subgroup problem, a generalization of the phase estimation and order-finding problems that has among its special cases an efficient quantum algorithm for the *discrete logarithm* problem, another problem thought to be intractable on a classical computer.

5.1 The quantum Fourier transform

> *A good idea has a way of becoming simpler and solving problems other than that for which it was intended.*
> – Robert Tarjan

One of the most useful ways of solving a problem in mathematics or computer science is to *transform* it into some other problem for which a solution is known. There are a few transformations of this type which appear so often and in so many different contexts that the transformations are studied for their own sake. A great discovery of quantum computation has been that some such transformations can be computed much faster on a quantum computer than on a classical computer, a discovery which has enabled the construction of fast algorithms for quantum computers.

One such transformation is the *discrete Fourier transform*. In the usual mathematical notation, the discrete Fourier transform takes as input a vector of complex numbers, $x_0, \ldots, x_{N-1}$ where the length N of the vector is a fixed parameter. It outputs the transformed data, a vector of complex numbers $y_0, \ldots, y_{N-1}$, defined by

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k/N} . \tag{5.1}$$

The *quantum Fourier transform* is exactly the same transformation, although the conventional notation for the quantum Fourier transform is somewhat different. The quantum Fourier transform on an orthonormal basis $|0\rangle, \ldots, |N-1\rangle$ is defined to be a linear operator with the following action on the basis states,

$$|j\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k/N} |k\rangle . \tag{5.2}$$

Equivalently, the action on an arbitrary state may be written

$$\sum_{j=0}^{N-1} x_j |j\rangle \longrightarrow \sum_{k=0}^{N-1} y_k |k\rangle , \tag{5.3}$$

where the amplitudes y_k are the discrete Fourier transform of the amplitudes x_j. It is not obvious from the definition, but this transformation is a unitary transformation, and thus can be implemented as the dynamics for a quantum computer. We shall demonstrate the unitarity of the Fourier transform by constructing a manifestly unitary quantum circuit computing the Fourier transform. It is also easy to prove directly that the Fourier transform is unitary:

Exercise 5.1: Give a direct proof that the linear transformation defined by Equation (5.2) is unitary.

Exercise 5.2: Explicitly compute the Fourier transform of the n qubit state $|00\ldots0\rangle$.

In the following, we take $N = 2^n$, where n is some integer, and the basis $|0\rangle, \ldots, |2^n - 1\rangle$ is the computational basis for an n qubit quantum computer. It is helpful to write the state $|j\rangle$ using the binary representation $j = j_1 j_2 \ldots j_n$. More formally, $j = j_1 2^{n-1} + j_2 2^{n-2} + \cdots + j_n 2^0$. It is also convenient to adopt the notation $0.j_l j_{l+1} \ldots j_m$ to represent the *binary fraction* $j_l/2 + j_{l+1}/4 + \cdots + j_m/2^{m-l+1}$.

With a little algebra the quantum Fourier transform can be given the following useful *product representation*:

$$|j_1, \ldots, j_n\rangle \rightarrow \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle\right)}{2^{n/2}}. \tag{5.4}$$

This product representation is so useful that you may even wish to consider this to be the *definition* of the quantum Fourier transform. As we explain shortly this representation allows us to construct an efficient quantum circuit computing the Fourier transform, a proof that the quantum Fourier transform is unitary, and provides insight into algorithms based upon the quantum Fourier transform. As an incidental bonus we obtain the classical fast Fourier transform, in the exercises!

The equivalence of the product representation (5.4) and the definition (5.2) follows from some elementary algebra:

$$|j\rangle \rightarrow \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i j k/2^n}|k\rangle \tag{5.5}$$

$$= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i j \left(\sum_{l=1}^{n} k_l 2^{-l}\right)}|k_1 \ldots k_n\rangle \tag{5.6}$$

$$= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi i j k_l 2^{-l}}|k_l\rangle \tag{5.7}$$

$$= \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[\sum_{k_l=0}^{1} e^{2\pi i j k_l 2^{-l}}|k_l\rangle\right] \tag{5.8}$$

$$= \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[|0\rangle + e^{2\pi i j 2^{-l}}|1\rangle\right] \tag{5.9}$$

$$= \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle\right)}{2^{n/2}}. \tag{5.10}$$

The product representation (5.4) makes it easy to derive an efficient circuit for the quantum Fourier transform. Such a circuit is shown in Figure 5.1. The gate R_k denotes the unitary transformation

$$R_k \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}. \tag{5.11}$$

To see that the pictured circuit computes the quantum Fourier transform, consider what happens when the state $|j_1 \ldots j_n\rangle$ is input. Applying the Hadamard gate to the first bit produces the state

$$\frac{1}{2^{1/2}}\left(|0\rangle + e^{2\pi i 0.j_1}|1\rangle\right)|j_2 \ldots j_n\rangle, \tag{5.12}$$

