COL7160 : Quantum Computing

Lecture 13: Factoring to Order Finding Reduction

Instructor: Rajendra Kumar **Scribe:** Maalav Mehta

1 Introduction

Building on last lecture's discussion on Phase Estimation and IQFT, this lecture addresses Integer Factoring. We will first demonstrate how the factoring problem can be classically reduced to Order Finding in polynomial time. We will then complete the circuit by reducing Order Finding back to Phase Estimation, thereby establishing an efficient quantum algorithm for integer factoring.

2 Group Theory Basics

Definition 1. A *group* is a set G together with a binary operation $\cdot : G \times G \to G$ satisfying the following three properties:

1. **Associativity:** For all $g, h, f \in G$, $(g \cdot h) \cdot f = g \cdot (h \cdot f)$.
2. **Identity:** There exists an element $e \in G$ such that for all $g \in G$, $g \cdot e = e \cdot g = g$.
3. **Inverse:** For every $g \in G$, there exists $g^{-1} \in G$ such that $g \cdot g^{-1} = g^{-1} \cdot g = e$.

Example 2.

- $(\mathbb{Z}, +)$ is a group with identity 0 and inverse $-x$.
- $(\mathbb{Z}, \times)$ is *not* a group because most elements (like 2) do not have integer inverses.
- $(\mathbf{R}, \times)$ is *not* a group because 0 has no inverse.
- $(\mathbf{R} \setminus \{0\}, \times)$ is a group.
- $(\mathbb{Z}_n, +)$, the integers modulo n under addition, is a group.

What about $(\mathbb{Z}_n \setminus \{0\}, \times)$? This is a group if n is prime. If n is composite, it is not a group because elements that share a factor with n cannot have a multiplicative inverse. We define a special subset for which the multiplicative inverse always exists:

Definition 3. The multiplicative group of integers modulo n is defined as:

$$\mathbb{Z}_n^* = \{a \in \{1, 2, \ldots, n-1\} \mid \gcd(a, n) = 1\}$$

Lemma 4. *An element $a \in \mathbb{Z}_n$ has a multiplicative inverse modulo n if and only if $\gcd(a, n) = 1$. Thus, $\mathbb{Z}_n^*$ forms a group under multiplication modulo n.*

Proof. Suppose $\gcd(a, n) = 1$. By Bézout's identity, there exist integers x and y such that $ax + ny = 1$. Taking this equation modulo n, the term ny becomes 0, leaving us with $ax \equiv 1 \pmod{n}$. Thus, $x \pmod{n}$ acts as the multiplicative inverse of a modulo n.
Conversely, suppose a has a multiplicative inverse x modulo n. This implies $ax \equiv 1 \pmod{n}$, which means there must exist some integer y such that $ax = 1 + ny$, or equivalently, $ax - ny = 1$. Let d be any common divisor of a and n. Since d divides both a and n, it must also divide the linear combination $ax - ny$. Therefore, d must divide 1. The only positive integer that divides 1 is 1, meaning the greatest common divisor $\gcd(a, n)$ must be 1. □

2.1 Order of an Element

We now define the concept of the order of a group and the order of its elements, which will be central to our quantum algorithms.

Definition 5. The *order* of a group G, denoted $|G|$, is the total number of elements contained in the group. For the multiplicative group $\mathbb{Z}_N^*$, the order is given by Euler's totient function, denoted as $\varphi(N)$.

Definition 6. The *order of an element* $a \in G$ is the smallest positive integer r such that $a^r = e$, where e is the identity element of the group.

