reference frame

What has quantum mechanics to do with factoring?

N. David Mermin

N. David Mermin has been teaching quantum computation to students of computer science at Cornell University for the past six years. He is proud to say that if you Google "CS483," his CS483 lecture notes (in which what "can be shown" is shown in painstaking detail) come at or near the top of a very long list.

A quantum computer is a digital computer capable of exploiting quantum coherence among the physical two-state systems that store the binary arithmetic information.

To factor an integer is to find its (unique) expression as a product of prime numbers.

The most impressive, most important, and best-known thing a quantum computer can do is to factor with spectacular efficiency the product of two enormous prime numbers. *But what on earth can quantum mechanics have to do with factoring?*

This question bothered me for four years, from the time I heard about the discovery that a quantum computer was spectacularly good at factoring until I finally took the trouble to find out how it was done. The answer, you will be relieved—but, if you're like me, also a little disappointed—to learn, is that quantum mechanics has nothing at all directly to do with factoring. But it does have a lot to do with waves. Many important waves are periodic, so it is not very surprising that quantum mechanics might be useful in efficiently revealing features associated with periodicity.

Quantum mechanics is connected to factoring through periodicity. It turns out, for purely arithmetic reasons having nothing to do with quantum mechanics, that if we have an efficient way to find the period of a periodic function, then, as we shall see below, we can easily factor the product of two enormous prime numbers. And a quantum computer provides an extremely efficient way to find periods.

All of the above is of considerable practical importance, because the great difficulty in factoring such a product—where the two enormous prime numbers are typically each several hundred digits long—is the basis for the security of the most widely used encryption scheme (called RSA[1] encryption) for protecting private information sent over the internet. In 1994 Peter Shor discovered[2] that a quantum computer would be super efficient at period finding and thereby pose a potential threat to innumerable secrets. Whence the explosion of interest in developing quantum computation. The threat is only potential because no quantum computer capable of anything like serious period finding currently exists.

I suspect the emphasis has been put on factoring rather than period finding because factoring is more famously associated with RSA code breaking, although, as it happens, period finding can be used directly to crack the RSA code, without any need for a detour into factoring. Factoring is also a mathematical concept more familiar to the general public than period finding.

The focus on factoring has led to some spectacular misrepresentations of Shor's algorithm in what Einstein called "the secular press." For example, the *New York Times* science writer George Johnson says in his book on quantum computation, *A Shortcut Through Time* (Alfred A. Knopf, 2003), that when the algorithm has done its job, "the solutions—the factors of the number being analyzed—will all be in superposition." Elsewhere he says that a quantum computer can "try out all the possible factors simultaneously, in superposition, then collapse to reveal the answer." Neither of these statements bears the slightest resemblance to what the algorithm actually does.

A lesson in modular arithmetic

Such misinformation can give rise to a lot of confusion. The first step to enlightenment is to understand the purely arithmetic connection between factoring and period finding. Then you can forget all about factoring. The link is surprisingly simple, if you're acquainted with modular arithmetic and are willing to accept, as an empirical fact, that a procedure that *might* give you the factors, if repeated not terribly many times, almost certainly *will* give them.

In modular arithmetic, two integers are said to be equal (or "congruent") modulo a particular integer N (written $\equiv$) if they differ by a multiple of N. Modulo N the infinite set of integers wraps around a circle into the finite set $0, 1, 2, 3, \ldots, N-1$. Here, for example, are the powers of 3 modulo 5: $3^2 \equiv 4$, because $3^2 = 9 = 5 + 4$; $3^3 \equiv 2$, because $3^3 = 27 = 5 \times 5 + 2$; $3^4 \equiv 1$, because $3^4 = 81 = 16 \times 5 + 1$; and $3^5 \equiv 3$, because $3^5 = 243 = 48 \times 5 + 3$. After that it repeats: $3^6 \equiv 3^2$, $3^7 \equiv 3^3$, $3^8 \equiv 3^4$, and so on; 3^x modulo 5 is a periodic function of x with period 4. Starting at $x = 1$, it produces the sequence 342134213421 . . . . The number 4 has a different period modulo 5. Since $4^2 = 16 = 3 \times 5 + 1$, the sequence produced by 4 is 41414141 . . . . The period is 2.

Why should something as simple as modular arithmetic enable you to do something as hard as factoring the product of two enormous prime numbers? By itself, it can't. But if you have a really good period-finding machine—and Shor's discovery was that a quantum computer is just such an engine—then there is an easy way to learn the two primes using modular arithmetic. Here is what you do:

Pick an integer a at random. With overwhelming probability, a will not be a multiple of either of those two enormous primes. That being so, it is easy to show that *some* power of a must be equal to 1 modulo N. For modulo N there are only N distinct numbers, $0, 1, 2, \ldots, N-1$. So if you imagine a list of $N + 1$ different modulo-N powers of a, the list has to contain at least one pair of distinct powers of a, with $a^y \equiv a^x$, and $y > x$. This means that $a^y - a^x = a^x(a^{y-x} - 1)$ is a multiple of N.

