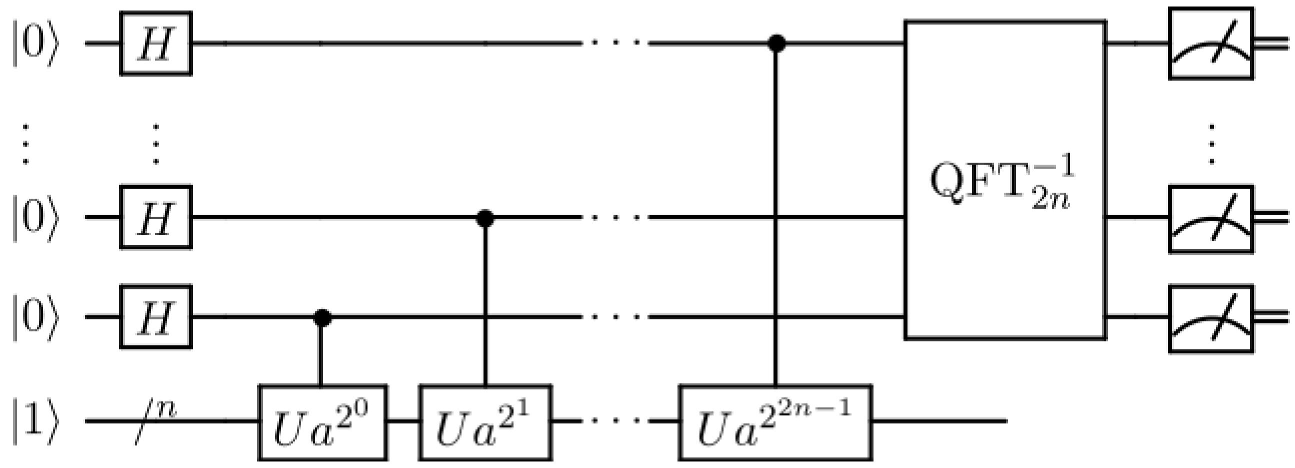
Shor’s Algorithm

Shor's algorithm is a quantum algorithm for finding the prime factors of an integer. It was developed in 1994 by the American mathematician Peter Shor. Shor proposed multiple similar algorithms solving the **factoring problem**, the **discrete logarithm problem**, and the **period (order) finding problem**. The discrete logarithm and the factoring problems are instances of the period finding problem.



Shor's Algorithm – What is period finding problem?

The **modulo** operation returns the remainder of a division. Given two positive numbers a and n, $a\ (mod\ n)$ is the remainder of the Euclidean division of a by n. For example:

$$7\ (mod\ 15) = 7, \qquad 49\ (mod\ 15) = 4$$

Right hand side of equation can be rewritten with modulo operation too, such as

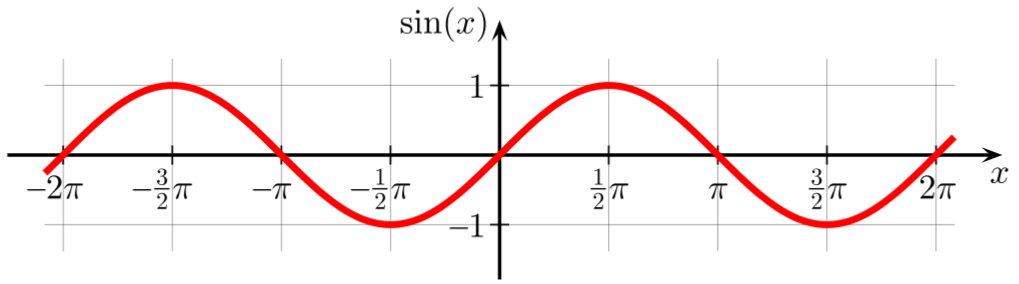
$$7 \equiv 7\ (mod\ 15), \qquad 49 \equiv 4\ (mod\ 15)$$

Period finding problem involves finding the period (repeating cycle) of a periodic function. A function $f(x)$ is said to be periodic if,

$$f(x) = f(y), \text{if and only if}\ y = x + r$$

where r is some nonzero constant, which is called the period of the function. With module operation, a periodic function is defined as,

$$f(x) = f(y), \text{if and only if}\ \ y \equiv x\ (mod\ r)$$



For example, sin function has a period of 2π:

$$\sin(x + 2\pi) = \sin(x)\,, \qquad x + 2\pi \equiv x\ (mod\ 2\pi)$$

Shor's Algorithm – What is discrete logarithm problem?

