Chapter 21

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

A quantum computer is a computational model that may be physically realizable and may have an exponential advantage over Turing machines in solving certain computational problems. In this chapter we survey the model, its relations to "classical" computational models such as probabilistic and deterministic Turing machines and the most important algorithms for quantum computers.

The strong Church-Turing thesis. As complexity theorists, the main reason to study quantum computers is that they pose a serious challenge to the strong Church-Turing thesis that stipulates that any physically reasonable computation device can be simulated by a Turing machine with

polynomial overhead. Quantum computers seem to violate no fundamental laws of physics and yet currently we do not know any such simulation. In fact, there's some evidence to the contrary: there's a polynomial-time algorithm for quantum computers to factor integers, where despite much effort no such algorithm is known for deterministic or probabilistic Turing machines. In fact, the conjectured hardness of this problem underlies several widely used encryption schemes such as the RSA cryptosystem. Thus the feasibility of quantum computers is very interesting for anyone interested in the security of these schemes. Physicists are also interested in quantum computers as studying them may shed light on quantum mechanics, a theory which, despite its great success in predicting experiments, is still not fully understood.

21.1 Quantum physics

Quantum phenomena are counterintuitive. To see this, consider the basic experiment of quantum mechanics that proves the wave nature of electrons: the 2-slit experiment. (See Figure 21.1.) A source fires electrons one by one at a wall. The wall contains two tiny slits. On the far side are small detectors that light up whenever an electron hits them. We measure the number of times each detector lights up during the hour. The results are as follows. When we cover one of the slits, we observe the strongest flux of electrons right behind the open slit, as one would expect. When both slits are open we would expect that the number of electrons hitting any particular position would be the sum of number hitting it when the first slit is open and the number hitting it when the second slit is open. Instead, what happens is that there's an "interference" phenomenon of electrons coming through two slits. In particular, at several detectors the total electron flux is *lower* when both slit are open as compared to when a single slit is open. This defies explanation if electrons behave as particles or "little balls".

Figure unavailable in pdf file.

Figure 21.1: 2-slit experiment

The only explanation physics has for this experiment is that an electron does not behave as a ball. It should be thought of as *simultaneously* going through both slits at once, kind of like a wave. Rather than thinking of the electron has having some non-negative probability of reaching a point x via

slit i, we think of it has having some *amplitude* $\alpha_{x,i}$, where this amplitude is a complex number (in particular, it can be a negative number). The probability of an electron hitting x when slit i is open is proportional to $|\alpha_{x,i}|^2$ and the probability of hitting x when both slits are open is proportional to $|\alpha_{x,1} + \alpha_{x,2}|^2$. In particular, if, say, $\alpha_{x,1}$ is positive and $\alpha_{x,2}$ is negative then the electron may hit x with smaller probability when both slits are open than when only one of them is open.

"Nonsense!" you might say. "I need proof that the electron actually went through both slits." So you propose the following modification to the experiment. Position two detectors at the slits; these light up whenever an electron passed through the slit. Now you can test the hypothesis that the electron went through both slits simultaneously. If you put such detectors at the slits, you will see that each electron indeed went through only one slit, but now you'll also see that the interference phenomenon disappears and the graph of electron hits on the wall becomes a simple sum! The explanation is roughly as follows: the quantum nature of particles "collapses" when they are "observed." More specifically, a quantum system evolves according to certain laws, but "observation" from nosy humans and their detectors "collapses" the quantum state and this is a nonreversible operation. (This may seem mysterious and it is; see Chapter notes.) 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 Chapter notes).

21.2 Quantum superpositions

Now we describe a *quantum register*, a basic component of the quantum computer. Recall the classical register, the building block of the memory in your desktop computer. An n-bit classical register with n bits consists of n particles. Each of them can be in 2 states: up and down, or 0 and 1. Thus there are 2^n possible configurations, and at any time the register is in one of these configurations.

The n-bit quantum register is similar, except at any time it can exist in a *superposition* of all 2^n configurations. (And the "bits" are called "qubits.") Each configuration $S \in \{0,1\}^n$ has an associated amplitude $\alpha_S \in \mathbb{C}$ where

$\mathbb{C}$ is the set of complex numbers.

$$\alpha_S = \text{amplitude of being in configuration } S$$

Physicists like to denote this system state succinctly as $\sum_S \alpha_S |S\rangle$. This is their notation for describing a general vector in the vector space $\mathbb{C}^{2^n}$, expressing the vector as a linear combination of basis vectors. The basis contains a vector $|S\rangle$ for each configuration S. The choice of the basis used to represent the configurations is immaterial so long as we fix a basis once and for all.

At every step, actions of the quantum computer —physically, this may involve shining light of the appropriate frequency on the quantum register, etc.— update α_S according to some physics laws. Each computation step is essentially a linear transformation of the system state. Let α denote the current configuration (i.e., the system is in state $\sum_S \alpha_S |S\rangle$) and U be the linear operator. Then the next system state is $\beta = U\alpha$. Physics laws require U to be unitary, which means $UU^\dagger = I$. (Here $U^\dagger$ is the matrix obtained by transposing U and taking the complex conjugate of each entry.) Note an interesting consequence of this fact: the effect of applying U can be reversed by applying the operator $U^\dagger$: thus quantum systems are *reversible*. This imposes strict conditions on which kinds of computations are permissible and which are not.

As already mentioned, during the computation steps, the quantum register is isolated from the outside world. Suppose we open the system at some time and observe the state of the register. If the register was in state $\sum_S \alpha_S |S\rangle$ at that moment, then

$$\mathbf{Pr}[\text{we see configuration S}] = |\alpha_S|^2 \tag{1}$$

In particular, we have $\sum_S |\alpha_S|^2 = 1$ at all times. Note that observation is an irreversible operator. We get to see one configuration according to the probability distribution described in (1) and and the rest of the configurations are lost forever.

