# CS 593/MA 592 - Intro to Quantum Computing
# Spring 2024
# Tuesday, March 19 - Lecture 10.2

Today's scribe: Aaron [Note: note proofread by Eric]

## 1 Factoring and Friends

**Prime Factorization Problem.**
INPUT = A positive integer $N$ expressed in binary.
OUTPUT = A prime factorization of $N$, i.e. a list of positive integers $p_1, ..., p_m$ and $a_1, ...a_m$, again expressed in binary, such that $p_1, ..., p_m$ are distinct primes and $N = p_1^{a_1} ... p_m^{a_m}$.

**Note.** If $N$ were expressed in unary then its expression would have length $N$, and prime factorization could be done in poly-time by only trial division. As is, the size of $N$ is the number of bits, $L = \lfloor \log_2 N \rfloor + 1$, and trial division would not be polynomial in $L$.

Prime Factorization is essentially equivalent to the following more general problem.

**Factorization Problem.**
INPUT = $N$ as before.
OUTPUT = Either an integer $a$ in binary $(1 < a < N)$ such that $a$ divides $N$ (we say $a$ is a nontrivial factor of $N$), or the statement "$N$ is prime."

Prime Factorization can be performed by recursively calling the Factorization problem and dividing out by the results, which only has to be done polynomially-many times in $L$. Contrast this with primality testing.

**Primality Testing.**
INPUT = $N$ as above
OUTPUT = YES if $N$ is prime, NO otherwise.

Primality Testing has an efficient classical algorithm. Miller-Rabin placed it in BPP (1980) prior to Shor's algorithm, while Agrawal-Kayal-Saxena devised a deterministic algorithm (2000). However, the Factorization Problem has no known classically-efficient algorithm. The best known algorithm, the Number Field Sieve, runs sub-exponentially but super-polynomially with run-time $O(2^{\sqrt[3]{L}})$. Quantumly, though...

**Theorem (Shor).** Factoring can be done (with bounded error) in quantum-polynomial time.

The algorithm is broken into two main procedures. We first reduce the problem of factoring to the problem of *order finding*, which is the goal of this lecture. Shor then solved this problem in quantum polynomial time (which we'll elaborate on next lecture).

**Order Finding.**
INPUT = $N$ as before, as well as another integer $x$, $1 < x < N$, expressed in binary so that $\gcd(x, N) = 1$.
OUTPUT = The order of $x$ modulo $N$, i.e. the smallest positive integer $r$ such that $x^4 \equiv 1 \mod N$.

**Def.** $(\mathbb{Z}/n\mathbb{Z})^* = \{x \in \mathbb{Z}/N\mathbb{Z} : \gcd(x, N) = 1\}$
$= \{$Multiplicative units in the ring $\mathbb{Z}/N\mathbb{Z}\}$.

## 2 The Classical Part of Shor's Algorithm

Given $N$ we want to either identify $N$ as prime or find a (nontrivial) factor. We will assume we have an oracle which solves the order-finding problem.

**Step 1.** If $N$ is even, output 2. (Since $N$ is given in binary, this is very easy).

**Step 2.** If $N = a^b$ for some $a, b \geq 2$, output $a$. (This takes $\mathcal{O}(L^3)$ time, and is worked through in Ex 5.17 N-C).

**Step 3.** Uniformly-randomly pick an integer $x$ with $2 < x < N$. Use the Euclidean algorithm ($\mathcal{O}(L^3)$) to compute $\gcd(x,N)$. If this is $> 1$, output $\gcd(x,N)$.

**Step 4.** Now $\gcd(x,N) = 1$. Use the aforementioned oracle to compute the order of $x \mod N$. Call it $r$.

**Step 5.** If $r$ is odd, return to step 3 (pick a new $x$).
**Alternate Step 5'.** If $r$ is odd, output "$N$ is prime." This does introduce some error, but we'll prove later that this guess is "usually" correct.

**Step 6.** Now $r$ is even. If $x^{r/2} \not\equiv -1 \mod N$, then one of $\gcd(x^{r/2}+1,N), \gcd(x^{r/2}-1,N)$ is a nontrivial factor of $N$ (will justify this shortly). Check which one and output it.

**Step 7.** If $x^{r/2} \equiv -1 \mod N$, return to step 3.
**Alternate Step 7'.** If $x^{r/2} \equiv -1 \mod N$, output "$N$ is prime." Again, we'll show this guess is usually correct.

That this works hinges upon two technical results.

**Prop 1.** Suppose $N$ is a positive integer and $y$ is a nontrivial solution to $y^2 \equiv 1 \mod N$ with $1 < y < N-1$. Then $\gcd(y+1,N)$ or $\gcd(y-1,N)$ is a nontrivial factor of $N$.

This justifies Step 6, taking $y = x^{r/2}$.

**Proof.** If $y^2 \equiv 1 \mod N$, then by definition $N$ divides $y^2 - 1 = (y-1)(y+1)$. So either $\gcd(N, y-1)$ or $\gcd(N, y+1)$ is $> 1$. Moreover, neither equals $N$ since $y < N-1$.

**Prop 2.** Suppose $N$ has prime factorization $p_1^{a_1}...p_m^{a_m}$ with each $p_i \neq 2$ and $m \geq 2$. If $x$ is a uniformly-randomly chosen element of $(\mathbb{Z}/N\mathbb{Z})^*$, then

$$\mathbb{P}[r \text{ is even}, x^{r/2} \not\equiv -1 \mod N] \geq 1 - \frac{1}{2^{m-1}}.$$

This result justifies the alternative steps 5' and 7' (and since the algorithm is already probabilistic, there's little harm in relying on them). The proof relies on two basic facts of ring theory which we will give without proof.

**Fact 1 (Chinese Remainder Theorem).** If $N$ factors as $= q_1...q_m$ where the $q_i$'s are pairwise coprime, then

$$\mathbb{Z}/N\mathbb{Z} \cong (\mathbb{Z}/q_1\mathbb{Z}) \times ... \times (\mathbb{Z}/q_m\mathbb{Z}).$$

In particular:

$$(\mathbb{Z}/N\mathbb{Z})^* \cong (\mathbb{Z}/q_1\mathbb{Z})^* \times ... \times (\mathbb{Z}/q_m\mathbb{Z})^*.$$

**Fact 2.** If $p$ is prime, then $(\mathbb{Z}/p^a\mathbb{Z})^* \cong \mathbb{Z}/(p^{a-1}(p-1))\mathbb{Z}$.

Combining these facts, we can express the unit group modulo $N$ as a product of cyclic groups, as:

$$(\mathbb{Z}/N\mathbb{Z})^* \cong \mathbb{Z}/(p_1^{a_1-1}(p_1-1))\mathbb{Z} \times \mathbb{Z}/(p_m^{a_m-1}(p_m-1))\mathbb{Z}.$$

We ran out of time for finishing the proof of 2, we'll probably do that next lecture.