MATEMÁTICA UNIVERSITÁRIA
nº. 22/23 – junho/dezembro 1997 – pp.1-13

Vendendo primos[1]

Paulo Ribenboim

Eu sou uma pessoa importante em uma fábrica que produz primos. E vou contar-lhes um interessante diálogo com um comprador vindo de um país exótico.

O Diálogo

Comprador: Eu desejo comprar alguns primos.

Eu (generosamente): Posso dar-lhe alguns primos gratuitamente: 2, 3, 5, 7, 11, 13, 17,

19, ...

Comprador (interrompendo minha generosa oferta): Muito obrigado, senhor. Mas eu quero primos com 100 dígitos. Você tem algum para vender?

Eu: Nesta fábrica nós podemos produzir primos tão grandes quanto você desejar. De fato, existe um velho método, devido a Euclides, do qual você já deve ter ouvido falar. Se tenho um número n qualquer de primos , digamos p_1, p_2,..., p_n, nós podemos multiplicá-los e adicionar 1 para obter o número $N = p_1 p_2 ... p_n + 1$. Então N é primo ou, se ele não é primo, podemos pegar um primo qualquer que divida N. É fácil ver que, dessa forma, conseguimos um primo que é diferente daqueles que misturamos. Chamemos esse primo de p_{n+1}. Se agora misturamos $p_1, p_2, ..., p_n, p_{n+1}$ como fizemos anteriormente, obtemos ainda um novo primo p_{n+2}. Repetindo esse processo podemos obter tantos primos quanto desejarmos, assim como tão grandes quanto queiramos, com certeza com pelo menos 100 dígitos.

[1]Conferência proferida na Universidade de Augsburg, em 23 de junho de 1992

Comprador: Você é muito gentil em explicar o seu processo. Mesmo em meu distante país eu tenho ouvido falar dele. Ele fornece primos que podem ser arbitrariamente grandes. Contudo, eu quero comprar primos que tenham exatamente 100 dígitos, nem mais e nem menos. Você os tem?

Eu: Sim. Há muito tempo – no começo do século passado – Bertrand observou que entre qualquer número $N > 1$ e o seu dobro $2N$ existe pelo menos um número primo. Esta observação experimental foi confirmada por Chebyshev com uma demonstração rigorosa. Portanto eu posso encontrar primos p_1, p_2, p_3 tais que

$$\begin{array}{ccccc} 10^{99} & < & p_1 & < & 2 \times 10^{99} \\ 2 \times 10^{99} & < & p_2 & < & 4 \times 10^{99} \\ 4 \times 10^{99} & < & p_3 & < & 8 \times 10^{99}. \end{array}$$

Comprador: Isto significa que você tem assegurado três primos com 100 dígitos e talvez um pouco mais. Mas eu quero comprar muitos primos com 100 dígitos. Quantos você pode produzir?

Eu: Jamais contei quantos primos com 100 dígitos poderiam ser eventualmente produzidos. Soube que meus colegas em outras fábricas calcularam o total de primos até 10^{17}. Nós usualmente escrevemos $\pi(N)$ para denotar o número de primos até o número N. Assim, o cálculo que mencionei fornece

$$\begin{array}{rcl} \pi(10^8) & = & 5.761.455, \\ \pi(10^9) & = & 50.847.534, \\ \pi(10^{12}) & = & 37.607.912.018, \\ \pi(10^{17}) & = & 2.625.557.157.654.233. \end{array}$$

Muito embora todos os primos até 10^{17} ainda não tenham sido produzidos por fábrica alguma, o cálculo de $\pi(10^{17})$ é exato.

Comprador (um pouco atônito): Se você não pode – como entendo – saber quantos primos de cada tamanho grande existem em estoque, como você pode dirigir a sua fábrica e garantir a entrega da mercadoria?

Eu: O seu país vende petróleo, não é? Você pode estimar a quantidade de petróleo em regiões não muito profundas quase com exatidão, mas não pode medir exatamente a quantidade total existente no subsolo. É justamente o mesmo que se passa conosco. Gaussfoi uma das primeiras pessoas a descobrie que

$$\pi(N) \sim \frac{N}{logN}$$

para valores grandes de N, o que foi confirmado, quase um século depois, por uma demonstração dada por Hadamard e de la Vallée Poussin. Este resultado é conhecido como o Teorema do Nú mero Primo.