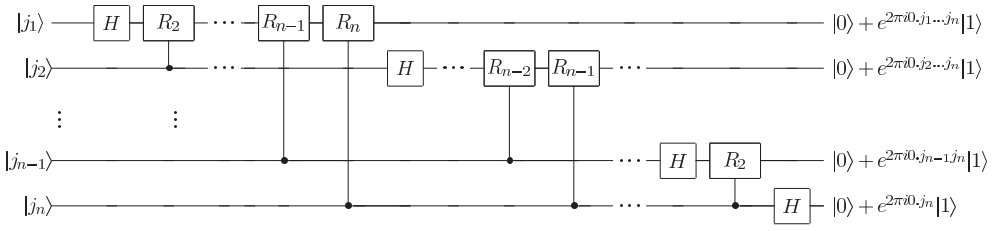


Figure 5.1. Efficient circuit for the quantum Fourier transform. This circuit is easily derived from the product representation (5.4) for the quantum Fourier transform. Not shown are swap gates at the end of the circuit which reverse the order of the qubits, or normalization factors of $1/\sqrt{2}$ in the output.

since $e^{2\pi i 0.j_1} = -1$ when $j_1 = 1$, and is +1 otherwise. Applying the controlled-R_2 gate produces the state

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2} |1\rangle \right) |j_2 \ldots j_n\rangle \, . \tag{5.13}$$

We continue applying the controlled-R_3, R_4 through R_n gates, each of which adds an extra bit to the phase of the co-efficient of the first $|1\rangle$. At the end of this procedure we have the state

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n} |1\rangle \right) |j_2 \ldots j_n\rangle \, . \tag{5.14}$$

Next, we perform a similar procedure on the second qubit. The Hadamard gate puts us in the state

$$\frac{1}{2^{2/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n} |1\rangle \right) \left(|0\rangle + e^{2\pi i 0.j_2} |1\rangle \right) |j_3 \ldots j_n\rangle \, , \tag{5.15}$$

and the controlled-R_2 through R_{n-1} gates yield the state

$$\frac{1}{2^{2/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n} |1\rangle \right) \left(|0\rangle + e^{2\pi i 0.j_2 \ldots j_n} |1\rangle \right) |j_3 \ldots j_n\rangle . \tag{5.16}$$

We continue in this fashion for each qubit, giving a final state

$$\frac{1}{2^{n/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n} |1\rangle \right) \left(|0\rangle + e^{2\pi i 0.j_2 \ldots j_n} |1\rangle \right) \ldots \left(|0\rangle + e^{2\pi i 0.j_n} |1\rangle \right) . \tag{5.17}$$

Swap operations (see Section 1.3.4 for a description of the circuit), omitted from Figure 5.1 for clarity, are then used to reverse the order of the qubits. After the swap operations, the state of the qubits is

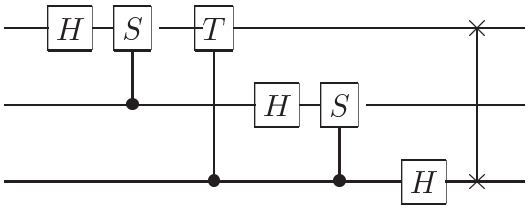
$$\frac{1}{2^{n/2}} \left(|0\rangle + e^{2\pi i 0.j_n} |1\rangle \right) \left(|0\rangle + e^{2\pi i 0.j_{n-1} j_n} |1\rangle \right) \ldots \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n} |1\rangle \right) . \tag{5.18}$$

Comparing with Equation (5.4) we see that this is the desired output from the quantum Fourier transform. This construction also proves that the quantum Fourier transform is unitary, since each gate in the circuit is unitary. An explicit example showing a circuit for the quantum Fourier transform on three qubits is given in Box 5.1.

