QuIC Seminar 14

Shor's Algorithm Part 3

Contents

Alright, we've done most of the heavy lifting up front and set the stage. The pieces are now ready to fall into place—this one should go quickly. Without further adieu, let's define the problem we need our quantum computer to solve.

14.1 Defining the quantum problem

Remember Simon's problem? Given a function $f : \{0,1\}^n \to \{0,1\}^n$ such that

$$f(x) = f(y) \iff x = y \oplus s, \tag{14.1}$$

determine s by querying f? In this case, the secret string s didn't do much practically for us, but we did find a fast quantum algorithm for figuring out what s was.

Can you guess how Shor tweaked this problem for factoring? Remember we care about the period of the modular exponential function.

Definition 14.1 (Shor's period-finding problem). Let $n = pq$ with p and q prime. Let $x \in \mathbb{Z}_n$ and consider the modular exponential function $f_x : \mathbb{Z}_+ \to \mathbb{Z}_n$ defined by

$$f_x(r) := x^r \mod n \tag{14.2}$$

which satisfies the condition

$$f(r) = f(t) \iff r = t + ks \tag{14.3}$$

where s is the period of f_x and k is an integer. **Shor's problem** is to determine the period s by querying f_x (as few times as possible).

Looks pretty similar to Simon's problem, right? Just wait until you see how similar the *algorithms* are.

14.2 Solving the quantum problem with a quantum algorithm

Remember the Deustch-Jozsa algorithm, or what we might call a *chant* or *dance* at this point? Hadamard! Query! Hadamard!

And do you remember how the Bernstein-Vazirani algorithm went? Hadamard! Query! Hadamard!

And Simon's algorithm? Hadamard! Query! Hadamard! (Post-process!)

Well, Shor learned how to dance from these folks, but that didn't stop him from adding a little razzle dazzle of his own to the quantum circuit dance floor. His new move? The Quantum Fourier Transform (QFT).

Shor's algorithm for period-finding

1. Hadamard!
2. Query!
3. Fourier transform!
4. Post-process!

As an interesting aside, we saw previously that the QFT on the all zero state is the same thing as Hadamards on each qubit in the all zero state. So, the first step in Shor's algorithm could be stated as a QFT, or more appropriately as a QFT†. Recall, we defined the QFT† to go from basis information to frequency information, and so the QFT goes from frequency information to basis information. What does this mean for Shor's period finding algorithm? The query takes place in frequency space. The procedure is thus: computational basis → frequency space → query → computational basis → measure. (And then make sense of the measurement results.)

Exercise 108: As a conceptual exercise, re-frame the DJ, BV, and Simon's algorithm in light of the QFT and the above discussion. How can we interpret/understand these algorithms by inserting a QFT for Hadamards?

Well this is a good conceptual understanding of what's going in the algorithm, and a good way to remember it, it's probably not the most enlightening on a first read. So, let's dig into the mathematical details to see what's going on.

As a guiding light to our understanding, let's look at the quantum circuit diagram for the first three steps of the period finding algorithm, shown in Fig. 14.1. (The similarity to Simon's algorithm should now be very clear!)

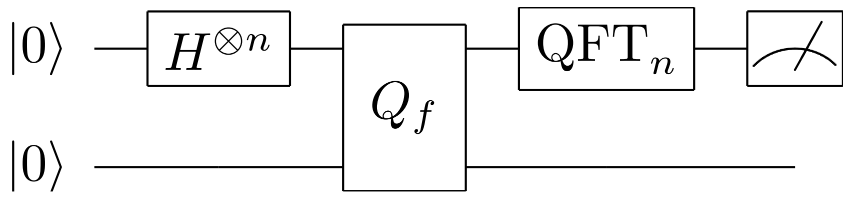


Figure 14.1: High-level quantum circuit diagram for Shor's period-finding algorithm.

The top register contains $n_Q := \lceil \log N^2 \rceil$ qubits and has dimension $Q := 2^{n_Q}$. Conceptually, we can remember that $Q \sim N^2$. The query Q_f is defined by

$$Q_f|r\rangle|k\rangle = |r\rangle|k \oplus f_x(r)\rangle \tag{14.4}$$

where $f_x(r) := x^r \mod N$ is the modular exponential function. We will come back to this later, but this query can in fact be efficiently implemented! Shor's algorithm really is end-to-end in the sense that all details are worked out down to circuit implementations of the oracle. (Unfortunately, it's one of the few which is end-to-end.)