What if we only observe a few bits of the register —a so-called *partial observation*? Then the remaining bits still stay in quantum superposition. We show this by an example.

Example 21.1
Suppose an n-bit quantum register is in the state

$$\sum_{s\in\{0,1\}^{n-1}} \alpha_s |0\rangle |s\rangle + \beta_s |1\rangle |s\rangle \tag{2}$$

(sometimes this is also represented as $\sum_{s\in\{0,1\}^{n-1}}(\alpha_s|0\rangle+\beta_s|1\rangle)|s\rangle$, and we will use both representations). Now suppose we observe just the first bit of the register and find it to be 0. Then the new state is

$$\left(\sum_{s\in\{0,1\}^{n-1}}\sqrt{|\alpha_s|^2}\right)^{-1}\sum_{s\in\{0,1\}^{n-1}}\alpha_s|0\rangle|s\rangle \tag{3}$$

where the first term is a rescaling term that ensures that probabilities in future observations sum to 1.

21.3 Classical computation using reversible gates

Motivated by the 2nd Law of Thermodynamics, researchers have tried to design computers that expend —at least in principle— zero energy. They have invented *reversible gates*, which can implement all classical computations in a reversible fashion. We will study reversible classical gates as a stepping stone to quantum gates; in fact, they are simple examples of quantum gates.

The *Fredkin gate* is a popular reversible gate. It has 3 Boolean inputs and on input (a,b,c) it outputs (a,b,c) if $a=1$ and (a,c,b) if $a=0$. It is *reversible*, in the sense that $F(F(a,b,c))=(a,b,c)$. Simple induction shows that if a circuit is made out of Fredkin gates alone and has m inputs then it must have m outputs as well. Furthermore, we can recover the inputs from the outputs by just applying the circuit in reverse. Hence a Fredkin gate circuit is *reversible*.

Figure unavailable in pdf file.

Figure 21.2: Fredkin gate and how to implement AND with it.

The Fredkin gate is *universal*, meaning that every circuit of size S that uses the familiar AND, OR, NOT gates (maximum fanin 2) has an equivalent Fredkin gate circuit of size $O(S)$. We prove this by showing that we can implement AND, OR, and NOT using a Fredkin gate some of whose inputs have been fixed 0 or 1 (these are "control inputs"); see Figure 21.2 for AND and Figure 21.3 for NOT; we leave OR as exercise. We also need to show how to copy a value with Fredkin gates, since in a normal circuit, gates can

have fanout more than 1. To implement a COPY gate using Fredkin gates is easy and is the same as for the the NOT gate (see Figure 21.3).

Thus to transform a normal circuit into a reversible circuit, we replace each gate with its Fredkin implementation, with some additional "control" inputs arriving at each gate to make it compute as AND/OR/NOT. These inputs have to be initialized appropriately.

The transformation appears in Figure 21.4, where we see that the output contains some junk bits. With a little more work (see Exercises) we can do the transformation in such a way that the output has no junk bits, just the original control bits. The reversible circuit starts with some input bits that are initialized to 0 and these are transformed into output bits.

Figure unavailable in pdf file.

Figure 21.3: Implementing NOT and COPY with Fredkin Gate

Figure unavailable in pdf file.

Figure 21.4: Converting a normal circuit C into an equivalent circuit C' of Fredkin gates. Note that we need additional control inputs

21.4 Quantum gates

A 1-input quantum gate is represented by a unitary 2×2 matrix $U = \left(\begin{smallmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{smallmatrix}\right)$. When its input bit is 0 the output is the superposition $U_{00}|0\rangle + U_{01}|1\rangle$ and when the input is 1 the output is the superposition $U_{10}|0\rangle + U_{11}|1\rangle$. When the input bit is in the superposition $\alpha_0|0\rangle + \beta_0|1\rangle$ the output bit is a superposition of the corresponding outputs

$$(\alpha_0 U_{00} + \beta_0 U_{10})|0\rangle + (\alpha_0 U_{01} + \beta_0 U_{11})|1\rangle. \tag{4}$$

More succinctly, if the input state vector is (α_0, β_0) then the output state vector is

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = U \begin{pmatrix} \alpha_0 \\ \beta_0 \end{pmatrix}$$

If $|\alpha|^2 + |\beta|^2 = 1$ then unitarity of U implies that $|\alpha'|^2 + |\beta'|^2 = 1$.

Similarly, a 2-input quantum gate is represented by a unitary 4×4 matrix R. When the input is the superposition $\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle +$

$\alpha_{11}|11\rangle$, the output is $\beta_{00}|00\rangle + \beta_{01}|01\rangle + \beta_{10}|10\rangle + \beta_{11}|11\rangle$ where

$$\begin{pmatrix}\beta_{00}\\ \beta_{01}\\ \beta_{10}\\ \beta_{11}\end{pmatrix} = R \begin{pmatrix}\alpha_{00}\\ \alpha_{01}\\ \alpha_{10}\\ \alpha_{11}\end{pmatrix}$$

In general, a quantum gate with k inputs is specified by a unitary $2^k \times 2^k$ matrix.

EXAMPLE 21.2
A Fredkin gate is also a valid 3-input quantum gate. We represent it by an 8×8 matrix that gives its output on all 2^3 possible inputs. This matrix is a permutation matrix (i.e., obtainable from the identity matrix by applying a permutation on all the rows) since the output $F(a, b, c)$ is just a permutation of the input (a, b, c). Exercise 3 asks you to verify that this permutation matrix is unitary.

A quantum circuit on n inputs consists of (a) an n-bit quantum register (b) a sequence of gates $(g_j)_{j=1,2,\ldots}$. If g_j is a k-input gate, then the circuit specification has to also give a sequence of bit positions $(j, 1), (j, 2), \ldots, (j, k) \in [1, n]$ in the quantum register to which this gate is applied. The circuit computes by applying these gate operations to the quantum register one by one in the specified order. The register holds the state of the computation, and only one gate is applied at any given time.

