Quantum Computing: Foundations to Frontier Fall 2018

Lecture 4

Lecturer: Henry Yuen *Scribes: Deeksha Adil, Xing Hu, Jinhui Li, Fengwei Sun*

1 Overview

In the last lecture we were looking at Simon's algorithm. In this lecture we complete the analysis of Simon's algorithm and move on to Quantum Fourier Transform and its applications.

2 Simon's Algorithm (continued)

Recall that Simon's algorithm solves the following problem. The input is a function $f : \{0,1\}^n \to \{0,1\}^n$ with the property, $\exists s \neq 0$ such that $\forall x, y$,

$$f(x) = f(y) \iff \begin{cases} x = y, & \text{or} \\ x = y \oplus s. \end{cases}$$

The problem is to find s. Simon's algorithm outputs s in polynomial time. To solve the problem classically, any algorithm has to at least find a collision if it has to recover s and for that we need at least $\Omega(2^{n/2})$ queries. Therefore, any classical algorithm requires exponentially many queries to f to find s. In the last lecture we were analyzing the main subroutine of the algorithm.

2.1 Basic Subroutine

Let us briefly recall the subroutine.

1. Apply the Hadamard operator to each of the first n qubits.
2. Next apply U_f to all $2n$ qubits.
3. Again apply the Hadamard operator to each of the first n qubits.
4. Finally measure the first n qubits.

In the previous lecture we had seen that after step 3 the state looks like,

$$|\psi\rangle = \frac{1}{2^n} \sum_y |y\rangle \left(\sum_x (-1)^{x \cdot y} |f(x)\rangle \right).$$

Now, it remains to analyse the outcome of measurement. When we measure we get a string $y \in \{0,1\}^n$ with probability,

$$P(y) = \left\| \frac{1}{2^n} \sum_x (-1)^{x \cdot y} |f(x)\rangle \right\|^2.$$

Let A denote the image space of the function f i.e., $A = \{z : \exists x, f(x) = z\}$. For every $x \in \{0,1\}^n$, there exists $x' = x \oplus s \in \{0,1\}^n$ such that $f(x) = f(x')$. Therefore, we have 2^{n-1} values in the range of A, i.e., $1/2 \times |\{0,1\}^n|$. For every value $a \in A$, let $x_1(a), x_2(a)$ denote the two strings for which $f(x) = a$. As a result of the property of the function f, we must have $x_2(a) = x_1(a) \oplus s$.

$$\begin{aligned}
P(y) &= \left\| \frac{1}{2^n} \sum_{a \in A} \left((-1)^{x_1(a)\cdot y} + (-1)^{x_2(a)\cdot y} \right) |a\rangle \right\|^2 \\
&= \left\| \frac{1}{2^n} \sum_{a \in A} \left((-1)^{x_1(a)\cdot y} + (-1)^{(x_1(a)\oplus s)\cdot y} \right) |a\rangle \right\|^2 \\
&= \left\| \frac{1}{2^n} \sum_{a \in A} \left((-1)^{x_1(a)\cdot y} + (-1)^{x_1(a)} (-1)^{s\cdot y} \right) |a\rangle \right\|^2 \\
&= \left\| \frac{1}{2^n} \sum_{a \in A} (-1)^{x_1(a)\cdot y} \left(1 + (-1)^{s\cdot y}\right) |a\rangle \right\|^2
\end{aligned}$$

Note that, $1 + (-1)^{s\cdot y} = 0$ when $s \cdot y = 1 \mod 2$ and $1 + (-1)^{s\cdot y} = 2$ otherwise. Using this, we get,

$$\begin{aligned}
P(y) &= \begin{cases} 0 & \text{if } s \cdot y = 1 \mod 2 \\ \frac{4}{2^{2n}} \left\| \sum_{a \in A} (-1)^{x_1(a)\cdot y} |a\rangle \right\|^2 & \text{otherwise} \end{cases} \\
&= \begin{cases} 0 & \text{if } s \cdot y = 1 \mod 2 \\ \frac{4}{2^{2n}} |A| & \text{otherwise} \end{cases} \\
&= \begin{cases} 0 & \text{if } s \cdot y = 1 \mod 2 \\ \frac{1}{2^{n-1}} & \text{otherwise.} \end{cases}
\end{aligned}$$

There are 2^{n-1} strings y such that its inner product with s is non zero. When we measure at step 4 of the subroutine we get one such string y uniformly at random.

