Computational Complexity: A Modern Approach

Sanjeev Arora and Boaz Barak
Princeton University

http://www.cs.princeton.edu/theory/complexity/

complexitybook@gmail.com

Chapter 10

Quantum Computation

"Turning to quantum mechanics.... secret, secret, close the doors! we always have had a great deal of difficulty in understanding the world view that quantum mechanics represents ... It has not yet become obvious to me that there's no real problem. I cannot define the real problem, therefore I suspect there's no real problem, but I'm not sure there's no real problem. So that's why I like to investigate things."
Richard Feynman, 1964

"The only difference between a probabilistic classical world and the equations of the quantum world is that somehow or other it appears as if the probabilities would have to go negative.."
Richard Feynman, in "Simulating physics with computers," 1982

Quantum computing is a new computational model that may be physically realizable and may provide an exponential advantage over "classical" computational models such as probabilistic and deterministic Turing machines. In this chapter we survey the basic principles of quantum computation and some of the important algorithms in this model.

One important reason to study quantum computers is that they pose a serious challenge to the *strong Church-Turing thesis* (see Section 1.6.3), which stipulates that every physically reasonable computation device can be simulated by a Turing machine with at most polynomial slowdown. As we will see in Section 10.6, there is a polynomial-time algorithm for quantum computers to factor integers, whereas despite much effort, no such algorithm is known for deterministic or probabilistic Turing machines. If in fact there is no efficient classical algorithm to factor integers (and indeed society currently relies on this conjecture since it underlies the presumed security of cryptographic schemes such as RSA) *and* if quantum computers are physically realizable, then the strong Church-Turing thesis is wrong. Physicists are also interested in quantum computers because studying them may shed light on quantum mechanics, a theory which, despite its great success in predicting experiments, is still not fully understood.

Very little physics is needed to understand the central results of quantum computing. One basic fact is that the physical parameters (energy, momentum, spin etc.) of an elementary particle such as an electron are *quantized* and can only take values in a discrete set. Second, contrary to our basic intuition, the value of a physical parameter of a particle (including location, energy, etc.) at any moment in time is not a single number. Rather the parameter has a kind of *probability wave* associated with it, involving a "smearing" or "superposition" over all possible values. The parameter only achieves a definite value when it is *measured* by an observer, at which point we say that the probability wave *collapses* to a single value.

This smearing of a parameter value until it is observed may seem analogous to philosophical musings such as "if a tree falls in a forest with no one present to hear it, does it make a sound?" But these probability waves are very real, and their interaction and

mutual interference creates experimentally measurable effects. Furthermore, according to quantum mechanics, the probability waves are associated not just with single particles, but also by any collection of particles (such as humans!). This interaction of probability waves corresponding to collections of particles is key to the power of quantum computing, and underlies the apparent exponential speedup provided by this model on certain problems. At the same time, it is simplistic to describe quantum computing —as many popular science authors do— as a "vastly parallel" computer. This "vast parallelism" is tightly regulated by the laws of quantum mechanics, which currently seems to allow exponential speedups only for a few well-structured problems.

The chapter is organized as follows. In Section 10.1 we describe the 2-slit experiment, one of many experiments that illustrate the smearing/interference effects of quantum mechanics. In Section 10.2 we formalize a simple quantum system called "qubit" (short for "quantum bit") that is the fundamental unit of quantum computing. We describe operations that can be performed on systems of one or few qubits, and illustrate them in Section 10.2.1 using the famous EPR paradox, an experiment that serves to demonstrate (and verify) the counterintuitive nature of quantum mechanics. Then in Section 10.3 we define the n-qubit *quantum register*, and operations (including computations) that can be performed on such registers. We define *quantum circuits* and the class **BQP**, which is the quantum analogue of **BPP**. The three ensuing sections describe three basic algorithms known for quantum computers, due to Grover, Simon and Shor respectively. Several important topics in quantum computing, including lower bounds, quantum cryptography and quantum error correction, are not covered in this chapter; see the chapter notes for links to further reading.

This chapter utilizes some basic facts of linear algebra and the space $\mathbb{C}^n$. These are reviewed in Appendix A; see also Section 10.3.1.

10.1 Quantum weirdness: the 2-slit experiment

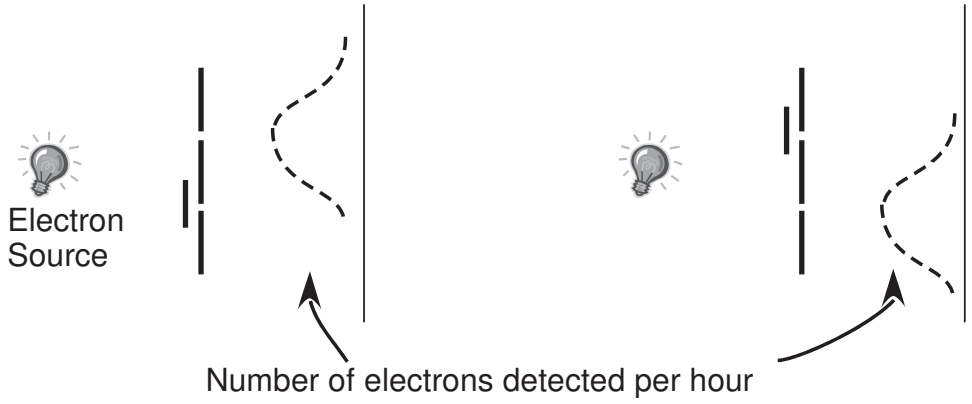


Figure 10.1 In the 2-slit experiment an electron source is placed between a wall with two slits and a detector array. When one slit is covered then, as expected, the number of electron detected is largest directly behind the open slit.

Now we describe an experiment, called the *2-slit experiment*, that illustrates the fact that basic physical properties of an elementary particle are "smeared."

Suppose that, as in Figure 10.1, a source that fires electrons one by one (say, at the rate of one electron per second) is placed in front of a wall containing two tiny slits. Beyond the wall we place an array of detectors that light up whenever an electron hits them. We measure the number of times each detector lights up during an hour. When we cover one of the slits, we would expect that the detectors that are directly behind the open slit will receive the largest number of hits, and as Figure 10.1 shows, this is indeed the case. When

both slits are uncovered we expect that the number of times each detector is hit is the sum of the number of times it is hit when the first slit is open and the number of times it is hit when the second slit is open. In particular, uncovering both slits should only *increase* the number of times each location is hit.

Surprisingly, this is not what happens. The pattern of hits exhibits the "interference" phenomena depicted in Figure 10.2. In particular, at several detectors the total hit rate is *lower* when both slits are open as compared to when a single slit is open. This defies explanation if electrons behave as particles or "little balls".

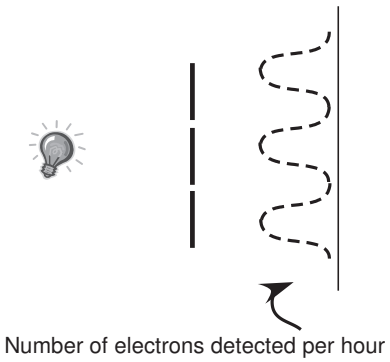


Figure 10.2 When both slits are open in the 2-slit experiment, the number of electrons detected at each position is *not* the sum of numbers when either slit is opened. There are even positions that are hit when each slit is open on its own, but are *not* hit when both slits are open.

According to quantum mechanics, it is wrong to think of an electron as a "little ball" that can either go through the first slit or the second (i.e., has a definite property). Rather, somehow the electron instantaneously explores all possible paths to the detectors through all open slits. Some paths are taken with positive "amplitude" and some with negative "amplitude" (see the quote from Feynman at the start of the chapter) and two paths arriving at a detector with opposite signs will cancel each other. The end result is that the distribution of hit rates depends upon the number of open slits, since the electron "finds out" how many slits are open via this exploration of all possible paths.

You might be skeptical about this "path exploration," and to check if it is actually going on, you place a detector at each slit that lights up whenever an electron passes through that slit. Thus if an electron is really going through both slits simultaneously, you hope to detect it at both slits. However, when you try to make the electron reveal its quantum nature this way, the quantum nature (i.e., interference pattern) disappears! The hit rates now observed exactly correspond to what they would be if electrons were little balls: the sum of the hits when each slit is open. The explanation is that, as stated above, *observing* an object "collapses" its distribution of possibilities, and so changes the result of the experiment.[1] One moral to draw from this is that quantum computers, if they are ever built, will have to be carefully isolated from external influences and noise, since noise may be viewed as a "measurement" performed by the environment on the system. Of course, we can never completely isolate the system, which means we have to make quantum computation tolerant of a little noise. This seems to be possible under some noise models, see the chapter notes.

[1] Of course, it is unclear why humans or detectors placed by humans serve to "collapse" the probability wave, and inanimate objects such as slits do not. This is one of the puzzles of quantum mechanics, see the chapter notes.

10.2 Quantum superposition and qubits

Now we describe quantum superposition using a very simple quantum system called a *qubit*, which lays the groundwork for our formal development of quantum computing in the next section. As a helpful example for readers who are new to quantum mechanics, we also describe the famous EPR paradox, though understanding it not strictly necessary to understand the rest of the chapter.

Classical computation involves manipulation of storage elements with finite memory: the tape cell of a Turing Machine, or a bit in case of a Boolean circuit. The analogous unit of storage in quantum computing is a *qubit*. We can think of it as an elementary particle that can be in two basic states (which could correspond to values of energy, or spin or some other physical parameter), which we denote by zero and one. However, unlike a classical bit, this particle can be simultaneously in both basic states. Thus the state of a qubit at any time is called a *superposition* of these basic states. Formally, we denote the basic states by $|0\rangle$ and $|1\rangle$ and generally allow a qubit to be in any state of the form $\alpha_0 |0\rangle + \alpha_1 |1\rangle$, where α_0, α_1 are called *amplitudes* and are complex numbers satisfying $|\alpha_0|^2 + |\alpha_1|^2 = 1$.[2] If isolated from outside influences, the qubit stays in this superposition, until it is observed by an observer. When the qubit is observed, with probability $|\alpha_0|^2$ it is revealed to be in state zero (i.e., $|0\rangle$) and with probability $|\alpha_1|^2$ it is revealed to be in state one (i.e., $|1\rangle$). After observation the amplitude wave *collapses* and the values of the amplitudes are irretrievably lost.

In this section we restrict attention to the case where the amplitudes are real (though possibly negative) numbers. The power and "weirdness" of quantum computing is already exhibited in this case (see also Exercise 10.5).

Analogously, a system of two qubits can exist in four basic states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ and the state of a 2-qubit system at any time is described by a superposition of the type

$$\alpha_{00} |00\rangle + \alpha_{01} |01\rangle + \alpha_{10} |10\rangle + \alpha_{11} |11\rangle .$$

where $\sum_{b_1,b_2} |\alpha_{b_1 b_2}|^2 = 1$. When this system is observed, its state is revealed to be $|b_1 b_2\rangle$ with probability $|\alpha_{b_1 b_2}|^2$.

We will sometimes denote the state $|xy\rangle$ as $|x\rangle |y\rangle$. Readers who are mystified by the $|\cdot\rangle$ notation (which unfortunately is inescapable due to long tradition) may wish to look at Note 10.1 for a more geometric description.

Example 10.2

The following are two legitimate state vectors for a 1-qubit quantum system: $\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle$ and $\frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle$. Even though in both cases, if the qubit is measured it will contain either 0 or 1 with probability $1/2$, these are considered distinct states and we will see that it is possible to differentiate between them using quantum operations.

Because states are always unit vectors, we often drop the normalization factor and so, say, use $|0\rangle - |1\rangle$ to denote the state $\frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle$.

We call the state where all coefficients are equal the *uniform* state. For example, the uniform state for a 2-qubit system is

$$|00\rangle + |01\rangle + |10\rangle + |11\rangle ,$$

(where we dropped the normalization factor of $\frac{1}{2}$). Using our earlier notation of $|x\rangle |y\rangle$ for $|xy\rangle$ (an operation that is easily checked to respect the distributive law), so we can also write the uniform state of a 2-qubit system as

$$(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)$$

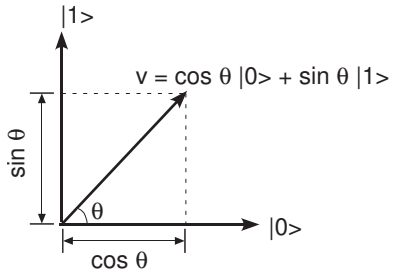
which shows that this state just consists of two 1-qubit systems in uniform state.

---

[2] We note that in quantum mechanics the above is known as a *pure* quantum state, see also the remarks following Definition 10.9.

Note 10.1 *(The geometry of quantum states)*

It is often helpful to think of quantum states geometrically as vectors. For example, in case of the single qubit system (with real amplitudes), the two basic states can be visualized as two orthogonal unit vectors $|0\rangle$ and $|1\rangle$ in $\mathbb{R}^2$ (say, $|0\rangle = (1,0)$ and $|1\rangle = (0,1)$). The state of the system, which we denoted by $\alpha_0|0\rangle + \alpha_1|1\rangle$, can be interpreted as a vector that is α_0 times the first vector and α_1 times the second. Since α_0, α_1 are real numbers satisfying $\alpha_0^2 + \alpha_1^2 = 1$, there is a unique angle $\theta \in [0, 2\pi)$ such that $\alpha_0 = \cos\theta, \alpha_1 = \sin\theta$. Thus we can think of the system state as $\cos\theta|0\rangle + \sin\theta|1\rangle$; that is, it is a unit vector that makes an angle θ with the $|0\rangle$ vector and an angle $\pi/2 - \theta$ with the $|1\rangle$ vector. When measured, the system's state is revealed to be $|0\rangle$ with probability $\cos^2\theta$ and $|1\rangle$ with probability $\sin^2\theta$.



Although it's harder to visualize states with complex coefficients or more than one qubit, geometric intuition can still be useful when reasoning about such states.

