Module 3: Quantum Computation

Contents

❖ Computation 1: Query-based algorithms

1.1 Query Complexity

Learning Outcomes
Upon following these notes and the corresponding lecture, students will be able to

- define what query complexity is and how we calculate it in the classical and quantum setting.
- describe the two ways to reversibly access a black box function.

To mathematically prove the advantage that quantum computers have over classical computers, we would love to be able to answer a question like the following:

"Does there exist a problem that can be efficiently solved with a quantum computer that **cannot** be solved efficiently with a classical computer?" In complexity theoretic language, it is asking if there is a problem that is in BQP, but not in P.

We don't really know how to prove this, because we don't know how to show that some problems **cannot** be solved efficiently. To work around this issue, we study a more limited model, and analyze what is called **query complexity**.

In query complexity, we assume that we have black box access to a Boolean function $f : \{0,1\}^n \to \{0,1\}$, and we want to know how many times we have to call this function to determine some property of the function. As you will see, some of these settings are quite artificial, but they provide good insight into the techniques that we know about quantum algorithm design, and are a proof of concept that there are settings where quantum computers perform better than classical. They are also often used to prove lower bounds on algorithms.

Question 1. What is the query complexity of a classical algorithm to call a function to determine the following properties?

- Is there any input x such that $f(x) = 1$?

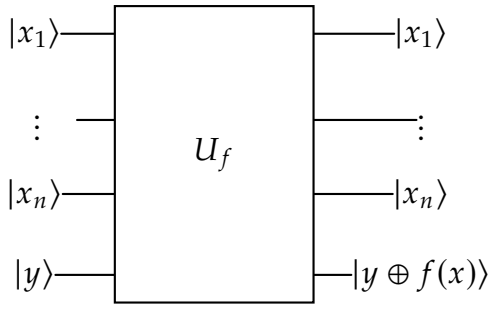
- Does $f(x) = 1$ for *most* of the inputs?

- Is f periodic?

To analyze the query complexity in a quantum setting, we need to embed this black box access to f into a quantum circuit. At the end of the previous module, we showed that if f can be computed by a classical circuit, then there exists a reversible circuit that computes f. Mathematically, we will express the general action of the reversible circuit as

$$(x, y, 0^k) \to (x, y \oplus f(x), 0^k). \tag{1}$$

Since the last register starts and ends with 0s for all inputs, we can just ignore it. Now we can embed our query to f as the reversible circuit with the following action:



Often when implementing quantum algorithms, we want the output to be stored in the phase instead of in an extra qubit:

$$|x\rangle \to (-1)^{f(x)} |x\rangle \tag{2}$$

This can be very useful for orchestrating interference patterns as we will see.

Question 2. Show that for a particular value of $|y\rangle$, we can use the above circuit to implement equation (2).

1.2 Deutsch's Algorithm

Learning Outcomes
Upon following these notes and the corresponding lecture, students will be able to

- describe the property of the function Deutsch's Algorithm is trying to determine.
- analyze the circuit of Deutsch's Algorithm.

Deutsch's algorithm was the first quantum algorithm proposed that demonstrated a speed up in query complexity over classical, though the speed up isn't too exciting. Nevertheless, the ideas used will give us the groundwork for thinking about more complex quantum algorithms.

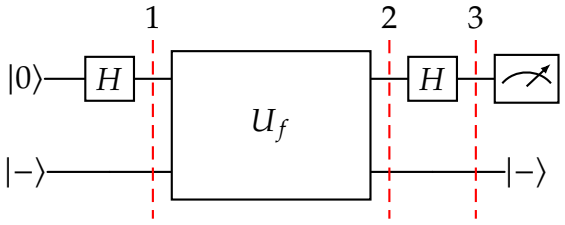
Consider a single bit Boolean function $f : \{0,1\} \to \{0,1\}$. We will denote its output bit for each input as

- $f(0) = b_0$
- $f(1) = b_1$

Given this function, we would like to determine the **parity** of $b_0 + b_1$, or more succinctly, we want to compute $b_0 \oplus b_1$.

Question 3. How many queries to f do we need classically to determine the parity of f?

I now claim that using a quantum computer, we can determine the parity using just one call to f. Here is the circuit for Deutsch's algorithm.



Question 4. What is the state of the system at 1?

Question 5. What is the state of the system at 2?

Question 6. What is the state of the system at 2 if $f(0) = f(1)$? What are the possible measurement outcomes for Deutsch's algorithm in this case?

Question 7. What is the state of the system at 3 if $f(0) \neq f(1)$? What are the possible measurement outcomes for Deutsch's algorithm in this case?

1.3 Deutsch-Josza Algorithm

Learning Outcomes
Upon following these notes and the corresponding lecture, students will be able to

- describe the property of the function the Deutsch-Josza Algorithm is trying to determine.
- apply the n-qubit Hadamard identity.
- analyze the circuit of Deutsch's Algorithm.

The Deutsch-Josza algorithm is a generalization of what we saw in the previous section. This time, we have access to a Boolean function with n-bit inputs:

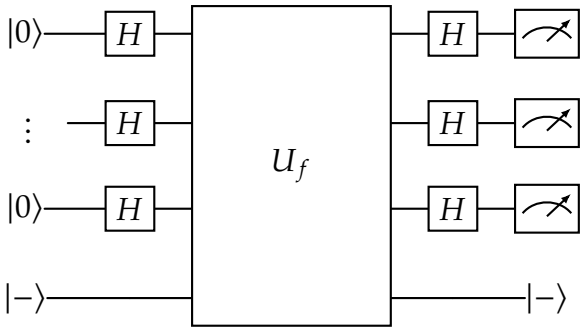
$$f : \{0,1\}^n \to \{0,1\} \tag{3}$$

and are **promised** that f satisfies one of the two following properties:

- f is a **constant function**, meaning that $f(x) = c$ for all inputs x
- f is a **balanced function**, meaning that $f(x) = 0$ for half of the inputs, and $f(x) = 1$ for the remaining half.

Question 8. How many queries do we need to make to this function to decide with 100% certainty which property is satisfied using a classical computer?

A quantum circuit can answer this question using just one query, with 0 probability of error. Here's the circuit:



To analyze this algorithm, we make use of an identity which will appear a lot throughout the rest of this class.

Proposition 1.1 (n-qubit Hadamard)**.** Let $x = x_1 x_2 \cdots x_n$ be the binary expansion of x. In other words, x_i is the i-th bit of x when x is written in binary. Then, we have the following identity:

$$H^{\otimes n} |x\rangle = H |x_1\rangle \otimes H |x_2\rangle \otimes \cdots \otimes H |x_n\rangle \tag{4}$$

$$= \frac{(|0\rangle + (-1)^{x_1} |1\rangle)}{\sqrt{2}} \otimes \cdots \otimes \frac{(|0\rangle + (-1)^{x_n} |1\rangle)}{\sqrt{2}} \tag{5}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle \tag{6}$$

where $x \cdot y$ is the bit wise dot product of x and y (i.e., $x \cdot y = x_1 y_1 + \cdots + x_n y_n$).

Question 9. Using the above identity, what is the state of the Deutsch-Jozsa algorithm before the call to U_f?

Question 10. What is the state of the Deutsch-Josza algorithm after the call to U_f?

Question 11. Using the above identity again, what is the state of the Deutsch-Josza algorithm after the second layer of H gates?

Question 12. What is the amplitude of $|0\cdots0\rangle$ if f is constant?

Question 13. What is the amplitude of $|0\cdots0\rangle$ if f is balanced?

Question 14. How can we use the measurement results to decide which property is held for the function f?

It turns out that if we allow for randomized classical algorithms where we can make errors, a simple sampling algorithm will very quickly be able to decide which property is held with high confidence. Because of this, the quantum speedup is not as glamorous as it seems.

1.4 Bernstein-Vazirani

Learning Outcomes
Upon following these notes and the corresponding lecture, students will be able to

- describe the property of the function the Bernstein-Vazirani Algorithm is trying to determine.
- analyze the circuit of the Bernstein-Vazirani's Algorithm.

