Learning Objectives:

- See main aspects in which quantum mechanics differs from local deterministic theories.
- Model of quantum circuits, or equivalently QNAND programs
- The complexity class **BQP** and what we know about its relation to other classes
- Ideas behind Shor's Algorithm and the Quantum Fourier Transform

22
Quantum computing

> *"We always have had (secret, secret, close the doors!) … 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. … Can I learn anything from asking this question about computers–about this may or may not be mystery as to what the world view of quantum mechanics is?"* , Richard Feynman, 1981

> *"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, 1981

There were two schools of natural philosophy in ancient Greece. *Aristotle* believed that objects have an *essence* that explains their behavior, and a theory of the natural world has to refer to the *reasons* (or "final cause" to use Aristotle's language) as to why they exhibit certain phenomena. *Democritus* believed in a purely mechanistic explanation of the world. In his view, the universe was ultimately composed of elementary particles (or *Atoms*) and our observed phenomena arise from the interactions between these particles according to some local rules. Modern science (arguably starting with Newton) has embraced Democritus' point of view, of a mechanistic or "clockwork" universe of particles and forces acting upon them.

While the classification of particles and forces evolved with time, to a large extent the "big picture" has not changed from Newton till Einstein. In particular it was held as an axiom that if we knew fully the current *state* of the universe (i.e., the particles and their properties such as location and velocity) then we could predict its future state at any point in time. In computational language, in all these theories the

state of a system with n particles could be stored in an array of $O(n)$ numbers, and predicting the evolution of the system can be done by running some efficient (e.g., $poly(n)$ time) computation on this array.

22.1 THE DOUBLE SLIT EXPERIMENT

Alas, in the beginning of the 20th century, several experimental results were calling into question the "billiard ball" theory of the world. One such experiment is the famous double slit experiment. One way to describe it is as following. Suppose that we buy one of those baseball pitching machines, and aim it at a soft plastic wall, but put a metal barrier between the machine and the wall that has a single slit. If we shoot baseballs at the wall, then some of the baseballs would bounce off the metal barrier, while some would make it through the slit and dent the plastic wall. If we now carve out an additional slit in the metal barrier then more balls would get through, and so the plastic wall would be *even more dented*. (See Fig. 22.1)

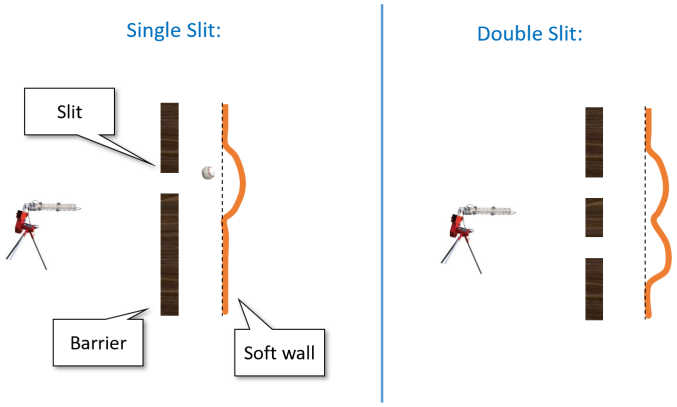

Figure 22.1: In the "double baseball experiment" we shoot baseballs from a gun at a soft wall through a hard barrier that has one or two slits open in it. There is only "constructive interference" in the sense that the dent in each position in the wall when both slits are open is the sum of the dents when each slit is open on its own.

So far this is pure common sense, and it is indeed (to my knowledge) an accurate description of what happens when we shoot baseballs at a wall. However, this is not the same when we shoot *photons*. Amazingly, if we shoot with a "photon gun" (i.e., lasers) at a wall equipped with photon detectors through some barrier, then (as shown Fig. 22.2) we sometimes see *fewer* hits when the two slits are open than one only ones of them is!.[1] In particular there are positions in the wall that are hit when the first slit is open, hit when the second gun is open, but are *not hit at all when both slits are open!*.

It seems as if each photon coming out of the gun is aware of the global setup of the experiment, and behaves differently if two slits are open than if only one is. If we try to "catch the photon in the act" and

[1] A nice illustrated description of the double slit experiment appears in this video.

