Testy dla liczb pierwszych o czasie wielomianowym

Ian STEWART,
Wielka Brytania

W roku 1801 Carl Friedrich Gauss stwierdził (w 329. artykule *Disquisitiones Arithmeticae*), że „problemy odróżniania liczb pierwszych od złożonych i rozkład tych ostatnich na czynniki pierwsze są uznane za jedne z najważniejszych i najbardziej użytecznych w arytmetyce". Do roku 2001 te same problemy, dzięki wynalazkowi kryptosystemów opartych na faktoryzacji liczb pierwszych, okazały się kluczowe dla rynku w Internecie. Wydaje się zadziwiające, że coś nowego da się powiedzieć o parze tak starożytnych i tak długo badanych problemów. Tymczasem nowe odkrycia piętrzą się! Najnowsze z nich – algorytm, który w czasie wielomianowym rozstrzyga, czy dana liczba jest pierwsza – ma wyjątkowe znaczenie. Autorami tego algorytmu są: Manindra Agrawal, Neeraj Kayal i Nintin Saxena z Politechniki w Kanpur w Indiach.

Powszechnie znany jest fakt, że z dwóch problemów wymienionych przez Gaussa sprawdzanie, czy dana liczba jest pierwsza, jest znacznie łatwiejsze niż rozkład liczby złożonej na czynniki. Najbardziej naturalnym testem, czy dana liczba jest pierwsza, jest próbowanie podzielności przez kolejne możliwe czynniki, co daje wrażenie, że testowanie, czy liczba jest pierwsza, musi opierać się na faktoryzacji. Wydaje się przy tym też, że liczba operacji matematycznych, potrzebnych do przetestowania liczby, wynosi około $10^{n/2}$ dla n-cyfrowej liczby N. Oznacza to, że potrzeba około $\sqrt{N}$ kroków do przetestowania wszystkich możliwych dzielników liczby N, gdyż tylko liczba 2 i liczby (nieparzyste) aż do $\sqrt{N}$ muszą być brane pod uwagę. Ale oba te wrażenia są błędne.

Po pierwsze, jest wiele testów, które dają odpowiedź „tak" lub „nie", bez znajdowania jakiegokolwiek dzielnika, jeśli odpowiedź brzmi „nie". Klasyczny taki test opiera się na twierdzeniu Wilsona: N jest liczbą pierwszą wtedy i tylko wtedy, gdy N dzieli liczbę $(N-1)!+1$. Test ten nie jest jednak zbyt praktyczny: jeśli N ma 100 cyfr, to obliczenia wymagają pomnożenia 10^{100} liczb. Istnieją jednak także wydajne testy. Wiele z nich wywodzi się z małego twierdzenia Fermata: jeśli p jest liczbą pierwszą i a nie jest wielokrotnością p, to

$$a^{p-1} \equiv 1 \mod p. \tag{1}$$

Jest szybki sposób na to, by obliczyć $a^{p-1} \bmod p$, oparty na powtarzaniu podnoszenia do kwadratu i używaniu dwójkowego rozwinięcia liczby $p-1$. Na przykład, aby obliczyć $2^{22} \bmod 23$, obliczamy (zawsze modulo 23) liczby

$$2^1 \equiv 2, \quad 2^2 \equiv 4, \quad 2^4 \equiv 16 \equiv -7, \quad 2^8 \equiv 49 \equiv 3, \quad 2^{16} \equiv 9.$$

Mamy zatem

$$2^{22} \equiv 2^{16} \cdot 2^4 \cdot 2^2 \equiv 9 \cdot (-7) \cdot 4 \equiv (-17) \cdot 4 \equiv 1,$$

jak tego zresztą oczekiwaliśmy, skoro 23 jest liczbą pierwszą.

Pozytywny wynik testu Fermata jest warunkiem koniecznym tego, by dana liczba była pierwsza, ale nie jest wystarczające. Wiele liczb p, znanych jako liczby Carmichaela, spełnia (1) dla wszystkich a niepodzielnych przez p. Najmniejsza z nich to $p = 561 = 3 \cdot 11 \cdot 17$. Alford, Granville i Pomerance udowodnili, że jest ich nieskończenie wiele. Jednak bardziej wyrafinowane warianty testu Fermata naprawdę weryfikują, czy dana liczba jest pierwsza. Do niedawna najbardziej wydajny był test APR (Adelman–Pomerance–Rumely), którego czas działania był rzędu $(\log N)^{\log\log\log N}$.

Z grubsza biorąc, są dwa rodzaje algorytmów. Wydajne są klasy P, co oznacza, że działają w czasie wielomianowym. Oznacza to, iż dla danych na wejściu N bitów uzyskują odpowiedź po wykonaniu co najwyżej kN^r kroków, gdzie k i r są stałymi. Nieefektywne algorytmy są klasy nie-P i wykonywane są w czasie gorszym niż wielomianowy, przy czym bardzo niewydajne pracują w czasie wykładniczym kr^N lub dłuższym.