To manipulate the state of a qubit, we have to use a *quantum operation*, which is a function that maps the current state to the new state. In this section we will only use operations on single qubits. Quantum mechanics allows only *unitary* operations, which are linear operations that preserve the invariant $|\alpha_0|^2 + |\alpha_1|^2 = 1$. In the case of single qubit operations with real coefficients, this means that the allowed operations involve either a *reflection* of the state vector about a fixed vector in $\mathbb{R}^2$ or a *rotation* of the state vector by some angle $\theta \in [0, 2\pi)$.

10.2.1 EPR paradox

The EPR paradox, named after its proposers, Einstein, Podosky, and Rosen [EPR35] was a thought experiment that shows that quantum mechanics allows systems in two far corners of the universe to instantaneously coordinate their actions, seemingly contradicting the axiom of Einstein's special theory of relativity that nothing in the universe can travel faster than light. Einstein, who despite his status as a founder of quantum theory (with his 1905 paper on the photoelectric effect) was never quite comfortable with it, felt that quantum mechanics must be modified to remove such paradoxes.

In 1964 John Bell showed how to turn the EPR thought experiment into an *actual* experiment. Two systems far away from each other in the universe have a shared quantum state (actually, a 2-qubit system). This shared state allows them to coordinate their actions in a way that is provably impossible in a "classical" system.

Since then Bell's experiment has been repeated in a variety of settings, always with the same result: the predictions of quantum mechanics are correct, contrary to Einstein's intuition. Today, the EPR paradox is not considered a paradox, since the systems involved do not *transmit* information faster than the speed of light— they merely act upon information that was already shared, albeit in the form of a quantum superposition. Now we describe a version of Bell's experiment (or, more accurately, a variant due to Clauser et al [CHSH69]):

The parity game. We start by describing a game that seems to involve no quantum mechanics at all. Two players Alice and Bob are isolated from each other. The experimenter asks them to participate in the following guessing game.

1. The experimenter chooses two random bits $x, y \in_R \{0,1\}$.
2. He presents x to Alice and y to Bob.
3. Alice and Bob respond with bits a, b respectively.
4. Alice and Bob win if and only if $a \oplus b = x \wedge y$, where $\oplus$ denotes the XOR operation (addition modulo 2).

Note that the players' isolation from each other can be ensured using the special theory of relativity. The players are separated by a light year (say), each accompanied by an assistant of the experimenter. At a designated time, the experimenter's assistants toss their independent random coins to create x and y, present them to Alice and Bob respectively, receive their answers, and transmit everything to the experimenter at a central location. Alice and Bob, being separated by a light year, cannot exchange any information between the time they received x, y and before they gave their answers.

It is easy for Alice and Bob to ensure that they win with probability at least $3/4$ e.g., by always sending $a = b = 0$. Now we show that this is the best they can do, which seems intuitive since the setup forbids them from *coordinating* their responses. Thus a *strategy* for the players is a pair of functions $f, g : \{0,1\} \to \{0,1\}$ such that the players' answers a, b are computed only as functions of the information they see, namely, $a = f(x)$ and $b = g(y)$. A *probabilistic strategy* is a distribution on strategies.

Theorem 10.3 *([Bel64, CHSH69]) In the above scenario, no (deterministic or probabilistic) strategy used by Alice and Bob can cause them to win with probability more than $3/4$.* $\diamondsuit$

PROOF: Assume for the sake of contradiction that there is a (possibly probabilistic) strategy that causes them to win with probability more than $3/4$. By a standard averaging argument there is a fixed choice of the players' randomness that succeeds with at least the same probability, and hence we may assume without loss of generality that the players' strategy is deterministic.

The function $f : \{0,1\} \to \{0,1\}$ that Alice uses can be one of only four possible functions: it can be either the constant function zero or one, the function $f(x) = x$ or the function $f(x) = 1 - x$. We analyze the case that $f(x) = x$; the other cases are similar. Now Alice's response a is merely x, so the players win iff $b = (x \wedge y) \oplus x$. On input y, Bob needs to find b that makes them win. If $y = 1$ then $x \wedge y = x$ and hence $b = 0$ will ensure their win with probability 1. However, if $y = 0$ then $(x \wedge y) \oplus x = x$ and since Bob does not know x, the probability that his output b is equal to x is at most $1/2$. Thus the total probability of a win is at most $3/4$. ■

The Parity Game with sharing of quantum information. Now we show that if Alice and Bob can share a 2-qubit system (which they created in a certain state, and split between them before they they were taken a light year apart) then they can circumvent Theorem 10.3 and win the parity game with probability better than $3/4$ using the following strategy:

1. Before the game begins, Alice and Bob prepare a 2-qubit system in the state $|00\rangle + |11\rangle$, which we will call the *EPR state*.
2. Alice and Bob split the qubits - Alice takes the first qubit and Bob takes the second qubit. Note that quantum mechanics does not require the individual bits of a 2-qubit quantum system to be physically close to one another. It is important that Alice and Bob have not *measured* these qubits yet.

3. Alice receives the qubit x from the experimenter, and if $x = 1$ then she applies a rotation by $\pi/8$ (22.5°) operation to her qubit. Since the operation involves only her qubit, she can perform it even with no input from Bob. (The semantics of performing a single qubit operation on a multiple-qubit system follow the natural intuition, but see Section 10.3.3 for a formal description.)

4. Bob receives the qubit y from the experimenter, and if $y = 1$ he applies a rotation by by $-\pi/8$ ($-22.5°$) operation to his qubit.

5. Both Alice and Bob measure their respective qubits and output the values obtained as their answers a and b.

Note that the order in which Alice and Bob perform their rotations and measurements does not matter - it can be shown that all orders yield exactly the same distribution (e.g., see Exercise 10.6). While splitting a 2-qubit system and applying unitary transformations to the different parts may sound far fetched, this experiment had been performed several times in practice, verifying the following prediction of quantum mechanics:

Theorem 10.4 *With the above strategy, Alice and Bob win with probability at least* 0.8.♢

PROOF: Recall that Alice and Bob win the game if they output a different answer when $x = y = 1$ and the same answer otherwise. The intuition behind the proof is that unless $x = y = 1$, the states of the two qubits will be "close" to one another (with the angle between being at most $\pi/8 = 22.5°$) and in the other case the states will be "far" (having angle $\pi/4$ or 45°). Specifically we will show that (denoting by a Alice's output and by b Bob's):

(1) If $x = y = 0$ then $a = b$ with probability 1.

(2) If $x \neq y$ then $a = b$ with probability $\cos^2(\pi/8) \geq 0.85$

(3) If $x = y = 1$ then $a = b$ with probability $1/2$.

Implying that the overall acceptance probability is at least $\frac{1}{4} \cdot 1 + \frac{1}{2} \cdot 0.85 + \frac{1}{4} \cdot \frac{1}{8} = 0.8$.

In the case **(1)** both Alice and Bob perform no operation on their qubits, and so when measured it will be either in the state $|00\rangle$ or $|11\rangle$, both resulting in Alice and Bob's outputs being equal. To analyze case **(2)**, it suffices to consider the case that $x = 0, y = 1$ (the other case is symmetrical). In this case Alice applies no transformation to her qubit, and Bob rotates his qubit in a $-\pi/8$ angle. Imagine that Alice first measures her qubit, and then Bob makes his rotation and measurements (this is OK as the order of measurements does not change the outcome). With probability $1/2$, Alice will get the value 0 and Bob's qubit will collapse to the state $|0\rangle$ rotated by a $-\pi/8$ angle, meaning that when measuring Bob will obtain the value 0 with probability $\cos^2(\pi/8)$. Similarly, if Alice gets the value 1 then Bob will also output 1 with $\cos^2(\pi/8)$ probability.

To analyze case **(3)**, we just use direct computation. In this case, after both rotations are performed, the 2-qubit system has the state

$$\begin{aligned}
&(\cos(\pi/8)|0\rangle + \sin(\pi/8)|1\rangle)(\cos(\pi/8)|0\rangle - \sin(\pi/8)|1\rangle) + \\
&\quad (-\sin(\pi/8)|0\rangle + \cos(\pi/8)|1\rangle)(\sin(\pi/8)|0\rangle + \cos(\pi/8)|1\rangle) = \\
&\quad \left(\cos^2(\pi/8) - \sin^2(\pi/8)\right)|00\rangle - 2\sin(\pi/8)\cos(\pi/8)|01\rangle + \\
&\quad 2\sin(\pi/8)\cos(\pi/8)|10\rangle + \left(\cos^2(\pi/8) - \sin^2(\pi/8)\right)|11\rangle .
\end{aligned}$$

But since

$$\cos^2(\pi/8) - \sin^2(\pi/8) = \cos(\pi/4) = \tfrac{1}{\sqrt{2}} = \sin(\pi/4) = 2\sin(\pi/8)\cos(\pi/8) ,$$

all coefficients in this state have the same absolute value and hence when measured the 2-qubit system will yield either one of the four values $00, 01, 10$ and 11 with equal probability $1/4$. ■

The constant 0.8 can be somewhat improved upon, see Exercise 10.1. Also, there are known games with more dramatic differences in success probabilities between the classical and quantum cases. In an interesting twist, in recent years the ideas behind EPR's and Bell's experiments have been used for a practical goal: encryption schemes whose security depends only on the principles of quantum mechanics, rather than any unproven conjectures such as $\mathbf{P} \neq \mathbf{NP}$ (see chapter notes).

10.3 Definition of quantum computation and BQP

In this section we describe quantum operations, which lead to the definition of quantum gates, quantum computation, and **BQP**, the class of languages with efficient quantum decision algorithms.

10.3.1 Some necessary linear algebra

We use in this chapter several elementary facts and notations involving the space $\mathbb{C}^M$. These are reviewed in Section A.5 of the appendix, but here is a quick reminder:

- If $z = a + ib$ is a complex number (where $i = \sqrt{-1}$), then $\overline{z} = a - ib$ denotes the *complex conjugate* of z. Note that $z\overline{z} = a^2 + b^2 = |z|^2$.
- The *inner product* of two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{C}^m$, denoted by $\langle \mathbf{u}, \mathbf{v} \rangle$, is equal to $\sum_{x \in [M]} \mathbf{u}_x \overline{\mathbf{v}}_x$.[3]
- The *norm* of a vector $\mathbf{u}$, denoted by $\|\mathbf{u}\|_2$, is equal to $\sqrt{\langle \mathbf{u}, \mathbf{u} \rangle} = \sqrt{\sum_{x \in [M]} |\mathbf{u}_x|^2}$.
- If $\langle \mathbf{u}, \mathbf{v} \rangle = 0$ we say that $\mathbf{u}$ and $\mathbf{v}$ are *orthogonal*.
- A set $\{\mathbf{v}^i\}_{i \in [M]}$ of vectors in $\mathbb{C}^M$ is an *orthonormal basis* of $\mathbb{C}^M$ if for every $i, j \in [M]$, $\langle \mathbf{v}^i, \mathbf{v}^j \rangle$ is equal to 1 if $i = j$ and equal to 0 if $i \neq j$.
- If A is an $M \times M$ matrix, then A^* denotes the *conjugate transpose* of A. That is, $A^*_{x,y} = \overline{A}_{y,x}$ for every $x, y \in [M]$.
- An $M \times M$ matrix A is *unitary* if $AA^* = I$, where I is the $M \times M$ identity matrix.

Note that if z is a *real* number (i.e., z has no imaginary component) then $\overline{z} = z$. Hence, if all vectors and matrices involved are real then the inner product is equal to the standard inner product of $\mathbb{R}^n$ and the conjugate transpose operation is equal to the standard transpose operation. Also, for real vectors $\mathbf{u}, \mathbf{v}$, $\langle \mathbf{u}, \mathbf{v} \rangle = \cos\theta \|\mathbf{u}\|_2 \|\mathbf{v}\|_2$, where θ is the angle between the $\mathbf{u}$ and $\mathbf{v}$.

The next claim (left as Exercise 10.2) summarizes properties of unitary matrices:

Claim 10.5 *For every $M \times M$ complex matrix A, the following conditions are equivalent:*

1. *A is unitary (i.e., $AA^* = I$).*
2. *For every vector $\mathbf{v} \in \mathbb{C}^M$, $\|A\mathbf{v}\|_2 = \|\mathbf{v}\|_2$.*
3. *For every orthonormal basis $\{\mathbf{v}^i\}_{i \in [M]}$ of $\mathbb{C}^M$, the set $\{A\mathbf{v}^i\}_{i \in [M]}$ is an orthonormal basis of $\mathbb{C}^M$.*
4. *The columns of A form an orthonormal basis of $\mathbb{C}^M$.*
5. *The rows of A form an orthonormal basis of $\mathbb{C}^M$.*

[3] Some quantum computing texts use $\sum_{x \in [M]} \overline{\mathbf{u}}_x \mathbf{v}_x$ instead.

10.3.2 The quantum register and its state vector

In a standard digital computer, we implement a bit of memory by a physical object that has two states: the ON or 1 state and the OFF or 0 state. By taking m such objects together we have an *m-bit register* whose state can be described by a string in $\{0,1\}^m$. A *quantum register* is composed of m qubits, and its state is a *superposition* of all 2^m basic states (the "probability wave" alluded to in Section 10.1): a vector $\mathbf{v} = \langle \mathbf{v}_{0^m}, \mathbf{v}_{0^{m-1}1}, \ldots, \mathbf{v}_{1^m} \rangle \in \mathbb{C}^{2^m}$, where $\sum_x |\mathbf{v}_x|^2 = 1$. According to quantum mechanics, when *measuring* the register (i.e., reading its value) we will obtain the value x with probability $|\mathbf{v}_x|^2$ and furthermore this operation will *collapse* the state of the register to the vector $|x\rangle$ (in other words, the coefficients corresponding to the basic states $|y\rangle$ for $y \neq x$ will become 0). In principle such a quantum register can be implemented by any collection of m objects that can have an ON and OFF states, although in practice there are significant challenges for such an implementation.

10.3.3 Quantum operations

Now we define operations allowed by quantum mechanics.