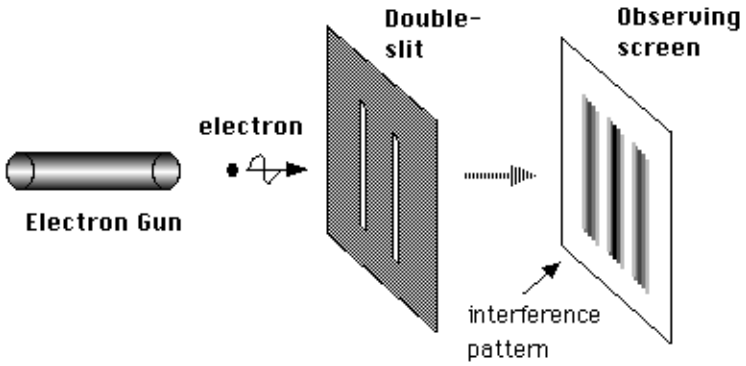

Figure 22.2: The setup of the double slit experiment in the case of photon or electron guns. We see also *destructive* interference in the sense that there are some positions on the wall that get *fewer* hits when both slits are open than they get when only one of the slits is open. Image credit: Wikipedia.

place a detector right next to each slit so we can see exactly the path each photon takes then something even more bizarre happens. The mere fact that we *measure* the path changes the photon's behavior, and now this "destructive interference" pattern is gone and the number of times a position is hit when two slits are open is the sum of the number of times it is hit when each slit is open.

You should read the paragraphs above more than once and make sure you appreciate how truly mind boggling these results are.

22.2 QUANTUM AMPLITUDES

The double slit and other experiments ultimately forced scientists to accept the following picture of the world. Let us go back to the baseball experiment. Suppose that the probability a ball passes through the left slit is p_L and the probability that it passes through the right slit is p_R. Then, if we shoot N balls out of each gun, we expect the wall will be hit $(p_L + p_R)N$ times.

In the quantum world, it can sometimes be the case that in both the first and second case the wall is hit with positive probabilities p_L and p_R respectively but somehow when both slits are open the wall (or a particular position in it) is not hit at all. It's almost as if the probabilities can "cancel each other out".

To understand the way we model this in quantum mechanics, it is helpful to think of a "lazy evaluation" approach to probability. We can think of a probabilistic experiment such as shooting a baseball through two slits in two different ways:

- When a ball is shot, "nature" tosses a coin and decides if it will

go through the left slit (which happens with a certain probability p_L), right slit (which happens with a certain probability p_R), or bounce back. If it passes through one of the slits then it will hit the wall. Later we can look at the wall and find out whether or not this event happened, but the fact that the event happened or not is determined independently of whether or not we look at the wall.

- The other viewpoint is that when a ball is shot, "nature" computes the probabilities p_L and p_R as before, but does *not* yet "toss the coin" and determines what happened. Only when we actually look at the wall, nature tosses a coin and with probability $p_L + p_R$ ensures we see a dent. That is, nature uses "lazy evaluation", and only determines the result of a probabilistic experiment when we decide to *measure* it.

While the first scenario seems much more natural, the end result in both is the same (the wall is hit with probability $p_L + p_R$) and so the question of whether we should model nature as following the first scenario or second one seems like asking about the proverbial tree that falls in the forest with no one hearing about it.

However, when we want to describe the double slit experiment with photons rather than baseballs, it is the second scenario that lends itself better to a quantum generalization. Quantum mechanics associates a number α known as an *amplitude* with each probabilistic experiment. This number α can be *negative*, and in fact even *complex*. We never observe the amplitudes directly, since whenever we *measure* an event with amplitude α, nature tosses a coin and determines that the event happens with probability $|\alpha|^2$. However, the sign (or in the complex case, phase) of the amplitudes can affect whether two different events have *constructive* or *destructive* interference.

If you don't find the above description confusing and unintuitive, you probably didn't get it. Please make sure to re-read the above paragraphs until you are thoroughly confused.

Specifically, consider an event that can either occur or not (e.g. "detector number 17 was hit by a photon"). In classical probability, we model this by a probability distribution over the two outcomes: a pair of non-negative numbers p and q such that $p + q = 1$, where p corresponds to the probability that the event occurs and q corresponds to the probability that the event does not occur. In quantum mechanics, we model this also by pair of numbers, which we call *amplitudes*. This is a pair of (potentially negative or even complex) numbers α and β such that $|\alpha|^2 + |\beta|^2 = 1$. The probability that the event occurs is $|\alpha|^2$

and the probability that it does not occur is $|\beta|^2$. In isolation, these negative or complex numbers don't matter much, since we anyway square them to obtain probabilities. But the interaction of positive and negative amplitudes can result in surprising *cancellations* where somehow combining two scenarios where an event happens with positive probability results in a scenario where it never does.