EXAMPLE 21.3
Suppose we have an n-bit quantum register in the state $\sum_{S \in 0,1^n} \alpha_S |S\rangle$. If we apply a 1-input quantum gate U to the first wire, the new system state is computed as follows. First "factor" the initial state by expressing each n-bit configuration as a concatenation of the first bit with the remaining $n-1$ bits:

$$\sum_{S' \in \{0,1\}^{n-1}} \alpha_{0,S'} |0S'\rangle + \alpha_{1,S'} |1S'\rangle. \quad (5)$$

(Formally we could express everything we are doing in terms of tensor product of vector spaces but we will not do that.)

To obtain the final state apply U on the first bit in each configuration as explained in equation (4). This yields

$$\sum_{S' \in \{0,1\}^{n-1}} (\alpha_{0,S'} U_{00} + \alpha_{1,S'} U_{10}) \left|0S'\right\rangle + (\alpha_{0,S'} U_{01} + \alpha_{1,S'} U_{11}) \left|1S'\right\rangle \quad (6)$$

We can similarly analyze the effect of applying a k-input quantum gate on any given set of k bits of the quantum register, by first "factoring" the state vector as above.

21.4.1 Universal quantum gates

You may now be a little troubled by the fact that the set of possible 1-input quantum gates is the set of all unitary 2×2 matrices, an uncountable set. That seems like bad news for the Radio Shacks of the future, who may feel obliged to keep all possible quantum gates in their inventory, to allow their customers to build all possible quantum circuits.

Luckily, Radio Shack need not fear. Researchers have shown the existence of a small set of "universal" 2-input quantum gates such that every circuit composed of S arbitrary k-input quantum gates can be simulated using a circuit of size $2^{\mathrm{poly}(k)} \cdot S\mathrm{poly}(\log S)$ composed only of our universal gates. The simulation is not exact and the distribution on outputs is only approximately the same as the one of the original circuit. (All of this assumes the absence of any outside noise; simulation in presence of noise is a topic of research and currently seems possible under some noise models.)

In any case, we will not need any fancy quantum gates below; just the Fredkin gate and the following 1-input gate called the Hadamard gate:

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

21.5 BQP

DEFINITION 21.4
A language $L \subseteq \{0,1\}^*$, is in **BQP** iff there is a family of quantum circuits (C_n) of size n^c s.t. $\forall x \in \{0,1\}^*$:

$$x \in L \Rightarrow \mathbf{Pr}[C(x)_1 = 1] \geq \frac{2}{3}$$
$$x \notin L \Rightarrow \mathbf{Pr}[C(x)_1 = 1] \leq \frac{1}{3}$$

Here $C(x)_1$ is the first output bit of circuit C, and the probability refers to the probability of observing that this bit is 1 when we "observe" the outputs at the end of the computation.

The circuit has to be uniform, that is, a deterministic polynomial time (classical) Turing machine must be able to write down its description.

At first the uniformity condition seems problematic because a classical Turing machine cannot write complex numbers needed to describe quantum gates. However, the machine can just express the circuit approximately using universal quantum gates, which comprise a finite family. The approximate circuit computes the same language because of the gap between the probabilities 2/3 and 1/3 used in the above definition.

THEOREM 21.5
P $\subseteq$ **BQP**

PROOF: Every language in **P** has a uniform circuit family of polynomial size. We transform these circuits into reversible circuits using Fredkin gates, thus obtaining a quantum circuit family for the language. ■

THEOREM 21.6
BPP $\subseteq$ **BQP**

PROOF: Every language in **BPP** has a uniform circuit family (C_n) of polynomial size, where circuit C_n has n normal input bits and an additional $m = \text{poly}(n)$ input wires that have to be initialized with random bits.

We transform the circuit into a reversible circuit C'_n using Fredkin gates. To produce "random" bits, we feed m zeros into an array of Hadamard gates and plug their outputs into C'_n; see Figure 21.5.

A simple induction shows that for any N if we start with an N-bit quantum register in the state $|\mathbf{0}\rangle$ and apply the Hadamard gate one by one

Figure unavailable in pdf file.

Figure 21.5: Turning a BPP circuit into an equivalent quantum circuit. An array of Hadamard gates turns the all-0 string into a uniform superposition of all m-bit strings.

on all the bits, then we obtain the superposition

$$\sum_{S\in\{0,1\}^N} \frac{1}{2^{N/2}} |S\rangle \tag{7}$$

(Note: the case $N = 1$ follows from the definition of the Hadamard gate.)

The correctness of the above transformation is now clear since the "random" inputs of circuit C'_n receive the output from the array of Hadamard gates, in other words, a uniform quantum superposition of all possible m-bit strings. Thus the output bit of C'_n is a uniform superposition of the result on each bit string. If we perform an observation on this output bit at the end, then the the probability that it is observed to be 1 is exactly the probability that C_n accepts when the random gates are fed a truly random string. ■

21.6 Factoring integers using a quantum computer

This section proves the following famous result.

THEOREM 21.7 (SHOR [**?**])
There is a polynomial size quantum circuit that factors integers.

We describe Kitaev's proof of this result since it uses eigenvalues, a more familiar and intuitive concept than the Fourier transforms used in Shor's algorithm.

DEFINITION 21.8 (EIGENVALUE)
λ is an *eigenvalue* of matrix M if there is a vector e (called the *eigenvector*) , s.t.:

$$M \cdot e = \lambda e$$

Fact: If M is unitary, then $|M \cdot x|_2 = 1$ for each unit vector, so $|\lambda| = 1$. In other words there is a $\theta \in [0, 1)$ such that

$$\lambda = e^{2\pi\iota\theta} = \cos(2\pi\theta) + \iota \sin(2\pi\theta).$$

Here $\iota = \sqrt{-1}$.