2.2 The Algorithm

We now look at the main algorithm.

1. Run the basic subroutine $m = 100n^2$ times. This gives us $y_1, y_2, ..., y_m$ that satisfy $y_i \cdot s = 0 \mod 2$
2. Solve the system of linear equations for s using any standard technique such as Gaussian Elimination (with arithmetic done modulo 2).

Running the subroutine $O(n^2)$ times, with high probability we get $n - 1$ linearly independent equations and we can easily solve for s.

3 Quantum Fourier Transform (QFT)

Simply speaking, Fourier transform is a way to break up functions into simpler pieces (called frequencies). In quantum computing, the quantum Fourier transform is an algorithm to apply linear transformation on quantum bits, and is the quantum analogue of the discrete Fourier transform.

3.1 Discrete Fourier Transform

Given n numbers: $\psi_0, \psi_1, \ldots, \psi_{N-1} \in \mathbb{C}^N$. We can interpret ψ_x as a discrete signal, then

$$\psi_x = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \widehat{\psi_k} e^{\frac{-2\pi i k x}{N}}$$

$$x = 0, \ldots, N-1$$

The coefficients $\{\widehat{\psi_k}\}$ are called the Fourier coefficients of $\{\psi_x\}$ and represents the contributions of each frequency function $e^{\frac{-2\pi i k x}{N}}$. Let $\phi_k(x) = \frac{1}{\sqrt{N}} e^{\frac{-2\pi i k x}{N}}$ (the frequency functions), then $\phi_k(x)$ can be interpreted as vectors in $\mathbb{C}^N$ and $\{\phi_k(x)\}$ form an orthonormal basis.

Now ψ_x can be written as $\psi_x = \sum_{k=0}^{N-1} \widehat{\psi_k} \phi_k(x)$.

3.2 Quantum Fourier Transform

In quantum set up, if we are working in n-qubit space, then the Hilbert space is $(\mathbb{C}^2)^{\otimes n} \cong \mathbb{C}^{2^n}$. Let $N = 2^n$, $|\psi\rangle = \sum_{x=0}^{N-1} \psi_x |x\rangle$. Then $|\psi\rangle = \sum_{k=0}^{N-1} \widehat{\psi_k} |\phi_k\rangle$, where $|\phi_k\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{\frac{-2\pi i k x}{N}} |x\rangle$. The Fourier transform is simply the change of basis: $F : |\phi_k\rangle \longrightarrow |k\rangle$, which is an unitary operator s.t.

$$F|\psi\rangle = \sum_{k=0}^{N-1} \widehat{\psi_k} F|\phi_k\rangle = \sum_{k=0}^{N-1} \widehat{\psi_k} |k\rangle = |\widehat{\psi}\rangle$$

The quantum Fourier transform (QFT) is an algorithm that implements this unitary F.

Question: Why is this useful?

If we measure $|\widehat{\psi}\rangle$, we get outcome $|k\rangle$ with probability $|\widehat{\psi_k}|^2$. However, we cannot really figure out ψ_k because quantum states are fragile: when you measure $|\widehat{\psi}\rangle$, the quantum state collapses and the information about $\widehat{\psi_k}$'s disappears. So why is it useful? QFT is useful because it can perform Fourier transform on exponentially long vectors ($N = 2^n$- dimensional vector) by implementing F with only $poly(n)(= polylog(N))$ number of gates and ancillas. In other words, it trades off between the ability to directly access the Fourier coefficients of ψ and the ability to compute (in a sense) the Fourier transform of an exponentially large vector.

Let F_N denote the $N \times N$ Fourier transform unitary that does the transformation:

$$F_N|x\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \exp\left(\frac{2\pi i k x}{N}\right) |k\rangle$$

where x, k are n-bit numbers.