> (R) **Complex vs real, other simplifications** If (like the author) you are a bit intimidated by complex numbers, don't worry: you can think of all amplitudes as *real* (though potentially *negative*) numbers without loss of understanding. All the "magic" of quantum computing already arises in this case, and so we will often restrict attention to real amplitudes in this chapter.
>
> We will also only discuss so-called *pure* quantum states, and not the more general notion of *mixed* states. Pure states turn out to be sufficient for understanding the algorithmic aspects of quantum computing.
>
> More generally, this chapter is not meant to be a complete description of quantum mechanics, quantum information theory, or quantum computing, but rather illustrate the main points where these differ from classical computing.

Quantum mechanics is a mathematical theory that allows us to calculate and predict the results of the double-slit and many other experiments. If you think of quantum mechanics as an explanation as to what "really" goes on in the world, it can be rather confusing. However, if you simply "shut up and calculate" then it works amazingly well at predicting experimental results. In particular, in the double slit experiment, for any position in the wall, we can compute numbers α and β such that photons from the first and second slit hit that position with probabilities $|\alpha|^2$ and $|\beta|^2$ respectively. When we open both slits, the probability that the position will be hit is proportional to $|\alpha + \beta|^2$, and so in particular, if $\alpha = -\beta$ then it will be the case that, despite being hit when *either* slit one or slit two are open, the position is *not hit at all* when they both are. If you are confused by quantum mechanics, you are not alone: for decades people have been trying to come up with explanations for "the underlying reality" behind quantum mechanics, from Bohmian mechanics to Many worlds as well as many others. However, none of these interpretation have gained universal acceptance and all of those (by design) yield the same experimental predictions. Thus at this point many scientists prefer to just ignore the question of what is the "true reality" and go back to simply "shutting up and calculating".

22.3 BELL'S INEQUALITY

There is something weird about quantum mechanics. In 1935 Einstein, Podolsky and Rosen (EPR) tried to pinpoint this issue by highlighting a previously unrealized corollary of this theory. They showed that the idea that nature does not determine the results of an experiment until it is measured results in so called "spooky action at a distance". Namely, making a measurement of one object may instantaneously effect the state (i.e., the vector of amplitudes) of another object in the other end of the universe.

Since the vector of amplitudes is just a mathematical abstraction, the EPR paper was considered to be merely a thought experiment for philosophers to be concerned about, without bearing on experiments. This changed when in 1965 John Bell showed an actual experiment to test the predictions of EPR and hence pit intuitive common sense against the quantum mechanics. Quantum mechanics won: it turns out that it *is* in fact possible to use measurements to create correlations between the states of objects far removed from one another that cannot be explained by any prior theory. Nonetheless, since the results of these experiments are so obviously wrong to anyone that has ever sat in an armchair, that there are still a number of Bell denialists arguing that this can't be true and quantum mechanics is wrong.

So, what is this Bell's Inequality? Suppose that Alice and Bob try to convince you they have telepathic ability, and they aim to prove it via the following experiment. Alice and Bob will be in separate closed rooms.[2] You will interrogate Alice and your associate will interrogate Bob. You choose a random bit $x \in \{0,1\}$ and your associate chooses a random $y \in \{0,1\}$. We let a be Alice's response and b be Bob's response. We say that Alice and Bob win this experiment if $a \oplus b = x \wedge y$. In other words, Alice and Bob need to output two bits that *disagree* if $x = y = 1$ and *agree* otherwise.[3]

Now if Alice and Bob are not telepathic, then they need to agree in advance on some strategy. It's not hard for Alice and Bob to succeed with probability $3/4$: just always output the same bit. However, by doing some case analysis, we can show that no matter what strategy they use, Alice and Bob cannot succeed with higher probability than that:[4]

Theorem 22.1 — Bell's Inequality. For every two functions f, g : $\{0,1\} \to \{0,1\}$, $\Pr_{x,y\in\{0,1\}}[f(x) \oplus g(y) = x \wedge y] \leq 3/4$.

Proof. Since the probability is taken over all four choices of $x, y \in \{0,1\}$, the only way the theorem can be violated if if there exist two

[2] If you are extremely paranoid about Alice and Bob communicating with one another, you can coordinate with your assistant to perform the experiment exactly at the same time, and make sure that the rooms are sufficiently far apart (e.g., are on two different continents, or maybe even one is on the moon and another is on earth) so that Alice and Bob couldn't communicate to each other in time the results of their respective coins even if they do so at the speed of light.

[3] This form of Bell's game was shown by Clauser, Horne, Shimony, and Holt.