Comprador: Você quer dizer que $\pi(N)$ é aproximadamente igual a $N/logN$, com um pequeno erro?

Eu: Sim. Para ser mais preciso, o erro relativo, isto é, o valor absoluto da diferença $|\pi(N) - N/logN|$ dividido por $\pi(N)$, tende a zero quando N cresce indefinidamente.

Comprador: Então, por causa do erro, você não pode ser mais específico em sua estimativa. Exceto se você estimar o erro.

Eu: Correto (o comprador não é tolo ...). Chebyshev mostrou, antes mesmo do Teorema do Número Primo ser provado, que se N é grande então

$$0,9\frac{N}{logN} < \pi(N) < 1,1\frac{N}{logN} .$$

Para contar primos com 100 dígitos,

$$0,9\frac{10^{99}}{99log10} < \pi(10^{99}) < 1,1\frac{10^{99}}{99log10} ,$$

$$0,9\frac{10^{100}}{100log10} < \pi(10^{100}) < 1,1\frac{10^{100}}{100log10} .$$

É fácil estimar a diferença $\pi(10^{100}) - \pi(10^{99})$, que dá o número de primos com exatamente 100 dígitos:

$$3,42 \times 10^{97} < \pi(10^{100}) - \pi(10^{99}) < 4,38 \times 10^{97}.$$

Comprador: Você está rico! Acho que você tem mais primos do que nós temos petróleo. Mas eu gostaria de saber como a sua fábrica produz os primos com 100 dígitos. Eu tenho uma idéia, mas não estou seguro da eficiência do meu mé todo. A idéia é a seguinte:

1. Escreva todos os números com 100 dígitos.
2. Risque, em seqüência, todos os múltiplos de 2, de 3, de 5, ...,de cada primo p menor que 10^{99}. Para isto, localize o primeiro múltiplo de p e risque este e cada p-ésimo número seguinte.

O que restar serão os primos entre 10^{99} e 10^{100}, isto é, os primos com exatamente 100 dígitos.

Eu: Esse procedimento é correto e já foi descoberto por Eratóstenes (no terceiro século A.C.). De fato, você pode parar quando já tiver riscado todos os múltiplos de todos os primos menores que 10^{50}.

Entretanto, esse método de produção é muito lento. Isto explica porque os arqueólogos nunca encontraram uma fábrica de primos entre as ruinas gregas, mas tão somente templos dedicados a Apolo, estátuas de Afrodite (conhecida como Vênus, desde o tempo dos romanos) e outras ruinas feias que dão testemunho de um alto grau de decadência.

Mesmo com computadores, esse processo é muito lento para ser prático. Imagine um computador que escreve 10^6 dígitos por segundo.

- Existem $10^{100} - 10^{99} = 10^{99} \times 9$ números com 100 dígitos.
- Esses números têm um total de $10^{101} \times 9$ dígitos.
- São necessários $10^{95} \times 9$ segundos para escrever esses números, o que é cerca de $1,5 \times 10^{94}$ minutos, o que é cerca de 25×10^{92} horas, portanto mais de 10^{91} dias, o que é da ordem de 3×10^{88} anos e equivale a 3×10^{86} séculos!

E depois de escrever os números (se ainda existir um Depois...) há muito mais para ser feito.

Diante do comprador insatisfeito, acrescentei:

Eu: Existem atalhos, mas mesmo assim o método ainda seria muito lento. Assim, em lugar de tentar listar os primos com 100 dígitos, nossa fábrica usa algorítmos rápidos para produzir primos em número suficiente para atender os pedidos de nossos clientes.

Comprador: Estou maravilhado. Eu nunca havia pensado o quanto é importante ter um método rápido. Você pode contar-me o procedimento usado em sua fábrica? Estou realmente curioso. [Sim, esse comprador estava sendo muito inquisitivo. Hoje estou convencido de que ele era um espião].

