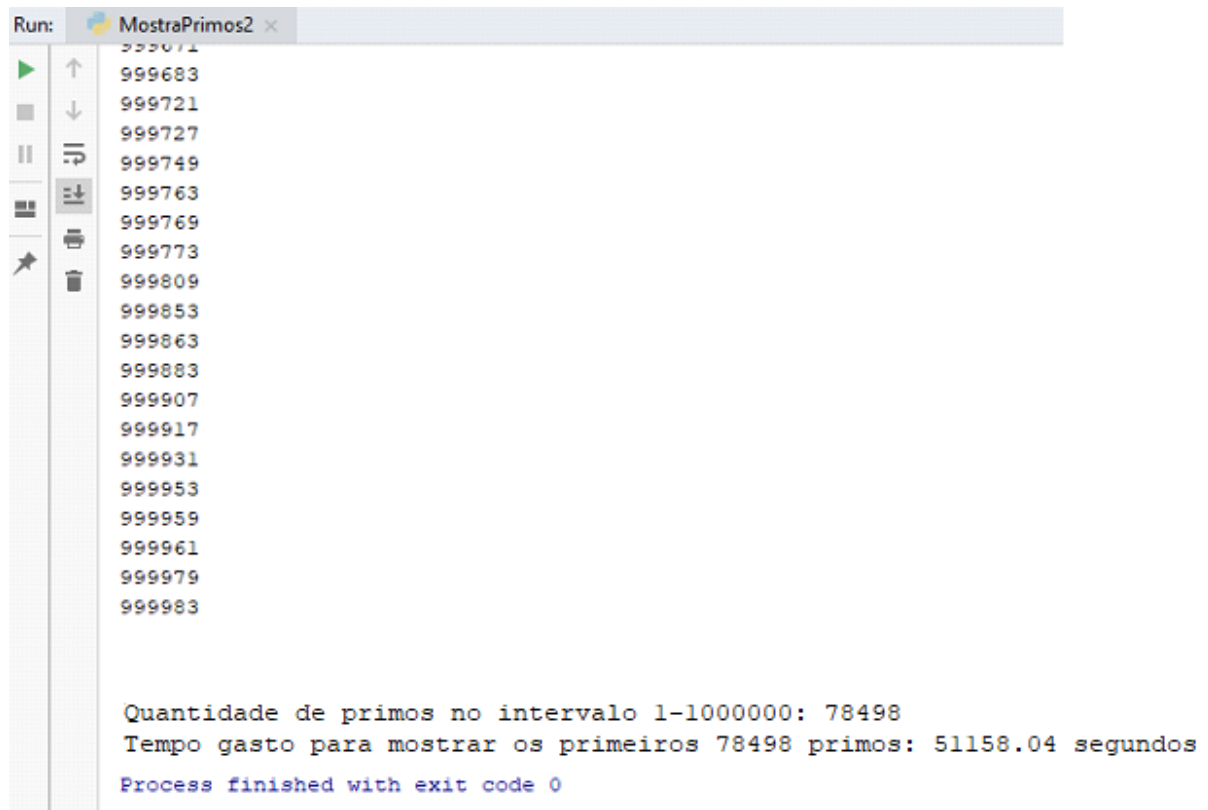
Figura 2 - Código do programa "MostraPrimos.py"

```

#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”



```
Run: MostraPrimos2 x
999671
999683
999721
999727
999749
999763
999769
999773
999809
999853
999863
999883
999907
999917
999931
999953
999959
999961
999979
999983

Quantidade de primos no intervalo 1-1000000: 78498
Tempo gasto para mostrar os primeiros 78498 primos: 51158.04 segundos
Process finished with exit code 0
```

Figura 4 - Resultado com o método de “Análise da Quantidade de Divisores”

```
Run: MostraPrimos2 x
999671
999683
999721
999727
999749
999763
999769
999773
999809
999853
999863
999883
999907
999917
999931
999953
999959
999961
999979
999983

Quantidade de primos no intervalo 1-1000000: 78498
Tempo gasto para mostrar os primeiros 78498 primos: 9.62 segundos

Process finished with exit code 0
```

Figura 5 - Utilizando o método de "Radiciação do Número"