Critérios de primalidade

Carlos Gustavo T. de A. Moreira, Nicolau C. Saldanha

19 de junho de 2012

Desde tempos remotos, problemas concernentes a números primos têm fascinado os matemáticos. De fato, Karl Friedrich Gauß (1777–1855) chegou a afirmar em seu Disquisitiones Arithmeticae (1801): "O problema de distinguir números primos de compostos e de decompor esses últimos em seus fatores primos é conhecido como sendo um dos mais importantes e úteis na aritmética. . . . . a dignidade da própria ciência parece requerer que todos os meios possíveis sejam explorados para a solução de um problema tão elegante e tão celebrado".

Não se conhece nenhuma fórmula simples para gerar primos arbitrariamente grandes. Uma palavra imprecisa mas importante nesta frase é "simples". Existem fórmulas que geram números primos, mas que são tão complicadas que não ajudam muito nem a gerar números primos explicitamente nem a responder perguntas teóricas sobre a distribuição dos primos. Um exemplo de fórmula para p_n, o n-ésimo primo, é

$$p_n = \left\lfloor 1 - \frac{1}{\log 2} \log \left(-\frac{1}{2} + \sum_{d|P_{n-1}} \frac{\mu(d)}{2^d - 1} \right) \right\rfloor,$$

onde $P_{n-1} = p_1 p_2 \cdots p_{n-1}$; aqui mu é função de Möbius: $\mu(n) = 0$ se n for múltiplo de algum quadrado de primo e $\mu(n) = (-1)^k$ se n for o produto de k primos distintos. Deixamos a demonstração a cargo do leitor. Outra fórmula é

$$p_n = \lfloor 10^{2^n} c \rfloor - 10^{2^{n-1}} \lfloor 10^{2^{n-1}} c \rfloor,$$

onde

$$c = \sum_{n=1}^{\infty} \frac{p_n}{10^{2^n}} = 0.0203000500000007\ldots.$$

A inutilidade desta última fórmula vem do fato que para calcular c devemos encontrar todos os primos; a fórmula se tornaria mais interessante se existisse outra interpretação para o número real c, o que parece muito improvável. Mills ([6]) também provou que existem números reais $A > 1$ tal que $\lfloor A^{3^n} \rfloor$ é primo para todo $n \in \mathbb{N}$.

Um tipo de fórmula para primos, de certa forma mais intrigante, são polinômios de coeficientes inteiros em S variáveis com a seguinte propriedade quase mágica: a intersecção da imagem de $\mathbb{N}^S$ com $\mathbb{N}$ é exatamente o conjunto dos números primos. Note que se tomarmos um ponto de $\mathbb{N}^S$ "ao acaso", o valor do polinômio neste ponto quase certamente será negativo; assim, é difícil usar o polinômio para gerar primos.

A melhor maneira conhecida de gerar primos extremamente grandes é procurar *primos de Mersenne*, i.e., números da forma $2^p - 1$, p primo. Deixamos como exercício para o leitor verificar que se $2^n - 1$ é primo então n é primo. Os primeiros primos de Mersenne são 3, 7, 31 e 127, correspondentes a $p = 2, 3, 5, 7$. São conhecidos hoje (fevereiro de 2011) 47 primos de Mersenne e os 9 maiores primos conhecidos são todos desta forma: o maior deles corresponde a $p = 43112609$ e tem 12978189 algarismos; os outros oito correspondem aos seguintes valores de p: 42643801, 37156667,

32582657, 30402457, 25964951, 24036583, 20996011, 13466917. Conjectura-se que existam infinitos primos de Mersenne. Lembramos que um inteiro positivo é *perfeito* se ele for igual à soma de seus divisores positivos próprios. Os primeiros números perfeitos são 6 e 28; não é difícil verificar que os números perfeitos pares são exatamente os números da forma $2^{p-1}(2^p-1)$, onde 2^p-1 é primo de Mersenne. Um dos poucos problemas em aberto da Matemática que foram formulados na antiguidade é decidir se existem números perfeitos ímpares.