$F_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} =$ Hadamard gate

$$F_4 = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \to \text{(reordering columns)} \to \frac{1}{\sqrt{4}} \left[\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & -1 & i & -i \\ \hline 1 & 1 & -1 & -1 \\ 1 & -1 & -i & i \end{array}\right] = \frac{1}{\sqrt{4}} \begin{bmatrix} F_2 & A_2 F_2 \\ F_2 & -A_2 F_2 \end{bmatrix}$$

where we define

$$A_2 = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}.$$

For F_4, in the first matrix we wrote, the columns are associated with the basis states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ (in that order). If we reorder the columns so that the ones corresponding to basis states $|00\rangle$ and $|10\rangle$ are on the left, and the columns corresponding to basis states $|01\rangle$ and $|11\rangle$ are on the right, then we get the matrices on the right. We're not changing the linear operator, but just reordering the columns just so it's evident that F_4 has a nice recursive structure.

This recursive structure holds for F_N when $N = 2^n$:

$$F_N = \frac{1}{\sqrt{N}} \begin{bmatrix} F_{\frac{N}{2}} & A_{\frac{N}{2}} F_{\frac{N}{2}} \\ F_{\frac{N}{2}} & -A_{\frac{N}{2}} F_{\frac{N}{2}} \end{bmatrix}$$

where for all M, we define

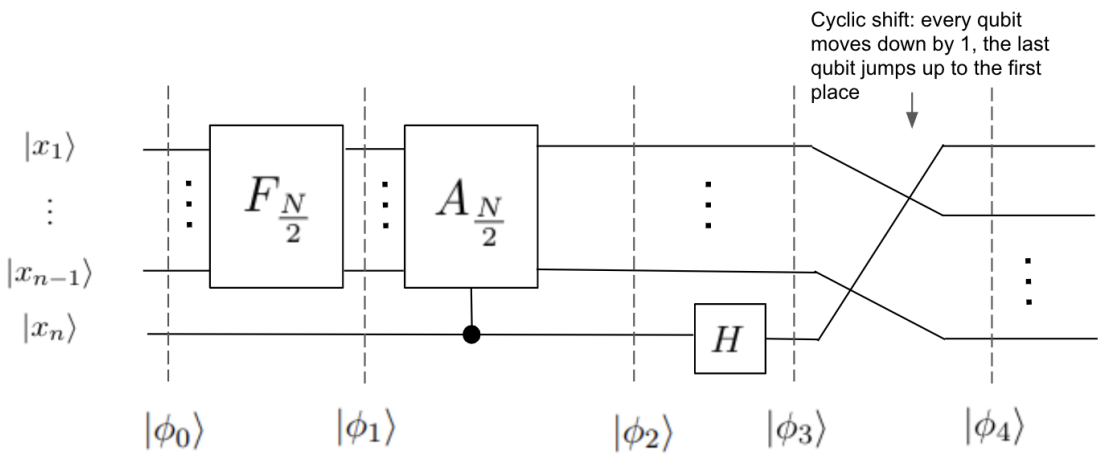
$$A_M = \begin{bmatrix} 1 & & & & \\ & \omega_{2M} & & & \\ & & \omega_{2M}^2 & & \\ & & & \ddots & \\ & & & & \omega_{2M}^{M-1} \end{bmatrix}$$

with $\omega_{2M} = e^{\frac{2\pi i}{2M}} = e^{\frac{\pi i}{M}}$.

Again, this recursive structure appears if we order the columns so that the "even" columns (columns corresponding to basis states that end in 0) are on the left, and the "odd" columns are on the right.

3.3 QFT Circuit

Assuming that there's a circuit for unitaries, $F_{\frac{N}{2}}$ and $A_{\frac{N}{2}}$, the circuit below implements a circuit for F_N.



3.4 Circuit Analysis

Now we will show that the circuit above actually implements the unitary F_N. We prove this by calculating the action of the circuit on all basis states $|x_1, \ldots, x_n\rangle$.

