Chapter 15

Quantum Computation

It used to be supposed in Science that if everything were known about the Universe at any particular moment then we could predict what it will be through all the future... More modern science, however, has come to the conclusion that when we are dealing with atoms and electrons we are quite unable to know the exact state of them; our instruments being made of atoms and electrons themselves.

Alan M. Turing

We live in a computationally powerful universe. The laws of physics fill the world with a rich palette of objects and interactions: particles with forces between them, electromagnetic waves that can be generated and felt, energies that can be held and released. These interactions let us store, process, and transmit information, using everything from the neurons in our heads to the books in our libraries and the computers in our pockets. Indeed, if the physics of our universe could not support computation, it's doubtful that it could support life.

But the laws of physics are stranger and more wonderful than we can see in our everyday lives. In the first quarter of the 20th century, physicists learned that the fabric of our material world—its atoms, electrons and photons—behave rather differently from the objects of classical physics, such as billiard balls and planets. This new physics is counterintuitive yet strangely beautiful. It teaches us that electrons can be in two places at once, particles can spread and diffract like waves, and objects light-years apart can be inextricably entangled with each other. Can quantum physics help us compute in fundamentally new ways?

In Chapter 7 we discussed the Church–Turing Thesis, the claim that any reasonable computing device can be simulated by a Turing machine, and therefore by the programming languages and computers we know. Let's consider a sharper claim, about the devices we can actually build in the physical world:

> *Physical Church–Turing Thesis*: Any function that can be computed in finite time by a physical device can be computed by a Turing machine.

This is equivalent to the claim that, given enough time, a classical computer can simulate any physical process with any precision we desire.

Unlike the Church–Turing Thesis, which is a philosophical position about what counts as an algorithm, the Physical Church–Turing Thesis is a scientific claim about the physical universe. It may or may not be true. It is even, in principle, amenable to experiment. If you came across a physical device that solved every case of the Halting Problem you asked it about, this would be strong evidence that some physical processes are uncomputable.

Personally, we believe that the Physical Church–Turing Thesis is correct. It is conceivable that there are exotic physical systems that can somehow cram an infinite amount of computation into a finite amount of space and time, but this seems unlikely. If spacetime is continuous even at the smallest scales,
15.1 it is certainly true that describing, or simulating, a physical system exactly takes an infinite amount of information. But that doesn't mean we can access these infinitely microscopic scales and use them to build a computer.

Now let's consider a stronger claim—not just that Turing machines can compute anything a physical device can, but that they can do so in roughly the same amount of time, where "roughly the same" means within a polynomial factor:

> *Strong Physical Church–Turing Thesis*: Any function that can be computed in polynomial time by a physical device can also be calculated in polynomial time by a Turing machine.

We believe that this is false. Indeed, we believe that there are physical devices that are exponentially more efficient than a Turing machine, and therefore exponentially hard for a Turing machine to simulate.

One of the first to consider this question was the Nobel prize-winning physicist Richard Feynman. In 1981, he asked whether quantum mechanical processes can be simulated by classical computers in time proportional to the volume of space and time we wish to simulate. He argued that this is impossible, and that simulating a quantum system takes an exponential amount of resources on a classical computer. To simulate quantum systems, he said, we need quantum computers instead. But if quantum computers can simulate quantum physics exponentially faster than classical computers can, maybe they can solve other problems exponentially faster as well.

Our understanding of quantum computation is at a very early stage, but what little we know is extremely exciting. In addition to being more powerful than their classical counterparts, many quantum algorithms possess an entrancing beauty, and force us to think about computation in ways we never dreamed of before. In this chapter, we will meet algorithms that break cryptosystems by listening to the harmonics of the integers, find needles in haystacks by rotating and reflecting around them, and tell who can win a game by sending a wave through a tree.
15.2

15.1 Particles, Waves, and Amplitudes

Much of the strangeness of the quantum world can be illustrated with a classic experiment. Suppose we fire a beam of electrons at a metal plate with two slits in it, which we call A and B. On the other side of the plate is a detecting screen, which makes a dot of light at each place an electron hits it. If we cover A and let the electrons pass through B, we see a bright stripe on the screen. The same thing happens, of course, if we cover B and let the electrons go through A.

What happens if we uncover both screens? We expect to see two bright stripes, one behind each slit. After all, the brightness at each point on the screen is proportional to the fraction of electrons that arrive

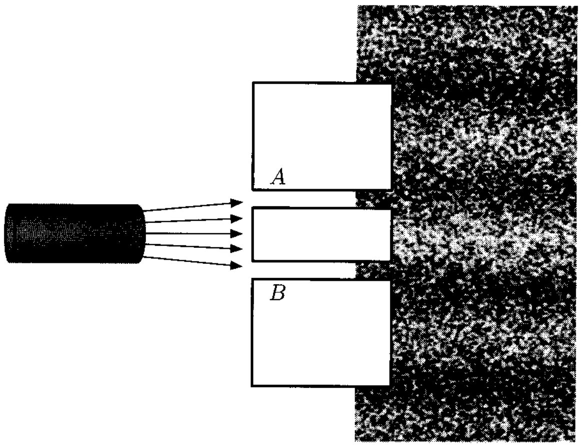


FIGURE 15.1: The two-slit experiment. We fire a beam of electrons through a pair of slits and onto a detecting screen. Instead of two bright stripes, we see an interference pattern of light and dark stripes.

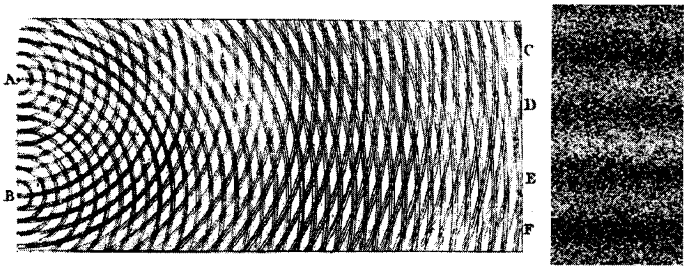


FIGURE 15.2: A sketch by Thomas Young showing waves of light or water emanating from two slits, A and B. At C, D, E, and F, these waves arrive with opposite phases, canceling out and creating dark stripes. At the points in between, they arrive with the same phase, creating bright stripes. For illustration, we align this sketch with the image from Figure 15.1.

there. For each point x, there are two ways that an electron can get there: one through A, and the other through B. Isn't the total probability that the electron arrives at x just the sum of these two probabilities?

As intuitive as this might sound, it is simply not true. Instead, we see a pattern of alternating light and dark stripes, as shown in Figure 15.1. This kind of *interference pattern* occurs in optics as well; Figure 15.2 shows a sketch by the 19th century physicist Thomas Young, who performed similar experiments with light. Waves emanate from both slits and spread out in concentric circles. If these waves are in phase with each other when they hit the screen, they form a single wave of greater intensity, like the photons in a laser beam. If they are out of phase, on the other hand, they cancel each other out. The same thing happens with ripples in a tank of water. Perhaps the electrons in the beam interact with each other, just as water molecules interact to form ripples in their surface?

The situation is stranger than that. In fact, the same pattern appears even when we do the experiment with one electron at a time. In that case, we can't explain the interference pattern in terms of electrons interacting with each other. In some sense, *each electron interacts with itself.* In the 19th century, we thought of electrons as little steel balls, with definite positions and velocities in space. But even a single electron acts to some extent like a wave, spreading out in space and interfering with itself at every point.

Moreover, this wave can't be described simply as a probability distribution. Rather than a real-valued probability, its value at each point is a complex number, called the *amplitude.* Each amplitude has both a length and an angle, or *phase,* in the complex plane. The probability associated with an amplitude a is $|a|^2$, its absolute value squared. But when we combine two paths with amplitudes a and b, we add their amplitudes before taking the absolute value, so the total probability is

$$P = |a+b|^2 .$$

If a and b have the same phase, such as at the point x in Figure 15.3, the paths interfere *constructively* and P is large. If they have opposite phases, such as at x', they interfere *destructively* and cancel. So when we add the amplitudes for the two ways the electron could get to each point on the screen, there are some where it shows up with a probability greater than their sum—and others where it never shows up at all.

15.3 This phenomenon is at the heart of quantum algorithms. Like an electron whose wave function spreads out in space, quantum computers have a kind of parallelism. They can be in a superposition of many states at once, corresponding to many possible solutions to a problem, or many possible paths through the computer's state space. By carefully designing how these states interfere, we can arrange for the wrong answers to cancel out, while creating a peak of probability at the right one.

There is one more version of the two-slit experiment that we should mention. Suppose we add a pair of detectors at the slits, which measure which slit the electron passes through. Now the interference pattern disappears: we see a pair of bright stripes, one behind each slit, just as we expected in the first place. The law of classical probability, in which the total probability of an event is the sum of the probabilities of all the ways it can happen, has been restored.

This process is called *decoherence.* As the following exercise shows, we can think of it as randomizing the phase of a quantum amplitude, and retaining information only about its absolute value.

Exercise 15.1 *Suppose that a and b are amplitudes corresponding to paths in a quantum process, such as the two paths to a given point in the two-slit experiment. Depending on their relative phase, and on whether they interfere constructively or destructively, show that their combined probability $P = |a+b|^2$ can lie anywhere in the range*

$$(|a|-|b|)^2 \le P \le (|a|+|b|)^2 .$$

Now suppose that their relative phase is random. For instance, suppose we multiply b by $\mathrm{e}^{\imath\theta}$ where θ is a random angle between 0 and 2π. Show that averaging over θ gives

$$\mathop{\mathbb{E}}_{\theta} P = \mathop{\mathbb{E}}_{\theta} \left| a + \mathrm{e}^{\imath\theta} b \right|^2 = |a|^2 + |b|^2 .$$

In other words, when the phase is random, the combined probability of the two paths is the sum of the two probabilities, just as in the classical case.

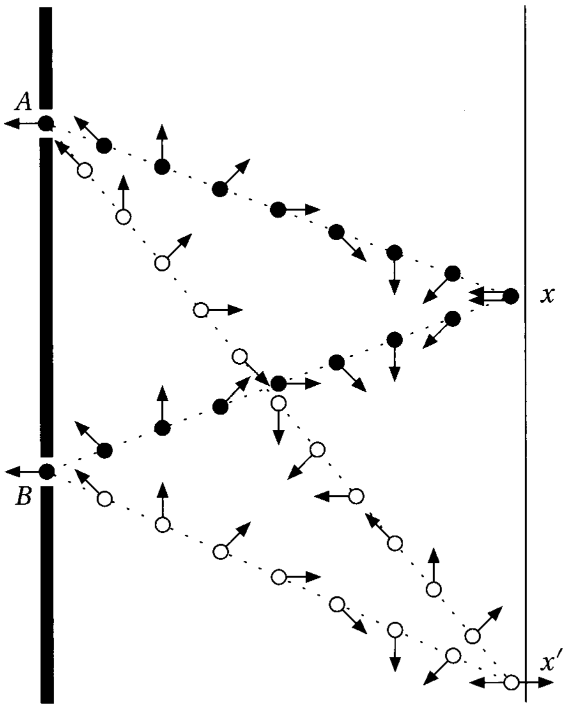


FIGURE 15.3: Constructive and destructive interference in the two-slit experiment. The phase of the electron rotates with time, so the phase of each path depends on its length. The two paths leading to x arrive with the same phase, so the electron has a large probability of hitting the screen there. The two paths leading to x' arrive with opposite phases, so they cancel and the total probability the electron hits the screen at that point is zero.

Decoherence helps explain why the world of our everyday experience looks classical rather than quantum. Quantum states are fragile. When they are allowed to interact with large, blundering objects like humans, detectors, or even stray electromagnetic waves, their phases quickly randomize, and they collapse into classical probability distributions. To build a quantum computer, we have to create a physical system where every interaction is carefully controlled—where the states within the computer interact strongly with each other, without becoming decohered by stray interactions with their environment. Clever people are performing amazing feats of physics and engineering in pursuit of this goal. This chapter is about what problems we can solve if they succeed.

15.2 States and Operators

In the classical world, we analyze our computers not in terms of physical quantities like voltages and currents, but in terms of the logical quantities—the bits—they represent. In the same way, to get from quantum physics to quantum computers, we need to consider quantum systems that can store digital information. The simplest such system is a quantum bit, or *qubit* for short. Like a classical bit, it can be true or false—but since it is quantum, it can be in a superposition of these.

In this section, we will write down the mathematical machinery for describing systems of qubits, and how their states can be transformed and measured. We will see that, mathematically speaking, quantum algorithms are not so different from classical randomized algorithms. The main difference is that they have complex-valued amplitudes instead of real-valued probabilities, allowing these amplitudes to interfere constructively or destructively, rather than simply add.

15.2.1 Back to the State Space

Let's start by viewing good-old-fashioned classical computation in a somewhat bizarre way. As we discussed in Chapter 8, a computer with m bits of memory has 2^m possible states. Let's view each of these states as a basis vector in a 2^m-dimensional vector space. Each elementary step transforms the computer's state by multiplying it by a $2^m \times 2^m$ matrix, and the entire program can be thought of as the $2^m \times 2^m$ matrix that is the product of all of its steps.

For instance, suppose we have a computer with 2 bits, x_1 and x_2. It has four possible states, which we write in the form $|x_1 x_2\rangle$. This notation $|\cdot\rangle$ is called a "ket," but it is really just a column vector. These states form a basis for a four-dimensional state space:

$$|00\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \; |01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \; |10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \; |11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}.$$

We have ordered these vectors so that x_1 is the most significant bit and x_2 is the least significant, but this is an arbitrary choice.

Now imagine a step of the program that carries out the instruction

$$x_2 := x_1 .$$

This maps old states to new states in the following way:

$$|00\rangle \to |00\rangle, \; |01\rangle \to |00\rangle, \; |10\rangle \to |11\rangle, \; |11\rangle \to |11\rangle .$$

We can think of this as multiplying the state by a 4×4 matrix,

$$U = \begin{pmatrix} 1 & 1 & & \\ 0 & 0 & & \\ & & 0 & 0 \\ & & 1 & 1 \end{pmatrix},$$

where matrix entries not shown are zero. Each column of U has a single 1, since a deterministic computation gives exactly one new state for each old state.

Now suppose we have a *randomized* algorithm, with instructions like this:

With probability $1/2$, set $x_2 := x_1$. Otherwise do nothing.

This corresponds to a stochastic matrix, i.e., one whose columns sum to 1:

$$U = \begin{pmatrix} 1 & 1/2 & & \\ 0 & 1/2 & & \\ & & 1/2 & 0 \\ & & 1/2 & 1 \end{pmatrix}.$$

Vectors in the state space now correspond to probability distributions. For instance, applying U to the state $|10\rangle$ gives the state

$$U|10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1/2 \\ 1/2 \end{pmatrix} = \frac{1}{2}(|10\rangle + |11\rangle),$$

in which the computer has an equal probability of being in the state $|10\rangle$ or the state $|11\rangle$.

Of course, if our computer is classical, it is in one of these states or the other. But for many purposes, it makes sense to think of the "state" of the computer as a probability distribution, so that it is, in some sense, in both states at once. If the computer has m bits of memory, this probability distribution is a 2^m-dimensional vector, with one component for each of the 2^m possible truth assignments.

This picture of computation may seem needlessly complicated. Does it really make sense to say that if your computer has one gigabyte of memory, then its state is a $2^{2^{33}}$-dimensional vector, and each step of your program multiplies the state by a $2^{2^{33}}$-dimensional matrix? We agree that if our goal is to understand classical computation, this point of view is rather cumbersome. But the benefit of this approach is that quantum computers—which really *can* be in many states at once—are now just a small step away.

15.2.2 Bras and Kets

She was nae get o' moorland tips,
Wi' tawted ket, an' hairy hips...

Robert Burns, *Poor Mailie's Elegy*

Rather than a real-valued vector whose components are probabilities, the state of a quantum computer is a complex-valued vector whose components are called *amplitudes*. For instance, the state of a single qubit can be written

$$|v\rangle = \begin{pmatrix} v_0 \\ v_1 \end{pmatrix} = v_0|0\rangle + v_1|1\rangle,$$

where v_0 and v_1 are complex, and where

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

are the basis vectors corresponding to the qubit being 0 (false) and 1 (true) respectively. We say that $|v\rangle$ is a *superposition* of the basis states $|0\rangle$ and $|1\rangle$ with amplitudes v_0 and v_1 respectively. Note that $|0\rangle$ is not the zero vector—it is a basis vector of length 1 in the state space.

Writing column vectors as "kets" is one half of a handy notational device invented by Paul Dirac. Here's the other half. We can also write a state as a row vector, or "bra" $\langle v|$. For a single qubit, this looks like

$$\langle v| = (v_0^*, v_1^*) = v_0^* \langle 0| + v_1^* \langle 1| .$$

The inner product of two vectors is a "bra-ket" (yes, this is what passes for humor among physicists):

$$\langle v \,|\, w\rangle = \sum_i v_i^* w_i .$$

Note that when changing a ket to a bra, we automatically take its complex conjugate, since this is essential when defining the inner product of complex-valued vectors. In particular, the inner product of a vector with itself is the square of its 2-norm, or its Euclidean length:

$$\langle v \,|\, v\rangle = \sum_i v_i^* v_i = \sum_i |v_i|^2 = \|v\|_2^2 .$$

15.2.3 *Measurements, Projections, and Collapse*

Physically, the states $|0\rangle$ and $|1\rangle$ might correspond to an electron having its magnetic field pointing north or south, or a photon being horizontally or vertically polarized, or any other pair of states that a quantum system can be in. By performing a measurement on this system, we can observe the qubit's truth value. If its state is $|v\rangle = v_0|0\rangle + v_1|1\rangle$, the probability that we observe $|0\rangle$ or $|1\rangle$ is the square of the absolute value of the corresponding amplitude,

$$P(0) = |v_0|^2 \quad \text{and} \quad P(1) = |v_1|^2 .$$

The total probability is $|v_0|^2 + |v_1|^2 = \langle v \,|\, v\rangle = 1$.

We can express these probabilities in terms of projection operators. Let Π_0 be the operator that projects onto $|0\rangle$,

$$\Pi_0 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} .$$

The probability that we observe the truth value 0 is

$$P(0) = \|\Pi_0|v\rangle\|_2^2 = \langle v|\Pi_0|v\rangle , \tag{15.1}$$

where in the second equality we used the fact that $\Pi^2 = \Pi$ for any projection operator Π. If we define Π_1 similarly as the projection operator onto $|1\rangle$, these projection operators sum to the identity,

$$\Pi_0 + \Pi_1 = \mathbb{1} ,$$

so the total probability is

$$P(0) + P(1) = \langle v|\Pi_0|v\rangle + \langle v|\Pi_1|v\rangle = \langle v|(\Pi_0 + \Pi_1)|v\rangle = \langle v \,|\, v\rangle = 1 .$$

Let's say this one more way. We can also use bra-ket notation to write *outer* products. Given two states $|v\rangle = \sum_i v_i|i\rangle$ and$|w\rangle = \sum_i w_i|i\rangle$, their outer product $|v\rangle\langle w|$ is the matrix where

$$(|v\rangle\langle w|)_{ij} = \langle i \,|\, v\rangle\langle w \,|\, j\rangle = v_i w_j^* .$$

The projection operator corresponding to observing a basis state is its outer product with itself,

$$\Pi_0 = |0\rangle\langle 0| \quad \text{and} \quad \Pi_1 = |1\rangle\langle 1| ,$$

so for instance

$$P(0) = \langle v|\Pi_0|v\rangle = \langle v\,|0\rangle\langle 0|\,v\rangle = |\langle v\,|0\rangle|^2 .$$

In the two-slit experiment, the electron is in a superposition of two states, one for each slit it can go through. When we place detectors at the slits and measure which one of the states it is in, this superposition is destroyed and the electron behaves like a tiny billiard ball that passed through one slit or the other. This is true for qubits as well. After we have observed the truth value 0, the qubit has gone from the superposition $v_0|0\rangle + v_1|1\rangle$ to the observed state $|0\rangle$. We normalize it so that the total probability is 1, giving

$$|0\rangle = \frac{\Pi_0|v\rangle}{\langle v|\Pi_0|v\rangle} .$$

This process is sometimes called the *collapse of the wave function.* Opinions differ as to whether it is a real physical process or simply a convenient shorthand for calculation. Without weighing in too heavily on this debate, we point out that it is not so different mathematically from the classical probabilistic case. If we learn something about a probability distribution, from our point of view it "collapses" to the 15.4
distribution conditioned on that knowledge. This conditional distribution is its image under a projection operator, normalized so that the total probability is 1.

What happens if we perform a partial measurement, and only measure part of the system? Suppose we have a two-qubit state, i.e., a four-dimensional vector

$$|v\rangle = \begin{pmatrix} v_{00} \\ v_{01} \\ v_{10} \\ v_{11} \end{pmatrix} = v_{00}|00\rangle + v_{01}|01\rangle + v_{10}|10\rangle + v_{11}|11\rangle .$$

If we measure just the first qubit x_1, the probability that we will observe $|1\rangle$ is the total probability of the two states in which x_1 is true, i.e., $|v_{10}|^2 + |v_{11}|^2$. As in the one-qubit case above, this observation can be described by a projection operator Π that projects onto the states that will give us this outcome. In this example, we have

$$\Pi = \begin{pmatrix} 0 & & & \\ & 0 & & \\ & & 1 & \\ & & & 1 \end{pmatrix} ,$$

and we can write the probability that the first qubit is true as

$$P(x_1 = 1) = \|\Pi|v\rangle\|_2^2 = \langle v|\Pi|v\rangle .$$

After this observation, we can think of the state as having collapsed to its image under Π,

$$|v'\rangle = \frac{\Pi|v\rangle}{\langle v|\Pi|v\rangle} ,$$

which is a superposition over $|10\rangle$ and $|11\rangle$.

How much difference do complex numbers make? Since the probability that qubit is 0 or 1 depends only on the amplitudes' absolute value, we might think that their *phases*—their angles in the complex plane—don't matter. Indeed, if two states differ by an overall phase, so that $|v'\rangle = e^{\imath\theta}|v\rangle$ for some θ, they are physically identical.

However, the *relative* phases between v's components play a crucial role. For instance, the states

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \text{ and } \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

are very different. If we measure the truth value of the qubit, we get $P(0) = P(1) = 1/2$ in either case—but as we will see below, this is not the only measurement we can perform.

15.2.4 Unitary Matrices

Each step of a classical randomized algorithm transforms the probability distribution by multiplying it by a matrix U. This matrix has to preserve the total probability, so like the transition matrices of Markov chains we discussed in Chapter 12, it must be *stochastic*. That is, it must have nonnegative entries, and each column must sum to 1.

What kind of transition matrix makes sense for a quantum computer? It again has to preserve the total probability, but now this is defined as $\|v\|_2^2 = \langle v \mid v \rangle$. What does this tell us about U? Given a matrix U, its *Hermitian conjugate* $U^\dagger = (U^T)^*$ is the complex conjugate of its transpose. If we transform the ket $|v\rangle$ to $|v'\rangle = U|v\rangle$, the bra $\langle v|$ becomes $\langle v'| = \langle v|U^\dagger$. If the total probability is preserved, we have

$$\langle v' \mid v' \rangle = \langle v|U^\dagger U|v\rangle = \langle v \mid v \rangle . \tag{15.2}$$

More generally, as Problem 15.1 shows, U preserves inner products. For all v and w,

$$\langle v' \mid w' \rangle = \langle v|U^\dagger U|w\rangle = \langle v \mid w \rangle .$$

It follows that

$$U^\dagger U = \mathbb{1} ,$$

and therefore

$$U^\dagger = U^{-1} .$$

Equivalently, the columns of U form an orthonormal basis. If $|u_i\rangle$ denotes the ith column,

$$\langle u_i \mid u_j \rangle = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j , \end{cases}$$

and similarly for the rows.

Matrices with this property are called *unitary*. In particular, they are invertible—they correspond to reversible physical processes, and can neither create nor destroy information. Geometrically, unitary matrices are a complex-valued generalization of the *orthogonal* matrices, which are real-valued rotations and reflections for which $U^T U = \mathbb{1}$. Every step of a quantum computer, no matter how complicated, is just a rotation or reflection in some high-dimensional space.

Exercise 15.2 *Show that if U is a unitary matrix, all its eigenvalues are on the unit circle in the complex plane. That is, they are of the form* $e^{\imath\theta} = \cos\theta + \imath\sin\theta$ *for some angle* θ.

Exercise 15.3 *Show that the unitary matrices U whose entries are* 0*s and* 1*s are exactly the permutation matrices—namely, those with a single* 1 *in each row and each column, so that multiplying a vector by U permutes its components. Such matrices correspond to reversible classical computation.*

Let's summarize what we have seen so far. On a purely mathematically level, we can go from classical randomized computation to quantum computation just by replacing real probabilities with complex amplitudes, 1-norms with 2-norms, and stochastic matrices with unitary ones. Of course, these changes make an enormous difference in the structure of quantum states and operations—and they lead to some very strange phenomena, which we will explore in the next section. But first, let's look at some simple unitary operators, and see how we can string them together to make quantum circuits and algorithms.