For now, we have enough to examine the effect of the above circuit. It follows very closely the analysis of Simon's algorithm.

$$|0\rangle|0\rangle \xmapsto{H^{\otimes Q}} \frac{1}{\sqrt{Q}}\sum_{r=0}^{Q-1}|r\rangle|0\rangle \tag{14.5}$$

$$\xmapsto{Q_f} \frac{1}{\sqrt{Q}}\sum_{r=0}^{Q-1}|r\rangle|f_x(r)\rangle \tag{14.6}$$

$$\tag{14.7}$$

At this point, we use the principle of implicit measurement to "condition" on a particular value of the second register. More specifically, since we're done with the bottom register of qubits, we can imagine they are measured and the result is some value f. Since the two registers are entangled, this process affects the top register. Specifically, it selects all valid states k such that $f_x(k) = f$.

Recall that $f_x(r)$ is periodic. Let's call the period s. This means that if k is a valid state such that $f_x(k) = f$, then so is $k + s$, $k + 2s$, and so on. Thus if we had measured the bottom register as f, the top register would be in a state

$$\frac{1}{\sqrt{L}}\left[|k\rangle + |k+s\rangle + \cdots |k+(L-1)s\rangle\right] = \frac{1}{\sqrt{L}}\sum_{l=0}^{L-1}|k+ls\rangle \tag{14.8}$$

where the prefactor $L^{-1/2}$ ensures normalization.

Exercise 109: How do we know there is some finite number L of such terms? Could the number be infinite? (*Hint:* What is the domain of f_x?) What is the largest L could be?

So we have worked out the case where we *conditioned* on a particular measured value f. To go to the general case is simple enough, now: We just simply iterate over the possible values of f, which corresponds to adding a second summation over the parameter k in (14.8). Explicitly, the wavefunction after tracing out the bottom register is

$$\frac{1}{\sqrt{Q}}\frac{1}{\sqrt{L}}\sum_{k=0}^{Q-1}\sum_{l=0}^{L-1}|k+ls\rangle \tag{14.9}$$

where again s is the period of the modular exponential function $f_x(r) := x^r \mod N$. As we've traced out the bottom register, we no longer write it and are only looking at the top register.

At this point, we bust out the new dance move we've built up to blow everyone away: the Quantum Fourier transform (QFT). All we need to do is apply what we already know–namely, the effect the QFT has on basis states. We remember that conceptually the QFT maps from basis information to frequency information. More analytically, we apply Definition 13.3 (using linearity) to the state (14.9) to get

$$|\Psi\rangle := \frac{1}{Q\sqrt{L}}\sum_{j=0}^{Q-1}\sum_{k=0}^{Q-1}\sum_{l=0}^{L-1}\omega^{j(k+ls)}|j\rangle \tag{14.10}$$

where $\omega := e^{2\pi i/Q}$.

Exercise 110: Prove (14.11) by applying the definition of the QFT to (14.9) and using linearity.

Noting that $\omega^{j(k+ls)} = \omega^{jk}\omega^{jls}$, we factor the first term out of the summation over l to write

$$\boxed{|\Psi\rangle = \frac{1}{Q\sqrt{L}}\sum_{j=0}^{Q-1}\sum_{k=0}^{Q-1}\omega^{jk}\sum_{l=0}^{L-1}\omega^{jls}|j\rangle} \tag{14.11}$$

14.3 What can be measured? Destructive and constructive interference

The pertinent question now given the state $|\Psi\rangle$ is:

What possible values can we measure?

If we were to measure some state $|j\rangle$, the coefficients of this state better not sum to zero. (Otherwise, of course, there would be no probability of measuring this state.) What does this mean in terms of the coefficients

$$\sum_{l=0}^{L-1} \omega^{jls} = \sum_{l=0}^{L-1} e^{2\pi ijls/Q} \tag{14.12}$$

where j is an integer $j = 0, 1, 2, ..., Q-1$?