Discrete logarithm problem is a fundamental problem in the field of cryptography and computational number theory. It defines as finding the solution for function $f(x)$,

$$f(x) = a^x \equiv b \ (mod\ N)$$

where a, b, N are constant, and N is a prime number.

For example, let's take $a = 7, b = 4, N = 15$

$$f(x) = 7^x \equiv 4 \ (mod\ 15)$$

By testing few small positive integers,

$$\text{when } x = 2, \qquad 7^2 = 49 = 3 \times 15 + 4 \equiv 4 \ (mod\ 15)$$
$$\text{when } x = 6, \qquad 7^6 = 117649 = 7843 \times 15 + 4 \equiv 4 \ (mod\ 15)$$
$$\text{when } x = 10, \qquad 7^{10} = 282475249 = 18831683 \times 15 + 4 \equiv 4 \ (mod\ 15)$$
$$\vdots$$

The solution is $x = 2, 6, 10, \dots, 2 + 4n$ where n is a positive integer.

We notice that $\boldsymbol{f(x)}$ has a period of 4. $\boldsymbol{f(2) \equiv f(6) \equiv 4 \ (mod\ 15)}$

Shor’s Algorithm – What is factoring problem?

Factoring problem involves finding the prime factors of a composite (not prime) number, which is a number that can be divided by numbers other than 1 and itself.
A complete factoring algorithm is possible if we're able to efficiently factor an arbitrary integer N, find two integers p and q greater than 1, such that

$$N = p \cdot q$$

For example, $15 = 3 \cdot 5$. Then for complete factoring problem we can keep solve this problem until only primes factors remain. For example,

$$120 = 2 \cdot 60 = 2 \cdot 2 \cdot 30 = 2 \cdot 2 \cdot 2 \cdot 15 = 2 \cdot 2 \cdot 2 \cdot 3 \cdot 5$$

To solve the factoring problem, Shor’s algorithm consists of two parts

- A **classical reduction** of the factoring problem to the problem of order-finding.
- A **quantum algorithm** to solve the order-finding problem.

Shor’s Algorithm – Classical Reduction

Before explaining how to perform classical reduction, we need to introduction several terms:

- Greatest common divisor
- Chinese remainder theorem
- Euler's totient function
- Fermat’s little theorem
- Euler's theorem
- Order finding problem

Shor’s Algorithm – Classical Reduction

Greatest common divisor (GCD) of two integers, which are not all zero, is the largest positive integer that divides each of the integers. For two integers a, b, it is denoted as $\gcd(a, b)$.

$$\gcd(15, 21) = 3, \qquad \gcd(21, 98) = 7, \qquad \gcd(7,15) = 1$$

Chinese remainder theorem states that if one knows the remainders of the Euclidean division of an integer n by several integers, then one can determine *uniquely* the remainder of the division of n by the product of these integers, under the condition that the divisors are pairwise coprime (no two divisors share a common factor other than 1).

Let $p_1, p_2, p_3, \ldots, p_n$ be pairwise coprime ($\gcd(p_i, p_j) = 1$, where $i \neq j$). The system of n equations

$$\begin{cases} x \equiv a_1 \ (mod\ p_1) \\ x \equiv a_2 \ (mod\ p_2) \\ \qquad \vdots \\ x \equiv a_n \ (mod\ p_n) \end{cases}$$

has a unique solution for $x \ (mod\ N)$, where $N = p_1 \cdot p_2 \cdot p_3 \cdot \ldots \cdot p_n$. There could be more solution, such as x_1 and x_2, but they are congruent modulo N.

$$x_1 \equiv x_2 \equiv x \ (mod\ N)$$

Shor’s Algorithm – Classical Reduction

Chinese remainder theorem implies we can represent an element $x \ (mod\ pq)$ by one element of $a \ (mod\ p)$ and one element of $b \ (mod\ q)$, and vice versa.

