TESTES DE PRIMALIDADE

DIOGO POÇAS

Resumo. Embora o conceito de número primo seja simples e já muito antigo, o impacto que ainda tem é extraordinário. Actualmente os números primos são usados em várias áreas, como na criptografia de chave pública (onde se toma partido da dificuldade existente em factorizar um número grande), na geração de códigos com dígitos de controlo (como o código ISBN), e em geradores de números pseudoaleatórios. Dado um número n, será n primo? Haverá uma maneira "eficiente" de responder a essa pergunta? Neste artigo vamos investigar dois testes de primalidade, o teste de Miller–Rabin e o teste AKS. Vamos apresentar os respectivos algoritmos em pseudocódigo e discutir a sua eficiência.

1. Introdução

Um número natural é primo quando tem apenas dois divisores distintos: ele próprio e 1. Os primeiros primos são 2, 3, 5, 7, 11, 13, ...

Uma vez que os números primos têm imensas aplicações, o problema de decidir sobre a primalidade de um número com vários dígitos é bastante importante. O nosso objectivo é então o de encontrar algoritmos eficientes que decidam se um número é primo ou não. Para medir a eficiência de um algoritmo, temos de calcular o número de operações necessárias para que este termine, em função do argumento recebido. Diz-se que um algoritmo corre em tempo polinomial se lhe puder ser associado um polinómio que majore o número de operações que o algoritmo necessita para terminar, quando avaliado no tamanho do respectivo argumento.

O tamanho de um argumento é o número de dígitos necessários para o representar na base 2. Assim, o inteiro $2 = 10_2$ tem tamanho 2, o inteiro $10 = 1010_2$ tem tamanho 4, e um número n tem tamanho $\log_2 n$ ou, como passaremos a escrever, $\log n$. Como regra geral, o tamanho de um argumento é a medida que se utiliza em questões de eficiência—faz mais sentido comparar algoritmos em diferentes tamanhos de argumentos do que em diferentes argumentos.

2. Divisão sucessiva

Antes de apresentarmos os testes mais complicados, começamos por estudar a eficiência do teste de primalidade que aprendemos na escola, designado teste de divisão sucessiva.

A ideia nasce da própria definição de número primo. Para sabermos se um número é primo, só temos de testar a divisibilidade desse número pelos naturais que lhe são inferiores. Por exemplo, para verificar se 35 é primo, basta-nos verificar se 35 é divisível por algum número de 2 a 34. Na verdade, o número de passos pode ser reduzido, já que um número composto n tem sempre um divisor menor ou igual

Publicado em *Números, cirurgias e nós de gravata: 10 anos de Seminário Diagonal no IST*, J. P. Boavida, R. P. Carpentier, L. Cruz-Filipe, P. S. Gonçalves, E. Grifo, D. Henriques, A. R. Pires (editores), IST Press, 2012.

a $\sqrt{n}$.[1] Ao efectuar o teste da divisão sucessiva a 35, percorreríamos os naturais 2, 3, 4 e 5, e iríamos descobrir que 35 é divisível por 5, pelo que não é primo.

Assim sendo, podemos criar um algoritmo que realiza o teste da divisão sucessiva:

Recebe um inteiro $n \geq 2$ de tamanho $\log n$;
Devolve "verdadeiro" se n é primo, "falso" caso contrário.

1. Inicializar $i = 2$;
2. Repetir até $i > \sqrt{n}$:
   2a. Se i divide n, devolve "falso" e termina;
   2b. $i = i + 1$;
3. Devolve "verdadeiro" e termina.

Para avaliar a eficiência deste algoritmo vamos aproximar o número de operações elementares[2] necessárias para o teste terminar. Neste caso as operações a considerar são as verificações de divisibilidade (passo 2a). O número de operações a realizar depende do número de vezes que executamos o ciclo no passo 2. Na pior das hipóteses, quando n é primo, o programa corre todos os naturais de 2 a $\sqrt{n}$ antes de responder "verdadeiro". Logo, o número de operações que este algoritmo leva a terminar é da ordem de $\sqrt{n}$. Observe-se que tamanho do argumento era $d = \log n$ e logo $n = 2^d$.

Concluímos então que o teste de divisão sucessiva, que todos aprendemos na escola, tem complexidade exponencial da ordem de $2^{d/2}$ para um argumento de tamanho d. O número de operações a realizar cresce rapidamente com o tamanho do argumento, e portanto trata-se de um teste pouco eficiente. Existem algumas variantes deste método (por exemplo, testar a divisibilidade apenas para 2 e os inteiros ímpares), mas todas têm a mesma ordem de complexidade.

