QI: Integer Computation: Factoring, Multiplication

Gregory Croisdale, Cade Brown, and Rebecca Ryan May 6, 2020

This paper was written for PHYS494 at the University of Tennessee, Knoxville taught by George Siopsis in Spring, 2020.

Contents

1 Abstract

In this paper, we discuss some of the historically important quantum algorithms involving integers and their factors. We compare the feasibility and speed of some of the most impressive classical algorithms with our own implementation of the well-discussed "breaker of RSA": Shor's Algorithm[4,5]. We describe attempts and problems with implementing the Lucas-Lehmer Primality Test[6] and the Fourier Transform for Multiplication[7].

2 Accompanying Materials

The source code of all the algorithms we refer to in this paper can be found and ran from this GitHub Project:

`https://github.com/gcrois/QFastInteger`.

A static version of the demo page is available for viewing at the following address:

`https://gcrois.github.io/QFastInteger/demo.html`.

3 Introduction

3.1 The Importance of Primes

[4][5][6] The current computer systems we have are due to the prime numbers. Because prime numbers only have themselves and one as their factors, they are unique in the way that they cannot be broken down further while remaining a stable integer. Therefore, prime numbers serve as the basis of encryption schemes, because they few and far between in comparison to composite numbers.

3.2 Encryption

[4][6][9][21] Encryption schemes are based heavily on prime numbers due to the fact that any two prime numbers multiplied together will ascertain a quick result, however, the reverse of this process is very labor intensive. In order for the safest encryption, these primes must be truly random. However, the prime numbers chosen must be somewhat related. This relationship does decrease the security of the encryption, but is necessary when considering asymmetric cryptography.

3.3 Error Correction

[4][6][21][25] Error correction is required to ensure that the recipient receives uncorrupted data, that is, the original message has not been compromised. In quantum error correction, the main culprits are decoherence and noise, which must be addressed in order for the system to be completely fault-tolerant. Classically, this is done by redundancy, however, due to the no-cloning theorem [25] this is impossible because quantum computing processes are reversible.

3.4 The Importance of Factoring

[4][6] Factoring is important in order to determine the possible prime numbers p and q that were used in encrypting the original data, N. In order to determine p and q, all of the prime factors of N must be found. This is not a trivial process as p and q tend to be very large, in addition to showing an almost nonexistent correlation. This could be done with more than two prime numbers, but for simplicity's sake only two have been considered.

3.4.1 Greatest Common Denominator

The greatest common denominator (gcd) of a set of integers is the largest number that divides evenly into all integers without a remainder. For example, $gcd(35, 50) = 5$, where each factor can be broken down into smaller parts $(35 = 5^1 * 7; 50 = 5^2 * 2)$. This is very important for decryption, as the gcd will help in breaking down the possible prime numbers that were used to encrypt data in the first place. When prime numbers are raised to powers of prime numbers, that decreases the safety of the encryption which is why it is preferable to choose large prime numbers, without prime exponents, which are much harder to factor.

3.4.2 Least Common Multiple

The least common multiple (lcm) is the smallest number that is divisible by two integers. Specifically, it is the smallest number of prime factorizations of the gcd. Taking the lcm and expressing it as prime powers is a way to calculate prime factorizations that would otherwise be harder to decrypt. This follows the same logic as for determining a gcd, where too many prime options are discouraged due to ease of factorizability.

3.5 Importance of Fourier Transform

[8][22][15][14] The Fourier Transform (FT) is a mathematical relationship that allows for functions to be broken down into frequencies of sines and cosines, represented by a Fourier Series. The combination of two functions f and g will produce two new functions (h and j), depending on whether f acts on g or g acts on f where f and g are sine or cosine functions. The resultant is a transformation of the functions acting upon each other.

3.5.1 Convolution

Convolution is where two functions acting upon each other produce a new function, based on which function is acting on the other. This is how the discrete fourier transform (DFT) works, as a pointwise product of their signals. The quantum fourier transform (QFT) does not work in this way due to entanglement, and that this is not a reversible process.