15.2.5 The Pauli and Hadamard Matrices, and Changing Basis

Three of the most important 2×2 unitary operators are the *Pauli matrices*:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -\imath \\ \imath & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

How do these act on our basis states $|0\rangle$ and $|1\rangle$? The effect of X is easy to describe—it acts like a classical NOT gate and flips the qubit's truth value.

On the other hand, Z's effect is fundamentally quantum. In a sense it leaves the truth value unchanged, but it induces a phase change of -1 if the qubit is true. That is, $|0\rangle$ and $|1\rangle$ are eigenvectors of Z with eigenvalues $+1$ and -1 respectively. Finally, $Y = \imath XZ$ switches the two truth values, and adds a phase change of $\pm\imath$ as well.

Another important one-qubit operation is the *Hadamard matrix*,

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

For instance, H transforms the states $|0\rangle$ and $|1\rangle$ to superpositions of $|0\rangle$ and $|1\rangle$, which we call $|+\rangle$ and $|-\rangle$:

$$H|0\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \text{ and } H|1\rangle = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

These are the eigenvectors of X, with eigenvalues $+1$ and -1 respectively. Since H is its own inverse, we can write

$$X = HZH \quad \text{and} \quad Z = HXH.$$

Thus H acts as a basis change, transforming Z to X and vice versa. In this basis, Z acts like a NOT gate, switching $|+\rangle$ and $|-\rangle$ just as X switches $|0\rangle$ and $|1\rangle$.

As we commented above, if we take $|+\rangle$ or $|-\rangle$ and measure the qubit's truth value, we have $P(0) = P(1) = 1/2$ in both cases. But applying H to $|+\rangle$ or $|-\rangle$ changes them back to $|0\rangle$ and $|1\rangle$. Thus these two states are as different as they possibly could be—we just lose this difference if we measure them without transforming them first.

We will refer to the states $|0\rangle, |1\rangle$ as the *computational basis*, or the Z-basis since they are the eigenvectors of Z. But this is just one of many bases we could use to describe a qubit's state space. If we want to measure a qubit $|v\rangle$ in the X-basis, for instance, along the basis vectors $|+\rangle$ and $|-\rangle$, we can do this by applying the basis change H and then measuring in the computational basis. The probability that we observe $|+\rangle$ or $|-\rangle$ is then $|\langle +|v\rangle|^2 = |\langle 0|Hv\rangle|^2$ and $|\langle -|v\rangle|^2 = |\langle 1|Hv\rangle|^2$ respectively. We can measure in any basis we like, as long as we can efficiently carry out the basis change between that basis and the computational one.

Exercise 15.4 *Show that the Pauli matrices have the following properties. First, they obey the cyclically symmetric relations* $XY = \imath Z$, $YZ = \imath X$, *and* $ZX = \imath Y$. *Secondly, they* anticommute *with each other: in other words,* $XY = -YX$, $YZ = -ZY$, *and* $ZX = -XZ$. *Finally,* $X^2 = Y^2 = Z^2 = \mathbb{1}$.

Exercise 15.5 *Write the projection operators* Π_+ *and* Π_- *that project onto* X*'s eigenvectors.*

15.2.6 Multi-Qubit States and Tensor Products

How do we define states with multiple qubits? If $|u\rangle, |v\rangle$ are vectors of dimension n and m, their *tensor product* $|u\rangle \otimes |v\rangle$ is an (nm)-dimensional vector whose components are the products of those of $|u\rangle$ and $|v\rangle$. For instance, the tensor product of two one-qubit states looks like

$$\begin{pmatrix} u_0 \\ u_1 \end{pmatrix} \otimes \begin{pmatrix} v_0 \\ v_1 \end{pmatrix} = \begin{pmatrix} u_0 v_0 \\ u_0 v_1 \\ u_1 v_0 \\ u_1 v_1 \end{pmatrix}.$$

In particular, our two-qubit basis vectors are tensor products of one-qubit ones. For instance,

$$|01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |0\rangle \otimes |1\rangle.$$

If $|u\rangle$ and $|v\rangle$ are independent probability distributions then $|u\rangle \otimes |v\rangle$ is the joint distribution, where the probability of each combination of values is the product of their probabilities. Similarly, if $|u\rangle$ and $|v\rangle$ are two qubits, the amplitudes in their joint state $|u\rangle \otimes |v\rangle$ are the products of their amplitudes.

We will see, however, that most states with two or more qubits can't be written as tensor products. Quantum states can be *entangled*, so that their qubits are not independent of each other. This is analogous to the fact that a joint probability distribution can have correlations, so that it is not a product of independent distributions. However, quantum entanglement turns out to be much weirder than classical correlations can be.

We can also define the tensor product of two operators. If we have two qubits x_1, x_2 and we apply two operators A and B to x_1 and x_2 respectively, the resulting operator is their tensor product $A \otimes B$. We can write this as a 4×4 matrix, in which each 2×2 block consists of a copy of B multiplied by an entry of A:

$$A \otimes B = \left(\begin{array}{c|c} A_{11} B & A_{12} B \\ \hline A_{21} B & A_{22} B \end{array} \right) = \begin{pmatrix} A_{11} B_{11} & A_{11} B_{12} & A_{12} B_{11} & A_{12} B_{12} \\ A_{11} B_{21} & A_{11} B_{22} & A_{12} B_{12} & A_{12} B_{22} \\ A_{21} B_{11} & A_{21} B_{12} & A_{22} B_{11} & A_{22} B_{12} \\ A_{21} B_{21} & A_{21} B_{22} & A_{22} B_{12} & A_{22} B_{22} \end{pmatrix},$$

where we have arbitrarily ordered our basis so that x_1 is the most significant qubit and x_2 is the least significant. For instance, the tensor product of the Hadamard matrix with itself is

$$H \otimes H = \frac{1}{\sqrt{2}} \left(\begin{array}{c|c} H & H \\ \hline H & -H \end{array} \right) = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} .$$

Exercise 15.6 *Show that*

$$(A \otimes B)(|u\rangle \otimes |v\rangle) = A|u\rangle \otimes B|v\rangle$$

and

$$(\langle u| \otimes \langle v|)(|u'\rangle \otimes |v'\rangle) = \langle u \,|\, u'\rangle \langle v \,|\, v'\rangle .$$

Exercise 15.7 *Show that the two-qubit state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ is entangled. That is, it is not the tensor product of any pair of one-qubit states.*

15.2.7 Quantum Circuits

When we discuss classical computation, we don't use elementary logical gates. Thanks to the hard work of our 20th-century forebears, we can describe classical algorithms at the software level—using high-level pseudocode—rather than at the hardware level. For quantum computing, however, we are at a very early stage of understanding what higher-level programming constructs we can use. So, while doubtless our descendants will pity us—just as we pity our grandparents who programmed with punch cards and wires—we still often describe quantum algorithms in terms of circuits, where each gate consists of an elementary quantum operation.

Let's introduce a notation for quantum circuits. We draw a one-qubit operator U like this:

$$-\boxed{U}- .$$

This gate takes a one-qubit input $|v\rangle$, entering through the "wire" on the left, and transforms it to $U|v\rangle$. We compose two such gates by stringing them together,

$$-\boxed{U_1}-\boxed{U_2}- \;=\; -\boxed{U_2 U_1}- .$$

If we have two qubits, applying two operators in parallel gives their tensor product,

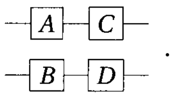
$$\begin{array}{c} -\boxed{A}- \\ -\boxed{B}- \end{array} = A \otimes B .$$

A major benefit of these circuit diagrams is that certain algebraic identities become obvious, as the following exercise shows:

Exercise 15.8 *If A, B, C, D are matrices of the same size, show that*

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD) .$$

Hint: consider the diagram



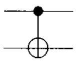
15.5 To get any computation off the ground, our qubits need to interact with each other. One of the simplest and most useful quantum gates is the "controlled-NOT," which originated in reversible computation. It takes two qubits $|a\rangle$ and $|b\rangle$ as its inputs, and flips the truth value of $|b\rangle$ if $|a\rangle$ is true:

$$|a,b\rangle \to |a, b \oplus a\rangle$$

where $\oplus$ denotes exclusive OR, or equivalently addition mod 2. As a matrix, this is

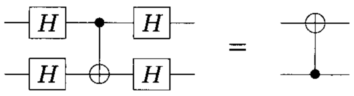
$$\begin{pmatrix} 1 & 0 & & \\ 0 & 1 & & \\ & & 0 & 1 \\ & & 1 & 0 \end{pmatrix} = \left(\begin{array}{c|c} \mathbb{1} & \\ \hline & X \end{array}\right).$$

Diagrammatically, we represent a controlled-NOT like this:

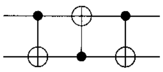


The top qubit is the "control" a, and the bottom qubit is the "target" b. But the next exercise shows that these names are naive. There is no such thing as a one-way flow of information in quantum mechanics.

Exercise 15.9 *Show that if we transform to the X-basis, a controlled-NOT gate goes the other way:*

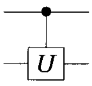


Exercise 15.10 *Show that the circuit*



switches the two qubits.

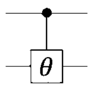
More generally, given any one-qubit operator U, we can define a *controlled-U* gate that applies U to the target qubit if the control qubit is true. We draw this as



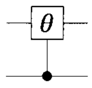
$$= \left(\begin{array}{c|c} \mathbb{1} & \\ \hline & U \end{array}\right).$$

Thus we could also call the controlled-NOT gate the controlled-X gate.

Another useful gate is the *controlled phase shift*, which induces a phase shift if both qubits are true and otherwise leaves the state unchanged. This gate is symmetric with respect to the two qubits, and we write



$=$



$$= \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & e^{\imath\theta} \end{pmatrix}.$$

We can write the entries of this matrix as $e^{\imath x_1 x_2 \theta}$ where $x_1, x_2 \in \{0,1\}$. In particular, setting $\theta = \pi$ gives a controlled-Z gate.

Classically, a handful of basic logical operations suffice to carry out any computation we desire. In particular, if we have AND, OR, and NOT gates—or, for that matter, just NANDs or NORs—we can build a circuit for any Boolean function. What unitary operators do we need to carry out universal quantum computation, and implement any unitary operator?

Since quantum operators live in a continuous space, we say that a set S of quantum gates is *universal* if, for any 2^n-dimensional unitary matrix U and any ε, there is a circuit on n qubits made of these gates that approximates U to within an error ε. One such set consists of the controlled-NOT, the Hadamard, and the one-qubit gate 15.6

$$Z^{1/4} = \begin{pmatrix} 1 & 0 \\ 0 & e^{\imath\pi/4} \end{pmatrix}.$$

In general, the number of gates in our circuit has to increase as ε decreases, just as we need larger denominators to find better rational approximations to a real number. But even among families of matrices U that can be carried out exactly by a finite circuit, some require more gates than others. That's where computational complexity comes in.

15.3 Spooky Action at a Distance

(secret, secret, close the doors!) we always have had a great deal of difficulty in understanding the world view that quantum mechanics represents. At least I do, because I'm an old enough man that I haven't got to the point that this stuff is obvious to me. Okay, I still get nervous with it.

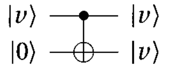
Richard Feynman [277]

One of the strangest aspects of quantum mechanics is *entanglement*, in which the states of two distant quantum systems are inextricably linked. While parts of a classical system can be correlated with each other, quantum entanglement is a much more curious phenomenon, and has led to some of the deepest debates about the nature of physical reality.

In this section we describe how quantum gates like the controlled-NOT can create entangled pairs of qubits, how measuring these qubits can produce paradoxical results, and how entanglement can—or can't—be used to communicate across vast distances. These phenomena have more to do with quantum information or communication than computation per se, but they help reveal the counterintuitive behavior of quantum systems.

15.3.1 Cloning and Entanglement

Suppose that we apply a controlled-NOT to a state in which the target qubit is $|0\rangle$. Since we flip the target qubit if the control qubit is $|1\rangle$, it seems that the effect is to produce two copies of the control qubit:



This is certainly true if $|v\rangle = |0\rangle$ or $|1\rangle$. But what if $|v\rangle$ is a superposition of these two? If each qubit is independently in the state $|v\rangle = v_0|0\rangle + v_1|1\rangle$, their joint state would be the tensor product

$$|v\rangle \otimes |v\rangle = v_0^2|00\rangle + v_0 v_1|01\rangle + v_1 v_0|10\rangle + v_1^2|11\rangle .$$

But since the amplitudes of $|v\rangle \otimes |v\rangle$ are quadratic functions of those of $|v\rangle$, no linear operator—let alone a unitary one—can transform $|v\rangle$ to $|v\rangle \otimes |v\rangle$. That is, there is no matrix M such that, for all $|v\rangle$,

$$M\big(|v\rangle \otimes |0\rangle\big) = M \begin{pmatrix} v_0 \\ 0 \\ v_1 \\ 0 \end{pmatrix} = \begin{pmatrix} v_0^2 \\ v_0 v_1 \\ v_1 v_0 \\ v_1^2 \end{pmatrix} .$$

This result is called the *No-Cloning Theorem.*

15.7 This inability to copy quantum information seems rather strange at first. In classical computation, we take it for granted that we can copy one bit to another. For instance, we routinely feed the output of one logical gate into the inputs of multiple gates, simply by splitting a wire into several branches.

However, a kind of No-Cloning Theorem also holds for classical randomized algorithms. If x is a bit in a randomized algorithm, we can't create another bit y which has the same distribution as x *but is independent of* x, since their joint probabilities $P(x,y) = P(x)P(y)$ would be quadratic functions of x's probabilities. We can clone truth values, but not probability distributions of truth values.

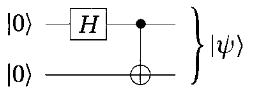
If it doesn't copy a qubit, what *does* the controlled-NOT do in this case? Rather than $|v\rangle \otimes |v\rangle$, the state it generates is

$$v_0|00\rangle + v_1|11\rangle = \begin{pmatrix} v_0 \\ 0 \\ 0 \\ v_1 \end{pmatrix} .$$

These two qubits always have the same truth value, as opposed to $|v\rangle \otimes |v\rangle$ where all four combinations are possible. Therefore, if we measure the truth value of one qubit, we also learn the truth value of the other one. To focus on a specific example, consider the state

$$|\psi\rangle = \frac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big) . \tag{15.3}$$

We could generate $|\psi\rangle$ from an initial state of $|00\rangle$ using the following circuit:



As Exercise 15.7 showed, this two-qubit state is *entangled.* That is, there is no way to write it as the tensor product of two one-qubit states. This entanglement is the source of some very surprising phenomena. To explore them, let's welcome two famous characters onto the stage: Alice and Bob.

15.3.2 Alice, Bob, and Bell's Inequalities

How wonderful that we have met with a paradox.
Now we have some hope of making progress!
Niels Bohr

Suppose two qubits are in the entangled state $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. We give one of these qubits to Alice and the other to Bob, and they fly off in opposite directions until they are a light-year apart. If Bob measures his qubit, he immediately knows the value of Alice's qubit as well. To put it differently, his measurement collapses not just his qubit, but also Alice's, no matter how far apart they are. Has some sort of physical effect flown instantaneously across the distance between them?

At first glance, there's nothing paradoxical going on here. We create long-range correlations in the classical world all the time. For instance, we could write two copies of a letter, seal them in envelopes, and give one each to Alice and Bob. If Bob opens his envelope, he immediately knows what's written on Alice's letter as well, even if it is a light-year away. Bob hasn't affected Alice's letter physically—he has simply learned something about it.

A priori, we might expect quantum states to be like unopened letters, which appear to be in a superposition just because we don't know their contents yet. If that were true, the only thing that collapses when we measure quantum states would be our uncertainty about them. The entanglement between Alice's and Bob's qubits would simply be a form of correlation between their underlying states.

In this section, we will see a clever proof that this is not the case. There are experiments that Alice and Bob can perform that yield correlations stronger than any pair of classical states, *no matter how they are correlated*, could produce. There is simply no way to think of each qubit as having an underlying state that determines the outcome of the measurements that Alice or Bob could perform. They are inextricably entangled in a way that classical systems cannot be.

These results call into question some of our most cherished assumptions about the physical world. To understand them, we need to delve a little deeper into the physics and mathematics of measurement.

In quantum mechanics, each physical quantity or *observable*—such as energy, momentum, or angular momentum—is associated with a linear operator M. This operator is *Hermitian*, i.e., $M^\dagger = M$. As the following exercise shows, this means that its eigenvalues are real and that its eigenvectors form a basis.

Exercise 15.11 *Suppose that $M^\dagger = M$. Show that M's eigenvalues are real, and that any pair of eigenvectors $|v\rangle, |w\rangle$ with different eigenvalues are orthogonal, i.e., that $\langle v \mid w\rangle = 0$. Hint: consider the product $\langle v|M|u\rangle$.*

If we measure a state v in this basis, we observe one of M's eigenvalues λ according to the probability distribution

$$P(\lambda) = \langle v|\Pi_\lambda|v\rangle ,$$

where Π_λ projects onto the subspace spanned by the eigenvector(s) with eigenvalue λ. In particular, since

$$M = \sum_\lambda \lambda \Pi_\lambda ,$$

the expectation of λ is

$$\mathbb{E}[\lambda] = \sum_\lambda \lambda P(\lambda) = \langle v|M|v\rangle . \tag{15.4}$$

Since this is the expected value of the observable corresponding to M, we will call it $\mathbb{E}[M]$ below.

Now imagine that Alice and Bob's qubits each consists of a particle such as a photon. We can think of the Z operator as measuring the photon's spin, or angular momentum, around the z axis. Its eigenvalue is $+1$ or -1 depending on whether this spin is counterclockwise or clockwise. Similarly, the X operator measures the spin around the x-axis.

Exercise 15.12 *Check that the Pauli matrices X, Y, and Z are Hermitian as well as unitary. What does this mean about their eigenvalues?*

Classically, angular momentum is a vector, and we can measure its components along both the z-axis and the x-axis. But two operators have the same eigenvectors if and only if they commute. Since $XZ = -ZX$, these two operators have no eigenvectors in common. We can't measure two noncommuting observables simultaneously—there is simply no state for which both observables are well-defined. This is best known in the form of *Heisenberg's uncertainty principle*, where we can't measure both a particle's
15.8 position and its momentum.

On the other hand, if Alice measures her qubit in one basis, and Bob measures his qubit in another, the two corresponding operators commute. For instance, Alice can measure her qubit in the Z-basis or the X-basis by applying one of these operators:

$$Z_A = Z \otimes \mathbb{1} \text{ or } X_A = X \otimes \mathbb{1} .$$

Each of these applies Z or X to Alice's qubit—the one on the left of the tensor product—while leaving Bob's unchanged. Similarly, Bob can measure his qubit in either the H-basis or the H'-basis, applying

$$H_B = \mathbb{1} \otimes H \text{ or } H'_B = \mathbb{1} \otimes H' ,$$

where

$$H' = XHX = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix} .$$

Geometrically, H and H' measure the spin around axes that are an angle $\pi/4$ away from the z-axis and x-axis as shown in Figure 15.4.

While we can't measure Z_A and X_A simultaneously, we can certainly measure, say, Z_A and H_B simultaneously, since they commute: $Z_A H_B = H_B Z_A = Z \otimes H$. The same is true of X_A and H'_B, and so on. If Alice and Bob send us their results, we can check to see how often they agree, and how correlated their measurements are.

As Problems 15.7 and 15.8 show, if the bases that Alice and Bob use are an angle θ apart, the correlation between their measurements is $\cos\theta$. Since $\theta = \pi/4$ for each pair of bases except Z_A and H'_P, for which $\theta = 3\pi/4$, we have

$$\mathbb{E}[Z_A H_B] = \mathbb{E}[X_A H_B] = \mathbb{E}[X_A H'_B] = -\mathbb{E}[Z_A H'_B] = \frac{1}{\sqrt{2}} .$$

But something funny is going on. Consider the following quantity:

$$W = Z_A H_B + X_A H_B + X_A H'_B - Z_A H'_B . \tag{15.5}$$

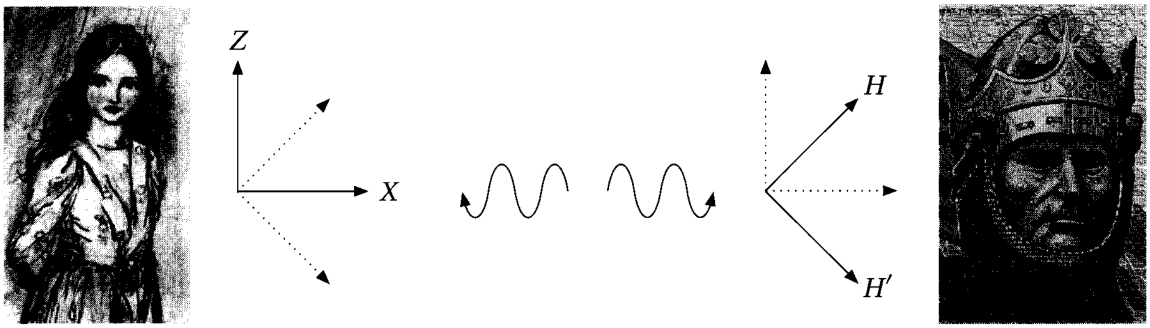


FIGURE 15.4: Alice and Bob. Alice can measure her qubit in either the Z-basis or the X-basis, and Bob can measure his in the H-basis or the H'-basis. The resulting measurements are more correlated than any classical probability distribution, correlated or not, can account for.

Since the expectation of the sum is the sum of the expectations, we have

$$\mathbb{E}[W] = \frac{4}{\sqrt{2}} = 2\sqrt{2}.$$

On the other hand, we can factor W as follows,

$$W = Z_A\left(H_B - H'_B\right) + X_A\left(H_B + H'_B\right). \tag{15.6}$$

Since each of the four operators appearing in W yields $+1$ or -1, there are two possibilities. If $H'_B = H_B$ then $W = 2X_AH_B$, and if $H'_B = -H_B$ then $W = 2Z_AH_B$. In either case we have

$$W \in \{\pm 2\}.$$

Certainly the expectation of W has to lie between -2 and $+2$, and so

$$|\mathbb{E}[W]| \leq 2. \tag{15.7}$$

This is one form of *Bell's inequality*. It holds for any set of four random variables Z_A, X_A, H_B, H'_B that take values in $\{\pm 1\}$, and which are chosen according to any joint probability distribution, with arbitrary correlations between them. Yet it is flatly contradicted by (15.5). How can this be?

When we factored W as in (15.6), we assumed that the Z_A appearing in Z_AH_B, for instance, is the same as the Z_A appearing in $Z_AH'_B$. That is, we assumed that whether or not Alice chooses to measure her photon along the z-axis, there is some value Z_A that *she would observe if she did*, and that this value is not affected by Bob's choice of basis a light-year away.

Physically, we are assuming that the universe is *realistic*, i.e., that the outcome of Alice's experiments has a probability distribution that depends on the state of her photon, and that it is *local*, i.e., that influences cannot travel faster than light. The fact that Bell's inequality is violated—and this violation has been shown very convincingly in the laboratory—tells us that one or both of these assumptions is false.

Einstein called this phenomenon "spukhafte Fernwirkung," or as it is often translated, "spooky action at a distance." He found it profoundly disturbing, and he's not the only one. It seems that we must either accept that physics is nonlocal, which violates special relativity, or that the outcome of a measurement is not a consequence of a system's state. Opinions differ on the best way to resolve this paradox. For now, let's take for granted that Bob really can affect Alice's qubit by collapsing it from afar. Can he use this effect to send Alice a message? 15.9

15.3.3 Faster-Than-Light Communication, Mixed States, and Density Matrices

If Bob's choices have an instantaneous effect on Alice's qubit, no matter how far away she is, it stands to reason that they could use this effect to communicate faster than light. Of course, given special relativity, this means that they can also communicate backwards in time, which would be even more useful. Let's see if we can make this work.

Suppose, once again, that Alice and Bob share an entangled pair of qubits, $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. If Bob measures his qubit in the Z-basis, observing $|0\rangle$ or $|1\rangle$, then Alice's qubit will be in the same state. The following exercise shows that $|\psi\rangle$ is identical to the entangled state we would get if we used $|+\rangle$ and $|-\rangle$ instead of $|0\rangle$ and $|1\rangle$:

Exercise 15.13 *Show that*

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(|{+}{+}\rangle + |{-}{-}\rangle\right) = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right).$$

Show this both by direct calculation and by showing that

$$(H \otimes H)|\psi\rangle = |\psi\rangle .$$

Hint: start with a circuit that generates $|\psi\rangle$, and use Exercise 15.9.

Therefore, if Bob measures in the X-basis, observing $|+\rangle$ or $|-\rangle$ with equal probability, Alice's qubit will again be in the same state as Bob's.

Bob can't control the outcome of his measurement, but he can control which basis he measures in. So he attempts to communicate a bit of information to Alice in the following way. If he wants to send a 0, he measures his qubit in the Z-basis, and if he wants to send a 1 he measures in the X-basis. In the first case, the state of Alice's qubit is $|v\rangle = |0\rangle$ or $|1\rangle$, each with probability 1/2. In the second case, it is $|v\rangle = |+\rangle$ or $|-\rangle$, again with equal probability. The question is whether Alice can distinguish between these two scenarios. Is there is any measurement she can perform where the outcomes have different probability distributions depending on which basis Bob used?