Do 2001 roku było wiadomo, że sprawdzanie, czy liczba jest pierwsza może być wykonane w czasie niewiele dłuższym od wielomianowego, tak że w przypadku

Rozwiązanie zadania F 619.
Wysokie ciśnienie lepiej zniesie gumowy balonik. Jest on elastyczny, więc powiększające się ciśnienie na zewnątrz spowoduje tylko niewielką zmianę objętości wody w środku, ale siły działające na jego powłokę będą się równoważyć. Teoretycznie może on zanurzyć się na dowolnie dużą głębokość.

W wypadku puszki ciśnienie wewnątrz pozostaje podczas zanurzania bez zmian, a ciśnienie na zewnątrz rośnie. Póki siły sprężystości stali równoważą powstałe naprężenie, puszka utrzyma swój kształt, jednak w pewnym momencie zostanie ona zgnieciona.

liczby 200-cyfrowej można to było zupełnie swobodnie wykonać na zwykłym, dobrym komputerze. Z drugiej strony, wydawało się, że rozłożenie na czynniki pierwsze liczby 200-cyfrowej nawet przy użyciu najlepszych metod zajęłoby czas, jaki upłynął od zarania ludzkiej cywilizacji. Jednak nie było i nadal nie ma teoretycznej gwarancji, że faktoryzacja liczby to problem klasy nie-P. Choć więc sądzi się powszechnie, że nie istnieje żaden algorytm faktoryzacji działający w czasie wielomianowym, to może jest inaczej, a my po prostu nie jesteśmy wystarczająco sprytni, żeby go odkryć. Pod koniec roku 2001 nie istniał przecież żaden algorytm sprawdzania, czy dana liczba jest pierwsza, działający w czasie wielomianowym, a tymczasem w 2002 roku już się taki pojawił wraz z wkroczeniem do akcji wspomnianych matematyków indyjskich. Wymyślili oni mianowicie nowy wariant testu Fermata. Zamiast pracować z liczbami, zajęli się wielomianami jednej zmiennej. Punktem wyjścia ich rozważań był fakt, że gdy x to niewiadoma, a p to liczba pierwsza, która nie dzieli liczby całkowitej a, to

$$(x-a)^p \equiv (x^p - a) \mod p.$$

Testowanie tej kongruencji nie prowadzi jeszcze do wielomianowego czasu działania algorytmu. Zamiast tego algorytm testuje powyższą nierówność modulo różne proste wielomiany, które dla wygody przyjmuje się w postaci $x^r - 1$ dla pewnego całkowitego r. Jedna iteracja algorytmu sprawdza, czy zachodzą jednocześnie kongruencje

$$\text{(2)} \qquad \begin{cases} (x-a)^p \equiv (x^p - a) \mod x^r - 1 \\ (x-a)^p \equiv (x^p - a) \mod p\,. \end{cases}$$

Kongruencje są spełnione dla wszystkich liczb pierwszych p. Dla danego a i r także przez niektóre liczby złożone, ale żadna liczba złożona nie spełnia powyższej kongruencji dla wszystkich a i r. Trik polega więc na tym, by utrzymać złożoność obliczeniową algorytmu, zapewniając jednocześnie, że żadna liczba złożona nie prześlizgnie się przez zastawioną sieć.

Test (2) może być przeprowadzony w czasie rzędu $r^2 \log^3 p$, przy użyciu powtarzalnego podnoszenia do kwadratu lub w czasie $r \log^2 p$, gdy użyjemy szybkiego mnożenia Fouriera. Jest to zatem czas wielomianowy. Zaczynamy od wybrania odpowiedniego r; powinna to być liczba pierwsza rzędu $\log^6 p$, a $r-1$ musi mieć dzielnik pierwszy wielkości co najmniej $r^{1/2+d}$ dla pewnego $d > 0$. Takie r istnieje (Fouvry, Baker, Harman). Następnie algorytm sprawdza kongruencje (2) dla małej liczby kandydatów na a (ta mała liczba jest rzędu $r \log p$). I – jak dowiódł zespół hinduskich matematyków – algorytm ten rozstrzyga pierwszość liczby N w czasie rzędu $\log^{12} N$, co jest czasem wielomianowym (oszacowanie to zostało jeszcze poprawione).

Istnienie algorytmu rozstrzygającego w czasie wielomianowym to, czy dana liczba jest pierwsza, zmienia nasze myślenie o całym zagadnieniu. Ale metody tu rozwinięte zdają się nic nie podpowiadać, jak rozwiązać drugi z problemów Gaussa – faktoryzację. Czy istnieje algorytm klasy P znajdujący dzielniki liczby złożonej? Większość ekspertów myśli, że nie, ale teraz nie są już tego tak pewni, jak dotychczas.

Tłumaczył Witold SADOWSKI

Rozwiązanie zadania M 1060.
Niech F będzie punktem symetrycznym do punktu E względem prostej DB. Wówczas punkt F leży na odcinku AB i $AF = CE$. Zatem trójkąty AFD i CED są przystające. Ponieważ

$$\sphericalangle EPB + \sphericalangle EQB = 180^\circ,$$

więc punkty P, E, Q, B leżą na jednym okręgu. Analogicznie stwierdzamy, że punkty A, F, P, D leżą na jednym okręgu. Zatem

$$\sphericalangle APD = \sphericalangle AFD = \sphericalangle CED = \sphericalangle BEQ = \sphericalangle BPQ,$$

skąd wynika, że punkty A, P i Q leżą na jednej prostej.