3.5.2 Trigonometric Parsing

Trigonometric parsing is how the FT is able to break down a function into sines and cosines, which then require sound analysis of the Fourier Series. This allows for each component to be broken down into fundamental harmonics, where each is a pythagorized set creating a new harmonic, known as the spectrum of the signal. Each note is a combination of sines and cosines, of which vibrate at a specific frequency and allow the Fourier coefficients to be determined thus giving the amplitude of a longitudinal sound wave. Each transformed peak corresponds to a specific amplitude on an FT graph, which gives a clear picture of the wave function required to create a specific sound.

3.5.3 Image Processing

The discrete fourier transform (DFT) takes a 2-Dimensional image, and calculates the product of the spatial domain and basis set. The sum of all of these products will recreate the transformed image, which can be returned to the original image using the inverse fourier transform (IFT). Increasing the dimensions of FT calculations will decrease the number of necessary computations.

3.6 Shor's Algorithm

Shor's Algorithm takes an input number N, and finds the prime factors (p, q) of N. This implements polynomial time QFT to factor these integers. Should a quantum computer be able to handle the stress, it would hypothetically be able to break RSA encryptions that are currently in place. N must be odd, because N/2 should not be able to be broken down into powers of 2 or else there would be too many computational wrap-arounds without substantial remainders. If N were to be even, the encryption would be much easier to break due to similarity in prime factors.

3.7 Lucas-Lehmer Prime Check

The Lucas-Lehmer (LL) test is able to determine whether or not a Mersenne number $(2^n - 1)$ is a prime number where n must also be prime. Because prime factoring is a very slow process, the guess and check method for determining a Mersenne prime is much faster, and thus saves time in prime finding. This generation of prime numbers increases the speed in which prime numbers are determined, albeit not necessarily the prime factors of N specifically. However, the LL test will bring us that much closer to determining possible prime factors.

3.8 RSA Encryption

RSA Encryption is used in data transfers, where two keys are made, one private and one public. As an asymmetric process, this cryptography is very hard to break without the private key. To create the public key (N = pq), where p and q are two prime numbers with similar magnitudes but not easily related. Computing Carmichael's totient function on the public key where lambda(N) = lcm(lambda(p), lambda(q)), and then choose a value of e such that 1 < e < lambda(N) has a gcd(e, lambda(N) = 1), and thus e and lambda(N) are coprime. Now, the private key is simply the modular inverse of e(mod(lambda(N)) = d. Should anyone try and intercept this information en route to the recipient, they would have to know the encrypted message (E) that was sent, where E = 1modN. In addition, another key (C) would have to be given, where E(1/C(mod(d))) = R. Solving for d, the intercepter would be able to determine the original encrypted message.

4 Implementation of Shor's Algorithm for Factoring Integers

Shor's algorithm is perhaps one of the most anticipated quantum algorithms, for the risk it poses to encryption (see below in the document), as well as the interesting result that drastically improves the computational complexity of integer factorization, a very interesting problem.

This algorithm will be interesting to implement, because it uses elements such as QFT, as well as partial computation on classical and quantum systems, which combine to solve the problem in polynomial time $O((\log n)^2(\log\log n)(\log\log\log n)$.

While probabilistic algorithms already exist for primality testing on classical computers, finding out which numbers divide another number is still a very hard to solve problem using solely classical computers.

Shor's algorithm: https://qudev.phys.ethz.ch/static/content/QSIT15/Shors%20Algorithm.pdf

5 Breaking RSA

6 Lucas-Lehmer Primality Test

7 Fast Fourier Transform (FFT) Based Integer Multiplication

Integers can be multiplied quickly using FFT (Fast Fourier Transform) and convolution along their digits. This provides a monumental speedup as opposed to the naive complexity of $O(N^2)$ and some other algorithms (Katatsuba: $O(N^{1.58})$). Specifically, the FFT can yield theoretical algorithms of complexity $O(N\log N)$, a monumental speedup at larger values.