3. Teste de Miller–Rabin

O teste de Miller–Rabin é um teste probabilístico, isto é, não determina com toda a certeza se um número é primo. No entanto, é bastante rápido e tem a vantagem de o podermos realizar várias vezes para aumentar o grau de certeza. Este teste foi inventado em 1980.

Todos os testes probabilísticos de primalidade partem de uma proposição da forma "Se p é primo, então p verifica a propriedade X", onde X pode ser verificada em tempo polinomial. Assim, o teste probabilístico consiste em verificar se p tem a propriedade X. Dependendo do resultado, se X não for satisfeita podemos dizer que p é composto. Se X for satisfeita, não podemos afirmar se p é primo ou composto. Esta é a desvantagem dos testes probabilísticos.

Não podemos deixar de frisar que, nos algoritmos desta natureza, uma resposta de "composto" é sempre definitiva, ou seja, podemos ter falsos positivos mas não falsos negativos. É explorando esta assimetria que podemos majorar a probabilidade de erro por um valor tão pequeno quanto se queira, como veremos mais à frente.

Dizemos que dois inteiros a e b são congruentes módulo p, o que se representa por $a \equiv b \mod p$, se p divide $b - a$. Isto é equivalente a afirmar que a e b têm o mesmo resto de divisão por p.

Seja n um natural ímpar e u, k naturais, com u ímpar e $n - 1 = 2^k u$. Para um dado a de 1 a $n - 1$, dizemos que n satisfaz a *propriedade de Miller–Rabin em a*

[1] Se $n = pq$ e tanto p como q são maiores que $\sqrt{n}$, então $n > \sqrt{n}\sqrt{n} = n$, o que é um absurdo.

[2] Uma operação elementar é uma instrução considerada simples, executada várias vezes durante o algoritmo. Neste artigo consideramos como operações elementares as divisões, multiplicações e congruências (cálculo de restos).

se uma das seguintes condições é satisfeita: $a^u \equiv 1 \mod n$, ou existe um natural $\ell < k$ com $a^{2^\ell u} \equiv -1 \mod n$.

Um teorema importante é que todos os primos ímpares satisfazem a propriedade de Miller–Rabin para qualquer valor de a. Isto segue de dois resultados, dos quais o primeiro é o Pequeno Teorema de Fermat. Fermat provou que, se um número p é primo e a é um natural de 1 a $p-1$, então $a^{p-1} \equiv 1 \mod p$. Este resultado é de demonstração simples.

Outro ingrediente para o Teste de Miller–Rabin é o seguinte: Suponhamos que p é primo e x é um natural tal que $x^2 \equiv 1 \mod p$. Então p divide $x^2-1 = (x+1)(x-1)$, pelo que p divide $x+1$ ou p divide $x-1$. Em suma, temos que

$$x^2 \equiv 1 \mod p \qquad \text{implica} \qquad x \equiv 1 \text{ ou } x \equiv -1 \mod p.$$

Podemos aplicar o critério à equação $a^{p-1} \equiv 1 \mod p$, tomando[3] $x = a^{(p-1)/2}$. Temos que $a^{(p-1)/2} \equiv \pm 1 \mod p$. Para além disso, este processo pode ser repetido se $a^{(p-1)/2} \equiv 1 \mod p$ e $(p-1)/2$ for par. É imediato então obter que qualquer primo ímpar satisfaz a propriedade de Miller–Rabin.

Veremos ainda que a propriedade de Miller–Rabin tem a vantagem de ser verificável em tempo polinomial.

O teste de Miller–Rabin consiste em, dado um número ímpar, verificar se satisfaz ou não a propriedade de Miller–Rabin para um dado a. Se não satisfizer, então não é primo; Se satisfizer, nada podemos concluir. Em testes probabilísticos, é frequente usar o termo *pseudoprimo*. Este termo define um número que, não sendo primo, partilha uma certa propriedade com os números primos.