Case 1: $x_n = 0$**.** Let $y = (x_1, \ldots, x_{n-1})$ be the $(n-1)$-bit prefix of x. Then the states of the circuit are as follows:

- Beginning: $|x\rangle = |x_1, \ldots, x_n\rangle = |y, 0\rangle$
- After applying $F_{N/2}$: $(F_{N/2}|y\rangle) \otimes |0\rangle$
- After the controlled $A_{N/2}$ (which)doesn't do anything): $(F_{N/2}|y\rangle) \otimes |0\rangle$.
- After applying H on the last qubit: $\frac{1}{\sqrt{2}} F_{N/2}|y\rangle \otimes (|0\rangle + |1\rangle)$.
- After the shift: $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes F_{N/2}|y\rangle$

Why is this correct?

$$\begin{aligned} F_{N/2}|y\rangle &= \frac{1}{\sqrt{N/2}} \sum_{j=0}^{N/2-1} \exp\left(\frac{2\pi i j y}{N/2}\right) |j_1 \ldots j_{n-1}\rangle \\ &= \frac{1}{\sqrt{N/2}} \sum_{j=0}^{N/2-1} \exp\left(\frac{2\pi i j (2y)}{N}\right) |j_1 \ldots j_{n-1}\rangle \\ &= \frac{1}{\sqrt{N/2}} \sum_{j=0}^{N/2-1} \exp\left(\frac{2\pi i j x}{N}\right) |j_1 \ldots j_{n-1}\rangle \end{aligned}$$

The last equality holds because

$$x = \sum_{i=1}^{n} x_i 2^{n-i} = \sum_{i=1}^{n-1} x_i 2^{n-i} = 2 \sum_{i=1}^{n-1} x_i 2^{n-i-1} = 2y.$$

Since j is a $(n-1)$-bit number, it is interpreted as $j = j_1 2^{n-2} + j_2 2^{n-3} + \ldots + j_{n-1}$. Plugging this back in, the final state of the circuit is

$$= \frac{1}{\sqrt{N}}(|0\rangle + |1\rangle) \otimes \sum_{j=0}^{N/2-1} \exp\left(\frac{2\pi ijx}{N}\right) |j_1 \ldots j_{n-1}\rangle$$
$$= \frac{1}{\sqrt{N}} \left(\sum_{j=0}^{N/2-1} \exp\left(\frac{2\pi ijx}{N}\right) |0, j_1, \ldots, j_{n-1}\rangle + \sum_{j=0}^{N/2-1} \exp\left(\frac{2\pi ijx}{N}\right) |1, j_1, \ldots, j_{n-1}\rangle \right)$$

Claim: Fix $j = (j_1, \ldots, j_{n-1})$. For $i = 0, 1$ let $k_i = (i, j_1, \ldots, j_{n-1})$. Then

$$\exp\left(\frac{2\pi ijx}{N}\right) = \exp\left(\frac{2\pi ik_0 x}{N}\right) \qquad \text{obvious because as integers } k_0 = j$$
$$= \exp\left(\frac{2\pi ik_1 x}{N}\right) \qquad k_1 = 2^{n-1} + j_1 2^{n-2} + \ldots + j_{n-1}$$

but notice that

$$\exp\left(\frac{2\pi ik_1 x}{N}\right) = \exp\left(\frac{2\pi i(2^{n-1} + j)x}{N}\right)$$
$$= \exp\left(\frac{2\pi i 2^{n-1} x}{N}\right) \exp\left(\frac{2\pi ijx}{N}\right)$$
$$= \exp\left(\frac{2\pi i 2^n y}{N}\right) \qquad x = 2y$$
$$= 1$$

Then we can rewrite our state as

$$\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \exp\left(\frac{2\pi ikx}{N}\right) |k\rangle$$

which is exactly $F_N|x_1 \ldots x_n\rangle$, as desired.

