Kapitel 1
Multiplication of Long Integers (Faster than Long Multiplication)

Arno Eigenwillig und Kurt Mehlhorn

An algorithm for multiplication of integers is taught already in primary school: To multiply two positive integers a and b, you multiply a by each digit of b and arrange the results as the rows of a table, aligned under the corresponding digits of b. Adding up yields the product $a \times b$. Here is an example:

$$\begin{array}{r} 5678 \cdot 4321 \\ \hline 22712 \\ 17034 \\ 11356 \\ 5678 \\ \hline 24534638 \end{array}$$

The multiplication of a by a single digit is called *short multiplication*, and the whole method to compute $a \times b$ is called *long multiplication*. For integers a and b with very many digits, long multiplication does indeed take long, even if carried out by a modern computer. Calculations with very long integers are used in many applications of computers, for example, in the encryption of communication on the Internet (see Chapters **??** and **??**), or, to name another example, in the reliable solution of geometric or algebraic problems.

Fortunately, there are better ways to multiply. This is good for the applications needing it. But it is also quite remarkable in itself because long multiplication is so familiar and looks so natural that any substantial improvement comes as a surprise.

In the rest of this chapter, we will investigate:

1. How much effort does it take to do long multiplication of two numbers?
2. How can we do better?

Computer scientists do not measure the effort needed to carry out an algorithm in seconds or minutes because such information will depend on the

Max Planck Institute for Informatics, Saarbrücken, Germany

hardware, the programming language, and the details of the implementation (and next year's hardware will be faster anyway). Instead, computer scientists count the number of *basic operations* performed by an algorithm. A basic operation is something, a computer or a human can do in a single step. The basic operations we need here are basic computations with the digits $0, 1, 2, \ldots, 8, 9$.

1. **Multiplication of two digits:** When given two digits x and y, we know the two digits u and v that make up their product $x \times y = 10 \times u + v$. We trust our readers to remember the multiplication table!
   Examples: For the digits $x = 3$ and $y = 7$ we have $x \times y = 3 \times 7 = 21 = 10 \times 2 + 1$, so the resulting digits are $u = 2$ and $v = 1$. For $x = 3$ and $y = 2$, the resulting digits are $u = 0$ and $v = 6$.
2. **Addition of three digits:** When given three digits x, y, z, we know the two digits u and v of their sum: $x + y + z = 10 \cdot u + v$. It will soon become obvious why we want to add three digits at once.
   Example: For $x = 3$, $y = 5$ and $z = 4$ the result is $u = 1$ and $v = 2$ because $3 + 5 + 4 = 12 = 10 \times 1 + 2$.

How many of these basic operations are done in a long multiplication? Before we can answer this question, we need to look at two simpler algorithms, addition and short multiplication, used during long multiplication.

The addition of long numbers

How much effort does it take to add two numbers a and b? Of course, this depends on how many digits they have. Let us assume that a and b both consist of n digits. If one of them is shorter than the other, one can put zeros in front of it until it is as long as the other. To add the two numbers, we write one above the other. Going from right to left, we repeatedly do addition of digits. Its result $10 \times u + v$ gives us the result digit v for the present column as well as the digit u carried to the next column. Here is an example with numbers $a = 6917$ and $b = 4269$ with $n = 4$ digits each:

$$\begin{array}{r} 6917 \\ 4269 \\ \hline {\scriptstyle 1\,1\,0\,1\;} \\ \hline 11186 \end{array}$$

The carry digit v from the leftmost column is put in front of the result without further computation. Altogether, we have done n basic operations, namely one addition of digits per column.

Short multiplication: a number times a digit

During long multiplication, we need to multiply the number a, the left factor, with a digit y from the right factor. We now look at this *short multiplication* in more detail. To do so, we write down its intermediate results more carefully than usual: We go from right to left over the digits of a. We multiply each digit x of a by the digit y and write down the result $10 \times u + v$ in a separate row, aligned such that v is in the same column as x. When all digits are multiplied, we add all the two-digit intermediate results. This gives us the result of the short multiplication which is usually written as a single row.