Fact: If $M \cdot e = \lambda e$ then $M^k \cdot e = \lambda^k e$. Hence e is still an eigenvector of M^k and λ^k is the corresponding eigenvalue.

Now we are ready to describe Kitaev's algorithm and prove Theorem 21.7.
PROOF: Let N be the number to be factored. As usual, Z_N^* is the set of numbers mod N that are co-prime to N. Simple number theory shows that for every $a \in Z_N^*$ there is a smallest integer r such that $a^r \equiv 1 \pmod N$; this r is called the *order* of a. The algorithm will try to find the order of a random element of Z_N^*. It is well-known if we can do this then we can factor N with high probability; here's a sketch. First, note that with probability at least $1/2$, the element a has even order. Now if $(a^r - 1) \equiv 0 \pmod N$, then $(a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) \equiv 0 \pmod N$. If a is random, with probability $\geq \frac{1}{2}$, $a^{\frac{r}{2}} \neq 1 \pmod N, a^{\frac{r}{2}} \neq -1 \pmod N$ (this is a simple exercise using the Chinese remainder theorem). Thus hence $gcd(N, a^{\frac{r}{2}} - 1) \neq N, 1$. Thus, knowing r we can compute $a^{r/2}$ and compute $gcd(N, a^{\frac{r}{2}} - 1)$. Thus with probability at least $1/4$ (over the choice of a) we obtain a factor of N.

The factoring algorithm is a mixture of a classical and a quantum algorithm. Using classical random bits it generates a random $a \in Z_N^*$ and then constructs a quantum circuit. Observing the output of this quantum circuit a few times followed by some more classical computation allows it to obtain r, the order of a, with reasonable probability. (Of course, we could in principle describe the entire algorithm as a quantum algorithm instead of as a mixture of a classical and a quantum algorithm, but our description isolates exactly where quantum mechanics is crucial.)

Consider a classical reversible circuit of size $\text{poly}(\log N)$ that acts on numbers in Z_N^*, and computes $U(x) = ax \pmod N$. Then we can view this circuit as a quantum circuit operating on a quantum register. If the quantum register is in the superposition[1]

$$\sum_{x \in Z_N^*} \alpha_x |x\rangle,$$

then applying U gives the superposition

$$\sum_{x \in Z_N^*} \alpha_x |\, ax \pmod N \rangle.$$

[1] Aside: This isn't quite correct; the cardinality of Z_N^* is not a power of 2 so it is not immediately obvious how to construct a quantum register whose only possible configurations correspond to elements of Z_N^*. However, if we use the nearest power of 2, everything we are about to do will still be approximately correct. There are also other fixes to this problem.

Figure unavailable in pdf file.

Figure 21.6: Conditional-U circuit

Interpret this quantum circuit as an $N \times N$ matrix —also denoted U— and consider its eigenvalues. Since $U^r = I$, we can easily check that its eigenvalues are $e^{2\pi\iota\theta}$ where $\theta = \frac{j}{r}$ for some $j = 0, 1, 2, \ldots, r-1$. The algorithm will try to obtain a random eigenvalue. It thus obtains —in binary expansion— a number of form $\frac{j}{r}$ where j is random. Chances are good that this j is coprime to r, which means that $\frac{j}{r}$ is an irreducible fraction. Even knowing only the first $2 \log N$ bits in the binary expansion of $\frac{j}{r}$, the algorithm can round off to the nearest fraction whose denominator is at most N (this can be done; see Section **??**) and then it reads off r from the denominator.

Next, we describe how to do this estimation of the eigenvalue. The discussion assumes we know how to produce a quantum register whose state corresponds to $\mathbf{e}$, a random quantum vector of U; see Section 21.6.3 for how to do this (more precisely, something that is "just as good.")

21.6.1 Phase estimation: the first few bits

Now we describe the basic primitive promised above. We have a reversible classical circuit U with n inputs/outputs and poly(n) size (U is also thought of as a quantum circuit). We have an n-bit quantum register whose (quantum) state is given by $\mathbf{e}$, which happens to be an eigenvector of U corresponding to an eigenvalue $\lambda = e^{2\pi\iota\theta}$. We wish to apply some quantum circuit of size *poly*(n)—which as it turns out will incorporate many copies of U—on this register, such that we can compute the first $2n$ bits of of θ. In this section we only manage to compute the first $O(\log n)$ bits; the next section shows how to finish the job.

First, we notice that applying U on the state $\mathbf{e}$ puts it in the state $\lambda\mathbf{e}$. Thus the register's state has undergone a *phase shift* —i.e., multiplication by a scalar. The algorithm we describe measures this λ, and is therefore called *phase estimation*.

Now define a conditional-U circuit (Figure 21.6), whose input is (b, x) where b is a bit, and cond-$U(b, x) = (b, x)$ if $b = 0$ and (b, Ux) if $b = 1$. We leave it as an exercise how to implement this using U and some Fredkin gates.

Using a cond-U circuit and two Hadamard gates, we can build a quantum circuit —the "basic block"— shown in Figure 21.7. When this is applied

Figure unavailable in pdf file.