As we discussed in Section 15.2.6, if Alice measures her qubit, the probability that she observes a given outcome is $P = \langle v|\Pi|v\rangle$ for some projection operator Π. If Bob measured in the Z-basis, the probability she observes this outcome is the average of P over $|v\rangle = |0\rangle$ and $|1\rangle$,

$$P = \frac{1}{2}\left(\langle 0|\Pi|0\rangle + \langle 1|\Pi|1\rangle\right),$$

while if he uses the X-basis the average probability is

$$P = \frac{1}{2}\left(\langle +|\Pi|+\rangle + \langle -|\Pi|-\rangle\right).$$

But these two expressions are exactly the same. The first sums over the diagonal entries of Π in the Z-basis, and the second sums over the diagonal entries in the X-basis. The trace of a matrix is basis-independent, so in both cases we have

$$P = \frac{1}{2}\,\mathrm{tr}\,\Pi .$$

In other words, no matter what measurement Alice performs, she sees the same probability distribution of outcomes regardless of which basis Bob measured in. His choice of basis has precisely zero effect on anything Alice can observe—just as if physics were local after all.

It's important to note that the "state" of Alice's qubit in, say, the first case is not a quantum superposition of $|0\rangle$ and $|1\rangle$. It is a *probabilistic mixture* of these two quantum states, where each state has a real probability rather than a complex amplitude. How can we represent such a mixture mathematically?

Let's start with a *pure state*, i.e., a quantum state represented by a complex vector $|v\rangle$. If we define its *density matrix* ρ as the outer product of $|v\rangle$ with itself,

$$\rho = |v\rangle\langle v| ,$$

then we can write the probability of an observation associated with a projection operator Π as

$$P = \langle v|\Pi|v\rangle = \mathrm{tr}\,\Pi\rho .$$

Now suppose we have a *mixed state*, i.e., a probability distribution over pure states $|v\rangle$ where each one appears with probability $P(v)$. If we define its density matrix as

$$\rho = \sum_v P(v)\,|v\rangle\langle v| ,$$

then the average probability of this observation can again be written as $\mathrm{tr}\,\Pi\rho$ since

$$P = \sum_v P(v)\,\langle v|\Pi|v\rangle = \mathrm{tr}\,\Pi\rho .$$

Exercise 15.14 *Show that the density matrix ρ of any mixed state obeys $\mathrm{tr}\,\rho = 1$. Show also that ρ is a projection operator, i.e., that $\rho^2 = \rho$, if and only if it is a pure state $\rho = |v\rangle\langle v|$ for some $|v\rangle$.*

Now let's calculate the density matrix of Alice's qubit in the two scenarios. It is the same regardless of which basis Bob measures in, since

$$\rho = \frac{1}{2}\big(|0\rangle\langle 0| + |1\rangle\langle 1|\big) = \frac{1}{2}\big(|+\rangle\langle +| + |-\rangle\langle -|\big) = \frac{1}{2}\mathbb{1} .$$

This is called the *completely mixed state*. Since the identity matrix $\mathbb{1}$ is the same in every basis, ρ is a uniform probability distribution over all basis states, no matter which state we measure it in. Again, Bob's choice of basis has no effect on the probabilities of Alice's measurements.

We have shown that Alice and Bob can't use "spooky action at a distance" to communicate. This suggests that it might not be so spooky after all—or rather, that it might not really be an action. To an empiricist, an effect is not real unless it can be measured, so there is really no sense in which Bob can affect Alice's qubit. However, the fact that Bell's inequality is violated forces us to admit that something more than classical correlation is going on. We leave the reader to ponder this, and move on to some practical uses for entanglement.

15.3.4 Using Entanglement to Communicate

Even though entanglement doesn't let us communicate faster than light, it is still a useful resource for quantum communication. Consider the following technique, called *superdense coding*.

Before they leave for their respective journeys, Alice and Bob buy a supply of entangled pairs of qubits at their local space station, each of which is in the state $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. They each take one qubit from each pair, shake hands, and then fly light-years apart.

Later on, Alice wants to send Bob two classical bits about her travels. Let's call these bits a and b. She takes her half of an entangled pair, applies Z if a is true, and then applies X if b is true. Let's call the resulting two-qubit state $|\psi_{a,b}\rangle$. This yields one of four possible states, where Alice's qubit is on the left:

$$\begin{aligned}
|\psi_{0,0}\rangle &= |\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\
|\psi_{0,1}\rangle &= X_A|\psi\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle) \\
|\psi_{1,0}\rangle &= Z_A|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \\
|\psi_{1,1}\rangle &= X_A Z_A|\psi\rangle = \frac{1}{\sqrt{2}}(|10\rangle - |01\rangle) .
\end{aligned}$$

Alice then sends her qubit to Bob, so that he possesses the entire two-qubit state.

The four states $|\psi_{a,b}\rangle$ are called the *Bell basis*. They are orthogonal, so Bob can learn a and b by performing an appropriate measurement. Specifically, by applying a unitary operator U, Bob can change the Bell basis $|\psi_{a,b}\rangle$ to the computational basis $|a,b\rangle$, and read a and b from the qubit's truth values. We ask you to design a quantum circuit for U in Problem 15.13. Thus this protocol lets Alice send Bob two bits at a time, or vice versa.

How much does this cost? Each time Alice and Bob do this, they use up one of their entangled pairs. Let's say that they use one bit of entanglement, or one "ebit." In addition, they actually have to send a qubit across the space between them. If we think of ebits, qubits, and classical bits as resources that we can use for communication, this gives us the following inequality:

$$1 \text{ qubit} + 1 \text{ ebit} \geq 2 \text{ bits} .$$

If an entangled pair allows us to encode two classical bits into one qubit, can it do the reverse? Indeed it can. Suppose that in addition to her half of the entangled pair, Alice has a qubit in the state $|v\rangle$. This gives us a three-qubit state

$$|\Psi\rangle = |v\rangle \otimes |\psi\rangle = |v\rangle \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) ,$$

where Alice holds the first and second qubit and Bob holds the third. A little calculation shows that this state can be rewritten in the following form:

$$\begin{aligned}
|\Psi\rangle = \frac{1}{2}\big(&|\psi_{0,0}\rangle \otimes |v\rangle \\
&+|\psi_{0,1}\rangle \otimes X|v\rangle \\
&+|\psi_{1,0}\rangle \otimes Z|v\rangle \\
&+|\psi_{1,1}\rangle \otimes ZX|v\rangle\big) ,
\end{aligned}$$

Here Alice holds $|\psi_{a,b}\rangle$ and Bob holds $|v\rangle$, $X|v\rangle$, $Z|v\rangle$, or $ZX|v\rangle$.

Exercise 15.15 *Confirm that* $|\Psi\rangle$ *can be written this way.*

Alice measures her two qubits in the Bell basis $|\psi_{a,b}\rangle$, thus collapsing $|\Psi\rangle$ to one of these four states. Having learned a and b, she transmits those two classical bits to Bob. He then applies X to his qubit if b is true and Z to his qubit if a is true, in that order. This *disentangles* his qubit from Alice's, and puts it in the state $|v\rangle$.

Note that the qubit $|v\rangle$ has not been cloned. Bob ends up with $|v\rangle$, but Alice's copy of $|v\rangle$ has been measured and hence destroyed. We can say that $|v\rangle$ has been "teleported" from Alice to Bob. In theory, we could do this even with quantum states composed of many particles. But before you jump into the quantum teleporter, keep in mind that it destroys the original.

Quantum teleportation is especially astonishing because Alice manages to send a quantum state to Bob *without knowing it herself.* If Alice tried to measure her qubit in order to learn its amplitudes, her first measurement would collapse it to one basis state or the other—and since qubits can't be cloned, she can only measure it once. In any case, no finite amount of classical information can completely convey a qubit, since the real and imaginary parts of its amplitudes have an infinite number of digits. Instead, Alice sends her qubit "through" the entangled pair, giving Bob two classical bits as directions for how to extract it. The fact that we can do this gives us another inequality,

$$1 \text{ ebit} + 2 \text{ bits} \geq 1 \text{ qubit}.$$

Since superdense coding and quantum teleportation were discovered in the early 1990s, a great deal more has been learned about the capacity of different types of quantum communication. For one especially charming result, we refer the reader to Problems 15.14 and 15.15.

But enough about communication. It's time to delve into computation, and show some examples of 15.10
what quantum algorithms can do.

15.4 Algorithmic Interference

Quantum systems possess a fascinating kind of parallelism—the ability of an electron to pass through two slits at once, or of a computer's qubits to be in a superposition of many states. Since unitary operators are linear, they act on every component of the state simultaneously. In some sense, this lets a quantum algorithm test many possible solutions to a problem at once.

However, as we will see, parallelism alone doesn't account for the power of quantum computation. Even classical randomized algorithms can explore many solutions at once in a probabilistic sense. What makes quantum computation unique is the combination of parallelism with *interference*: the fact that complex amplitudes can combine in phase or out of phase. By applying the right unitary transformation, we can use constructive interference to make the right answers more likely, while using destructive interference to cancel out the wrong ones.

In this section, we will show how interference lets a quantum computer learn some things about a function by asking fewer questions than a classical computer would need. This will culminate in Simon's problem, which quantum computers can solve exponentially faster than classical ones can. Then, in Section 15.5, we will show how some of the same ideas lead to Shor's FACTORING algorithm, and how to break public-key cryptography.

15.4.1 *Two for One: Deutsch's Problem*

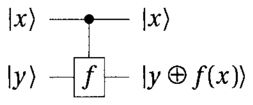
Suppose we have a function f. Classically, if we want to learn something about its values $f(x)$, we have to evaluate it, or *query* it, once for each value of x we care about. We can think of each query as asking an oracle to give us $f(x)$, or calling a subroutine that calculates $f(x)$. Either way, if we want to learn $f(x)$ for N different values of x, this costs us N queries. In quantum mechanics, on the other hand, we can apply f to a superposition of many inputs at once. This allows us, in some sense, to obtain $f(x)$ for multiple values of x in parallel.

Let's start with the simplest possible case, where f takes a single bit x of input and returns a single bit of output. Consider the following problem, which was first proposed in 1985 by David Deutsch: are f's two values $f(0)$ and $f(1)$ the same or different? Equivalently, what is their parity $f(0) \oplus f(1)$? Classically, in order to solve this problem we have to query $f(0)$ and $f(1)$ separately. But quantum mechanically, a single query suffices.

First let's fix what we mean by a query in a quantum algorithm. If we have a qubit y, we can't simply set $y = f(x)$ as we would in the classical case, since this would destroy whatever bit of information was stored in y before. Instead, we query $f(x)$ reversibly by flipping y if $f(x)$ is true. This gives the following unitary operator, where $|x,y\rangle$ is shorthand for $|x\rangle \otimes |y\rangle$:

$$U_f|x,y\rangle = |x, y \oplus f(x)\rangle .$$

We represent U_f in a quantum circuit as follows:



For instance, if $f(0) = 0$ and $f(1) = 1$ then U_f is an ordinary controlled-NOT, while if $f(0) = 1$ and $f(1) = 0$ then U_f flips $|y\rangle$ if and only if $|x\rangle$ is false.

Now suppose we prepare $|x\rangle$ in a uniform superposition of its possible values, i.e., in the state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and that we prepare $|y\rangle$ in the state $|0\rangle$. Then

$$U_f(|+\rangle \otimes |0\rangle) = \frac{1}{\sqrt{2}}\left(|0, f(0)\rangle + |1, f(1)\rangle\right) . \tag{15.8}$$

With a single query, we have created a state that contains information about both $f(0)$ and $f(1)$. We seem to have gotten two queries for the price of one.

Unfortunately, this state is less useful than one might hope. If we measure x, we see either $|0, f(0)\rangle$ or $|1, f(1)\rangle$, each with probability $1/2$. Thus we learn either $f(0)$ or $f(1)$, but we have no control over which. We could do just as well classically by choosing x randomly and querying $f(x)$.

If we want to do better than classical randomized computation, we need to use interference. Let's create a state in which $f(x)$ is encoded in the phase of $|x\rangle$'s amplitude, instead of in the value of $|y\rangle$. To do this, first note that we can write the query operator U_f as

$$U_f|x,y\rangle = |x\rangle \otimes X^{f(x)}|y\rangle .$$

Now prepare $|y\rangle$ in the state $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ instead of $|0\rangle$. Since $|-\rangle$ is an eigenvector of X with eigenvalue -1, we have

$$U_f|x,-\rangle = (-1)^{f(x)}|x,-\rangle .$$

Again preparing $|x\rangle$ in a uniform superposition $|+\rangle$ then gives

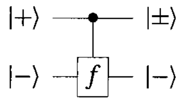
$$U_f(|+\rangle \otimes |-\rangle) = \frac{1}{\sqrt{2}}\left((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right) \otimes |-\rangle .$$

Unlike the state in (15.8), the two qubits are now unentangled. Since we can ignore an overall phase, we can factor out $(-1)^{f(0)}$ and write the state of the first qubit as

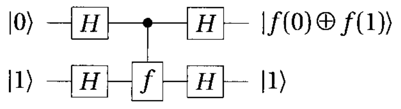
$$|\psi\rangle = \frac{1}{\sqrt{2}}\left((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right) \equiv \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle\right) . \tag{15.9}$$

This is $|+\rangle$ if $f(0) = f(1)$, and $|-\rangle$ if $f(0) \neq f(1)$. Thus if we measure $|\psi\rangle$ in the X-basis, we can learn the parity $f(0) \oplus f(1)$ while only calling the query operator once.

We can express this algorithm as the following circuit,



If we wish to start and end in the computational basis, we can add Hadamard gates on either side, giving



(15.10)

Note the similarity to Exercise 15.9, in which a controlled-NOT goes the other way when transformed to the X-basis.

This algorithm is the first example of a useful trick in quantum computing, called *phase kickback*. We prepare the "output" in an eigenvector, and its eigenvalue affects the phase of the "input." Instead of measuring the output, we throw it away, and learn about the function by measuring the input. This may seem strange. But as Exercise 15.9 showed, information in quantum mechanics always flows both ways.

Before we proceed, let's talk a little more about these quantum queries. How do we actually implement U_f? And if we can implement it, why can't we just look at its matrix entries, and learn whatever we want to know about f?

Suppose we have a device, such as a classical computer, which calculates f. We feed x into the input, activate the device, and then read $f(x)$ from the output. Like any other physical object, this device is governed by the laws of quantum mechanics. For that matter, our classical computers are just quantum computers that spend most of their time in classical states. So we are free to feed it a quantum input rather than a classical one, by preparing the input in a superposition of multiple values of x. We can then implement U_f by arranging for the device to flip a bit y if $f(x) = 1$.

We are assuming here that, after each query, the device's internal state is unentangled from the input. Otherwise, it will partially measure the input state and destroy the quantum superposition over its values. To avoid entanglement, and to be re-usable, the device must return to exactly the same state it had before the query.

This is not the case for the computers on our desks. In addition to the "official" bits of their memory, their state also includes the thermal fluctuations of their particles. If we try to return the memory to its initial state by erasing it, this act of erasure transfers information, and therefore entanglement, to these thermal bits in the form of heat. In principle this obstacle can be overcome by performing the computation reversibly, and returning the memory to its original state without erasing any bits.

However, even if we have an idealized, reversible device in our hands that computes f, this doesn't
15.5 give us instant knowledge of all of f's values. It takes time to feed each value of x into the input of the device, and observe the output $f(x)$. Similarly, being able to implement U_f doesn't mean we can look inside it and see its matrix entries. We have to probe it by applying it to a quantum state and then performing some measurement on the result.

Many of the quantum algorithms we discuss in this chapter treat a function f as a "black box," and try to discover something about f by querying it. When dealing with such problems, we will think of their complexity as the number of queries we make to f, or equivalently, the number of times U_f appears in our circuit. This model makes sense whenever the values $f(x)$ are difficult to compute or obtain, or simply if we want to assume as little as possible about f's internal structure. With this in mind, let's move on to functions of many bits, rather than one.

15.4.2 Hidden Parities and the Quantum Fourier Transform

Suppose that f's input is an n-bit string $\mathbf{x} = x_1 x_2 \cdots x_n$, or equivalently, an n-dimensional vector whose components are integers mod 2. For now, we will assume that f still gives a single bit of output. We define the query operator U_f as

$$U_f|\mathbf{x}, y\rangle = |\mathbf{x}, y \oplus f(\mathbf{x})\rangle\,,$$

where $\mathbf{x}$ is stored in the first n qubits and y is the $(n+1)$st.

Just as we did in the case $n = 1$, let's prepare the input in a uniform superposition over all possible $\mathbf{x}$. This is easy, since

$$\frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}} |\mathbf{x}\rangle = \underbrace{|+\rangle \otimes \cdots \otimes |+\rangle}_{n \text{ times}}\,.$$

If you don't have a supply of $|+\rangle$s, you can start with n qubits in the state $|0\rangle$ and apply a Hadamard gate to each one.

Now let's use the phase kickback trick again and see what we can learn from it. If we prepare y in the eigenvector $|-\rangle$ and apply U_f, we get

$$U_f \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}} |\mathbf{x}, -\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}} U_f|\mathbf{x}, -\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}} (-1)^{f(\mathbf{x})} |\mathbf{x}, -\rangle = |\psi\rangle \otimes |-\rangle\,,$$

where

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}} (-1)^{f(\mathbf{x})} |\mathbf{x}\rangle\,. \qquad (15.11)$$

Like the state $|\psi\rangle$ in (15.9) that we used to solve Deutsch's problem, this state contains the values of $f(\mathbf{x})$ in the phases of $\mathbf{x}$'s amplitudes. What information about f can we extract from it? And in what basis should we measure it?

Stepping back a little, suppose we have a state of the form

$$|\psi\rangle = \sum_{\mathbf{x}} a_{\mathbf{x}} |\mathbf{x}\rangle\,.$$

We can think of its 2^n amplitudes $a_{\mathbf{x}}$ as a function a that assigns a complex number to each vector $\mathbf{x}$. Just as for a function defined on the integers or the reals, we can consider a's Fourier transform. That is, we can write a as a linear combination of basis functions, each of which oscillates at a particular frequency.

In this case, each "frequency" is, like $\mathbf{x}$, an n-dimensional vector mod 2. The basis function with frequency $\mathbf{k}$ is

$$(-1)^{\mathbf{k}\cdot\mathbf{x}} = (-1)^{k_1 x_1 + \cdots + k_n x_n},$$

and the corresponding basis state is

$$|\mathbf{k}\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}} (-1)^{\mathbf{k}\cdot\mathbf{x}} |\mathbf{x}\rangle .$$

Note that we have normalized $|k\rangle$ so that $\|\mathbf{k}\|^2 = 1$.

As we discussed in Section 3.2.3, the Fourier transform is just a unitary change of basis, where we write $|\psi\rangle$ as a linear combination of the $|\mathbf{k}\rangle$s instead of the $|\mathbf{x}\rangle$s:

$$|\psi\rangle = \sum_{\mathbf{k}} \tilde{a}_{\mathbf{k}} |\mathbf{k}\rangle .$$

The amplitudes $\tilde{a}_{\mathbf{k}}$ are the Fourier coefficients of $a_{\mathbf{x}}$,

$$\tilde{a}_{\mathbf{k}} = \langle \mathbf{k} | \psi \rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}} (-1)^{\mathbf{k}\cdot\mathbf{x}} a_{\mathbf{x}} . \tag{15.12}$$

This Fourier transform is precisely the kind of thing quantum computers are good at. Let Q denote the unitary matrix that transforms the $|\mathbf{x}\rangle$ basis to the $|\mathbf{k}\rangle$ basis. We can write its entries as a product over the n bits of $\mathbf{k}$ and $\mathbf{x}$,

$$Q_{\mathbf{k},\mathbf{x}} = \langle \mathbf{k} | \mathbf{x} \rangle = \frac{1}{\sqrt{2^n}} (-1)^{\mathbf{k}\cdot\mathbf{x}} = \prod_{i=1}^{n} \frac{1}{\sqrt{2}} (-1)^{k_i x_i} .$$

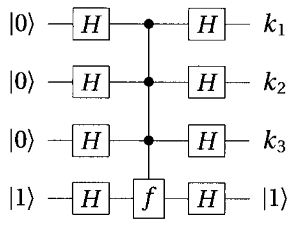
Since the entries of the Hadamard gate are $H_{k,x} = \frac{1}{\sqrt{2}}(-1)^{kx}$, we can write Q as the tensor product

$$Q = \underbrace{H \otimes \cdots \otimes H}_{n \text{ times}} .$$

Thus we can implement Q simply by applying H to each qubit.

We call Q the *quantum Fourier transform*, or QFT for short. To be precise, it is the QFT on the group $\mathbb{Z}_2^n$, the set of n-dimensional vectors mod 2. We will consider QFTs over other groups later. We stress, however, that Q is not a quantum algorithm for the Fourier transform, since the Fourier coefficients $\tilde{a}_{\mathbf{k}}$ are not given to us as its output. Rather, they are encoded in $|\psi\rangle$'s amplitudes, and must be revealed by measurement in the Fourier basis.

Specifically, let's return to the state of $|\psi\rangle$ of (15.11). The following circuit prepares ψ and then applies Q, where for illustration $n = 3$:



(15.13)

This leaves us with the transformed state $Q|\psi\rangle \otimes |1\rangle$, in which the bits $k_1 k_2 \cdots k_n$ of the frequency are written in the computational basis. If we measure these bits, we observe a given $\mathbf{k}$ with probability

$$P(\mathbf{k}) = \left|\langle \mathbf{k}|\psi\rangle\right|^2 = |\tilde{a}_{\mathbf{k}}|^2 \, ,$$

where

$$a_{\mathbf{x}} = \frac{1}{\sqrt{2^n}}(-1)^{f(\mathbf{x})} \quad \text{and} \quad \tilde{a}_{\mathbf{k}} = \frac{1}{2^n}\sum_{\mathbf{x}}(-1)^{\mathbf{k}\cdot\mathbf{x}+f(\mathbf{x})} .$$

Compare this with the case $n=1$, namely the circuit (15.10) for Deutsch's problem. There the two possible frequencies were $k=0$ and $k=1$. If $f(0)=f(1)$ we had $|\tilde{a}_0|^2 = 1$ and $|\tilde{a}_1|^2 = 0$, and vice versa if $f(0) \neq f(1)$.

We now have everything we need to solve two more problems proposed in the early days of quantum computing. One, the *Deutsch–Jozsa problem*, is admittedly artificial, but it illustrates what the quantum Fourier transform can do. Suppose I promise you that one of the following two statements is true:

1. f is constant, i.e., $f(\mathbf{x})$ is the same for all $\mathbf{x}$.

2. f is balanced, i.e., $f(\mathbf{x}) = 0$ for half of all $\mathbf{x}$, and $f(\mathbf{x}) = 1$ for the other half.

The question is, which one? The trick is to consider the probability $P(\mathbf{0})$ of observing the frequency $\mathbf{0} = (0,\ldots,0)$. The corresponding Fourier coefficient is

$$\tilde{a}_{\mathbf{0}} = \frac{1}{\sqrt{2^n}}\sum_{\mathbf{x}} a_{\mathbf{x}} = \frac{1}{2^n}\sum_{\mathbf{x}}(-1)^{f(\mathbf{x})} .$$

This is just the average of $(-1)^{f(\mathbf{x})}$ over all $\mathbf{x}$, which is ± 1 if f is constant and 0 if f is balanced. Therefore,

$$P(\mathbf{0}) = |\tilde{a}_{\mathbf{0}}|^2 = \begin{cases} 1 & \text{if } f \text{ is constant} \\ 0 & \text{if } f \text{ is balanced} . \end{cases}$$

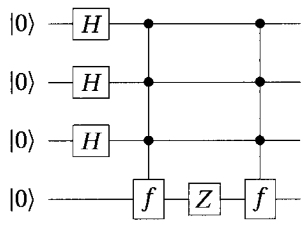
Assuming my promise is true, one query of f and one observation of $\mathbf{k}$ is all it takes. Just return "constant" if $k_i = 0$ for all i, and return "balanced" otherwise.

We can use the same setup to solve another problem, posed by Bernstein and Vazirani. Again I make a promise about f: this time, that $f(\mathbf{x}) = \mathbf{k}\cdot\mathbf{x} \bmod 2$ for some $\mathbf{k}$. Equivalently, I promise that $f(\mathbf{x})$ is the parity of some fixed subset of $\mathbf{x}$'s components, namely those x_i where $k_i = 1$. In this case, $|\psi\rangle$ is precisely the basis state $|\mathbf{k}\rangle$, so measuring $|\psi\rangle$ in the Fourier basis reveals $\mathbf{k}$ with probability 1. As for the Deutsch and Deutsch–Jozsa problems, one quantum query is enough to discover $\mathbf{k}$.