Definition 10.6 *(quantum operation)*
A *quantum operation* for an m-qubit register is a function $F : \mathbb{C}^{2^m} \to \mathbb{C}^{2^m}$ that maps its previous state to the new state and satisfies the following conditions:

Linearity: F is a linear function. That is, for every $\mathbf{v} \in \mathbb{C}^{2^n}$, $F(\mathbf{v}) = \sum_x \mathbf{v}_x F(|x\rangle)$.

Norm preservation: F maps unit vectors to unit vectors. That is, for every $\mathbf{v}$ with $\|\mathbf{v}\|_2 = 1$, $\|F(\mathbf{v})\|_2 = 1$.

The second condition (norm preservation) is quite natural given that only unit vectors can describe states. The linearity condition is imposed by the theory of quantum mechanics. Together, these two conditions imply that every quantum operation F can be described by a $2^m \times 2^m$ *unitary* matrix. The following is immediate.

Lemma 10.7 *(Composition of quantum operations) If A_1, A_2 are matrices representing any quantum operations, then their composition (i.e., applying A_1 followed by applying A_2) is also a quantum operation whose matrix is $A_2 A_1$. In particular, since $A_1 A_1^* = I$, every quantum operation has a corresponding "inverse" operation that cancels it. (Quantum computation is "reversible.")* ♢

Since quantum operations are linear, it suffices to describe their behavior on any linear basis for the space $\mathbb{C}^{2^m}$ and so we often specify quantum operations by the way they map the standard basis. However, not every classical operation is unitary, so designing quantum operations requires care.

10.3.4 Some examples of quantum operations

Here are some examples of quantum operations:

Flipping qubits. If we wish to "flip" the first qubit in an m-qubit register, (i.e., apply the NOT operation on the first qubit), then this can be done as a quantum operation that maps the basis state $|b, x\rangle$ for $b \in \{0,1\}, x \in \{0,1\}^{m-1}$ to the basis state $|1-b, x\rangle$. The matrix of this operation just performs a permutation on the standard basis, and permutation matrices are always unitary. **Important note on notation:** This example involved an operation

on the first qubit, so the remaining qubits in x are unaffected and unnecessarily cluttering the notation. From now on, whenever we describe operations on only a subset of qubits, we will often drop the unaffected qubits from the notation. The above operation can be described as $|0\rangle \mapsto |1\rangle$ and $|1\rangle \mapsto |0\rangle$.

Reordering qubits. If we wish to exchange the values of two qubits the following operation (again, unitary since it is a permutation of basic states) suffices: $|01\rangle \mapsto |10\rangle$ and $|10\rangle \mapsto |01\rangle$, with $|00\rangle$ and $|11\rangle$ being mapped to themselves. This operation is described by the following $2^2 \times 2^2$ matrix (where we index the rows and columns according to lexicographical order $|00\rangle, |01\rangle, |10\rangle, |11\rangle$):

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

Note that by combining such operations we can arbitrarily reorder the qubits of an m-qubit register.

Copying qubits. Now suppose we wish to copy the first qubit into the second. Proceeding naively, we might try the following: both $|10\rangle$ and $|11\rangle$ map to $|11\rangle$ whereas both $|00\rangle$ and $|01\rangle$ map to $|00\rangle$. However, this is not a reversible operation and hence not unitary! In fact, the so-called *no cloning theorem* rules out any quantum operation that copies qubits; see the Chapter notes. However, while designing quantum algorithms it usually suffices to copy a qubit in "write once" fashion, by keeping around a supply of fresh qubits in a predetermined state, say $|0\rangle$, and only writing them over once. Now the operation $|xy\rangle \mapsto |x(x \oplus y)\rangle$ provides the effect of copying the first qubit, assuming the algorithm designer takes care to apply it only where the second qubit is a fresh (i.e., unused) qubit in state $|0\rangle$, and thus the operation never encounters the states $|01\rangle, |11\rangle$. Since this operation negates the second qubit y if and only if x is in the state $|1\rangle$ it known as the *controlled NOT* (or CNOT for short) operation in the literature.

Rotation on single qubit. Thinking of the phase of a qbit as a 2-dimensional vector as in Note 10.1, we may wish to apply a rotation this state vector by an angle θ. This corresponds to the operation $|0\rangle \mapsto \cos\theta\,|0\rangle + \sin\theta\,|1\rangle$, and $|1\rangle \mapsto -\sin\theta\,|0\rangle + \cos\gamma\,|1\rangle$, described by the matrix $\left(\begin{smallmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{smallmatrix}\right)$, which is unitary. Note that when $\theta = \pi$ (i.e., 180°) this amounts to flipping the sign of the state vector (i.e., the map $\mathbf{v} \mapsto -\mathbf{v}$).

AND of two bits. Now consider the classical AND operation, concretely, the operation that replaces the first qubit of the register by the AND of the first two bits. One would try to think of this as a linear operation $|b_1 b_2\rangle \mapsto |b_1 \wedge b_2\rangle\,|b_2\rangle$ for $b_1, b_2 \in \{0,1\}$. But this is unfortunately not reversible and hence not unitary.

However, there is a different way to achieve the effect of an AND operation. This uses a "reversible AND", which uses an additional scratchpad in the form of a fresh qubit b_3. The operation is $|b_1\rangle\,|b_2\rangle\,|b_3\rangle \mapsto |b_1\rangle\,|b_2\rangle\,|b_3 \oplus (b_1 \wedge b_2)\rangle$ for all $b_1, b_2, b_3 \in \{0,1\}$. This operation is unitary (in fact, permutation matrix) and thus a valid quantum operation, described by the following matrix

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

As before, the algorithm designer will only apply this operation when b_3 is a fresh qubit in state $|0\rangle$. This operation is also known in quantum computing as the *Toffoli gate*. One can similarly obtain a "reversible OR" quantum operation. Together, the reversible OR and AND gates are key to showing that quantum computers can simulate ordinary Turing machines (see Section 10.3.7).

The Hadamard operation. The *Hadamard* gate it is the single qubit operation that (up to normalization) maps $|0\rangle$ to $|0\rangle + |1\rangle$ and $|1\rangle$ to $|0\rangle - |1\rangle$. More succinctly, the state $|b\rangle$ is mapped to $|0\rangle + (-1)^b |1\rangle$. The corresponding matrix is $\frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$.

Note that if we apply a Hadamard gate to every qubit of an m-qubit register, then for every $x \in \{0,1\}^m$, the state $|x\rangle$ is mapped to

$$(|0\rangle + (-1)^{x_1}|1\rangle)(|0\rangle + (-1)^{k_2}|1\rangle)\cdots(|0\rangle + (-1)^{x_m}|1\rangle) = \sum_{y\in\{0,1\}^m} \left(\prod_{i\,:\,y_i=1} (-1)^{x_i}\right)|y\rangle = \sum_{y\in\{0,1\}^m} -1^{x\odot y}|y\rangle\,, \quad (1)$$

where $x \odot y$ denotes the dot product modulo 2 of x and y. The unitary matrix corresponding to this operation is the $2^m \times 2^m$ matrix whose $(x,y)^{th}$ entry is $\frac{-1^{x\odot y}}{\sqrt{2^n}}$ (identifying $[2^m]$ with $\{0,1\}^m$). This operation plays a key role in quantum algorithms[4].

10.3.5 Quantum computation and BQP

Even though the rules of quantum mechanics allow an arbitrary unitary matrix operation to be applied on the current state of a qantum register, not all such operations can be feasibly implemented. However, highly 'local" operations— those that act on only a finite number of qubits— could perhaps be implemented. We thus define these as *elementary steps* in quantum computation.

Definition 10.8 *(Elementary quantum operations or quantum gates)* An quantum operation is called *elementary*, or sometimes a *quantum gate*, if it acts on three or less qubits of the register.[5] ♢

Note that an elementary operation on an m-qubit register can be specified by three indices in $[m]$ and an 8×8 unitary matrix. For example, if U is any 8×8 unitary matrix that has to be applied to the qbits numbered $2,3,4$ then this can be viewed as a an elementary quantum operation $F : \mathbb{C}^{2^m} \to \mathbb{C}^{2^m}$ that maps the basis state $|x_1x_2\ldots x_m\rangle$ to the state $|x_1\rangle(U|x_2x_3x_4\rangle)|x_5\ldots x_m\rangle$ for all $x_1, x_2, \ldots, x_m \in \{0,1\}$.

Now we can define quantum computation: it is a sequence of elementary operations applied to a quantum register.

[4] We will encounter this matrix again in Chapters 1 and 19 where we describe the *Walsh-Hadamard* error correcting code. (Though there will describe it as a 0/1 matrix over GF(2) rather than ± 1 matrix over $\mathbb{C}$.)

[5] The constant three is arbitrary in the sense that replacing it with every constant greater or equal to two would lead to an equivalently powerful model.

Definition 10.9 *(Quantum Computation and the class* **BQP***)*
Let $f : \{0,1\}^* \to \{0,1\}$ and $T : \mathbb{N} \to \mathbb{N}$ be some functions. We say that f is *computable in quantum $T(n)$-time* if there is a polynomial-time classical TM that on input $(1^n, 1^{T(n)})$ for any $n \in \mathbb{N}$ outputs the descriptions of quantum gates $F_1, \ldots, F_T$ such that for every $x \in \{0,1\}^n$, we can compute $f(x)$ by the following process with probability at least $2/3$:

1. Initialize an m qubit quantum register to the state $|x0^{n-m}\rangle$ (i.e., x padded with zeroes), where $m \leq T(n)$.
2. Apply one after the other $T(n)$ elementary quantum operations $F_1, \ldots, F_T$ to the register.
3. Measure the register and let Y denote the obtained value. (That is, if $\mathbf{v}$ is the final state of the register, then Y is a random variable that takes the value y with probability $|\mathbf{v}_y|^2$ for every $y \in \{0,1\}^m$.)
4. Output Y_1.

A Boolean function $f : \{0,1\}^* \to \{0,1\}$ is in **BQP** if there is some polynomial $p : \mathbb{N} \to \mathbb{N}$ such that f is computable in quantum $p(n)$-time.

Some remarks are in order:

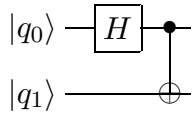
1. This definition easily generalizes to functions with more than one bit of output.
2. Elementary operations are represented by 8×8 matrices of complex numbers, which a TM cannot write per se. However, it suffices for the TM to write the most significant $O(\log T(n))$ bits of the complex number; see Exercise 10.8.
3. It can be shown that the set of elementary operations or gates (which is an infinite set) can be reduced without loss of generality to two *universal operations*; see Section 10.3.8
4. Readers familiar with quantum mechanics or quantum computing may notice that our definition of quantum computation disallows several features that are allowed by quantum mechanics, such as *mixed* states that involve both quantum superposition and probability and measurement in different bases than the standard basis. However, none of these features adds to the computing power of quantum computers. Another feature that we do not explicitly allow is performing partial measurements of some of the qubits in the course of the computation. Exercise 10.7 shows that such partial measurements can always be eliminated without much loss of efficiency, though it will sometime be convenient for us to describe our algorithms as using them.

Quantum versus probabilistic computation: At this point the reader may think that the quantum model "obviously" gives exponential speedup as the states of registers are described by 2^m-dimensional vectors and operations are described by $2^m \times 2^m$ matrices. However, this is not the case. One can describe even ordinary probabilistic computation in a similar way: we can think of the state of an m-qubit register as a 2^m-dimensional vector whose x^{th} coordinate denotes the probability that the register contains the string x, and considering probabilistic operations as linear stochastic maps from $\mathbb{R}^{2^m}$ to $\mathbb{R}^{2^m}$: see Exercise 10.4. The added power of quantum computing seems to derive from the fact that here we allow vectors to have *negative* coefficients (recall Feynman's quote from the start of the chapter), and the norm that is preserved at each step is the Euclidean (i.e., ℓ_2) norm rather than the sum (i.e., ℓ_1) norm (see also Exercise 10.5). Note also that classical computation, whether deterministic or probabilistic, is a subcase of quantum computation, as we see in Section 10.3.7.

10.3.6 Quantum circuits

Definition 10.9 is reminiscent of the the definition of classical *straight-line* programs, which as we saw in Chapter 6 is an equivalent model to Boolean circuits (see Note 6.4). Similarly, one can define quantum computation and **BQP** also in terms of *quantum circuits* (in fact, this is the definition appearing in most texts). Quantum circuits are similar to Boolean circuits: these are directed acyclic graphs with sources (vertices with in-degree zero) denoting the inputs, sinks (vertices with out-degree zero) denoting the outputs, and internal nodes denoting the gates. One difference is that this time the gates are labeled not by the operations AND,OR and NOT but by 2×2, 4×4 or 8×8 unitary matrices. Another difference is that (since copying is not allowed) the out-degree of gates and even inputs cannot be arbitrarily large but rather the out-degree of each input vertex is one, and the in-degree and out-degree of each gate are equal (and are at most 3). We also allow special "workspace" or "scratchpad" inputs that are initialized to the state $|0\rangle$.