Integer multiplication of 2 integers (A and B), which have base-d 'digits' (not necessarily base 10) $a_0, ..., a_{N-1}$ such that $\sum_{i=0}^{N-1} a_i d^i = A$ and $b_0, ..., b_{N-1}$ such that $\sum_{i=0}^{N-1} b_i d^i = B$ have the result (we'll call it C) with $2 * N$ 'digits'.

The formula for any given digit of 'C' (not accounting for overflow) is: $C_i = \sum_{j=0}^{i} a_j c_{i-j} d^i$

Thus, in the following vector-vector product: ab^T:

$$\begin{bmatrix} a_0b_0 & a_0b_1 & ... & a_0b_{N-1} \\ a_1b_0 & a_1b_1 & ... & a_1b_{N-1} \\ ... & ... & ... & ... \\ a_{N-1}b_0 & a_{N-1}b_1 & ... & a_{N-1}b_{N-1} \end{bmatrix}$$

Each cross-diagonal sum will give the corresponding entry of c

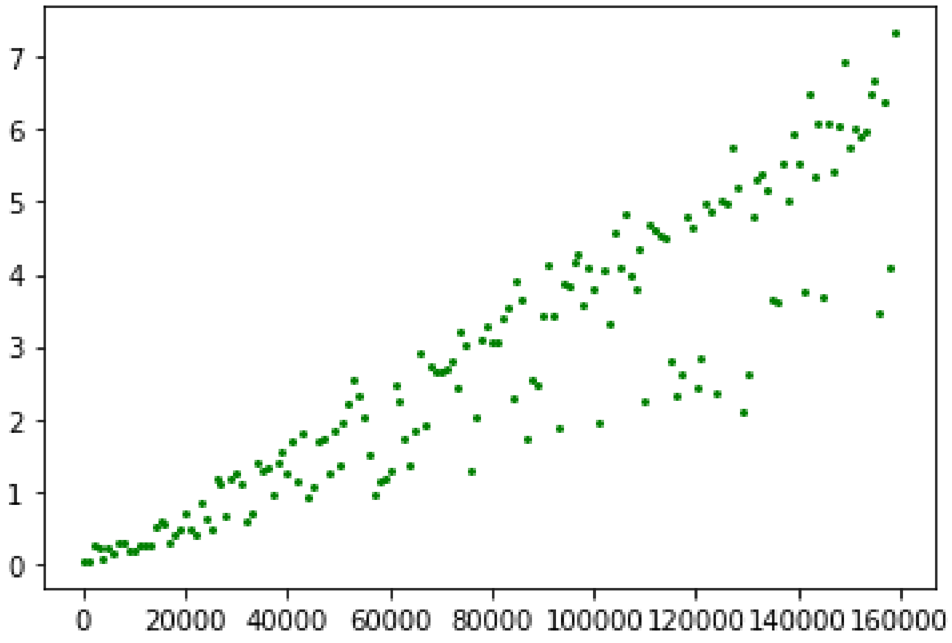
Thus, $c_0 = a_0b_0$ and $c_1 = a_0b_1 + a_1b_0$, and so on

Obviously, directly summing these entries yields a complexity of $O(N^2)$. However, there is a quicker way to do convolution for two 1-dimensional sequences, using the Fast Fourier Transform (FFT).

The formula is: $A * B = \text{IFFT}(\text{FFT}(a) \odot \text{FFT}(b))$

Where IFFT is the Inverse Fast Fourier Transform, and $\odot$ means pointwise multiplication. Using the Cooley-Tukey FFT algorithm (standard, and plenty of packages already implement this), the complexity is reduced to $O(3N \log N + N) = O(N \log N)$ (since point-wise multiplication of small elements is $O(N)$). This is the drastic speedup that allows faster integer multiplication.

We implemented this in Python (so, there were inefficiencies), but we did get an approximately correct complexity of runtime (albeit much slower than would be possible in C or some lower language). We show the time (in seconds) required for an NxN bit multiplication (N=0 through 160000)



We also tried to implement it as a Quantum circuit. However, we soon researched and found out that this is impossible for the general case [1] because of the pointwise multiplication of the FFT results. The reason for this is that a quantum gate that could perform multiplication of the state of 2 qubits would be non-unitary and thus irreversable, and therefore would not work. So, this method of integer multiplication is impossible to implement on a quantum computer.

8 Conclusion

We will visually demonstrate the fastest classical computing methods to the aforementioned problems in both a Jupyter notebook and an HTML5 website for easy accessibility. We will animate the algorithms using Google Charts and a small input.

References

[1] Lomont, C. *Quantum convolution and quantum correlation algorithms are physically impossible*.

`https://www.researchgate.net/publication/2191142_Quantum_convolution_and_quantum_correlation_algorithms_are_physically_impossible. 2003.`

[2] Afflick, D. *"Parity Error Checking"*. 2020.

[3] Ballard, D, Brown, C. Computer Vision*, Prentice-Hall, 1982, pp 24-30.*

[4] BerkeleyX. *"Chapter 5: QFT, Period Finding, Shor's Algorithm", pp 49-68.*.

[5] Crandall, R, Pomerance, C. *"Prime Numbers: A Computational Perspective" Springer 2001, pp 191-226.*.

[6] de Wolf, R. *"Quantum Computing Lecture Notes", 2012, pp 19-29*.

[7] Donis-Vela, A, Garcia-Escartin, J.C. *"A quantum primality test with order finding" Quantum Information and Computation, 2018, Vol 18, 13-14*.

[8] Fourier, JBJ. *"The Analytical Theory of Heat" University Press, 1878, pg. 166-209*.

[9] Goldreich, O. *Digital Image Processing, Addison-Wesley Publishing Company, 1992, pp 81-125.*

[10] Gonzales, R, Woods, R. *"Parity Error Checking"*.

[11] Harnett, K. *"Mathematicians Discover the Perfect Way to Multiply" Quanta Magazine, 2019*.

[12] Horn, B. *Robot Vision, MIT Press, 1986, Ch. 6-7*.

[13] Jain, A. *Fundamentals of Digital Image Processing, Prentice-Hall, 1989, pp 15-20.*

[14] Katznelson, Y. *"An Introduction to Harmonic Analysis" 1976.*

[15] Keller, W *An Introduction to Probability Theory and its Applications, 1971, Vol 2*.

[16] Kessler, B *"A "Sound" Approach to Fourier Transforms: Using Music to Teach Trigonometry" 2007 Bridges Donostia Conference Proceedings, 2007, 135-142*.

[17] Li, B, Babu, GJ *"Convolution Theorem and Asymptotic Efficiency" A Graduate Course on Statistical Interference, Springer. 2019, pp. 259-327*.

[18] Lin, FX *"Shor's Algorithm and the Quantum Fourier Transform" Quantum Mechanics and Quantum Computation, 2016, pp 1-16*.

[19] Lomont, C. *"Quantum convolution and quantum correlation algorithms are physically impossible" Quantum Physics, 2003, pp 1-10*.

[20] Marion, A. *An Introduction to Digital Image Processing, Chapman and Hall, 1991, Ch. 9.*

[21] Nielsen, MA, Chuang, IL. *"Quantum Computation and Quantum Information", Cambridge University Press, 2000*.

[22] Osgood, B. *"The Fourier Transform and its Applications" Stanford Univ, 2007.*

[23] Rivest, R, Shamir, A, Adleman, L. *"A Method for Obtaining Digital Signatures and Public-Key Cryptosystems" Communications of the ACM, 1978, pp 120-126*.

[24] Shor, P. *"Algorithms for quantum computation: discrete logarithms and factoring", 35th Annual Symposium on Foundations of Computer Science, 1994, 124-134*.

[25] Wootters, W, Zurek, W. *"A Single Quantum Cannot be Cloned", Nature 1982, pp 802-803* 2020.