Example system of 2 equations:

$$x \equiv 1 \ (mod\ 3), x \equiv 4 \ (mod\ 5)$$

We can easily find $x = 4, 19, 34, 49, \ldots$, which is $x \equiv 4 \ (mod\ 15)$.

$$x \equiv 4 \ (mod\ 15) \text{ can write as } x \equiv (1, 4) \ (mod\ 3, mod\ 5)$$

To compute $7^3 \ (mod\ 15)$:

$$7^3 \ (mod\ 15) \equiv 7 \times 7 \times 7 (mod\ 15)$$
$$\equiv (1 \times 1 \times 1, 2 \times 2 \times 2) \equiv (1, 8) \equiv (1, 3) \ (mod\ 3, mod\ 5)$$
$$\equiv 13 \ (mod\ 15)$$

To compute $7^4 \ (mod\ 15)$:

$$7^4 \ (mod\ 15) \equiv 7 \times 7 \times 7 \times 7 (mod\ 15)$$
$$\equiv (1 \times 1 \times 1 \times 1, 2 \times 2 \times 2 \times 2) \equiv (1, 16) \equiv (1, 1) \ (mod\ 3, mod\ 5)$$
$$\equiv 1 \ (mod\ 15)$$

Shor’s Algorithm – Classical Reduction

Euler's totient function counts the positive integers up to a given integer n that are relatively prime to n. It is written as $\varphi(n)$. In other words, it is the number of integers k in the range $1 \le k \le n$ for which the greatest common divisor $\gcd(n, k) = 1$.
For example, $n = 15$, there are 8 numbers coprime to 15: 1, 2, 4, 7, 8, 11, 13, 14

$$\varphi(15) = 8$$

To compute Euler’s totient function

$$\varphi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\cdots\left(1 - \frac{1}{p_r}\right)$$

where $n = p_1^{k_1} p_2^{k_2} \cdots p_r^{k_r}$

For example,

$$15 = 3 \times 5, \qquad \varphi(15) = 15\left(1 - \frac{1}{3}\right)\left(1 - \frac{1}{5}\right) = 8$$

Shor’s Algorithm – Classical Reduction

Fermat’s little theorem states that if p is a prime number, then for any integer a, such as $a \leq \mathrm{p}$. The number $a^p - a$ is an integer multiple of p.

$$a^p \equiv a \ (mod\ p) \Rightarrow a^p - a \equiv 0\ (mod\ p)$$

If a is coprime to p

$$a^{p-1} \equiv 1\ (mod\ p) \Rightarrow a^{p-1} - 1 \equiv 0\ (mod\ p)$$

For example,

$$a = 1, p = 2, \qquad 1^2 \equiv 1\ (mod\ 2)$$

$$a = 2, p = 7, \qquad 2^7 \equiv 128 \equiv 2 (mod\ 7)$$

Euler's theorem is a generalization of Fermat's little theorem: For any modulus n and any integer a coprime to n,

$$a^{\varphi(n)} \equiv 1\ (mod\ n)$$

For example,

$$\varphi(15) = 8 \Rightarrow a^8 \equiv 1\ (mod\ 15)$$

$$1^8 \equiv 1, 2^8 \equiv 1, 4^8 \equiv 1, 7^8 \equiv 1 \cdots (mod\ 15)$$

Shor’s Algorithm – Classical Reduction

Order-finding problem is similar to discrete logarithm problem. But instead of finding the solution for function $f(x)$, it finds the period of function $f(x)$,

$$f(x) = a^x \ (mod\ N)$$

Find period, or order r, which is the smallest (non-zero) positive integer such that:

$$a^r \ (mod\ N) \equiv 1 \quad \text{or} \quad a^r \equiv 1\ (mod\ N)$$

Using a similar example $a = 7$ and $N = 15$:

$$7^0 \equiv 1\ (mod\ 15)$$
$$7^1 \equiv 7\ (mod\ 15)$$
$$7^2 = 49 \equiv 4\ (mod\ 15)$$
$$7^3 = 343 \equiv 13\ (mod\ 15)$$
$$7^4 = 2401 \equiv 1\ (mod\ 15)$$
$$\vdots$$

We find the order $r = 4$

Shor's Algorithm – Classical Reduction

1. If N is not an even integer or a perfect power of prime, we start the algorithm.
2. Pick a random number $1 < a < N$
3. Compute $K = \gcd(a, N)$, the greatest common divisor of a and N.
4. Determine whether $K == 1$ or not.
    1. If $K \neq 1$, then K is a nontrivial factor of N. **We done $\boldsymbol{p = K,\ q = \frac{N}{K}}$.**
    2. If $K = 1$, then use the **quantum algorithm** to find the order r of a, where $a^r \equiv 1\ (mod\ N)$.
5. If r is odd, then go back to step 2.
6. Compute $g = \gcd(a^{\frac{r}{2}} + 1, N)$. Determine whether $g == 1$ or not
    1. If $g \neq 1$, then g is a nontrivial factor of N. **We done $\boldsymbol{p = g,\ q = \frac{N}{g}}$.**
    2. If $g = 1$, then go back to step 2.

