

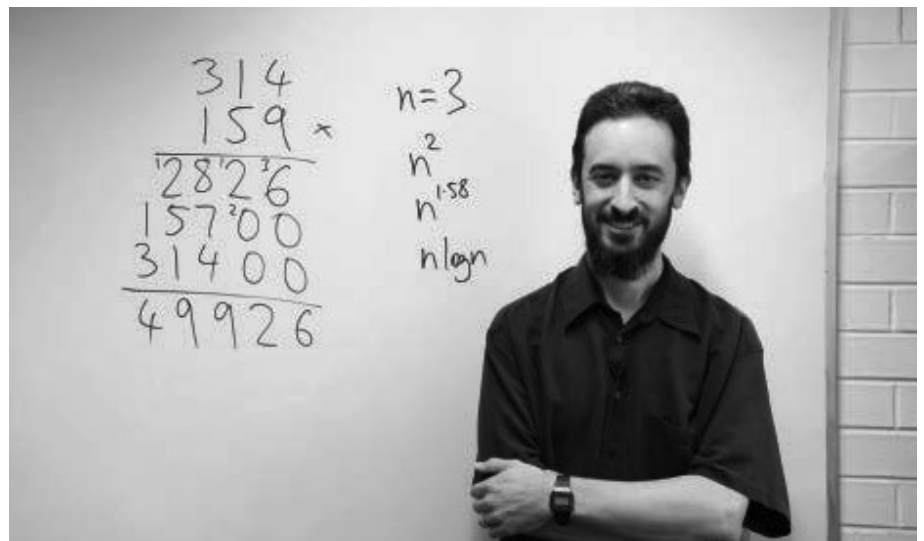
Multiplication Hits the Speed Limit

A problem “around since antiquity” may have been resolved by a new algorithm.

A PAPER POSTED ONLINE in March 2019 presents what may be essentially the fastest possible algorithm for one of the oldest problems in mathematics: whole number multiplication. The new algorithm, which can multiply two n -digit numbers in approximately $n(\log n)$ steps, breaks all previous records and reaches what mathematicians conjectured decades ago should be the fundamental speed limit for any multiplication procedure.

“This problem has been around since antiquity,” said Joshua Cooper of the University of South Carolina in Columbia. “It’s extraordinary to see the state of the art reach what people believe is the truth” about how fast multiplication can be carried out.

The new algorithm outstrips other algorithms only for extremely large numbers, so for now its practical applications are limited. Its theoretical implications, however, are vast. Multiplication lies at the core of nearly every mathematical operation: its speed is as central to arithmetical complexity theory as the speed of light is to physics, said Joris van der Hoeven of the French National Center for Sci-



David Harvey, above, demonstrating how standard multiplication is impractical when multiplying astronomically large numbers together. Below, Joris van der Hoeven, who worked with Harvey on the new algorithm to speed multiplication of such numbers.

entific Research in Paris, who created the new algorithm along with David Harvey of Australia’s University of New South Wales in Sydney.

The new paper immediately implies, for example, that it is possible to calculate the first n digits of the reciprocal or square root of a number in approximately $n(\log n)$ steps, and the first n digits of transcendental constants such as π and e in roughly $n(\log^2 n)$ steps.



“Now we know that all these algorithms that depend on multiplication are the time complexity that we thought they were,” Cooper said.

Too Many Logs

The standard multiplication algorithm children learn in elementary school takes approximately n^2 steps, since every digit of the first number must be multiplied by every digit of the second number. For millennia, no one knew any significantly faster multiplication procedure than this simple method. In 1960, Andrey Kolmogorov—one of the preeminent mathematicians of the 20th century—challenged attendees of a seminar at Moscow State University to prove there is no multiplication algorithm that runs in fewer than about n^2 steps.

Anatoly Karatsuba, a 23-year-old student attending the seminar, set out to meet this challenge, but instead proved the opposite. Karatsuba came up with a clever but elementary way to combine the digits of two numbers to compute their product in approximately $n^{1.58}$ steps.

“It’s one of these incredible things that seems so simple once you see it, but no one saw it until Karatsuba did,” Harvey said.

