Multiplying Integers with Fourier Transforms

Michael Angel

May 2019

1 Introduction

This paper is about using Fourier transforms to multiply large integers. This idea was introduced by Arnold Schönhage and Volker Strassen in 1971 [1]. Their paper includes two similar algorithms, the first using Fourier transforms with the Cooley-Tukey Fast Fourier Transform (FFT) and the second using number theoretic transforms. The focus of this paper is an implementation of the former, which is both fast and elegant.

Multiplication of large integers has various applications. Projects to search for primes rely on fast integer multiplication, like the Great Internet Mersenne Prime Search. Another application is in computing transcendental numbers; we can use these algorithms to find more digits of π. Various other problems reduce to integer multiplication.

The classical approach to multiplying integers has time complexity $\mathcal{O}(n^2)$. In 1962 Anatoly Karatsuba was the first to discover a faster technique, which calculates multiplication in $\mathcal{O}(n^{\log_2 3})$ time [2]. Schönhage and Strassen's algorithm improves on this time, and they conjectured in 1971 the existence of an $\mathcal{O}(n\log n)$ integer multiplication algorithm.

On March 18th, 2019, David Harvey and Joris Van Der Hoeven published an algorithm that achieved $\mathcal{O}(n\log n)$ running time [3]. Their technique builds on the Schönhage-Strassen algorithm but instead of performing a FFT over the complex numbers or finite integer rings they instead transform over certain multivariate polynomial rings where multiplication is performed by additions and subtractions in the complex numbers and thus can be computed quickly.

This paper will go on to examine using the discrete Fourier transform (DFT) to multiply integers, but, interestingly, the DFT over the complex numbers reduces to multiplication of large integers [4], so we can consider the reverse: using fast integer multiplication to compute DFT. Indeed Harvey and Van Der Hoeven's $\mathcal{O}(n\log n)$ integer multiplication algorithm can theoretically produce an algorithm to calculate DFT over $\mathbb{C}$ faster than the the traditional FFT [3]. However, so far, Harvey and Van Der Hoeven's algorithm only realizes its superior running time with numbers that are larger than can be of practical use.

2 Schönhage–Strassen Algorithm

The Schönhage–Strassen algorithm proceeds first by considering integers as polynomials of their bases. For example:

$$\mathbf{4092} = \mathbf{2}(10)^0 + \mathbf{9}(10)^1 + \mathbf{0}(10)^2 + \mathbf{4}(10)^3$$
$$\mathbf{373} = \mathbf{3}(10)^0 + \mathbf{7}(10)^1 + \mathbf{3}(10)^2 + \mathbf{0}(10)^3$$

The coefficients of the product polynomial is a convolution of the coefficients of the factor polynomials. Therefore, given two input polynomials the goal is to compute the convolution which can be done in the Fourier domain by a point-wise multiplication.

The Schönhage–Strassen algorithm proceeds by converting the input polynomials to the Fourier domain using the FFT, then conducting a point-wise multiplication, and finally an inverse FFT to yield the product polynomial.

It remains to process the resulting polynomial back to an integer by carrying values that exceed the base to their adjacent values. This carry operation is illustrated by our example:

$4092 \cdot 373$
$\big(2(10)^0 + 9(10)^1 + 0(10)^2 + 4(10)^3\big) \cdot \big(3(10)^0 + 7(10)^1 + 3(10)^2 + 0(10)^3\big)$
$= 6(10)^0 + 41(10)^1 + 69(10)^2 + 39(10)^3 + 28(10)^4 + 12(10)^5 + 0(10)^6$
Here we perform the **carry operation** to obtain digits between 0 and 9:
$= 6(10)^0 + 1(10)^1 + 3(10)^2 + 6(10)^3 + 2(10)^4 + 5(10)^5 + 1(10)^6$
$\Rightarrow 4092 \cdot 373 = 1526316$

To summarize, the Schönhage–Strassen algorithm can be described in 4 steps:
Step 1: Compute the Fourier transform of the inputs using FFT
Step 2: Point-wise multiply the transformed inputs
Step 3: Take the inverse Fourier transform of the product
Step 4: Perform the carry operation

The accuracy of this algorithm hinges on how many bits we use to precisely represent the elements of $\mathbb{C}$. Schönhage and Strassen proved that complete accuracy requires $\mathcal{O}(\log n)$-bit numbers [1].

To analyze the running time of this algorithm consider that the FFT performs $\mathcal{O}(n\log n)$ multiplications, where n is maximum number of digits of the inputs. We can describe the running time, $M(n)$ recursively:
$M(n) = \mathcal{O}(nM(n')) + \mathcal{O}(n\log n), \quad n' = \mathcal{O}(\log n)$
Explicitly this is $\mathcal{O}(n\log n \log\log n \cdots 2^{\mathcal{O}(\log^* n)})$, where "$\log^* n$" is the number of times you take log of n until you are less than or equal to 1.

3 Number Theoretic Transforms

Schönhage and Strassen proposed a second algorithm in 1971 using number theoretic transforms that has a better running time. This algorithm is actually a modified version of the first. It is slightly more complicated but adds a lot to the framework of understanding for integer multiplication algorithms.

Consider the discrete Fourier transform, from $\mathbb{C}^N$ to itself. This transform is linear and bijective. We can characterize it as the following matrix:

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_N & \omega_N^2 & \cdots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \cdots & \omega_N^{2(N-1)} \\ \vdots & & & & \\ 1 & \omega_N^{n-1} & \omega_N^{2(n-1)} & \cdots & \omega_N^{(N-1)^2} \end{bmatrix}$$

Where ω_N is an Nth root of unity.

The key insight is that the discrete Fourier transform can be generalized to work over any ring. The matrix representation above would just have ω_N as a principal Nth root of unity in the ring. Schönhage and Strassen used the ring $\mathbb{Z}/m$, where $m = 2^k + 1$ for some k. This specific number transform is called a Mersenne Number Transform.

Instead of controlling precision by adjusting bit length for representation of numbers in $\mathbb{C}$, this algorithm chooses k high enough to accommodate the number of digits. Working in the finite ring limits the number of roots of unity calculations and the running time overall is $\mathcal{O}(n\log n\log\log n)$.

4 Conclusion

The Schönhage-Strassen algorithm is a powerful application of Fourier transforms. Implementations of this algorithm are still state of the art in computing large integers; the GNU Multi-Precision Library uses Schönhage-Strassen to compute numbers with digits between about 33,000 to 150,000 [5]. The basic ideas from the seminal 1971 paper, using the FFT and point-wise multiplication in the Fourier domain, are still motivating current research as seen in Harvey and Van Der Hoeven's recent breakthrough. In the future we will likely see further advancements that allow for practical and fast algorithms for the multiplication of large integers.

5 References

[1] A. Schönhage and V. Strassen, *Schnelle Multiplikation grosser Zahlen*, Computing (Arch. Elektron. Rechnen) 7 (1971), 281–292.

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

[3] David Harvey, Joris Van Der Hoeven. Integer multiplication in time O(n log n) (2019).
https://hal.archives-ouvertes.fr/hal-02070778/document

[4] David Harvey, Joris Van Der Hoeven, *On the complexity of integer matrix multiplication*, J. Symbolic Comput. 89 (2018), 1–8.

[5] "MUL_FFT_THRESHOLD". GMP developers' corner, 24 November 2010.
https://web.archive.org/web/20101124062232/http://gmplib.org/devel/MUL_FFT_THRESHOLD.html