Shor's Algorithm – Quantum Algorithm

The first important part Shor's algorithm is **quantum Fourier transform (QFT)**. QFT is a quantum implementation of the discreet Fourier transform. Using quantum computing, QFT is exponentially faster than the famous Fast Fourier Transform of classical computers.

The classical Fourier transform acts on a vector $(x_0, x_1, \dots, x_{N-1})$ and maps it to the vector $(y_0, y_1, \dots, y_{N-1})$ according to the formula:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{\frac{2\pi i}{N}\cdot(jk)} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega_N^{jk}$$

where $k = 0,1,2, \dots, N-1$ and $\omega_N = e^{\frac{2\pi i}{N}}$

Similarly, the **QFT** acts on a quantum state $|x\rangle = \sum_{j=0}^{N-1} x_j |n\rangle$ and maps it to a quantum state $|y\rangle = \sum_{k=0}^{N-1} y_k |k\rangle$ according to the same formula above. In case that $|j\rangle$ is a basis state, the QFT can also be expressed as the map:

$$|j\rangle \to \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk}$$

Shor's Algorithm – Quantum Algorithm

The **QFT** can be performed efficiently on a quantum computer with a decomposition into the product of simpler unitary matrices. The **QFT** can be viewed as a unitary matrix acting on quantum state vectors:

$$F_N = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{(N-1)2} & \omega^{(N-1)3} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}$$

where $\omega = e^{\frac{2\pi i}{N}}$. For example, in case of $N = 4$ and $\omega = e^{\frac{2\pi i}{4}} = i$:

$$F_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

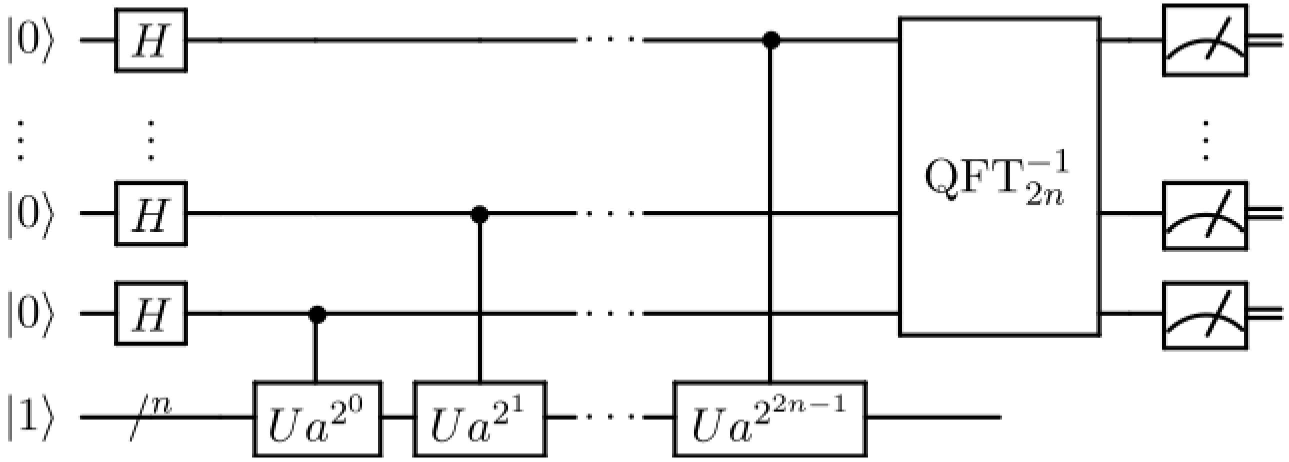
Shor's Algorithm – Quantum Algorithm

From step 4.2. "If $K = 1$, then use the **quantum algorithm** to find the order r of a, where $a^r \equiv 1\ (mod\ N)$."
The goal of the quantum order-finding subroutine of Shor's algorithm is finding the order r:

$$a^r \equiv 1\ (mod\ N)$$

where r is the smallest positive integer, not zero.

