Speeding up the Karatsuba Algorithm

Satish Ramakrishna

Department of Physics & Astronomy,

Rutgers, the State University of New Jersey,

136 Frelinghuysen Road Piscataway, NJ 08854-8019[*]

Kamesh Aiyer

Kashi Software Inc,

11 Magazine Street,

Cambridge, MA 02139[†]

Abstract

This paper describes an $\sim \mathcal{O}(n)$ pre-compute technique to speed up the Karatsuba algorithm for multiplying two numbers.

Keywords: Karatsuba; memory; CPU

The Karatsuba algorithm [1, 2], an $\mathcal{O}(n^{\log_2 3})$ technique to multiply two n-digit numbers, has been surpassed by newer techniques that are $\mathcal{O}(n \times \log n \times \log\log n)$ [3–7] and $\mathcal{O}(n \times \log n)$ [8] respectively. However, the simplicity of the algorithm allows improvements that are easily implemented and can be reduced to fewer multiplications, supplemented by look-ups.

THE ORIGINAL ALGORITHM

For simplicity, consider multiplying two n-digit numbers x and y, written as

$$\begin{aligned} x &= x_0 + x_1 10^m \\ y &= y_0 + y_1 10^m \end{aligned} \tag{1}$$

where for simplicity,we use $n = 2^k$, and $m = \frac{n}{2}$ and work in base-10. The product can be simplified to

$$\begin{aligned} x.y &= x_0 y_0 + (x_0 y_1 + x_1 y_0) \times 10^m + x_1 y_1 10^{2m} \\ &= x_0 y_0 + ((x_0 + x_1)(y_0 + y_1) - x_0 y_0 - x_1 y_1) \times 10^m + x_1 y_1 10^{2m} \end{aligned} \tag{2}$$

so that the product of the n-digit numbers can be reduced to the multiplication of three m-digit (and occasionally $m+1$-digit) numbers, instead of four m-digit numbers. Note that multiplications by 10 are ignored in the complexity calculations, as they can be reduced to shifts. The order of magnitude (”complexity”) of the number of computations to multiply these two numbers can be reduced to the relation

$$\mathcal{M}(n) = 3\mathcal{M}(\frac{n}{2}) + \mathcal{O}(n) \rightarrow \mathcal{M}(n) \sim \mathcal{O}(n^{\log_2 3}) \tag{3}$$

This may be seen simply as follows. At every step, the starting number (initially $n = 2^s$ digits long) is split into two numbers with half the number of digits. After s steps, there are 3^s multiplications of single-digit numbers that need to be performed. This number can be written as $3^s = 3^{\log_2 n}$ which can be further re-written as $n^{\log_2 3}$, hence the result. This calculation is exact for a number which has a number of digits that is a power of 2.

doi:10.20944/preprints201906.0087.v1

THE IMPROVED VERSION

We generalize the above algorithm as follows and write (in base-B, though for simplicity, we will use $B = 10$)

$$x = x_0 + x_1 \times B^m + x_2 \times B^{2m} + ... + x_N \times B^{Nm}$$
$$y = y_0 + y_1 \times B^m + y_2 \times B^{2m} + ... + y_N \times B^{Nm} \tag{4}$$

As can be quickly checked, each of the numbers $x_0, ..., x_N, y_0, ..., y_N$ are m-digits long and $(N + 1)m = n$ where n is the total number of digits in x. When one multiplies the two numbers x, y, the number of multiplications (the order of complexity) is $\mathcal{M} = (N + 1) + \mathcal{C}(N + 1, 2) \sim \frac{(N+1)^2}{2} \sim \frac{n^2}{2m^2}$ individual products of m-digit (and occasionally $m + 1$-digit) numbers. For instance, if $N = 1$, as in the usual Karatsuba technique, $\mathcal{M} = 2+1 = 3$ which is what we use in the order-of-magnitude estimate in Equation (3).

In the Karatsuba technique, the m-digit numbers are further multiplied by the same technique, carrying on recursively till we are reduced to single-digit multiplications. That leads to the recursive complexity calculation noted in Equation (3).