Figure 21.7: Basic building block consists of a Conditional-U and two Hadamard gates.

to a quantum register whose first bit is 0 and the remaining bits are in the state $\mathbf{e}$, then we can measure the corresponding eigenvalue λ by repeated measurement of the first output bit.

$$\begin{aligned} |0\rangle|e\rangle &\xrightarrow{H_1} \frac{1}{\sqrt{2}}|0\rangle|e\rangle + \frac{1}{\sqrt{2}}|1\rangle|e\rangle \\ &\xrightarrow{\text{cond-U}} \frac{1}{\sqrt{2}}|0\rangle|e\rangle + \frac{\lambda}{\sqrt{2}}|1\rangle|e\rangle \\ &\xrightarrow{H_2} \frac{1}{2}((1+\lambda)|0\rangle|e\rangle + (1-\lambda)|1\rangle|e\rangle) \end{aligned} \tag{8}$$

Let $p(0)$ denote the probability of observing a 0 in the first bit. Then

$$\begin{aligned} p(0) &= \left|\frac{1+e^{2\pi\iota\theta}}{2}\right|^2 = \frac{|1+\cos(2\pi\theta)+\iota\sin(2\pi\theta)|^2}{2} \\ &= \frac{\left|1+2\cos(2\pi\theta)+\cos^2(2\pi\theta)+\sin^2(2\pi\theta)\right|}{4} \\ &= \frac{1+\cos(2\pi\theta)}{2}. \end{aligned} \tag{9}$$

(The second line uses the fact that $|a+\iota b|^2 = a^2+b^2$).

We will refer to this bit as the *phase bit*, since repeatedly measuring it allows us to compute better and better estimates to $p(0)$ and hence θ and hence λ. Actually, instead of measuring repeatedly we can just design a quantum circuit to do the repetitions by noticing that the output is just a scalar multiple of $\mathbf{e}$, namely, $\lambda\mathbf{e}$. If we were to repeat the above calculation with $\lambda\mathbf{e}$ instead of $\mathbf{e}$, we find that the probability of measuring 0 in the first bit is again given by (9). So we can just feed the new state $\lambda\mathbf{e}$ into another basic block with a fresh phase bit initialized to 0, and so on (see Figure 21.8). We measure phase bits for all the blocks all at once at the end. Observing the number of phase bits that are 0 gives us an estimate for λ.

Are we done? Unfortunately, no. Obtaining an estimate to the first m bits of λ means approximating it within an additive error 2^{-m}, which involves about 2^{2m} trials (in general estimating the bias of an unknown coin up to error ϵ with probability $1-\delta$ requires tossing it $O(\log(1/\delta)/\epsilon^2)$ times).

Figure unavailable in pdf file.

Figure 21.8: Repeating the basic experiment to get better estimate of λ.

Thus the iterated construction needs 2^{2m} copies of the basic block and the circuit size is poly(n) only when $m = O(\log n)$. Thus simple repetition is a very inefficient way to obtain accurate information about λ.

21.6.2 Better phase estimation using structure of U

This section gives a more efficient phase estimation technique that works only for U that have the following property: if U has size poly(n) then for every $k \geq 1$, computing U^{2^k} only requires a circuit of size poly($n + k$). Luckily, the U we are interested in does have this property: $U^{2^k}(x) = a^{2^k} x \pmod N$, and a^{2^k} is computable by circuits of size poly($\log N + \log k$) using fast exponentiation.

Using this property of U, we can implement a conditional-U^{2^k} circuit using a quantum circuit of size poly($\log N + k$). The eigenvalues of U^{2^k} are λ^{2^k}. If $\lambda = e^{2\pi\iota\theta}$ where $\theta \in [0, 1)$ (see Figure 21.9) then $\lambda^{2^k} = e^{2\pi\iota\theta 2^k}$. Since $e^{2\pi\iota\theta 2^k}$ is the same complex number as $e^{2\pi\iota\alpha}$ where $\alpha = 2^k\theta \pmod 1$, measuring λ^{2^k} actually gives us $2^k\theta \pmod 1$. The most significant bit of $2^k\theta \pmod 1$ is nothing but the kth bit of θ. Using $k = 0, 1, 2, \ldots 2 \log N$ we can obtain[2] the first $2 \log N$ bits of θ.

As in Figure 21.8, we can bundle these steps into a single cascading circuit where the output of the conditional-$U^{2^{k-1}}$ circuit feeds into the conditional-U^{2^k} circuit. Each circuit has its own set of $O(\log N)$ phase bits; measuring the phase bits of the kth circuit gives an estimate of the kth bit of θ that is correct with probability at least $1 - 1/N$. All phase bits are measured in a single stroke at the end. The union bound implies that the probability that all phase bits are correct is at least $1 - 2 \log N/N$.

[2] There is a slight inaccuracy here, since we are blurring the distinction between measuring λ and θ. For example, when θ is very close to 1/2, then $e^{2\pi\iota\theta} \approx -1$. So the complex number $1 + \lambda$ is close 0 and the phase bit almost always comes up 1. Now we cannot tell whether the binary expansion of θ is more like $0.1000x$ or $0.0111x$, both of which are close to 1/2. But then the estimation of the 1st and 2nd bits allows us to infer which of these two cases occurs, and hence the value of the first bit. Thus the correct estimation procedure involves looking at estimated values of each pair of successive bits.

Figure unavailable in pdf file.

Figure 21.9: Eigenvalue $\lambda = e^{2\pi\iota\theta}$ in the complex plane.

21.6.3 Uniform superpositions of eigenvectors of U

To finish, we need to show how to put a quantum register into a state corresponding to a random eigenvector. Actually, we only show how to put the quantum register into a uniform superposition of eigenvectors of U. This suffices for our cascading circuit, as we will argue shortly.

First we need to understand what the eigenvectors look like. Recall that $\left\{1, a, a^2, \ldots, a^{r-1}\right\}$ is a subgroup of Z_N^*. Let B be a set of representatives of all cosets of this subgroup. In other words, for each $x \in Z_N^*$ there is a unique $b \in B$ and $l \in \{0, 1, \ldots, r-1\}$ such that $x = ba^l \pmod N$. It is easily checked that the following is the complete set of eigenvectors, where $\omega = e^{\frac{2\pi\iota}{r}}$:

$$\forall j \in \{0, 1, \ldots, r-1\}, \forall b \in B \qquad \mathbf{e}_{j,b} = \sum_{l=0}^{r-1} \omega^{jl} \left| \, ba^l \pmod N \right\rangle \tag{10}$$

The eigenvalue associated with this eigenvector is $\omega^{-j} = e^{-\frac{2\pi\iota j}{r}}$.