How many gates does this circuit use? We start by doing a Hadamard gate and $n-1$ conditional rotations on the first qubit – a total of n gates. This is followed by a Hadamard gate and $n-2$ conditional rotations on the second qubit, for a total of $n + (n-1)$ gates. Continuing in this way, we see that $n + (n-1) + \cdots + 1 = n(n+1)/2$ gates are required,

Box 5.1: Three qubit quantum Fourier transform

For concreteness it may help to look at the explicit circuit for the three qubit quantum Fourier transform:



Recall that S and T are the phase and $\pi/8$ gates (see page xxiii). As a matrix the quantum Fourier transform in this instance may be written out explicitly, using $\omega = e^{2\pi i/8} = \sqrt{i}$, as

$$\frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^1 & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega^1 & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega^1 \end{bmatrix}. \tag{5.19}$$

plus the gates involved in the swaps. At most $n/2$ swaps are required, and each swap can be accomplished using three controlled-NOT gates. Therefore, this circuit provides a $\Theta(n^2)$ algorithm for performing the quantum Fourier transform.

In contrast, the best classical algorithms for computing the discrete Fourier transform on 2^n elements are algorithms such as the *Fast Fourier Transform (FFT)*, which compute the discrete Fourier transform using $\Theta(n2^n)$ gates. That is, it requires exponentially more operations to compute the Fourier transform on a classical computer than it does to implement the quantum Fourier transform on a quantum computer.

At face value this sounds terrific, since the Fourier transform is a crucial step in so many real-world data processing applications. For example, in computer speech recognition, the first step in phoneme recognition is to Fourier transform the digitized sound. Can we use the quantum Fourier transform to speed up the computation of these Fourier transforms? Unfortunately, the answer is that there is no known way to do this. The problem is that the amplitudes in a quantum computer cannot be directly accessed by measurement. Thus, there is no way of determining the Fourier transformed amplitudes of the original state. Worse still, there is in general no way to efficiently prepare the original state to be Fourier transformed. Thus, finding uses for the quantum Fourier transform is more subtle than we might have hoped. In this and the next chapter we develop several algorithms based upon a more subtle application of the quantum Fourier transform.

Exercise 5.3: (**Classical fast Fourier transform**) Suppose we wish to perform a Fourier transform of a vector containing 2^n complex numbers on a classical computer. Verify that the straightforward method for performing the Fourier transform, based upon direct evaluation of Equation (5.1) requires $\Theta(2^{2n})$ elementary arithmetic operations. Find a method for reducing this to $\Theta(n2^n)$ operations, based upon Equation (5.4).

Exercise 5.4: Give a decomposition of the controlled-R_k gate into single qubit and CNOT gates.

Exercise 5.5: Give a quantum circuit to perform the inverse quantum Fourier transform.

Exercise 5.6: (**Approximate quantum Fourier transform**) The quantum circuit construction of the quantum Fourier transform apparently requires gates of exponential precision in the number of qubits used. However, such precision is never required in any quantum circuit of polynomial size. For example, let U be the ideal quantum Fourier transform on n qubits, and V be the transform which results if the controlled-R_k gates are performed to a precision $\Delta = 1/p(n)$ for some polynomial $p(n)$. Show that the error $E(U,V) \equiv \max_{|\psi\rangle} \|(U-V)|\psi\rangle\|$ scales as $\Theta(n^2/p(n))$, and thus polynomial precision in each gate is sufficient to guarantee polynomial accuracy in the output state.

5.2 Phase estimation

The Fourier transform is the key to a general procedure known as *phase estimation*, which in turn is the key for many quantum algorithms. Suppose a unitary operator U has an eigenvector $|u\rangle$ with eigenvalue $e^{2\pi i\varphi}$, where the value of φ is unknown. The goal of the phase estimation algorithm is to estimate φ. To perform the estimation we assume that we have available *black boxes* (sometimes known as *oracles*) capable of preparing the state $|u\rangle$ and performing the controlled-U^{2^j} operation, for suitable non-negative integers j. The use of black boxes indicates that the phase estimation procedure is not a complete quantum algorithm in its own right. Rather, you should think of phase estimation as a kind of 'subroutine' or 'module' that, when combined with other subroutines, can be used to perform interesting computational tasks. In specific applications of the phase estimation procedure we shall do exactly this, describing how these black box operations are to be performed, and combining them with the phase estimation procedure to do genuinely useful tasks. For the moment, though, we will continue to imagine them as black boxes.

The quantum phase estimation procedure uses two registers. The first register contains t qubits initially in the state $|0\rangle$. How we choose t depends on two things: the number of digits of accuracy we wish to have in our estimate for φ, and with what probability we wish the phase estimation procedure to be successful. The dependence of t on these quantities emerges naturally from the following analysis.

The second register begins in the state $|u\rangle$, and contains as many qubits as is necessary to store $|u\rangle$. Phase estimation is performed in two stages. First, we apply the circuit shown in Figure 5.2. The circuit begins by applying a Hadamard transform to the first register, followed by application of controlled-U operations on the second register, with U raised