Por exemplo, suponhamos que queríamos aplicar o teste de Miller–Rabin para $n = 121$, $a = 3$. Pondo em evidência a maior potência de 2 em $121 - 1$, obtemos $120 = 2^3 \times 15$, pelo que temos de calcular os valores de $3^{15}, 3^{30}, 3^{60}$ e $3^{120} \mod 121$. Obtemos

$$3^{15} \equiv 1 \mod 121, \quad 3^{30} \equiv 1 \mod 121, \quad 3^{60} \equiv 1 \mod 121, \quad 3^{120} \equiv 1 \mod 121,$$

e portanto 121 satisfaz a propriedade de Miller–Rabin. No entanto, 121 não é primo, pois $121 = 11^2$, o que mostra que o teste de Miller–Rabin não é 100% fiável, isto é, existem números que não são primos mas que não são denunciados pelo teste de Miller–Rabin. A estes números damos o nome de *pseudoprimos fortes*. Por exemplo, 121 é um pseudoprimo forte para o natural $a = 3$. Outro exemplo é 2047, que é um pseudoprimo forte para o natural $a = 2$.

Em seguida, mostramos um algoritmo para o teste de Miller–Rabin:

> *Recebe* inteiros positivos n (ímpar, de tamanho $\log n$) e $a < n$;
> *Devolve* "composto" se n é detectado como composto, "possível primo" caso contrário.
>
> 1. Obter a factorização $n - 1 = 2^k u$, com u ímpar;
> 2. Calcular $s = a^u \bmod n$;
> 3. Se $s = 1$ ou $s = n - 1$ devolve "possível primo" e termina;
> 4. Repetir $k - 1$ vezes:
> 4a. $s = s^2 \bmod n$;
> 4b. Se $s = 1$, devolve "composto" e termina;
> 4c. Se $s = n - 1$, devolve "possível primo" e termina;
> 5. Devolve "composto" e termina.

Vamos primeiro analisar a eficiência computacional deste método, contando o número de operações elementares, que neste caso correspondem às multiplicações e divisões. Encontrar a factorização $n - 1 = 2^k u$ termina em tempo polinomial,

[3]Note-se que, se p é primo e é maior que 2, então $p - 1$ é par.

pois basta dividir $p-1$ por 2 sucessivamente até obter um número ímpar (termina, no máximo, em $\log n$ passos[4]). O cálculo de $a^u \bmod n$ também termina em tempo $\log n$. Isto passa-se pois conhece-se um método muito rápido para calcular potências, conhecido em inglês como *ladder exponentiation*. O ciclo que aparece no passo 4 é efectuado no máximo $k-1$ vezes, e em cada passo são efectuadas duas operações elementares para o cálculo de $s^2 \bmod n$. Como k é da ordem de $\log n$, o tempo de execução do passo 4 também é da ordem de $\log n$. Somando os tempos de execução e fazendo a substituição $n = 2^d$, concluímos que o teste de Miller–Rabin tem complexidade linear, da ordem de d, para um argumento de tamanho d.

Ou seja, o teste de Miller–Rabin é eficiente. No entanto, este teste falha em catalogar todos os números e temos de lidar com a existência de pseudoprimos. Vimos que existem números que passam o teste de Miller–Rabin para um certo a, sem no entanto serem primos. Mas é preciso notar que um número pseudoprimo para um certo a pode não o ser para um a' diferente. Na verdade, Monier e Rabin demonstraram que, para um número composto n, a proporção de naturais de 2 a $n-1$ para as quais n é pseudoprimo é inferior a 25%. Por outras palavras, se um número n for composto e escolhermos aleatoriamente um natural para inicializar o teste de Miller–Rabin, o teste 'desmascara' este número em 75% dos casos.

É aqui que reside a beleza deste teste probabilístico. E se executarmos o teste de Miller–Rabin duas vezes, para dois valores de a diferentes? Se o número for composto, as hipóteses de passar o teste de Miller–Rabin são inferiores a uma em dezasseis. Se executarmos o teste mais algumas vezes, esta probabilidade decresce geometricamente. Executando o teste um grande número de vezes, a probabilidade de n não ser detectado torna-se tão reduzida quanto queiramos, sem no entanto perdermos a eficiência do teste, que continua a ser polinomial.

Os algoritmos probabilísticos desta forma são muito utilizados na prática, mesmo em situações críticas, pois a sua probabilidade de erro desce rapidamente com o número de execuções que realizamos. Na verdade, o teste de Miller–Rabin é mais usado que o teste AKS que a seguir apresentaremos. Por exemplo, suponhamos que executamos o teste de Miller–Rabin cem vezes para um inteiro n, e em todas elas obtemos a resposta 'possível primo'. A probabilidade de n ser composto é inferior a 1 em 4^{100}, o que é menor que a probabilidade de ganhar o Euromilhões sete vezes consecutivas (um fenómeno bastante improvável, como sabemos).