There's a lot of symbols here! We can simplify and still retain the essence of this expression. Explicitly, we can consider the sum

$$S(s,Q) := \sum_{q} e^{2\pi iqs/Q} \tag{14.13}$$

where $q = 0, 1, 2, ..., LQ$ for some integer L and $s < Q$ is an integer. (You should convince yourself this is the exact same expression as above, just with one fewer symbol.)

Example 1: An instructive sum to zero.
The goal of this example is straightforward: Evaluate $S(3,4)$.

The payoff will be a good understanding of the types of states we can measure in Shor's algorithm.

The key objects of interest for this example are the exponential terms

$$E(q) := e^{2\pi iqs/Q} = e^{3\pi iq/2} \tag{14.14}$$

where $q = 0, 1, 2, 3$. (We are taking $L = 1$ here for simplicity, though this is not necessary.) A simple calculation produces the following table.

q	E(q)
0	1
1	-i
2	-1
3	i

Thus, the summation $S(3,4)$ is zero! In terms of Shor's algorithm, this would mean that the state $|j\rangle$ with this coefficient could never be measured.

Figure 14.2 below gives a geometric picture for evaluating this sum.

In the above example, we saw a case where s/Q was not an integer and the sum $S(s,Q)$ was zero. It turns out that this is the general case!

Theorem 14.1 (Destructive interference in Shor's algorithm). Define

$$S(s,Q) := \sum_{q=0}^{Q-1} e^{2\pi iqs/Q} \tag{14.15}$$

where s and Q are integers. If s/Q is not an integer, then $S(s,Q) = 0$.

Before a potentially non-enlightening algebraic proof, it's actually easy to see why this is geometrically. Each exponential $\exp(2\pi iqs/Q)$ for $q = 0, 1, 2, ..., Q-1$ is a complex number with unit length. As such, it can be represented by a vector pointing from the origin to a point on the unit circle. If s/Q is not an integer, then every vector will be equally spaced (in terms of angles) in the complex plane, therefore resulting in the total sum being zero.

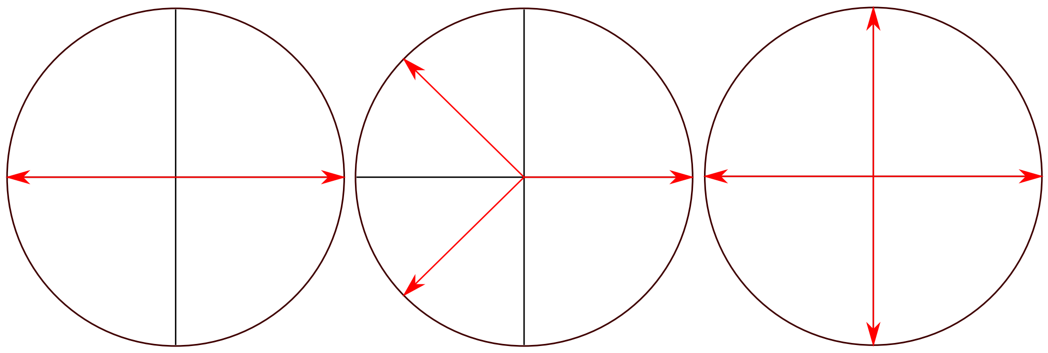


Figure 14.2: Representing the sum $S(s, Q)$ where $s = 1$ geometrically. The horizontal axis is the real axis, and the vertical axis is the imaginary axis. On the left, $Q = 2$. In the middle, $Q = 3$. And on the right, $Q = 4$. In each, it is clear that the sum over all vectors is identically zero.

This should be enough intuition, but it's nice to have an algebraic argument as well, which we include below. The idea is the same, though.