Such circuits are often described in the literature using diagrams such as the one below, depicting a circuit that on input $|q_0\rangle |q_1\rangle$ first applies the Hadamard operation on $|q_0\rangle$ and then applies the mapping $|q_0 q_1\rangle \mapsto |q_0(q_0 \oplus q_1)\rangle$:



10.3.7 Classical computation as a subcase of quantum computation

In Section 10.3.3, we saw quantum implementations of the classical NOT and AND operations. More generally, we can efficiently simulate any classical computation using quantum operations:

Lemma 10.10 *(Boolean circuits as a subcase of quantum circuits) If $f : \{0,1\}^n \to \{0,1\}^m$ is computable by a Boolean circuit of size S then there is a sequence of $2S+m+n$ quantum operations computing the mapping $|x\rangle \left|0^{2m+S}\right\rangle \mapsto |x\rangle |f(x)\rangle \left|0^{S+m}\right\rangle$.* ♢

Proof: Replace each Boolean gate (AND,OR, NOT) by its quantum analog as already outlined. The resulting computation maps $|x\rangle \left|0^{2m}\right\rangle \left|0^{S}\right\rangle \mapsto |x\rangle |f(x)0^m\rangle |z\rangle$, where z is the string of values taken by the internal wires in the Boolean circuit (these correspond to "scratchpad" memory used by the quantum operations at the gates) and the string 0^m consists of qubits unused so far. Now copy $f(x)$ onto the string 0^m using m operations of the form $|bc\rangle \mapsto |b(b \oplus y)\rangle$. Then run the operations corresponding to the Boolean operations in reverse (applying the inverse of each operation). This erases the original copy of $f(x)$ as well as $|z\rangle$ and leaves behind clean bits in state $|0\rangle$, together with one copy of $f(x)$. ■

Since a classical Turing machine computation running in $T(n)$ steps has an equivalent Boolean circuit of size $O(T(n) \log T(n))$ it also follows that $\mathbf{P} \subseteq \mathbf{BQP}$. Using the Hadamard operation that maps $|0\rangle$ to $|0\rangle + |1\rangle$ we can get a qubit that when measured gives $|0\rangle$ with probability $1/2$ and $|1\rangle$ with probability $1/2$, simulating a coin toss. Thus the following corollary is immediate:

Corollary 10.11 $\mathbf{BPP} \subseteq \mathbf{BQP}$. ♢

10.3.8 Universal operations

Allowing every 3-qubit quantum operation as "elementary" seems problematic since this set is infinite. By contrast, classical Boolean circuits only need the gates AND, OR and NOT. Fortunately, a similar result holds for quantum computation. The following theorem (whose proof we omit) shows that there is a set of few operations that suffice to construct any quantum operation:

Theorem 10.12 *(Universal basis for quantum operations* [Deu89, Kit97]*)*
For every $D \geq 3$ and $\epsilon > 0$ there is $\ell \leq 100(D \log 1/\epsilon)^3$ such that the following is true. Every $D \times D$ unitary matrix U can be approximated as a product of unitary matrices $U_1, \ldots, U_\ell$ in the sense that its $(i,j)$the entry for each $i, j \leq D$ satisfies

$$\left| U_{i,j} - \left(U_\ell \cdots U_1 \right)_{i,j} \right| < \epsilon \,,$$

and each U_r corresponds to applying either the Hadamard gate $\frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$, the Toffoli gate $|abc\rangle \mapsto |ab(c \oplus a \wedge b)\rangle$ or the phase shift gate $\left(\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right)$, on at most 3 qubits.

It can be shown that such ϵ-approximation for, say, $\epsilon < \frac{1}{10^T}$ suffices for simulating any T-time quantum computation (see Exercise 10.8), and hence we can replace any computation using T arbitrary elementary matrices by a computation using only one of the above three gates. Other universal gates are also known and in particular Shi [Shi03] showed that for the purpose of quantum computation, the Hadamard and Toffoli gates alone suffice (this uses the fact that complex numbers are not necessary for quantum computation, see Exercise 10.5).

One corollary of Theorem 10.12 is that 3-qubit gates can be used to simulate k-qubit gates for every constant $k > 3$ (albeit at a cost exponential in k). This means that when designing quantum algorithms we can consider every k-qubit gate as elementary as long as k is smaller than some absolute constant. We can use this fact to obtain a quantum analog of the "`if` *cond* `then`" construct of classical programming languages. That is, given a T step quantum circuit for an n-qubit quantum operation U then we can compute the quantum operation *Controlled-U* in $O(T)$ steps, where Controlled-U maps a vector $|x_1 \ldots x_n x_{n+1}\rangle$ to $|U(x_1 \ldots x_n)x_{n+1}\rangle$ if $x_{n+1} = 1$ and to itself otherwise. The reason is that we can transform every elementary operation F in the computation of U to the analogous "Controlled-F" operation. Since the "Controlled-F" operation depends on at most 4 qubits, it too can be considered elementary.

10.4 Grover's search algorithm.

We now describe Grover's algorithm, one of the basic and quite useful algorithms for quantum computers. This section can be read independently of sections 10.5 and 10.6, that describe Simon's and Shor's algorithms, and so the reader who is anxious to see the integer factorization algorithm can skip ahead to Section 10.5.

Consider the **NP**-complete problem SAT of finding, given an n-variable Boolean formula φ, whether there exists an assignment $a \in \{0,1\}^n$ such that $\varphi(a) = 1$. Using "classical" deterministic or probabilistic TM's, we do not know how to solve this problem better than the trivial $\mathrm{poly}(n)2^n$-time algorithm.[6] We now show a beautiful algorithm due to Grover that solves SAT in $\mathrm{poly}(n)2^{n/2}$-time on a quantum computer. This is a significant improvement over the classical case, even if it falls way short of showing that $\mathbf{NP} \subseteq \mathbf{BQP}$. In fact, Grover's algorithm solves an even more general problem, namely, satisfiability of a circuit with n inputs.

Theorem 10.13 *(Grover's Algorithm* [Gro96]*)*
There is a quantum algorithm, that given as input every *polynomial-time computable function $f : \{0,1\}^n \to \{0,1\}$ (i.e., represented as a circuit computing f) finds in* $\mathrm{poly}(n)2^{n/2}$ *time a string a such that $f(a) = 1$ (if such a string exists).*

[6] There are slightly better algorithms for special cases such as 3SAT.

Grover's algorithm is best described geometrically. We assume that the function f has a *single* satisfying assignment a. (The techniques described in Chapter 17, Section 17.4.1 allow us to reduce the general problem to this case.) Consider an n-qubit register, and let $\mathbf{u}$ denote the *uniform state vector* of this register. That is, $\mathbf{u} = \frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x\rangle$. The angle between $\mathbf{u}$ and the basis state $|a\rangle$ is equal to the inverse cosine of their inner product $\langle \mathbf{u}, |a\rangle \rangle = \frac{1}{2^{n/2}}$. Since this is a positive number, this angle is smaller than $\pi/2$ (90°), and hence we denote it by $\pi/2 - \theta$, where $\sin\theta = \frac{1}{2^{n/2}}$ and hence (using the inequality $\theta \geq \sin\theta$ for $\theta > 0$), $\theta \geq 2^{-n/2}$.

The algorithm starts with the state $\mathbf{u}$, and at each step it gets nearer the state $|a\rangle$. If its current state makes an angle $\pi/2 - \alpha$ with $|a\rangle$ then at the end of the step it makes an angle $\pi/2 - \alpha - 2\theta$. Thus, in $O(1/\theta) = O(2^{n/2})$ steps it will get to a state $\mathbf{v}$ whose inner product with $|a\rangle$ is larger than, say, $1/2$, implying that a measurement of the register will yield a with probability at least $1/4$.

The main idea is that to rotate a vector $\mathbf{w}$ towards the unknown vector $|a\rangle$ by an angle of θ, it suffices to take two *reflections* around the vector $\mathbf{u}$ and the vector $\mathbf{e} = \sum_{x \neq a} |a\rangle$ (the latter is the vector orthogonal to $|a\rangle$ on the plane spanned by $\mathbf{u}$ and $|a\rangle$). See Figure 10.3 for a "proof by picture".

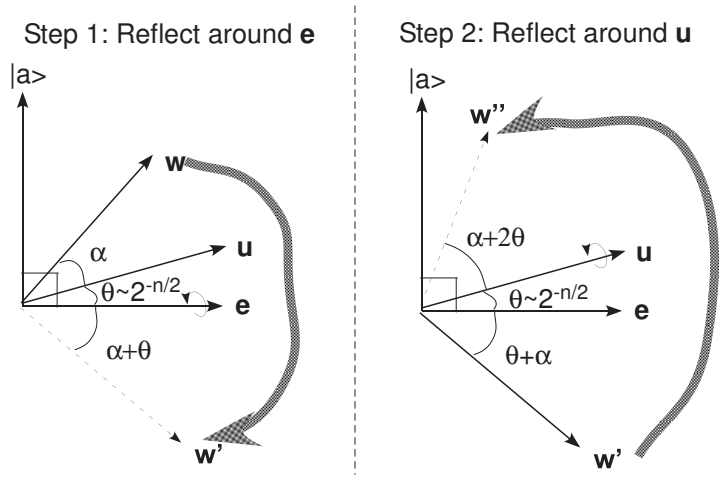


Figure 10.3 We transform a vector $\mathbf{w}$ in the plane spanned by $|a\rangle$ and $\mathbf{u}$ into a vector $\mathbf{w}''$ that is 2θ radians close to $|a\rangle$ by performing two reflections. First, we reflect around $\mathbf{e} = \sum_{x \neq a} |x\rangle$ (the vector orthogonal to $|a\rangle$ on this plane), and then we reflect around $\mathbf{u}$. If the original angle between $\mathbf{w}$ and $|a\rangle$ was $\pi/2 - \theta - \alpha$ then the new angle will be $\pi/2 - \theta - \alpha - 2\theta$. We can restrict our attention to the plane spanned by $\mathbf{u}$ and $|a\rangle$ as the reflections leave all vectors orthogonal to this plane fixed.

To complete the algorithm's description, we need to show how we can perform the reflections around the vectors $\mathbf{u}$ and $\mathbf{e}$. That is, we need to show how we can in polynomial time transform a state $\mathbf{w}$ of the register into the state that is $\mathbf{w}$'s reflection around $\mathbf{u}$ (respectively, $\mathbf{e}$). In fact, we will not work with an n-qubit register but with an m-qubit register for m that is polynomial in n. However, the extra qubits will only serve as "scratch workspace" and will always contain zero except during intermediate computations (thanks to the "cleanup" idea of the proof of Lemma 10.10), and hence can be safely ignored.

Reflecting around e. Recall that to reflect a vector $\mathbf{w}$ around a vector $\mathbf{v}$, we express $\mathbf{w}$ as $\alpha\mathbf{v} + \mathbf{v}^\perp$ (where $\mathbf{v}^\perp$ is orthogonal to $\mathbf{v}$) and output $\alpha\mathbf{v} - \mathbf{v}^\perp$. Thus the reflection of $\mathbf{w}$ around $\mathbf{e}$ is equal to $\sum_{x \neq a} \mathbf{w}_x |x\rangle - \mathbf{w}_a |a\rangle$. Yet, it is easy to perform this transformation:

1. Since f is computable in polynomial time, we can compute the transformation $|x\sigma\rangle \mapsto |x(\sigma \oplus f(x))\rangle$ in polynomial time (this notation ignores the extra workspace that may be needed, but this won't make any difference). This transformation maps $|x0\rangle$ to $|x0\rangle$ for $x \neq a$ and $|a0\rangle$ to $|a1\rangle$.

2. Then, we apply the elementary transformation (known as a Z gate) $|0\rangle \mapsto |0\rangle$, $|1\rangle \mapsto -|1\rangle$ on the qubit σ. This maps $|x0\rangle$ to $|x0\rangle$ for $x \neq a$ and maps $|a1\rangle$ to $-|a1\rangle$.

3. Then, we apply the transformation $|x\sigma\rangle \mapsto |x(\sigma \oplus f(x))\rangle$ again, mapping $|x0\rangle$ to $|x0\rangle$ for $x \neq a$ and maps $|a1\rangle$ to $|a0\rangle$.

The final result is that the vector $|x0\rangle$ is mapped to itself for $x \neq a$, but $|a0\rangle$ is mapped to $-|a0\rangle$. Ignoring the last qubit, this is exactly a reflection around $|a\rangle$.

Reflecting around u. To reflect around $\mathbf{u}$, we first apply the Hadamard operation to each qubit, mapping $\mathbf{u}$ to $|0\rangle$. Then, we reflect around $|0\rangle$ (this can be done in the same way as reflecting around $|a\rangle$, just using the function $g : \{0,1\}^n \to \{0,1\}$ that outputs 1 iff its input is all zeroes instead of f). Then, we apply the Hadamard operation again, mapping $|0\rangle$ back to $\mathbf{u}$.

Together these operations allow us to take a vector in the plane spanned by $|a\rangle$ and $\mathbf{u}$ and rotate it 2θ radians closer to $|a\rangle$. Thus if we start with the vector $\mathbf{u}$, we will only need to repeat them $O(1/\theta) = O(2^{n/2})$ to obtain a vector that, when measured, yields $|a\rangle$ with constant probability.

This completes the proof of Theorem 10.13. For the sake of completeness, Figure 10.4 contains the full description of Grover's algorithm. ■

Grover's Search Algorithm.

Goal: Given a polynomial-time computable $f : \{0,1\}^n \to \{0,1\}$ with a unique $a \in \{0,1\}^n$ such that $f(a) = 1$, find a.

Quantum register: We use an $n+1+m$-qubit register, where m is large enough so we can compute the transformation $|x\sigma 0^m\rangle \mapsto |x(\sigma \oplus f(x))0^m\rangle$.