Eu: Quando você compra um carro Mercedes, você não pergunta como ele foi fabricado. Você escolhe a sua cor favorita, rosa, lilás, ou verde com detalhes em laranja, você o dirige e você é feliz porque todos sentem inveja de você.

Nossa fábrica entregará os primos que você solicitar e faremos melhor do que a Mercedes. Nós garantimos o nosso produto por toda a vida. Adeus, senhor. [Ele deve ter entendido: Adeus, senhor...].

Após o Diálogo

Espero que após o diálogo com o comprador espião vocês tenham ficado curiosos para saber sobre o nosso processo rápido de produção de primos grandes. Vou contar-lhes um dos nossos mais caros segredos. Em nossa fábrica existem duas divisões principais.

1) Produção de primos.
2) Controle de qualidade.

Produção de Primos

Uma das bases de nossos métodos de produção foi descoberta há muito tempo atrás por Pocklington [4]. Vou enunciar e demonstrar esse teorema na situação particular adaptada às nossas necessidades de produção. A seguir analisarei como ele pode ser usado para obter primos, num tempo surpreendentemente curto, com o número de dígitos requerido.

Critério de Pocklington. *Sejam p um primo ímpar, k um número natural não divisível por p, $1 \leq k < 2(p+1)$ e $N = 2kp + 1$. Então as seguintes condições são equivalentes:*

1) *N é primo.*
2) *Existe um número natural a, $2 \leq a < N$ tal que*

$$a^{kp} \equiv -1 (mod\ N).$$

e

$$mdc(a^k + 1, N) = 1.$$

Demonstração: 1) $\Rightarrow$ 2). Suponhamos que N seja primo. Como é sabido, existe algum inteiro a, $1 < a < N$, tal que $a^{N-1} \equiv 1 (mod\ N)$ e $a^m \not\equiv 1 (mod\ N)$ se $1 < m < N - 1$. Tal número a é chamado *raíz primitiva módulo N*. Então $a^{2kp} \equiv 1 (mod\ N)$, mas $a^{kp} \not\equiv 1 (mod\ N)$. Logo $a^{kp} \equiv -1 (mod\ N)$. Ainda, $a^k \not\equiv -1 (mod\ N)$ pois, caso contrário, $a^{2k} \equiv 1 (mod\ N)$, o que não é verdade. Portanto $mdc(a^k + 1, N) = 1$.

2) $\Rightarrow$ 1). Para mostrar que N é primo, provaremos que se q é um divisor primo de N então $\sqrt{N} < q$. Disto segue-se que N não pode ter dois (iguais ou distintos) fatores primos e, portanto, N é primo.

Seja q um fator primo de N. Então $a^{kp} \equiv -1 (mod\ q)$ e conseqüentemente $mdc(a, q) = 1$. Seja e a ordem de a módulo q. Pelo

pequeno teorema de Fermat e divide $q-1$. Similarmente e divide $2kp = N-1$, porque $a^{2kp} \equiv 1(mod\ q)$. Notemos que $a^k \not\equiv 1(mod\ q)$ pois, caso contrário, teríamos $a^{kp} \equiv 1(mod\ q)$ e de $a^{kp} \equiv -1(mod\ q)$ seguir-se-ia então que $q = 2$ e N seria par, o que é falso. De $mdc(a^k+1, N) = 1$ decorre que $a^k \not\equiv -1(mod\ q)$. Então $a^{2k} \not\equiv 1(mod\ q)$ e, por conseguinte, $e \nmid 2k = (N-1)/p$. Mas $e|(N-1)$, logo $(N-1)/e$ é um inteiro e portanto $p \nmid (N-1)/e$. Como $N-1 = e(N-1)/e$ e $p|(N-1)$, segue-se que $p|e$ e, por conseguinte, $p|q-1$. Ainda, $2|(q-1)$; logo $2p|(q-1)$ e portanto $2p+1 \leq q$. Segue-se então que $N = 2kp+1 < 2 \times 2(p+1)p+1 = 4p^2+4p+1 = (2p+1)^2 \leq q^2$. Portanto $\sqrt{N} < q$ e isto conclui a demonstração.