Case 2: $x_n = 1$**.** Let $y = (x_1, \ldots, x_{n-1})$, and note that $x = 2y + 1$. Then we have the output of the circuit being

$$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes A_{N/2} F_{N/2} |y\rangle$$

Let's calculate this:

$$= \frac{1}{\sqrt{N}}(|0\rangle - |1\rangle) \otimes \sum_{j=0}^{N/2-1} \exp\left(\frac{2\pi ijy}{N/2}\right) A_{N/2} |j_1 \ldots j_{n-1}\rangle$$
$$= \frac{1}{\sqrt{N}}(|0\rangle - |1\rangle) \otimes \sum_{j=0}^{N/2-1} \exp\left(\frac{2\pi ij(2y+1)}{N}\right) |j_1 \ldots j_{n-1}\rangle \qquad \left(A_{N/2}|j\rangle = \omega_N^j = \exp(2\pi ij/N)\right)$$
$$= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \exp\left(\frac{2\pi ikx}{N}\right) |k\rangle$$
$$= F_n|x\rangle.$$

4 Implementation of A_N

Recall the definition of A_N:

$$A_N = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & \omega_{2N} & 0 & \dots & 0 \\ \vdots & \vdots & \omega_{2N}^2 & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \omega_{2N}^{N-1} \end{bmatrix}, where\ \omega_{2N} = e^{\frac{2\pi i}{2N}}$$

For any integer $0 \leq x < 2^n$, we can write $A_N|x\rangle = \omega_{2N}^x|x\rangle$. Since x can also be expressed as the sum of its binary digits, i.e.

$$x = x_1 2^{n-1} + x_2 2^{n-2} + \cdots + x_n,$$

We have the following:

$$A_N = \omega_{2N}^{x_1 2^{n-1} + x_2 2^{n-2} + \cdots + x_n}|x_1 x_2 \dots x_n\rangle = (\omega_{2N}^{x_1 2^{n-1}}|x_1\rangle) \otimes (\omega_{2N}^{x_2 2^{n-2}}|x_2\rangle) \otimes \dots (\omega_{2N}^{x_n}|x_n\rangle)$$

We can also rewrite

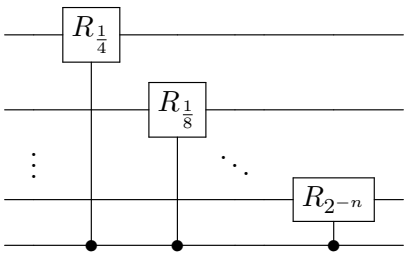
$$\omega_{2N}^{2^t} = e^{\frac{2\pi i 2^t}{2 \cdot 2^n}} = e^{\frac{2\pi i}{2 \cdot 2^{n-t}}} = \omega_{2(2^{n-t})}$$

Let $R_\theta = \begin{bmatrix} 1 & 0 \\ 0 & e^{\pi i \theta} \end{bmatrix}$. This matrix adds a phase $e^{i\pi\theta}$ if the state is $|1\rangle$, or nothing if the state is $|0\rangle$.

By substituting a series of R_θ into the previous equation, we get

$$A_N = (R_{\frac{1}{4}}|x_1\rangle) \otimes (R_{\frac{1}{8}}|x_2\rangle) \otimes \dots (R_{2^{-(n+1)}}|x_n\rangle)$$

This leads to the design of our controlled $A_{N/2}$ unitary (as used in the QFT circuit) as the following:



According to the Solovay-Kitaev Theorem, we can implement each conditional-rotation gate up to ϵ accuracy very efficiently in terms of our universal gate set.

Question: Unrolling the recursion, how many elementary gates does F_N require?

Answer: We need $O(n)$ gates for the implementation of $A_{N/2}$ (which acts on $n-1$ qubits), and then we invoke the circuit for $F_{N/2}$ (which acts on $n-1$ qubits as well). There is also $O(n)$ gates needed to implement the cyclic shift of qubits at the end. Thus, if we let $T(n)$ denote the number of gates needed to implement F_N, the total number of elementary gates required would be

$$T(n) = O(n) + T(n-1) = O(n^2).$$

5 Applications of Quantum Fourier Transform

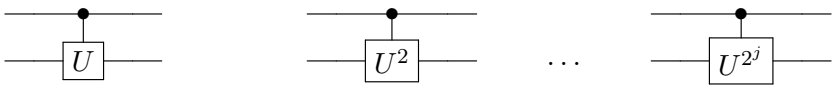
5.1 Phase Estimation

Setup: Consider a unitary operation U on n-qubits.