The Bernstein-Vazirani algorithm is given black box access to a function $f : \{0,1\}^n \to \{0,1\}$ that we know is in the form

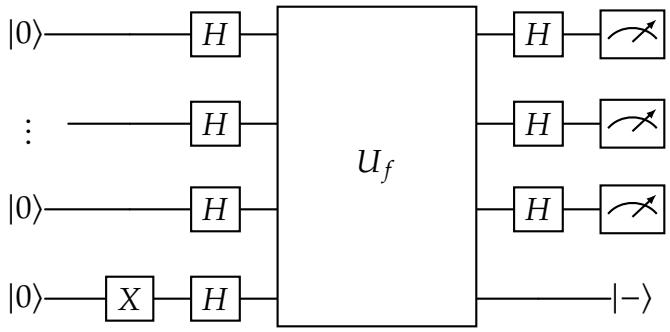
$$f_s(x) = x \cdot s \pmod 2 \tag{7}$$

for some mystery string $s \in \{0,1\}^n$. The goal of this algorithm is to figure out what s is.

Question 15. Let's consider an example where $n = 5$ and the secret string is $s = 10110$. What is $f(11101)$?

Question 16. What is a strategy we can use using a classical computer to decide what s is? What is the optimal query complexity classically?

Here is the circuit for the Bernstein-Vazirani algorithm:



Question 17. What is the state of the algorithm before the query to U_f?

Question 18. What is the state of the algorithm after the query to U_f?

Question 19. What is the state of the algorithm after the second layer of H gates?

Bernstein and Vazirani chose this problem since there is a way to have all the amplitudes for $y \neq s$ interfere destructively to become 0, while the amplitudes for s all "point in the same direction" and interfere constructively to become 1. We have found a way to achieve a linear query complexity speed up using a quantum algorithm, but can we do even better? Are there setting where we can achieve exponential speed up?

1.5 Simon's Algorithm

Learning Outcomes
Upon following these notes and the corresponding lecture, students will be able to

- describe the property of the function the Simon's Algorithm is trying to determine.
- prove the generalized Birthday paradox.
- analyze how hard it is to decide the property classically.
- analyze the circuit of the Bernstein-Vazirani's Algorithm.

In this problem, we will consider a function with an n bit input and an n bit output. The function $f : \{0,1\}^n \to \{0,1\}^n$ will encode a secret string s in the following way.

$$f(x) = f(y) \iff x \oplus y = s \iff x \oplus s = y \tag{8}$$

As we have been doing, given blackbox access to this function, the goal is to find s.

Question 20. Let f be a function that takes an n bit input and satisfies the property above. What is the size of the domain of this function? What is the size of the range of this function?

To determine what s is, we need to find a pair x and y such that $f(x) = f(y)$, and then take the sum module 2 of these strings to recover s.

Question 21. How many queries will we need classically in the worst case to determine s?

What if we use a randomized classical algorithm? In this case, we can show that we will require approximately $\sqrt{2^n} = 2^{n/2}$ queries. Let's try to prove this together. To prove this, we will use a general version of the Birthday Paradox.

Suppose we have a set of items, each with a uniformly random tag from $\{1, 2, \ldots, T\}$. How many samples do we need to collect before we have at least two items with the same tag with probability greater than $1/2$?

Question 22. What is the probability that a random pair of items have matching tags?

Question 23. Suppose we have chosen m items so far. How many different ways can we pair two items from this set (the tags do not have to match)?

Question 24. Determine how many items m we have to choose until the probability that there is a collision is over $1/2$.

Now suppose that this randomized algorithm queries the function using t bit strings, $x_1, x_2, \ldots, x_t$.

- If we find a pair such that $f(x_i) = f(x_j)$, then we are done.
- If none of these $x_i's$ are matches, then we know that $s \neq x_i \oplus x_j$ for all i, j pairs. In other words, we have ruled out $\binom{t}{2} \sim \frac{t^2}{2}$ possibilities, and all other choices are equally likely. In the worst case, we need to find t such that the number of items we rule out equals all possible inputs.

  We can conclude then, that classically we need at least $\Omega(2^{n/2})$ queries.

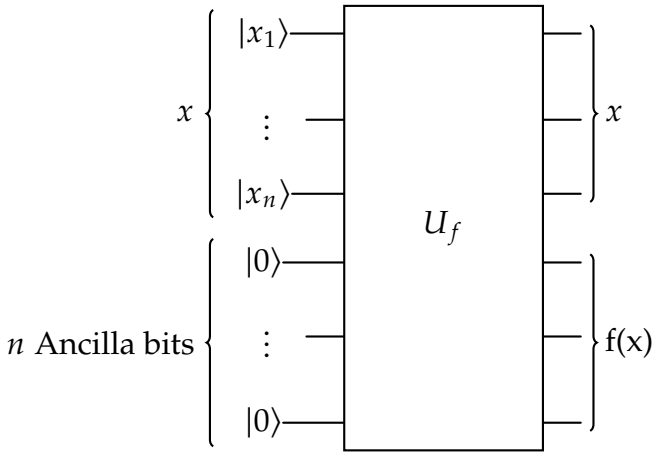
How can we solve this problem using a quantum computer? The unitary encoding the function would act as follows:

$$U_f(\vec{x}, \vec{0}) = (\vec{x}, \vec{0} \oplus f(x)) \tag{9}$$

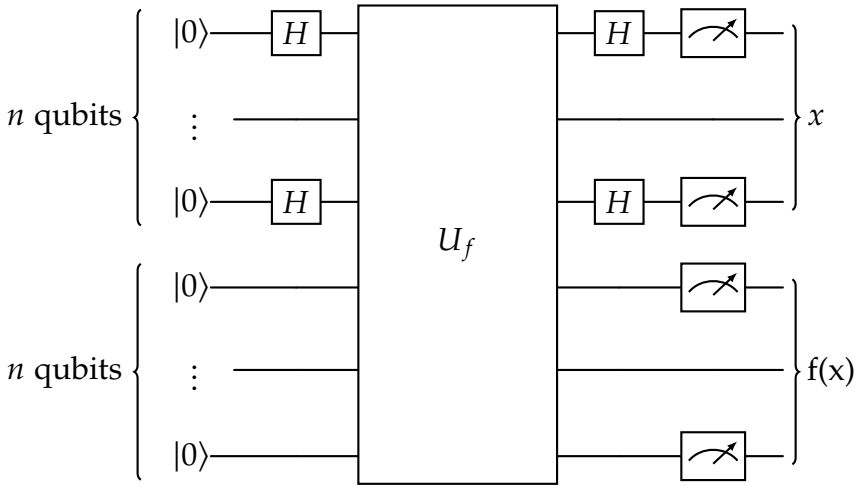
or in ket notation

$$U_f \left|x\right\rangle \left|0\right\rangle = \left|x\right\rangle \left|f(x)\right\rangle . \tag{10}$$

As a circuit, they would look like the following.



Now the circuit for solving Simons problem is just a small addition to the above circuit and is drawn below.



The neat thing about this algorithm is that we don't actually care about what our measurement result is, but just the interference pattern that is created. Let's go through the circuit to see what we mean by this.

Question 25. What is the state of the algorithm before the U_f gate?

Question 26. What is the state of the algorithm after the U_f gate?

It turns out that we can measure the last n qubits before applying the H gates on the first n qubits. The output distribution will be the same in either case!

Question 27. Suppose that upon measuring the second register at this stage, we get the result $|w\rangle$. What is the state of the first register?

It would be great if we could have multiple copies of the above state, because then we can directly measure the first register to recover all the relevant states. The problem is that if we rerun this experiment, it is extremely unlikely (how unlikely?) that we measure $|w\rangle$ again!

Instead, what this circuit is doing is measuring in the H basis by applying the H gates.

Question 28. What is the state of the superposition after the final Hadamard gates?

Question 29. Suppose we measured the first register and observe some random $|z\rangle$. What can we say about the coefficient of such a $|z\rangle$?

