CS 506: An Introduction to Quantum Computing

University of Illinois Chicago

(Class Notes)

Student: Shri Krishna

Date: November 5, 2025

Factoring Problem

Given integer $N > 1$, find $1 < p < N$ such that p divides N in time polynomial in $\log_2 N$.

Used in RSA cryptography (Assumes factoring cannot be solved in polynomial time)

Let x be a **non-trivial square root of unity modulo** N. i.e., it satisfies the following conditions:

$$x^2 = 1 \pmod{N}$$
$$x \neq 1 \pmod{N}$$
$$x \neq -1 \pmod{N}$$

Then,

$$\text{GCD}(x-1, N) \text{ is a factor of } N$$

The "Order" of an integer x modulo N is the smallest integer $r > 0$ such that:

$$x^r = 1 \pmod{N}$$

Fact: $r < N/2$

Example: What is the order of $x/3$ modulo $N/5$?

$$3^1 = 3 \pmod{5}$$
$$3^2 = 4 \pmod{5}$$
$$3^3 = 2 \pmod{5}$$
$$3^4 = 1 \pmod{5} \quad \Rightarrow r = 4$$

Suppose, given integer a, we can find its order r (done by quantum algo)

$$a^r = 1 \pmod{N}$$

Suppose r is an **even number**. Then $x = a^{r/2}$ is a **square root of unity** (Need to check the non-trivialness)

$$x^2 = a^r = 1 \pmod{N}$$

So:

$$x^2 = a^r = 1 \pmod{N}$$
$$x^2 = a^r - 1 = 0 \pmod{N}$$

This expression factors into:

$$\left(a^{r/2} - 1\right)\left(a^{r/2} + 1\right) = 0 \pmod{N}$$

where $x = a^{r/2} - 1 \quad \& \quad x = a^{r/2} + 1$

We **need** the following:

$$\begin{aligned} x &\neq 1 \pmod N \quad \Rightarrow a^{r/2} \neq 1 \pmod N \quad \text{(Not Possible)} \\ x &\neq -1 \pmod N \quad \Rightarrow a^{r/2} \neq -1 \pmod N \quad \text{(Possible)} \end{aligned}$$

Number-Theoretic Result: If we choose a randomly and uniformly from $\{2, \ldots, N-1\}$, then:

$$P_r\left[\text{Order } r \text{ of } x \text{ is even and } a^{r/2} \neq -1 \pmod N\right] > \frac{1}{2}$$

and

$$p = \text{GCD}(a^{r/2} - 1, N)$$

Function for a:

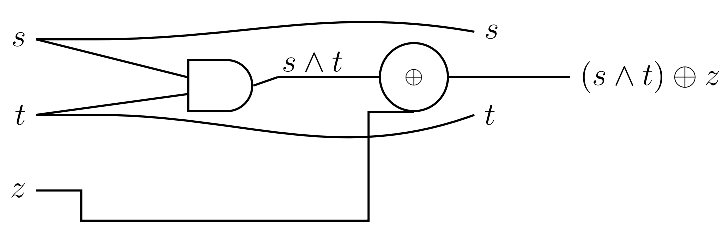
$$f(i) = a^i \pmod N$$

The periodicity of f is r:

$$\begin{aligned} f(1) &= a \pmod N \\ f(2) &= a^2 \pmod N \\ &\vdots \\ f(r) &= a^r = 1 \pmod N \\ f(r+1) &= a^{r+1} = a \pmod N \end{aligned}$$

Classical Implementation and Reversibility: Implement f classically using poly$(\log_2 N)$ gates such as AND, OR, NOT (with at most 2 inputs). Make every classical gate reversible.

Example: Reversible AND Gate



Note: $\mathcal{O}(1)$ additional gates to make it reversible.

Output Transformation: $(s, t, z) \to (s \wedge t) \oplus z$

The **Quantum circuit for** f has 1-qubit, 2-qubit, and 3-qubit gates. 3-qubit gates can be replaced by 1-qubit and 2-qubit gates.

The goal is to compute $a^i \pmod N$, where $1 \leq i < N$.

Computing the binary representation for i:

$$i = (b_{n-1} b_{n-2} \ldots b_j \ldots b_1 b_0)_2$$

Then:

$$\begin{aligned} a^i &= a^{\left(2^{n-1} b_{n-1} + 2^{n-2} b_{n-2} + \cdots + 2^j b_j + \cdots + 2^1 b_1 + 2^0 b_0\right)} \pmod N \\ &= a^{2^{n-1} b_{n-1}} \cdot a^{2^{n-2} b_{n-2}} \cdot \cdots \cdot a^{2^j b_j} \cdot \cdots \cdot a^{2^1 b_1} \cdot a^{2^0 b_0} \pmod N \end{aligned}$$

Then:

$$\left(\left(a^{2^{n-1}b_{n-1}} \bmod N\right) \cdot \left(a^{2^{n-2}b_{n-2}} \bmod N\right) \cdots \left(a^{2^{j}b_{j}} \bmod N\right) \ldots \left(a^{2^{1}b_{1}} \bmod N\right) \cdot \left(a^{2^{2}b_{2}} \bmod N\right)\right) \bmod N$$

Modular Exponentiation:

$$\begin{aligned} y &= 1 \\ w &= a \quad (\text{Note: } w = a^{2^0}) \end{aligned}$$

Steps:

Note: $b_0 = (x \bmod 2 = 1)$

1.

$$\begin{aligned} \text{if } b_0 = 1 \text{ then} \quad & y \leftarrow y \times w \pmod N \\ & w \leftarrow w^2 \pmod N \quad (\text{now } w = a^{2^1}) \\ & x \leftarrow \lfloor x/2 \rfloor \end{aligned}$$

2.

$$\begin{aligned} \text{if } b_1 = 1 \text{ then} \quad & y \leftarrow y \times w \pmod N \\ & w \leftarrow w^2 \pmod N \quad (\text{now } w = a^{2^2}) \end{aligned}$$

3.

$$\text{if } b_2 = 1 \text{ then} \quad y \leftarrow y \times w \pmod N$$

- The entire modular exponentiation algorithm goes through N such steps.
- The core operation, $y \leftarrow y \times w \pmod N$, is composed of adding, multiplying, or dividing two $\mathcal{O}(n)$ bit binary numbers.
- A single $\mathcal{O}(n)$-bit addition, multiplication, or division is an $\mathcal{O}(n^2)$ bit operation.
- A **bit operation** means performing addition, subtraction, division, or multiplication on 2 bits (i.e., using AND, OR, NOT gates).
- The total computation of $f(i) = a^i \pmod N$ uses $\mathcal{O}(n^3)$