Proof. We can use the "exponential sum formula"

$$\sum_{q=0}^{Q-1} e^{iqx} = \frac{1 - e^{iQx}}{1 - e^{ix}}. \tag{14.16}$$

Using $x = 2\pi s/Q$, we thus have

$$S(s, Q) := \sum_{q=0}^{Q-1} e^{2\pi is/Q} = \frac{1 - e^{2\pi is}}{1 - e^{2\pi is/Q}} = 0. \tag{14.17}$$

In the last step, we used that s/Q is not an integer so that the denominator isn't zero. □

Since we singled out the case where s/Q is not an integer, it is natural to ask about the case when s/Q is an integer. Here, the sum is simpler to evaluate. Every exponential $\exp(2\pi iqs/Q)$ has the form $e^{2\pi i} = 1$ raised to some integer (namely, qs/Q). Thus, every term in the sum is one, and the total sum is Q. We have therefore proved the following.

Theorem 14.2 (Constructive interference in Shor's algorithm). Define

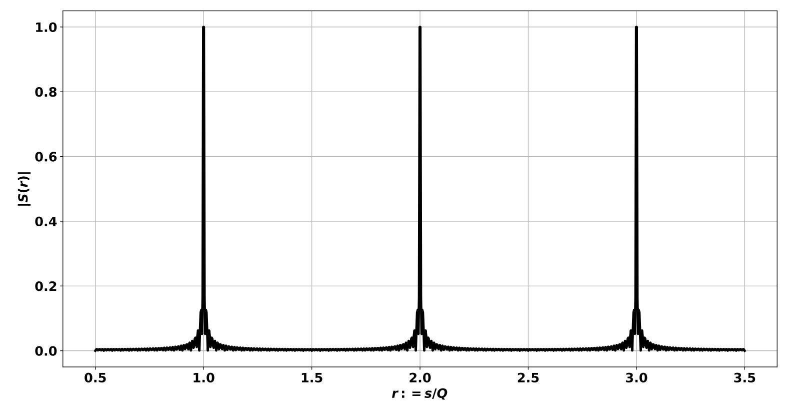
$$S(s, Q) := \sum_{q=0}^{Q-1} e^{2\pi iqs/Q} \tag{14.18}$$

where s and Q are integers. If s/Q is an integer, then $S(s, Q) = Q$.

We now translate back to the original notation from the state $|\Psi\rangle$ in (14.11) to see the important takeaway. Namely, we learned that we can only measure a state $|j\rangle$ with amplitude

$$\sum_{l=0}^{L-1} e^{2\pi ijls/Q} \tag{14.19}$$

if js/Q is an integer. In other words, if we measure $|j\rangle$, then we know that js/Q is an integer. Actually, this is not *completely* true, as the next section explains.



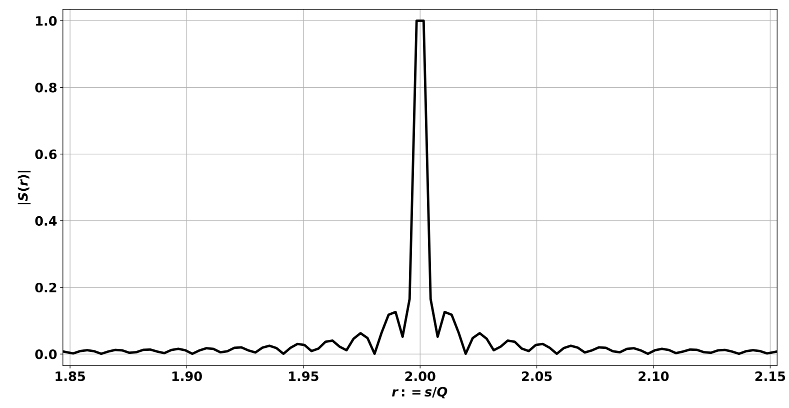


Figure 14.3: A plot of $|S(r)|$ defined in (14.20). Note that the vertical axis has be scaled to one for aesthetics. As can be seen, there is *perfect* constructive interference when s/Q is an integer—here 1, 2, or 3. There is destructive interference when s/Q is far from an integer, and partial constructive interference when s/Q is close to an integer. This is magnified in the bottom plot where we have zoomed in around $r = 2$.

14.3.1 An important subtlety: Partial constructive interference

Although we have the two mutually exclusive cases

1. s/Q is an integer $\implies$ constructive interference,
2. s/Q is not an integer $\implies$ destructive interference,

it is more important to treat the cases on a continuum. For example, what if $s/Q = 0.9999999$? This is not an integer, but it's certainly *close* to one[1]. What happens in this case? Is there perfect destructive interference? Or do we get "mostly" constructive interference?