However, note that if we simply pre-computed the individual m-digit multiplications and looked up the individual multiplications, we end up with essentially $\sim \frac{n^2}{2m^2}$ lookups rather than actual multiplications. Indeed, lookups take, on average, 1/5 the time taken for single-digit multiplication (and then we have to multiply by the number of operations $\mathcal{L}$ required to perform the lookup), hence the complexity when lookups are added are $\sim \frac{n^2}{5m^2} \times \mathcal{L}$ in comparison with the Karatsuba method. As we will show below, $\mathcal{L} \sim 6m$, so that the total complexity of the algorithm is $\sim \frac{n^2}{m}$. Since m can be chosen to be a fraction of n, i.e., $m = \frac{n}{N+1}$, the complexity is $\sim (N + 1)n$. When compared to the Karatsuba technique, this is much quicker than $n^{1.58}$. This is the main result of this short note.

The lookups of m-digit multiplications need to be performed against a table of size $B^m \times B^m$. This lookup, as can be verified by the "divide-and-conquer" technique, is (for $B = 10$) of complexity $\sim \log_2(10^{2m}) = 2m \log_2 10 \sim 6m$. There are some additional additions and subtractions, which add additional (though sub-dominant) complexity $\sim \frac{n}{m}$ as can be easily checked and are detailed in the below example.

Analyzing this further, we could choose to mix and match, i.e., apply k Karatsuba-style divide-by-two-and-conquer steps, then apply the lookup method to look-up 3^k pre-calculated products of $\frac{n}{2^k}$ digits. Alternatively, we could use the above technique (break-up into m digit blocks) with $m = \frac{n}{2^k}$, we'd have to look-up $2^k + \mathcal{C}(2^k, 2)$ products. It is clear that $3^k < 2^k + \mathcal{C}(2^k, 2)$, so a hybrid divide-by-two-and-conquer with lookups algorithm is the quickest way to speed up the calculation. A graph of the reduced complexity (essentially $3^N(\frac{2}{3})^{N-k}$) achieved this way is plotted in Figure 1 - clearly, cutting the recursion off early is advantageous.

A little reflection will show why divide-and-conquer by 2 for k times followed by lookup is the most efficient way to carry out the above procedure. Each time we divide an n-digit number into $N+1$ blocks of m-digits, we have to (recursively) perform $(N+1)+\mathcal{C}(N+1, 2)$ multiplications. After k such recursions, we are left with $(N+1)^k$ blocks of $\frac{n}{(N+1)^k}$ digits each and have to perform $((N+1)+\mathcal{C}(N+1, 2))^k$ multiplications. At this point, if we look up pre-computed products of numbers of this type, that is a complexity factor of $\sim \frac{n}{(N+1)^k}$. The total number of operations is

$$\sim ((N+1)+\mathcal{C}(N+1,2))^k \times \frac{n}{(N+1)^k} = n(1+\frac{N}{2})^k$$

which is smallest for smallest N, i.e., $N = 1$. The complexity then matches exactly the complexity of the Karatsuba algorithm.