Lemma 7. *Let $a \in G$ be an element with order r. Then $a^x = a^y$ if and only if $x \equiv y \pmod{r}$.*

Proof. Without loss of generality, suppose $x \geq y$. The condition $a^x = a^y$ is equivalent to $a^{x-y} = e$. Let us apply the division algorithm to divide $x - y$ by r, giving a quotient q and a remainder s, such that $x - y = qr + s$ with $0 \leq s < r$. We can then write:

$$e = a^{x-y} = a^{qr+s} = (a^r)^q \cdot a^s = (e)^q \cdot a^s = a^s$$

Since r is defined as the *smallest* positive integer for which $a^r = e$, and we have $a^s = e$ with $s < r$, the only non-contradictory possibility is $s = 0$. Therefore, $x - y = qr$, meaning r divides $x - y$, which is exactly $x \equiv y \pmod{r}$. The converse follows immediately by reversing the steps. □

Theorem 8 (Euler's Theorem)**.** *For any $a \in \mathbb{Z}_N^*$, $a^{\varphi(N)} \equiv 1 \pmod{N}$.*

Proof. Let the elements of the multiplicative group $\mathbb{Z}_N^*$ be denoted by the set $X = \{x_1, x_2, \ldots, x_{\varphi(N)}\}$. Consider the set of elements $aX = \{ax_1, ax_2, \ldots, ax_{\varphi(N)}\}$, where $a \in \mathbb{Z}_N^*$.
Since $\mathbb{Z}_N^*$ is a group, it is closed under multiplication, so each $ax_i \in \mathbb{Z}_N^*$. Furthermore, all elements in aX are distinct modulo N; if $ax_i \equiv ax_j \pmod{N}$, multiplying both sides by a^{-1} yields $x_i \equiv x_j \pmod{N}$.
Thus, aX is simply a permutation of the elements of X. Equating the product of the elements in both sets, we have:

$$\prod_{i=1}^{\varphi(N)} (ax_i) \equiv \prod_{i=1}^{\varphi(N)} x_i \pmod{N}$$

Factoring out a from the left-hand side gives:

$$a^{\varphi(N)} \left(\prod_{i=1}^{\varphi(N)} x_i \right) \equiv \prod_{i=1}^{\varphi(N)} x_i \pmod{N}$$

Let $P = \prod_{i=1}^{\varphi(N)} x_i$. Since each x_i has a multiplicative inverse modulo N, the product P also has a multiplicative inverse $P^{-1} \pmod{N}$. Multiplying both sides by P^{-1}, we obtain:

$$a^{\varphi(N)} \equiv 1 \pmod{N}$$

□

Remark 9. The order r of any element $a \in \mathbb{Z}_N^*$ must divide the group order $\varphi(N)$. To see this easily, consider Theorem 8 (Euler's Theorem), which states $a^{\varphi(N)} \equiv 1 \pmod{N}$. Since $1 \equiv a^0 \pmod{N}$, we can write $a^{\varphi(N)} \equiv a^0 \pmod{N}$. By applying Lemma 7 with $x = \varphi(N)$ and $y = 0$, we immediately obtain $\varphi(N) \equiv 0 \pmod{r}$. This means r divides $\varphi(N)$.

3 Factoring and Order Finding

Factoring Problem: Given a composite integer $N \geq 2$, output its complete prime factorization $N = p_1^{e_1} p_2^{e_2} \ldots p_m^{e_m}$.

Remark 10. Although the formal problem asks for the complete factorization, finding just *one* non-trivial factor of N suffices to solve it efficiently.
If we have a subroutine that finds a non-trivial factor d, we can simply divide N by d and recursively apply the algorithm to both d and N/d. Since any integer N can have at most $\log_2 N$ prime factors, this recursion will terminate after at most $\log_2 N$ splits. Testing whether a number is prime at each recursive step can be done classically in polynomial time (e.g., using the AKS primality test). Therefore, if finding a single factor takes poly-$\log(N)$ time, finding the complete factorization takes poly-$\log(N)$ time as well.
For this reason, we will solely focus on extracting a single non-trivial factor.

Order Finding Problem: Given an integer N and an integer $a \in \mathbb{Z}_N^*$, output the order of a modulo N (the smallest positive integer r such that $a^r \equiv 1 \pmod N$).
We will now show that if we have a black-box that solves Order Finding, we can solve Factoring.