Fix b and consider the uniform superposition:

$$\frac{1}{r}\sum_{j=0}^{r-1} \mathbf{e}_{j,b} = \frac{1}{r}\sum_{j=0}^{r-1}\sum_{l=0}^{r-1} \omega^{jl} \left| \, ba^l \pmod N \right\rangle \tag{11}$$

$$= \frac{1}{r}\sum_{l=0}^{r-1}\sum_{j=0}^{r-1} \omega^{jl} \left| \, ba^l \pmod N \right\rangle. \tag{12}$$

Separating out the terms for $l = 0$ and using the formula for sums of geometric series:

$$= \frac{1}{r}\Big(\sum_{j=0}^{r-1} |b\rangle + \sum_{l=1}^{r-1} \frac{(\omega^l)^r - 1}{\omega^l} \left| \, ba^l \pmod N \right\rangle\Big) \tag{13}$$

since $\omega^r = 1$ we obtain

$$= |b\rangle \tag{14}$$

■

Thus if we pick an arbitrary b and feed the state $|b\rangle$ into the quantum register, then that can also be viewed as a uniform superposition $\frac{1}{r}\sum_j \mathbf{e}_{j,b}$.

21.6.4 Uniform superposition suffices

Now we argue that in the above phase estimation based algorithm, a uniform superposition of eigenvectors is just as good as a single eigenvector.

Fixing b, the initial state of the quantum register is

$$\frac{1}{r}\sum_j |\mathbf{0}\rangle\,|\mathbf{e}_{j,b}\rangle,$$

where $\mathbf{0}$ denotes the vector of phase bits that is initialized to 0. After applying the quantum circuit, the final state is

$$\frac{1}{r}\sum_j |c_j\rangle\,|\mathbf{e}_{j,b}\rangle,$$

where $|c_j\rangle$ is a state vector for the phase bits that, when observed, gives the first $2\log N$ bits of j/r with probability at least $1-1/N$. Thus observing the phase bits gives us whp a random eigenvalue.

21.7 Quantum Computing: a Tour de horizon

Will give 1-para intros to quantum information, quantum crypto, the fact that there is an oracle for which **NP** is not in **BQP**, grover's algorithm, models of quantum computation, decoherence and error correction,quantum interactive proofs.

Exercises

§1 Implement an OR gate using the Fredkin gate.

§2 Verify that the Fredkin gate is a valid quantum gate.

§3 Given any classical circuit computing a function f from n bits to n bits, describe how to compute the same function with a reversible circuit that is a constant factor bigger and has no "junk" output bits.

Hint: Use our transformation, then copy the output to a safe place, and then run the circuit in reverse to erase the junk outputs.

§4 Suppose the description of a quantum circuit U is given to you. Describe an implementation of the Conditional-U circuit.

§5 Let N be composite. For $a \in \mathbb{Z}_N^*$ let $r = \texttt{ord}(a)$ be the order of a, i.e. r is the minimal number such that $a^r = 1 \pmod N$. Show that if $a \in Z_N^*$ is randomly chosen then with probability at least $1/10$ **(1)** $\texttt{ord}(a)$ is even and **(2)** $a^{\texttt{ord}(a)/2} \not\equiv \pm 1 \pmod N$.

Hint: use the Chinese remainder theorem and the fact that for a prime p the group $\mathbb{Z}_{p^*}$ is cyclic.

§6 Prove Lemma 21.9.

§7 Complete the proof of Lemma 21.10 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.

Hint: let $d = gcd(r, M)$, $r' = r/d$ and $M' = M/d$. Now use the same argument as in the case that M and r are coprime to argue that there exist $\Omega(\frac{r}{d \log r})$ values $x \in \mathbb{Z}_{M'}$ satisfying this condition, and that if x satisfies it then so does $x + cM$ for every c.

§8 (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.

Chapter notes and history

The "meaning" of quantum theory has eluded (or certainly puzzled) most scientists. (The physicist Penrose [**?**] has even sought to link human consciousness to the collapse of the quantum wave function, though this is controversial.) No one doubts that quantum effects exist at microscopic scales. The problem lies in explaining why they do not manifest themselves at the macrosopic level (or at least not to human consciousness). A *Scientific American* article by Yam [**?**] describes various explanations that have been advanced over the years. The leading theory is *decoherence*, which tries to use quantum theory to explain the absence of macroscopic quantum effects. Researchers are not completely comfortable with this explanation.

The issue is undoubtedly important to quantum computing, which 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. One theoretical idea is to treat decoherence as a form of noise, and to build noise-tolerance into the computation —a nontrivial process. For details of this and many other topics, see the books by Kitaev, Shen, and Vyalyi [**?**].

Feynman [**?**] was the first to suggest the possibility that quantum mechanics might allow Turing Machines more computational power than classical TMs. In 1985 Deutsch [**?**] defined a quantum turing machine, though in retrospect his definition is unsatisfactory. Better definitions then appeared in Deutsch-Josza [**?**], Bernstein-Vazirani [**?**] and Yao [**?**], at which point quantum computation was firmly established as a field.

The book by Nielsen and Chuang [**?**] gives a comprehensive treatment of quantum computation.

Alternative presentation of Shor's algorithm.