As an example for this, we look again at the long multiplication 5678×4321: The first of the short multiplications it needs is this one:

$$\begin{array}{r} 5\,6\,7\,8 \cdot 4 \\ \hline \quad 3\,2 \\ \quad 2\,8 \\ \quad 2\,4 \\ 2\,0 \\ 0\,0\,1\,0 \\ \hline 2\,2\,7\,1\,2 \end{array}$$

How many basic operations did we use? For each of the n digits of a, we have done one multiplication of digits. In the example above: four multiplications of digits for the four digits of 5678. After that, we have added the intermediate results in $n+1$ columns. In the rightmost column, there is a single digit which we can just copy to the result without computing. In the other n columns are two digits and a carry from the column to its right, so one addition of digits suffices to add them. This means we needed to do n additions of digits. Together with the n multiplications of digits, it has taken $2 \times n$ basic operations to multiply an n-digit number a with a digit y.

The analysis of long multiplication

Let us now analyze the number of basic operations used by long multiplication of two numbers a and b, each of which has n digits. In case one is shorter than the other, we can pad it with zeros at the front.

For each digit y of b we need to do one short multiplication $a \times y$. This needs $2 \times n$ basic operations, as we saw above. Because there are n digits in b, long multiplication needs n short multiplications which together account for $n \times (2 \times n) = 2 \times n^2$ basic operations.

The results of the short multiplications are aligned under the respective digits of b. To simplify the further analysis, we put zeros in the empty positions:

$$
\begin{array}{r}
5\,6\,7\,8 \cdot 4\,3\,2\,1 \\
\hline
2\,2\,7\,1\,2\,0\,0\,0 \\
0\,1\,7\,0\,3\,4\,0\,0 \\
0\,0\,1\,1\,3\,5\,6\,0 \\
0\,0\,0\,0\,5\,6\,7\,8 \\
\hline
2\,4\,5\,3\,4\,6\,3\,8
\end{array}
$$

The results of short multiplication are added with the method described previously. We add the first row to the second row, their sum to the third row, and so on, until all n rows have been added. This needs $n-1$ additions of long numbers. In our example, n is equal to 4, and we need these $n-1=3$ additions: $22712000+1703400=24415400$, $24415400+113560=24528960$ and $24528960+5678=24534638$.

How many basic operations do these $n-1$ additions need? To answer this, we have to know how many digits are required for the intermediate sums in this chain of additions. With a little bit of thinking, we can convince ourselves that the final result $a \times b$ has at most $2 \times n$ digits. While we add the parts of this result, the numbers can only get longer. Therefore, all intermediate sums have at most $2 \times n$ digits, like the final result. Therefore, we do $n-1$ additions of numbers with at most $2 \times n$ digits. According to our analysis of addition, this requires at most $(n-1) \times (2 \times n) = 2 \times n^2 - 2 \times n$ basic operations. Together with the $2 \times n^2$ basic operations used by the short multiplications, this yields a grand total of at most $4 \times n^2 - 2 \times n$ basic operations carried out in the long division of two n-digit numbers.

Let us see what this means for a concrete example. If we have to multiply really long numbers, say, with 100 000 digits each, then it takes almost 40 billion basic operations to do one long multiplication, including 10 billion multiplications of digits. In other words: Per digit in the result, this long multiplication needs, on average, 200 000 basic operations which is clearly a bad ratio. This ratio gets much worse if the number of digits increases: For 1 million digits, long multiplication needs almost 4 trillion basic operations (of which 1 trillion are multiplications of digits). On average, it spends about 2 million basic operations for a single digit in the result.

Karatsuba's method

Let us now do something smarter. We look at an algorithm for multiplying two n-digit numbers that needs much fewer basic operations. It is named after the Russian mathematician Anatolii Alexeevitch Karatsuba, who came up with its main idea (published 1962 with Yu. Ofman[1]). We first describe

[1] A. Karatsuba, Yu. Ofman: "Multiplication of multidigit numbers on automata" (in Russian), *Doklady Akad. Nauk SSSR* **145** (1962), pp. 293–294; English translation in *Soviet*

the method for numbers with one, two, or four digits, and then for numbers of any length.

