Notes 8.370/18.435 Fall 2022

Lecture 19 Prof. Peter Shor

In class, I Ialked a little bit about Fourier series and the discrete Fourier transform, to give some motivation for the quantum Fourier transform. In the interest of getting these lecture notes out on time, I'm not going to put these into the notes right now (I actually did the same thing last year). I may come back and revise it.

The quantum Fourier transform will be very useful in a number of quantum algorithms, which we will be covering in class. Namely, we will use them in the phase estimation, the factoring, and the discrete logarithm algorithms.

The quantum Fourier transform is very similar to the discrete Fourier transform. In fact, the discrete Fourier transform takes the amplitudes of the input to the quanutm Fourier transform to the amplitudes of the output.

The quantum Fourier transform takes input $|j\rangle$ to

$$|j\rangle \longrightarrow \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} e^{2\pi i\, jk/n} |k\rangle \; .$$

The inverse transform takes

$$|k\rangle \longrightarrow \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} e^{-2\pi i\, kj/n} |j\rangle \; .$$

In matrix form, this is

$$M_{FT} = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \dots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{n-2} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \dots & \omega^{n-3} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{n-1} & \omega^{n-2} & \omega^{n-3} & \dots & \omega \end{pmatrix}$$

where $\omega = e^{2\pi i/n}$. Note that $M^\dagger = M^{-1}$ is just M with ω replaced by ω^{-1}.

The reason the quantum Fourier transform works is that

$$\sum_{k=0}^{n-1} \omega^{k\ell} = \begin{cases} n & \text{if } \ell \text{ is a multiple of } n \\ 0 & \text{otherwise} \end{cases} . \tag{1}$$

This is not hard to prove. If ℓ is a multiple of n, all the terms in the sum are 1, and the sum is n. If ℓ is not a multiple of n, we have a geometric series, and the sum is

$$\sum_{k=0}^{n-1} \omega^{k\ell} = \frac{\left(\omega^\ell\right)^n - 1}{\omega^\ell - 1} = 0,$$

because $\omega^n = 1$. Note that Eq. (1) shows that M is unitary.

In this lecture, we will be taking $n = 2^L$, as the quantum Fourier transform is particularly easy to implement in this case. The quantum Fourier transform presented above is an L-qubit gate. For large L, experimental physicists cannot hope to build an apparatus that takes L qubits and makes them interact so as to execute an arbitrary L-qubit gate.[1] So how can we break the quantum Fourier transform into 1- and 2-qubit gates? It turns out that we only need to use two kinds of gates, the first is a Hadamard gate and the second is a controlled R_j gate, where j = 2, 3, 4, ..., L. Here $R_j = \begin{pmatrix} 1 & 0 \\ 0 & 2^{2\pi i/2^r} \end{pmatrix}$.

What we will do is represent $|j\rangle$ and $|k\rangle$ as sequences of qubits. Let $j = j_1 j_2 j_3 \dots j_{L-1}$ in binary, and let $k = k_1 k_2 k_3 \dots k_{L-1}$. Thus,

$$j = \sum_{s=0}^{L-1} \frac{1}{2^s} j_s \qquad \text{and} \qquad k = \sum_{t=0}^{L-1} \frac{1}{2^t} k_t$$

Recall that the quantum Fourier transform takes

$$|j\rangle \longrightarrow \frac{1}{\sqrt{2^L}} \sum_k e^{2\pi i j k/2^L} |k\rangle \tag{2}$$

We will use the binary decimal notation

$$0.a_1 a_2 a_3 \dots = \frac{1}{2} a_1 + \frac{1}{4} a_2 + \frac{1}{8} a_3 + \dots$$

and rewrite Eq. 2 as

$$|j_1\rangle |j_2\rangle \dots |j_L\rangle \rightarrow \frac{1}{\sqrt{2^L}} (|0\rangle + e^{2\pi i 0.j_n} |1\rangle) \dots (|0\rangle + e^{2\pi i 0.j_{n-2} j_{n-1} j_n} |1\rangle) \dots (|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n} |1\rangle)$$

There are two things to show. First, we will show how this rewriting lets us find a circuit for the QFT, and second, why we are allowed to rewrite it like this.