4 Reduction: Factoring to Order Finding

We want to factor an odd composite number N (if N is even, divide out all factors of 2 in preprocessing). We can assume N is not a prime power (i.e., $N \neq p^k$ for any prime p and integer $k \geq 2$). This is because prime powers can be easily checked and factored classically and deterministically in polynomial time.
To check if $N = p^k$, observe that since $p \geq 2$, we must have $k \leq \log_2 N$. Thus, there are only at most $\log_2 N$ possible values for the exponent k. For each possible $k \in \{2, 3, \ldots, \lfloor \log_2 N \rfloor\}$, we can perform a binary search over the integer range $[2, N]$ to find the integer k-th root of N. If we successfully find such an integer root x where $x^k = N$, then x is a non-trivial factor of N, so we are done. Because the search space is of size N, a binary search takes $O(\log N)$ steps. Testing all possible values of k therefore takes $O(\log^2 N)$ operations overall, which is polynomial in the input size.
Having ruled out prime powers, the randomized reduction strategy works as follows:

Algorithm 1 Reduction of Factoring to Order Finding

Input: A composite integer N (not even, not a prime power).
Pick an integer a uniformly at random from $\{2, \ldots, N-1\}$.
Compute $d = \gcd(a, N)$ using the Euclidean Algorithm.
if $d > 1$ **then**
    return d ▷ Lucky guess: we found a non-trivial factor directly!
end if
▷ Now $a \in \mathbb{Z}_N^*$. We use the Order Finding oracle.
Find the order r of a modulo N.
if r is odd **then**
    Fail and retry with a new a.
end if
if $a^{r/2} \equiv -1 \pmod N$ **then**
    Fail and retry with a new a.
end if
return $\gcd(a^{r/2} - 1, N)$ or $\gcd(a^{r/2} + 1, N)$.

4.1 Why does this work?

Suppose the algorithm does not fail. Since r is the order, $a^r \equiv 1 \pmod N$. Because r is even, we can write:

$$a^r - 1 \equiv 0 \pmod N \implies (a^{r/2} - 1)(a^{r/2} + 1) \equiv 0 \pmod N$$

This means N divides the product $(a^{r/2} - 1)(a^{r/2} + 1)$.
Could it be that N simply divides one of these terms entirely?

- N cannot divide $a^{r/2} - 1$. If it did, $a^{r/2} \equiv 1 \pmod N$, which contradicts r being the *smallest* positive integer (the order) satisfying this property.

- N cannot divide $a^{r/2}+1$. We explicitly check for this condition (Step 9). If $a^{r/2} \equiv -1 \pmod N$, the algorithm flags it as a failure and restarts.

Since N divides the product but divides neither term completely, some of the prime factors of N must divide $a^{r/2}-1$ and the rest must divide $a^{r/2}+1$. Thus, taking $\gcd(a^{r/2}-1, N)$ is guaranteed to yield a non-trivial factor of N.

4.2 Bounding the Failure Probability

The algorithm fails if r is odd or if $a^{r/2} \equiv -1 \pmod N$. We will prove that for a randomly chosen a, the probability of failure is at most $1/2$.

Theorem 11. *If N is an odd composite integer that is not a prime power, and a is chosen uniformly at random from $\mathbb{Z}_N^*$, then:*

$$\Pr\left[r \text{ is even and } a^{r/2} \not\equiv -1 \pmod N\right] \geq \frac{1}{2}$$

Proof. For simplicity, assume N is the product of two distinct odd primes, $N = p_1 p_2$. By the Chinese Remainder Theorem (CRT), there is an isomorphism:

$$\mathbb{Z}_N^* \cong \mathbb{Z}_{p_1}^* \times \mathbb{Z}_{p_2}^*$$

Choosing $a \in \mathbb{Z}_N^*$ uniformly at random is equivalent to choosing $a_1 \in \mathbb{Z}_{p_1}^*$ and $a_2 \in \mathbb{Z}_{p_2}^*$ uniformly at random and independently.
Let r_1 be the order of $a_1 \pmod{p_1}$ and r_2 be the order of $a_2 \pmod{p_2}$. The order r of a modulo N is the least common multiple of their separate orders:

$$r = \text{lcm}(r_1, r_2)$$

Let for any integer x, $v(x)$ denote the exponent of the highest power of 2 dividing x. The algorithm fails if r is odd, or if r is even and $a^{r/2} \equiv -1 \pmod N$. Let us analyze these two failure conditions in terms of $v(r_1)$ and $v(r_2)$:

1. r **is odd:** Since $r = \text{lcm}(r_1, r_2)$, r is odd if and only if both r_1 and r_2 are odd. This implies that $v(r_1) = 0$ and $v(r_2) = 0$. Consequently, $v(r_1) = v(r_2)$.