This can be analyzed using modular arithmetic:

$$x \cdot z \mod 2 = y \cdot z \mod 2 \tag{11}$$

$$(x - y) \cdot z \mod 2 = 0. \tag{12}$$

When working in binary, $x - y$ is equivalent to $x \oplus y$. Therefore what we get is that for the string z we recovered,

$$(x \oplus y) \cdot z = s \cdot z = 0. \tag{13}$$

So in 1 run of Simon's algorithm we found a random z that is orthogonal to s!

The measurement yields a random z such that $s \cdot z \equiv 0(\mod 2)$. We can repeat this $O(n)$ times to get a set of linearly independent strings who are all orthogonal to s. Once we have this, we can use Gaussian elimination (mod 2) to find s in $O(n^3)$ time.

$$\begin{bmatrix} - & - & - \\ - & - & - \\ & \vdots & \\ - & - & - \end{bmatrix} \begin{matrix} \\ z_1 \\ z_2 \\ \\ z_m \end{matrix} \begin{bmatrix} - & - & - \\ - & - & - \\ & & \\ - & - & - \end{bmatrix} \cdot \begin{bmatrix} | \\ s \\ | \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{14}$$

This gives us a polynomial time quantum algorithm to find s, whereas classically the best we could do was still exponential.

1.6 Query Based Algorithm Wrap Up

We've looked at several algorithms in this strange query model, which achieves speedups in a non-standard way. You may be suspicious that we are sweeping too many details under the rug, and for that you would be correct. To actually *implement* Simon's algorithm, you need an actual circuit to compute f, and when given to the actual circuit (as opposed to a black-box oracle), classical algorithms can exploit the details of the circuit to significantly reduce the number of queries.

Unfortunately, because of this reason these algorithms we have seen so far are not actually very practical for finding ways to speed up our computations. However, they provided valuable practice using some tools that will be useful for analyzing other quantum algorithms. Furthermore, I hope it gave you a peek into the workflow of a computer science researcher, and some ways that we try to separate the power of classical and quantum computing. It is not perfect, but it provides some concrete examples and intuition behind why quantum computers may excel at certain tasks over classical computers.

❖ Computation 2: Quantum Fourier Transform and Shor's Algorithm

"How dare we dream

Of solving problems that else would take more time

Than has passed since the cosmos's Big Bang!"

- Peter Shor

2.1 Quantum Fourier Transform

Learning Outcomes

Upon following these notes and the corresponding lecture, students will be able to

- describe and analyze the effect of the quantum Fourier transform on a given input state.

We begin by reviewing how to transition between the bit string and integer representation. To compute the value of the bit string 11001, we multiply it as follows:

$$1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 16 + 8 + 1 = 25. \tag{15}$$

More generally, if we have a string $x = x_1 \cdots x_n$ where x_i is the i-th bit of x, we can write this as an integer using the sum

$$x_1 \cdot 2^{n-1} + x_2 \cdot 2^{n-2} + \cdots + x_n \cdot 2^0 = \sum_{k=1}^{n} x_k \cdot 2^{n-k}. \tag{16}$$

Throughout this section, we will frequently take complex numbers to the power of integers, and it will be useful to have a way to toggle between the integer representation of the power and the bit string representation.

$$\omega^x = \omega^{x_1 \cdot 2^{n-1} + x_2 \cdot 2^{n-2} + \cdots + x_n 2^0} \tag{17}$$

$$= \omega^{x_1 \cdot 2^{n-1}} \cdot \omega^{x_2 \cdot 2^{n-2}} \cdot \cdots \cdot \omega^{x_n \cdot 2^0} \tag{18}$$

$$= \prod_{k=1}^{n} \omega^{x_k \cdot 2^{n-k}} \tag{19}$$

For the rest of the course, we will use the shorthand $N = 2^n$. One important reason we will be interested in complex numbers is because they are a great tool for analyzing periodic functions. In particular, the N-th roots of unity give us a way to keep track of the "period" in the way a clock would.

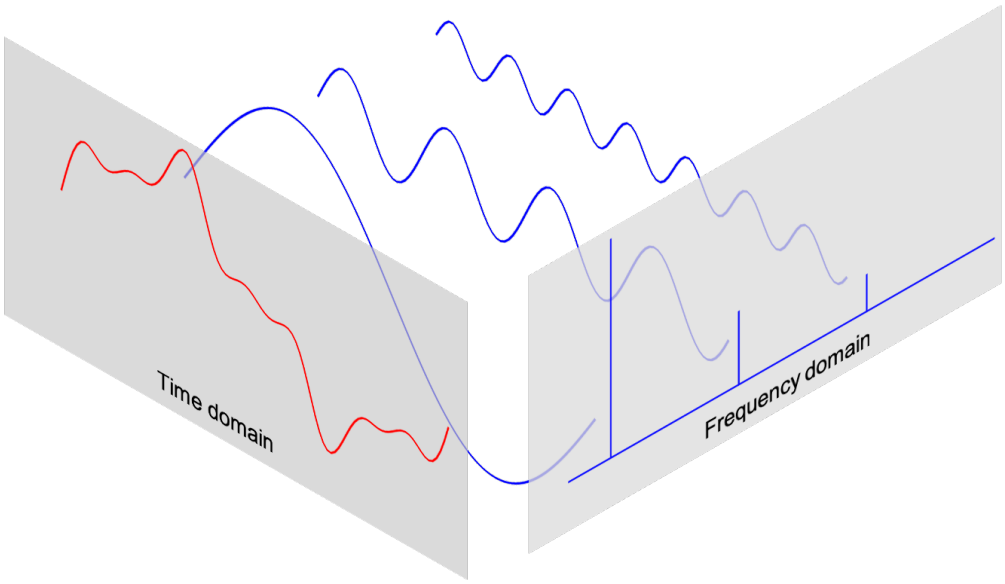
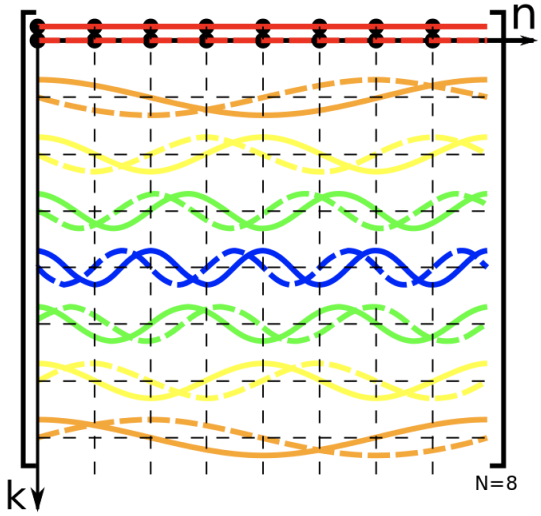


Figure 1: Taken from Python Numerical Methods

You may have learned about the Discrete Fourier Transform (DFT). The DFT is often used for signal processing, where a signal could be a sound wave, radio signal, or daily temperature readings. Usually we describe these signals in the **time domain**. Instead of doing this, we can take a slice of time to describe a signal in the **frequency domain**. By doing this, we have a discrete set of items to build our wave out of, and we can safely discard frequencies that are too high or low for the human ear.



Early in the investigation of quantum algorithms, researchers figured out that a quantum circuit can compute the Fourier transform very efficiently when the input vector is encoded in the amplitudes of a quantum state. Note that this is not necessarily a faster way to compute the classical Fourier transform, since the output is only accessible via quantum measurement.

Definition 2.1 (Quantum Fourier Transform)**.** Let $N = 2^n$. For $x \in \{0, 1, \ldots, N-1\}$:

- Standard basis state: A length N column vector with a 1 in the k-th location

$$|k\rangle = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{20}$$

- Fourier basis state ($\omega = e^{2\pi i/N}$ is the first N-th root of unity):

$$|\tilde{k}\rangle = \frac{1}{\sqrt{N}} \begin{bmatrix} \omega^0 \\ \omega^k \\ \omega^{2k} \\ \vdots \\ \omega^{(N-1)k} \end{bmatrix}. \tag{21}$$

The n-qubit **Quantum Fourier Transform** or QFT_N is the transformation that performs

$$QFT_N |x\rangle = |\tilde{x}\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{x \cdot y} |y\rangle \tag{22}$$

We can model the action of QFT_N by the matrix

$$QFT_N := \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}. \tag{23}$$

Let's get familiar with the Fourier basis by working through an explicit example.