Operation	**State** (neglecting normalizing factors)
	Initial state: $\left|0^{n+m+1}\right\rangle$
Apply Hadamard operation to first n qubits.	$\mathbf{u}\left|0^{m+1}\right\rangle$ (where $\mathbf{u}$ denotes $\sum_{x\in\{0,1\}^n}|x\rangle$)
For $i = 1,\ldots,2^{n/2}$ do:	$\mathbf{v}^i\left|0^{m+1}\right\rangle$ We let $\mathbf{v}^1 = \mathbf{u}$ and maintain the invariant that $\langle \mathbf{v}^i, |a\rangle\rangle = \sin(i\theta)$, where $\theta \sim 2^{-n/2}$ is such that $\langle \mathbf{u}, |a\rangle\rangle = \sin(\theta)$
Step 1: *Reflect around* $\mathbf{e} = \sum_{x\neq a}|x\rangle$:	
1.1 Compute $\|x\sigma 0^m\rangle \mapsto |x(\sigma \oplus f(x))0^m\rangle$	$\sum_{x\neq a}\mathbf{v}^i_x|x\rangle\left|0^{m+1}\right\rangle + \mathbf{v}^i_a|a\rangle|10^m\rangle$
1.2 If $\sigma = 1$ then multiply vector by -1, otherwise do not do anything.	$\sum_{x\neq a}\mathbf{v}^i_x|x\rangle\left|0^{m+1}\right\rangle - \mathbf{v}^i_a|a\rangle|10^m\rangle$
1.3 Compute $\|x\sigma 0^m\rangle \mapsto |x(\sigma \oplus f(x))0^m\rangle$.	$\mathbf{w}^i\left|0^{m+1}\right\rangle = \sum_{x\neq a}\mathbf{v}^i_x|x\rangle\left|0^{m+1}\right\rangle - \mathbf{v}^i_a|a\rangle|00^m\rangle$. ($\mathbf{w}^i$ is $\mathbf{v}^i$ reflected around $\sum_{x\neq a}|x\rangle$.)
Step 2: *Reflect around* $\mathbf{u}$:	
2.1 Apply Hadamard operation to first n qubits.	$\langle \mathbf{w}^i, \mathbf{u}\rangle|0^n\rangle\left|0^{m+1}\right\rangle + \sum_{x\neq 0^n}\alpha_x|x\rangle\left|0^{m+1}\right\rangle$, for some coefficients α_x's (given by $\alpha_x = \sum_z(-1)^{x\odot z}\mathbf{w}^i_z|z\rangle$).
2.2 Reflect around $\|0\rangle$:	
2.2.1 If first n-qubits are all zero then flip $n+1^{st}$ qubit.	$\langle \mathbf{w}^i, \mathbf{u}\rangle|0^n\rangle|10^m\rangle + \sum_{x\neq 0^n}\alpha_x|x\rangle\left|0^{m+1}\right\rangle$
2.2.2 If $n+1^{st}$ qubit is 1 then multiply by -1	$-\langle \mathbf{w}^i, \mathbf{u}\rangle|0^n\rangle|10^m\rangle + \sum_{x\neq 0^n}\alpha_x|x\rangle\left|0^{m+1}\right\rangle$
2.2.3 If first n-qubits are all zero then flip $n+1^{st}$ qubit.	$-\langle \mathbf{w}^i, \mathbf{u}\rangle|0^n\rangle\left|0^{m+1}\right\rangle + \sum_{x\neq 0^n}\alpha_x|x\rangle\left|0^{m+1}\right\rangle$
2.3 Apply Hadamard operation to first n qubits.	$\mathbf{v}^{i+1}\left|0^{m+1}\right\rangle$ (where $\mathbf{v}^{i+1}$ is $\mathbf{w}^i$ reflected around $\mathbf{u}$)
Measure register and let a' be the obtained value in the first n qubits. If $f(a') = 1$ then output a'. Otherwise, repeat.	

Figure 10.4 Grover's Search Algorithm

10.5 Simon's Algorithm

Although beautiful, Grover's algorithm still has a significant drawback: it is merely quadratically faster than the best known classical algorithm for the same problem. In contrast, in this section we show *Simon's algorithm* that is a polynomial-time quantum algorithm solving a problem for which the best known classical algorithm takes *exponential* time.

Simon's problem: *Given:* A polynomial-size classical circuit for a function $f : \{0,1\}^n \to \{0,1\}^n$ such that there exists $a \in \{0,1\}^n$ satisfying $f(x) = f(y)$ iff $x = y \oplus a$ for every $x, y \in \{0,1\}^n$.
Goal: find this string a.

Theorem 10.14 *(Simon's Algorithm* [Sim94]*)*
There is a polynomial-time quantum algorithm for Simon's problem.

Two natural questions are (1) why is this problem interesting? and (2) why do we believe it is hard to solve for classical computers? The best answer to (1) is that, as we will see in Section 10.6, a generalization of Simon's problem turns out to be crucial in the quantum polynomial-time algorithm for the famous *integer factorization* problem. Regarding (2), of course we do not know for certain that this problem does not have a classical polynomial-time algorithm (in particular, if $\mathbf{P} = \mathbf{NP}$ then there obviously exists such an algorithm). However, some intuition why it may be hard can be gleaned from the following *black box* model: suppose that you are given access to a black box (or oracle) that on input $x \in \{0,1\}^n$, returns the value $f(x)$. Would you be able to learn a by making at most a subexponential number of queries to the black box? It is not hard to see that if a is chosen at random from $\{0,1\}^n$ and f is chosen at random subject to the condition that $f(x) = f(y)$ iff $x = y \oplus a$ then no algorithm can successfully recover a with reasonable probability using significantly less than $2^{n/2}$ queries to the black box. Indeed, an algorithm using fewer queries is very likely to never get the same answer to two distinct queries, in which case it gets no information about the value of a.

10.5.1 Proof of Theorem 10.14

Simon's algorithm is actually quite simple. It uses a register of $2n + m$ qubits, where m is the number of workspace bits needed to compute f. (Below we will ignore the last m qubits of the register, since they will be always set to all zeroes except in intermediate steps of f's computation.) The algorithm first uses n Hadamard operations to set the first n qubits to the uniform state and then apply the operation $|xz\rangle \mapsto |x(z \oplus f(x)\rangle$ to the register, resulting (up to normalization) in the state

$$\sum_{x\in\{0,1\}^n} |x\rangle\,|f(x)\rangle = \sum_{x\in\{0,1\}^n} (|x\rangle + |x \oplus a\rangle)\,|f(x)\rangle\,. \tag{2}$$

We then *measure* the second n bits of the register, collapsing its state to

$$|xf(x)\rangle + |(x \oplus a)f(x)\rangle \tag{3}$$

for some string x (that is chosen uniformly from $\{0,1\}^n$). You might think that we're done as the state (3) clearly encodes a, however we cannot directly learn a from this state: if we measure the first n bits we will get with probability $1/2$ the value x and with probability $1/2$ the value $x \oplus a$. Even though a can be deduced from these two values combined, each one of them on its own yields no information about a. (This point is well worth some contemplation, as it underlies the subtleties involved in quantum computation and demonstrates why a

quantum algorithm is *not* generally equivalent to performing exponentially many classical computation in parallel.)

However, consider now what happens if we perform another n Hadamard operations on the first n bits. Since this maps x to the vector $\sum_y (-1)^{x \odot y} |y\rangle$, the new state of the first n bits will be

$$\sum_y \left((-1)^{x \odot y} + (-1)^{(x \oplus a) \odot y} \right) |y\rangle = \sum_y \left((-1)^{x \odot y} + (-1)^{x \odot y} (-1)^{a \odot y} \right) |y\rangle . \quad (4)$$

For every $y \in \{0,1\}^m$, the y^{th} coefficient in the state (4) is nonzero if and only if if and only if $a \odot y = 0$, and in fact if measured, the state (4) yields a uniform $y \in \{0,1\}^n$ satisfying $a \odot y = 0$.

Repeating the entire process k times, we get k uniform strings $y_1, \ldots, y_k$ satisfying $y \odot a = 0$ or in other words, k *linear equations* (over the field GF(2)) on the variables $a_1, \ldots, a_n$. It can be easily shown that if, say, $k \geq 2n$ then with high probability there will be $n-1$ linearly independent equations among these (see Exercise 10.9), and hence we will be able to retrieve a from these equations using Gaussian elimination. This completes the proof of Theorem 10.14. For completeness, a full description of Simon's algorithm can be found in Figure 10.5. ■

Simon's Algorithm.

Goal: Given a polynomial-time computable $f : \{0,1\}^n \to \{0,1\}^n$ such that there is some $a \in \{0,1\}^n$ satisfying $f(x) = f(y)$ iff $y = x \oplus a$ for every $x, y \in \{0,1\}^n$, find a.

Quantum register: We use an $2n+m$-qubit register, where m is large enough so we can compute the transformation $|xz0^m\rangle \mapsto |x(z \oplus f(x))0^m\rangle$. (Below we ignore the last m qubits of the register as they will always contain 0^m except in intermediate computations of f.)

Operation	**State** (neglecting normalizing factors)
	Initial state: $\left|0^{2n}\right\rangle$
Apply Hadamard operation to first n qubits.	$\sum_x |x0^n\rangle$
Compute $\|xz\rangle \mapsto \|x(y \oplus f(x))\rangle$	$\sum_x |xf(x)\rangle = \sum_x (|x\rangle + |x \oplus a\rangle) |f(x)\rangle$
Measure second n bits of register.	$(|x\rangle + |x \oplus a\rangle) |f(x)\rangle$
Apply Hadamard to first n bits.	$\left(\sum_y (-1)^{x \odot y} (1 + (-1)^{a \odot y}) |y\rangle\right) |f(x)\rangle$ = $2 \sum_{y: a \odot y = 0} (-1)^{x \odot y} |y\rangle |f(x)\rangle$
Measure first n qubits of register to obtain a value y such that $y \odot a = 0$. Repeat until we get a sufficient number of linearly independent equations on a.	

Figure 10.5 Simon's Algorithm

10.6 Shor's algorithm: integer factorization using quantum computers

The *integer factorization* problem is to find, given an integer N, the set of all *prime factors* of N (i.e., prime numbers that divide N). By a polynomial-time algorithm for this problem we mean an algorithm that runs in time polynomial in the description of N, i.e., poly$(\log(N))$ time. Although people have thought about factorization for at least 2000 years, we still do not know of a polynomial-time algorithm for it: the best classical algorithm takes roughly $2^{(\log N)^{1/3}}$ steps to factor N [LLMP90]. In fact, the presumed difficulty of this problem underlies many popular encryption schemes (such as RSA, see Section 9.2.1). Therefore, it was quite a surprise when in 1994 Peter Shor showed the following result, which is now the

most famous algorithm for quantum computers, and the strongest evidence that **BQP** may contain problems outside of **BPP**.

Theorem 10.15 *(Shor's Algorithm: Factoring in* **BQP** [Sho97]*)*
There is a quantum algorithm that given a number N, runs in time $\mathrm{poly}(\log(N))$ *and outputs the prime factorization of N.*

Shor's ideas in nutshell. The algorithm uses the following observations. First, since N has at most $\log N$ factors, it clearly suffices to show how to find a *single* factor of N in $\mathrm{poly}(\log N)$ time because we can then repeat the algorithm with N divided by that factor, and thus find all factors. Second, it is a well-known fact that in order to find a single factor, it suffices to be able to find the *order* of a random number $A \pmod N$, in other words, the smallest r such that $A^r \equiv 1 \pmod N$. This is detailed in Section 10.6.4, but the idea is that with good probability, the order r of A will be even and furthermore $A^{r/2-1}$ will have a non-trivial common factor with N, which we can find using a GCD (greatest common divisor) computation. Third, the mapping $A \mapsto A^x \pmod N$ is computable in $\mathrm{poly}(\log N)$ time even on classical TMs (and so in particular by quantum algorithms) using *fast exponentiation*; see Exercise 10.10.

Using these observations we can come up with a simple $\mathrm{polylog}(N)$-time quantum algorithm that transforms a quantum register initialized to all zeros into the state that is the uniform superposition of all states of the type $|x\rangle$, where $x \leq N$ and satisfies $A^x \equiv y_0 \pmod N$ for some randomly chosen $y_0 \leq N-1$. By elementary number theory, the set of such x's form an *arithmetic progression* of the type $x_0 + ri$ for $i = 1, 2, \ldots$ where $A^{x_0} \equiv y_0 \pmod N$ and r is the order of A.

By now the problem is beginning to look quite a bit like Simon's problem, since we have created a quantum state involving a strong periodicity (namely, an arithmetic progression) and we are interested in determining its period. In engineering and mathematics, a classical tool for detecting periods is the Fourier Transform (see Section 10.6.1). Below, we describe the *quantum Fourier transform (QFT)*, which allows us to detect periods in a quantum state. This is a quantum algorithm that takes a register from some arbitrary state $f \in \mathbb{C}^M$ into a state whose vector is the Fourier transform $\hat{f}$ of f. The QFT takes only $O(\log^2 M)$ elementary steps and is thus very efficient. Note that we cannot say that this algorithm "computes" the Fourier transform, since the transform is stored in the amplitudes of the state, and as mentioned earlier, quantum mechanics give no way to "read out" the amplitudes per se. The only way to get information from a quantum state is by *measuring* it, which yields a single basis state with probability that is related to its amplitude. This is hardly representative of the entire fourier transform vector, but sometimes (as is the case in Shor's algorithm) this is enough to get highly non-trivial information, which we do not know how to obtain using classical (non-quantum) computers.

10.6.1 The Fourier transform over $\mathbb{Z}_M$

We now define the *Fourier transform over* $\mathbb{Z}_M$ (the group of integers in $\{0, \ldots, M-1\}$ with addition modulo M). We give a definition that is specialized to the current context. For more discussion on the Fourier transform, a tool that has found numerous uses in complexity theory, see Chapter 22.

Definition 10.16 For every vector $f \in \mathbb{C}^M$, the *Fourier transform of f* is the vector $\hat{f}$ where the x^{th} coordinate of $\hat{f}$ is[7]

$$\hat{f}(x) = \tfrac{1}{\sqrt{M}} \sum_{y \in \mathbb{Z}_M} f(x)\omega^{xy} \,,$$

[7] In the context of Fourier transform it is customary and convenient to denote the x^{th} coordinate of a vector f by $f(x)$ rather than f_x.

where $\omega = e^{2\pi i/M}$. ♢

The Fourier transform is simply a representation of f in the *Fourier basis* $\{\chi_x\}_{x\in\mathbb{Z}_M}$, where χ_x is the vector/function whose y^{th} coordinate is $\frac{1}{\sqrt{M}\omega^{xy}}$. Now the inner product of any two vectors χ_x, χ_z in this basis is equal to

$$\langle \chi_x, \chi_z \rangle = \frac{1}{M} \sum_{y\in\mathbb{Z}_M} \omega^{xy}\overline{\omega^{zy}} = \frac{1}{M} \sum_{y\in\mathbb{Z}_M} \omega^{(x-z)y} .$$

But if $x = z$ then $\omega^{(x-z)} = 1$ and hence this sum is equal to 1. On the other hand, if $x \neq z$, then this sum is equal to $\frac{1}{M}\frac{1-\omega^{(x-y)M}}{1-\omega^{x-y}} = \frac{1}{M}\frac{1-1}{1-\omega^{x-y}} = 0$ using the formula for the sum of a geometric series. In other words, this is an *orthonormal* basis which means that the Fourier transform map $f \mapsto \hat{f}$ is a *unitary* operation.