1. Use **quantum phase estimation** with unitary U representing the operation of multiplying by $a\ (mod\ N)$. Then we will measure a phase $\phi = \frac{s}{r}$.
2. Use **continued fractions algorithm** to extract the period r from the measurement outcomes obtained in the previous stage.



Shor's Algorithm – Quantum Algorithm

1. Use **quantum phase estimation** with unitary U representing the operation of multiplying by $a\ (mod\ N)$. Then we will measure a phase $\phi = \frac{s}{r}$.

We have a unitary operator:

$$U|x\rangle = |a \cdot x\ (mod\ N)\,\rangle$$

A superposition of the states in this cycle $|u_0\rangle$ would be an eigenstate of U:

$$|u_0\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}\left|a^k\ (mod\ N)\right\rangle \quad \text{and} \quad U|u_0\rangle = |u_0\rangle$$

Prove:

$$U|u_0\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}\left|a \cdot a^k\ (mod\ N)\right\rangle = \frac{1}{\sqrt{r}}|a \cdot a^{r-1}\ (mod\ N)\,\rangle + \frac{1}{\sqrt{r}}\sum_{k=0}^{r-2}\left|a \cdot a^k\ (mod\ N)\right\rangle$$

$$= \frac{1}{\sqrt{r}}|a^r\ (mod\ N)\,\rangle + \frac{1}{\sqrt{r}}\sum_{k=1}^{r-1}\left|a^k\ (mod\ N)\right\rangle$$

Since $a^r \equiv 1 \equiv a^0\ (mod\ N)$

$$= \frac{1}{\sqrt{r}}|a^0\ (mod\ N)\,\rangle + \frac{1}{\sqrt{r}}\sum_{k=1}^{r-1}\left|a^k\ (mod\ N)\right\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}\left|a^k\ (mod\ N)\right\rangle = |u_0\rangle$$

Shor's Algorithm – Quantum Algorithm

1. Use **quantum phase estimation** with unitary U representing the operation of multiplying by $a\ (mod\ N)$. Then we will measure a phase $\phi = \frac{s}{r}$.

Similar, we can define another eigenstate and apply the same unitary operator:

$$|u_1\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi i k}{r}} \left| a^k\ (mod\ N) \right\rangle \quad \text{and} \quad U|u_1\rangle = e^{\frac{2\pi i}{r}} |u_1\rangle$$

Prove:

$$U|u_1\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi i k}{r}} \left| a \cdot a^k\ (mod\ N) \right\rangle$$

$$= \frac{1}{\sqrt{r}} e^{-\frac{2\pi i (r-1)}{r}} \left| a \cdot a^{r-1}\ (mod\ N) \right\rangle + \frac{1}{\sqrt{r}} \sum_{k=0}^{r-2} e^{\frac{2\pi i}{r}} e^{-\frac{2\pi i (k-1)}{r}} \left| a \cdot a^k\ (mod\ N) \right\rangle$$

$$= \frac{1}{\sqrt{r}} e^{\frac{2\pi i}{r}} \left| a^r\ (mod\ N) \right\rangle + \frac{1}{\sqrt{r}} e^{\frac{2\pi i}{r}} \sum_{k=1}^{r-1} e^{-\frac{2\pi i k}{r}} \left| a^k\ (mod\ N) \right\rangle = \frac{1}{\sqrt{r}} \sum_{k=1}^{r-1} \left| a^k\ (mod\ N) \right\rangle$$

$$= \frac{1}{\sqrt{r}} e^{\frac{2\pi i}{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi i k}{r}} \left| a^k\ (mod\ N) \right\rangle = e^{\frac{2\pi i}{r}} |u_1\rangle$$

Shor's Algorithm – Quantum Algorithm

1. Use **quantum phase estimation** with unitary U representing the operation of multiplying by $a\ (mod\ N)$. Then we will measure a phase $\phi = \frac{s}{r}$.

Then we can define general eigenstate and apply the same unitary operator:

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi i k}{r} \cdot s} |a^k\ (mod\ N)\rangle \quad \text{and} \quad U\,|u_s\rangle = e^{\frac{2\pi i}{r} \cdot s}\,|u_s\rangle$$

where $0 \leq s \leq r-1$, and each eigenstate is unique.