Question 30. Write down the 1-qubit QFT matrix. Use the matrix to find all of the Fourier basis states.

Question 31. Write down the 2-qubit QFT matrix. Use the matrix to find the first Fourier basis state, $|\tilde{1}\rangle$.

Let's see if we can describe Fourier basis states for the general case. To do this, we will use the integer to bit string mapping we discussed earlier. Recall that the mapping we are interested in is

$$|x\rangle \leftrightarrow \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{x \cdot y} |y\rangle . \tag{24}$$

Let $y = y_1 \cdots y_n$ be the bitstring representation of y.

Question 32. Write down the value of y using the bits $y_1, \ldots, y_n$.

Let's use this to rewrite the power of the N-th root of unity.

$$\omega^{xy} = \omega^{x[y_1 \cdot 2^{n-1} + y_2 \cdot 2^{n-2} + \cdots + y_n \cdot 2^0]} \tag{25}$$

$$= \prod_{j=1}^{N} \omega^{x \cdot y_j \cdot 2^{n-j}}. \tag{26}$$

We can then rewrite the Fourier basis state as

$$|\tilde{x}\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{x \cdot y} |y\rangle \tag{27}$$

$$= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \prod_{j=1}^{N} \omega^{x \cdot y_j \cdot 2^{n-j}} |y\rangle \tag{28}$$

$$= \frac{1}{\sqrt{N}} \left(|0\rangle + \omega^{x \cdot 2^{n-1}} |1\rangle\right) \otimes \left(|0\rangle + \omega^{x \cdot 2^{n-2}} |1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + \omega^{x \cdot 2^{0}} |1\rangle\right) \tag{29}$$

Question 33. Consider a 3-qubit Fourier transform. What is $|\tilde{7}\rangle$? What is $|\tilde{7}\rangle$ written in the form of equation (29)?

Question 34. What is the amplitude of $|101\rangle$ in $|\tilde{7}\rangle$?

Equation (29) gives us a way to view the mapping as a tensor product of n qubits:

$$|x\rangle = |x_1\rangle \otimes \cdots \otimes |x_n\rangle \tag{30}$$

$$\leftrightarrow \frac{1}{\sqrt{N}} \left(|0\rangle + \omega^{x \cdot 2^{n-1}} |1\rangle\right) \otimes \left(|0\rangle + \omega^{x \cdot 2^{n-2}} |1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + \omega^{x \cdot 2^{0}} |1\rangle\right). \tag{31}$$

To summarize the algorithm, we will perform the following mapping between qubits and then reverse the order of the qubits using swap gates at the end:

$$|x_k\rangle \rightarrow \frac{1}{\sqrt{2}} \left(|0\rangle + \omega^{x \cdot 2^{k-1}} |1\rangle\right). \tag{32}$$

Note that the state is like a $|+\rangle$ state with an extra relative phase. Let's try to determine what this relative phase is.

Before stating it generally, let's take a closer look at this for a particular example. Let's look at $|\tilde{7}\rangle$ from the 3 qubit Fourier transform we were studying earlier.

Question 35. The input to the QFT circuit will be $|7\rangle = |1\rangle \otimes |1\rangle \otimes |1\rangle$. Write down the relative phase of each qubit after the QFT circuit is applied using the bitstring representation of x.

We can analyze the amplitudes more generally for an n-qubit QFT circuit as follows.

$$\omega^{x\cdot 2^{n-k}} = e^{\frac{2\pi i\cdot 2^{n-k}}{2^n}\cdot x} \tag{33}$$

$$= e^{2\pi i\left[\frac{2^{n-k}}{2^n}\right]\left[x_1\cdot 2^{n-1}+x_2\cdot 2^{n-2}+\cdots+x_n\cdot 2^{n-n}\right]} \tag{34}$$

$$= \prod_{j=1}^{n} e^{2\pi i\left[\frac{2^{n-k}}{2^n}\right]\cdot x_j\cdot 2^{n-j}} \tag{35}$$

$$= \prod_{j=1}^{n} e^{2\pi i\left[\frac{2^{2n-k-j}}{2^n}\right]\cdot x_j} \tag{36}$$

$$= \prod_{j=n-k+1}^{n} e^{2\pi i[2^{n-k-j}]\cdot x_j}. \tag{37}$$

Note the index change in the last line. In the case where $n-k-j<0$, then the exponent is an integer multiple of $2\pi i$, which makes the term in the product always equal 1. The condition can be simplified to $n-k<j$, meaning we can discard the terms in the product less than or equal to $n-k$.

The final line gives insight into what the circuit may need to look like. Since x_j is the j-th bit of x, we see that when $x_j=0$, it will kill off that entire term (set it to 1). In other words, we only want to apply the phase when $x_j=1$. This sounds a lot like a controlled gate!

Question 36. What is the relative phase of the second qubit after applying a QFT circuit to the three qubit input state $|101\rangle$?

The controlled gate we want to apply has to apply a relative phase conditioned on the x_j-th qubit being 1. To do this, let's define a new gate:

$$P_a = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^a} \end{bmatrix}. \tag{38}$$

We have all the pieces now to construct the algorithm.

Algorithm 1 Quantum Fourier Transform

1: **for** $k = 1$ to n **do** ▷ Apply $|x^k\rangle \rightarrow |0\rangle + e^{2\pi i \cdot x \cdot 2^{k-1}} |1\rangle$
2: Apply H to $|x_k\rangle$
3: **for** $j = k + 1$ to n **do**
4: if $x_j = 1$, apply R_{j-k+1}
5: **end for**
6: **end for**
7: Reorder the qubits using swap gates

Question 37. Draw the Quantum Fourier Transform circuit for 4 qubits.

2.2 Properties of the Fourier Transform

Learning Outcomes
Upon following these notes and the corresponding lecture, students will be able to

- describe the property of the function period finding is trying to decide.
- produce and interpret a continued fractions representation of a decimal valued number.
- analyze the period finding algorithm and its correctness.

The power of the QFT is in its ability to *represent* periodic sequences or functions. This may mean that it is also effective at *finding* periodic structure in sequences as well! Before exploring this thought, let's look at some properties of the Fourier Transform.

Let me introduce some more notation we will use through the remainder of this section. We've learned and practiced the mapping from a standard basis state to a Fourier basis state. What happens if we apply the Fourier transform to a superposition of standard basis states?

Question 38. Consider the two qubit state

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle + \alpha_2 |2\rangle + \alpha_3 |3\rangle . \tag{39}$$

What is the result of applying the QFT to this state?

More generally, if we have an n qubit state

$$|\psi\rangle = \sum_{x=0}^{N-1} \alpha_x |x\rangle \tag{40}$$

and apply QFT_N to it, we get the following:

$$QFT_N |\psi\rangle = QFT_N \left(\sum_{x=0}^{N-1} \alpha_x |x\rangle \right) \tag{41}$$

$$= \sum_{x=0}^{N-1} \alpha_x QFT_N |x\rangle \tag{42}$$

$$= \sum_{x=0}^{N-1} \alpha_x \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{x\cdot y} |y\rangle \tag{43}$$

$$= \sum_{y=0}^{N-1} \underbrace{\sum_{x=0}^{N-1} \frac{1}{\sqrt{N}} \alpha_x \omega^{x\cdot y}}_{\widetilde{\alpha_y}} |y\rangle \tag{44}$$

We will call the amplitudes $\widetilde{\alpha_y}$ the Fourier amplitudes. Using the vector notation, this means the QFT is doing the following transformation:

$$\begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{N-2} \\ \alpha_{N-1} \end{bmatrix} \xrightarrow{QFT_N} \begin{bmatrix} \widetilde{\alpha_0} \\ \widetilde{\alpha_1} \\ \vdots \\ \widetilde{\alpha_{N-2}} \\ \widetilde{\alpha_{N-1}} \end{bmatrix} \tag{45}$$

Question 39. Consider the state $|\psi\rangle$ from Question 38. What is $\widetilde{\alpha_3}$?