Para se obter primos de um tamanho requerido, digamos com 100 dígitos, o critério de Pocklington é aplicado como segue.

Primeiro passo: Escolher, por exemplo, um primo p_1 com $d_1 = 5$ dígitos. Encontrar $k_1 < 2(p_1+1)$ tal que $p_2 = 2k_1p_1+1$ tenha $d_2 = 2d_1 = 10$ dígitos ou $d_2 = 2d_1-1 = 9$ dígitos e exista $a_1 < p_2$ satisfazendo as condições $a_1^{k_1p_1} \equiv -1(mod\ p_2)$ e $mdc(a_1^{k_1}+1, p_2) = 1$. Pelo critério de Pocklington p_2 é primo.

Passos seguintes: Repetir o mesmo procedimento começando com o primo p_2 para obter p_3, etc... Para produzir um primo com 100 dígitos, o processo deve ser repetido 5 vezes. Na última etapa k_5 deve ser escolhido tal que $2k_5p_5+1$ tenha 100 dígitos.

Exeqüibilidade do Algorítmo

Dados p e k, com $1 \leq k < 2(p+1)$, k não múltiplo de p, se $N = 2kp+1$ é um primo, então ele tem uma raiz primitiva. Seria demasiadamente técnico explicar em detalhes os resultados seguintes, alguns conhecidos dos especialistas, outros ainda não publicados. Decorre de uma forma generalizada da hipótese de Riemann que se x é um número real positivo grande e o número inteiro a não é um quadrado então a razão

$$\frac{\#\{\text{primos } q \leq x \text{ tais que } a \text{ é uma raiz primitiva módulo } q\}}{\#\{\text{primos } q \leq x\}}$$

converge; se a é um primo, o limite é pelo menos igual à constante de Artin

$$\prod_{q \text{ primo}} (1 - \frac{1}{q(q-1)}) \approx 0,37.$$

Melhor: dados inteiros positivos a e b nãoquadrados e um primo q grande, a probabilidade de a ou b ser uma raiz primitiva módulo q é muito grande. Para $a = 2$ e $b = 3$ ela é de pelo menos 58%. A correspondente probabilidade aumenta substancialmente quando consideramos três inteiros positivos e nãoquadrados a, b e c.

Isto sugere que procedamos como se segue. Dado o primo p, escolha k não múltiplo de p, $1 \leq k < 2(p+1)$. Se $N = 2kp + 1$ é um primo, então muito provavelmente 2, 3 ou 5 é uma raiz primitiva módulo N. Se não for este o caso, é mais prático escolher outro inteiro k', como k, e investigar se $N' = 2k'p + 1$ é primo.

Surge então a questão: Quais são as chances de encontrar k tal que N seja um primo? Analiso agora esse ponto.

1. De acordo com um caso especial do famoso teorema de Dirichlet (ver [5], [6]), dado p, existem infinitos inteiros $k \geq 1$ tais que $2kp+1$ é um primo. Isto pode ser provado de maneira elementar.
2. Quaão pequeno k pode ser tal que $2kp+1$ seja um primo? Um caso especial de um teorema profundo de Linnik assegura: "Para cada p suficientemente grande, na progressão aritmética com primeiro termo 1 e diferença $2p$, existe um primo $p_1 = 2kp + 1$ satisfazendo $p_1 \leq (2p)^L$, onde L é uma constante positiva independente de p" (ver [5]).
3. Recentemente Heath e Brown mostraram que $L \leq 5,5$.
4. No critério de Pocklington requer-se encontrar $k < 2(p+1)$ tal que $p_1 = 2kp + 1$ seja primo. Isto implica que $p_1 < (2p+1)^2$. Nenhum teorema conhecido garante que tais valores pequenos de k levam a um primo.
5. Um trabalho recente de Bombieri, Friedlander e Iwaniec trata de primos p para os quais existem primos pequenos $p_1 = 2kp + 1$. Seus resultados, os quais dizem respeito a médias, apontam para a existência de uma proporção razoavelmente grande de primos p com primos pequenos $p_1 = 2kp + 1$.