The simplest case is, of course, the multiplication of two numbers consisting of one digit each ($n = 1$). Multiplying them needs a single basic operation, namely one multiplication of digits, which immediately gives the result.

The next case we look at is the case $n = 2$, that is, the multiplication of two numbers a and b having two digits each. We split them into halves, that is, into their digits:

$$a = p \times 10 + q \quad \text{and} \quad b = r \times 10 + s.$$

For example, we split the numbers $a = 78$ and $b = 21$ like this:

$$p = 7,\ q = 8, \quad \text{and} \quad r = 2,\ s = 1.$$

We can now rewrite the product $a \times b$ in terms of the digits:

$$\begin{aligned} a \times b &= (p \times 10 + q) \times (r \times 10 + s) \\ &= (p \times r) \times 100 + (p \times s + q \times r) \times 10 + q \times s. \end{aligned}$$

Continuing the example $a = 78$ and $b = 21$, we get

$$78 \cdot 21 = (7 \cdot 2) \cdot 100 + (7 \cdot 1 + 8 \cdot 2) \cdot 10 + 8 \cdot 1 = 1638.$$

Writing the product $a \times b$ of the two-digit numbers a and b as above shows how it can be computed using **four** multiplications of one-digit numbers, followed by additions. This is precisely what long multiplication does.

Karatsuba had an idea that enables us to multiply the two-digit numbers a and b with just **three** multiplications of one-digit numbers. These three multiplications are used to compute the following auxiliary products:

$$\begin{aligned} u &= p \times r, \\ v &= (q - p) \times (s - r), \\ w &= q \times s. \end{aligned}$$

Computing v deserves extra attention because it involves the subtraction of digits. We need a new kind of basic operation: subtraction of digits. It is used twice in computing v: subtraction of digits. The results $(q - p)$ and $(s - r)$ are again single digits, but possibly with a negative sign. Multiplying them to get v requires a multiplication of digits and an application of the usual rules to determine the sign ("minus times minus gives plus", and so on).

Why does all this help to multiply a and b? The answer comes from this formula:

Physics Doklady **7** (1963), pp. 595–596. Karatsuba describes his method to efficiently compute the square a^2 of a long number a. The *multiplication* of numbers a and b is reduced to squaring with the formula $a \times b = \frac{1}{4}((a + b)^2 - (a - b)^2)$.

$$u + w - v = p \times r + q \times s - (q - p) \times (s - r) = p \times s + q \times r.$$

Karatsuba's trick consists in using this formula to express the product $a \times b$ in terms of the three auxiliary products u, v, and w:

$$a \times b = u \times 10^2 + (u + w - v) \times 10 + w.$$

Let us carry out Karatsuba's trick for our example $a = 78$ and $b = 21$ from above. The three Karatsuba multiplications are

$$\begin{aligned} u &= 7 \times 2 &&= 14, \\ v &= (8 - 7) \times (1 - 2) &&= -1, \\ w &= 8 \times 1 &&= 8. \end{aligned}$$

We obtain

$$\begin{aligned} 78 \times 21 &= 14 \times 100 + (14 + 8 - (-1)) \times 10 + 8 \\ &= 1400 + 230 + 8 \\ &= 1638. \end{aligned}$$

We have used two subtractions of digits, three multiplications of digits, and several additions and subtractions of digits to combine the results of the three multiplications.

Karatsuba's method for 4-digit numbers

Having dealt with the case of $n = 2$ digits above, we now look at the case of $n = 4$ digits, that is, the multiplication of two numbers a and b with four digits each. Just like before we can split each of them into two halves p and q, or r and s, respectively. These halves are not digits anymore, but two-digit numbers:

$$a = p \times 10^2 + q \quad \text{and} \quad b = r \times 10^2 + s.$$

Again, we compute the three auxiliary products from these four halves:

$$\begin{aligned} u &= p \times r, \\ v &= (q - p) \times (s - r), \\ w &= q \times s. \end{aligned}$$