2.2.1 The Fourier transform converts between translation and phase

Let

$$|\psi\rangle = \sum_{x=0}^{N-1} \alpha_x |x\rangle \tag{46}$$

and

$$|\psi_{+j}\rangle = \sum_{x=0}^{N-1} \alpha_x |x + j \bmod N\rangle = \sum_{x=0}^{N-1} \alpha_x |x + j\rangle . \tag{47}$$

The $+j$ state is simply the state we get by shifting the amplitudes cyclically by j positions. The last inequality is just a notational difference, where we will abbreviate the mod N in this notation.

Question 40. Suppose we have the following two qubit state:

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle + \alpha_2 |2\rangle + \alpha_3 |3\rangle . \tag{48}$$

What is $|\psi_{+1}\rangle$?

Question 41. Apply the QFT to $|\psi_{+j}\rangle$.

Again, let's take a look at what happens to an n-qubit state $|\psi_{+j}\rangle$ and apply the Fourier transform to it.

$$|\widetilde{\psi_{+j}}\rangle = \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} \frac{\alpha_x \omega^{(x+j)\cdot y}}{\sqrt{N}} |y\rangle \tag{49}$$

$$= \sum_{y=0}^{N-1} \omega^{jy} \underbrace{\sum_{x=0}^{N-1} \frac{\alpha_x \omega^{x\cdot y}}{\sqrt{N}}}_{\widetilde{\alpha}_y} |y\rangle \tag{50}$$

$$= \sum_{y=0}^{N-1} \omega^{jy} \widetilde{\alpha_y} |y\rangle . \tag{51}$$

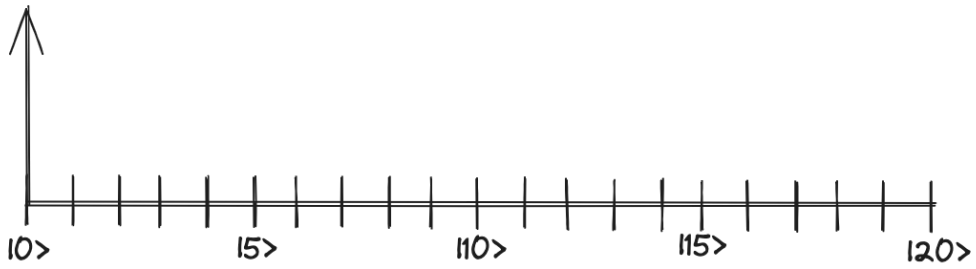
We can think of the index in the standard basis as representing the index of a particular wave x that we use to construct our periodic function. When we translate our position by applying the $+j$ transformation and apply the Fourier transform after this, we now want to pick the $x + j$-th wave instead.

2.2.2 Fourier Transform of Factors

Suppose r is an integer that divides N (N/r is an integer). Define the state

$$|\phi_r\rangle = \sqrt{\frac{r}{N}} \sum_{k=0}^{\frac{N}{r}-1} |kr\rangle \,. \tag{52}$$

Question 42. Let $N = 21$ and $r = 3$. What are the amplitudes that appear in this vector? Draw them in the figure below.



Proposition 2.2. Let r be an integer that divides N. Then,

$$QFT_N \, |\phi_r\rangle = |\phi_{N/r}\rangle \,. \tag{53}$$

Question 43. Continuing the example from Question 42, what is $|\phi_{N/r}\rangle$?

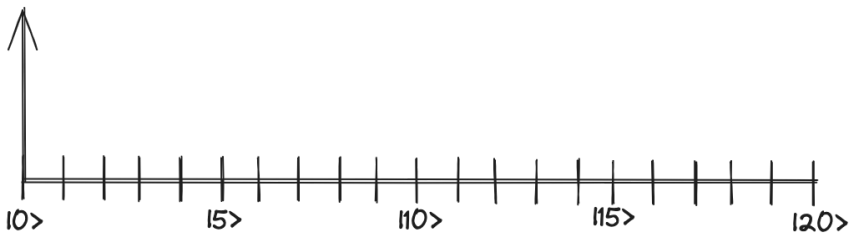
Question 44. Still continuing the example from Question 42, what is $QFT_N \, |\phi_r\rangle$?

2.2.3 Periodic Superpositions with a Shift

The crux of Shor's algorithm is the solution to the following problem: Suppose we have a periodic superposition but with a shift:

$$\sqrt{\frac{r}{N}} \sum_{k=0}^{\frac{N}{r}-1} |kr + l\rangle \,. \tag{54}$$

Can we find an algorithm to find r?



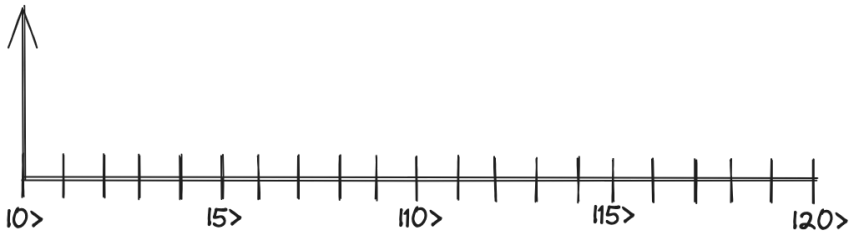
To further complicate the problem, every time we generate a new version of the state, the shift constant l will be different. This causes issues later, because when we use the QFT for factoring, the offset does not allow us to cleanly divide N in the way that it does in Proposition 2.2.

Let's formalize the problem. We consider starting with some state

$$|\phi_r\rangle = \frac{1}{\sqrt{s}} \sum_{k=0}^{s-1} |kr + l\rangle \tag{55}$$

where we have defined $0 \leq l < r$ and $s = \left\lfloor \frac{N}{r} \right\rfloor$. There are two challenges we will have to resolve to correctly figure out what r is.

1. N will generally be a power of 2, meaning that r does not divide N most of the time.
2. The shift l causes the QFT to not work as cleanly.



Proposition 2.3. Consider applying an n-qubit QFT to some state

$$|\phi_r\rangle = \frac{1}{\sqrt{s}} \sum_{k=0}^{s-1} |kr + l\rangle \,. \tag{56}$$

We can express this state as

$$QFT_N \,|\phi_r\rangle = \sum_{a=0}^{N-1} \widetilde{\alpha_a} \,|a\rangle \,. \tag{57}$$

If we measure this output state, then with high probability we measure a value a such that

1. $\left|a - k\frac{N}{r}\right| \leq \frac{1}{2}$ for some k.
2. $\gcd(k, r) = 1$.

The first property can be rewritten as

$$\left|\frac{a}{N} - \frac{k}{r}\right| \leq \frac{1}{2N}. \tag{58}$$

From our measurement result, we know what a is, and N depends on the system size. What the above equation is saying is that $\frac{a}{N}$ is a *very* close approximation of $\frac{k}{r}$. We can use this information to find k and r, independent of what l is!

2.3 Period Finding

Let's look at our first algorithm that uses the QFT. This will be another query based algorithm, meaning that we will have access to some blackbox function, and are trying to decide some property for this function.

Suppose we have some function $f : \{0, \dots, N-1\} \to \{0, \dots, M-1\}$ or alternatively in the bitstring notation, $f : \{0,1\}^n \to \{0,1\}^m$. Here, we've defined the integers such that $N = 2^n$ and $M \leq 2^m$. This function will also have the property that for some integer r,

$$f(x) = f(y) \Leftrightarrow x - y \text{ is an integer multiple of } r \leq M < \sqrt{N}. \tag{59}$$

Given quantum access to this function via a unitary U_f, the goal is to find r.

Example.

This problem is interesting because there is no known efficient classical algorithm! The best we can do is a brute force search for "collisions" $f(x) = f(y)$, which on average will require $\sim 2^{n/2}$ time.