What is so special about the Fourier basis? For one thing, if we identify vectors in $\mathbb{C}^M$ with functions mapping $\mathbb{Z}_M$ to $\mathbb{C}$, then it's easy to see that every function χ in the Fourier basis is a *homomorphism* from $\mathbb{Z}_M$ to $\mathbb{C}$ in the sense that $\chi(y+z) = \chi(y)\chi(z)$ for every $y, z \in \mathbb{Z}_M$. Also, every function χ is *periodic* in the sense that there exists $r \in \mathbb{Z}_M$ such that $\chi(y+r) = \chi(z)$ for every $y \in \mathbb{Z}_M$ (indeed if $\chi(y) = \omega^{xy}$ then we can take r to be ℓ/x where ℓ is the least common multiple of x and M). Thus, intuitively, if a function $f : \mathbb{Z}_M \to \mathbb{C}$ is itself periodic (or roughly periodic) then when representing f in the Fourier basis, the coefficients of basis vectors with periods agreeing with the period of f should be large, and so we might be able to discover f's period from this representation. This does turn out to be the case, and is a crucial point in Shor's algorithm.

Fast Fourier Transform.

Denote by FT_M the operation that maps every vector $f \in \mathbb{C}^M$ to its Fourier transform $\hat{f}$. The operation FT_M is represented by an $M \times M$ matrix whose (x,y)th entry is ω^{xy}. The trivial algorithm to compute it takes M^2 operations. The famous *Fast Fourier Transform* (FFT) algorithm computes the Fourier transform in $O(M \log M)$ operations. We now sketch the idea behind this algorithm as the same idea will be used in the *quantum* Fourier transform algorithm described in Section 10.6.2.

Note that

$$\hat{f}(x) = \frac{1}{\sqrt{M}} \sum_{y\in\mathbb{Z}_M} f(y)\omega^{xy} = \frac{1}{\sqrt{M}} \sum_{y\in\mathbb{Z}_M, y \text{ even}} f(y)\omega^{-2x(y/2)} + \omega^x \frac{1}{\sqrt{M}} \sum_{y\in\mathbb{Z}_M, y \text{ odd}} f(y)\omega^{2x(y-1)/2} .$$

Now since ω^2 is an $M/2$th root of unity and $\omega^{M/2} = -1$, letting W be the $M/2 \times M/2$ diagonal matrix with diagonal entries $\omega^0, \ldots, \omega^{M/2-1}$, we get that

$$FT_M(f)_{low} = FT_{M/2}(f_{even}) + WFT_{M/2}(f_{odd}) \quad (5)$$

$$FT_M(f)_{high} = FT_{M/2}(f_{even}) - WFT_{M/2}(f_{odd}) \quad (6)$$

where for an M-dimensional vector $\mathbf{v}$, we denote by $\mathbf{v}_{even}$ (resp. $\mathbf{v}_{odd}$) the $M/2$-dimensional vector obtained by restricting $\mathbf{v}$ to the coordinates whose indices have least significant bit equal to 0 (resp. 1) and by $\mathbf{v}_{low}$ (resp. $\mathbf{v}_{high}$) the restriction of $\mathbf{v}$ to coordinates with most significant bit 0 (resp. 1).

Equations (5) and (6) are the crux of the divide-and-conquer idea of the FFT algorithm, since they allow to replace a size-M problem with two size-$M/2$ subproblems, leading to a recursive time bound of the form $T(M) = 2T(M/2) + O(M)$ which solves to $T(M) = O(M \log M)$.

10.6.2 Quantum Fourier Transform over $\mathbb{Z}_M$

The *quantum Fourier transform* is an algorithm to change the state of a quantum register from $f \in \mathbb{C}^M$ to its Fourier transform $\hat{f}$.

Lemma 10.17 *(Quantum Fourier Transform* [BV93]*)*
For every m and $M = 2^m$ there is a quantum algorithm that uses $O(m^2)$ elementary quantum operations and transforms a quantum register in state $f = \sum_{x\in\mathbb{Z}_m} f(x)|x\rangle$ into the state $\hat{f} = \sum_{x\in\mathbb{Z}_M} \hat{f}(x)|x\rangle$, where $\hat{f}(x) = \frac{1}{\sqrt{M}} \sum_{y\in\mathbb{Z}_m} \omega^{xy} f(x)$.

PROOF: The crux of the algorithm is Equations (5) and (6), which allow the problem of computing FT_M, the problem of size M, to be split into two identical subproblems of size $M/2$ involving computation of $FT_{M/2}$, which can be carried out recursively using the same elementary operations. (Aside: Not every divide-and-conquer classical algorithm can be implemented as a fast quantum algorithm; we are really using the structure of the problem here.)

Quantum Fourier Transform FT_M
Initial state: $f = \sum_{x\in\mathbb{Z}_M} f(x)|x\rangle$
Final state: $\hat{f} = \sum_{x\in\mathbb{Z}_M} \hat{f}(x)|x\rangle$.

Operation	**State** (neglecting normalizing factors)
	$f = \sum_{x\in\mathbb{Z}_M} f(x)\|x\rangle$
Recursively run $FT_{M/2}$ on $m-1$ most significant qubits	$(FT_{M/2}f_{even})\|0\rangle + (FT_{M/2}f_{odd})\|1\rangle$
If LSB is 1 then compute W on $m-1$ most significant qubits (see below).	$(FT_{M/2}f_{even})\|0\rangle + (WFT_{M/2}f_{odd})\|1\rangle$
Apply Hadmard gate H to least significant qubit.	$(FT_{M/2}f_{even})(\|0\rangle + \|1\rangle) + (WFT_{M/2}f_{odd})(\|0\rangle - \|1\rangle) =$ $(FT_{M/2}f_{even} + WFT_{M/2}f_{odd})\|0\rangle + (FT_{M/2}f_{even} - WFT_{M/2}f_{odd})\|1\rangle$
Move LSB to the most significant position	$\|0\rangle(FT_{M/2}f_{even} + WFT_{M/2}f_{odd}) + \|1\rangle(FT_{M/2}f_{even} - WFT_{M/2}f_{odd}) = \hat{f}$

The transformation W on $m-1$ qubits can be defined by $|x\rangle \mapsto \omega^x = \omega^{\sum_{i=0}^{m-2} 2^i x_i}$ (where x_i is the i^{th} qubit of x). It can be easily seen to be the result of applying for every $i \in \{0,\ldots,m-2\}$ the following elementary operation on the i^{th} qubit of the register: $|0\rangle \mapsto |0\rangle$ and $|1\rangle \mapsto \omega^{2^i}|1\rangle$.

The final state is equal to $\hat{f}$ by (5) and (6). (We leave verifying this and the running time to Exercise 10.14.) ■

10.6.3 Shor's Order-Finding Algorithm.

We now present the central step in Shor's factoring algorithm: a quantum polynomial-time algorithm to find the *order* of an integer A modulo an integer N.

Lemma 10.18 *There is a polynomial-time quantum algorithm that on input A, N (represented in binary) finds the smallest r such that $A^r = 1 \pmod N$.* ♢

PROOF: Let $m = \lceil 5 \log M \rceil$ and let $M = 2^m$. Our register will consist of $m + \text{polylog}(N)$ qubits. Note that the function $x \mapsto A^x \pmod N$ can be computed in $\text{polylog}(N)$ time (see Exercise 10.10) and so we will assume that we can compute the map $|x\rangle|y\rangle \mapsto |x\rangle|y \oplus \llcorner A^x \pmod N \lrcorner\rangle$ (where $\llcorner X \lrcorner$ denotes the representation of the number $X \in \{0,\ldots,N-1\}$ as a binary string of length $\log N$).[8] Now we describe the order-finding algorithm. It uses a tool of elementary number theory called *continued fractions* which allows us to approximate

[8] To compute this map we may need to extend the register by some additional $\text{polylog}(N)$ many qubits, but we can ignore them as they will always be equal to zero except in intermediate computations.

(using a classical algorithm) an arbitrary real number α with a rational number p/q where there is a prescribed upper bound on q (see Section 10.6.5).

Order finding algorithm.

Goal: Given numbers N and $A < N$ such that $gcd(A, N) = 1$, find the smallest r such that $A^r = 1 \pmod N$.

Quantum register: We use an $m + \text{polylog}(N)$-qubit register, where $m = \lceil 5 \log N \rceil$. Below we treat the first m bits of the register as encoding a number in $\mathbb{Z}_M$.

Operation	**State** (*including* normalizing factors)
Apply Fourier transform to the first m bits.	$\frac{1}{\sqrt{M}} \sum_{x \in \mathbb{Z}_M} \|x\rangle \|0^n\rangle$
Compute the transformation $\|x\rangle\|y\rangle \mapsto \|x\rangle\|y \oplus (A^x \pmod N)\rangle$.	$\frac{1}{\sqrt{M}} \sum_{x \in \mathbb{Z}_M} \|x\rangle \|A^x \pmod N\rangle$
Measure the second register to get a value y_0.	$\frac{1}{\sqrt{K}} \sum_{\ell=0}^{K-1} \|x_0 + \ell r\rangle \|y_0\rangle$ where x_0 is the smallest number such that $A^{x_0} = y_0 \pmod N$ and $K = \lfloor (M - 1 - x_0)/r \rfloor$.
Apply the Fourier transform to the first register.	$\frac{1}{\sqrt{M}\sqrt{K}} \left(\sum_{x \in \mathbb{Z}_n} \sum_{\ell=0}^{K-1} \omega^{(x_0 + \ell r)x} \|x\rangle \right) \|y_0\rangle$

Measure the first register to obtain a number $x \in \mathbb{Z}_M$. Find a rational approximation a/b with a, b coprime and $b \leq N$ that approximates the number $\frac{x}{M}$ within $1/(10M)$ accuracy (see Section 10.6.5). If found such approximation and $A^b = 1 \pmod N$ then output b.

In the analysis, it will suffice to show that this algorithm outputs the order r with probability at least $\Omega(1/\log N)$ (we can always amplify the algorithm's success by running it several times and taking the smallest output).

Analysis: the case that $r|M$

We start by analyzing the algorithm in the case that $M = rc$ for some integer c. Though very unrealistic (remember that M is a power of 2!) this gives the intuition why Fourier transforms are useful for detecting periods.

CLAIM: *In this case the value x measured will be equal to ac for a random $a \in \{0, \ldots, r-1\}$.*

The claim concludes the proof since it implies that $x/M = a/r$ where a is random integer less than r. Now for every r, at least $\Omega(r/\log r)$ of the numbers in $[r-1]$ are co-prime to r. Indeed, the prime number theorem (see Section A.3 in the appendix) says that there at least this many primes in this interval, and since r has at most $\log r$ prime factors, all but $\log r$ of these primes are co-prime to r. Thus, when the algorithm computes a rational approximation for x/M, the denominator it will find will indeed be r.

To prove the claim, we compute for every $x \in \mathbb{Z}_M$ the absolute value of $|x\rangle$'s coefficient before the measurement. Up to some normalization factor this is

$$\left| \sum_{\ell=0}^{c-1} \omega^{(x_0 + \ell r)x} \right| = \left| \omega^{x_0 c' c} \right| \left| \sum_{\ell=0}^{c-1} \omega^{r\ell x} \right| = 1 \cdot \left| \sum_{\ell=0}^{c-1} \omega^{r\ell x} \right| . \tag{7}$$

If c does not divide x then ω^r is a c^{th} root of unity, so $\sum_{\ell=0}^{c-1} w^{r\ell x} = 0$ by the formula for sums of geometric progressions. Thus, such a number x would be measured with zero probability. But if $x = cj$ then $\omega^{r\ell x} = w^{rcj\ell} = \omega^{Mj} = 1$, and hence the amplitudes of all such x's are equal for all $j \in \{0, 2, \ldots, r-1\}$. ■

The general case

In the general case, where r does not necessarily divide M, we will not be able to show that the measured value x satisfies $M|xr$. However, we will show that with $\Omega(1/\log r)$ probability, **(1)** xr will be "almost divisible" by M in the sense that $0 \leq xr \pmod M < r/10$ and **(2)** $\lfloor xr/M \rfloor$ is coprime to r. Condition **(1)** implies that $|xr - cM| < r/10$ for $c = \lfloor xr/M \rfloor$.

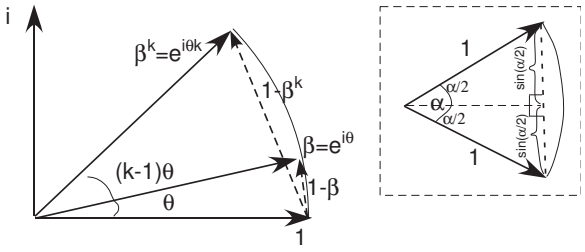


Figure 10.6 A complex number $z = a + ib$ can be thought of as the two-dimensional vector (a, b) of length $|z| = \sqrt{a^2 + b^2}$. The number $\beta = e^{i\theta}$ corresponds to a unit vector of angle θ from the x axis.For any such β, if k is not too large (say $k < 1/\theta$) then by elementary geometric considerations $\frac{|1-\beta^k|}{|1-\beta|} = \frac{2\sin(\theta/2)}{2\sin(k\theta/2)}$. We use here the fact (proved in the dotted box above) that in a unit cycle, the chord corresponding to an angle α is of length $2\sin(\alpha/2)$.

Dividing by rM gives $\left|\frac{x}{M} - \frac{c}{r}\right| < \frac{1}{10M}$. Therefore, $\frac{c}{r}$ is a rational number with denominator at most N that approximates $\frac{x}{M}$ to within $1/(10M) < 1/(4N^4)$. It is not hard to see that such an approximation is unique (Exercise 10.11) and hence in this case the algorithm will come up with c/r and output the denominator r (see Section 10.6.5).

Thus all that is left is to prove the next two lemmas. The first shows that there are $\Omega(r/\log r)$ values of x that satisfy the above two conditions and the second shows that each is measured with probability $\Omega((1/\sqrt{r})^2) = \Omega(1/r)$.