21.8 Quantum Notations

21.9 Simon's Algorithm

21.10 Integer factorization using quantum computers.

21.10.1 Reduction to Order Finding

21.10.2 Quantum Fourier Transform over $\mathbb{Z}_M$.

The *Fourier transform* over an Abelian group G is a linear operation that transforms a $|G|$-dimensional vector from representation in the standard basis to a representation in the *Fourier basis*, which is a different orthonormal basis. For $M = 2^m$ the Fourier over the group $\mathbb{Z}_M$ (numbers $\{0, \ldots, M-1\}$ with addition modulo M) can be represented as a $2^m \times 2^m$ unitary matrix. We'll now show this matrix can be implemented using $O(m^2)$ basic matrices. This means that we can transform a quantum system whose register is in state f to a system whose register is in the state corresponding to the Fourier transform $\hat{f}$ of f. This does not mean that we can compute in $O(m^2)$ the Fourier transform over $\mathbb{Z}_M$ - indeed this is not sufficient time to even write the output! Nonetheless, this transformation still turns out to be very useful, and is crucial to Shor's factoring algorithm in a way analogous to the

use of the Hadamard transformation (which is a Fourier transform over the group $\{0,1\}^n$ with the operation $\oplus$) was crucial to Simon's algorithm.

Fourier transform over $\mathbb{Z}_M$. For $M = 2^m$, let ω be a primitive M^{th} root of unity (e.g., $\omega = e^{2\pi i/M}$). A function $\chi : \mathbb{Z}_M \to \mathbb{C}$ is called a *character* of $\mathbb{Z}_M$ if $\chi(y+z) = \chi(y)\chi(z)$ for every $y, z \in \mathbb{Z}_M$. $\mathbb{Z}_M$ has M characters $\{\chi_x\}_{x\in\mathbb{Z}_M}$ where $\chi_x(y) = \frac{1}{\sqrt{M}}\omega^{xy}$ (the $\frac{1}{\sqrt{M}}$ factor is for normalization). These characters define an orthonormal basis since (denoting by $\overline{z}$ the complex conjugate of z)

$$\langle \chi_x, \chi_y \rangle = \frac{1}{M}\sum_{z=0}^{M-1} \omega^{xz}\overline{\omega^{yz}} = \frac{1}{M}\sum_{z=0}^{M-1} \omega^{(x-y)z}$$

which is equal to 1 if $x = y$ and to $\frac{1}{M}\frac{1-\omega^{(x-y)M}}{1-\omega^{x-y}} = 0$ if $x \neq y$ (the latter equality follows by the formula for the sum of a geometric series and the fact that $\omega^{\ell M} = 1$ for every ℓ). The *Fourier transform* of $f : \mathbb{Z}_M \to \mathbb{C}$ is the representation of f in this basis. For convenience we'll let $\hat{f}(x)$ denote the coefficient of χ_{-x} in this representation. Thus $f = \sum_{x=0}^{M-1} \hat{f}(x)\chi_{-x}$ and so $\hat{f}(x) = \langle f, \chi_{-x}\rangle = \frac{1}{\sqrt{M}}\sum_{y=0}^{M-1}\omega^{xy}f(x)$. We let $FT_M(f)$ denote the vector $(\hat{f}(0), \ldots, \hat{f}(M-1))$.

Note that

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

Since ω^2 is an $M/2$th root of unity and $\omega^{M/2} = -1$ we get that if W is the $M/2$ diagonal matrix with diagonal $\omega^0, \ldots, \omega^{M/2-1}$ then

$$FT_M(f)_{low} = FT_{M/2}(f_{even}) + WFT_{M/2}(f_{odd}) \quad (15)$$

$$FT_M(f)_{high} = FT_{M/2}(f_{even}) - WFT_{M/2}(f_{odd}) \quad (16)$$

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 (15) and (16) are the crux of the well known *Fast Fourier Transform* (FFT) algorithm that computes the Fourier transform in $O(M \log M)$ (as opposed to the Naive $O(M^2)$) time. We'll use them for the *quantum* Fourier transform algorithm, obtaining the following lemma:

LEMMA 21.9
There's an $O(m^2)$-step quantum algorithm that transforms a 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: We'll use the following algorithm:
Quantum Fourier Transform FT_M
Initial state: $f = \sum_{x\in\mathbb{Z}_M} f(x)|x\rangle$
Final state: $\hat{f} = \sum_{x\in\mathbb{Z}_M} \hat{f}(x)|x\rangle$.

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

The transformation W on $m-1$ bits which is $|x\rangle \mapsto \omega^x = \omega^{\sum_{i=0}^{m-2} 2^i x_i}$ (where x_i is the i^{th} bit of x) is the composition of $W_0, \ldots, W_{m-2}$ where W_i is a one qubit gate mapping $|0\rangle$ to $|0\rangle$ and $|1\rangle$ to $\omega^{2^i}|1\rangle$.

The final state is equal to $\hat{f}$ by (15) and (16). (We leave verifying this and the running time to Exercise 6.)

■

21.10.3 The Order-Finding Algorithm.

We'll now present a quantum algorithm that on input a number $A < N$, finds the order of A modulo N. That is, we'll find the smallest r such that $A^r = 1 \pmod N$. Note that using the repeated squaring algorithm, we

can compute the map $|x\rangle|0^n\rangle \mapsto |A^x \pmod N\rangle|0^n\rangle$ for some n polynomial in $\log N$. We choose $M = 2^m$ such that m is polynomial in $\log N$ and $M > 100N^2$.

Order Finding Algorithm:

Registers x is m bit register (which we think as a number in $\mathbb{Z}_M$) and y is an n. Initial state of the registers is $|0^m\rangle|0^n\rangle$.

Step 1 Apply the Fourier transform to the first register to obtain the state (ignoring normalization factors) $\sum_{x\in\mathbb{Z}_M}|x\rangle)|0^n\rangle$. That is, we obtain the uniform state in the first register. (Note that we can get essentially the same result by applying the Hadamard gate to each bit.)