In contrast, Problem 15.19 shows that classical computers need n queries to solve the Bernstein–Vazirani problem. This gives us our first problem where quantum computing reduces the number of queries by more than a constant. Our first *exponential* speedup is right around the corner.
15.11

Exercise 15.16 *Give a randomized classical algorithm for the Deutsch–Jozsa problem where the average number of queries is* 3 *or less.*

Exercise 15.17 *Several early papers generated the state* $|\psi\rangle$ *of* (15.11) *in a more elaborate way. The idea is to compute* $f(\mathbf{x})$*, apply the* Z *operator, and then disentangle the target qubit from the input qubits by "uncomputing"* $f(\mathbf{x})$*:*



Show that this works, although it uses two queries instead of one.

15.4.3 *Hidden Symmetries and Simon's Problem*

We close this section with Simon's problem. This was the first case in which quantum computers were proved to be exponentially faster than classical ones, and was also an important precursor to Shor's algorithm for FACTORING. It illustrates one of quantum computation's most important strengths—its ability to detect a function's symmetries.

Once again, let f be a function of n-dimensional vectors $\mathbf{x}$ whose coefficients are integers mod 2. But now, rather than giving a single bit of output, $f(\mathbf{x})$ returns a symbol in some set S. We don't care much about the structure of S, or the specific values f takes. We just care about f's symmetries—that is, which values of $\mathbf{x}$ have the same $f(\mathbf{x})$.

Specifically, I promise that each input $\mathbf{x}$ has exactly one partner $\mathbf{x}'$ such that $f(\mathbf{x}) = f(\mathbf{x}')$. Moreover, I promise that there is a fixed vector $\mathbf{y}$ such that

$$f(\mathbf{x}) = f(\mathbf{x}') \text{ if and only if } \mathbf{x}' = \mathbf{x} \oplus \mathbf{y}. \tag{15.14}$$

For instance, if $n = 3$ and $\mathbf{y} = 101$, then

$$f(000) = f(101),\ f(001) = f(100),\ f(010) = f(111),\ f(011) = f(110),$$

and these four values of f are distinct. How many times do we need to query f to learn $\mathbf{y}$?

As before, we define a query operator U_f which writes $f(\mathbf{x})$ reversibly by XORing it with another set of qubits. If we prepare $\mathbf{x}$ in a uniform superposition and prepare the "output" qubits in the state $|\mathbf{0}\rangle = |0,\ldots,0\rangle$, this gives

$$U_f \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}} |\mathbf{x}, \mathbf{0}\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}} |\mathbf{x}, f(\mathbf{x})\rangle .$$

Now suppose we measure the value of $f(\mathbf{x})$. We will observe one of the 2^{n-1} possible outputs f_0, all of which are equally likely. After we have done so, the state has collapsed to a superposition of those $\mathbf{x}$ with $f(\mathbf{x}) = f_0$. According to my promise, the resulting state is $|\psi\rangle \otimes |f_0\rangle$ where

$$|\psi\rangle = \frac{1}{\sqrt{2}} \left(|\mathbf{x}_0\rangle + |\mathbf{x}_0 \oplus \mathbf{y}\rangle \right)$$

for some $\mathbf{x}_0$.

This state $|\psi\rangle$ is a superposition over two states $\mathbf{x}_0$ and $\mathbf{x}_0' = \mathbf{x}_0 \oplus \mathbf{y}$. For those unschooled in quantum mechanics, it is very tempting to think that we can just add these states together, and get $\mathbf{y} = \mathbf{x}_0 \oplus \mathbf{x}_0'$. However, this kind of interaction between two terms in a superposition just isn't allowed. Instead, we have to perform some measurement on $|\psi\rangle$, and learn something about $|\mathbf{y}\rangle$ from the result.

One thing we shouldn't do is measure $|\psi\rangle$ in the $\mathbf{x}$-basis, since this would just yield a uniformly random $\mathbf{x}$. We should measure $|\psi\rangle$ is some other basis instead. The right choice is—you guessed it—the Fourier basis. According to (15.12), applying the QFT gives the amplitudes

$$\tilde{a}_{\mathbf{k}} = \frac{1}{\sqrt{2^n}} \frac{1}{\sqrt{2}} \left((-1)^{\mathbf{k}\cdot\mathbf{x}_0} + (-1)^{\mathbf{k}\cdot(\mathbf{x}_0 \oplus \mathbf{y})} \right) = \frac{(-1)^{\mathbf{k}\cdot\mathbf{x}_0}}{\sqrt{2^{n-1}}} \left(\frac{1+(-1)^{\mathbf{k}\cdot\mathbf{y}}}{2} \right). \tag{15.15}$$

Note that $\mathbf{x}_0$, and therefore the value f_0 we happened to observe, just affects the overall phase. This phase disappears when we calculate the probability $P(\mathbf{k})$ that we observe a given frequency vector $\mathbf{k}$,

$$P(\mathbf{k}) = |\tilde{a}_{\mathbf{k}}|^2 = \frac{1}{2^{n-1}} \left(\frac{1+(-1)^{\mathbf{k}\cdot\mathbf{y}}}{2} \right)^2 .$$

This probability depends on whether $\mathbf{k}\cdot\mathbf{y}$ is 0 or 1, or geometrically, whether $\mathbf{k}$ is perpendicular to $\mathbf{y}$ or not. We can write it as follows,

$$P(\mathbf{k}) = \begin{cases} 1/2^{n-1} & \text{if } \mathbf{k} \perp \mathbf{y} \\ 0 & \text{otherwise}. \end{cases}$$

Thus we observe a frequency vector $\mathbf{k}$ chosen randomly from the 2^{n-1} vectors perpendicular to $\mathbf{y}$. The set of such vectors forms an $(n-1)$-dimensional subspace, and we can determine $\mathbf{y}$ as soon as we have observed a set of $\mathbf{k}$s that span this subspace. As Problem 15.20 shows, this happens with constant probability after n observations, and with high probability after $n + o(n)$ of them. So, we can find $\mathbf{y}$ by querying f about n times.

Classically, on the other hand, it takes an exponential number of queries to determine $\mathbf{y}$. Using the Birthday Problem from Appendix A.3.3, Problem 15.21 shows that it takes about $\sqrt{2^n} = 2^{n/2}$ queries to find a pair $\mathbf{x}, \mathbf{x}'$ such that $f(\mathbf{x}) = f(\mathbf{x}')$. Until then, we have essentially no information about $\mathbf{y}$.

Simon's algorithm shows that quantum computers can solve certain problems—at least in the black-box model, where the number of queries is what matters—exponentially faster than classical computers can. It is also our first example of a *Hidden Subgroup Problem*, where a function f is defined on a group G and our goal is to find the subgroup H that describes f's symmetries. In this case, $G = \mathbb{Z}_2^n$ and $H = \{\mathbf{0}, \mathbf{y}\}$. In the next section, we will see how Shor's algorithm solves FACTORING by finding the symmetry—in particular, the periodicity—of a function defined on the integers.

15.5 Cryptography and Shor's Algorithm

It is fair to say that most of the interest in quantum computation stems from Peter Shor's discovery, in 1994, that quantum computers can solve FACTORING in polynomial time. In addition to being one of the most fundamental problems in mathematics, FACTORING is intimately linked with modern cryptography.

In this section, we describe the RSA public-key cryptosystem and why an efficient algorithm for FACTORING would break it. We then show how Shor's algorithm uses the quantum Fourier transform, and a little number theory, to solve FACTORING in polynomial time. We go on to show that quantum computers can also solve the DISCRETE LOG problem, and break another important cryptographic protocol.

15.5.1 The RSA Cryptosystem

It may be roundly asserted that human ingenuity cannot concoct a cipher which human ingenuity cannot resolve.

Edgar Allen Poe

Several times in this book, we have discussed *one-way functions*—functions f such that we can compute $f(x)$ from x in polynomial time, but we can't invert f and compute x from $f(x)$. For instance, we believe that modular exponentiation, $f(x) = a^x \bmod p$, is one such function.

Now suppose that f has a *trapdoor*: a secret key d which, if you know it, lets you invert f easily. Then I can arrange a *public-key cryptosystem*, in which anyone can send me messages that only I can read. I publicly announce how to compute f, and anyone who wants to send me a message m can encrypt it in the form $f(m)$. But only I know the secret key d, so only I can invert $f(m)$ and read the original message.

This is the idea behind the *RSA cryptosystem*, named after Rivest, Shamir, and Adleman who published it in 1977. Today, it is the foundation for much of electronic commerce. Here's how it works.

I publish an n-bit integer N, which is the product of two large primes p, q that I keep to myself, and an 15.12
n-bit integer e. Thus my public key consists of the pair (N, e). If you want to send me a message written as an n-bit integer m, you encrypt it using the following function:

$$f(m) = m^e \bmod N .$$

I have a private key d known only to me, such that

$$f^{-1}(m) = m^d \bmod N .$$

In other words, for all m we have

$$(m^e)^d = m^{ed} \equiv_N m , \tag{15.16}$$

so I can decrypt your message m^e simply by raising it to the dth power. Both encryption and decryption can be done in polynomial time using the repeated-squaring algorithm for Modular Exponentiation in Section 3.2.2.

For what pairs e, d does (15.16) hold? Recall Fermat's Little Theorem, which we saw in Sections 4.4.1 and 10.8. It states that if p is prime, then for all a,

$$a^{p-1} \equiv_p 1 .$$

What happens if we replace the prime p with an arbitrary number N? Let $\mathbb{Z}_N^*$ denote the set of numbers between 1 and $N-1$ that are mutually prime to N. This set forms a group under multiplication. As we discuss in Appendix A.7, for any element a of a group G we have

$$a^{|G|} = 1 .$$

If p is prime then $|\mathbb{Z}_p^*| = p - 1$, recovering the Little Theorem.

More generally, $|\mathbb{Z}_N^*|$ is given by Euler's *totient function* $\varphi(N)$. For instance, for $N = 15$ we have

$$\mathbb{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\} ,$$

so $\varphi(15)=8$. Thus for any N and any $a \in \mathbb{Z}_N^*$,

$$a^{\varphi(N)} \equiv_N 1\,. \tag{15.17}$$

It follows that (15.16) holds whenever

$$ed \equiv_{\varphi(N)} 1\,.$$

This implies that e and d are mutually prime to $\varphi(N)$, and that $d = e^{-1}$ is the multiplicative inverse of e in the group $\mathbb{Z}_{\varphi(N)}^*$. Note that for (15.16) to hold, we also need the message m to be in $\mathbb{Z}_N^*$. However, if $N = pq$ where p and q are large primes, the probability that m isn't mutually prime to N is at most $1/p + 1/q$, which is vanishingly small.

How hard is it to break this system, and decrypt messages that are intended for someone else? Cracking my RSA key would consist of deriving my private key d from my public key (N, e). Let's give this problem a name:

RSA KEYBREAKING

Input: Integers N, e with $\gcd(e, \varphi(N)) = 1$

Output: $d = e^{-1}$ in $\mathbb{Z}_{\varphi(N)}^*$

If you know φ, you can find inverses in $\mathbb{Z}_\varphi^*$ in polynomial time using a version of Euclid's algorithm (see Problem 2.2). Thus RSA KEYBREAKING can be reduced to the problem TOTIENT of finding $\varphi(N)$ given N.

TOTIENT, in turn, can be reduced to FACTORING for the following reason. If $N = pq$ where p and q are distinct primes,

$$\varphi(N) = (p-1)(q-1)\,. \tag{15.18}$$

Exercise 15.18 *Prove* (15.18). *Better yet, work out its generalization in Problem 15.25.*

Thus we have a chain of reductions

$$\text{RSA KEYBREAKING} \le \text{TOTIENT} \le \text{FACTORING}\,.$$

Do these reductions go the other way? The following exercise shows that TOTIENT is just as hard as FACTORING, at least for the case $N = pq$:

Exercise 15.19 *Suppose $N = pq$ where p and q are prime. Show that, given N and $\varphi(N) = (p-1)(q-1)$, we can derive p and q.*

In addition, Problem 15.28 shows that if we can solve RSA KEYBREAKING—or, for that matter, if we can obtain even a single pair (e, d) where $ed \equiv_{\varphi(N)} 1$—then there is a randomized polynomial-time algorithm which factors N. This gives

$$\text{FACTORING} \le_{\text{RP}} \text{RSA KEYBREAKING}\,,$$

where $\le_{\text{RP}}$ denotes a randomized polynomial-time reduction as in Section 10.9.

Thus if we allow randomized algorithms, FACTORING and RSA KEYBREAKING are equally hard. However, we could conceivably decrypt individual messages, extracting the eth root of m^e, without possessing the private key d. Thus decrypting one message at a time might be easier than RSA KEYBREAKING, which
15.13 breaks the public key once and for all.

The best known classical algorithm for factoring an n-bit integer has running time $\exp\left(O(n^{1/3}\log^{2/3} n)\right)$. This is much better than simple approaches like trial division or the sieve of Eratosthenes, but it is still exponential in n. It is generally believed that FACTORING is not in P, and therefore that RSA KEYBREAKING 15.14 is hard for classical computers to break. What about quantum ones?

15.5.2 *From Factoring to Order-Finding*

Shor's algorithm solves FACTORING by finding the periodicity of a certain number-theoretic function. We have already seen a hint of this idea in Section 10.8.2 where we discussed the Miller–Rabin test for PRIMALITY, but we present it here from scratch.

We start by asking about the square roots of 1. For any N, there are at least two elements $y \in \mathbb{Z}_N^*$ such that $y^2 \equiv_N 1$, namely 1 and $-1 \equiv_N N-1$. But if N is divisible by two distinct odd primes, there are at least two more square roots of 1 in $\mathbb{Z}_N^*$, which we call the *nontrivial* ones. If we can find one of them, we can obtain a proper divisor of N. To see this, suppose that $y^2 \equiv_N 1$. Then for some k,

$$y^2 - 1 = (y+1)(y-1) = kN\,.$$

If $y \not\equiv_N \pm 1$, neither $y+1$ nor $y-1$ is a multiple of N. In that case, $\gcd(y+1, N)$ and $\gcd(y-1, N)$ must each be proper divisors of N.

How can we find a nontrivial square root of 1? Given an integer $c \in \mathbb{Z}_N^*$, its *order* is the smallest $r > 0$ such that

$$c^r \equiv_N 1\,.$$

In other words, r is the periodicity of the function

$$f(x) = c^x \bmod N\,.$$

This gives us a potential algorithm for FACTORING. Suppose that we can find the order r of a given c. If we're lucky, r is even and $c^{r/2}$ is a square root of 1. We know that $c^{r/2} \not\equiv 1$ since by definition r is the smallest such power. If we're even luckier, $c^{r/2} \not\equiv_N -1$ and $c^{r/2}$ is a nontrivial root.

For example, let $N = 21$ and let $c = 2$. The powers of 2 mod 21 are

x	0	1	2	3	4	5	6	7	8	9	10	11	12	$\cdots$
$c^x \bmod N$	1	2	4	8	16	11	1	2	4	8	16	11	1	$\cdots$

The order is $r = 6$, since $2^6 = 64 \equiv_{21} 1$ is the first time this sequence repeats. Then $c^{r/2} = 2^3 = 8$ is one of the nontrivial square roots of 1 in $\mathbb{Z}_{21}^*$, and $\gcd(9, 21)$ and $\gcd(7, 21)$ are N's factors 3 and 7.

This doesn't always work. If $c = 5$, its order is $r = 6$, but $c^{r/2} = 20 \equiv_N -1$. As another example, if $c = 16$ then its order $r = 3$ is odd. Let's call $c \in \mathbb{Z}_N^*$ *good* if its order r is even and $c^{r/2} \not\equiv_N -1$. How can we find a good c?

Simple—just choose c randomly. To be more precise, first choose c randomly from $\{0, \ldots, N-1\}$, and then check that c is mutually prime to N. If it isn't, $\gcd(c, N)$ is a proper divisor of N and we can declare victory already—so we might as well assume that c is a uniformly random element of $\mathbb{Z}_N^*$. If a large fraction of the elements of $\mathbb{Z}_N^*$ are good, we only need to try this a few times before we find a good one. The following theorem, which we prove in Problem 15.29, shows that this is true.

Theorem 15.1 *If N is divisible by ℓ distinct odd primes, then at least a fraction $1-1/2^{\ell-1}$ of the elements of $\mathbb{Z}_N^*$ are good.*

Thus if N is divisible by two or more odd primes, a random element of $\mathbb{Z}_N^*$ is good with probability at least $1/2$. Then we can find a good c, and a proper divisor of N, with $O(1)$ trials on average.

This covers every case except where N is even, prime, or a prime power. As Problem 10.41 shows, we can tell if N is a prime power in polynomial time, and we can certainly tell if it is even, so we can find a divisor in those cases too.

If we can find a proper divisor d of N, we can apply the same algorithm recursively to d and N/d and break them down further. If N has n bits, it has only $O(n)$ prime factors (see Exercise 4.21) so we only need to do this $O(n)$ times. Finally, we can use the polynomial-time algorithm for PRIMALITY to tell when we have broken N all the way down to the primes.

We can summarize all this as a chain of reductions,

$$\text{FACTORING} \le \text{NONTRIVIAL SQUARE ROOT OF } 1 \le_{\text{RP}} \text{ORDER FINDING}.$$

So how can we find the order of a given c? Equivalently, how can we find the periodicity of the function $f(x)=c^x \bmod N$?

Let's leave the number theory behind and treat $f(x)$ as a black box. Given a function that repeats with periodicity r, how can we find r? If we query it classically, computing $f(1)$, $f(2)$, and so on, it will take r queries before we see the sequence repeat. This is bad news, since in our application r could be exponentially large.

But periodicity is a kind of symmetry, and Simon's algorithm showed us that quantum computers can detect some kinds of symmetry exponentially faster than a classical computer can. In the next section, we will see exactly how Shor's algorithm does this.

15.5.3 Finding Periodicity With the QFT

Suppose that f is a function from the integers to some arbitrary set S, and that f is periodic in the following sense. There is an r such that, for all x, x',

$$f(x)=f(x') \text{ if and only if } x \equiv_r x'. \tag{15.19}$$

Intuitively, we can learn about f's oscillations by measuring its Fourier transform, and this is exactly what we will do. Just as the QFT on the group $\mathbb{Z}_2^n$ helped us solve Simon's problem, we can find r using the QFT on the integers, or rather on $\mathbb{Z}_M$, the group of integers mod M for some M.

What should M be? As we will see below, if r divides M then the observed frequency is always a multiple of a fundamental frequency M/r, allowing us to determine r from a few observations. Since r divides $\varphi(N)$, we would set $M=\varphi(N)$ if we could—but if we knew $\varphi(N)$, we would already know how to factor N. For now, we will do the analysis as if r divides M. Happily, in the next section we will see that everything still works as long as M is sufficiently large.

We start by creating a uniform superposition over all x in $\mathbb{Z}_M$, and then using the unitary operator U_f to write $f(x)$ on another set of qubits. This gives the state

$$\frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x, f(x)\rangle .$$

We then measure $f(x)$, and observe some random value f_0. This collapses the state to $|\psi\rangle \otimes |f_0\rangle$ where $|\psi\rangle$ is the uniform superposition over the x such that $f(x) = f_0$. According to (15.19), this is precisely the set of x that are equivalent mod r to some x_0. There are M/r such values of x, so we have

$$|\psi\rangle = \frac{1}{\sqrt{M/r}} \sum_{x \equiv_r x_0} |x\rangle = \sqrt{\frac{r}{M}} \sum_{t=0}^{M/r-1} |tr + x_0\rangle . \tag{15.20}$$

The quantum Fourier transform over $\mathbb{Z}_M$ is exactly the unitary operator we described in Section 3.2.3,

$$Q_{kx} = \frac{1}{\sqrt{M}} \omega^{kx} ,$$

where

$$\omega = e^{2\imath\pi/M}$$

is the Mth root of 1 in the complex plane. Analogous to (15.12), applying Q to a state $|\psi\rangle = \sum_x a_x |x\rangle$ gives amplitudes $\tilde{a}_k$ that are Fourier coefficients,

$$\tilde{a}_k = \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} \omega^{kx} a_x .$$

Below we will discuss how to implement Q efficiently, i.e., with a polynomial number of quantum gates.

The Fourier coefficients of the state (15.20) are

$$\tilde{a}_k = \frac{\sqrt{r}}{M} \sum_{t=0}^{M/r-1} \omega^{k(tr+x_0)} = \frac{\sqrt{r}}{M} \omega^{kx_0} \sum_{t=0}^{M/r-1} \omega^{krt} . \tag{15.21}$$

As in (15.15), the value $f_0 = f(x_0)$ that we observed only affects the overall phase. When we measure $|\psi\rangle$ in the Fourier basis, the probability $P(k)$ that we observe a given frequency k is then

$$P(k) = |\tilde{a}_k|^2 = \frac{r}{M^2} \left| \sum_{t=0}^{M/r-1} \omega^{krt} \right|^2 .$$

There are now two cases. If k is a multiple of M/r, which we have assumed here is an integer, then $\omega^{kr} = 1$. In that case each term in the sum over t is 1, so these amplitudes interfere constructively and $P(k) = 1/r$. On the other hand, if k is not a multiple of M/r then they interfere destructively, rotating in the complex plane and canceling out, and $P(k) = 0$. Thus

$$P(k) = \begin{cases} 1/r & \text{if } k \text{ is a multiple of } M/r \\ 0 & \text{otherwise} . \end{cases}$$

Note that all r multiples of M/r between 0 and $M-1$ are equally likely, so we can write $k = bM/r$ where b is chosen randomly from $0, \ldots, r-1$. In acoustic terms, M/r is the fundamental frequency, and k is chosen randomly from its harmonics.

If each observation gives us a random harmonic, how long does it take to learn the fundamental frequency? Suppose we observe the frequencies $k_1 = b_1 M/r$, $k_2 = b_2 M/r$, and so on. As soon as the b_i are relatively prime, i.e., $\gcd(b_1, b_2, \ldots) = 1$, then $\gcd(k_1, k_2, \ldots) = M/r$. With constant probability even the first two observations suffice, since as Problem 10.26 shows, b_1 and b_2 are mutually prime with probability $6/\pi^2$ when r is large. Thus we can determine M/r from a constant number of observations on average, and we simply divide M by M/r to obtain r.

In this analysis, the interference was either completely constructive or completely destructive, because of our assumption that r divides M. We relax this assumption in the next section. While the analysis is a little more difficult, it involves some machinery that is quite beautiful in its own right.

15.5.4 Tight Peaks and Continued Fractions

What happens when the "fundamental frequency" M/r is not an integer? We will show that when M is large enough, the probability distribution $P(k)$ is still tightly peaked around multiples of M/r. Then with a little extra effort, we can again find the periodicity r.

First let ℓ be the number of x between 0 and $M-1$ such that $x \equiv_r x_0$, and note that ℓ is either $\lfloor M/r \rfloor$ or $\lceil M/r \rceil$. Preparing a uniform superposition over $\mathbb{Z}_M$ and measuring $f(x)$ then gives the state

$$|\psi\rangle = \frac{1}{\sqrt{\ell}} \sum_{t=0}^{\ell-1} |rt + x_0\rangle .$$

Applying the QFT as before, (15.21) becomes

$$\tilde{a}_k = \frac{1}{\sqrt{M\ell}} \sum_{t=0}^{\ell-1} \omega^{k(x_0+rt)} = \frac{\omega^{kx_0}}{\sqrt{M\ell}} \sum_{t=0}^{\ell-1} \omega^{krt} . \tag{15.22}$$

This is a geometric series, and summing it gives

$$\tilde{a}_k = \frac{\omega^{kx_0}}{\sqrt{M\ell}} \left(\frac{1-\omega^{kr\ell}}{1-\omega^{kr}} \right) .$$

The probability $P(k)$ that we observe the frequency k is then

$$P(k) = |\tilde{a}_k|^2 = \frac{1}{M\ell} \frac{\left|1-\omega^{kr\ell}\right|^2}{\left|1-\omega^{kr}\right|^2} . \tag{15.23}$$

Let's rewrite this in terms of the phase angle θ between successive terms in the series (15.22). That is,

$$\omega^{kr} = \mathrm{e}^{\imath\theta} \text{ where } \theta = \frac{2\pi k r}{M} . \tag{15.24}$$

Then using the identity

$$\left|1-\mathrm{e}^{\imath\theta}\right|^2 = 4\sin^2\frac{\theta}{2} ,$$

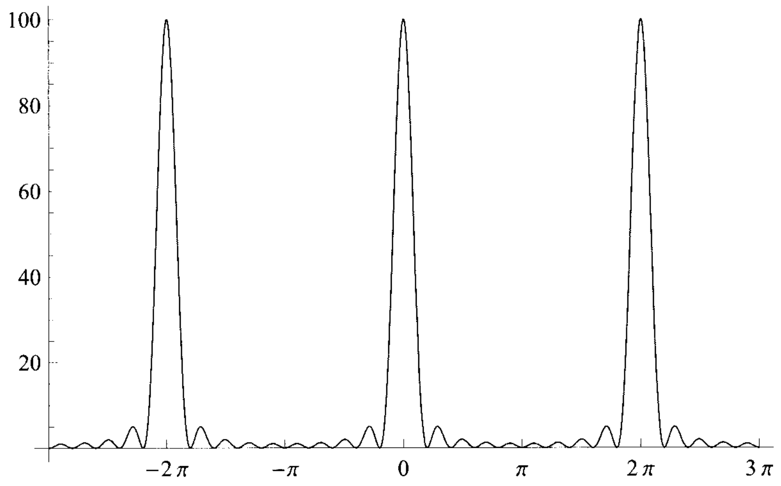


FIGURE 15.5: A plot of the function $\sin^2(\ell\theta/2)/\sin^2(\theta/2)$ appearing in the probability $P(k)$. It peaks with a value of ℓ^2 at the multiples of 2π, and falls off rapidly for other values of θ. Here $\ell = 10$.

we can rewrite (15.23) as

$$P(k) = \frac{1}{M\ell}\left(\frac{4\sin^2 \ell\theta/2}{4\sin^2 \theta/2}\right) = \frac{1}{M\ell}\left(\frac{\sin^2 \ell\theta/2}{\sin^2 \theta/2}\right). \tag{15.25}$$

Intuitively, if θ is small enough, i.e., close enough to an integer multiple of 2π, the terms in the sum (15.22) interfere constructively and $P(k)$ is large. Otherwise, their phases rotate rapidly in the complex plane and interfere destructively, making $P(k)$ very small. As Figure 15.5 shows, this intuition is correct. The function $\sin^2(\ell\theta/2)/\sin^2(\theta/2)$ appearing in (15.25) is tightly peaked at the multiples of 2π, so $P(k)$ is as well. How tight are these peaks, and how close to them are our observations likely to be?