How can we use this to find a circuit? First, let's divide the $\frac{1}{\sqrt{2^L}}$ normalization constant equally among the terms on the righthand side. Now, let's think about the first term

$$\frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i 0.j_n} |1\rangle).$$

Here $0.j_n$ is either 0 or $1/2$, so this exponential is either 1 or -1. We thus have that if $|j_n\rangle = |0\rangle$, this term is $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and if $|j_n\rangle = |1\rangle$, this term is $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Now, we know a gate that implements this transform on $|j_n\rangle$ — the Hadamard gate. It turns out that this Hadamard gate needs to be the last thing we do in our circuit. Let's think about the second term next. What we need to implement is

$$|j_{n-1}\rangle \rightarrow (|0\rangle + e^{2\pi i 0.j_{n-1} j_n} |1\rangle)$$

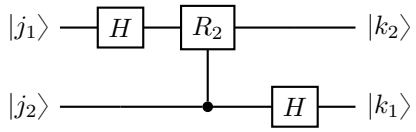
[1] Note that the same thing holds for classical computers ... we implement L-bit Boolean functions not by building special transistors for them, but by breaking them down into NOT, AND, and OR gates, so we only ever have to build 2-bit gates.

How can we implement this? The first thing we do is take

$$|j_{n-1}\rangle \to (|0\rangle + e^{2\pi i 0.j_{n-1}} |1\rangle)$$

We've nearly got it. After applying this, what we need to do is if both the $|j_n\rangle = |1\rangle$ and this qubit we've just transformed (which will end up being $|k_2\rangle$, are $|1\rangle$, then we need to multiply the phase by $e^{2\pi i/4} = i$. We can do this—it's just the gate R_2. But note that we need to hqve the qubit $|j_n\rangle$ available for this, which is why we need to process $|j_n\rangle$ last.

We've worked out how to implement the first two terms. The circuit for this is



where R_2 is the gate $\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$. Note how we need to process $|j_1\rangle$ first so that we have $|j_2\rangle$ available when we're processing $|j_1\rangle$.

Now, we can explain how to produce an arbitrary term in the above product. This is

$$|j_s\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_s j_{s+1} \ldots j_L} |1\rangle)$$

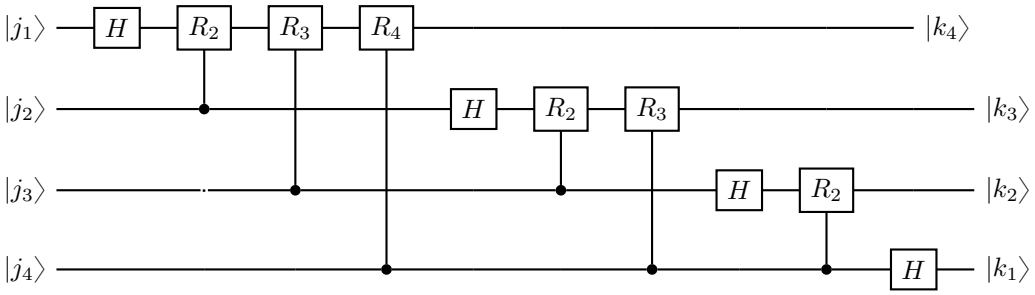
We first apply a Hadamard gate to take

$$|j_s\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_s} |1\rangle)$$

We then apply C-R_2, C-R_3, C-R_4, C-R_{L-s+1} gates between this qubit and the qubits $j_{s+1}, j_{s+2}, \ldots, j_L$ to take

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_s} |1\rangle) \to \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_s j_{s+1} \ldots j_L} |1\rangle)$$

This gives us the circuit, which we give here for $L = 4$ It should be easy enough for you to see the generalization to arbitrary L, but if you want to see it explicitly, the circuit for the general case is given in the textbook (p. 219).



Note that the output qubits come out in reverse order as the input qubits.

Note also that reversing the circuit gives exactly the same gates, (the circuit is symmetric if you flip top and bottom and right and left, because the $|k_t\rangle$ qubits appear in the opposite order as the $|j_s\rangle$ qubits), except that the complex conjugate is applied. This shows that the inverse quantum Fourier transform is the complex conjugate of the quantum Fourier transform, something we already worked out.

There is one final thing I want to say. When people first saw the quantum Fourier transform circuit, some of them raised the objection that the C-R_ℓ gate is so close to the identity that it couldn't possibly be doing anything, so there must be something wrong with the circuit. In fact, they were right about the first part of this—the C-R_ℓ gate hardly changes the state at all. What this means is that you can get a very good approximate Fourier transform by just ignoring the C-R_ℓ gates for moderately large ℓ. This lets you do approximate quantum Fourier transform with many fewer gates.