2. r **is even and** $a^{r/2} \equiv -1 \pmod N$**:** By the Chinese Remainder Theorem, this congruence holds if and only if $a_1^{r/2} \equiv -1 \pmod{p_1}$ and $a_2^{r/2} \equiv -1 \pmod{p_2}$. Recall that $a_1^{r_1} \equiv 1 \pmod{p_1}$. If $a_1^{r/2} \equiv -1 \pmod{p_1}$, it implies that $r/2$ is not a multiple of r_1, but r is a multiple of r_1. In terms of prime factorizations, this requires the highest power of 2 dividing r_1 to be exactly the highest power of 2 dividing r. Thus, $v(r_1) = v(r)$. Applying the exact same logic modulo p_2, we must also have $v(r_2) = v(r)$. Consequently, this failure case also requires $v(r_1) = v(r_2)$.

In both failure scenarios, a strictly necessary condition is that $v(r_1) = v(r_2)$. We will now bound the probability of this event.
Let 2^{v_1} and 2^{v_2} be the highest powers of 2 dividing p_1-1 and p_2-1, respectively. Since p_1 and p_2 are odd, $v_1, v_2 \geq 1$. Because $\mathbb{Z}_{p_1}^*$ is a cyclic group of order $p_1 - 1$, it has a generator g. Thus, our uniformly random choice $a_1 \in \mathbb{Z}_{p_1}^*$ can be written as $a_1 = g^k$, where k is chosen uniformly at random from $\{1, 2, \ldots, p_1 - 1\}$. The order of a_1 is given by $r_1 = \frac{p_1-1}{\gcd(k, p_1-1)}$.

- If k is odd (which happens with probability $1/2$), $\gcd(k, p_1 - 1)$ is odd, meaning no powers of 2 are canceled out from $p_1 - 1$. Thus, $v(r_1) = v_1$.
- If k is an odd multiple of 2 (which happens with probability $1/4$), exactly one power of 2 is canceled, giving $v(r_1) = v_1 - 1$.
- Generally, if k is an odd multiple of 2^j (probability $1/2^{j+1}$), then $v(r_1) = v_1 - j$.

This establishes that $\Pr[v(r_1) = v_1 - j] = 1/2^{j+1}$ for $0 \leq j < v_1$. A symmetric argument holds for a_2 and $v(r_2)$. Since a_1 and a_2 are chosen completely independently, $v(r_1)$ and $v(r_2)$ are independent random variables. We want to bound the probability that they are exactly equal.

$$\Pr[v(r_1) = v(r_2)] = \sum_i \Pr[v(r_1) = i] \cdot \Pr[v(r_2) = i]$$

We can upper bound $\Pr[v(r_1) = i] \leq 1/2$ for all i. Substituting this into the sum:

$$\Pr[v(r_1) = v(r_2)] \leq \sum_i \frac{1}{2} \cdot \Pr[v(r_2) = i] = \frac{1}{2} \sum_i \Pr[v(r_2) = i]$$

Since the sum of all probabilities for the mutually exclusive outcomes of $v(r_2)$ must exactly equal 1, we conclude that:

$$\Pr[v(r_1) = v(r_2)] \leq \frac{1}{2} \cdot 1 = \frac{1}{2}$$

Thus, the total probability of encountering a "bad case" is at most $1/2$. Therefore, the success probability of the reduction is at least $1/2$. □

Remark 12. Because the success probability is at least $1/2$ on any single run, repeating the algorithm k times guarantees we find a factor with probability at least $1 - (1/2)^k$, which converges to 1 exponentially fast.

References

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, UK, 10th anniversary edition edition, 2010.