Fact: the eigenvalues of U are all on the unit circle: $e^{2\pi i\theta}$. Without loss of generality, we assume that $0 \leq \theta < 1$.

Suppose we are given an eigenvector $|\psi\rangle$ of U as a quantum state. It has the eigenvalue $e^{2\pi i\theta}$, but θ is not known to us, and it is what we would like to find out.

Suppose we are also given the ability to apply controlled U^{2^j} for a range of j's. In other words, we are able to run the following quantum circuits:



Using this ability, along with the eigenvector $|\psi\rangle$, we can obtain a good estimate of θ with high probability.

Assumption: Suppose that $\theta = 0.\theta_1\theta_2\ldots\theta_t$ can be represented using exactly t bits. Equivalently, $\theta = \frac{\theta_1}{2} + \frac{\theta_2}{4} + \cdots + \frac{\theta_t}{2^t}$. Then the circuit below would be able to solve the Phase Estimation problem:

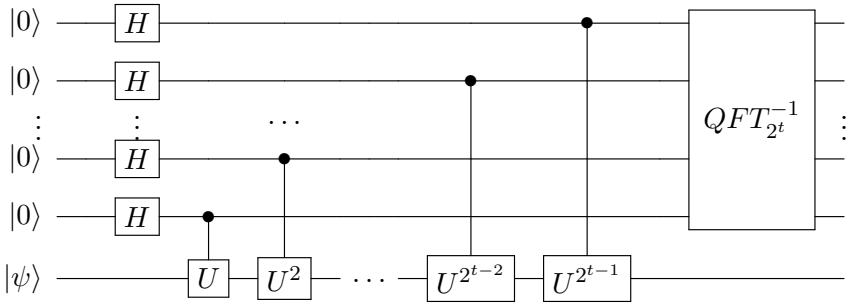


Illustration of why it works:

$$|\phi_0\rangle : |0\rangle^{\otimes t} \otimes |\psi\rangle$$

$$|\phi_1\rangle : (H|0\rangle)^{\otimes t} \otimes |\psi\rangle = \frac{1}{2^{t/2}} \underbrace{(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes \cdots \otimes (|0\rangle + |1\rangle)}_{\text{t}} \otimes |\psi\rangle$$

$$|\phi_2\rangle : \frac{1}{2^{t/2}}(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes \cdots \otimes (|0\rangle + e^{i2\pi\theta}|1\rangle) \otimes |\psi\rangle$$

$$\vdots$$

$$|\phi_{t+1}\rangle : \frac{1}{2^{t/2}}(|0\rangle + e^{i2\pi\theta 2^{t-1}}|1\rangle) \otimes (|0\rangle + e^{i2\pi\theta 2^{t-2}}|1\rangle) \otimes \cdots \otimes (|0\rangle + e^{i2\pi\theta}|1\rangle) \otimes |\psi\rangle = \frac{1}{2^{t/2}} \sum_{x \in \{0,1\}^t} e^{i2\pi\theta x}|x\rangle|\psi\rangle$$

As was mentioned above, $\theta = \frac{\theta_1}{2} + \frac{\theta_2}{4} + \cdots + \frac{\theta_t}{2^t}$.

Let $\tilde{\theta} = 2^t\theta = 2^t(\frac{\theta_1}{2} + \frac{\theta_2}{4} + \cdots + \frac{\theta_t}{2^t}) = 2^{t-1}\theta_1 + 2^{t-2}\theta_2 + \cdots + \theta_t$. Then we have

$$|\phi_{t+1}\rangle = \frac{1}{2^{t/2}} \sum_{x \in \{0,1\}^t} e^{i2\pi\tilde{\theta}\frac{x}{2^t}}|x\rangle|\psi\rangle$$

Notice that the binary expansion of $\tilde{\theta}$ is $\theta_1\theta_2 \ldots \theta_t$. Therefore, applying the inverse Quantum Fourier Transform gate $QFT_{2^t}^{-1}$ to $|\phi_{t+1}\rangle$ will give us $|\tilde{\theta}\rangle|\psi\rangle$. After this step, we have recovered θ in the output qubits.