Other mathematicians quickly found improvements to Karatsuba’s algorithm. Then in 1971, Arnold Schönhage and Volker Strassen made another huge leap, devising an algorithm whose running time is about $n(\log n)(\log(\log n))$ —vastly more efficient than $n^{1.58}$ for large values of n . A streamlined version of Schönhage and Strassen’s algorithm lies at the heart of the GNU Multiple Precision Arithmetic Library used by all the standard arithmetic software packages (although for numbers smaller than a few hundred thousand digits, the library uses other approaches, including Karatsuba’s algorithm).

Schönhage and Strassen’s algorithm, which laid the groundwork for the new algorithm announced this past March, leverages the fast Fourier transform, a procedure for sampling and reconstructing polynomials that is used widely in signal processing. It is easy to translate an integer multiplication problem into a problem about polynomials: Simply use the digits of

the two numbers as the coefficients of two polynomials. So, for example, if you want to multiply 635 and 258, you can convert the two numbers into the polynomials $6x^2+3x+5$ and $2x^2+5x+8$. Multiplying these two polynomials gives $12x^4+36x^3+73x^2+49x+40$, and if we plug in the value $x=10$, the polynomial outputs the product of 635 and 258, namely 163,830.

If you calculate the polynomial product by multiplying the two polynomials term by term, as students learn to do in algebra class, this translation doesn’t achieve any speedup over the n^2 integer multiplication algorithm. Yet there is a way to vastly speed up polynomial multiplication. Any polynomial whose highest exponent is k is completely determined by its values at $k+1$ different inputs. So in the example above, the product polynomial, whose highest exponent is 4, is uniquely determined by its values at any five inputs. That means that if we choose our five favorite x values, evaluate $6x^2+3x+5$ and $2x^2+5x+8$ at those five values and then multiply the corresponding outputs, those five multiplications already give us enough information to reconstruct the product polynomial (compared with nine multiplications if we multiply the two polynomials term by term).

What this analysis sweeps under the rug, though, is the cost of first evaluating $6x^2+3x+5$ and $2x^2+5x+8$ at the five inputs, and then reconstructing the product polynomial at the end of the

No one thought it would be possible to bring the running time of integer multiplication down to roughly n steps, which would put multiplication on a par with addition.

procedure. That’s where the fast Fourier transform comes in: It provides a speedy way to do such polynomial evaluations and reconstructions, provided the five input values are chosen carefully (specifically, they should be the five complex numbers whose fifth powers equal 1).

Using the fast Fourier transform, combined with a more sophisticated way of converting numbers into polynomials than the naïve digit-by-digit translation above, Schönhage and Strassen were able to achieve their $n(\log n)(\log(\log n))$ algorithm. For more than three decades after their work, no one could come up with anything significantly faster.

Yet computer scientists found the $n(\log n)(\log(\log n))$ running time disturbingly inelegant. For that reason, Schönhage and Strassen’s algorithm didn’t seem like the final word on the subject.

No one thought it would be possible to bring the running time of integer multiplication all the way down to roughly n steps, which would put multiplication on a par with addition. However, Schönhage and Strassen, along with many others, suspected the true complexity of multiplication—the running time of the fastest possible algorithm—should be $n(\log n)$, not $n(\log n)(\log(\log n))$.

“Everybody feels like multiplication is more complicated than addition,” said Martin Fürer of Pennsylvania State University in University Park. “But everyone thought the $\log(\log n)$ term should not be necessary. From an aesthetic point of view, it doesn’t look nice. Such a fundamental task as multiplication should have a nice complexity.”

The End of the Story

In 2007, Fürer finally managed to whittle down the $\log(\log n)$ term in Schönhage and Strassen’s algorithm to something slightly smaller. Fürer’s algorithm was impractical for any multiplications people might want to carry out in real life, but the theoretical advance electrified Harvey and van der Hoeven. Over the past five years, they have collaborated on a series of about 10 papers further improving Fürer’s bound. “There were twice as many papers that never got written, and algorithms that never saw the light of

day because they were superseded by something else,” Harvey said.

Finally, in March 2019 the pair figured out how to eliminate the $\log(\log n)$ term completely. Their new algorithm uses a higher-dimensional version of the fast Fourier transform, combined with a method they devised for increasing the number of sampling points to take advantage of additional speedups when the number of sampling points is a power of two. “It’s definitely the hardest paper I ever worked on,” Harvey said.

