Shor's Algorithm

2025-07-06

Shor's algorithm represents one of the most groundbreaking achievements in quantum computing, demonstrating that quantum computers could theoretically break widely-used cryptographic systems. This polynomial-time quantum algorithm can efficiently factor large integers by exploiting quantum superposition and interference to find the period of modular exponentiation.

blog: https://tetraquark.vercel.app/posts/quantum_shor/?src=pdf

email: quarktetra@gmail.com

Basic Elements of Quantum Algorithms

The most basic element of a QC is a quantum bit, qubit for short, which is a two-level quantum system. It spans a two dimensional Hilbert space denoted as H_2. H_2 is equipped with a fixed basis $(|0\rangle, |1\rangle)$, a so-called computational basis. States $|0\rangle$ and $|1\rangle$ are called the basis states. A general state of a single quantum bit is a vector that can be written as:

$$c_0|0\rangle + c_1|1\rangle, \tag{1}$$

where $|c_0|^2 + |c_1|^2 = 1$

We can extend this definition to multiple qubits: for example, a system of two qubits describes a four-dimensional Hilbert space $H_4 = H_2 \otimes H_2$ having orthonormal basis $(|00\rangle, |01\rangle, |10\rangle, |11\rangle)$. A state of a two-qubit system is a unit-length vector

$$c_0|00\rangle + c_1|01\rangle + c_2|10\rangle + c_3|11\rangle, \tag{2}$$

with $|c_0|^2 + |c_1|^2 + |c_2|^2 + |c_3|^2 = 1$.

One of the most important gates in QC is the Hadamard gate, denoted by H, and it is defined as follows:

$$H|\mathbf{x}\rangle = \frac{1}{\sqrt{2}} \sum_{\mathbf{y}} (-1)^{\mathbf{x}.y} |\mathbf{y}\rangle \tag{3}$$

Applying H to the computational basis we get

$$\begin{aligned} |0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \\ |1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \end{aligned} \tag{4}$$

Hadamard gate basically creates superpositions out of pure states, and it can also be written in the matrix form as follows:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{5}$$

Using Hadamard transformations along with phase shift operations, one can implement quantum Fourier transform (QFT), which is basically the classical discrete Fourier transform applied to the quantum state vector:

$$QFT|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=1}^{N-1} e^{\frac{-2\pi i x.y}{N}} |y\rangle \tag{6}$$

This transformation is the key element in Shor's factorization algorithm as we will discuss below.

Shor's Algorithm

Factoring a large number, N, into its primes is a hard problem. In the 1970s, it was shown that factorization can be mapped into a period finding problem, which is also a hard problem, and there are no known classical algorithms that can do this computation efficiently. However, the period finding problem has the obvious structure of periodicity, and quantum computers can make use of this internal feature of the problem to yield exponential speed up over classical algorithms.

Below are the steps of the factorization algorithm:

- You pick a random number a which is smaller than $\sqrt{N}$.
- Calculate the *period* of a, denoted by r, so that $a^r - 1$ is a multiple of N, i.e. $a^r = 1 \text{ Mod } N$
  - This means $(a^{r/2} - 1)(a^{r/2} + 1)$ is a multiple of N.

  - Therefore $a^{r/2} \pm 1$ and N have common divisors.

- Calculate GCD$(N, a^{r/2} \pm 1)$, GCD being greatest common divisor.

Except for the computation of the period r, there are very efficient methods to execute the algorithm above, and the period finding part is exactly where Shor's algorithm is applied.

Here are the steps of Shor's quantum algorithm to compute the period of a number a:

1. Select the smallest integer q satisfying $N^2 < Q < 2N^2$ where $Q = 2^q$,
2. Prepare the input register as a uniform superposition of numbers 0 to $Q-1$:
   - $|s\rangle = \frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x\rangle$.
3. Append the ancillary bit $|f(x)\rangle = |a^x \operatorname{Mod} N\rangle$ to get the composite state as:
   - $\frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x, f(x)\rangle$.
4. Apply the inverse QFT to the input register only (i.e. exclude the ancillary bit):
   - $\frac{1}{Q} \sum_{x=0}^{Q-1} \sum_{y=0}^{Q-1} e^{\frac{2\pi i x y}{Q}} |y, f(x)\rangle$
   - The first thing we note is the periodicity and the range of f. As the index x runs from 0 to $Q-1$, $f(x)$, executing Mod N operation, will run from 0 to $N-1$. So we can reorder the summation over x with respect to the output $f(x)$, which we will name as z for simplicity. So $\sum_{x=0}^{Q-1} = \sum_{z=0}^{N-1} \sum_{x \in \{0,1,\cdots,Q-1\}; f(x)=z}$
   - x ranges from 0 to $Q-1$, and let's mark the smallest value of x that satisfies the relation $f(x) = z$ as x_0. Due to the periodicity of f, the total number of instances of x that will satisfy $f(x) = z$ is $\lfloor \frac{Q-x_0-1}{r} + 1 \rfloor$. Let's label these x values with a dummy index b. Essentially we are mapping x to $x_0 + rb$ to write the summation as:
   - $\sum_{x \in \{0,1,\cdots,Q-1\}; f(x)=z} e^{\frac{2\pi i x y}{Q}} = \sum_{b=0}^{\lfloor \frac{Q-x_0-1}{r} \rfloor} e^{\frac{2\pi i y (x_0 + rb)}{Q}} = e^{\frac{2\pi i x_0 y}{Q}} \sum_{b=0}^{\lfloor \frac{Q-x_0-1}{r} \rfloor} e^{\frac{2\pi i r b y}{Q}}$
5. Make a measurement on the ancillary bit. This will result in an integer z. The input register state will collapse into a superposition in the subspace of x values that satisfies $f(x) = z$, which is what we have calculated above.
   - This is a superposition of many states, which will cause interference. If the phase factors $e^{\frac{2\pi i r b y}{Q}}$ align, it will be constructive interference. For the phase factors to be aligned as they are summed over with the index b, it is required to have $e^{\frac{2\pi i r y}{Q}}$ to be close to the real axis, i.e. $\frac{ry}{Q}$ needs to be close to some integer c. When we make a measurement on the input register state, due to the constructive interference, we will most probably measure a value of y such that $\frac{ry}{Q}$ will be close to an integer.
6. Perform classical continued fraction expansion:

- So we have the measured value of y, and we know the value of Q since we set it at the beginning of the algorithm. Therefore, we know the value $\frac{y}{Q}$. We also know that $\frac{yr}{Q}$ needs to be close to an integer c, which implies that $\frac{y}{Q}$ is close to $\frac{c}{r}$. What we need to do is to express $\frac{y}{Q}$ as a fraction $\frac{d}{s}$ with the conditions $s < N$ and $\left|\frac{y}{Q} - \frac{d}{s} < \frac{1}{2Q}\right|$. This computation can be executed efficiently by classical algorithms.

7. The value of s is very likely to be the period r we are looking for, and we can verify this quickly by computing if $a^s = 1 \,\mathrm{Mod}\, N$. If so, we have successfully computed the period, otherwise we try other candidates $\frac{d}{s}$ around $\frac{y}{Q}$. If none of them works, we go back to step 1 and start over.