Just like before, we obtain the product $a \times b$ from the auxiliary products as

$$a \times b = u \times 10^4 + (u + w - v) \times 10^2 + w.$$

Example: We look again at the task of multiplying $a = 5678$ and $b = 4321$. We begin by splitting a and b into the halves $p = 56$ and $q = 78$ as well as

$r = 43$ and $s = 21$. We compute the auxiliary products

$$\begin{aligned} u &= 56 \times 43 &&= 2408, \\ v &= (78 - 56) \times (21 - 43) &&= -484, \\ w &= 78 \times 21 &&= 1638. \end{aligned}$$

It follows that

$$\begin{aligned} 5678 \times 4321 &= 2408 \times 10000 + (2408 + 1638 - (-484)) \times 100 + 1638 \\ &= 24080000 + 453000 + 1638 \\ &= 24534638. \end{aligned}$$

In this calculation, we had to compute three auxiliary products of two-digit numbers. In the previous section, we investigated how to do that with Karatsuba's method, using only three multiplications of digits each time. This way, we can compute the three auxiliary products using only $3 \times 3 = 9$ multiplications of digits and several additions and subtractions. Long division would have taken 16 multiplications of digits and several additions.

Karatsuba's method for numbers of any length

Recall how we have built Karatsuba's method for multiplying 4-digit numbers from Karatsuba's method for 2-digit numbers. Continuing in the same way, we can build the multiplication of 8-digit numbers from three multiplications of 4-digit numbers, and the multiplication of 16-digit numbers from three multiplications of 8-digit numbers, and so on. In other words, Karatsuba's method works for any number n of digits that is a power of 2, such as $2 = 2^1$, $4 = 2 \times 2 = 2^2$, $8 = 2 \times 2 \times 2 = 2^3$, $16 = 2 \times 2 \times 2 \times 2 = 2^4$, and so on.

The general form of Karatsuba's method is this: Two numbers a and b, each consisting of $n = 2 \times 2 \times 2 \times \cdots \times 2 = 2^k$ digits, are split into

$$a = p \times 10^{n/2} + q \quad \text{and} \quad b = r \times 10^{n/2} + s.$$

Then their product $a \times b$ is computed as follows, using three multiplications of numbers having $n/2 = 2^{k-1}$ digits each:

$$a \times b = p \times r \times 10^n + (p \times r + q \times s - (q - p) \times (s - r)) \times 10^{n/2} + q \times s$$

Multiplying two numbers of 2^k digits in this way takes only three times (and not four times) as many multiplications of digits as multiplying two numbers with 2^{k-1} digits. This leads to the following table which compares how many multiplications of digits are used by Karatsuba's method, or by long multiplication, respectively, to multiply numbers with n digits.

digits	Karatsuba	long multiplication
$1 = 2^0$	1	1
$2 = 2^1$	3	4
$4 = 2^2$	9	16
$8 = 2^3$	27	64
$16 = 2^4$	81	256
$32 = 2^5$	243	1024
$64 = 2^6$	729	4096
$128 = 2^7$	2187	16 384
$256 = 2^8$	6561	65 536
$512 = 2^9$	19 638	262 144
$1024 = 2^{10}$	59 049	1 048 576
$1\,048\,576 = 2^{20}$	3 486 784 401	1 099 511 627 776
...	...	...
$n = 2^k$	3^k	4^k

Using logarithms, it is easy to express the entries in the table as functions of n: For $n = 2^k$, the column for long multiplication contains the value 4^k. We write log for the logarithm with base 2. We have $k = \log(n)$ and

$$4^k = 4^{\log(n)} = (2^{\log(4)})^{\log(n)} = n^{\log(4)} = n^2.$$

For $n = 2^k$, the column for Karatsuba's method contains the value

$$3^k = 3^{\log(n)} = (2^{\log(3)})^{\log(n)} = n^{\log(3)} = n^{1{,}58\ldots}.$$