As you might imagine (not from the section title!), since each exponent will be close to an integer, each exponential in $S(s, Q)$ will be close to one. Although there won't be *perfect* constructive interference, there will still be constructive interference. We can quantify how much based on how close s/Q is to an integer. Before this, however, an example to build intuition:

[1]Pun intended.

Example 2: Partial constructive interference in Shor's period finding algorithm.
Consider the sum $S(s, Q)$. Let's fix Q to be some power of 2, for example 256. Now, let's examine how the sum behaves for different ratios $s/Q =: r$. Explicitly, we consider the modulus of the sum

$$|S(r)| := \left| \sum_{q=1}^{256} e^{2\pi i r q} \right| \tag{14.20}$$

and plot this as a function of r from 0.5 to 3.5. The result is shown in Figure 14.3. As we see here, we get perfect constructive interference when s/Q is exactly an integer. However, as we have argued, this is an idealized scenario—the value of s/Q lies on a continuum. Thus, the second case (when s/Q is not an integer) needs to be clarified.

As we have argued above, we see (especially in the bottom plot of Fig. 14.3 that when s/Q is *close* to an integer, we get *partial* constructive interference. And when s/Q is *far* from an integer, we get *destructive* interference.

The above argument and example shows that we can refine our findings as follows:

If we measure a state $|j\rangle$, then we know that js/Q is *close* to an integer.

This is qualitative, but important for conceptually understanding the post-processing of Shor's algorithm. The following exercise makes the above statement more quantitative.

Exercise 111: Let $m \in \mathbb{Z}$ and $js/Q = m + \epsilon$ where $1 \gg \epsilon \in \mathbb{R}$.

(a) Show that the amplitude of the state $|j\rangle$ (ignoring global phase) in (14.11) is

$$\text{amp}(|j\rangle) = \frac{1 - e^{2\pi i \epsilon L}}{1 - e^{2\pi i \epsilon}}. \tag{14.21}$$

(b) Show that the probability p of measuring the state $|j\rangle$ is

$$p = 2\frac{[1 - \cos 2\pi\epsilon L][1 + \cos 2\pi\epsilon]}{1 - \cos 4\pi\epsilon} \tag{14.22}$$

(c) Using $\cos(x) \approx 1 - x^2/2$, show that we must have

$$\epsilon \leq \frac{1}{\pi}\sqrt{1 - p/L^2} \tag{14.23}$$

in order to have a probability of at least p for measuring the state $|j\rangle$.

14.4 Doing the post-processing

Alright, big picture:

After running the quantum circuit, we have an integer

$$j = mQ/s \pm \epsilon \tag{14.24}$$

where $m \in \mathbb{Z}$ and $1 \ll \epsilon \in \mathbb{R}$.

Our goal is to determine the period s of the modular exponential function, which allows us to factor N. We know what Q is (the dimension of the top register of the circuit), and we know what j is because we measured it. We do *not* know what the integer m is.

Since we know j and Q, we can determine an approximation not of s, but of m/s. In particular, rearranging (14.24), we have that

$$\left| \frac{j}{Q} - \frac{m}{s} \right| \leq \frac{\epsilon}{Q} \tag{14.25}$$

Thus the ratio j/Q, which we know, is *close* to m/s, which is (close to) what we what to know.

After running Shor's circuit, we can look at the resulting probability distribution—the diffraction pattern of the number N. This information will tell us several integers j where the probability peaks, and the height/width of the probability distribution at these locations tells us ϵ from (14.23). For example, if $p = 1$, then $\epsilon = 0$ and we measure $j/Q = m/s$ exactly.

Generally, we are not going to get a perfect distribution with probability spikes of one at particular integers. There will be some distribution about clusters of integers, as mentioned, and we will actually take away floating point numbers $a_1, ..., a_k$ as well as ϵ.

Now the question is: We have some floating point (decimal) number a that is ϵ away from an integer multiple of $1/s$. Specifically, we know an a and ϵ such that

$$|a - m/s| \leq \epsilon \tag{14.26}$$

where $m \in \mathbb{Z}$. How do we determine m/s from this?