Os problemas considerados acima são muito difíceis. Na prática, podemos ignorar essas considerações e encontrar, com algumas tentativas, o valor apropriado de k.

Tempo Estimado para Produzir Primos com 100 Dígitos

O tempo de execução de um algoritmo depende da velocidade do computador e do número de operações bit (i.e. operações com dígitos) necessárias. Como ponto de partida para esta análise podemos assumir que o computador executa 10^6 operações bit por segundo. Se estimarmos um limite superior para o número de operações bit, dividindo esse valor limite por 10^6, obtemos um limite superior para o número de segundos necessários.

Olhando mais de perto esse procedimento vemos que ele consiste numa seqüência das seguintes operações com números naturais: multiplicação ab módulo n, potência a^b módulo n, cálculo de máximo divisor comum.

É bem conhecido (ver [1], [2]), e não é difícil demonstrar, que para cada uma dessas operações existem $C > 0$ e um inteiro $e \geq 1$ tais que o número de operações bit necessárias para executar o cálculo é no máximo Cd^e, onde d é o maior dos números de dígitos dos números envolvidos. Combinando essas estimativas, obtém-se para o método um limite superior do mesmo tipo Cd^e, com $C > 0$, $e \geq 1$ e d o máximo dos números de dígitos de todos os inteiros envolvidos no cálculo.

Não é minha intenção dar valores explícitos para C e e quando p, k e a são dados. Deixem-me apenas dizer que se C e e são bem pequenos então a execução do algorítmo é muito rápida. Enfatizo que, nesta estimativa, o tempo necessário para a procura de k e de a não foi levado em conta.

Essas considerações deixam claro que resta muito mais para ser entendido sobre a produção de primos e a exeqüibilidade do algoritmo. Esta tarefa é deixada para a divisão de pesquisa e desenvolvimento de nossa empresa e eu admiro os nossos colegas da subdivisão de pesquisa que enfrentam os profundos mistérios dos números primos.

Antes de percorrer rapidamente a nossa divisão de controle de qualidade, eu gostaria de fazer alguns breves comentários sobre nossas considerações anteriores. Eles referem-se à complexidade de um algoritmo.

Um algoritmo A chama-se *polinomial* se existem inteiros C e e (dependendo do algoritmo) tais que o número de operações bit (ou, equivalentemente, o tempo) necessário para a sua execução sobre os números naturais, com no máximo d dígitos , é no máximo Cd^e.

Um algoritmo que não é polinomial é definitivamente muito caro e é

rejeitado pela nossa fábrica. Um dos principais objetos de pesquisa é a construção de algoritmos que sejam polinomiais. Na prática, o algoritmo para produzir primos de um dado tamanho é polinomial, muito embora isto ainda não tenha sido assegurado por uma demonstração.

Controle de Qualidade

A divisão de controle de qualidade de nossa fábrica cuida para que os primos que vendemos sejam de fato primos. Quando o método de Pocklington é utilizado, necessitamos preocuparnos somente com erros tolos de cálculo eventualmente cometidos, pois esse método leva automaticamente a números primos. Se outros métodos são utilizados, como o que abordarei logo em seguida, deve haver um controle. A divisão de controle de qualidade também ocupa-se de trabalho de consultoria. Um número grande N é apresentado, juntamente com a questão: N é um número primo?

Portanto, nossa divisão de controle de qualidade também trabalha com testes de primalidade. Como esta é uma atividade remunerável, atualmente existem muitos testes de primalidade disponíveis. Posso classificá-los resumidamente dos seguintes quatro pontos de vista:

1. Testes para números genéricos.
   Testes para números de formas especiais tais como $F_n = 2^{2^n} + 1$ (números de Fermat), $M_p = 2^p - 1$ (p primo, números de Mersenne), etc...
2. Testes completamente justificáveis por teoremas.
   Testes baseados em justificativa que depende de formas da hipótese de Riemann sobre os zeros da função zeta ou de argumentos heurísticos.
3. Testes determinísticos.
4. Testes probabilísticos ou de Monte Carlo.