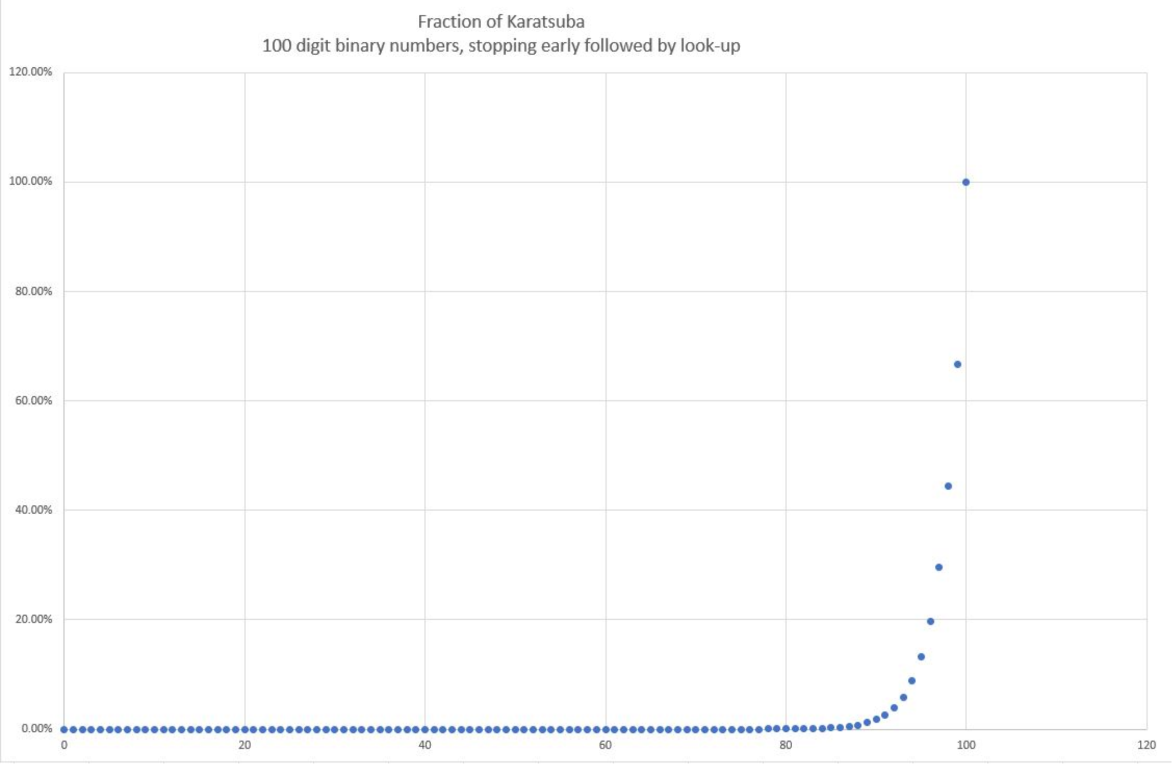
The above arithmetic is demonstrated in the case below for $N = 4, B = 10$, where we have used $n = 5m$,

$$\begin{aligned} x &= x_0 + x_1 B^m + x_2 B^{2m} + x_3 B^{3m} + x_4 B^{4m} \\ y &= y_0 + x_1 B^m + y_2 B^{2m} + y_3 B^{3m} + y_4 B^{4m} \end{aligned} \tag{5}$$

which leads to the product

$$\begin{aligned} x.y = B^0 x_0 y_0 + B^m \left((x_0+x_1)(y_0+y_1) - x_0y_0 - x_1y_1\right) \\ +B^{2m} \left((x_0+x_2)(y_0+y_2) - x_0y_0 - x_2y_2 + x_1y_1\right) \\ +B^{3m} \left((x_0+x_3)(y_0+y_3) - x_0y_0 - x_3y_3 + (x_1+x_2)(y_1+y_2) - x_1y_1 - x_2y_2\right) \\ +B^{4m} \left((x_0+x_4)(y_0+y_4) - x_0y_0 - x_4y_4 + (x_1+x_3)(y_1+y_3) - x_1y_1 - x_3y_3 + x_2y_2\right) \\ +B^{5m} \left((x_1+x_4)(y_1+y_4) - x_1y_1 - x_4y_4 + (x_2+x_3)(y_2+y_3) - x_2y_2 - x_3y_3\right) \\ +B^{6m} \left((x_2+x_4)(y_2+y_4) - x_2y_2 - x_4y_4 + x_3y_3\right) \\ +B^{7m} \left((x_3+x_4)(y_3+y_4) - x_3y_3 - x_4y_4\right) + B^{8m} \left(x_4y_4\right) \end{aligned} \tag{6}$$

FIG. 1. Plot of the Efficiency of cutting off Karatsuba early



This expression has $5 + \mathcal{C}(5, 2) = 15$ independent products that can be pre-computed, i.e.., the 5 simple products $x_0y_0, x_1y_1, x_2y_2, x_3y_3, x_4y_4$ and the 10 combination products $(x_0 + x_1)(y_0 + y_1), (x_0 + x_2)(y_0 + y_2), (x_0 + x_3)(y_0 + y_3), (x_0 + x_4)(y_0 + y_4), (x_1 + x_2)(y_1 + y_2), (x_1 + x_3)(y_1 + y_3), (x_1 + x_4)(y_1 + y_4), (x_2 + x_3)(y_2 + y_3), (x_2 + x_4)(y_2 + y_4), (x_3 + x_4)(y_3 + y_4)$. If these products are found in a pre-computed table of m and $(m+1)$-digit numbers, we would not need any multiplications at all, just 15 lookups, for any m, with $n = 5m$.