Question 45. Consider the function $f : \{0,1\}^4 \to \{0,1\}^4$, where

$$f(s) = 3^s \mod 16. \tag{60}$$

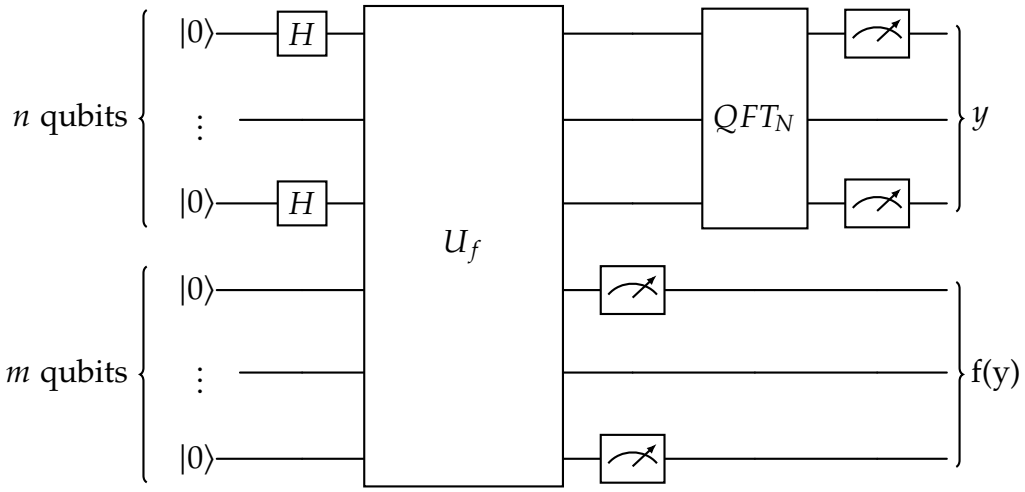
What is the period of f?

I'll begin by just writing out the algorithm, and we will go over the analysis after seeing it. The algorithm will use two registers, the first using n qubits (to store numbers mod N), and the second register will have $\lceil \log M \rceil$ qubits to store numbers mod M. We will also have quantum access to the function f via a unitary U_f which has the action

$$U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle . \tag{61}$$

Algorithm 2 Period Finding

1: Start with $|0 \dots 0\rangle |0 \dots 0\rangle$
2: Apply $H^{\otimes n}$ on the first register to get $\frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} |y\rangle |0\rangle$
3: Apply U_f
4: Measure the 2nd register.
5: Ignore the 2nd register and apply $QFT_N (\text{mod } N)$ to the 1st register.
6: Measure the 1st register to get value a.
7: Postprocessing: Use a and N to find k and r.



Let's go through each step starting at 3 to see what the state is. At step 3, we will have prepared the state

$$\frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} |y\rangle |f(y)\rangle \tag{62}$$

Now recall that the function $f(y)$ is periodic in r, meaning that for every $f(y)$, there will be r inputs that map to it.

Question 46. Consider the function we used in Question 45. If we had unitary access to this function and measured $|9\rangle$ in the second register, what is the state of the algorithm?

More generally, measuring the second register will allow us to collapse into a superposition of inputs that all map to the same state. More concretely, suppose we measured the value w in the second register. Then the first register will be an even superposition of all $|jr + l\rangle$ such that $f(jr + l) = w$, and $0 \leq j < \lfloor \frac{N}{r} \rfloor$, $0 \leq l \leq r - 1$. We can write this in ket notation too:

$$\frac{1}{\sqrt{s}} \sum_{j=0}^{s-1} |jr + l\rangle |w\rangle \qquad s := \left\lfloor \frac{N}{r} \right\rfloor. \tag{63}$$

We can safely ignore the second register for the remaining analysis, because the two registers are now in a product state.

By Proposition 2.3, we know that applying a QFT to this state and measuring will allow us with high probability to observe a state a such that

$$\left| a - k\frac{N}{r} \right| \leq \frac{1}{2} \qquad \gcd(k, r) = 1. \tag{64}$$

The above equation can be combined with our assumption that $r < \sqrt{N}$ to get

$$\left| \frac{a}{N} - \frac{k}{r} \right| \leq \frac{1}{2N} \leq \frac{1}{2r^2}. \tag{65}$$

We will be using **continued fractions** to find k and r.

We will assume that $\gcd(k, r) = 1$. To analyze the final step, let's look at how continued fractions works. Let $\gamma = \frac{a}{N}$. We know that with high probability,

$$\left| \gamma - \frac{k}{r} \right| \leq \frac{1}{2r^2}. \tag{66}$$

The continued fractions representation of a real number is a sequence of integers $a_0, a_1, \ldots, a_n$ as:

$$\gamma \approx a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots \frac{1}{a_n}}}} \tag{67}$$

We can clip this sequence at some a_k to get an approximation of γ. We will define the P_k and Q_k as the numerator and denominator we get respectively for the approximation of γ.

Question 47. Let $\gamma = 7.27$. Find the continued fractions representation of γ. What is P_3 and Q_3?

Once the continued fractions representation is found, we get a series of approximations to γ:

$$\frac{P_0}{Q_0}, \frac{P_1}{Q_1}, \frac{P_2}{Q_2}, \cdots, \frac{P_n}{Q_n} \tag{68}$$

such that

$$Q_0 < Q_1 < Q_2 < \cdots < Q_n. \tag{69}$$

Each of the Q_j's are candidates for r, and with access to the function f we can directly test the n candidates to see if we have the correct period. We know this will happen by the theorem stated below.

Theorem 2.4 (Proven in Nielsen and Chuang)**.** If $\left|\gamma - \frac{k}{r}\right| \leq \frac{1}{r^2}$, then $k = P_j$ and $r = Q_j$ for some j.

Our algorithm will try each one and take the smallest Q_j such that

$$x^{Q_j} = 1 \mod N. \tag{70}$$

We end the section with two important facts:

- If γ is a rational number, eventually for some n, $\frac{P_n}{Q_n} = \gamma$.
- $\frac{P_j}{Q_j}$ is the best approximation to γ by *any* ratio of integers whose denominator is $\leq Q_j$.

2.4 RSA and Number Theory

"We know more than we did before. Let's use that."

- Cypher

Learning Outcomes
Upon following these notes and the corresponding lecture, students will be able to

- explain how the RSA algorithm works, and describe why it is difficult for classical algorithms to break.
- apply properties about integers mod N.
- analyze the period finding algorithm and its correctness.

We've seen an example of private key cryptography when talking about quantum money, but here we will be interested in **public key cryptography**. In such a scheme, there exists what is called a public key, which anyone can easily know and uses to encrypt a message. On the other hand, each receiving party will have their own private key, which is required to efficiently decrypt a message.

In this section, we will take a look at the RSA protocol, which is one of the most commonly used public key cryptosystems today.

Question 48. Suppose the public keys are $e = 5$ and $M = 26$. If we encrypt the message "B" which we will represent using the integer 2, what is the cypher text?

Question 49. Show that $d = 17$ is a valid private key to decrypt the message.

More generally, the following is the description of the algorithm.

1. Choose two prime numbers p and q.
2. Multiply them to form M.
3. Compute the "Euler function" of M, $\phi(M)$.

$$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26$$

4. Choose an encryption key e such that
   - $1 < e < \phi(M)$.
   - $\gcd(e, M) = 1$ and $\gcd(e, (p-1)(q-1)) = 1$.
5. Choose d such that

$$de (\text{mod } \phi(N)) = 1. \tag{71}$$

- Public keys are e and M.
- Private keys are p, q, d.

If you know p, q, e, you can use Euclid's algorithm to efficiently compute d.

To encrypt a message, we would use the following protocol. Let m be the plaintext message. The cypher text c is computed by

$$c = m^e \mod M. \tag{72}$$

To decrypt the message, we simply perform

$$m = c^d \mod M. \tag{73}$$

If an attacker intercepted the cypher text c, it would be practically impossible to decode it unless they knew what d was.

The computer science community believes that finding the plaintext m from the cyphertext c requires solving an exponentially hard problem. Knowing whether this is true or not has large implications for the security of information, leading to lots of interest in the complexity of the factoring problem.

As usual, let's take a look at how difficult it would be to factor numbers classically. Note that when we analyze the hardness of factoring, we are interested in the number of digits used to represent M, which is $w = O(\log M)$. So an efficient algorithm would mean an algorithm that runs in polynomial time with respect to $w = O(\log M)$.

