CSCI 432: Shor's Algorithm

Name: _____

Collaborators: _____

April 2026

The Problem

Given a composite integer N, the **integer factoring problem** asks us to find a nontrivial factor of N. In other words, we want to find some integer d such that

$$1 < d < N \qquad \text{and} \qquad d \mid N.$$

Factoring is easy for small numbers but becomes computationally difficult for large integers. Many cryptographic systems rely on this difficulty.

Shor's Algorithm is a quantum algorithm that reduces factoring to a **period-finding** problem. The key idea is:

> Choose an integer a with $\gcd(a, N) = 1$, define $f(x) = a^x \bmod N$, and find the period r of f.

If r is even and $a^{r/2} \not\equiv -1 \pmod{N}$, then

$$\gcd(a^{r/2} - 1, N) \quad \text{and} \quad \gcd(a^{r/2} + 1, N)$$

give nontrivial factors of N.

Three ideas we will study today:

1. Classical reduction from factoring to order finding
2. Quantum period finding using superposition and the QFT
3. Why the algorithm succeeds and what its runtime means

Warm-Up: Modular Arithmetic and Periods

For this worksheet, recall:

- $a \bmod N$ means the remainder after dividing a by N
- $\gcd(a, N)$ is the greatest common divisor of a and N

- The **order** of $a \pmod N$ is the smallest positive integer r such that

$$a^r \equiv 1 \pmod N$$

provided $\gcd(a, N) = 1$

Consider

$$N = 15, \qquad a = 2.$$

1. Compute the first several values of $2^x \bmod 15$ and fill in the table.

x	$2^x \bmod 15$
0	
1	
2	
3	
4	
5	
6	
7	

2. What is the smallest positive r such that

$$2^r \equiv 1 \pmod{15}?$$

$r =$ _____

3. Why does this mean that the function

$$f(x) = 2^x \bmod 15$$

is periodic?

4. Using your value of r, compute:

$$2^{r/2} - 1 \qquad \text{and} \qquad 2^{r/2} + 1$$

and then evaluate

$$\gcd(2^{r/2} - 1, 15), \qquad \gcd(2^{r/2} + 1, 15).$$

5. What factors of 15 do you get?

Part I: Classical Reduction

Idea: Before using anything quantum, Shor's Algorithm first does some classical work. The algorithm either gets lucky immediately by computing a gcd, or it reduces factoring to the problem of finding the order of $a \pmod N$.

Algorithm 1: Classical Reduction for Factoring

Input: An odd composite integer N
Output: A nontrivial factor of N, or a request to find the order r of some $a \pmod N$

1. Pick a random integer a with $1 < a < N$
2. Compute $d = \gcd(a, N)$
3. If $d > 1$, return d as a factor
4. Otherwise, find the order r of $a \pmod N$
5. If r is odd, try again with a new a
6. If $a^{r/2} \equiv -1 \pmod N$, try again with a new a
7. Otherwise return
$$\gcd(a^{r/2} - 1, N) \quad \text{and} \quad \gcd(a^{r/2} + 1, N)$$

---

6. Why is Step 2 useful? What happens if $\gcd(a, N) > 1$ right away?

7. Explain why the equation
$$a^r \equiv 1 \pmod N$$
can be rewritten as
$$(a^{r/2} - 1)(a^{r/2} + 1) \equiv 0 \pmod N$$
when r is even.

8. Why does the case
$$a^{r/2} \equiv -1 \pmod N$$
fail to give us a useful factor?

9. Complete the following table for $N = 21$ and each choice of a.

a	$\gcd(a, 21)$	order r	r even?	$a^{r/2} \equiv -1 \pmod{21}$?	useful?
2					
4					
5					
7					

10. For one useful row from the table above, compute the two gcd values that produce factors of 21.

11. In your own words, what problem is left once the classical part finishes?

Part II: Quantum Period Finding

Idea: The hard part classically is finding the order r. The quantum subroutine finds information about the period of

$$f(x) = a^x \bmod N.$$

Instead of checking values one at a time, the quantum algorithm evaluates many inputs at once in superposition, then uses interference to reveal information about the period.

Algorithm 2: Quantum Period Finding (High-Level)

Input: Integers a, N with $\gcd(a, N) = 1$
Output: Information that can be used to recover the period r

1. Prepare two quantum registers
2. Put the first register into a superposition of many values of x
3. Compute $f(x) = a^x \bmod N$ into the second register
4. Measure the second register
5. The first register collapses to a superposition of values of x that are consistent with one output value
6. Apply the Quantum Fourier Transform (QFT) to the first register
7. Measure the first register to obtain a value related to multiples of $1/r$
8. Use classical post-processing (continued fractions) to recover r

---

12. What does it mean for the first register to be in a superposition of many x values?

13. Why do repeated values of $f(x) = a^x \bmod N$ matter for this algorithm?

14. Suppose the period is $r = 4$. List several integers that belong to the same congruence class modulo 4:

$$x_0,\ x_0 + 4,\ x_0 + 8,\ x_0 + 12, \ldots$$

If the second register is measured, why might the first register collapse to a sum of values like these?

15. Fill in the blanks:

> Measuring the second register does _____ the period. It instead leaves the first register in a state that still contains _____ information.

16. What is the role of the Quantum Fourier Transform in one sentence?

17. Why do peaks after the QFT tend to occur near multiples of

$$\frac{k}{r}?$$

18. Why is a final classical step still needed even after the quantum measurement?

Part III: Input, Output, and Correctness Conditions

For the following questions, think about the **classical reduction** part of Shor's Algorithm together with the period-finding subroutine.

36. Precondition (P)**:** State the conditions that must be true before the algorithm starts.

37. Postcondition (Q)**:** State what it means for the algorithm to succeed.

38. Failure / Retry Condition: State the situations where the algorithm does not finish with a factor and must choose a new a.

39. Intermediate Fact: After the period-finding step returns a candidate r, what must we check before using gcd to extract factors?

Part IV: Correctness Argument Skeleton

Use the following structure to organize a correctness argument.

40. Initialization: Explain why the algorithm begins in a valid state when we choose some a with $1 < a < N$.

41. Main Step: Assume the period-finding subroutine returns the true order r. Show why the algorithm returns correct factors whenever:

$$r \text{ is even} \qquad \text{and} \qquad a^{r/2} \not\equiv -1 \pmod{N}.$$

42. Termination: Explain what it means for the algorithm to terminate successfully.

43. Probabilistic Nature: Why is it acceptable for the algorithm to sometimes fail on one choice of a but succeed after repeating?

End of Worksheet