Lemma 10.19 *There exist $\Omega(r/\log r)$ values $x \in \mathbb{Z}_M$ such that:*

1. $0 < xr \pmod M < r/10$
2. *$\lfloor xr/M \rfloor$ and r are coprime* ♢

Lemma 10.20 *If x satisfies $0 < xr \pmod M < r/10$ then, before the measurement in the final step of the order-finding algorithm, the coefficient of $|x\rangle$ is at least $\Omega(\frac{1}{\sqrt{r}})$.* ♢

Proof of Lemma 10.19: We prove the lemma for the case that r is coprime to M, leaving the general case as Exercise 10.15. In this case, the map $x \mapsto rx \pmod M$ is a permutation of $\mathbb{Z}_M^*$. There are at least $\Omega(r/\log r)$ numbers in $[1..r/10]$ that are coprime to r (take primes in this range that are not one of r's at most $\log r$ prime factors) and hence $\Omega(r/\log r)$ numbers x such that $rx \pmod M = xr - \lfloor xr/M \rfloor M$ is in $[1..r/10]$ and coprime to r. But this means that $\lfloor rx/M \rfloor$ can not have a nontrivial shared factor with r, as otherwise this factor would be shared with $rx \pmod M$ as well. ■

Proof of Lemma 10.20: Let x be such that $0 < xr \pmod M < r/10$. The absolute value of $|x\rangle$'s coefficient in the state before the measurement is

$$\frac{1}{\sqrt{K}\sqrt{M}} \left| \sum_{\ell=0}^{K-1} \omega^{\ell r x} \right| , \tag{8}$$

where $K = \lfloor (M - x_0 - 1)/r \rfloor$. Note that $\frac{M}{2r} < K < \frac{M}{r}$ since $x_0 < N \ll M$.

Setting $\beta = \omega^{rx}$ (note that since $M \not| rx$, $\beta \neq 1$) and using the formula for the sum of a geometric series, this is at least

$$\frac{\sqrt{r}}{2M} \left| \frac{1-\beta^{\lceil M/r \rceil}}{1-\beta} \right| = \frac{\sqrt{r}}{2M} \frac{\sin(\theta \lceil M/r \rceil/2)}{\sin(\theta/2)} , \tag{9}$$

where $\theta = \frac{rx \pmod M}{M}$ is the angle such that $\beta = e^{i\theta}$ (see Figure 10.6 for a proof by picture of the last equality). Under our assumptions $\lceil M/r \rceil \theta < 1/10$ and hence (using the fact that $\sin\alpha \sim \alpha$ for small angles α), the coefficient of x is at least $\frac{\sqrt{r}}{4M} \lceil M/r \rceil \geq \frac{1}{8\sqrt{r}}$ ■

This completes the proof of Lemma 10.18 ■

10.6.4 Reducing factoring to order finding.

The reduction of the factoring problem to the order-finding problem is classical (in particular, predates quantum computing) and follows from the following two Lemmas:

Lemma 10.21 *For every nonprime N that is not a prime power, the probability that a random X in the set $\mathbb{Z}_N^* = \{X \in [N-1] : gcd(X,N) = 1\}$ has an even order r and furthermore, $X^{r/2} \neq -1 \pmod N$ is at least $1/4$.* ♢

Lemma 10.22 *For every N and Y, if $Y^2 = 1 \pmod N$ but $Y \pmod N \notin \{+1,-1\}$ then $gcd(Y-1,N) \notin 1, N$.* ♢

Together, lemmas 10.21 and 10.22 show that given a composite N that is not a prime power if we choose A at random in $[N-1]$ then with good probability either $gcd(A,N)$ or $gcd(A^{r/2}-1,N)$ will yield a non-trivial factor F of N. We can then use recursion to find the prime factors of F and N/F respectively, leading to a polylog(N) time factorization algorithm. (Note that if N is a prime power then it is easy to find its factorization by simply going over all $\ell \in [\log N]$ and trying the ℓ^{th} root of N.) Thus to prove Theorem 10.15 all that is left is to prove lemmas 10.21 and 10.22. The proofs rely on some basic facts from number theory; see Section A.3 in the Appendix for a quick review.

Proof of Lemma 10.22: Under our assumptions, N divides $Y^2 - 1 = (Y-1)(Y+1)$ but does not divide neither $Y-1$ or $Y+1$. But this means that $gcd(Y-1,N) > 1$ since if $Y-1$ and N were coprime, then since N divides $(Y-1)(Y+1)$, it would have to divide $Y+1$ (Exercise 10.12). Since $Y-1 < N$, obviously $gcd(Y-1,N) < N$ and hence we're done. ■

Proof of Lemma 10.21: We prove the lemma for the case $N = PQ$ for primes P, Q: the proof can be suitably generalized for every N. Now, by the Chinese Remainder Theorem every $X \in \mathbb{Z}_N^*$ is isomorphic to the pair $\langle X \pmod P, X \pmod Q\rangle$. In particular, choosing a random number $X \in \mathbb{Z}_N^*$ is equivalent to choosing two random numbers Y, Z in $\mathbb{Z}_P^*$ and $\mathbb{Z}_Q^*$ respectively and setting X to be the unique number corresponding to the pair $\langle Y, Z\rangle$. Now for every k, $X^k \pmod N$ is isomorphic to $\langle Y^k \pmod P, Z^k \pmod Q\rangle$ and so the *order* of X is the least common multiple of the orders of Y and Z modulo P and Q respectively. We will complete the proof by showing that with probability at least $1/2$, the order of Y is even: a number of the form $2^k c$ for $k \geq 1$ and c odd. We then show that with probability at least $1/2$, the order of Z has the form $2^\ell d$ for d odd and $\ell \neq k$. This implies that the order of X is $r = 2^{\max\{k,\ell\}} lcm(c,d)$ (where lcm denotes the least common multiple) which, means that $X^{r/2}$ will be equal to 1 in at least one coordinate. Since $-1 \pmod N$ is isomorphic to the tuple $\langle -1, -1\rangle$ this means that $X^{r/2} \neq -1 \pmod P$.

Thus all that is left is to prove the following:

- Y has even order with probability at least $1/2$.

  Indeed, the set of numbers in $\mathbb{Z}_P^*$ with odd order is a *subgroup* of $\mathbb{Z}_P^*$: if Y, Y' have odd orders r, r' respectively then $(YY')^{rr'} = 1 \pmod P$, which means that the order of YY' divides the odd number rr'. Yet -1 has even order, implying that this is a *proper* subgroup of $\mathbb{Z}_P^*$, taking at most $1/2$ of $\mathbb{Z}_P^*$.

- There is a number ℓ_0 such that with probability exactly $1/2$, the order of of a random $Z \in \mathbb{Z}_Q^*$ is a number of the form $2^\ell c$ for $\ell \leq \ell_0$. (This implies that for every fixed k, the probability that the order has the form $2^k d$ is at most $1/2$.)

  For every ℓ, define G_ℓ to be the subset of $\mathbb{Z}_Q^*$ whose order modulo Q is of the form $2^j c$ where $j \leq \ell$ and c is odd. It can be verified that for every ℓ, G_ℓ is a subgroup of $G_{\ell+1}$ and furthermore, because modulo a prime P the mapping $x \mapsto x^2 \pmod P$ is two-to-one and maps $G_{\ell+1}$ into G_ℓ (Exercise 10.13), $|G_\ell| \geq |G_{\ell+1}|/2$. It follows that if we take ℓ_0 to be the largest such that G_{ℓ_0} is a proper subgroup of $\mathbb{Z}_P^*$, then $|G_{\ell_0}| = |Z_P^*|/2$.

■

10.6.5 Rational approximation of real numbers

In many settings, including Shor's algorithm, we are given a real number in the form of a program that can compute its first t bits in poly(t) time. We are interested in finding a close approximation to this real number of the form a/b, where there is a prescribed upper bound on b. Continued fractions is a tool in number theory that is useful for this.

A *continued fraction* is a number of the following form:

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \ldots}}}$$

for a_0 a non-negative integer and $a_1, a_2, \ldots$ positive integers.

Given a real number $\alpha > 0$, we can find its representation as an *infinite* fraction as follows: split α into the integer part $\lfloor \alpha \rfloor$ and fractional part $\alpha - \lfloor \alpha \rfloor$, find recursively the representation R of $1/(\alpha - \lfloor \alpha \rfloor)$, and then write

$$\alpha = \lfloor \alpha \rfloor + \frac{1}{R} .$$

If we continue this process for n steps, we get a rational number, denoted by $[a_0, a_1, \ldots, a_n]$, which can be represented as $\frac{p_n}{q_n}$ with p_n, q_n coprime. The following facts can be proven using induction:

- $p_0 = a_0, q_0 = 1$ and for every $n > 1$, $p_n = a_n p_{n-1} + p_{n-2}$, $q_n = a_n q_{n-1} + q_{n-2}$.
- $\frac{p_n}{q_n} - \frac{p_{n-1}}{q_{n-1}} = \frac{(-1)^{n-1}}{q_n q_{n-1}}$

Furthermore, it is known that

$$\left| \frac{p_n}{q_n} - \alpha \right| < \frac{1}{q_n q_{n+1}} , \tag{10}$$

which implies that $\frac{p_n}{q_n}$ is the *closest* rational number to α with denominator at most q_n. It also means that if α is extremely close to a rational number, say, $\left|\alpha - \frac{a}{b}\right| < \frac{1}{4b^4}$ for some coprime a, b then we can find a, b by iterating the continued fraction algorithm for polylog(b) steps. Indeed, let q_n be the first denominator such that $q_{n+1} \geq b$. If $q_{n+1} > 2b^2$ then (10) implies that $\left|\frac{p_n}{q_n} - \alpha\right| < \frac{1}{2b^2}$. But this means that $\frac{p_n}{q_n} = \frac{a}{b}$ since there is at most one rational number of denominator at most b that is so close to α. On the other hand, if $q_{n+1} \leq 2b^2$ then since $\frac{p_{n+1}}{q_{n+1}}$ is closer to α than $\frac{a}{b}$, $\left|\frac{p_{n+1}}{q_{n+1}} - \alpha\right| < \frac{1}{4b^4}$, again meaning that $\frac{p_{n+1}}{q_{n+1}} = \frac{a}{b}$. It's not hard to verify that $q_n \geq 2^{n/2}$, implying that p_n and q_n can be computed in polylog(q_n) time.

10.7 BQP and classical complexity classes

What is the relation between **BQP** and the classes we already encountered such as **P**, **BPP** and **NP**? This is very much an open questions. It not hard to show that quantum computers are at least not infinitely powerful compared to classical algorithms:

Theorem 10.23 **BQP $\subseteq$ PSPACE** ♢

Proof Sketch: To simulate a T-step quantum computation on an m qubit register, we need to come up with a procedure `Coeff` that for every $i \in [T]$ and $x \in \{0,1\}^m$, the x^{th} coefficient (up to some accuracy) of the register's state in the i^{th} execution. We can compute `Coeff` on inputs x, i using at most 8 recursive calls to `Coeff` on inputs $x', i-1$ (for the at most 8 strings that agree with x on the three bits that the F_i's operation reads and modifies). Since we can reuse the space used by the recursive operations, if we let $S(i)$

denote the space needed to compute $\texttt{Coeff}(x,i)$ then $S(i) \leq S(i-1) + O(\ell)$ (where ℓ is the number of bits used to store each coefficient).

To compute, say, the probability that if measured after the final step the first qubit of the register is equal to 1, just compute the sum of $\texttt{Coeff}(x,T)$ for every $x \in \{0,1\}^n$. Again, by reusing the space of each computation this can be done using polynomial space. ■

In Exercise 17.7 later in the book you are asked to improve Theorem 10.23 to show that $\mathbf{BQP} \subseteq \mathbf{P}^{\#\mathbf{P}}$ (where $\#\mathbf{P}$ is the counting version of **NP** described in Chapter 17). One can even show show $\mathbf{BQP} \subseteq \mathbf{PP}$ [ADH97] (see Definition 17.6). But these are essentially the best bounds we know on **BQP**.

Does **BQP** = **BPP**? The main reason to believe this is false is the polynomial-time quantum algorithm for integer factorization, whereas no similar algorithm is believed to exist for probabilistic computation. Although this is not as strong as the evidence for, say $\mathbf{NP} \not\subseteq \mathbf{BPP}$ (after all **NP** contains thousands of well-studied problems that have resisted efficient algorithms), the factorization problem is one of the oldest and most well-studied computational problems, and the fact that we still know no efficient algorithm for it makes the conjecture that none exists appealing. Also note that unlike other famous problems that eventually found an algorithm (e.g., linear programming [Kha79] and primality testing [AKS04]), we do not even have a heuristic algorithm that is conjectured to work (even without proof) or experimentally works on, say, numbers that are product of two random large primes.

What is the relation between **BQP** and **NP**? It seems that quantum computers only offer a quadratic speedup (using Grover's search) on **NP**-complete problems. There are also oracle results showing that **NP** problems require exponential time on quantum computersl [BBBV97]. So most researchers believe that $\mathbf{NP} \not\subseteq \mathbf{BPP}$. On the other hand, there is a problem in **BQP** (the Recursive Fourier Sampling or RFS problem [BV93]) that is not known to be in the polynomial-hierarchy, let alone in **NP**. Thus it seems that **BQP** and **NP** may be incomparable classes.

10.7.1 Quantum analogs of NP and AM

Can we define an analog of **NP** in the quantum computing world? The class **NP** was defined using the notion of a certificate that is checked by a deterministic polynomial-time (classical) TM. However, quantum computation includes probabilistic classical computation as a subcase. Therefore the correct classical model to look at is the one where the certificate is verified by a polynomial-time randomized algorithm, namely, **MA** (see Definition 8.10). Thus the quantum analog of **NP** is denoted by **QMA**. More generally, one can define *quantum interactive proofs*, which generalize the definition of $\mathbf{AM}[k]$. These turn out to be surprisingly powerful. Three-round quantum interactive proofs suffice to capture **PSPACE**, as shown by Watrous [Wat03]. If the same were true of classical interactive proofs, then **PH** would collapse.