No entanto, o teste de Miller–Rabin não é um teste determinístico e, em rigor, não podemos afirmar que um número é primo porque resistiu ao teste várias vezes (até pode não o ser!). Não deixa de ser, contudo, uma poderosa arma para estudar a primalidade.

4. Teste AKS

O teste AKS é a resposta polinomial ao problema da primalidade. Foi inventado em 2002 por três matemáticos indianos: Agrawal, Kayan e Saxena. Começamos por observar o seguinte resultado: $n \geq 2$ é primo se e só se, para qualquer a,

$$(X+a)^n \equiv X^n + a \mod n.$$

É preciso notar que a congruência é feita nos polinómios de uma variável com coeficientes inteiros, e avaliada coeficiente a coeficiente. Podemos pensar nos polinómios como expressões da forma

$$a_n X^n + a_{n-1} X^{n-1} + \cdots + a_1 X + a_0, \qquad \text{com } a_i \in \mathbb{Z}.$$

[4]Recorde-se que a complexidade é definida com base no tamanho do argumento, que neste caso é $\log n$, e portanto tempo polinomial significa uma majoração por um polinómio em $\log n$.

A propriedade acima é de facto equivalente à primalidade de n, mas não pode ser verificada em tempo polinomial. No entanto, podemos enfraquecer essa propriedade, dizendo que, se $n \geq 2$ é primo e fixando um r, então existem polinómios f, g com

$$(X+a)^n = X^n + a + nf(X) + g(X^r - 1),$$

o que se costuma denotar por

$$(X+a)^n \equiv X^n + a \mod n, X^r - 1.$$

Se, para cada n, o valor de r for suficientemente pequeno (ou seja, majorado por um polinómio em $\log n$ fixo à partida), então esta propriedade pode ser verificada em tempo polinomial. No entanto, a propriedade deixa de ser equivalente à asserção 'n é primo'.

Um resultado fundamental para utilizar o teste AKS é que, para cada n, existe um valor de r suficientemente pequeno tal que, se a congruência se verifica para um conjunto de inteiros $a \in A$ suficientemente pequeno, então n é primo. O resultado concreto afirma então que, se n e r forem inteiros satisfazendo as seguintes condições:

- $n \geq 3$;
- $r < n$ e r é primo;
- para todo o a de 2 a r, $a \nmid n$;
- a ordem[5] de n em $(\mathbb{Z}/r\mathbb{Z})^\times$ é maior que $(2 \log n)^2$;
- para todo o a com $1 \leq a \leq 2\sqrt{r}\log n$, tem-se $(X+a)^n \equiv X^n + a \mod n, X^r - 1$;

então podemos concluir que n é primo ou é potência de um inteiro.

Todas as propriedades referidas no teorema podem ser verificadas em tempo polinomial. Pode ser demonstrado que existe um r nas condições do teorema e que é inferior a $20\lceil \log n \rceil^5$. Assim sendo, podemos elaborar o teste AKS como se segue:

> *Recebe* um inteiro $n \geq 2$ de tamanho $\log n$;
> *Devolve* "verdadeiro" se n é primo, "falso" caso contrário.
> 1. Se n for a potência de um inteiro, devolve "falso" e termina;
> 2. Encontra o menor primo r para o qual a ordem de n em $(\mathbb{Z}/r\mathbb{Z})^\times$ é maior que $(2\log n)^2$;
> 3. Se n for divisível por algum a com $2 \leq a \leq r$, devolve "falso" e termina;
> 4. Se $(X+a)^n \not\equiv X^n + a \mod n, X^r - 1$ para algum a com $1 \leq a \leq 2\sqrt{r}\log n$, devolve "falso" e termina;
> 5. Devolve "verdadeiro" e termina.

Vamos então estimar o número de operações para efectuar o teste AKS, que está dividido em três partes.

Na primeira parte procuramos saber se n é a potência de um inteiro, isto é, se $n = m^b$. Claramente que, se $m \geq 2$, então b é no máximo $\log n$. Para cada valor de b, tentamos achar o maior valor de m para o qual $m^b \leq n$. Cada exponenciação requer cerca de $\log_b n$ passos, e podemos encontrar o maior valor de m em $\log n$ passos. Logo o teste de potência leva na pior das hipóteses $(\log n)^3$ passos.