Question: What happens when θ is not expressable as t bits?

Answer: We can show that we have got a good approximation to θ, which requires some more detailed calculations.

5.2 Order Finding

In this lecture, we won't talk about Shor's algorithm in full, as there's a fair amount of details. Instead, we will illustrate the core quantum part of Shor's algorithm, which solves the problem of Order Finding.

Factoring problem: Given $N > 0$, find a nontrivial factor $1 < p < N$ such that p divides N.

Using a few tricks from basic number theory, the Factoring problem can be reduced to the problem of

Order finding: Given $x, N > 0$ such that $x < N$ and $gcd(x, N) = 1$, find the order of x in $\mathbb{Z}_N^*$, i.e. the smallest integer r such that $x^r = 1 \mod N$.

There is no known efficient classical algorithm for this problem. However, there exists an efficient quantum algorithm using Phase Estimation.

Important Note: When we say the algorithm is "efficient", it means that the algorithm has to run in polynomial time of the input size, i.e. $poly(log(N)) = poly(n)$.

Given the input to the Order Finding problem: x, N, we will set up an instance of the Phase Estimation problem. Let r be the order of x, which is what we want to solve.

Let $2^{n-1} < N < x^n$, which means that N can be represented with n bits.

Let U be the $n - qubit$ unitary operator such that for $y \in 0, 1^n$,

$$U|y\rangle = \begin{cases} |xy \mod N\rangle & 0 \leq y < N \\ |y\rangle & otherwise \end{cases}$$

Claim: U is a unitary operator, because $y \mapsto xy \mod N$ is a one-to-one mapping (since x is invertible).

Let the eigenvector be

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} exp[\frac{-2\pi isk}{r}]|x^k \mod N\rangle \qquad where\ 0 \leq s < r.$$

$$U|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} exp[\frac{-2\pi isk}{r}]U|x^k \mod N\rangle$$

$$= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} exp[\frac{-2\pi isk}{r}]|x^{k+1} \mod N\rangle$$

$$= exp[\frac{2\pi is}{r}]\frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} exp[\frac{-2\pi isk}{r}]|x^k \mod N\rangle = exp[\frac{2\pi is}{r}]|u_s\rangle$$

Thus, the eigenvalue of $|u_s\rangle$ is $exp[\frac{2\pi is}{r}]$.

This indicates that we can use Phase Estimation to get $\frac{s}{r}$ and compute r if:

1. we can implement the controlled gates U^{2^j} efficiently;
2. we can get our hands on $|u_s\rangle$.

First, we only need $O(n)$ bits to represent $\frac{s}{r}$, so we can do Phase Estimation on this for $t \approx n$. This means that we will need to have the ability to implement controlled U^{2^n}.

$$U^{2^n}|y\rangle = |xy^{2^n} \mod N\rangle$$

This is a very high power of y, but we can do this efficiently using modular exponentiation: the idea is to compute smaller powers of y and then combine them while doing $\mod N$. For example,

$$y^8 \mod N = (y^4 \mod N)^2 \mod N = ((y^2 \mod N)^2 \mod N)^2 \mod N$$

Therefore, we can do this in $poly(n)$ time.

Regarding the access to $|u_s\rangle$, we don't know how to prepare a specific $|u_s\rangle$, but we can easily prepare a superposition of them:

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle$$

If we run the Phase Estimation algorithm and get

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle |\widetilde{s/r}\rangle$$

Here $\widetilde{s/r}$ is an approximation of s/r. If we measure the second register, we will get $\theta \approx \frac{s}{r}$ for a uniformly random s.

Question: How do we recover r since we just have the binary expansion of $\frac{s}{r}$?

Answer: We use something called the Continued Fractions algorithm. The point is, we are looking for a fraction with the smallest numerator and denominator that is close to θ. This will get us $\frac{s}{r}$ with high probability. In the end, this allows us to get r with high probability.