Since a does not contain either prime factor of N, neither does a^x. So if the product of a^x with $a^{y-x} - 1$ is divisible by N, then $a^{y-x} - 1$ must all by itself be a multiple of N: $a^{y-x} \equiv 1$.

But if *some* nonzero power of a is equal to 1 modulo N, then there must be a *smallest* such power. If r is the smallest positive integer satisfying $a^r \equiv 1$, then the function $f(x) = a^x$ modulo N is periodic with period r. So if we have a

good period-finding machine—our quantum computer—then we can find r for any a. And with a little bit of luck (we'll return in a moment to just how much luck we need), knowing the period r actually enables us to factor N easily. We need two pieces of luck.

First, suppose we are lucky enough to have picked a random a whose period r is even: $r = 2m$. If $b \equiv a^m$, then $(b-1)(b+1) = b^2 - 1 \equiv a^r - 1$, so the product $(b-1)(b+1)$ must be a multiple of N. But $b - 1 \equiv a^m - 1$ cannot by itself be a multiple of N, since m is smaller than r, and the period r is the *smallest* power of a with $a^r - 1$ a multiple of N.

Second, suppose we are lucky enough to have picked an a for which $b + 1$ is also not a multiple of N. Then neither $b - 1$ nor $b + 1$ is a multiple of N, although their product is. Since N is the product of two prime numbers, $b - 1$ must be a multiple of one of the prime factors of N, and $b + 1$ a multiple of the other. One factor of N is then the greatest common divisor of $b - 1$ and N, while the other factor is the greatest common divisor of $b + 1$ and N.

Now we are finished. Given any two integers, there is a famous and childishly simple way to find their greatest common divisor, which has been known at least since Euclid. It can be carried out by anybody who can do long division, a skill, to be sure, that I recently learned from the *New York Times* is becoming increasingly rare. With the Euclidean algorithm, an efficient period-finding machine, and the two little bits of luck, we can factor the product of two large prime numbers.

How lucky must we be? Here, and only here, I will hide a rather elaborate argument behind the irritating phrase "it can be shown." It can be shown that if a is picked at random, then the probability of its modulo-N period r being even and $a^{r/2} + 1$ not being divisible by N—the two pieces of good fortune we require—is at least 50%. So if we have a good period-finding machine, it need not work on a^x modulo N for many different random integers a to enable us to find the factors of N. If we pick 20 different random a, then the odds against failure are more than a million to one.

Showing what can be shown is the only hard part of establishing the connection between period finding and factoring. But if you are willing to accept the happy fact that you will surely succeed in under 20 attempts (and you surely will), then you now understand on a practical level how to use a wonderful quantum period-finding machine to factor the product of two large primes.

Why period finding is hard

But before we can get to how the quantum period-finding machine does its magic, another question comes irritatingly, but irresistibly, to mind. What's so hard about finding the period of a periodic function? If I produce a graph of $\sin(kx)$, who needs a quantum computer to figure out the distance $d = 2\pi/k$ over which it starts repeating itself? Aren't repeating patterns easily recognized? Indeed, isn't the recognition of their periodicity the basis of the pleasure we take in them?

Yes indeed, provided the set of numbers whose repetition constitutes the periodic sequence has an easily recognizable structure. But the function whose period you need to learn, if you want to factor the enormous number N, is a^x modulo N. The sequence of integers this specimen churns out as x progresses from 1 up to the (in general huge) period r is virtually indistinguishable from random noise. You can't look for a pattern that repeats itself, because there is no pattern. Nothing you can discern from the sequence gives the slightest hint of when it is likely to start over again. Periodically repeating noise looks locally like—in fact locally *is*—just noise.

One thing that does distinguish the set of numbers that repeats from random noise is that within a period r, no value of a^x modulo N can appear twice. (For r is the *smallest* value for which $a^x \equiv a^{x+r}$.) So one sure way to find the period is just to evaluate a^x modulo N for successive values of x until you finally reproduce the first evaluation. The initial and final values of x are then guaranteed to differ by r.

The problem is that for cryptographic purposes, N is typically a number with 400 or more digits, which typically sets the scale for the number of digits in the period r. So this brute-force approach requires an impossibly large number of evaluations. A more subtle strategy is required. The quantum computational route to period finding has some exquisite subtleties. But that must be the subject of a future column.

References

1. R. L. Rivest, A. Shamir, L. Adleman, *Commun. ACM* **21**(2), 120 (1978).
2. P. W. Shor, in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science,* S. Goldwasser, ed., IEEE Computer Society Press, Los Alamitos, CA (1994), p. 124; *SIAM J. Comput.* **26**, 1484 (1997). ■