Na segunda parte procuramos um r nas condições do teorema. Para cada r, temos de calcular os valores $n, n^2, \ldots, n^{\lfloor 4(\log n)^2 \rfloor}$ que requerem cerca de $4(\log n)^2$ multiplicações no pior caso. Como há um r nas condições do teste e da ordem de $(\log n)^5$, este passo requer um número de operações da ordem de $(\log n)^7$. Para

[5] A ordem de n em $(\mathbb{Z}/r\mathbb{Z})^\times$ designa o menor natural k tal que $n^k \equiv 1 \pmod r$.

verificar, em cada passo, se r é primo, poderíamos ser tentados a aplicar recursivamente o algoritmo AKS, mas na verdade costuma-se utilizar um método de crivagem, isto é, ir criando uma tabela de números primos, de modo a que verificar se r é primo termine em tempo constante. A criação da tabela de primos requer cerca de $r \log\log n \simeq (\log n)^5 \log\log n$ operações no total, e logo a complexidade do algoritmo não aumenta.

Falta apenas o teste das congruências. Cada congruência exige um número de operações da ordem de $r^2 \log n$, por uma variante para polinómios da *ladder exponentiation*.[6] Temos de testar um número de congruências da ordem de $\sqrt{r} \log n$, ou seja, $(\log n)^{3.5}$. Logo este passo requer um número de operações da ordem de $(\log n)^{14.5}$.

Juntando todos estes valores e substituindo $\log n = d$, obtemos o resultado final: o teste AKS tem complexidade polinomial, da ordem de $d^{14.5}$.

Convém notar que o valor do expoente em d depende da implementação escolhida. Na verdade, com o passar dos anos, foram-se descobrindo variantes mais eficientes do teste. Actualmente consegue-se verificar a primalidade em tempo $d^{6+\varepsilon}$, para qualquer valor de $\varepsilon > 0$.

5. Conclusão e Bibliografia

Neste artigo vimos vários teste de primalidade. Como pudemos observar, os números primos foram abordados por uma grande quantidade de matemáticos ao longo dos tempos. Muitas tentativas de responder à questão da primalidade foram feitas. Até ao início do século, não se sabia se o problema da primalidade poderia ser resolvido em tempo polinomial. Mas em 2002 surgiu a resposta a esta questão, colocando o problema da primalidade na classe P, que é a classe dos problemas que podem ser resolvidos em tempo polinomial. Esta classe tem grande importância na teoria da computação, pois representa todos os problemas que são 'facilmente' resolvidos por um computador.

Para além dos testes mencionados, poderíamos apontar vários outros testes de primalidade conhecidos, tais como o teste de Solovay–Strassen, o crivo de Eratóstenes, o teste de Pepin e muitos outros. Estes testes são descritos com pormenor em vários outros sítios, quer na Internet, quer em livros e artigos publicados sobre números primos. A bibliografia no final deste artigo inclui algumas fontes para encontrar estes e outros testes: [1] é o livro no qual se baseia a maior parte do trabalho, embora seja um pouco avançado; [2] é o melhor ponto de partida para principiantes, e utilizei esta referência para perceber algumas partes que não estavam tão claras no livro anterior; [3] é um artigo sobre o teste AKS e a melhor fonte que encontrei para tratar essa parte do trabalho—já se trata de material mais avançado do que nas outras duas fontes.

Espero que este artigo tenha sido esclarecedor, e que sirva para mostrar ao leitor que os números primos são um assunto fascinante e muito extenso, onde há muitos resultados, muitas conjecturas em aberto e muitas respostas à mesma pergunta. A Matemática seria certamente muito pobre sem os números primos.

Antes de dar por terminado este artigo, gostaria de agradecer à Fundação Calouste Gulbenkian, responsável pelo programa Novos Talentos em Matemática, no âmbito do qual foi realizado o meu trabalho, e ao meu orientador, o professor Carlos Caleiro, que tiveram um papel importante na concepção do trabalho que originou este artigo.

[6] Cada multiplicação de polinómios pode ser feita com um número de operações elementares da ordem de r^2.

Referências

[1] Richard Crandall e Carl Pomerance, *Prime Numbers: A Computational Perspective*, Springer, New York, 2005.

[2] Martin Dietzfelbinger, *Primality Testing in Polynomial Time: From Randomized Algorithms to "PRIMES is in P"*, Springer-Verlag, Berlin, 2004.

[3] Andrew Granville, *It is easy to determine if a given number is prime*, Bull. Amer. Math. Soc., vol. 42, 2004, pp. 3–38.