A "Quantum Cook-Levin Theorem" was proven by Kitaev (unpublished, see Umesh Vazirani's lecture notes, which are linked from this book's website). This shows that a quantum analog of 3SAT, called Q 5SAT, is *complete* for **QMA**. In this problem are given m elementary quantum operations $H_1, H_2, \ldots, H_m$ on an n-bit quantum register. Each operation acts upon only 5 bits of the register (and hence is represented by a $2^5 \times 2^5$ matrix, which implicitly defines a $2^n \times 2^n$ matrix). Let H be the $2^n \times 2^n$ matrix $\sum_j H_j$. We are promised that either all eigenvalues of H are $\geq b$ or there is an eigenvalue of H that is $\leq a$ where $0 \leq a \leq b \leq 1$ and $b-a$ is at least $1/n^c$ where c is a constant. We have to determine which case holds.

The reader could try to prove this completeness result as an exercise. As a warmup, first show how to reduce 3SAT to Q 5SAT.

Chapter notes and history

Since a quantum computer is reversible (Lemma 10.7), an important precursor of quantum computing was a field called *reversible computing* [Ben87], which seeks to find thermodynamic limits to the speed of classical computers. Toffoli's gate was invented in that context.

In 1982, Feynman [Fey82] pointed out that there seems to be no efficient simulation of quantum mechanics on classical Turing machines, and suggested that building quantum computers might allow us to run such simulations. (In fact, this still might be their most important application if they are ever built.) He also raised the possibility that quantum TMs may have more computational power than classical TMs. In 1985 Deutsch [Deu85] formally defined a quantum Turing machine, though in retrospect his definition is unsatisfactory. Better definitions then appeared in Deutsch-Josza [DJ92] and Bernstein-Vazirani [BV93]. The latter paper was the first to demonstrate the existence of a universal quantum TM that can simulate all other quantum TMs with only polynomial slowdown. Yao [Yao93] generalized these results to quantum circuits, and our definition of quantum computation follows Yao. (The Bernstein-Vazirani quantum TM model is known to be less noise-tolerant than the circuit model, and thus less likely to be realized.) Deutsch [Deu89] showed that a certain 3-qubit gate is *universal* for quantum circuits, while Solovay (unpublished manuscript, 1995) and, independently, Kitaev [Kit97], showed that universal gates can approximate every unitary matrix with precision exponentially small in the number of gates, yielding Theorem 10.12 (though we stated it with a particular universal basis mentioned in the book [NC00]).

Bernstein and Vazirani also introduced the quantum algorithm for computing the fourier transform, and gave evidence that it provides superpolynomial speedups over classical algorithms. The papers of Simon and Shor gave further evidence along these lines, and in particular Shor's paper caught the imagination of the scientific world, as well as of governments worldwide (who now feared for the security of their cryptosystems).

Quantum computation has a fascinating connection with cryptography. On the one hand, if quantum computers are ever built then Shor's algorithm and various generalizations thereof could be used to completely break the security of RSA and all other factoring or discrete-log based cryptosystems. On the other hand, it turns out that using quantum mechanics and the ideas underlying the EPR/Bell "paradox", it is possible to have *unconditionally secure* public key cryptography, a concept known as *quantum key distribution* [BB84] and more generally as *quantum cryptography*. That is, these cryptosystem are secure against even computationally unbounded adversaries. In fact, constructing these systems does not require the full-fledged power of quantum computers, and prototype implementations already exist. Still, there are very significant engineering challenges and issues that can compromise the real-world applicability and security of these systems. One should note however that even if quantum computers are built, it may very well be possible to still have conventional computational cryptography that is resistant even to polynomial-time quantum algorithms. For example, as far as we know quantum computers can at best invert one-way functions (Definition 9.4) quadratically faster than classical algorithms (using Grover's search). Thus, most researchers believe that *private key cryptography* (including even digital signatures!) will be just as resistant against quantum computers as it is against "classical" Turing machines. Even for public key cryptography, there are (few) candidates systems that are based on problems not known to have efficient quantum algorithms. Perhaps the most promising direction is basing such schemes on certain problems on *integer lattices* (see the notes for Chapter 9).

Grover's and Simon's algorithm actually operate in a more general model known as the *quantum black-box* model, in which an algorithm is given black-box access to an oracle computing the unitary transformation $|x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$ for some function f and tries to discover properties of f. There have been interesting upper bounds and lower bounds on the power of such algorithms. In particular, we know that Grover's algorithm is optimal in this model [BBBV97]. We also have several other "Grover-like" algorithms in this model; see the survey [Amb04]. One can view Grover's algorithm as evaluating an OR over $N = 2^n$-variables. Thus a natural question is whether it can be generalized into more general formulae; a particularly interesting special case is AND-OR trees (i.e., OR of ANDs of ORs ...) that arise in various applications such game strategies. This was question was open for a while, and in particular we didn't know if quantum algorithms can beat the best randomized algorithm for the full binary balanced AND-OR tree, which needs to look at $O(N^{\log(\frac{1+\sqrt{33}}{4})}) = O(N^{0.753..})$ variables [Sni81, SW86]. In a recent breakthrough, Farhi, Goldstone and Gutmann [FGG07] showed an $O(N^{1/2+o(1)})$-time quantum algorithm for this problem, a result that was generalized by Ambainis et al [ACR$^+$07] to hold for all AND-OR trees.

Research on quantum computing has generated some interesting insights on both "classical" computational complexity, and "non-computational" physics. A good example for a result of the first kind is the paper of Aharonov and Regev [AR04], that uses quantum insights to show a classical

computational complexity result (that a $\sqrt{n}$-approximation of the lattice shortest vector problem is in **coNP**). Examples for the results of the second kind include the works on quantum error correction (see below) and results on adiabatic computation [AvDK+04, AGIK07, vDMV01], that clarified this model and refuted some of the physicists' initial intuitions about it.

The chapter did not discuss the issue of *quantum error correction*, which tackles the following important issue: how can we run a quantum algorithm when at every possible step there is a probability of noise interfering with the computation? The issue is undoubtedly crucial, since an implementation of Shor's algorithms for interesting values of N requires hundreds of thousands of particles to stay in quantum superposition for large-ish periods of time. Thus far it is an open question whether this is practically achievable. Physicists' original intuition was that noise and decoherence will make quantum computing impractical; one obstacle cited was the *no-cloning theorem* [WZ82], which seems to rule out use of classical error-correction ideas in quantum computing. However, Shor's followup paper on *quantum error correction* [Sho95] contradicted this intuition and spurred much additional work. We now know that under reasonable noise models, so long as the probability of noise at a single step is lower than some constant threshold, one can perform arbitrarily long computations and get the correct answer with high probability; see the articles by Preskill [Pre97, Pre98]. Unfortunately, there are no estimates of the true noise rate in physical systems.

In fact it is unclear what the correct model of noise should be; this question is related to the issue of what is the reality underlying the quantum description of the world. Though the theory has had fantastic success in predicting experimental results (which perhaps is *the* criteria by which a physical theory is judged), some physicists are understandably uncomfortable with the description of nature as maintaining a huge array of possible states, and changing its behavior when it is observed. The popular science book [Bru04] contains a good (even if a bit biased) review of physicists' and philosophers' attempts at providing more palatable descriptions that still manage to predict experiments.

On a more technical level, while no one doubts that quantum effects exist at microscopic scales, scientists question why they do not manifest themselves at the macrosopic level (or at least not to human consciousness). Physicist Penrose [Pen90] has gone so far as to make a (highly controversial) suggestion about a link between human consciousness and the collapse of the probability wave. A *Scientific American* article by Yam [Yam97] describes various other explanations that have been advanced over the years, including *decoherence* (which uses quantum theory to explain the absence of macroscopic quantum effects) and *hidden variable theories* (which restore a deterministic order to world). No single explanation seems to please all researchers.

Finally, we note that since qubits are such a simple example of a quantum system, there is a growing movement to teach quantum mechanics using qubits and quantum computing rather than, say, the standard model of the hydrogen atom or electron-in-a-box. This is an interesting example of how the computational worldview (as opposed to computation in the sense of number-crunching) is seeping into the sciences.

For details of these and many other topics in quantum computing and information, see the books by Kitaev, Shen, and Vyalyi [KVS02] and Nielsen and Chuang [NC00]. Some excellent lecture notes and surveys can be found on the home pages of Umesh Vazirani and Scott Aaronson. Aaronson's Scientific American article [Aar08] provides an excellent popular-science exposition of the field.

Exercises

10.1 Show a quantum strategy that enables Alice and Bob to win the parity game of theorems 10.3 and 10.4 with probability 0.85.

10.2 Prove Claim 10.5. H454

10.3 For each one of the following operations: Hadamard, NOT, controlled-NOT, rotation by $\pi/4$, and Toffoli, write down the 8×8 matrix that describes the mapping induced by applying this operation on the first qubits of a 3-qubit register.

10.4 Define a linear function $F : \mathbb{R}^{2^m} \to \mathbb{R}^{2^m}$ to be an *elementary probabilistic operation* if it satisfies the following conditions:

- F is *stochastic*: that is, for every $\mathbf{v} \in \mathbb{R}_m$ such that $\sum_x \mathbf{v}_x = 1$, $\sum_x (A\mathbf{v})_x = 1$.
- F depends on at most three bits. That is, there is a linear function $G : \mathbb{R}^{2^3} \to \mathbb{R}^{2^3}$ and three

coordinates $i < j < k \in [m]$ such that for every vector of the form $|x_1 x_2 \cdots x_m\rangle$,

$$F|x_1 \cdots x_m\rangle = \sum_{a,b,c \in \{0,1\}} (G|x_i x_j x_k\rangle)_{abc} |x_1..x_{i-1} a x_{i+1}..x_{j-1} b x_{j+1}...x_{k-1} c x_{k+1}..x_m\rangle .$$

Let $f : \{0,1\}^* \to \{0,1\}$ and $T : \mathbb{N} \to \mathbb{N}$ be some functions. We say that f is *computable in probabilistic $T(n)$-time* if for every $n \in \mathbb{N}$ and $x \in \{0,1\}^n$, $f(x)$ can be computed by the following process:

(a) Initialize an m bit register to the state $|x0^{n-m}\rangle$ (i.e., x padded with zeroes), where $m \leq T(n)$.

(b) Apply one after the other $T(n)$ elementary operations $F_1, \ldots, F_T$ to the register (where we require that there is a polynomial-time TM that on input $1^n, 1^{T(n)}$ outputs the descriptions of $F_1, \ldots, F_T$).

(c) Measure the register and let Y denote the obtained value. (That is, if $\mathbf{v}$ is the final state of the register, then Y is a random variable that takes the value y with probability $\mathbf{v}_y$ for every $y \in \{0,1\}^n$.)

We require that the first bit of Y is equal to $f(x)$ with probability at least $2/3$.

Prove that a function $f : \{0,1\}^* \to \{0,1\}$ is computable in $p(n)$-probabilistic $p(n)$-time per the above definition for some polynomial p iff $f \in \mathbf{BPP}$.

10.5 Prove that if $f \in \mathbf{BQP}$ then f has a quantum polynomial-time algorithm in which all of the matrices are real— contain no numbers of the form $a + ib$ for $b \neq 0$. (This exercise can be thought of as showing that the power of quantum mechanics as opposed to classical probabilistic computation comes from the fact that we allow negative numbers in state representations, and not from the fact that we allow complex numbers.)

H454

10.6 Suppose that a two-qubit quantum register is in an arbitrary state $\mathbf{v}$. Show that the following three experiments will yield the same probability of output:

(a) Measure the register and output the result.

(b) First measure the first qubit and output it, then measure the second qubit and output it.

(c) First measure the second qubit and output it, then measure the first qubit and output it.

10.7 Suppose that f is computed in T time by a quantum algorithm that uses a partial measurements in the middle of the computation, and then proceeds differently according to the result of that measurement. Show that f is computable by $O(T)$ elementary operations.

10.8 Show that in a quantum computation that runs for T steps, we can replace each gate with any other gate (i.e., 8×8 matrix) which is the same in the $10 \log T$ most significant bits. Show that the amplitudes in the resulting final states are the same in the first T bits.

10.9 Prove that if for some $a \in \{0,1\}^n$, the strings $y_1, \ldots, y_{n-1}$ are chosen uniformly at random from $\{0,1\}^n$ subject to $y_i \odot a = 0$ for every $i \in [n-1]$, then with probability at least $1/10$, there exists no nonzero string $a' \neq a$ such that $y_i \odot a' = 0$ for every $i \in [n-1]$. (In other words, the vectors $y_1, \ldots, y_{n-1}$ are linearly independent.)

10.10 Prove that given $A, x \in \{0, \ldots, M-1\}$, we can compute (using a classical TM!) $A^x \pmod M$ in time polynomial in $\log M$. H454

10.11 Prove that for every $\alpha < 1$, there is at most a single rational number a/b such that $b < N$ and $|\alpha - a/b| < 1/(2N^2)$.

10.12 Prove that if A, B are numbers such that N and A are coprime but N divides AB, then N divides B. H454

10.13 Complete the proof of Lemma 10.21:

(a) Prove that for every prime P, the map $x \mapsto x^2 \pmod P$ is two-to-one on $\mathbb{Z}_P^*$.

(b) Prove that if X's order modulo P is of the form $2^j c$ for some $j \geq 1$ and odd c, then the order of X^2 is of the form $2^{j-1}c'$ for odd c'.

(c) Complete the proof of Lemma 10.21 for an arbitrary composite N that is not a prime power.

10.14 Prove Lemma 10.17.

10.15 Complete the proof of Lemma 10.19 for the case that r and M are not coprime. That is, prove that also in this case there exist at least $\Omega(r/\log r)$ values x's such that $0 \leq rx \pmod M \leq r/2$ and $\lceil M/x \rceil$ and r are coprime. H454

10.16 (Uses knowledge of continued fractions) Suppose $j, r \leq N$ are mutually coprime and unknown to us. Show that if we know the first $2 \log N$ bits of j/r then we can recover j, r in polynomial time.