In the simpler case where r divides M, θ is exactly a multiple of 2π when k is a multiple of M/r. So let's assume that k is the closest integer to a multiple of M/r. That is, assume that for some integer b we have

$$\left|k - \frac{bM}{r}\right| \le 1/2, \tag{15.26}$$

so that k is either $\lfloor bM/r \rfloor$ or $\lceil bM/r \rceil$. Plugging this in to (15.24) shows that

$$|\theta - 2\pi b| \le \frac{\pi r}{M}. \tag{15.27}$$

So if M is much larger than r, θ is indeed very close to a multiple of 2π.

Now write $\ell = M/r$. The fact that it is, instead, $\lfloor M/r \rfloor$ or $\lceil M/r \rceil$ makes only a minute difference in what follows. If we write

$$\phi = \frac{\ell}{2}(\theta - 2\pi b) = \frac{M}{2r}(\theta - 2\pi b),$$

then combining (15.25) and (15.27) gives

$$P(k) = \frac{1}{M\ell}\left(\frac{\sin^2\phi}{\sin^2\phi/\ell}\right) \quad \text{where } |\phi| \leq \pi/2. \tag{15.28}$$

For ϕ in this range, a little calculus gives the inequality

$$\frac{\sin^2\phi}{\sin^2\phi/\ell} \geq \frac{4}{\pi^2}\ell^2, \tag{15.29}$$

so that (15.28) gives

$$P(k) \geq \frac{4}{\pi^2}\frac{\ell}{M} = \frac{4}{\pi^2}\frac{1}{r}. \tag{15.30}$$

Recall that in the simpler case where r divides M, the observed frequency k equals each of the r multiples of M/r with probability $1/r$. What (15.30) says is that k is the integer closest to each of these multiples with probability just a constant smaller than $1/r$. To put this differently, if we sum over all r multiples, the probability that k is the integer closest to one of them is at least $4/\pi^2$.

Let's assume from now on that k is indeed one of these closest integers. We can rewrite (15.26) as

$$\left|\frac{k}{M} - \frac{b}{r}\right| \leq \frac{1}{2M}, \tag{15.31}$$

To put this differently, b/r is a good approximation to k/M, but with a smaller denominator $r < M$. Thus we can phrase the problem of finding r as follows: given a real number $y = k/M$, how can we find a rational approximation $b/r \approx y$ with high accuracy but a small denominator?

There is a beautiful theory of such approximations, namely the theory of *continued fractions*. For example, consider the following expression:

$$\pi = 3 + \cfrac{1}{7 + \cfrac{1}{15 + \cfrac{1}{1 + \cfrac{1}{292 + \cdots}}}}.$$

The coefficients $3, 7, 15, 1, 292, \ldots$ can be obtained by iterating the Gauss map $g(x) = 1/x \bmod 1$, which we encountered in Problem 2.7, and writing down the integer part $\lfloor 1/x \rfloor$ at each step.

15.15 If we cut this fraction off at 3, 7, 15, and so on, we get a series of rational approximations for π which oscillate around the correct value:

$$3, \frac{22}{7}, \frac{333}{106}, \frac{355}{113}, \frac{103\,993}{33\,102}, \ldots$$

Rationals obtained from y's continued fraction expansion in this way are called *convergents*. It turns out that the convergents include all the best rational approximations to y, in the following sense. We say that a rational number b/r is an *unusually good* approximation of y if

$$\left|y - \frac{b}{r}\right| < \frac{1}{2r^2}.$$

We call this unusually good because, for a typical denominator r, the closest that b/r can get to y is about $1/r$. The following classic theorem of number theory was proved by Lagrange:

Theorem 15.2 *If b/r is an unusually good approximation for y, it is one of y's convergents.*

See Problem 15.24 for a proof.

Looking back at (15.31), we see that b/r is an unusually good approximation to k/M whenever $M > r^2$. Since $r < N$ where N is the number we're trying to factor, it suffices to take $M > N^2$. So we measure the frequency k, use the continued fraction expansion of k/M to generate all the convergents with denominators smaller than N, and check the denominator of each one. As soon as we find a denominator r such that $f(r) = f(0)$, we're done.

If b and r are not mutually prime, the convergent b/r reduces to a fraction whose denominator is some proper divisor of r. However, if b is chosen uniformly from $0, \ldots, r-1$, then b is mutually prime to r with probability $\varphi(r)/r$ where φ is Euler's totient function. Problem 15.26 guides you through an elementary proof of the following fact: there is a constant $C > 0$ such that

$$\frac{\varphi(r)}{r} \geq \frac{C}{\log r}. \tag{15.32}$$

Since r has $O(n)$ bits, the probability that b is mutually prime to r is then

$$\frac{\varphi(r)}{r} = \Omega(1/\log r) = \Omega(1/n),$$

so the number of attempts we need to make is just $O(n)$. In fact, we really only need $O(\log n)$ attempts because of the stronger result that, for some $C' > 0$, 15.16

$$\frac{\varphi(r)}{r} \geq \frac{C'}{\log\log r}. \tag{15.33}$$

This can be proved using the Prime Number Theorem, which bounds the density of the primes.

There's only one thing missing from this algorithm. How can we carry out the quantum Fourier transform on $\mathbb{Z}_M$? That is, how can we implement the unitary operator Q with an efficient quantum circuit? The good news is that, having gone through all this analysis, we are free to take any value of M greater than N^2, and can set M to make the QFT as simple as possible. As we will see next, if M is a power of 2 we can apply a divide-and-conquer approach analogous to the Fast Fourier Transform of Section 3.2.3, and carry out the QFT with a polynomial number of elementary gates.

15.5.5 *From the FFT to the QFT*

In Section 3.2.3, we showed how to carry out the Fourier transform recursively, by dividing a list of length M into two sublists of size $M/2$ and Fourier transforming each one. We can use the same idea in the quantum setting. That is, if Q_M denotes the QFT on $\mathbb{Z}_M$, we can write a quantum circuit for Q_M recursively in terms of that for $Q_{M/2}$.

Our goal is to transform from the $|x\rangle$ basis to the $|k\rangle$ basis:

$$Q_M|x\rangle = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} \omega_M^{kx} |k\rangle,$$

where once again $\omega_M = e^{2\imath\pi/M}$ denotes the Mth root of 1. Let's write x and k in the following way:

$$x = 2x' + x_0 \quad \text{and} \quad k = \frac{M}{2} k_0 + k' ,$$

where $x_0, k_0 \in \{0,1\}$ and $0 \le x', k' < M/2$. In particular, suppose that $M = 2^m$ and we write x and k in binary. Then x_0 is the least significant bit of x, k_0 is the most significant bit of k, and x' and k' are the remaining $m-1$ bits of x and k respectively.

In order to use the same m qubits for both x and k, it will be convenient to store the bits of x in reverse order, starting with the least significant. Thus we will write $|x\rangle = |x_0, x'\rangle$ and $|k\rangle = |k_0, k'\rangle$, so that the first qubit contains x_0 at the beginning of the process and ends up containing k_0. The remaining $m-1$ qubits start out containing x' and end up containing k'.

We can write the phase factor ω_M^{kx} as a product of three terms,

$$\omega_M^{kx} = \omega_{M/2}^{k'x'} \, \omega_M^{k'x_0} \, (-1)^{k_0 x_0} ,$$

where we used the facts $\omega_M^{M/2} = -1$ and $\omega_M^2 = \omega_{M/2}$. These three terms will correspond to three steps of our algorithm. First, we apply $Q_{M/2}$ to $|x'\rangle$, transforming it to a superposition of $|k'\rangle$ while leaving x_0 unchanged:

$$(\mathbb{1} \otimes Q_{M/2}) |x_0, x'\rangle = \frac{1}{\sqrt{M/2}} \sum_{k'=0}^{M/2-1} \omega_{M/2}^{k'x'} |x_0, k'\rangle .$$

We then apply a "twiddle operator" W, which applies a phase shift that depends on x_0 and k':

$$W |x_0, k'\rangle = \omega_M^{k'x_0} |x_0, k'\rangle .$$

We will discuss how to implement W shortly. Finally, we apply a Hadamard gate to the first qubit, transforming x_0 to k_0:

$$(H \otimes \mathbb{1}) |x_0, k'\rangle = \frac{1}{\sqrt{2}} \sum_{k_0=0,1} (-1)^{k_0 x_0} |k_0, k'\rangle .$$

Putting all this together, we have

$$\begin{aligned}
|x\rangle = |x_0, x'\rangle &\xrightarrow{\mathbb{1}\otimes Q_{M/2}} \frac{1}{\sqrt{M/2}} \sum_{k'=0}^{M/2-1} \omega_{M/2}^{k'x'} |x_0, k'\rangle \\
&\xrightarrow{W} \frac{1}{\sqrt{M/2}} \sum_{k'=0}^{M/2-1} \omega_{M/2}^{k'x'} \omega_M^{k'x_0} |x_0, k'\rangle \\
&\xrightarrow{H\otimes\mathbb{1}} \frac{1}{\sqrt{M/2}} \frac{1}{\sqrt{2}} \sum_{x_0=0,1} \sum_{k'=0}^{M/2-1} \omega_{M/2}^{k'x'} \omega_M^{k'x_0} (-1)^{k_0 x_0} |k_0, k'\rangle \\
&= \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} \omega_M^{kx} |k\rangle .
\end{aligned}$$

Thus we can write

$$Q_M = (H \otimes \mathbb{1}) \, W (\mathbb{1} \otimes Q_{M/2}) . \tag{15.34}$$

If we look back at Section 3.2.3, we see that each of these steps corresponds exactly to a step in the classical FFT. Applying $Q_{M/2}$ corresponds to Fourier transforming the sublists where x is odd or even—that is, where x_0 is 0 or 1. Here, using the power of quantum parallelism, we handle both these sublists at once. The operator W applies a phase shift $\omega_M^{k'}$ when x_0 is true, and this is exactly the "twiddle factor" appearing in (3.7) when x is odd. The same is true for the factor $(-1)^{k_0}$ induced by the Hadamard when x_0 is true.

Let's look more closely at the twiddle operator W. Since it has no effect unless $x_0 = 1$, we should think of it as a controlled phase-shift gate, or a series of such gates, with x_0 as their control bit. If we write k' in binary,

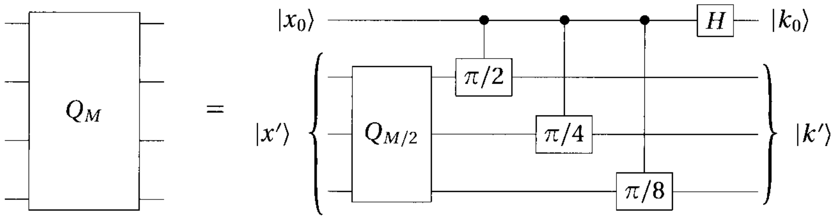
$$k' = 2^{m-2}k_1 + 2^{m-3}k_2 + \cdots + k_m = (M/4)k_1 + (M/8)k_2 + \cdots + k_{m-1},$$

we can write the phase shift as

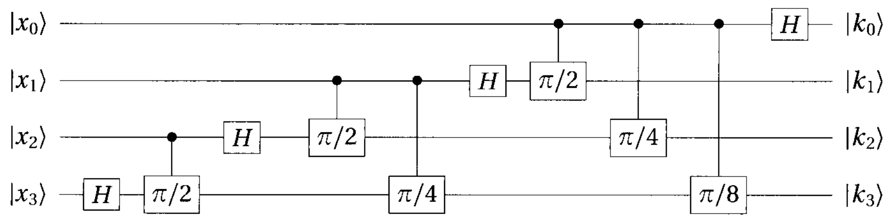
$$\omega_M^{k'x_0} = e^{\imath\pi k_1 x_0/2}\, e^{\imath\pi k_2 x_0/4} \cdots e^{2\imath\pi k_{m-1} x_0/M}.$$

Each one of these is a controlled phase shift gate as described in Section 15.2.7, which applies a phase shift of $e^{\imath\pi/2^j}$ if both x_0 and k_j are true.

Now that we know how to implement W, we can turn (15.34) into a recursive description of a circuit. For example, the QFT for $M = 2^m$ where $m = 4$ is given by



We unfold this recursively until we get to the base case $m = 1$ and $M = 2$, for which Q_2 is simply the Hadamard gate H. This gives



How many gates do these circuits have? Let $T(m)$ be the number of elementary gates in the circuit for Q_M where $M = 2^m$. We can see from this picture that

$$T(m) = T(m-1) + m.$$

Given the base case $T(1) = 1$, we have

$$T(m) = 1 + 2 + 3 + \cdots + m = O(m^2),$$

and the total number of gates is polynomial in the number of bits of M. To ensure that M is large enough for the continued fraction technique of Section 15.5.4 to work, we take M to be the smallest power of 2 greater than N^2. If N has n bits then $m \leq 2n$, and we can implement the QFT for $\mathbb{Z}_M$ with $O(n^2)$ gates.

Wrapping all this up, there is a randomized quantum algorithm for ORDER FINDING, and therefore, by the reductions in Section 15.5.2, for FACTORING. This algorithm succeeds after a polynomial number of trials, and each one takes a polynomial number of elementary quantum operators.

Let's define a quantum complexity class, analogous to the class BPP we defined for randomized computation in Section 10.9:

> BQP, for "bounded-error quantum polynomial time," is the class of problems that can be solved by polynomial-time quantum algorithms that give the right answer at least 2/3 of the time.

Then we have shown that

FACTORING is in BQP.

On a practical level—where "practical" is defined rather broadly—this means that quantum computers can break the leading modern cryptosystem. More importantly, it shows that quantum physics helps us solve one of the oldest and most fundamental problems in mathematics—a problem that has challenged us for thousands of years. 15.17

15.5.6 *Key Exchange and* DISCRETE LOG

Don't let him know she liked them best,
For this must ever be
A secret, kept from all the rest,
Between yourself and me.

Lewis Carroll, *Alice in Wonderland*

Shor discovered a quantum algorithm for another important number-theoretic problem, DISCRETE LOG. Like his algorithm for FACTORING, it uses the quantum Fourier transform as its main algorithmic tool, and we present it here. But first, let's describe a clever cryptographic protocol for having a private conversation even when anyone can overhear, and how solving DISCRETE LOG would let an eavesdropper break it.

One day, Alice calls Bob on the phone. "Hello!" she says. "I have something important to tell you, and I'm worried someone might be listening in. Shall we speak Navajo?"

"Good idea!" Bob replies. "But wait... whoever is listening just heard us say that. What if they speak Navajo too?" It seems impossible for Alice and Bob to agree on a secret key that will allow them to communicate privately, if the only communication channel they have is public. But in fact there is a way they can do this, if we assume that certain functions are hard to invert.

Suppose that Alice and Bob have publicly agreed on an n-bit prime p and a primitive root $a \in \mathbb{Z}_p^*$, i.e., an a such that every element of $\mathbb{Z}_p^*$ can be written $a^x \bmod p$ for some x. Alice picks x and Bob picks y, where x and y are uniformly random n-bit integers that they keep to themselves. Alice sends $a^x \bmod p$ to Bob, and Bob sends $a^y \bmod p$ to Alice. They can compute these in polynomial time using the algorithm of Section 3.2.2 for MODULAR EXPONENTIATION.

Now that she knows a^y, Alice can raise it to the xth power, and get $a^{xy} \bmod p$. Similarly, Bob can raise a^x to the yth power, and get the same thing. Now that both of them have $a^{xy} \bmod p$, they can use it as a secret key to encrypt the rest of their conversation—say, by XORing its bits with each n-bit packet they send each other. This trick is called *Diffie–Hellman key exchange.*

Now suppose that Eve has been listening in all this time. Eve knows a^x and a^y, as well as a and p. She is faced with the following problem: 15.18

BREAKING KEY EXCHANGE

Input: A n-bit prime p, a primitive root a, and $a^x, a^y \in \mathbb{Z}_p^*$

Output: a^{xy}

Eve could solve this problem if she could determine x and y from a^x and a^y—that is, if she could determine their logarithms base a in $\mathbb{Z}_p^*$. This is the DISCRETE LOG problem, which we first encountered in Section 3.2.2. This gives the reduction

$$\text{BREAKING KEY EXCHANGE} \leq \text{DISCRETE LOG}.$$

Note that since x and y are chosen uniformly, in order for our key exchange scheme to be secure we need DISCRETE LOG to be hard almost all the time, rather than just in the worst case. We made a similar assumption in Sections 11.1.4 and 11.4.3, where we used DISCRETE LOG in protocols for bit commitment and pseudorandom number generation.

Given everything we've seen so far, the quantum algorithm for DISCRETE LOG is easy to describe. Given g and a, we want to find the x such that $g = a^x$. We define a function of two variables,

$$f(s,t) = g^s a^{-t} = a^{xs-t}.$$

The exponents s,t belong to the group $\mathbb{Z}_{p-1}$. We create a uniform superposition over all pairs s,t,

$$\frac{1}{p-1}\sum_{s,t=0}^{p-2} |s,t\rangle$$

Calculating and measuring $f(s,t)$ collapses this state to the uniform superposition over the pairs s,t such that $f(s,t) = a^{t_0}$ for some t_0. Since a is a primitive root, a^x is a one-to-one function from $\mathbb{Z}_{p-1}$ to $\mathbb{Z}_p^*$. So these are the $p-1$ pairs such that $xs - t = t_0$, and writing $t = xs - t_0$ we have

$$|\psi\rangle = \frac{1}{\sqrt{p-1}}\sum_{s=0}^{p-2} |s, xs - t_0\rangle.$$

These pairs form a line in the s,t plane with slope x and intercept $-t_0$. Our goal is to obtain the slope x.

Once again, we apply the quantum Fourier transform, but now a two-dimensional one. Let Q_{p-1} denote the QFT on $\mathbb{Z}_{p-1}$. If we apply $Q_{p-1} \otimes Q_{p-1}$, the amplitude of each frequency vector $|k,\ell\rangle$ is

$$\tilde{a}_{k,\ell} = \frac{1}{p-1}\sum_{s,t=0}^{p-2} \omega^{(k,\ell)\cdot(s,t)} a_{s,t} = \frac{\omega^{-\ell t_0}}{(p-1)^{3/2}} \sum_{s=0}^{p-2} \left(\omega^{(k,\ell)\cdot(1,x)}\right)^s,$$

where $\omega = e^{2\imath\pi/(p-1)}$ and $(k,\ell)\cdot(s,t) = ks + \ell t$. The probability $P(k,\ell)$ that we observe a given frequency vector is then

$$P(k,\ell) = \left|\tilde{a}_{k,\ell}\right|^2 = \frac{1}{(p-1)^3}\left|\sum_{s=0}^{p-2}\left(\omega^{(k,\ell)\cdot(1,x)}\right)^s\right|^2 = \begin{cases} \frac{1}{p-1} & \text{if } (k,\ell)\cdot(1,x) \equiv_{p-1} 0 \\ 0 & \text{otherwise}. \end{cases}$$

In Simon's algorithm, the observed frequency vector **k** was chosen uniformly from the subspace perpendicular to **y**. Now (k,ℓ) is chosen randomly from the $p-1$ vectors that are perpendicular, mod $p-1$, to $(1,x)$. Since

$$(k,\ell)\cdot(1,x) = k + \ell x \equiv_{p-1} 0,$$

if ℓ is invertible in $\mathbb{Z}_{p-1}$, we can write x as

$$x = -k\ell^{-1} \bmod p-1.$$

We can find ℓ^{-1} in polynomial time using Euclid's algorithm as in Problem 2.2, so we can determine x as soon as we observe a frequency vector (k,ℓ) where ℓ is invertible.

Since ℓ is uniformly random (exercise!) the probability that it is invertible is $\varphi(p-1)/(p-1)$. By (15.32) this probability is $\Omega(1/n)$, and in fact it is $\Omega(1/\log n)$ by (15.33). Thus it takes just $O(\log n)$ trials to find such an invertible ℓ. We skip the details of implementing Q_{p-1}, but we can do so to high accuracy with poly(n) elementary gates. Thus DISCRETE LOG, and therefore BREAKING KEY EXCHANGE, are in BQP.

Like RSA encryption, Diffie–Hellman key exchange will cease to be secure whenever quantum computers are built. In addition, the pseudorandom number generator described in Section 11.4.1 may no longer be secure, in the sense that a quantum computer might be able to distinguish its output from a string of truly random numbers.

15.6 Graph Isomorphism and the Hidden Subgroup Problem

We have seen two problems, FACTORING and DISCRETE LOG, that quantum computers can solve in polynomial time but that, as far as we know, classical computers cannot. How much more powerful are quantum computers than classical ones? What other problems are in BQP that we believe are outside P?

Our understanding of quantum algorithms is at a very early stage. Nevertheless, as we will see in Section 15.7, it seems unlikely that they can efficiently solve NP-complete problems. Therefore our current guess is that BQP does not contain NP. If this is correct, then the best targets for quantum algorithms are problems that are "just outside" P—those which don't seem to be in P, but which don't seem to be NP-complete either.

As we discussed in Section 6.6, we know that, unless P = NP, there are an infinite number of problems that lie in this middle ground. However, there are relatively few "naturally occurring" candidates. Besides FACTORING and DISCRETE LOG, one of the few others is GRAPH ISOMORPHISM, the problem of telling whether two graphs are topologically identical. We repeat its definition here:

GRAPH ISOMORPHISM

Input: Two graphs $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$

Question: Is there a permutation $\pi : V_1 \to V_2$ such that $\pi(G_1) = G_2$, i.e., $(u,v) \in E_1$ if and only if $(\pi(u), \pi(v)) \in E_2$?

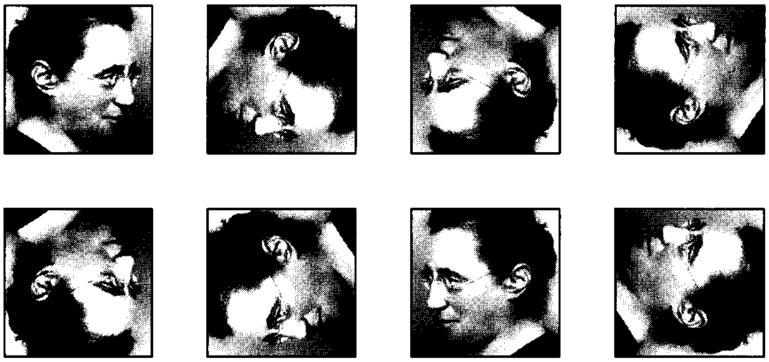


FIGURE 15.6: The rotations and reflections of a square form a group with 8 elements. Pictured is Emmy Noether, who established a deep relationship between symmetries and conservation laws in physics.

If such a permutation exists, we write $G_1 \cong G_2$.

We showed in Section 11.1 that GRAPH ISOMORPHISM is in the class $\mathsf{NP} \cap \mathsf{coAM}$, since GRAPH NONISOMORPHISM has a simple interactive proof. Thus we believe that GRAPH ISOMORPHISM is not NP-complete. However, while many special cases of GRAPH ISOMORPHISM are in P, such as planar graphs and graphs of constant degree, there is no known polynomial-time algorithm that works for general graphs.

At first blush, GRAPH ISOMORPHISM seems very different from number-theoretic problems like FACTORING and DISCRETE LOG. 15.19 However, Simon's and Shor's algorithms work by detecting the periodicities or symmetries of a function. We will show in this section that GRAPH ISOMORPHISM can, in principle, be solved the same way, by treating it as an instance of a more general problem called HIDDEN SUBGROUP. While this has not yet led to an efficient quantum algorithm for GRAPH ISOMORPHISM, it has produced some very interesting results along the way. We will use a little bit of group theory—if this is new to you, we recommend Appendix A.7 for a brief introduction.

