1.9 Fast Multiplication

In the previous chapter, we saw two ancient algorithms for multiplying two n-digit numbers in $O(n^2)$ time: the grade-school lattice algorithm and the Egyptian peasant algorithm.

Maybe we can get a more efficient algorithm by splitting the digit arrays in half and exploiting the following identity:

$$(10^m a + b)(10^m c + d) = 10^{2m} ac + 10^m (bc + ad) + bd$$

This recurrence immediately suggests the following divide-and-conquer algorithm to multiply two n-digit numbers x and y. Each of the four sub-products ac, bc, ad, and bd is computed recursively, but the multiplications in the last line are *not* recursive, because we can multiply by a power of ten by shifting the digits to the left and filling in the correct number of zeros, all in $O(n)$ time.

```
SplitMultiply(x, y, n):
  if n = 1
      return x · y
  else
      m ← ⌈n/2⌉
      a ← ⌊x/10^m⌋;  b ← x mod 10^m        ⟨⟨x = 10^m a + b⟩⟩
      c ← ⌊y/10^m⌋;  d ← y mod 10^m        ⟨⟨y = 10^m c + d⟩⟩
      e ← SplitMultiply(a, c, m)
      f ← SplitMultiply(b, d, m)
      g ← SplitMultiply(b, c, m)
      h ← SplitMultiply(a, d, m)
      return 10^(2m) e + 10^m (g + h) + f
```

Correctness of this algorithm follows easily by induction. The running time for this algorithm follows the recurrence

$$T(n) = 4T(\lceil n/2 \rceil) + O(n).$$

The recursion tree method transforms this recurrence into an *increasing* geometric series, which implies $T(n) = O(n^{\log_2 4}) = O(n^2)$. In fact, this algorithm multiplies each digit of x with each digit of y, just like the lattice algorithm. So I guess that didn't work. Too bad. It was a nice idea.

In the mid-1950s, Andrei Kolmogorov, one of the giants of 20th century mathematics, publicly conjectured that there is *no* algorithm to multiply two n-digit numbers in subquadratic time. Kolmogorov organized a seminar at Moscow University in 1960, where he restated his "n^2 conjecture" and posed several related problems that he planned to discuss at future meetings. Almost exactly a week later, a 23-year-old student named Anatoliĭ Karatsuba presented Kolmogorov with a remarkable counterexample. According to Karatsuba himself,

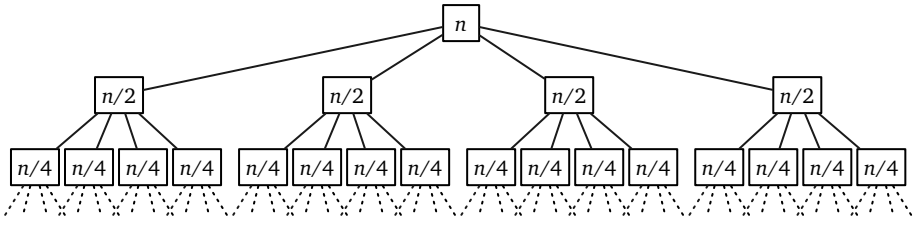

Figure 1.14. The recursion tree for naïve divide-and-conquer multiplication

> After the seminar I told Kolmogorov about the new algorithm and about the disproof of the n^2 conjecture. Kolmogorov was very agitated because this contradicted his very plausible conjecture. At the next meeting of the seminar, Kolmogorov himself told the participants about my method, and at that point the seminar was terminated.

Karatsuba observed that the middle coefficient $bc+ad$ can be computed from the other two coefficients ac and bd using only *one* more recursive multiplication, via the following algebraic identity:

$$ac + bd - (a-b)(c-d) = bc + ad$$

This trick lets us replace the four recursive calls in the previous algorithm with only three recursive calls, as shown below:

```
FastMultiply(x, y, n):
  if n = 1
      return x · y
  else
      m ← ⌈n/2⌉
      a ← ⌊x/10^m⌋;  b ← x mod 10^m        ⟨⟨x = 10^m a + b⟩⟩
      c ← ⌊y/10^m⌋;  d ← y mod 10^m        ⟨⟨y = 10^m c + d⟩⟩
      e ← FastMultiply(a, c, m)
      f ← FastMultiply(b, d, m)
      g ← FastMultiply(a − b, c − d, m)
      return 10^(2m) e + 10^m (e + f − g) + f
```

The running time of Karatsuba's FastMultiply algorithm follows the recurrence

$$T(n) \leq 3T(\lceil n/2 \rceil) + O(n)$$

Once again, the recursion tree method transforms this recurrence into an increasing geometric series, but the new solution is only $T(n) = O(n^{\log_2 3}) = \boldsymbol{O(n^{1.58496})}$, a significant improvement over our earlier quadratic time bound.[10]

[10] My presentation simplifies the actual history slightly. In fact, Karatsuba proposed an algorithm based on the formula $(a+b)(c+d) - ac - bd = bc + ad$. This algorithm also runs in $O(n^{\lg 3})$ time, but the actual recurrence is slightly messier: $a-b$ and $c-d$ are still m-digit numbers, but $a+b$ and $c+d$ might each have $m+1$ digits. The simplification presented here is due to Donald Knuth.

Karatsuba's algorithm arguably launched the design and analysis of algorithms as a formal field of study.

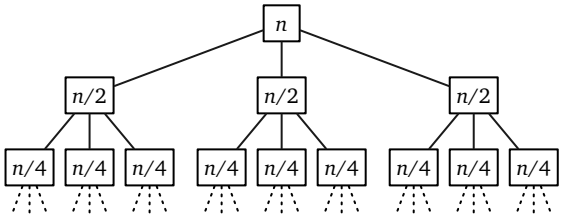

Figure 1.15. The recursion tree for Karatsuba's divide-and-conquer multiplication algorithm

We can take Karatsuba's idea even further, splitting the numbers into more pieces and combining them in more complicated ways, to obtain even faster multiplication algorithms. Andrei Toom discovered an infinite family of algorithms that split any integer into k parts, each with n/k digits, and then compute the product using only $2k-1$ recursive multiplications; Toom's algorithms were further simplified by Stephen Cook in his PhD thesis. For any fixed k, the Toom-Cook algorithm runs in $O(n^{1+1/(\lg k)})$ time, where the hidden constant in the $O(\cdot)$ notation depends on k.

Ultimately, this divide-and-conquer strategy led Gauss (yes, really) to the discovery of the ***Fast Fourier transform***.[11] The basic FFT algorithm itself runs in $O(n \log n)$ time; however, using FFTs for integer multiplication incurs some small additional overhead. The first FFT-based integer multiplication algorithm, published by Arnold Schönhage and Volker Strassen in 1971, runs in $O(n \log n \log\log n)$ time. Schönhage-Strassen remained the theoretically fastest integer multiplication algorithm for several decades, before Martin Fürer discovered the first of a long series of technical improvements. Finally, in 2019, David Harvey and Joris van der Hoeven published an algorithm that runs in $O(n \log n)$ time.[12]

1.10 Exponentiation

Given a number a and a positive integer n, suppose we want to compute a^n. The standard naïve method is a simple for-loop that performs $n-1$ multiplications by a:

[11] See http://algorithms.wtf for lecture notes on Fast Fourier transforms.

[12] Schönhage-Strassen is actually the fastest algorithm *in practice* for multiplying integers with more than about 75000 digits; the more recent algorithms of Fürer, Harvey, van der Hoeven, and others would be faster "in practice" only for integers with more digits than there are particles in the universe.