Um teste determinístico aplicado a um número N assegurará que N é um número primo ou que N é um número composto. Um teste de Monte Carlo aplicado a N atestará que N é um número composto ou que N tem uma grande probabilidade de ser primo.

Antes de continuar, deixem-me dizer que o principal problema que seduz os pesquisadores é o seguinte: será possível encontrar um teste de

primalidade, completamente justificado e determinístico para números genéricos, que seja polinomial? Ou será provado que não pode existir um tal teste de primalidade?

Este é um problema profundo e atormentador.

Seria demasiado tedioso, e mesmo complexo, tentar descrever todos os métodos e algoritmos usados em testes de primalidade. Assim, vou concentrar-me somente sobre o teste pseudoprimo forte, que é do tipo Monte Carlo.

Pseudoprimos. Sejam N um primo e a um inteiro tal que $1 < a < N$. Pelo pequeno teorema de Fermat, $a^{N-1} \equiv 1(mod\ N)$.

Entretanto, a recíproca não é verdadeira. O menor exemplo é $N = 341 = 11 \times 31$; com $a = 2$, $2^{340} \equiv 1(mod\ 341)$.

Dados dois números a e N, com $1 < a < N$ e $mdc(a, N) = 1$, o número N é chamado um *pseudoprimo na base* a se N é composto e $a^{N-1} \equiv 1(mod\ N)$. Para cada $a \geq 2$, existem infinitos pseudoprimos na base a. Agora, observem que cada primo ímpar N satisfaz a seguinte propriedade:

$(\star)$ Para todo a, $2 \leq a < N$, com $mdc(a, N) = 1$, escrevendo $N - 1$ na forma $N - 1 = 2^s d$ (com $s \geq 1$, d ímpar), ou $a^d \equiv 1(mod\ N)$ ou existe r, $0 \leq r \leq s$ tal que $a^{2^r d} \equiv -1(mod\ N)$.

De novo, a recíproca não é verdadeira, conforme ilustrado por $N = 2047 = 23 \times 89$, com $a = 2$.

Dados dois números a e N, com $1 < a < N$ e $mdc(a, N) = 1$, o número N é chamado *pseudoprimo forte na base* a se N é composto e a condição $(\star)$ é satisfeita.

Foi demonstrado por Pomerance, Selfridge e Wagstaff que para todo $a \geq 2$ existem infinitos pseudoprimos fortes na base a.

O Teste Pseudoprimo Forte. As principais etapas no teste pseudoprimo forte para um número N são as seguintes:

1. Escolher $k > 1$ números a, $2 \leq a < N$, tais que $mdc(a, N) = 1$. Isto pode ser feito facilmente por meio de testes com divisão e não requer conhecimento dos fatores primos de N. Se $mdc(a, N) > 1$ para algum a, $1 < a < N$, então N é composto.
2. Para cada base a escolhida verificar se a condição $(\star)$ é satisfeita.

Se existe a tal que $(\star)$ não é satisfeita então N é composto. Portanto, se N é um primo, a condição $(\star)$ é satisfeita para toda base a.

Os eventos em que a condição $(\star)$ é satisfeita para diferentes bases podem ser legitimamente considerados como independentes se as bases são escolhidas ao acaso.

Agora, Rabin provou o seguinte (ver [5]): "Seja N composto. Então o número de bases a em relação às quais N é pseudoprimo forte é menor que $\frac{1}{4}(N-1)$". Logo, se N é composto, a probabilidade de $(\star)$ ser satisfeita para k bases é no máximo $1/4^k$. Portanto, a atestação de que N é um primo quando $(\star)$ é satisfeita para k bases distintas é incorreta em somente um dentre 4^k números . Por exemplo, se $k = 30$, a atestação é incorreta somente uma vez em cada 10^{18} números.

O teste pseudoprimo forte é polinomial e é aplicável a qualquer número.

Admitindo-se a forma generalizada da hipótese de Riemann como verdadeira, Miller mostrou que (ver [5]): "Se N é composto, existe uma base a, com $mdc(a, N) = 1$ e tal que $a < (\log N)^{2+\epsilon}$, para a qual $(\star)$ não é satisfeita".