Uma questão relacionada com a de *gerar* números primos é a de *testar* se um determinado número é primo. Com o advento dos computadores, a partir da década de 60, surgiram inúmeras tentativas de se obter um algoritmo eficiente para o teste de primalidade de um número. A relevância desse problema tem crescido imensamente em anos recentes devido à utilização intensa de números primos em algoritmos de criptografia, como os algoritmos RSA e El Gammal para criptografia pública. Dessa forma o problema do teste de primalidade se tornou um importante problema para a ciência da computação teórica. Sobre esse ponto de vista duas coisas são requeridas: um certificado de prova de que o algoritmo realmente produz a resposta correta; e uma medida da eficiência do algoritmo, isto é, quão bem o algoritmo faz uso dos recursos computacionais (como o tempo ou número de passos executados, espaço ou memória utilizada) em função do tamanho da entrada do problema para a obtenção da solução.

Existe um algoritmo bastante simples, devido ao matemático grego Eratóstenes (ca. 240 A.C.), para testar se qualquer inteiro positivo n é primo: calcule o resto da divisão de n por cada inteiro m com $2 \leq m \leq \sqrt{n}$. Se o resto for 0 em algum caso então n é composto e encontramos um divisor; se isto nunca ocorrer, n é primo. O inconveniente deste algoritmo é que ele é muito lento. O tamanho da entrada do algoritmo para um dado número n é o tamanho da sua codificação em bits, que é aproximadamente $k = \log_2 n$ pois $2^k \leq n < 2^{k+1}$. Portanto, em termos do tamanho da entrada k, temos que o número de operações é $O(\sqrt{n}) = O(2^{k/2})$, ou seja, o algoritmo tem complexidade de tempo exponencial no tamanho da entrada.

Uma ideia mais bem sucedida para testar primalidade é a de usar o pequeno teorema de Fermat, que afirma que para todo primo p e todo inteiro a temos $a^p \equiv a \pmod{p}$ (lembramos que $a \equiv b \pmod{n}$ significa que $b-a$ é múltiplo de n). Para testar a primalidade de n, tomamos a, $1 < a < n$, e calculamos $a^{n-1} \bmod n$. Se n for primo teremos $a^{n-1} \equiv 1 \pmod{n}$; qualquer outro resultado indica que n é composto mesmo sem termos encontrado um fator de n. Observe que para calcular $a^{n-1} \bmod n$ não precisamos calcular $a \cdot a \cdots a$, $n-1$ vezes. Podemos fazer esta conta com menos de $4 \log_2 n$ operações envolvendo inteiros menores do que n^2: se $n-1 = \sum_{0 \leq i < N} b_i 2^i$, $N = \lfloor \log_2(n-1) \rfloor$, então definimos

$$p_k = a^{\sum_{0 \leq i < k} b_{N-k+i} 2^i} \bmod n$$

e temos $p_0 = 1$, $p_N = a^{n-1} \bmod n$, e podemos calcular p_{k+1} a partir de p_k com uma operação de elevar ao quadrado, tomar o resto da divisão por n, possivelmente multiplicar por a e novamente de tomar o resto da divisão por n.

Se $a^{n-1} \equiv 1 \pmod{n}$, por outro lado, não demonstramos que n é primo; se n for composto satisfazendo $a^{n-1} \equiv 1 \pmod{n}$ dizemos que n é um *pseudoprimo* na base a. Pseudoprimos existem mas são raros (ver [4]): o menor pseudoprimo na base 2 é $341 = 11 \cdot 31$ e existem apenas 21 853 pseudoprimos na base 2 menores do que $2,5 \cdot 10^{10}$ (contra 1 091 987 405 primos). Pomerance (melhorando um resultado anterior de Erdős) provou que se $P\pi_a(x)$ é o número de pseudoprimos até x na base a temos

$$P\pi_a(x) \leq x \cdot e^{-\frac{\log x \log\log\log x}{2 \log\log x}}$$

para x suficientemente grande (os logaritmos são todos na base e). Vale comparar este resultado com o Teorema dos Números Primos, conjecturado por Gauß e pro-

vado independentemente por Hadamard e de la Vallée Poussin em 1896, que diz que o número de primos até x é

$$\pi(x) \approx \frac{x}{\log x}.$$

Em particular, há muito mais primos do que pseudoprimos.

Uma ideia natural para contornar a dificuldade criada pela existência de pseudoprimos é a de testar vários valores de a. É um fato interessante que existam alguns raros números compostos n, chamados *números de Carmichael*, com a propriedade de que se $0 < a < n$ e $\text{mdc}(a, n) = 1$ então $a^{n-1} \equiv 1 \pmod{n}$; o leitor pode verificar que o menor número de Carmichael é 561. Foi demonstrado recentemente por Alford, Granville e Pomerance que se $CN(x)$ é a quantidade de números de Carmichael menores do que x então

$$CN(x) \geq x^{2/7}$$

para x suficientemente grande, o que implica na existência de infinitos números de Carmichael.