The algorithm entails rounding off the complex numbers involved in the fast Fourier transform to achieve a balance of speed and precision that is “kind of amazing,” Cooper said. “They’re performing exact integer multiplication, but in the process they’re passing into this other world using complex numbers and polynomials and doing an approximate computation, then coming back and getting an exact answer.”


The $n(\log n)$ bound means Harvey and van der Hoeven’s algorithm is faster than Schönhage and Strassen’s algorithm, or Fürer’s algorithm, or any other known multiplication algorithm, provided n is sufficiently large. For now, “sufficiently large” means almost unfathomably large: Harvey and van der Hoeven’s algorithm doesn’t even kick in until the number of bits in the two numbers being multiplied is greater than 2 raised to the 1729¹² power. (By comparison, the number of particles in the observable Universe is commonly put at about 2⁷⁰.)

Harvey and van der Hoeven made no efforts to optimize their algorithm. This was partly because they were focused on the theoretical advance, and partly because they were tickled when their back-of-the-envelope calculations led them to the number 1729, which, in a famous anecdote, the mathematician Srinivasa Ramanujan called a “very interesting” number (because it is the smallest number that can be written as a sum of two cubes in two different ways). “When I saw this, I burst out laughing,” Harvey recalled.

Other researchers will likely find ways to tweak Harvey and van der Hoeven’s algorithm so it outperforms other algorithms at smaller and smaller

The algorithm entails rounding off the complex numbers in the fast Fourier transform to achieve a balance of speed and precision that is “kind of amazing,” Cooper said.

numbers. What they are unlikely to do, many researchers agree, is come up with any algorithm qualitatively faster than $n(\log n)$. No one knows how to prove this—as a rule, establishing there are no algorithms faster than some bound is much harder than coming up with a new fast algorithm.

Nevertheless, “It would really surprise us if it is possible to do better than $n(\log n)$,” van der Hoeven said. “We feel that the story for integer multiplication ends here.” 

Further Reading

Fürer, M.
Faster Integer Multiplication. *SIAM J. Comput.*, Volume 39 Issue 3, 2009, pages 979-1005
<http://bit.ly/2CFvmG6>

Harvey, D. and van der Hoeven, J.
Integer Multiplication in Time $O(n \log n)$.
<http://bit.ly/2QcOrr6>

Karatsuba, A.
The Complexity of Computations. *Proceedings of the Steklov Institute of Mathematics*, Volume 211, 1995, pages 169-183
<http://bit.ly/32M1oed>

Schönhage, A. and Strassen, V.
Schnelle Multiplikation grosser Zahlen. *Computing*, Volume 7, Issue 3-4, September 1971, pages 281-292.
<https://link.springer.com/article/10.1007/BF02242355>

Erica Klarreich is a mathematics and science journalist based in Berkeley, CA, USA.

© 2020 ACM 0001-0782/20/1 \$15.00

Milestones

ACM Recognizes Distinguished Members

ACM recently inducted 62 Distinguished Members for their outstanding contributions to the field.

The 2019 inductees are long-standing ACM members selected by their peers for a range of accomplishments that have contributed to technologies that underpin how we live, work, and play.

“Each year, it is our honor to select a new class of Distinguished Members,” explains ACM president Cheri M. Pancake. “In everything we do, our overarching goal is to build a community wherein computing professionals can grow professionally and, in turn, contribute to the field and the broader society. We are delighted to recognize these individuals for their contributions to computing, and we hope that the careers of the 2019 ACM Distinguished Members will continue to prosper through their participation with ACM.”

The 2019 ACM Distinguished Member program recognizes contributions in a wide range of technical areas, including artificial intelligence, human-computer interaction, computer engineering, computer science education, cybersecurity, graphics, and networking.

The ACM Distinguished Member program recognizes up to 10% of ACM worldwide membership based on professional experience and significant achievement in the computing field. Candidates must have at least 15 years of professional experience in computing, five years of continuous professional ACM membership, and have achieved a significant level of accomplishment or made a significant impact in computing, computer science, and/or information technology.

Distinguished Members serve as mentors and role models, guiding technical career development and contributing to the field beyond the norm.

The list of new Distinguished Members can be viewed at <http://bit.ly/2CBLJDB>.