We would need to perform additions and subtractions, of course and there are 34 of them in the above example. That number of elementary operations depends, however, only upon $\frac{n}{m}$.

Memory Requirements

Typical RSA encryption algorithms use $\sim$ 1000-digit base-10 composite numbers that are the product of five-hundred-digit primes. If one were to attack the problem by pre-computing keys, i.e., pre-multiplying pairs of five-hundred-digit primes ($n = 500 \sim 2^9$) and storing the results of multiplying all possible 6-digit numbers ($m = 6 \sim 2^3$), one has a complexity

doi:10.20944/preprints201906.0087.v1

$\sim \frac{n^2}{m} = 85n \sim 42,500$, which is worse than the new $\sim n \log_2 n \sim 4500$ complexity [8], albeit the fact that the newer approach also has multipliers, which we have not accounted for. If we use the hybrid method (Karatsuba followed by look-up of 6-digit products), the complexity is $\sim 3^6 \times 6 \sim 4200$, which is arguably much better (no pre-factors missing) than even the $n \log_2 n$ algorithms. We would need to store $\sim 10^{12}$ twelve-digit numbers, roughly 20 TB of memory, which is a very reasonable size.

CONCLUSION

This paper presents a rapid pre-computed approach to speeding up multiplications. Though one needs to pre-compute and store all possible m-digit multiplications, one can compute the products of two integers with number of digits equal to any integer times m in time proportional to the number of digits (times the afore-mentioned integer). Memory is cheaper than CPU-speed, so this is a method that can be exploited in other (for instance signal-processing) situations to speed up intensive calculations too.

Useful conversations are acknowledged with Dr. B. Kumar. As this paper was being prepared, an article about using pre-stored calculations was released, where the Eratosthenes sieve was sped up in calculation complexity [9].

* ramakrishna@physics.rutgers.edu

† kamesh@acm.org

[1] An. Karatsuba and Yu. Ofman (1962). "Multiplication of Many-Digital Numbers by Automatic Computers". Proceedings of the USSR Academy of Sciences. 145: 293–294. Translation in the academic journal Physics-Doklady, 7 (1963), pp. 595–596Nuovo Cimento **B37** (1977).

[2] A. A. Karatsuba (1995). "The Complexity of Computations" (PDF). Proceedings of the Steklov Institute of Mathematics. 211: 169–183. Translation from Trudy Mat. Inst. Steklova, 211, 186–202 (1995)

[3] A. Schönhage and V. Strassen, "Schnelle Multiplikation großer Zahlen", Computing 7 (1971), pp. 281–292

[4] A. Toom. The Complexity of a Scheme of Functional Elements Realizing the Multiplication of

doi:10.20944/preprints201906.0087.v1

Integers. Soviet Mathematics - Doklady, 1963, v. 3, pp 714-716. Originally published in Russian

[5] S. A. Cook. ”On the Minimum Computation Time of Functions”. Ph.D. Thesis. Cambridge, MA: Harvard University, pp. 51-77, 1966.

[6] A. Dutt and V. Rokhlin, ”Fast Fourier transforms for nonequispaced data”, SIAM J. Sci. Comput. 14 (1993), no. 6, 1368 (1393. MR 1241591)

[7] M. F*ü*rer, ”Faster integer multiplication” STOC’07—Proceedings of the 39th Annual ACM Symposium on Theory of Computing, ACM, New York, 2007, pp. 57 (66. MR 2402428 (2009e:68124))

[8] D. Harvey and J. Van Der Hoeven. Integer Multiplication in time $\mathcal{O}(n \times \log n)$, https://hal.archives-ouvertes.fr/hal-02070778

[9] Helfgott, Harald Andrés. https://doi.org/10.1090/mcom/3438 ”An improved sieve of Eratosthenes”