The trivial algorithm would simply try every number $j \in [1, \sqrt{M}]$ and check if j divides M. This would require $O(2^w)$ iterations.

Other known classical algorithms include

- Quadratic Field Sieve: $O\left(2^{c\cdot\sqrt{w}}\right)$
- Number Field Sieve: $O\left(2^{m^{1/3}}\right)$

Though still exponential, these improvements were enough to require RSA schemes to go up from 512- to 768-bit encryption schemes to the sizes we see today.

In 1994, Peter Shor developed an algorithm to solve factoring on a quantum computer using just poly$(\log M)$ gates. Note that this is not even the number of queries, it is the exact circuit size.

Going beyond RSA, another popular public-key cryptosystem is called Diffie-Hellman, which requires solving the discrete log problem (also believed to be difficult for classical computers). Shor's algorithm for factoring is also able to solve discrete log.

Shor's algorithm critically uses period finding, using a reduction from period finding to factoring. In other words, he shows how an efficient algorithm for period finding can be used to solve factoring.

We know that the quantum algorithm for period finding only uses $O(1)$ queries to f. We would like to see if the unitary representing the query can be efficiently implemented. This is possible to analyze because we are interested in a particular function when factoring, namely searching for the period of the function

$$f_x(s) := x^s \mod M \tag{74}$$

where $\gcd(x, M) = 1$ and $M = p \cdot q$. If we can find an efficient classical algorithm to compute $x^s \mod M$, we will know what the classical circuit looks like, then convert it into a reversible gate which will represent U_f.

In this section, we'll be proving some basic number theoretic facts related to the factoring problem. First, we need to confirm that $f_x(s)$ defined above is indeed periodic.

Lemma 2.5. Let x, M be integers such that $\gcd(x, M) = 1$. Then

$$x^s \mod M \tag{75}$$

is periodic in x.

Sketch. If we tried computing the powers of $x \bmod M$, since the function can have only M distinct outcomes, if we take more than M powers we will eventually get a repeat. Let a and b be two powers where the equation evaluates to the same integer. We can express this mathematically as

$$x^a \mod M = x^b \mod M \tag{76}$$

$$x^b - x^a = k \cdot M \tag{77}$$

$$x^a \left(x^{b-a} - 1\right) = k \cdot M \tag{78}$$

for some integer k. Since $\gcd(x, M) = 1$, x and M do not share any prime factors. Thus for the equality to hold, M must evenly divide $x^{b-a} - 1$:

$$x^{b-a} - 1 = k'M \tag{79}$$

$$x^{b-a} = 1 + k'M \tag{80}$$

$$\Rightarrow x^{b-a} = 1 \mod M. \tag{81}$$

□

Furthermore, we can show that the period of the function is the smallest positive integer s such that $x^s = 1 \mod M$.

In practice, finding a non-trivial square root of 1 mod M is sufficient for factoring.

Lemma 2.6. Given a composite number N and an integer x such that

$$x^2 = 1 \mod N \tag{82}$$

and

$$x \neq \pm 1 \mod N, \tag{83}$$

we can factor N.

Proof. By our assumption, we have that

$$x^2 - 1 = kN \tag{84}$$

$$\Rightarrow (x+1)(x-1) = kN. \tag{85}$$

Furthermore by our second assumption that $x \neq \pm 1 \mod N$, neither $x-1$ nor $x+1$ is a multiple of N. The product is some multiple of N, so what we conclude is that each of $(x-1)$ and $(x+1)$ have some of N's prime factors. To find one of these, we simply compute

$$\gcd(x+1, N) \tag{86}$$

$$\gcd(x-1, N). \tag{87}$$

□

Example 2.7. Let $N = 15$. A number that satisfies the first condition is $x = 11$:

$$11^2 = 121 = 1 \mod 15. \tag{88}$$

We can also easily verify that $11 \neq \pm 1 \mod 15$. Now we compute the gcd of the pair of integers around 11 with 15 to recover the factors:

$$\gcd(12, 15) = 3 \tag{89}$$

$$\gcd(10, 15) = 5. \tag{90}$$

We have successfully recovered the factors 3 and 5.

Note that for an application like RSA, the number N we are trying to factor will always be a composite of two primes, meaning that the gcd will be sufficient for finding these numbers.

Question 50. Find a non-trivial square root of 1 mod 20.

2.5 Shor's Algorithm

We now have all the required pieces to see the full algorithm for factoring by Peter Shor.

Algorithm 3 Factoring

Pick x at random from $\{2, \ldots, N-1\}$
if $\gcd(x, N) \neq 1$ **then**
    $\gcd(x, N)$ is a non-trivial factor of N so we are done!
else
    Use quantum Period Finding algorithm (alg 2) to find smallest s such that $x^s = 1 \mod N$.
    Call this variable r.
    if r is odd **then**
        Start over..
    else if $x^{r/2} = \pm 1 \mod N$ **then**
        Start over..
    else
        $x^{r/2} \mod N$ is a non-trivial square root of 1 mod N ($\gcd(x^{r/2} - 1, N)$.
    end if
end if

How likely is it that our algorithm actually finishes? For any N that is not a power of a prime, if x is chosen at random from $\mathbb{Z}_N^*$ and r is the smallest s such that $x^s = 1 \mod N$, then with probability $\geq 3/8$,

- r is even, and
- $x^{r/2} \neq \pm 1 \mod N$.

❖ Computation 3: Quantum Search - Grover's Algorithm

Learning Outcomes
Upon following these notes and the corresponding lecture, students will be able to

- model NP complete problems as search problems.
- describe how to translate this search problem into the query model.
- describe geometrically how a quantum algorithm can accomplish search in $O(2^{n/2})$ queries.
- construct circuits to implement Grover's algorithm.
- describe the implications of generalizing the search problem for Grover's algorithm.

3.1 The Search Problem

Once Shor's algorithm was discovered, many researchers became very interested in the potential of quantum computers to solve problems that are believed to be hard for classical computers to solved. Formally, the question being asked was the following: "Can quantum computers solve NP-complete problems?"

Example 3.1. 3-SAT

- **Input:** A Boolean formula ϕ in 3-CNF form consisting of n-Boolean variables: $\phi(x_1, x_2, \ldots, x_n)$.

$$\text{3-CNF: } (x_2 \vee \overline{x}_7 \vee x_3) \wedge (x_5 \vee x_7 \vee x_9) \wedge \cdots \tag{91}$$

- **Output:** Does ϕ have a satisfying assignment? That is, is there a way of setting each x_i such that ϕ evaluates to TRUE?

I can "prove" to you that there is a satisfying assignment by giving you an assignment of the variables $x_1, \ldots, x_n$, which you just need to plug into the function and check for yourself.

The above example is important because not only is it in NP, it is actually a problem in a subset of NP problems called NP-complete. NP-complete problems are often referred to as "the hardest problems in NP", and is a significantly important set of problems because we know that an efficient solution to *any* NP-complete problem would mean there is an

efficient solution to *every* problem in NP! It is generally believed that it is impossible to solve NP-complete problems in polynomial time using a classical computer.

The final algorithm we study this class is again a query based algorithm. Per usual, we will have black-box access to the function $f : \{0,1\}^n \to \{0,1\}$ we are interested in in the form of a unitary U_f:

$$U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle . \tag{92}$$

We are interested in deciding some property of the function. For this last setting, we are interested in the search problem:

Definition 3.2 (Search Problem)**.** Let $f : \{0,1\}^n \to \{0,1\}$. Is there an n bit string x such that $f(x) = 1$?

This problem is interesting because any decision problem (including NP-complete ones) can be formulated in this way. For example, if we have a 3-SAT instance we can set $f(x_1 \cdots x_n) = \phi(x_1, \cdots, x_n)$. To simplify the analysis, we will first assume that f either has a unique solution or no solution at all. In other words,

$$|\{x | f(x) = 1\}| = 0 \text{ or } 1. \tag{93}$$

In the case that a solution does exist, we will denote it by the variable a.

Question 51. What is the classical query complexity for the above problem?