Podemos refinar o conceito de pseudoprimo para definir *pseudoprimos fortes* na base a. Para definir quando n é um pseudoprimo forte na base a inicialmente escrevemos $n-1 = 2^k \cdot b$, com b ímpar. Se $n > 2$ é primo deve existir um menor valor de j para o qual $(a^b)^{2^j} \equiv 1 \pmod{n}$ (observe que por Fermat $(a^b)^{2^k} \equiv 1 \pmod{n}$). Se $j = 0$ isto significa que $a^b \equiv 1 \pmod{n}$; caso contrário temos $(a^b)^{2^{j-1}} \equiv -1 \pmod{n}$ já que -1 é o único valor de x diferente de 1 (módulo n) para o qual $x^2 \equiv 1 \pmod{n}$. Assim, dizemos que n composto ímpar é um pseudoprimo forte na base a se *ou* $a^b \equiv 1 \pmod{n}$ *ou* existe $j' < k$ com $(a^b)^{2^{j'}} \equiv -1 \pmod{n}$. Claramente todo pseudoprimo forte na base a é um pseudoprimo na base a mas pseudoprimos fortes são mais raros do que pseudoprimos.

Existem infinitos pseudoprimos fortes em qualquer base $a > 1$: Pomerance provou que, se $SP\pi_a(x)$ é o número de pseudoprimos fortes na base a menores ou iguais a x então

$$SP\pi_a(x) \geq e^{(\log x)^{5/14}}$$

para todo x suficientemente grande (ver [7]). Não existem "números de Carmichael fortes": para todo número composto ímpar n existe $0 < a < n$ com $\text{mdc}(a, n) = 1$ e tal que n *não* é um pseudoprimo forte na base a. Melhor ainda, os valores de a que servem de testemunha para a não-primalidade de n são sempre relativamente frequentes. Mais precisamente, seja

$$\alpha(n) = \frac{1}{\varphi(n)} \{a \mid 0 < a < n, n \text{ é um pseudoprimo forte na base } a\}.$$

Então para todo número composto ímpar $n > 9$ temos $\alpha(n) \leq 1/4$. A igualdade vale exatamente para os compostos n das seguintes formas:

$$n = p_1 p_2, \quad p_1, p_2 \text{ primos}, \quad p_1 \equiv 3 \pmod{4}, \quad p_2 = 2p_1 - 1;$$
$$n = p_1 p_2 p_3, \quad p_1, p_2, p_3 \text{ primos}, \; p_i \equiv 3 \pmod{4}, \; n \text{ número de Carmichael}.$$

Os menores exemplos de números compostos n da primeira forma para a qual $\alpha(n) = 1/4$ são $n = 15 = 3 \cdot 5$, $n = 91 = 7 \cdot 13$ e $n = 703 = 19 \cdot 37$. O menor exemplo de número composto da segunda forma é $n = 8911 = 7 \cdot 19 \cdot 67$. Sabe-se que existem menos do que $CN^{(1/2+\epsilon)}$ números compostos n destas formas menores do que N; conjectura-se que o número de compostos n da primeira forma seja maior do que $CN^{(1/2-\epsilon)}$. Na maioria dos casos $\alpha(n)$ é muito menor.

O resultado acima serve de base para o *algoritmo Miller-Rabin*, um exemplo de teste de primalidade *probabilístico*. Dado n, tomamos t valores de a ao acaso no

intervalo $1 < a < n$ e verificamos para cada a se n passa no teste de primalidade na base a. Se n for ímpar composto, a probabilidade de que um dado a acuse a não-primalidade de a é maior do que $3/4$ (pelo teorema); assim, a probabilidade de que n escape a t testes é menor do que 4^{-t}. Este tipo de teste é extremamente útil em aplicações (como em criptografia) onde é importante criar primos relativamente grandes mas não existe a preocupação com demonstrações ou com perfeição absoluta.

Em outros contextos estamos interessados em *testes determinísticos*: dado um inteiro positivo n, queremos decidir, com certeza e em tempo curto (i.e., polinomial no número de dígitos), se n é primo ou composto.