15.6.1 Groups of Symmetries

*H*s are never upside down, except when they're sideways.

Rosemary Moore, age 4

When we say that an object is symmetric, we mean that there are transformations, or *automorphisms*, that leave it unchanged. For instance, if we rotate a square by $\pi/2$ or reflect it around one of its axes, it looks just the same as it did before. The inverse of an automorphism or the composition of two automorphisms is an automorphism, so the set of automorphisms forms a group. In this example, the automorphism group of the square has a total of 8 elements as shown in Figure 15.6, and is called the *dihedral group* (see Appendix A.7).

Similarly, when we say that a function $f(x)$ defined on a group G is symmetric, we mean that $f(x)$ stays the same if we transform x in some way. If f is periodic, shifting x by some h leaves $f(x)$ unchanged.

The set of all such h forms a subgroup,

$$H = \{h \in G : f(x) = f(x+h) \text{ for all } x \in G\}.$$

In Simon's algorithm, $G = \mathbb{Z}_2^n$ and $H = \{\mathbf{0}, \mathbf{y}\}$. In Shor's algorithm, $G = \mathbb{Z}_M$ and H consists of the multiples of r, assuming that r divides M.

To include *nonabelian* groups where the order of multiplication matters, we can shift x by multiplying on the right by an element h. We then define the *automorphism group* of f as the set of h such that this leaves $f(x)$ unchanged:

$$H = \{h \in G : f(x) = f(xh) \text{ for all } x \in G\}. \tag{15.35}$$

As in Simon's and Shor's algorithms, we assume that $f(x) = f(x')$ if and only if x and x' differ by an element of h. Then our goal is to determine H from a small number of queries to f. We can phrase this as the following problem:

HIDDEN SUBGROUP

Input: A function $f : G \to S$ with the property that there is a subgroup $H \subseteq G$ such that $f(x) = f(x')$ if and only if $x' = xh$ for some $h \in H$

Output: The subgroup H

All the exponential speedups in quantum algorithms that we have seen so far work by solving cases of HIDDEN SUBGROUP. Moreover, they do this with only a polynomial number of queries, even though groups like $\mathbb{Z}_M$ or $\mathbb{Z}_2^n$ are exponentially large.

Exercise 15.20 *State Shor's algorithm for* DISCRETE LOG *as a case of* HIDDEN SUBGROUP. *What is G, what is f, and what is H?*

How can we state GRAPH ISOMORPHISM in terms of symmetry? Let S_n denote the group of all $n!$ permutations of n objects. If Γ is a graph with n vertices, then as in Section 11.1.2 its automorphism group $\text{Aut}(\Gamma)$ is the subgroup of S_n consisting of permutations of its vertices that leave it unchanged,

$$\text{Aut}(\Gamma) = \{\sigma \in S_n : \sigma(\Gamma) = \Gamma\}.$$

Now suppose we place G_1 and G_2 side-by-side as in Figure 15.7. What is the automorphism group of the combined graph $G_1 \cup G_2$? If $G_1 \ncong G_2$, its only symmetries are those which permute G_1 and G_2 separately, so

$$\text{Aut}(G_1 \cup G_2) = \text{Aut}(G_1) \times \text{Aut}(G_2).$$

On the other hand, if $G_1 \cong G_2$ then half the symmetries of $G_1 \cup G_2$ switch G_1 and G_2, so $\text{Aut}(G_1 \cup G_2)$ is twice as large as the subgroup $\text{Aut}(G_1) \times \text{Aut}(G_2)$. This gives

$$\frac{|\text{Aut}(G_1 \cup G_2)|}{|\text{Aut}(G_1)|\,|\text{Aut}(G_2)|} = \begin{cases} 1 & \text{if } G_1 \ncong G_2 \\ 2 & \text{if } G_1 \cong G_2. \end{cases}$$

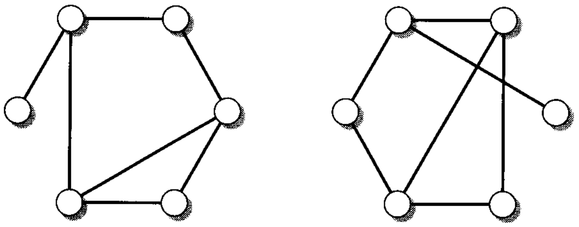


FIGURE 15.7: A pair of graphs G_1 and G_2. The automorphism group of the combined graph $G_1 \cup G_2$ depends on whether or not G_1 and G_2 are isomorphic.

Thus if we can determine the automorphism group of a graph Γ, or even estimate its size, we can solve GRAPH ISOMORPHISM. We can reduce this to HIDDEN SUBGROUP by defining a function f from S_n to the set of all graphs with n vertices, where $f(\sigma)$ is the result of permuting Γ with σ:

$$f(\sigma) = \sigma(\Gamma).$$

Then $f(\sigma\tau) = f(\sigma)$ if and only if $\tau(\Gamma) = \Gamma$, and f's automorphism group as defined in (15.35) is $\text{Aut}(\Gamma)$.

This shows that we can solve GRAPH ISOMORPHISM if we can solve HIDDEN SUBGROUP in the case where G is the permutation group S_n. Can we follow in Simon's and Shor's footsteps and find a quantum algorithm that does this?

15.6.2 Coset States and the Nonabelian Fourier Transform

In Simon's and Shor's algorithms, we start by creating a uniform superposition over the group G and querying f on this superposition of inputs. This gives the state

$$\frac{1}{\sqrt{|G|}} \sum_{x \in G} |x, f(x)\rangle.$$

We then measure $f(x)$ and observe some value f_0. This collapses the state to $|\psi\rangle \otimes |f_0\rangle$, where $|\psi\rangle$ is the uniform superposition over the set of x such that $f(x) = f_0$. According to the premise of HIDDEN SUBGROUP, the set of such x forms a *coset* of H, i.e., a set of the form $x_0 H = \{x_0 h : h \in H\}$ where $f(x_0) = f_0$. So in general we have

$$|\psi\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |x_0 h\rangle. \tag{15.36}$$

Since all values of f_0 are observed with equal probability, we can think of x_0 as being chosen uniformly. A state of the form (15.36) is called a *coset state*. In the case of GRAPH ISOMORPHISM, the hidden subgroup H is the automorphism group of $G_1 \cup G_2$, and all we need to do is determine H's size.

For simplicity, let's focus on the case where G_1 and G_2 are *rigid*, with no nontrivial symmetries of their own. That is, $\text{Aut}(G_1) = \text{Aut}(G_2) = \{1\}$ where 1 is the identity permutation. Then $H = \text{Aut}(G_1 \cup G_2)$ is either $\{1\}$ or $\{1, \tau\}$ where τ is a permutation that swaps the two graphs, and the coset state is

$$|\psi\rangle = \begin{cases} \frac{1}{\sqrt{2}}(|\sigma\rangle + |\sigma\tau\rangle) & \text{if } G_1 \cong G_2 \\ |\sigma\rangle & \text{if } G_1 \ncong G_2, \end{cases} \tag{15.37}$$

where $\sigma \in S_n$ is uniformly random. To solve GRAPH ISOMORPHISM, at least for rigid graphs, all we ask is the ability to distinguish between these two cases. What basis should we use to measure $|\psi\rangle$, and thus learn about H?

The Fourier basis, you say. Well, of course we agree. But what basis is that? On the integers or the reals, the Fourier transform measures the frequencies at which a function oscillates. But what do "frequency" and "oscillation" mean for functions defined on the set of all permutations?

Let's review the various Fourier transforms we have seen so far. On $\mathbb{Z}_2^n$, the Fourier basis functions are of the form

$$\phi_{\mathbf{k}}(\mathbf{x}) = (-1)^{\mathbf{k}\cdot\mathbf{x}} .$$

On $\mathbb{Z}_M$, they are

$$\phi_k(x) = \omega^{kx} \quad \text{where} \quad \omega = e^{2\imath\pi/M} .$$

What these have in common is that, for each frequency $\mathbf{k}$ or k, the function $\phi_k(x)$ is a *homomorphism* from the group into the complex numbers $\mathbb{C}$. That is,

$$\phi_k(x+y) = \phi_k(x)\phi_k(y) .$$

Both $\mathbb{Z}_2^n$ and $\mathbb{Z}_M$ are *abelian* groups, in which the binary operation is commutative: $x+y=y+x$. For any abelian group G, there are exactly $|G|$ such homomorphisms from G to $\mathbb{C}$, and they form an orthogonal basis for the vector space of all superpositions over G. It is the homomorphic nature of these basis functions that gives the Fourier transform all the properties that we know and love, such as the fact that the Fourier transform of the convolution of two functions is the product of their Fourier transforms.

The permutation group S_n, on the other hand, is nonabelian—the order of multiplication matters. If we swap $1 \longleftrightarrow 2$ and then $2 \longleftrightarrow 3$, we get the rotation $1 \longrightarrow 3 \longrightarrow 2$, but if do these things in the opposite order we get $1 \longrightarrow 2 \longrightarrow 3$. Since multiplication in $\mathbb{C}$ is commutative, any homomorphism from S_n to $\mathbb{C}$ must "forget" this kind of information.

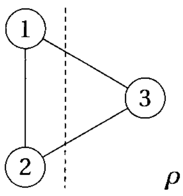
Moreover, the vector space in which $|\psi\rangle$ lives, of superpositions of elements of S_n, has dimension $|S_n| = n!$. So in order to define a Fourier transform, we need $n!$ different basis functions. But there are only two homomorphisms from S_n to $\mathbb{C}$: the trivial one which sends every element to 1, and the parity that sends even and odd permutations to $+1$ and -1 respectively.

To define a Fourier basis for a nonabelian group G, we need to go beyond homomorphisms from G to $\mathbb{C}$, and instead look at homomorphisms from G to the group U_d of d-dimensional unitary matrices. Such homomorphisms are called *representations* of G, and the Fourier basis functions we are used to in the abelian case are just the special case where $d = 1$.

For example, suppose that $G = S_3$. There are two one-dimensional representations, namely the trivial one and the parity. But there is also a two-dimensional representation, which permutes the three corners of a triangle by rotating and reflecting the plane as shown in Figure 15.8.

Representations like these give a geometric picture of the group in terms of rotations and reflections in some—possibly high-dimensional—space. For another example, consider Figure 15.9. For every even permutation σ of the 5 colors, there is a rotation that maps the 5 tetrahedra onto each other according to σ. This gives a three-dimensional representation of A_5, the subgroup of S_5 consisting of permutations with even parity.

While it would take us too far afield, there is a beautiful theory of Fourier transforms for nonabelian groups, in which representations play the role of frequencies. Specifically, each basis vector $|\rho, i, j\rangle$ corresponds to the i,j entry of some representation ρ. For instance, for S_3 we have $3! = 6$ basis vectors,



$$\rho(1) = \begin{pmatrix} 1 & \\ & 1 \end{pmatrix}$$

$$\rho(1 \longleftrightarrow 2) = \begin{pmatrix} 1 & \\ & -1 \end{pmatrix}$$

$$\rho(1 \to 2 \to 3 \to 1) = \begin{pmatrix} -1/2 & -\sqrt{3}/2 \\ \sqrt{3}/2 & -1/2 \end{pmatrix}$$

FIGURE 15.8: The two-dimensional representation of S_3. For the identity permutation, it gives the identity matrix; for the permutation that swaps 1 and 2, it reflects around the x-axis; and for the cyclic permutation $1 \to 2 \to 3 \to 1$, it rotates the plane counterclockwise by $2\pi/3$.

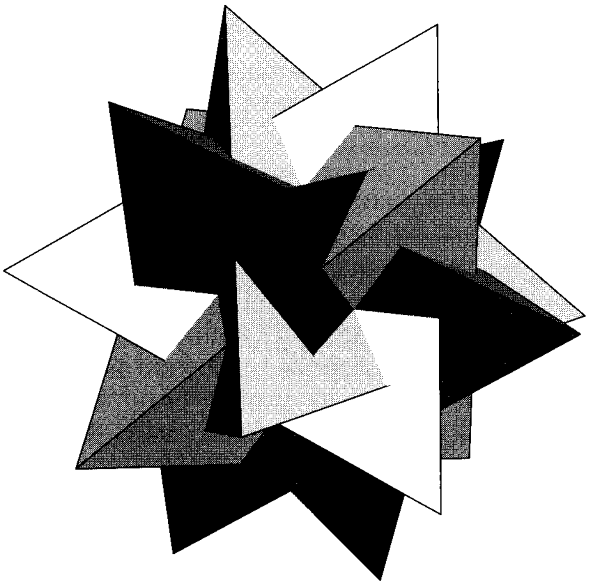


FIGURE 15.9: A three-dimensional representation of A_5, the group of even-parity permutations of 5 objects. See if you can tell how to switch two pairs of colors simultaneously.

corresponding to the trivial representation, the parity, and the four matrix entries of the two-dimensional representation. For many groups, including S_n, it is even possible to carry out the QFT efficiently, using nonabelian versions of the FFT and "quantizing" them in ways analogous to Section 15.5.5.

Sadly, all this machinery doesn't give us what we want. While the general approach of generating a coset state and measuring it in the Fourier basis does solve the HIDDEN SUBGROUP problem for some families of nonabelian groups, it fails for S_n. The situation is even worse than that. It can be shown that there is no measurement at all that we can perform on the coset state (15.37) that will distinguish an isomorphic pair of graphs from a non-isomorphic pair. Specifically, no matter what measurement we perform, the probability distribution of the outcomes we get in these two cases are exponentially close, so it would take an exponential number of experiments to distinguish them.

Some hope yet remains. It is known that if we have many copies of the coset state—that is, the tensor product of many $|\psi\rangle$s, where each one is a superposition over a randomly chosen coset—then there is a measurement that tells us whether the two graphs are isomorphic or not. However, this measurement must be highly entangled. In other words, rather than an independent series of measurements on the $|\psi\rangle$s, it is a complicated joint measurement along a basis where each basis vector corresponds to an entangled state.

However, while we can write this measurement down mathematically, we do not know how, or if, it can be carried out efficiently, i.e., by a quantum circuit with a polynomial number of gates. At the time we write this, nearly every proposed family of algorithms has been proved to fail. Our current intuition is that, if there is a quantum algorithm for GRAPH ISOMORPHISM, it doesn't work by reducing to HIDDEN SUBGROUP first.
15.20

15.6.3 *Quantum Sampling and the Swap Test*

There is another possible approach to GRAPH ISOMORPHISM. Suppose that, given a graph G with n vertices, we can generate the uniform superposition over all graphs G' such that $G' \cong G$, or equivalently over all the graphs we can get by permuting the vertices of G. If there are M such graphs, this is

$$|\psi(G)\rangle = \frac{1}{\sqrt{M}} \sum_{G' \cong G} |G'\rangle = \frac{\sqrt{M}}{n!} \sum_{\sigma \in S_n} |\sigma(G)\rangle\,. \tag{15.38}$$

For two graphs G_1, G_2, the states $|\psi(G_1)\rangle$ and $|\psi(G_2)\rangle$ are identical if $G_1 \cong G_2$, and orthogonal if $G_1 \ncong G_2$.

To distinguish between these two cases, we use a variant of phase kickback called the *swap test*. Given two quantum states $|\psi_1\rangle$ and $|\psi_2\rangle$, we want to tell whether $\langle\psi_1|\psi_2\rangle$ is 1 or 0, or more generally to estimate $\left|\langle\psi_1|\psi_2\rangle\right|^2$. We start with their tensor product, along with a qubit in the state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$:

$$|+\rangle \otimes |\psi_1\rangle \otimes |\psi_2\rangle\,.$$

We now apply a *controlled swap gate*, which swaps $|\psi_1\rangle$ and $|\psi_2\rangle$ if the first qubit is true, and otherwise leaves them alone. This gives

$$\frac{1}{\sqrt{2}}\big(|0\rangle \otimes |\psi_1\rangle \otimes |\psi_2\rangle + |1\rangle \otimes |\psi_2\rangle \otimes |\psi_1\rangle\big)\,.$$

Finally, we measure the qubit in the X-basis. Problem 15.33 shows that we observe $|+\rangle$ or $|-\rangle$ with probabilities

$$P(+) = \frac{1 + \left|\langle\psi_1|\psi_2\rangle\right|^2}{2} \quad \text{and} \quad P(-) = \frac{1 - \left|\langle\psi_1|\psi_2\rangle\right|^2}{2}. \tag{15.39}$$

Thus if $|\psi_1\rangle$ and $|\psi_2\rangle$ are identical, we always observe $|+\rangle$, but if they are orthogonal we observe $|+\rangle$ or $|-\rangle$ with equal probability. If we can generate the states $|\psi_1\rangle = |\psi(G_1)\rangle$ and $|\psi_2\rangle = |\psi(G_2)\rangle$, we can use this swap test to tell whether G_1 and G_2 are isomorphic or not.

Given a graph G, can we generate $|\psi(G)\rangle$? In the classical case, it's easy to generate a uniform distribution over all the graphs isomorphic to G—just generate a random permutation and apply it to G's vertices. Similarly, in the quantum setting, we can create a uniform superposition over S_n, and then apply a controlled-permutation operator to G. However, this generates the entangled state

$$\frac{1}{n!} \sum_{\sigma \in S_n} |\sigma\rangle \otimes |\sigma(G)\rangle .$$

To obtain $|\psi(G)\rangle$ we have to *disentangle* $\sigma(G)$ from σ. To do this we have to "uncompute" σ from $\sigma(G)$, telling what permutation takes us from G to a given $\sigma(G)$. But this seems to be just as hard as solving GRAPH ISOMORPHISM in the first place.

There is an interesting generalization of this problem, called *quantum sampling*. Given a probability distribution $P(x)$, such as the equilibrium distribution of a rapidly mixing Markov chain, can we generate the quantum state $\sum_x \sqrt{P(x)}\,|x\rangle$? In the classical case, if P_1 and P_2 are probability distributions over an exponentially large state space, we have no idea how to tell whether they are identical or disjoint—but if we can generate the analogous quantum states, the swap test lets us do this. 15.21

15.7 Quantum Haystacks: Grover's Algorithm

Shor's algorithm is a spectacular example of the power of quantum computing. But FACTORING and DISCRETE LOG seem rather special, in that they are reducible to finding the periodicity of a number-theoretic function. Is there a generic way in which quantum computers can solve search problems more quickly than classical ones? For instance, suppose there is a needle hidden in a haystack of size N. Classically, we have to look at all N blades of hay. Can a quantum computer do better?

In 1996, Lov Grover discovered a quantum algorithm that finds the needle with just $O(\sqrt{N})$ queries. While this improvement is merely quadratic, it makes an enormous difference if N is exponentially large. Imagine that we have a SAT formula with 80 variables and a unique satisfying assignment. If we compare Grover's algorithm with a brute-force search, the number of truth assignments we have to try decreases from 2^{80} to 2^{40}. At a rate of 10^9 assignments per second, this reduces our running time from 38 million years to 18 minutes.

On the other hand, changing N to $\sqrt{N}$ can't change an exponential to a polynomial—it can only improve the exponent. Moreover, as we will see in this section, Grover's algorithm is optimal. This means that if we want to solve an NP-complete problem in polynomial time, we have to find something about the specific structure of that problem that a quantum algorithm can exploit. Just as with classical computation, our current belief is that NP-complete problems are just as hard as haystacks, and that even quantum computers cannot solve them in polynomial time. In terms of complexity classes, we believe that $\mathsf{NP} \not\subseteq \mathsf{BQP}$.

15.7.1 *Rotating Towards the Needle*

Suppose there is a haystack with N locations—in quantum terms, a state space with N basis vectors $|j\rangle$ where $1 \le j \le N$. There is a single needle hidden at some location i, and we want to find it. We can query the haystack by looking at a location j and seeing if the needle is there. As in Section 15.4, each such query corresponds to applying a unitary operator that flips a qubit y if $j = i$:

$$|j\rangle \otimes |y\rangle \to \begin{cases} |j\rangle \otimes X|y\rangle & \text{if } j = i \\ |j\rangle \otimes |y\rangle & \text{if } j \ne i\,. \end{cases}$$

By preparing y in the eigenvector $|-\rangle$ we can use the phase kickback trick, and treat each query as this N-dimensional operator instead:

$$U|j\rangle = \begin{cases} -|j\rangle & \text{if } j = i \\ |j\rangle & \text{if } j \ne i\,. \end{cases}$$

This operator is almost identical to the identity, differing only at $U_{ii} = -1$.

How many queries do we need to find the needle? In other words, how many times do we have to apply U in order to obtain a state for which some measurement will tell us what i is? Grover's clever idea is to adapt, after each query, the superposition of locations at which we query the haystack. We do this by applying a "diffusion operator" D which has $(2/N) - 1$ on the diagonal and $2/N$ everywhere else:

$$D = \frac{2}{N}\begin{pmatrix} 1 & 1 & \cdots \\ 1 & 1 & \\ \vdots & & \ddots \end{pmatrix} - \mathbb{1}\,. \tag{15.40}$$

It is not obvious at first that D is unitary, so we recommend the following exercise.

Exercise 15.21 *Show that Grover's operator D is unitary, and that $D^2 = \mathbb{1}$.*

The algorithm starts in the uniform superposition over all N states, $|u\rangle = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} |j\rangle$. We then alternately apply U and D. After t iterations, we have the state

$$|\psi\rangle = \underbrace{DU\,DU \cdots DU}_{t \text{ times}}|u\rangle = (DU)^t\,|u\rangle\,.$$

Marvelously, alternating between U and D causes the amplitude at the needle's location $|i\rangle$ to interfere constructively, while the amplitude at all other locations cancels out. After just $t = O(\sqrt{N})$ queries, if we measure $|\psi\rangle$ we observe $|i\rangle$ with high probability.

How does this work? Consider Figure 15.10, where we reflect a state $|\psi\rangle$ first around a vector $|u\rangle$ and then around another vector $|v\rangle$. The composition of these two reflections is a rotation. Specifically, if the angle between $|u\rangle$ and $|v\rangle$ is θ, this rotates $|\psi\rangle$ by 2θ.

For any vector $|u\rangle$ of unit length, the operator R_u that reflects around $|u\rangle$ can be written

$$R_u = 2|u\rangle\langle u| - \mathbb{1}\,. \tag{15.41}$$

That is, R_u negates the component perpendicular to $|u\rangle$ and keeps the component parallel to $|u\rangle$ the same. To see this, try the following exercises.

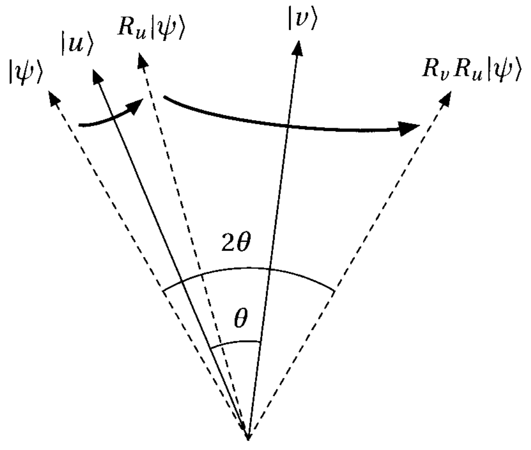


FIGURE 15.10: If we reflect $|\psi\rangle$ around two vectors, $|u\rangle$ and then $|v\rangle$, we rotate it by twice the angle θ between them.

Exercise 15.22 *Show that if* $|\psi\rangle = a|u\rangle + b|w\rangle$ *where* $\langle u \,|\, w\rangle = 0$, *then*

$$R_u|\psi\rangle = a|u\rangle - b|w\rangle ,$$

and that R_u *is unitary.*

Looking again at (15.40), we see that D is just the reflection operator around the uniform superposition $|u\rangle$,

$$D = 2|u\rangle\langle u| - \mathbb{1} = R_u ,$$

since $|u\rangle\langle u|$ is the N-dimensional matrix where every entry is $1/N$. As Problem 15.35 shows, this description also helps us implement D efficiently using elementary quantum gates. Similarly, up to a sign, the query operator reflects around the basis vector $|i\rangle$ corresponding to the needle's location:

$$U = \mathbb{1} - 2|i\rangle\langle i| = -R_i . \tag{15.42}$$

What happens when we iteratively apply U and D? While Grover's algorithm takes place in an N-dimensional space, the state $|\psi\rangle$ always lies in a two-dimensional subspace—namely, the space of vectors in which all the $|j\rangle$s other than $|i\rangle$ have the same amplitude. Let's call this subspace V. It is spanned by $|i\rangle$ and $|u\rangle$, or equivalently by $|i\rangle$ and the uniform superposition over all the states other than $|i\rangle$, which we denote $|v\rangle$:

$$|v\rangle = \frac{1}{\sqrt{N-1}} \sum_{j \neq i} |j\rangle = \frac{1}{\sqrt{N-1}} \left(\sqrt{N}|u\rangle - |i\rangle \right) .$$

Since $|i\rangle$ and $|v\rangle$ are perpendicular, we can write the projection operator onto V as $\mathbb{1}_V = |i\rangle\langle i| + |v\rangle\langle v|$. As far as vectors in V are concerned, we can replace $\mathbb{1}$ with $\mathbb{1}_V$ in (15.42), giving

$$U = |v\rangle\langle v| - |i\rangle\langle i| = 2|v\rangle\langle v| - \mathbb{1} = R_v .$$

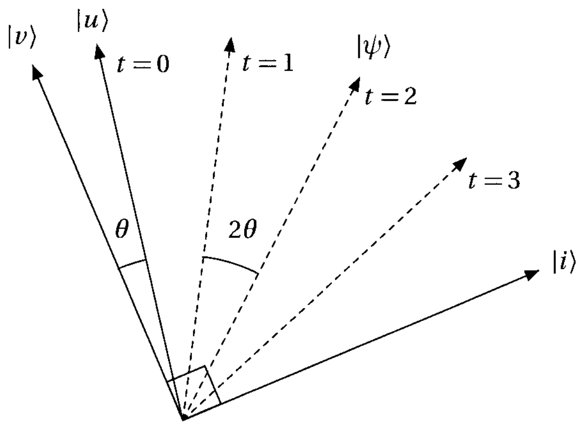


FIGURE 15.11: Rotating $|\psi\rangle$ from the initial uniform state $|u\rangle$ towards the state $|i\rangle$ marking the needle's location.