If we sum up all these eigenstates, the different phases cancel out all computational basis states except $|1\rangle$

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle$$

Since the computational basis state $|1\rangle$ is a superposition of these eigenstates:

$$U|1\rangle = U\ \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = \frac{1}{\sqrt{r}} e^{\frac{2\pi i}{r} \cdot s} \sum_{s=0}^{r-1} |u_s\rangle = e^{\frac{2\pi i}{r} \cdot s} |1\rangle$$

we will measure a phase $\phi = \frac{s}{r}$

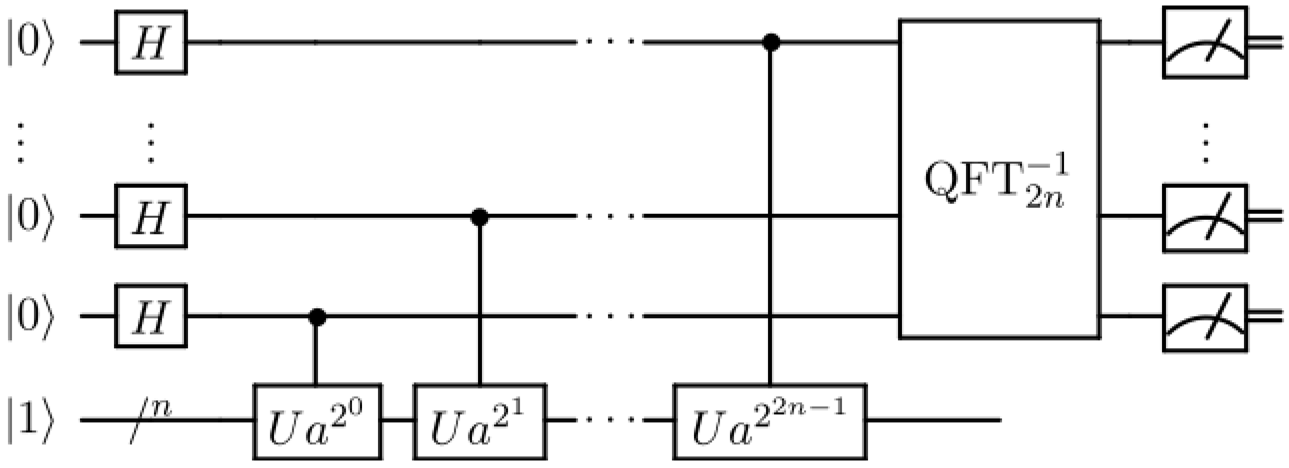
Shor’s Algorithm – Quantum Algorithm

The goal of the quantum order-finding subroutine of Shor's algorithm is finding the order r:

$$a^r \equiv 1 \ (mod\ N)$$

where r is the smallest positive integer, not zero.

1. Use **quantum phase estimation** with unitary U representing the operation of multiplying by $a\ (mod\ N)$. Then we will measure a phase $\phi = \frac{s}{r}$.
2. Use **continued fractions algorithm** to extract the period r from the measurement outcomes obtained in the previous stage.



Shor's Algorithm – Quantum Algorithm

1. Example using **quantum phase estimation.**

Give example $a = 7$ and $N = 15$:

$$7^0 \equiv 1 \ (mod\ 15)$$
$$7^1 \equiv 7 \ (mod\ 15)$$
$$7^2 \equiv 4 \ (mod\ 15)$$
$$7^3 \equiv 13 \ (mod\ 15)$$
$$7^4 \equiv 1 \ (mod\ 15)$$
$$\vdots$$

$$r = 4$$

With quantum phase estimation on the unitary operator U:

$$U|1\rangle \equiv |7\rangle$$
$$U^2|1\rangle \equiv |4\rangle$$
$$U^3|1\rangle \equiv |13\rangle$$
$$U^4|1\rangle \equiv |1\rangle$$
$$\vdots$$

With eigenstates:

$$|u_0\rangle = \frac{1}{2}(|1\rangle + |7\rangle + |4\rangle + |13\rangle)$$
$$|u_1\rangle = \frac{1}{2}(|1\rangle + e^{-\frac{2\pi i}{4}}|7\rangle + e^{-\frac{4\pi i}{4}}|4\rangle + e^{-\frac{6\pi i}{4}}|13\rangle)$$
$$|u_2\rangle = \frac{1}{2}(|1\rangle + e^{-2\frac{2\pi i}{4}}|7\rangle + e^{-2\frac{4\pi i}{4}}|4\rangle + e^{-2\frac{6\pi i}{4}}|13\rangle)$$
$$|u_3\rangle = \frac{1}{2}(|1\rangle + e^{-3\frac{2\pi i}{4}}|7\rangle + e^{-3\frac{4\pi i}{4}}|4\rangle + e^{-3\frac{6\pi i}{4}}|13\rangle)$$