Let us return to the question how much effort it takes to multiply two numbers of one million digits each. Long multiplication takes almost 4 trillion basic operations, including 1 trillion multiplication of digits. To use Karatsuba's method instead, we first need to put zeros in front of both numbers, to bring their length up to the next power of two which is $2^{20} = 1\,048\,576$. Without this padding, we could not split the numbers in halves again and again until we reach a single digit. With Karatsuba's method, we can multiply the two numbers using "only" 3.5 billion multiplications of digits, one 287th of what long multiplication needed. For comparison: one second is a 300th of the proverbial "five minutes". So we see: Karatsuba's method indeed requires much less computational effort, at least when counting only multiplications of digits, as we have done. A precise analysis also has to take the additions and subtractions of intermediate results into account. This will show that long multiplication is actually faster for numbers with only a few digits. But as numbers get longer, Karatsuba's method becomes superior because it produces less intermediate results. It depends on the properties of the particular computer and the representation of long numbers inside it when exactly Karatsuba's method is faster than long multiplication.

Summary

The recipe for success in Karatsuba's method has two ingredients:

The first is a very general one: The task "multiply two numbers of n digits each" is reduced to several tasks of the same form, but of smaller size, namely "multiply two numbers of $n/2$ digits each". We keep on subdividing until the problem has become simple: "multiply two digits". This problem-solving strategy is called *divide and conquer*, and it has appeared before in this book (for example, in Chapter **??** on fast sorting). Of course, a computer does not need a separate procedure for each size of the problem. Instead, there is a general procedure with a parameter n for the size of the problem, and this procedure invokes itself several times for the reduced size $n/2$. This is called *recursion*, and it is one of the most important and fundamental techniques in computer science. Recursion also has appeared before in this book (for example, in Chapter **??** on depth-first search).

The second ingredient in Karatsuba's method, specifically aimed at multiplication, is his trick to divide the problem in a way that results in three (instead of four) sub-problems of half the size. Accumulated over the whole recursion, this seemingly miniscule difference results in significant savings and gives Karatsuba's method its big advantage over long multiplication.

Further reading

1. A. A. Karatsuba: *The Complexity of Computations*. Proceedings of the Steklov Institute of Mathematics, Vol. 211, 1995, pages 169 - 183, available at `http://www.ccas.ru/personal/karatsuba/divcen.pdf`.

   A. A. Karatsuba reports about the history of his invention and describes it in his own words.

2. A. K. Dewdney: *The (New) Turing Omnibus*. Computer Science Press, Freeman, 2nd ed., 1993; reprint (paperback) 2001.

   These "66 excursions in computer science" include a visit to the multiplication algorithms of Karatsuba (for long numbers) and Strassen (a similar idea for matrices).

3. Wolfram Koepf: *Computeralgebra*. Springer, 2006.

   A gentle introduction to computer algebra. Unfortunately, only available in German.

4. Joachim von zur Gathen, Jürgen Gerhard: *Modern Computer Algebra*. Cambridge University Press, 2nd ed., 2003.

   This beautifully prepared textbook for advanced students of computer science and mathematics discusses Karatsuba's method and more advan-

ced methods (based on Fast Fourier Transformation) for the multiplication of polynomials.

5. Donald E. Knuth: *The Art of Computer Programming*, volume 2: Seminumerical Algorithms. Addison-Wesley, 3rd ed., 1998.

   This heavyweight classic of theoretical computer science treats Karatsuba's method and other, more advanced algorithms for efficient multiplication of integers, in particular the algorithm of Schönhage and Strassen, whose running time has a bound proportional to $n \times \log(n) \times \log(\log(n))$.

6. Martin Fürer: *Faster integer multiplication*. Annual ACM Symposium on Theory of Computing archive Proceedings of the thirty-ninth annual ACM symposium on Theory of Computing, 2007, pages 57 - 66

   The currently asymptotically best method for multiplying long integers. Its running time is proportional to $n \log(n) 2^{O(\log^*(n))}$.

7. Wikipedia:
   `http://en.wikipedia.org/wiki/Karatsuba_algorithm`
   `http://en.wikipedia.org/wiki/Multiplication_algorithm`

Acknowledgements

The authors thank H. Alt, M. Dietzfelbinger and C. Klost for helpful remarks on an earlier version of this chapter.