Thus each iteration of DU rotates $|\psi\rangle$ by 2θ, where θ is the angle between $|u\rangle$ and $|v\rangle$. As shown in Figure 15.11, our goal is to rotate $|\psi\rangle$ so that it is close to $|i\rangle$. Initially $|\psi\rangle = |u\rangle$, which is close to $|v\rangle$ when N is large. The angle between $|v\rangle$ and $|i\rangle$ is $\pi/2$, so we need $t = (\pi/2)/(2\theta) = \pi/(4\theta)$ steps to bring $|\psi\rangle$ close to $|i\rangle$. The fact that $|\psi\rangle$ starts at $|u\rangle$ instead of $|v\rangle$ means that the initial angle is actually $\pi/2-\theta$, but this only saves us half a step.

All that remains is to calculate the angle θ between $|u\rangle$ and $|v\rangle$. We can obtain θ from their inner product,

$$\langle u\,|\,v\rangle = \cos\theta = \sqrt{1-\frac{1}{N}} = 1-\frac{1}{2N} - O(N^{-2}).$$

Matching this with the Taylor series $\cos\theta = 1-\theta^2/2+O(\theta^4)$ then gives

$$\theta = \frac{1}{\sqrt{N}} + O(N^{-3/2}),$$

and so

$$t = \frac{\pi}{4\theta} = \frac{\pi}{4}\sqrt{N} - O(N^{-1/2}).$$

If we round t to the nearest integer, $|\psi\rangle$ will be within an angle θ of $|i\rangle$, so the probability of observing i will be

$$P(i) = \left|\langle\psi\,|\,i\rangle\right|^2 \geq \cos^2\theta = 1-O(\theta^2) = 1-O(1/N).$$

So if we perform $t = (\pi/4)\sqrt{N}$ iterations of DU and then measure $|\psi\rangle$, we observe the basis vector $|i\rangle$ with high probability. Note that if we run the algorithm for too many steps, $|\psi\rangle$ rotates back down to a uniform state, and then round and round again.

Exercise 15.23 *We assumed here that there is a unique solution to our search problem—or that there is exactly one needle. Show that if there are x needles, θ is given by*

$$\cos\theta = \sqrt{1-\frac{x}{N}}\,,$$

and therefore that we can find a solution, where all solutions are equally likely, after $O(\sqrt{N/x})$ queries.

Exercise 15.24 *Show that if $N = 4$ and there is a unique solution, or more generally if $x/N = 1/4$, Grover's algorithm finds a solution with probability 1 after just one iteration.*

Exercise 15.23 shows that the number of times we should iterate Grover's algorithm before measuring $|\psi\rangle$ depends on the number of needles in the haystack. But what if we don't know how many there are? In Problem 15.37, we modify Grover's algorithm so that it can deal with an unknown number of solutions.

Grover's algorithm is wonderful. But is it the best possible? Can we find the needle with even fewer queries, say N^α for some $\alpha < 1/2$? In the next section we will see that this is impossible—that where haystacks are concerned, a quadratic speedup is the best that quantum computers can achieve.

15.7.2 Entangling with the Haystack

There are many proofs that Grover's algorithm is optimal, i.e., that any algorithm needs to query the haystack $\Omega(\sqrt{N})$ times to find the needle. We focus here on a proof that is especially enlightening. In addition to giving us the result we want, it will let us explore the nature of entanglement more deeply.

If the haystack consists of a physical system with which our quantum queries can interact, it must be a quantum system. This means that it could be in a superposition of many different states, corresponding to different locations of the needle. To find the needle, our computer must measure the haystack—but this means that the computer and the haystack must become entangled.

In Section 10.5, we proved lower bounds on classical randomized algorithms by letting the adversary choose the worst possible probability distribution of instances. In the same way, we can prove lower bounds on quantum algorithms by letting him choose the worst possible *superposition* of instances. If we can show that some superposition requires t queries to solve, then t is a lower bound on the worst case as well.

To make this idea work, we endow the haystack with an N-dimensional state space. If the haystack is in state $|i\rangle$, the needle is at location i. Intuitively, the needle is hardest to find when every location has the same amplitude, so we will use the uniform superposition $|u\rangle = \frac{1}{\sqrt{N}}\sum_i |i\rangle$ to prove our lower bound.

For simplicity we assume that the computer's state space is also N-dimensional, with one state for each possible location. We also assume that the algorithm works perfectly, so that after running the algorithm the computer is in precisely the state $|i\rangle$. Both of these assumptions can be relaxed without changing the proof very much.

At the beginning of the algorithm, the computer is in some initial state $|\psi_0\rangle$. The computer and the haystack are unentangled, so their joint state is a tensor product,

$$|\Psi_{\text{init}}\rangle = |u\rangle \otimes |\psi_0\rangle\,.$$

As the algorithm proceeds, the state of the computer comes to depend on the state of the haystack, so their joint state becomes entangled. If $|\psi_i\rangle$ is the computer's state in the case where the haystack's state is $|i\rangle$, their joint state is a sum of tensor products,

$$\frac{1}{\sqrt{N}}\sum_i |i\rangle \otimes |\psi_i\rangle \,. \tag{15.43}$$

At the end of the algorithm we have $|\psi_i\rangle = |i\rangle$, so their final joint state is

$$|\Psi_{\text{final}}\rangle = \frac{1}{\sqrt{N}}\sum_i |i\rangle \otimes |i\rangle \,.$$

Like the entangled state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ shared by Alice and Bob in Section 15.3, the state $|\Psi_{\text{final}}\rangle$ is *maximally entangled*—the two systems are always in the same state. Our goal is to prove an upper bound on the amount by which each query can increase the entanglement, and therefore a lower bound on the number of queries we need to get from the unentangled state $|\Psi_{\text{init}}\rangle$ to $|\Psi_{\text{final}}\rangle$.

How can we quantify the amount of entanglement between two systems? The idea is that measuring one of them also measures the other to some extent. We can track this process using the density matrix formalism of Section 15.3.3, by calculating the mixed state ρ that one system will be in if we measure the other. If they are only slightly entangled, then ρ is close to a pure state. However, if they are maximally entangled like $|\Psi_{\text{final}}\rangle$ or Alice and Bob's pair of qubits, then measuring one measures the other completely. If all states are equally likely then ρ is the completely mixed state, i.e., the classical uniform distribution.

Recall that for a pure state $|v\rangle$, the density matrix is $|v\rangle\langle v|$. Its diagonal entries $\rho_{ii} = v_i v_i^* = |v_i|^2$ are the probabilities associated with $|v\rangle$'s components, while its off-diagonal entries $\rho_{ij} = v_i v_j^*$ contain information about their relative phase. As entanglement increases, the off-diagonal entries of ρ disappear, leaving only the diagonal ones. This process, called *decoherence*, removes all the quantum phase information, until only classical probabilities remain.

Problem 15.39 shows how to calculate the density matrix whenever the joint state is of the form (15.43). In our case, the density matrix of the haystack is

$$\rho_{ij} = \frac{1}{N}\langle \psi_j | \psi_i \rangle \,. \tag{15.44}$$

Note that ρ is unchanged by any step of the algorithm that doesn't make a query, since unitary operations on the computer's state space preserve the inner products $\langle \psi_j | \psi_i \rangle$. At the beginning of the algorithm we have $|\psi_i\rangle = |\psi_0\rangle$ for all i, and the haystack is in the pure state

$$\rho = |u\rangle\langle u| = \frac{1}{N}\begin{pmatrix} 1 & 1 & \cdots \\ 1 & 1 & \\ \vdots & & \ddots \end{pmatrix} . \tag{15.45}$$

As the computer learns more about the haystack, ψ_i and ψ_j become different for $i \neq j$, and the off-diagonal entries ρ_{ij} decrease. By the end of the algorithm, $|\psi_i\rangle$ and $|\psi_j\rangle$ are orthogonal for $i \neq j$, and the haystack is in the completely mixed state

$$\rho = \frac{1}{N}\begin{pmatrix} 1 & 0 & \cdots \\ 0 & 1 & \\ \vdots & & \ddots \end{pmatrix} = \frac{1}{N}\mathbb{1} \,. \tag{15.46}$$

How does each query change the computer's state $|\psi_i\rangle$, and therefore the haystack's density matrix ρ? The query operator U is now a unitary operator acting on the joint state. Specifically,

$$U\big(|i\rangle \otimes |\psi_i\rangle\big) = |i\rangle \otimes U_i |\psi_i\rangle ,$$

where U_i is the query operator from (15.42),

$$U_i = \mathbb{1} - 2|i\rangle\langle i| .$$

The reader might ask whether some other query operator would allow us to find the needle more quickly. However, since this operator imposes the largest possible phase shift when the computer looks at the right location—namely, -1, or an angle π in the complex plane—it can be shown to be optimal.

Applying (15.44), we see that U changes the haystack's density matrix from ρ to ρ' where

$$\begin{aligned}
\rho'_{ij} &= \frac{1}{N}\langle U_j \psi_j \,|\, U_i \psi_i\rangle \\
&= \frac{1}{N}\big(\langle\psi_j| - 2\langle\psi_j\,|\,j\rangle\langle j|\big)\big(|\psi_i\rangle - 2|i\rangle\langle i\,|\,\psi_i\rangle\big) \\
&= \frac{1}{N}\big(\langle\psi_j\,|\,\psi_i\rangle - 2\langle\psi_j\,|\,i\rangle\langle i\,|\,\psi_i\rangle - 2\langle\psi_j\,|\,j\rangle\langle j\,|\,\psi_i\rangle + 4\langle\psi_j\,|\,j\rangle\langle j\,|\,i\rangle\langle i\,|\,\psi_i\rangle\big) .
\end{aligned}$$

For $i = j$, there is no change at all, since ρ_{ii} is always $(1/N)\left|\psi_i\right|^2 = 1/N$. For $i \neq j$, we have $\langle j \mid i\rangle = 0$, eliminating the fourth term. Thus the off-diagonal terms decrease by

$$\rho'_{ij} = \rho_{ij} - \Delta_{ij} \quad \text{where} \quad \Delta_{ij} = \frac{2}{N}\big(\langle\psi_j\,|\,i\rangle\langle i\,|\,\psi_i\rangle + \langle\psi_j\,|\,j\rangle\langle j\,|\,\psi_i\rangle\big) . \tag{15.47}$$

To measure our progress, we will adopt a very simple measure of entanglement—namely, the sum of all of ρ's entries,

$$S = \sum_{ij} \rho_{ij} .$$

From (15.45) and (15.46), we see that $S = N$ at the beginning of the algorithm and $S = 1$ at the end. According to (15.47), each query changes S by

$$S' = S - \Delta \quad \text{where} \quad \Delta = \sum_{i,j:i\neq j} \Delta_{ij} .$$

We will show that $|\Delta| = O(\sqrt{N})$. Since S must decrease from N to 1, we will then need $\Omega(\sqrt{N})$ queries.

To bound $|\Delta|$, first note that when we sum over all i and j, the two terms in Δ_{ij} become complex conjugates of each other. Using the triangle inequality, we can then write

$$|\Delta| \leq \frac{4}{N}\left|\sum_{i,j:i\neq j}\langle\psi_j\,|\,i\rangle\langle i\,|\,\psi_i\rangle\right| \leq \frac{4}{N}\sum_j\left|\sum_{i\neq j}\langle\psi_j\,|\,i\rangle\langle i\,|\,\psi_i\rangle\right| .$$

Bounding the inner sum over i using the Cauchy–Schwarz inequality (see Appendix A.2.3) gives

$$\begin{aligned} |\Delta| &\leq \frac{4}{N}\sqrt{\sum_i \left|\langle\psi_i|i\rangle\right|^2} \times \sum_j \sqrt{\sum_{i\neq j}\left|\langle\psi_j|i\rangle\right|^2} \\ &= \frac{4}{N}\sqrt{\sum_i p_i} \times \sum_j \sqrt{1-p_j}, \end{aligned} \tag{15.48}$$

where $p_i = \left|\langle\psi_i|i\rangle\right|^2$ denotes the probability that observing the computer gives the right answer if the haystack is in state $|i\rangle$. Applying Cauchy–Schwarz again gives

$$\begin{aligned} |\Delta| &\leq \frac{4}{N}\sqrt{\sum_i p_i} \times \sqrt{N}\sqrt{\sum_j (1-p_j)} \\ &\leq 4\sqrt{Np(1-p)}, \end{aligned}$$

where $p = (1/N)\sum_i p_i$ is the average probability that the computer finds the needle if its location is uniformly random.

Since $\sqrt{p(1-p)} \leq 1/2$ for all $0 \leq p \leq 1$, we have

$$|\Delta| \leq 2\sqrt{N}. \tag{15.49}$$

Since S has to decrease from N to 1, this shows that the minimum number of queries is at least $\sqrt{N}/2$ when N is large, and therefore that any algorithm must make $\Omega(\sqrt{N})$ queries.

15.22 What about the constant in front of $\sqrt{N}$? In (15.49), we pessimistically assumed that $\sqrt{p(1-p)} = 1/2$. However, this is only true when $p = 1/2$, so for most of the algorithm the rate of decoherence is slower than (15.49) suggests. Using another type of argument, one can show that the constant $\pi/4$ in Grover's algorithm is in fact optimal if we want to find the solution with high probability. However, if we want to minimize the *expected* number of queries and we are willing to take the risk that the algorithm might take longer, Problem 15.46 shows that we can do slightly better.

This *quantum adversary* method has been used to prove lower bounds on the number of queries for a wide variety of problems. We explore another technique for proving lower bounds, the *polynomial method*, in Problems 15.40 through 15.45.
15.23

15.8 Quantum Walks and Scattering

We end this chapter with another algorithmic idea. In Chapters 12 and 13, we showed how random walks can explore the space of possible solutions to a problem, generating a random solution in polynomial time. Here we consider the quantum version of these walks, where waves with complex amplitudes, rather than real probabilities, propagate through space.

As always, the key difference is interference. By canceling out in some places and adding up in others, these quantum walks can spread through space faster than their classical cousins. By propagating down a tree and interfering with themselves in complex ways, they can even tell us who has a winning strategy in a two-player game.

The easiest way to think of this kind of algorithm is as a physical process taking place in continuous time. As we will see, each one works by designing a matrix called the Hamiltonian, which governs the interactions in the system and acts as the transition matrix of the walk. We then simply run Schrödinger's equation, the basic equation of quantum physics, for a certain amount of time.

We start by looking at the classical version of this process. If you are not already familiar with classical random walks and with the Poisson and Gaussian distributions, you may find Appendix A.4 useful.

15.8.1 Walking the Line

As a warm-up, we consider the classical random walk on the line, but in continuous time. Suppose we have an infinite line, with one vertex for each integer. Its adjacency matrix is

$$H=\frac{1}{2}\begin{pmatrix}\ddots &&&&& \\ & 0 & 1 &&& \\ & 1 & 0 & 1 && \\ && 1 & 0 & 1 & \\ &&& 1 & 0 & \\ &&&&& \ddots\end{pmatrix}. \tag{15.50}$$

We have normalized this to make it stochastic. In particular, it is the transition matrix of a random walk where we step to the left or the right with equal probability. If we write the current probability distribution as a column vector P, the distribution after t steps is $H^t P$.

Now suppose that we perform a random walk in continuous time. In each infinitesimal time interval of width $\mathrm{d}t$, there is a probability $\mathrm{d}t$ of changing the current probability distribution P_t to HP_t. Thus, to first order in $\mathrm{d}t$, we have

$$P_{t+\mathrm{d}t}=\mathrm{d}t\,HP_t+(1-\mathrm{d}t)P_t\,. \tag{15.51}$$

This gives us a linear differential equation,

$$\frac{\mathrm{d}}{\mathrm{d}t}P_t=(H-\mathbb{1})P_t\,.$$

Its solution is just what you expect,

$$P_t=\mathrm{e}^{(H-\mathbb{1})t}\,P_0\,. \tag{15.52}$$

However, $\mathrm{e}^{(H-\mathbb{1})t}$ is a *matrix* exponential, defined using the Taylor series:

$$\begin{aligned}\mathrm{e}^{(H-\mathbb{1})t}&=\mathrm{e}^{-t}\mathrm{e}^{Ht}=\mathrm{e}^{-t}\left(\mathbb{1}+Ht+\frac{(Ht)^2}{2}+\frac{(Ht)^3}{6}+\cdots\right)\\ &=\sum_{j=0}^{\infty}\frac{\mathrm{e}^{-t}t^j}{j!}H^j\,.\end{aligned} \tag{15.53}$$

We can interpret (15.53) as follows. First, choose j from a Poisson distribution with mean t, i.e., with probability $P(j)=\mathrm{e}^{-t}t^j/j!$. Then take j steps of the discrete-time random walk by applying H^j.

Given an initial distribution $P(0)$, how can we use (15.52) to calculate the distribution P_t at time t? We can do this using Fourier analysis, just as we did for the discrete-time walk on the cycle in Section 12.7.2. The Fourier basis vectors are $v_k = e^{\imath kx}$ where k ranges from $-\pi$ to π, and decomposing P_t in this basis gives

$$P_t(x) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \widetilde{P}_t(k)\, e^{\imath kx}\, dk\,.$$

If our initial distribution is concentrated at the origin, with $P_0(0)=1$ and $P_0(x)=0$ for all $x \neq 0$, its Fourier distribution is uniform and $\widetilde{P}_0(k)=1$ for all k.

The Fourier basis vectors are eigenvectors of H. Since $(Hv)_x = (v_{x-1}+v_{x+1})/2$ for any vector v, we have

$$H v_k = \lambda_k v_k\,,$$

where the eigenvalue is

$$\lambda_k = \frac{e^{-\imath k} + e^{\imath k}}{2} = \cos k\,.$$

It follows that v_k is also an eigenvector of $e^{(H-\mathbb{1})t}$,

$$e^{(H-\mathbb{1})t}\, v_k = e^{(\lambda_k - 1)t}\, v_k\,.$$

Applying $e^{(H-\mathbb{1})t}$ to P_0 then gives

$$\begin{aligned} P_t(x) &= e^{(H-\mathbb{1})t}\, \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{\imath kx}\, dk \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{(H-\mathbb{1})t} e^{\imath kx}\, dk \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{(\cos k - 1)t}\, e^{\imath kx}\, dk\,. \end{aligned} \tag{15.54}$$

The integral (15.54) looks difficult to evaluate, but it is easy to approximate when t is large. As the probability distribution spreads out, low frequencies dominate, and we can use the second-order Taylor series $\cos k \approx 1 - k^2/2$ for small k. At the cost of a small error, we can also extend the integral over k to the entire real line, giving

$$P_t(x) \approx \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-tk^2/2}\, e^{\imath kx}\, dk = \frac{1}{\sqrt{2\pi t}}\, e^{-x^2/(2t)}\,.$$

Thus $P_t(x)$ approaches a Gaussian distribution with variance t and width $O(\sqrt{t})$. If we carry out this process on a line or a cycle of length n rather than on the infinite line, the mixing time—the time it takes to reach a nearly uniform distribution—is $\Theta(n^2)$. This is the same result we obtained in Section 12.7 and Problem 12.36 for the discrete-time walk.

15.8.2 Schrödinger's Equation and Quantum Diffusion

Our next task is to define quantum walks analogous to the classical walk we just analyzed. First we need to review a little more physics. Throughout this chapter, we have described the steps of a quantum computer as unitary operators. But where do these unitary operators come from?

Since the eigenvalues of a unitary operator U are of the form $\mathrm{e}^{\imath\theta}$, applying it to a state rotates the amplitude ψ of each eigenvector by some angle θ in the complex plane. But it takes time to do this. In an infinitesimal time interval $\mathrm{d}t$, we can only rotate the state by an infinitesimal angle $\theta = \lambda\,\mathrm{d}t$. To first order in $\mathrm{d}t$ we have

$$\mathrm{e}^{\imath\lambda\,\mathrm{d}t} = 1 + \imath\lambda\,\mathrm{d}t\,,$$

so, analogous to (15.51), the amplitude evolves as

$$\psi_{t+\mathrm{d}t} = \psi_t + \imath\lambda\psi_t\,\mathrm{d}t\,.$$

We can write this as a differential equation,

$$\frac{\mathrm{d}}{\mathrm{d}t}\psi_t = \imath\lambda\psi_t\,. \tag{15.55}$$

Now suppose that we have a matrix H with the same eigenvectors as U, but with real eigenvalues λ. Then we can express the differential equation (15.55) for all eigenvectors at once in the following way:

$$\frac{\mathrm{d}}{\mathrm{d}t}|\Psi_t\rangle = \imath H|\Psi_t\rangle\,. \tag{15.56}$$

This is known as *Schrödinger's equation*, although physicists usually write it with $-\imath$ instead of $\imath$. Assuming that H does not change with time, its solution is 15.24

$$|\Psi_t\rangle = \mathrm{e}^{\imath Ht}\,|\Psi_0\rangle\,.$$

Again using the matrix exponential, we see that running Schrödinger's equation for a time interval t applies the following unitary operator to the state Ψ,

$$\begin{aligned}\mathrm{e}^{\imath Ht} &= \mathbb{1} + \imath Ht - \frac{(Ht)^2}{2} - \imath\frac{(Ht)^3}{6} + \cdots \\ &= \sum_{j=0}^{\infty}\frac{(\imath t)^j}{j!}H^j\,.\end{aligned} \tag{15.57}$$

This operator is unitary as long as H is Hermitian, i.e., $H^\dagger = H$:

Exercise 15.25 *Show that $\mathrm{e}^{\imath Ht}$ is unitary for all real t if and only if H is Hermitian.*

In physics, H is called the *Hamiltonian.* It describes all the interactions in the system, and all the transitions it can make from one state to another. In our setting, we can take H to be a weighted adjacency matrix of an undirected graph, normalized as in (15.50) so that its maximum eigenvalue is 1.

For the quantum walk on the line, we can solve for the amplitude $\Psi_t(x)$ just as we solved for $P_t(x)$ in the classical case. The eigenvectors of H are again the Fourier basis vectors $\mathrm{e}^{\imath kx}$, with eigenvalues $\lambda = \cos k$. Writing Ψ_t in this basis gives

$$\Psi_t(x) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \widetilde{\Psi}_t(k)\,\mathrm{e}^{\imath kx}\,\mathrm{d}k\,.$$

As before, we assume that the initial state is concentrated at the origin, $\Psi_0(0) = 1$ and $\Psi_0(x) = 0$ for all $x \neq 0$. Then $\widetilde{\Psi}_0(k) = 1$ for all k, and applying $\mathrm{e}^{\imath\lambda t}$ to each eigenvector gives

$$\Psi_t(x) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \mathrm{e}^{\imath\lambda t}\,\mathrm{e}^{\imath kx}\,\mathrm{d}k = \frac{1}{2\pi}\int_{-\pi}^{\pi} \mathrm{e}^{\imath t\cos k}\,\mathrm{e}^{\imath kx}\,\mathrm{d}k\,. \tag{15.58}$$

Except for the factor of $\imath$, and the replacement of $\lambda - 1$ with λ which gives an overall phase change, this looks just like the integral (15.54) for the classical random walk. Does it result in a similar probability distribution?

As Figure 15.12 shows, the probability distribution of the quantum walk is vastly different from the classical one. Rather than being confined to a Gaussian of width $\sqrt{t}$, the probability stretches across over the interval $[-t, t]$, and is modulated by rapid oscillations. Problem 15.50 shows that if we ignore these oscillations, the distribution scales asymptotically as

$$P_t(x) = |\Psi_t(x)|^2 \sim \frac{1}{\sqrt{t^2 - x^2}}\,. \tag{15.59}$$

Since it spreads out quadratically faster than in the classical case, the mixing time on a line or cycle of length n is now just $\Theta(n)$ instead of $\Theta(n^2)$.

What's going on? As in the classical case, the continuous-time walk includes paths of many different lengths, corresponding to different powers of H in the matrix exponential. But now, looking at (15.57), we see that paths of different lengths j contribute with different phases $\imath^j$. Combinatorially speaking, the vast majority of paths still return to within $O(\sqrt{t})$ of the origin—but now they interfere destructively, suppressing their total probability. On the other hand, far from the origin we have constructive interference, making the probability much greater than it would be in the tail of a Gaussian.