You may be wondering if this is *too* general for analyzing NP-complete problems, but there is a widely believed hypothesis that a brute force search is the best we can do for 3SAT. In other words, we believe that the lower bound shown above for general f holds for functions that are NP-complete too. We now want to ask what kind of speed-up is possible if we use a quantum computer in this setting.

3.2 Grover's Algorithm

Let $|a\rangle$ be the standard basis state representing the value such that $f(a) = 1$. We will also use the notation

$$|\psi\rangle := \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle \tag{94}$$

to represent the uniform superposition state.

One key fact about this algorithm is that the state of our system will **always** stay in the space spanned by $|a\rangle$ and $|\psi\rangle$.

Question 52. Letting $|v\rangle$ be the state of our system at any given time step, how can we express the above fact mathematically?

Note that $|a\rangle$ and $|\psi\rangle$ are not orthogonal.

Question 53. What is the overlap between $|a\rangle$ and $|\psi\rangle$? Express this using trigonometric functions. Draw the two vectors, with $|a\rangle$ as a vector pointing upward, and $|\psi\rangle$ in the right direction. What does this say about the **probability** of measuring $|a\rangle$?

We can create an orthonormal basis that represents the same subspace by subtracting the $|a\rangle$ component from $|\psi\rangle$.

Question 54. Create a new state $|e\rangle$ that is orthonormal to $|a\rangle$ by subtracting the $|a\rangle$ component from $|\psi\rangle$.

The last preparation step we need is to express the angle between $|e\rangle$ and $|\psi\rangle$. For very small values of θ, it is known that

$$\sin\theta \approx \theta. \tag{95}$$

From this, we can conclude that the angle formed between $|\psi\rangle$ and $|e\rangle$ is approximately θ.

The main steps of the algorithm can be described purely geometrically. We will discuss how to implement these steps later, but for now let's build the intuition behind what the algorithm is doing. I will use $|v\rangle$ to represent the current state of the system. Here are the steps that we repeat:

1. Reflect $|v\rangle$ over the $|e\rangle$ axis.
2. Reflect $|v\rangle$ over $|\psi\rangle$.

Question 55. Let θ be the angle between $|v\rangle$ and $|e\rangle$ before applying the two steps above. What is the angle between the two *after* we apply the two steps?

Question 56. What angle between $|v\rangle$ and $|e\rangle$ would give us the highest probability to measure $|a\rangle$?

3.2.1 Implementing Reflections

How can we implement the reflections? Before discussing that, let's think about how to express the action of these reflections mathematically. Suppose we have a state $|v\rangle$ that we want to reflect over some other state $|\phi\rangle$.

Question 57. Write down $|v\rangle$ as a superposition of $|\phi\rangle$ and $|\phi^{\perp}\rangle$. Write down the state $|v'\rangle$ after the reflection is completed.

3.2.2 Reflection over $|e\rangle$

The first reflection we need to do is over the $|e\rangle$ axis. That is, we want

$$|v\rangle = \alpha\,|e\rangle + \beta\,|a\rangle \tag{96}$$

$$|v'\rangle = \alpha\,|e\rangle - \beta\,|a\rangle \tag{97}$$

$$\tag{98}$$

In words, we want to multiply the $|a\rangle$ state by -1, and the rest of the states by +1. Remember that a is the unique bitstring satisfying $f(a) = 1$. We can summarize this action by

$$|v\rangle = \sum_x \alpha_x\,|x\rangle \longrightarrow |v'\rangle = \sum_x \alpha_x(-1)^{f(x)}\,|x\rangle\,. \tag{99}$$

Question 58. We know we can perform the above operation with access to U_f. U_f acts on $n+1$ qubits. What state does this qubit need to be in for us to apply the phase of $(-1)^{f(x)}$?

3.2.3 Reflection over $|\psi\rangle$

The mapping we want to perform here is

$$|v\rangle = \alpha |\psi\rangle + \beta |\psi^{\perp}\rangle \tag{100}$$

$$|v'\rangle = \alpha |\psi\rangle - \beta |\psi^{\perp}\rangle \tag{101}$$

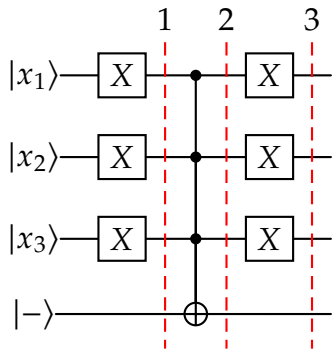
$$\tag{102}$$

where $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle$.

Since $|\psi\rangle$ is generated by applying n Hadamard gates, the following is true:

$$H^{\otimes n} |0 \cdots 0\rangle = |\psi\rangle \iff |0 \cdots 0\rangle = H^{\otimes n} |\psi\rangle . \tag{103}$$

If we apply n Hadamard gates to $|\psi^{\perp}\rangle$, we get a state $|\phi\rangle$ which is a uniform superposition over all states perpendicular to $|0 \cdots 0\rangle$. The goal is to now apply a phase of -1 to every standard basis state which is not equal to $|0 \cdots 0\rangle$. I claim that the following 3 qubit circuit accomplishes this:



If input state is $|000\rangle$:

If input state is not $|000\rangle$:

3.3 Generalized Search

What about the (more likely) case where there are multiple solutions? That is,

$$|\{x|f(x) = 1\}| = M > 1. \tag{104}$$

Assume (for now) that the number of solutions M is known. We would like to find any one of these solutions. We again define two orthogonal states, the uniform superpositions of the solution states and non-solution states respectively:

$$|\phi_1\rangle = \frac{1}{\sqrt{M}} \sum_{x:f(x)=1} |x\rangle \quad |\phi_0\rangle = \frac{1}{\sqrt{N-M}} \sum_{x:f(x)=0} |x\rangle . \tag{105}$$

Then our starting uniform superposition state can be expressed as a weighted sum of the above two states as

$$|\psi\rangle = \sqrt{\frac{N-M}{N}} |\phi_0\rangle + \sqrt{\frac{M}{N}} |\phi_1\rangle . \tag{106}$$

Since we know M, we can select the number of iterations c such that

$$(2c+1)\theta \approx \frac{\pi}{2} \tag{107}$$

in the same way we did in the previous section. In the case where $M << N$, $\theta \approx \sqrt{\frac{M}{N}}$, so the total number of iterations required is $O\left(\sqrt{\frac{M}{N}}\right)$.

Note: Since each iteration of Grover's algorithm advances the current state by 2θ, we can only guarantee that the final state is within θ of $|\phi_1\rangle$.

In the next section, we will look at an algorithm which will let us handle the case where M is unknown.

Before that though, we must resolve the question we started the section with. From the results in this section, Grover's algorithm does not provide us with an exponential speed-up over classical algorithms for search problems in the query model. Is there a quantum algorithm which can outperform Grover? It turns out this is not possible, and Grover's algorithm is "the best you can do" in this query model. There is a lower bound

(which we will not prove here) that states that even if you are guaranteed that the $|s_f|$ (the number of satisfying solutions) is 0 or 1, there is not much we can do to improve.

More formally, the lower bound states that any quantum circuit that can determine between $|s_f| = 0$ and $|s_f| = 1$ with high probability will require $\Omega\left(\sqrt{N}\right)$ queries to U_f.

3.4 Amplitude Estimation

In the previous section, we mentioned that if we want to *find* an x such that $f(x) = 1$, we need to have an approximate idea of what M is. There is a standard method for accomplish this that uses the QFT, but instead of discussing that I will outline a result that was published by myself and Sandy, along with two (then) undergrads who took this class two years ago.

Our algorithm is **iterative**, meaning that it jumps between a parameterized quantum portion and a classical postprocessing portion. Let a be the solutions, and θ_a be the angle formed between the starting state $|\psi\rangle$ and the superposition $|\phi_0\rangle$ over states that evaluate to 0. The two are related by the expression

$$a = \sin\theta_a. \tag{108}$$

If we know a, we know how many queries to make in the generalized search algorithm. The algorithm will output an estimate for θ_a. We do this by maintaining lower and upperbounds for what θ_a is.

Picture of how the algorithm works: