Introduction

This book is about one of the most important ideas in quantum information: Shor’s algorithm, a quantum algorithm that can factor large integers and solve discrete logarithms in polynomial time on an ideal fault-tolerant quantum computer (Shor, 1997). That sentence contains several ideas that we will build carefully from the beginning. For now, the main point is simple:

> Shor’s algorithm shows that a quantum computer can use wave-like interference to discover hidden periodic structure, and that hidden structure can reveal the factors of an integer.

The goal of this book is not only to show you the steps of the algorithm. The goal is to help you understand why the steps are natural. By the end, Shor’s algorithm should not feel like a magic trick. It should feel like a design pattern: encode a mathematical structure into a quantum state, make the useful global pattern interfere constructively, measure information about that pattern, and finish with classical computation.

We begin with the problem Shor solves.

Suppose someone gives you the number

$$N = 15.$$

Factoring means writing N as a product of smaller integers greater than 1:

$$15 = 3 \times 5.$$

For 15, this is easy. For a number with hundreds or thousands of decimal digits, no efficient classical factoring algorithm is known. “Efficient” here means that the running time grows like a polynomial in the number of input bits, rather than growing exponentially. This distinction is central in computational complexity theory, the mathematical study of how computational resources such as time and memory grow with input size.

Factoring matters because some widely used public-key cryptographic systems, especially RSA, base their security on the practical difficulty of factoring large composite integers. In RSA, a public key contains a large number N = pq, where p and q are large primes; knowing N is public, but knowing p and q allows one to compute the private key information (Rivest, Shamir, and Adleman, 1978). Shor's algorithm does not merely make factoring a little faster. In the standard theoretical model of quantum computation, it changes the scaling: it gives a polynomial-time quantum method for integer factoring (Shor, 1997).

That is why Shor's algorithm is famous. But fame can hide the actual idea. The algorithm is not "try all factors at once." That phrase is misleading. A quantum computer can create a superposition, which is a state described by many amplitudes at once, but measurement does not simply reveal all those amplitudes. If we naïvely place many candidate answers in superposition and then measure, we usually see only one random candidate. The real power comes from interference: amplitudes can add or cancel, like waves. A quantum algorithm must arrange the computation so that wrong or unhelpful information cancels and useful global information becomes likely to observe.

For Shor's algorithm, the useful global information is a period.

A period is a repeating interval in a pattern. For example, consider the sequence

$$1,\ 2,\ 1,\ 2,\ 1,\ 2,\ldots$$

This sequence has period 2, because it repeats every two steps. A slightly less obvious example comes from modular arithmetic. Modular arithmetic is arithmetic with remainders. When we write

$$17 \equiv 2 \pmod{15},$$

we mean that 17 and 2 have the same remainder after division by 15.

Now choose a = 2 and look at powers of 2 modulo 15:

$$2^0 \bmod 15 = 1,$$

$$2^1 \bmod 15 = 2,$$

$$2^2 \bmod 15 = 4,$$

$$2^3 \bmod 15 = 8,$$

$$2^4 \bmod 15 = 1.$$

After four steps, the value returns to 1, and then the pattern repeats:

$$1,\ 2,\ 4,\ 8,\ 1,\ 2,\ 4,\ 8, \ldots$$

The period is 4. In number theory, this period is called the order of 2 modulo 15. More generally, if a and N share no common factor except 1, the order of a modulo N is the smallest positive integer r such that

$$a^r \equiv 1 \pmod{N}.$$

This number r can help us factor N. In the example above, r = 4, so

$$2^{r/2} = 2^2 = 4.$$

Then

$$4^2 \equiv 1 \pmod{15},$$

which means

$$4^2 - 1 \equiv 0 \pmod{15}.$$

Factoring the left side gives

$$(4-1)(4+1) = 3 \times 5.$$

Now compute greatest common divisors:

$$\gcd(3, 15) = 3,$$

$$\gcd(5, 15) = 5.$$

We have found the factors of 15. This small example is not the whole algorithm, but it reveals the central reduction:

> Factoring can often be turned into the problem of finding the period of the function >

$$> f(x) = a^x \bmod N. >$$

Shor's quantum contribution is an efficient way to find that period.

The word "quantum" enters because a quantum computer uses the mathematics of quantum mechanics to process information. In ordinary classical computation, the basic unit of information is a bit, which is either 0 or 1. In quantum computation, the basic unit is a qubit. A qubit can be in a state described as a linear combination of the basis states $|0\rangle$ and $|1\rangle$:

$$\alpha|0\rangle + \beta|1\rangle.$$

The numbers α and β are called amplitudes. They are usually complex numbers, meaning they can include the imaginary unit i, where $i^2=-1$. When a qubit is measured in the standard basis, the probability of seeing 0 is $|\alpha|^2$, and the probability of seeing 1 is $|\beta|^2$, with

$$|\alpha|^2 + |\beta|^2 = 1.$$

This rule is part of the standard mathematical model of quantum computation (Nielsen and Chuang, 2010). The important lesson for now is that amplitudes are not probabilities themselves. Amplitudes can be positive, negative, or complex, and they can interfere before measurement. Probabilities appear only after we square their magnitudes.