Shor's Algorithm – Quantum Algorithm

1. Example using **quantum phase estimation.**

With eigenstates:

$$|u_0\rangle = \frac{1}{2}(|1\rangle + |7\rangle + |4\rangle + |13\rangle)$$
$$U|u_0\rangle = \frac{1}{2}(U|1\rangle + U|7\rangle + U|4\rangle + U|13\rangle)$$
$$= \frac{1}{2}(|7\rangle + |4\rangle + |13\rangle + |1\rangle) = |u_0\rangle$$

$$|u_1\rangle = \frac{1}{2}(|1\rangle + e^{-\frac{2\pi i}{4}}|7\rangle + e^{-\frac{4\pi i}{4}}|4\rangle + e^{-\frac{6\pi i}{4}}|13\rangle)$$
$$U\,|u_1\rangle = \frac{1}{2}(U|1\rangle + Ue^{-\frac{2\pi i}{4}}|7\rangle + Ue^{-\frac{4\pi i}{4}}|4\rangle + Ue^{-\frac{6\pi i}{4}}|13\rangle)$$
$$= \frac{1}{2}(|7\rangle + e^{-\frac{2\pi i}{4}}|4\rangle + e^{-\frac{4\pi i}{4}}|13\rangle + e^{-\frac{6\pi i}{4}}|1\rangle)$$
$$= e^{\frac{2\pi i}{4}}\frac{1}{2}(e^{-\frac{2\pi i}{4}}|7\rangle + e^{-\frac{4\pi i}{4}}|4\rangle + e^{-\frac{6\pi i}{4}}|13\rangle + e^{-\frac{8\pi i}{4}}|1\rangle) = e^{\frac{2\pi i}{4}}|u_1\rangle$$

Shor's Algorithm – Quantum Algorithm

1. Example using **quantum phase estimation.**

Sum up all these eigenstates:

$$U|1\rangle = U\ \frac{1}{\sqrt{4}}\sum_{s=0}^{3}|u_s\rangle = U\frac{1}{2}(|u_0\rangle + |u_1\rangle + |u_2\rangle + |u_3\rangle)$$

$$= \frac{1}{2}(|u_0\rangle + e^{\frac{2\pi i}{4}}|u_1\rangle + e^{\frac{4\pi i}{4}}|u_2\rangle + e^{\frac{6\pi i}{4}}|u_3\rangle)$$

$$= e^{\frac{2\pi i}{4}\cdot s}|1\rangle$$

where $0 \leq s \leq 3$

Shor’s Algorithm – Quantum Algorithm

2. Use **continued fractions algorithm** to extract the period r from the measurement outcomes obtained in the previous stage.

The **continued fractions algorithm** find integers b and c, where $\frac{b}{c}$ gives the best fraction approximation for the approximation measured from the quantum circuit. For $b, c < N$ and coprime b and c.

$$\frac{s}{r} = \frac{192}{256} = \frac{3}{4} = \frac{b}{c}$$

Shor's Algorithm – Quantum Algorithm

2. Use **continued fractions algorithm** to extract the period r from the measurement outcomes obtained in the previous stage.

Give example $a = 7, N = 15, \text{and } r = 4.$

$$U|1\rangle = \frac{1}{2}\left(|u_0\rangle + e^{\frac{2\pi i}{4}}|u_1\rangle + e^{\frac{4\pi i}{4}}|u_2\rangle + e^{\frac{6\pi i}{4}}|u_3\rangle\right)$$

Using 8 qubits for the quantum circuit, we could have the following measurements:

$$00000000 = 0\ (dec), \quad \frac{0}{256} = 0$$

$$01000000 = 64\ (dec), \quad \frac{64}{256} = \frac{1}{4}$$

$$10000000 = 128\ (dec), \quad \frac{128}{256} = \frac{1}{2}$$

$$11000000 = 192\ (dec), \quad \frac{192}{256} = \frac{3}{4}$$

Therefore, we find r could be 2 or 4. We can the larger probability one $r = 4$.