Step 2 Compute the transformation $|x\rangle|y\rangle \mapsto |x\rangle|y \oplus (A^x \pmod N)\rangle$. The state will now be $\sum_{x\in\mathbb{Z}_M}|x\rangle|A^x \pmod N\rangle$.

Step 3 Measure the second register to get a value y_0. Let x_0 be the smallest number such that $A^{x_0} = y_0 \pmod N$. Then the state will now be $\sum_{\ell=0}^{\lceil M/r\rceil-1}|x_0+\ell r\rangle|y_0\rangle$, where r denotes the order of A.

Step 4 Apply the Fourier transform to the first register. The state will be

$$\left(\sum_{x\in\mathbb{Z}_n}\sum_{\ell=0}^{\lceil M/r\rceil-1}\omega^{(x_0+\ell r)x}|x\rangle\right)|y_0\rangle$$

Step 5 Measure the first register to obtain a number $x \in \mathbb{Z}_M$.

Step 6 Find the best rational approximation a/b with a, b coprime to the fraction $\frac{x}{M}$ such that $b < 2M$. Check that $A^b = A$ - if so then output b. (We sketch the classical algorithm to find such approximation below.)

Analysis: the case that $r|M$

We start by analyzing this algorithm in the (rather unrealistic) case that $M = rc$ for some integer c. In this case we claim that the value x obtained in Step 5 will be equal to $c'c$ for random $c' \in 0,\ldots,r$. In particular it will satisfy $xr = 0 \pmod M$. Since with non-negligible probability c' and r will be coprime and hence the reduced version of x/M would be c'/r. Indeed, if $x = c'c$ then (up to some normalization factor) the absolute value for $|x\rangle$'s coefficient after Step 4 is

$$\left|\sum_{\ell=0}^{c-1}\omega^{(x_0+\ell r)x}\right| = \left|\omega^{x_0c'c}\right|\left|\sum_{\ell=0}^{c}\omega^{(rc)c'\ell}\right| = 1\ell \tag{17}$$

since $\omega^{rc} = \omega^M = 1$. However, if $x \neq 0 \pmod{c}$ then since ω^r is a c^{th} root of unity we get that $\sum_{\ell=0}^{c}]w^{r\ell x} = 0$ by the formula for sums of geometric progressions. Thus, such x would be obtained in step 5 with zero probability.

The case that $r \not| M$

We'll now not be able to show that the value x obtained in Step 5 satisfies $xr = 0 \pmod{M}$. Indeed, there might not exist such a number. However, we will show that with $\Omega(1/\log r)$ probability, this value will satisfy **(1)** $0 \leq xr \pmod{M} < r/10$ and **(2)** $\lceil xr/M \rceil$ is coprime to r. Thus, we'll have that for some c $|xr - cM| < r/10$, which dividing by rM gives

$$\left|\frac{x}{M} - \frac{c}{r}\right| < \frac{1}{10M}$$

. Since it's easy to see that for every $0 < \alpha < 1$ there's only one fraction a/b with a, b coprime and $b < N$ such that $|\alpha - a/b| < \frac{1}{N^2}$ we'll get in Step 6 the fraction c/r. We do this by proving the following lemma:

LEMMA 21.10
There exist $\Omega(r/\log r)$ values x such that:

1. $0 \leq xr \pmod{M} < r/10$
2. $\lceil xr/M \rceil$ *and* r *are coprime*
3. *The coefficient of $|x\rangle$ in the state of Step 4 is at least $\Omega(\frac{1}{\sqrt{r}})$.*

PROOF: The intuition behind the proof is that we expect $xr \pmod{M}$ to be distributed roughly uniformly, and that if $xr \pmod{M}$ is very small, then as in (17) the coefficient of $|x\rangle$ will be relatively large.

For starters, assume that r is coprime to M. Then, the map $x \mapsto rx \pmod{M}$ is a permutation of $\mathbb{Z}_M^*$ and we have a set of at least $r/(20\log r)$ x's such that $xr \pmod{M}$ is a number p between 0 and $r/10$ that is coprime to r. For every such x we have that $xr + \lceil r/M \rceil M = p$ which means that $\lceil r/M \rceil$ can't have a nontrivial shared factor with r, as otherwise this factor would be shared with p as well. The case that $gcd(r, M) = d$ for $d > 1$ is handled similarly (Exercise 7).

We now prove that if $0 \leq xr \pmod{M} < r/10$ then the coefficient of $|x\rangle$ in the state of Step 4 is at least $\Omega(\frac{1}{\sqrt{r}})$. For any such x, the absolute value of $|x\rangle$'s coefficient is up to the normalization equal to:

$$\left|\sum_{\ell=0}^{\lceil M/r \rceil - 1} \omega^{\ell r x}\right|$$

Let $\beta = \omega^{rx}$. If $\beta = 1$ then this sum is equal to $\lceil M/r \rceil$ and otherwise it's equal to $\left|\frac{1-\beta^{\lceil M/r \rceil}}{1-\beta}\right|$. This is equal to $\frac{\sin(\lceil M/r \rceil \theta)}{\sin \theta}$, where $\theta = \frac{rx \pmod M}{M}$ is the angle such that $\beta = e^{i\theta}$ (see Figure ??). Since $0 < \theta < \frac{r}{10M}$, we get that $\lceil M/r \rceil \theta \leq 1/9$ and in this range up to a constant we have that this proportion is again equal to $\lceil M/r \rceil$. Since the measurement of Step 3 restricted the space an almost uniform distribution over about M/x of the x's and the Fourier transform adds another $\frac{1}{\sqrt{M}}$ normalization factor, it can be shown that the normalization coefficient at this step is at least $\frac{1}{2}\sqrt{\frac{r}{M}}\frac{1}{\sqrt{M}} = \sqrt{r}M$ and so we're done. ■