As another example, Figure 15.13 shows a quantum walk on the square lattice. Instead of a two-dimensional Gaussian with width $\sqrt{t}$, its probability distribution is roughly uniform over a circle of radius $\Theta(t)$, so its mixing time on an $\ell \times \ell$ lattice is $O(\ell)$ instead of $O(\ell^2)$. Since quantum walks mix faster than classical ones, they can achieve certain search and sampling tasks faster than the classical random walks of Chapter 12. 15.25

Yet another kind of algorithm probes the properties of a graph by running a quantum walk on it. As we are about to see, we can tell who has a winning strategy by telling whether the corresponding tree is transparent or reflective.

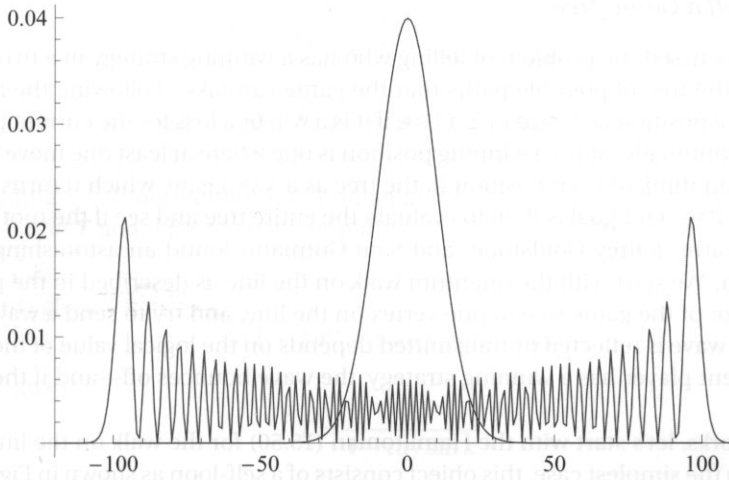


FIGURE 15.12: The quantum and classical walks on the line. The classical walk gives a Gaussian distribution of width $O(\sqrt{t})$, while the probability distribution $|\Psi|^2$ of the quantum walk is roughly uniform over the interval $[-t, t]$. Here $t = 100$.

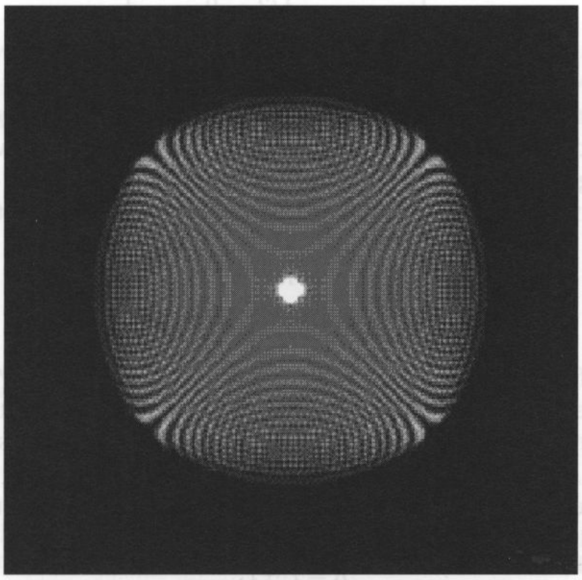


FIGURE 15.13: The probability distribution of a discrete-time quantum walk on the square lattice. Except for a bright spot in the center, the probability is roughly uniform over a circle of radius $t/\sqrt{2}$. Here $t = 100$.

15.8.3 *Scattering Off a Game Tree*

In Section 8.6.3, we discussed the problem of telling who has a winning strategy in a two-player game by recursively exploring the tree of possible paths that the game can take. Following the approach of Section 10.5, let's say that a position as `true` or `false` if it is a win or a loss for the current player, assuming that both players play optimally. Since a winning position is one where at least one move creates a loss for the other player, we can think of each position in the tree as a NAND gate, which returns `true` if at least one of its inputs is `false`. Our goal is then to evaluate the entire tree and see if the root is `true`.

In 2007, Edward Farhi, Jeffrey Goldstone, and Sam Gutmann found an astonishing quantum algorithm for this problem. We start with the quantum walk on the line as described in the previous section. We then attach the root of the game tree to one vertex on the line, and try to send a wave through it. Incredibly, whether this wave is reflected or transmitted depends on the logical value of the tree. If the root is `true` and the current player has a winning strategy, the wave bounces off—and if the root is `false`, it passes through.

To see how this works, let's start with the Hamiltonian (15.50) for the walk on the line, and attach an object to the origin. In the simplest case, this object consists of a self-loop as shown in Figure 15.14, giving an amplitude α with which the quantum walk stays at the origin instead of moving left or right. This gives

$$H = \begin{pmatrix} \ddots & & & & \\ & 0 & 1/2 & & \\ & 1/2 & \alpha & 1/2 & \\ & & 1/2 & 0 & \\ & & & & \ddots \end{pmatrix}. \tag{15.60}$$

Before this object came along, there were two eigenvectors with the same eigenvalue $\lambda = \cos k$, namely $e^{\imath kx}$ and $e^{-\imath kx}$, corresponding to right-moving and left-moving waves respectively. But now if we send a right-moving wave through the origin, part of it will be transmitted, and part of it will be reflected as a left-moving wave. We can write the combination of all three waves as a single state,

$$v(x) = \begin{cases} e^{\imath kx} + Re^{-\imath kx} & x \le 0 \\ Te^{\imath kx} & x \ge 0, \end{cases} \tag{15.61}$$

where R and T are the reflection and transmission amplitudes respectively. We then demand that this combined state be an eigenvector of the new Hamiltonian, so that it constitutes a steady flow of reflection and transmission consistent with Schrödinger's equation. There is one such eigenvector for each frequency k.

Requiring that v is an eigenvector lets us solve for R and T as a function of the self-loop amplitude α. First, taking $x = 0$ in (15.61) gives

$$R = T - 1.$$

Then applying the eigenvalue equation $Hv = \lambda v$ at the origin gives

$$\alpha v_0 + \frac{v_1 + v_{-1}}{2} = \lambda v_0. \tag{15.62}$$

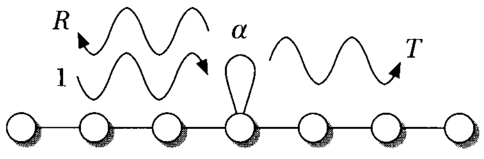


FIGURE 15.14: Adding a self-loop to the Hamiltonian at the origin. A wave entering from the left is transmitted with amplitude T and reflected with amplitude R, giving the eigenvector (15.61).

Substituting (15.61) gives

$$\alpha T + T e^{\imath k} - \imath \sin k = T \cos k\,,$$

and solving for T gives

$$T = \frac{1}{1 - \imath \alpha / \sin k}\,.$$

The probability with which the wave is transmitted is then

$$|T|^2 = \frac{1}{1 + \alpha^2 / \sin^2 k}\,.$$

When $\alpha = 0$ and the self-loop is absent, $|T|^2 = 1$ and the origin is transparent. As α increases, $|T|^2$ decreases and the self-loop becomes more and more reflective, until at $\alpha = \infty$ we have $|T|^2 = 0$ and the wave is reflected completely.

What happens when we scatter off a more complicated object? Suppose that the origin is attached to a "tadpole," an edge with amplitude β leading to a vertex that has a self-loop with amplitude α. Call this additional vertex u. Applying the eigenvalue equation $Hv = \lambda v$ at u and at the origin gives

$$\alpha v_u + \beta v_0 = \lambda v_u$$
$$\beta v_u + \frac{v_1 + v_{-1}}{2} = \lambda v_0\,.$$

Eliminating v_u, we get

$$\frac{\beta^2}{\lambda - \alpha} v_0 + \frac{v_1 + v_{-1}}{2} = \lambda v_0\,.$$

But this is exactly the equation (15.62) for a self-loop of amplitude α', where

$$\alpha' = \frac{\beta^2}{\lambda - \alpha}\,. \tag{15.63}$$

In general, for any object consisting of a tree with self-loops on its leaves, this "tadpole rule" lets us contract a branch of the tree and replace it with an equivalent self-loop as in Figure 15.15. If there are two tadpoles with $\beta = 1$ and self-loops α_1 and α_2, we can contract them both and add the resulting amplitudes together. This gives a single self-loop with amplitude

$$\alpha' = \frac{1}{\lambda - \alpha_1} + \frac{1}{\lambda - \alpha_2}\,. \tag{15.64}$$

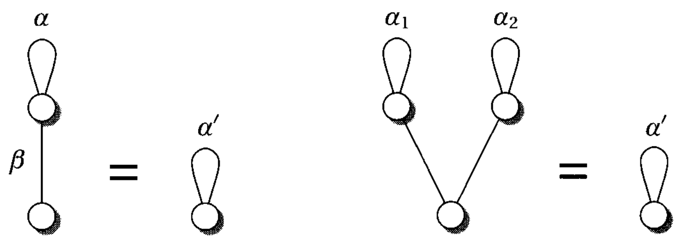


FIGURE 15.15: The tadpole rule (15.63) contracts a leaf of the tree into a self-loop. With (15.64), we can contract two leaves at once.

Note that α' depends on λ, so the entire object might be much more reflective for one eigenvector than for another.

What does all this have to do with NAND trees? Let's set $\lambda = 0$. Since $\lambda = \cos k$, this means that $k = \pi/2$. Then (15.64) gives

$$\alpha' = -\frac{1}{\alpha_1} - \frac{1}{\alpha_2}.$$

Thus α' is infinite if either α_1 or α_2 is zero, and $\alpha' = 0$ if $\alpha_1 = \alpha_2 = \infty$. But if ∞ and 0 represent `true` and `false` respectively, this means that α' is `true` if and only if at least one of α_1 and α_2 is `false`. Thus for the eigenvector with $\lambda = 0$, (15.64) acts exactly like a NAND gate with inputs α_1, α_2 and output α'.

This lets us treat the NAND tree as shown in Figure 15.16. All edges have weight $\beta = 1$. To get things started, a `false` leaf is a vertex without a self-loop, i.e., with $\alpha = 0$. A `true` leaf is connected to an additional node without a self-loop, which according to (15.63) is equivalent to a self-loop with $\alpha = \infty$. By repeatedly applying (15.64), we can contract the entire tree down to a self-loop with amplitude α at the origin. If the tree evaluates to `false`, then $\alpha = 0$ and it transmits the wave. If it is `true`, then $\alpha = \infty$ and it reflects.

How can we turn all this into an algorithm? We can't literally bounce the eigenvector with eigenvalue $\lambda = 0$ off the tree, since this would require a state distributed over the infinite line. Given finite resources, we can only create a *wave packet* of some finite length L:

$$\Psi(x) = \begin{cases} \frac{1}{\sqrt{L}} e^{\imath \pi x/2} & -L < x \le 0 \\ 0 & \text{otherwise}. \end{cases}$$

This packet moves to the right at unit speed, so after running Schrödinger's equation for time $t = L$ we can measure its position and tell whether it has been reflected or transmitted. How large does L, and therefore t, have to be?

According to Heisenberg's uncertainty principle, the width of the wave packet is inversely proportional to the spread of its frequencies in the Fourier basis. Specifically, Problem 15.51 shows that in the Fourier basis, a constant fraction of the probability lies on eigenvectors such that $|\lambda| \le \pi/L$. Thus both the width of the packet and the running time of the algorithm are inversely proportional to the largest $|\lambda|$ for which the NAND tree still works—that is, the largest $|\lambda|$ such that contracting the tree according to (15.64) leads to a clear difference in α, depending on whether it is `true` or `false`.

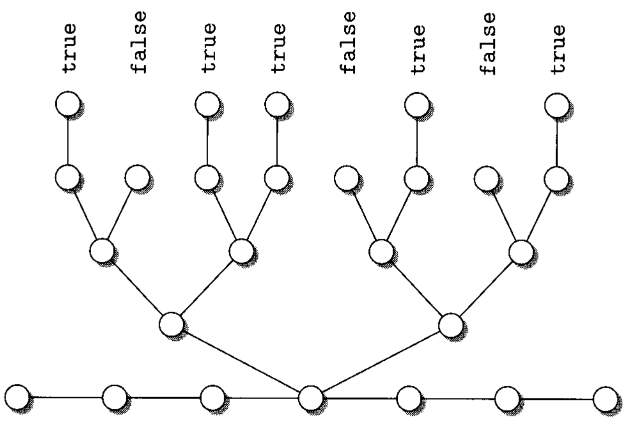


FIGURE 15.16: We can treat the NAND tree as a graph, in which true leaves have an additional vertex attached to them.

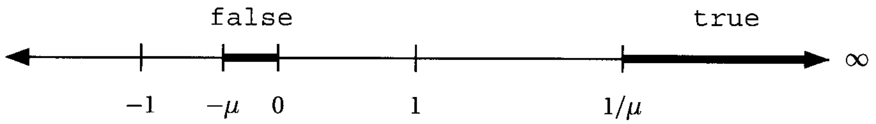


FIGURE 15.17: When $\lambda > 0$, the intervals $[-\mu, 0]$ and $[1/\mu, +\infty]$ correspond to `false` and `true` values respectively, for some $\mu > 0$ which changes from one level of the tree to the next.

How small does $|\lambda|$ have to be? Let's take $\lambda > 0$, since the case $\lambda < 0$ is similar. Equation (15.64) still acts like a NAND gate, but the logical values `true` and `false` no longer correspond exactly to α being infinite or zero. Instead, α will be in one of the two intervals shown in Figure 15.17,

$$\begin{aligned} -\mu \leq \alpha \leq 0 & \quad \text{for a } \texttt{false} \text{ node} \\ \alpha \geq 1/\mu & \quad \text{for a } \texttt{true} \text{ one,} \end{aligned}$$

for some small $\mu > 0$. For instance, contracting a `true` leaf with the tadpole rule gives a self-loop with $\alpha = 1/\lambda$, so initially we have $\mu = \lambda$. Each time we use (15.64), μ gets slightly larger, making these intervals wider. Our goal is to bound how quickly μ grows, so we can still distinguish `true` from `false` even after contracting the entire tree.

Let's assume that this correspondence holds with some value of μ for the inputs to a NAND gate, and see with what value μ' it holds for the output. For convenience, we assume that there is some $\varepsilon > 0$ such that at every level of the tree we have

$$\lambda, \mu \leq \varepsilon .$$

There are three cases to check. First, suppose that both inputs are `true`, so that $\alpha_1, \alpha_2 \geq 1/\mu$. Then the output should be `false`, so $\alpha' \geq -\mu'$.

Looking at (15.64), we see that α' is a decreasing function of both α_1 and α_2. So we can get a lower bound on α', and thus an upper bound on μ', by setting $\alpha_1 = \alpha_2 = 1/\mu$. Plugging these values into (15.64) gives

$$\alpha' \geq \frac{2}{\lambda - 1/\mu} = -\mu' \quad \text{where} \quad \mu' = \frac{2\mu}{1-\lambda\mu} \leq \frac{2\mu}{1-\varepsilon^2}\,.$$

Similarly, if both inputs are `false`, we set $\alpha_1 = \alpha_2 = -\mu$. The output should be `true`, and so

$$\alpha' \geq \frac{2}{\lambda+\mu} = -\frac{1}{\mu'} \quad \text{where} \quad \mu' = \frac{\mu+\lambda}{2}\,.$$

Finally, if one leaf is `true` and the other is `false`, we set $\alpha_1 = 1/\mu$ and $\alpha_2 = -\mu$. The output should again be `true`, giving

$$\alpha' \geq \frac{1}{\lambda - 1/\mu} + \frac{1}{\lambda+\mu} = -\frac{1}{\mu'} \quad \text{where} \quad \mu' = \frac{(\mu+\lambda)(1-\lambda\mu)}{1-2\lambda\mu-\mu^2} \leq \frac{\mu+\lambda}{1-3\varepsilon^2}\,.$$

Looking over these equations, we see that the first case is the most dangerous—if both leaves are `true`, then μ roughly doubles. Happily, in this case the output of the NAND gate is `false`, so this can only occur at half of the nodes along any path from a leaf to the root. So, we expect that μ doubles once for every two levels of the tree. In other words, if μ_j denotes the value of μ after we have contracted j levels of the tree, we expect that μ_{j+2} is roughly twice μ_j.

To confirm this, let's take the worst case of the above equations, in which every `true` node has one `true` input and one `false` one. Note that this is also the worst case for the classical randomized algorithm, as we discussed in Section 10.5. Applying this to a tree of depth two and simplifying gives

$$\mu_{j+2} \leq \frac{2}{1-6\varepsilon^2}(\mu_k+\lambda) \leq (2+\varepsilon)(\mu_j+\lambda)\,. \tag{15.65}$$

We assumed in the second inequality that $\varepsilon \leq 0.08$, but this is fine since we can take ε to be any small constant. Taking the initial value $\mu_0 = \lambda$ and iterating (15.65) implies the following upper bound,

$$\mu_j \leq 3(2+\varepsilon)^{j/2}\,\lambda\,. \tag{15.66}$$

Exercise 15.26 *Confirm* (15.65). *Then solve for μ_j exactly and confirm* (15.66).

Now suppose that the tree has N nodes and depth $k = \log_2 N$. Setting $j = k$, we find that at the root of the tree we have

$$\mu_k \leq 3(2+\varepsilon)^{k/2}\,\lambda = 3N^{\frac{1}{2}\log_2(2+\varepsilon)}\,\lambda \leq 3N^{\frac{1}{2}+\varepsilon}\,\lambda\,,$$

where we used the generous upper bound $\log_2(2+\varepsilon) \leq 1+2\varepsilon$. If we want to ensure that $\mu_k \leq \varepsilon$, we have to take

$$\lambda \leq \frac{\varepsilon}{3}N^{-\left(\frac{1}{2}+\varepsilon\right)} = \Theta\left(N^{-\left(\frac{1}{2}+\varepsilon\right)}\right).$$

Therefore, the length of the wave packet, and the running time of our algorithm, obey

$$t \sim L = \Theta\left(N^{\frac{1}{2}+\varepsilon}\right).$$

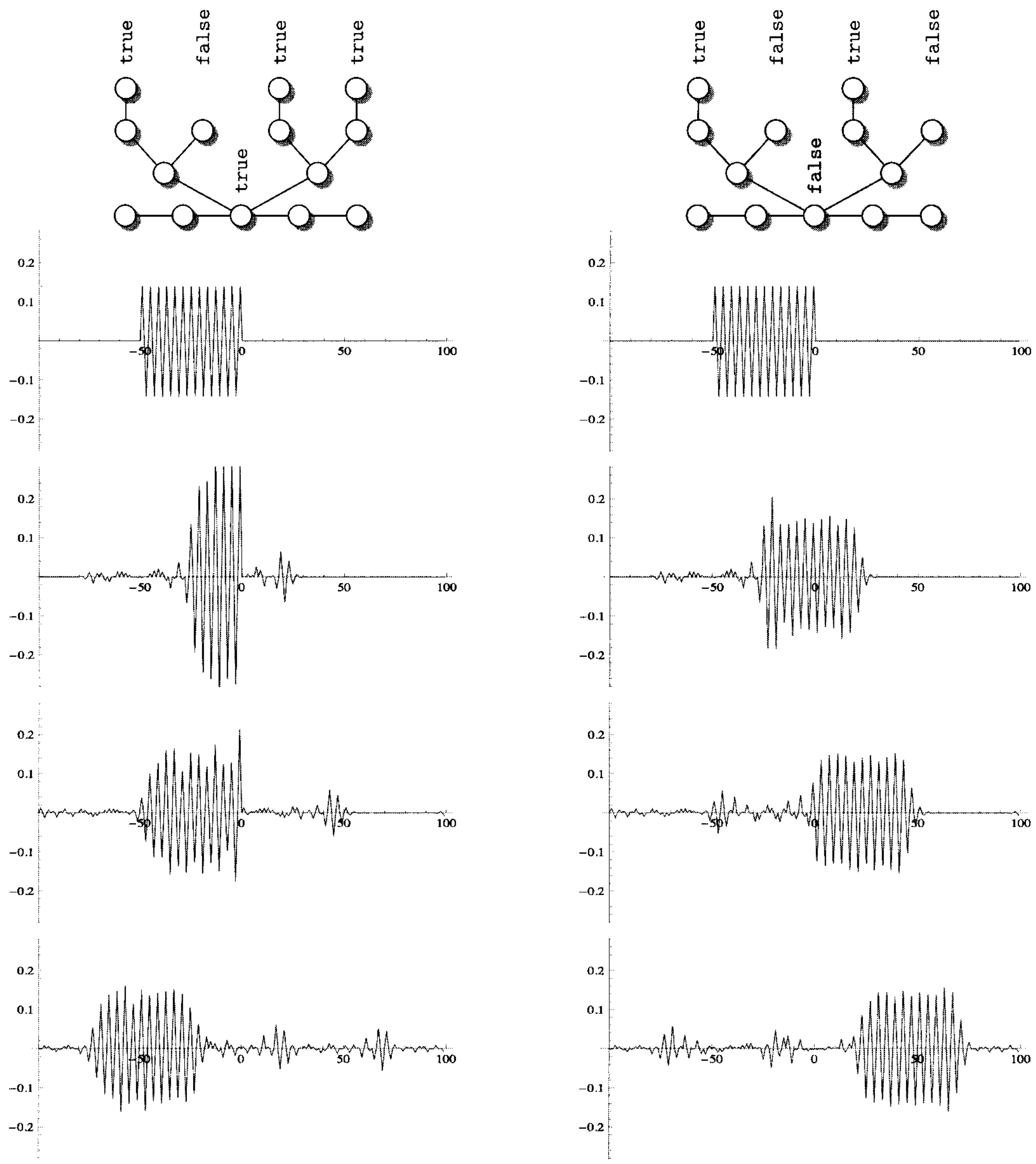


FIGURE 15.18: The Farhi–Goldstone–Gutmann algorithm in action. On the left, the NAND tree is `true`, and the wave packet is reflected. On the right, it is `false`, and the packet is transmitted. The width of the packet is $L = 50$. Top to bottom, we take snapshots at $t = 0, 25, 50$, and 75. We plot the real part of the amplitude in order to show the phase oscillations.

By making ε arbitrarily small, we can make this exponent as close to $1/2$ as we like, with a slight cost in the leading constant. In fact, our analysis here was deliberately crude—with a more careful calculation, one can get rid of ε completely, and obtain an algorithm that runs in time $\Theta(N^{1/2})$.

We show an example of this algorithm at work in Figure 15.18. On the left, the tree is `true`, and the wave packet is reflected. On the right, we change the tree to `false` by altering the truth value of one leaf, and the packet is transmitted instead.

How does this algorithm compare with classical algorithms for the NAND tree? We showed in Section 10.5 that the best possible randomized algorithm queries $\Theta(N^{0.753})$ leaves. Comparing the number of queries with the running time of a continuous-time algorithm may seem like comparing apples and oranges. However, we can think of this quantum algorithm as querying a *Hamiltonian oracle*, in which case the number of queries is proportional to t. Moreover, it is possible to transform this continuous-time algorithm into a discrete-time one carried out by a quantum circuit, in which the number of queries is again $\Theta(N^{1/2})$. While we focused here on balanced binary trees, this algorithm can also be generalized to evaluate any tree or Boolean formula, with essentially the same running time.

15.26 The Farhi–Goldstone–Gutmann algorithm shows that, in the right setting, even the simplest physical processes have surprising computational power. We have been thinking about quantum computing for less than twenty years, and there is every reason to believe that new algorithmic ideas will continue to emerge. As they do, we will learn more about the nature of complexity, and about the computational power of the universe in which we live.

Problems

You have nothing to do but mention the quantum theory, and people will take your voice for the voice of science, and believe anything.

George Bernard Shaw

15.1 Inner products. Show that if $\langle v|U^\dagger U|v\rangle = \langle v\,|\,v\rangle$ for all vectors $|v\rangle$, then $\langle v|U^\dagger U|w\rangle = \langle v\,|\,w\rangle$ for all $|v\rangle$ and $|w\rangle$, and therefore that $U^\dagger U = \mathbb{1}$.

15.2 Simulating complex numbers. Consider the following mapping from complex numbers to real 2×2 matrices:

$$c = re^{\imath\theta} \to r\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} = \begin{pmatrix} \operatorname{Re} c & -\operatorname{Im} c \\ \operatorname{Im} c & \operatorname{Re} c \end{pmatrix}.$$

Show that this mapping respects addition, multiplication, and inverses (except for zero). Formally, it is an isomorphism from the field $\mathbb{C}$ to the real-valued, invertible 2×2 matrices. Conclude that n-dimensional unitary matrices can be represented by real-valued $2n$-dimensional ones. Thus complex numbers, strictly speaking, are not necessary to quantum mechanics—but negative numbers are.

15.3 The square root of NOT. Construct a unitary matrix V such that $V^2 = X$. Such a V can be called the "square root of NOT," since it switches a qubit's truth value if we apply it twice. From a classical point of view, this is quite strange!