Um novo método de produção. Podemos usar o teste de Rabin para produzir números com 100 dígitos que podem ser atestados como números primos, com uma probabilidade de erro muito pequena.

1. Pegue um número N com 100 dígitos. Antes de fazer qualquer trabalho duro, é mais fácil, através de testes com divisão, descobrir se esse número tem ou não qualquer fator primo menor que, digamos, 1000. No último caso, conserve esse número; no primeiro caso descarte N, pegue outro número N' e proceda de modo similar.
2. Use $k = 30$ números pequenos a, primos com N, como bases para verificar se a condição $(\star)$ é satisfeita. Descarte N se para alguma base a a condição $(\star)$ não for satisfeita e repita o processo com algum outro número N'. Se $(\star)$ é satisfeita para todo a, então, de acordo com o cálculo de Rabin, podemos atestar que N é primo.

Procedendo assim estaremos incorrendo em erro em no máximo um dentre 10^{18} números. Quão azarados podemos ser ao escolher, em seqüência, somente números que são compostos? De acordo com as desigualdades de Chebyshev, anteriormente mencionadas, a proporção de números com 100 dígitos e que são primos não é menor que $\frac{3,42}{9\times10^2} \approx \frac{1}{260}$ e não é maior que $\frac{4,38}{9\times10^9} \approx \frac{1}{205}$. Empregados pouco inteligentes que poderiam pegar

números pares ou números divisíveis por 3, 5,..., ou por primos pequenos (digamos até 1000) certamente serão despedidos. Logo, a chance de se conseguir um primo aumenta e torna-se quase razoável usar o método de produção de Rabin.

Nós podemos vender o número N como se ele fosse primo, mesmo com uma "garantia de devolução do dinheiro" porque a probabilidade de estarmos vendendo um número composto é de somente 1 em cada 1.000.000.000.000. 000.000 de vendas! Esta é a melhor garantia que qualquer um pode conseguir em qualquer negócio. Estamos certos de que nossa empresa não irá à falência e continuará a sustentar generosamente minhas viagens para promover os nossos produtos – tudo complementado por pródigos jantares e vinhos finos, para ajudar a convencer nossos clientes de que primos são o modo de vida.

Apêndice

Existe um exemplo de um primo com 100 dígitos que foi calculado por L. Roberts usando o método de Pocklington.

$p_1 = 2333 \qquad k_1 = 2001 \qquad a_1 = 2$

$p_2 = 9336667 \qquad k_2 = 9336705 \qquad a_2 = 3$

$p_3 = 174347410924471 \qquad k_3 = 174347410924479 \qquad a_3 = 2$

$p_4 = 607940393921354891483080512 19$

$k_4 = 607940393921354891483080512 56 \qquad a_4 = 3$

$p_5 = 739183045122504318980574950295193587260702667372985056212 9$

$k_5 = 500000000000000000000000000000000000000000137 \qquad a_5 = 2$

$p_6 = 739183045122504318980574950295193587260905203527348622396 3$
$00677536380883042909432530860197905402334 7$

Agradecimentos a Bart Braden pelas amáveis sugestões de estilo.

Referências

[1] Le Veque, W. J. (editor) **Studies in Number Theory**, MAA. Washington, 1975.

[2] Mignotte, M. **Mathematics for Computer Algebra**, Springer Verlag, New York, 1991.

[3] Plaisted, D.A. *Fast verification, testing and generation of large primes*, Theor. Comp. Sci. 9 (1979), 1-16.

[4] Pocklington, H. C. *The determination of the prime or composite nature of large numbers by Fermat's theorem*, Proc. Cambridge Phil. Soc. 18 (1914/16), 29-30.

[5] Ribenboim, P. **The Book of Prime Numbers Records**, 2nd edition, Springer Verlag, New York, 1989.

[6] Ribenboim, P. **The Little Book of Big Primes**, Springer Verlag, New York,1991.

Queen's University
Kingston, Ontario, Canada K7L 3N6

Tradução: Antônio Paques
Revisão: Elon Lages Lima e Michel Spira.