For example, the Hadamard gate, one of the most important one-qubit quantum gates, transforms basis states as follows:

$$|0\rangle \mapsto \frac{|0\rangle + |1\rangle}{\sqrt{2}},$$

$$|1\rangle \mapsto \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

The minus sign matters. If two computational paths lead to the same final state with opposite amplitudes, they can cancel. If they lead with matching phases, they can reinforce. Shor's algorithm is built around arranging exactly this kind of reinforcement and cancellation, but across many computational states at once.

The main quantum tool in Shor's algorithm is the Quantum Fourier Transform, or QFT. The ordinary discrete Fourier transform is a mathematical operation that rewrites data in terms of frequencies. For example, a sound wave can be described either by its pressure at each moment in time or by the frequencies that make it up. The QFT is the quantum version of this transformation: it changes the amplitudes of a quantum state so that periodic structure becomes visible as frequency information. Shor's algorithm uses the QFT to turn the hidden period of a^x bmod N into measurement outcomes from which the period can be reconstructed (Shor, 1997; Ekert and Jozsa, 1996).

This book will build that statement slowly. We will not assume you already know quantum mechanics. We will start with linear algebra: vectors, complex numbers, inner products, tensor products, and matrices. Then we will use that language to define qubits, quantum gates, circuits, and measurement. On the number theory side, we will build modular arithmetic, greatest common divisors, Euler's theorem, and multiplicative order. These are the two streams that meet in Shor's algorithm: quantum interference and arithmetic periodicity.

A useful way to preview the full algorithm is this:

1. Choose a random number a modulo N.
2. If a already shares a nontrivial factor with N, we are done.
3. Otherwise, consider the periodic function

$$f(x) = a^x \bmod N.$$

4. Use a quantum circuit to learn information about the period r of this function.

5. Use classical postprocessing, especially continued fractions and greatest common divisors, to extract factors of N.

The quantum part does not directly output the factors. It outputs information that is likely to reveal the period. The period then becomes useful through classical number theory. This cooperation between quantum and classical computation is a major theme of the book. Shor’s algorithm is not purely quantum in the sense of replacing all classical computation. It is a hybrid algorithm: quantum computation supplies a kind of global information that appears hard to obtain classically, and classical computation verifies and uses that information.

This distinction is important for another reason. If you want to use the same principles for other problems, you should not copy the surface details of factoring. You should copy the structure:

- Find a mathematical object with hidden regularity.
- Encode that object into a reversible computation.
- Prepare a superposition over many inputs.
- Compute the function coherently, meaning without measuring midway.
- Use interference, often through a Fourier transform, to reveal global structure.
- Measure and then perform classical postprocessing.

This pattern appears most clearly in the hidden subgroup problem, a framework in which a function hides a subgroup of a group by being constant on certain structured sets called cosets and different on different cosets. Shor’s factoring and discrete logarithm algorithms can be understood as examples of efficient quantum algorithms for abelian hidden subgroup problems, where “abelian” means the group operation commutes: xy = yx for all group elements x and y (Nielsen and Chuang, 2010). We will return to this viewpoint near the end of the book, after the concrete algorithm is fully understood.

There is also a practical boundary to keep in mind. Shor's algorithm is a theorem about what can be done in the standard model of quantum computation, and it has deep consequences for cryptography. But factoring cryptographically relevant integers would require large, reliable, error-corrected quantum computers. Physical quantum devices are affected by noise, decoherence, and imperfect gates. The existence of Shor's algorithm does not mean that every current quantum device can break RSA. It means that if scalable fault-tolerant quantum computers are built, then cryptographic systems whose security depends on factoring or discrete logarithms would be vulnerable in a way they are not to known classical algorithms (Shor, 1997; Nielsen and Chuang, 2010).

So the right attitude is neither panic nor dismissal. The right attitude is understanding.

Shor's algorithm is worth studying because it teaches three lessons at once. First, it shows how quantum mechanics changes the theory of computation. Second, it reveals a beautiful connection between number theory and Fourier analysis. Third, it provides a reusable method for quantum algorithm design: look for hidden structure, encode it coherently, and use interference to make that structure observable.

We now begin that path from the ground up. The next chapter frames the computational problem of factoring: what it asks, why it matters, how it differs from related problems, and why hidden periodicity is the doorway through which quantum computation enters.

References

Ekert, A., and Jozsa, R. (1996). Quantum computation and Shor's factoring algorithm. Reviews of Modern Physics, 68(3), 733–753.

Nielsen, M. A., and Chuang, I. L. (2010). Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), 120–126.

Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing, 26(5), 1484–1509.

Document information

Introduction

Project	Shor's Algorithm from First Principles
Document	Document 1.4
Author	mujirin
Verifier	Not verified
Downloaded	July 05, 2026 19:15 KST
Status	Working
Document link	https://www.theorytrace.com/projects/shors-algorithm-from-first-principles/documents/introduction/