14.4.1 The continued fractions algorithm

The answer to the above question is the *continued fractions algorithm*. Nearly everyone agrees, regardless of their usual (lack of) pedagogy, that it is best explained through an example.

Example 3: The continued fractions algorithm.
Suppose we run Shor's algorithm and learn that $a = 0.334$. One might guess that $m/s = 1/3$, but there is a more systematic way of doing this. The systematic way is the continued fractions expansion, performed as follows:

$$\begin{aligned} 0.334 &= \frac{334}{1000} \\ &= \frac{1}{\frac{1000}{334}} \\ &= \frac{1}{2 + \frac{332}{334}} \\ &= \frac{1}{2 + \frac{1}{\frac{334}{332}}} \\ &= \frac{1}{2 + \frac{1}{1 + \frac{2}{332}}} \end{aligned}$$

At this point, we get to spare the LaTeX compiler and those with bad eyes! Why? The fraction 2/332 (the last one in the expansion, if you can't see it) is small! Hence it is reasonable to truncate here, from which we find that

$$0.334 \approx \frac{1}{2 + \frac{1}{1+0}} = \frac{1}{3}, \tag{14.27}$$

which matches our expectation.

Any real number can be expanded in such a fashion. As you might expect, rational numbers have a finite expansion, and irrational numbers have an infinite expansion. There is a theorem (see Nielsen and Chuang) which states that this procedure can be done efficiently and accurately.

The last step of the algorithm is to perform the continued fractions algorithm "a few more" times. In particular, we perform continued fractions (CF) with j_1/Q to get m_1/s, perform CF with j_2/Q to get m_2/s. If we are unlucky, we may have to perform continued fractions with another j_3/Q to get m_3/Q, but you get the idea.

The idea is that both m_1/s and m_2/s share the denominator of s. If m_1 and m_2 are different (i.e., we did not get unlucky), then we can determine s.

14.5 Tidying up: Some additional details

Shor's algorithm is intricate, but we still did not cover all details. One important one that we mentioned was evaluating the modular exponential oracle to compute $f_x(r) := x^r \mod N$.

14.5.1 The modular exponential oracle

Here, we will not derive a complete quantum circuit which implements this oracle. Instead, we will argue that it is indeed efficiently implementable. In future seminars, we may derive it, or rather build it up.

The idea as to why f_x has an efficient circuit is that multiplication is classically efficient. Indeed, multiplication between two b-bit numbers takes time $O(b^2)$ by the standard "grade-school" multiplication algorithm. Any classical circuit that is efficient can be easily made into a quantum circuit which is also efficient, for example using Toffoli gates. (A perhaps deeper statement is that classical computing is a subset of quantum computing—namely, quantum computing restricted to the computational basis.)

"But," you interject, "we're not talking about multiplication... we're talking about *exponentiation*!" A good point, but this doesn't change the efficiency. Why? Through the trick of *repeated squaring*.

Example 4: Repeated squaring.

How many multiplications does it take to do x^8?

If your answer is seven (or eight—module off by one errors), would you be surprised to hear that the minimum number of multiplications is three?

Rather than doing $x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x$, we can do the much more efficient $x \cdot x$ to get x^2 using one multiplication. Then, once we have x^2, we can do $x^2 \cdot x^2$ to get x^4 for a total of two multiplications. Finally, we do $x^4 \cdot x^4$ to get x^8 for a grand total of three multiplications.

So, as the above example shows, to compute $f_x(r) = x^r \mod N$, we do *not* have to do $O(r)$ multiplications but rather $O(\lg r)$ multiplications. Recall the range for r is one to N, so at most we have to do $O(\lg N)$ multiplications, which is efficient.

As I mentioned, we won't show how to do the exponentiation (which involves multiplication, which involves addition, which involves sums and carries, which involves a bunch of Toffoli gates) *right now*, but we may in future seminars[2]. For the moment, we can rest easy knowing that it *can* be evaluated efficiently.

References

I owe a great deal to Scott Aaronson's lecture notes which my exposition of Shor's algorithm is based off.

[2]There's also very cool quantum arithmetic circuits based on QFTs which we will cover at some point.