Existem vários testes determinísticos para formas especiais de n. Por exemplo, para números de Mersenne existe o critério de Lucas-Lehmer: dado um primo $p > 2$, queremos decidir se $n = 2^p - 1$ é primo. Sejam $S_0 = 4$, $S_1 = 4^2 - 2 = 14$, ..., $S_{k+1} = S_k^2 - 2$. Temos que n é primo se e somente se S_{p-2} é múltiplo de $2^p - 1$. Esta seqüência cresce muito rápido, mas basta fazer as contas módulo $2^p - 1$ (ver [3], [5]). O seguinte fato crucial para a demonstração pode ser verificado pelo leitor: para todo $k \geq 0$,

$$S_k = (2 + \sqrt{3})^{2^k} + (2 - \sqrt{3})^{2^k}.$$

Podemos modificar o algoritmo de Miller-Rabin para torná-lo determinístico testando todos os valores de a em um intervalo suficientemente grande: uma famosa generalização da hipótese de Riemann implica que o intervalo de 1 até $2(\log n)^2$ já é grande o bastante ([2]). Este algoritmo é rápido e geral, mas infelizmente depende de uma conjectura. Permaneceu em aberto por muito tempo se existe um critério determinístico, rápido e geral. Este problema for resolvido por Agrawal, Kayal e Saxena que criaram o algoritmo abaixo (veja [1], [3]). Aqui φ é a função de Euler: $\varphi(k)$ é o número de inteiros j entre 1 e k com $\mathrm{mdc}(j, k) = 1$; $\mathrm{ord}_r N$ denota o menor inteiro positivo k para o qual $N^k \equiv 1 \pmod{r}$.

1. Entrada $N > 1$.
2. Se $N = a^b$ com $b > 1$, retorna COMPOSTO.
3. Encontrar o menor r tal que $\mathrm{ord}_r N > \frac{1}{2}(\log N)^2$.
4. Se $\mathrm{mdc}(a, N) > 1$ para algum primo $a \leq r$, retorna COMPOSTO.
5. Se $\sqrt{N} < r$, retorna PRIMO.
6. Para $a = 1$ até $\lfloor \sqrt{\varphi(r)/2} \log_2 N \rfloor$ faça: Se $(x + a)^N \not\equiv x^N + a \pmod{x^r - 1, N}$, retorna COMPOSTO;
7. Retorna PRIMO.

A outra pergunta proposta por Gauß permanece em aberto: se existe algoritmo rápido para *fatorar* inteiros. Um tal algoritmo teria grande relevância prática, comprometendo as formas mais populares de criptografia. Existe ainda a possibilidade de que não exista um algoritmo rápido, mas que ainda assim exista uma máquina (no sentido físico) capaz de fatorar inteiros rapidamente. De fato, a mecânica quântica parece permitir a construção de um *computador quântico* e Peter Shor encontrou um "algoritmo" que permite a um computador quântico fatorar inteiros em tempo polinomial [8].

Referências

[1] M. Agrawal, N. Kayal e N. Saxena, *PRIMES is in P*, Ann. of Math. (2) 160 (2004), no. 2, 781–793.

[2] E. Bach, *Explicit bounds for primality testing and related problems*, Math. of Comp. 55 (1990), 355–380.

[3] F. Brochero, C. G. Moreira, N. C. Saldanha, E. Tengan, *Teoria dos números: um passeio com primos e outros números familiares pelo mundo inteiro*, Projeto Euclides, IMPA, 2010.

[4] M. Cipolla, *Sui numeri composti P, che verificano la congruenza di Fermat* $a^{P-1} \equiv 1 \pmod{P}$, Annali di Matematica (3) 9, 1904, 139–160.

[5] D. H. Lehmer, *An extended theory of Lucas' functions*, Ann. Math. 31 (1930), 419–448. Reimpresso em *Selected Papers*, (ed. D. McCarthy), vol 1, Ch. Babbage Res. Center, St. Pierre, Manitoba, Canada, 11–48 (1981).

[6] W. H. Mills, *A prime representing function*, Bull. Amer. Math. Soc., 53 (1947), 604.

[7] C. Pomerance, *A new lower bound for the pseudoprimes counting function*, Illinois J. Math, 26, 1982, 4–9.

[8] P. W. Shor, *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, SIAM J. Comput. 26 (5), 1484-1509 (1997).