[4] Theorem 22.1 below assumes that Alice and Bob use *deterministic* strategies f and g respectively. More generally, Alice and Bob could use a *probabilistic* strategy, or equivalently, each could choose f and g from some *distributions* $\mathcal{F}$ and $\mathcal{G}$ respectively. However the *averaging principle* (Remark 1.6.3) implies that if all possible deterministic strategies succeed with probability at most $3/4$, then the same is true for probabilistic strategies.

functions f, g that satisfy

$$f(x) \oplus g(y) = x \wedge y \tag{22.1}$$

for all the four choices of $x, y \in \{0,1\}^2$. Let's plug in all these four choices and see what we get (below we use the equalities $z \oplus 0 = z$, $z \wedge 0 = 0$ and $z \wedge 1 = z$):

$$\begin{aligned} f(0) \oplus g(0) &= 0 \quad (\text{plugging in } x=0, y=0) \\ f(0) \oplus g(1) &= 0 \quad (\text{plugging in } x=0, y=1) \\ f(1) \oplus g(0) &= 0 \quad (\text{plugging in } x=1, y=0) \\ f(1) \oplus g(1) &= 1 \quad (\text{plugging in } x=1, y=1) \end{aligned} \tag{22.2}$$

If we XOR together the first and second equalities we get $g(0) \oplus g(1) = 0$ while if we XOR together the third and fourth equalities we get $g(0) \oplus g(1) = 1$, thus obtaining a contradiction. ■

An amazing experimentally verified fact is that quantum mechanics allows for "telepathy".[5] Specifically, it has been shown that using the weirdness of quantum mechanics, there is in fact a strategy for Alice and Bob to succeed in this game with probability larger than $3/4$ (in fact, they can succeed with probability about 0.85, see Lemma 22.2).

[5] More accurately, one either has to give up on a "billiard ball type" theory of the universe or believe in telepathy (believe it or not, some scientists went for the latter option).

22.4 QUANTUM WEIRDNESS

Some of the counterintuitive properties that arise from quantum mechanics include:

- **Interference** - As we've seen, quantum amplitudes can "cancel each other out".
- **Measurement** - The idea that amplitudes are negative as long as "no one is looking" and "collapse" (by squaring them) to positive probabilities when they are *measured* is deeply disturbing. Indeed, as shown by EPR and Bell, this leads to various strange outcomes such as "spooky actions at a distance", where we can create correlations between the results of measurements in places far removed. Unfortunately (or fortunately?) these strange outcomes have been confirmed experimentally.
- **Entanglement** - The notion that two parts of the system could be connected in this weird way where measuring one will affect the other is known as *quantum entanglement*.

As counter-intuitive as these concepts are, they have been experimentally confirmed, so we just have to live with them.

R **More on quantum** The discussion in this lecture is quite brief and somewhat superficial. The chapter on quantum computation in my book with Arora (see draft here) is one relatively short resource that contains essentially everything we discuss here and more. See also this blog post of Aaronson for a high level explanation of Shor's algorithm which ends with links to several more detailed expositions. This lecture of Aaronson contains a great discussion of the feasibility of quantum computing (Aaronson's course lecture notes and the book that they spawned are fantastic reads as well). The videos of Umesh Vazirani'z EdX course are an accessible and recommended introduction to quantum computing. See the "bibliographical notes" section at the end of this chapter for more resources.

22.5 QUANTUM COMPUTING AND COMPUTATION - AN EXECUTIVE SUMMARY.

One of the strange aspects of the quantum-mechanical picture of the world is that unlike in the billiard ball example, there is no obvious algorithm to simulate the evolution of n particles over t time periods in $poly(n,t)$ steps. In fact, the natural way to simulate n quantum particles will require a number of steps that is *exponential* in n. This is a huge headache for scientists that actually need to do these calculations in practice.

In the 1981, physicist Richard Feynman proposed to "turn this lemon to lemonade" by making the following almost tautological observation:

> If a physical system cannot be simulated by a computer in T steps, the system can be considered as performing a computation that would take more than T steps.

So, he asked whether one could design a quantum system such that its outcome y based on the initial condition x would be some function $y = f(x)$ such that **(a)** we don't know how to efficiently compute in any other way, and **(b)** is actually useful for something.[6] In 1985, David Deutsch formally suggested the notion of a quantum Turing machine, and the model has been since refined in works of Deutsch and Josza and Bernstein and Vazirani. Such a system is now known as a *quantum computer*.

For a while these hypothetical quantum computers seemed useful for one of two things. First, to provide a general-purpose mecha-

[6] As its title suggests, Feynman's lecture was actually focused on the other side of simulating physics with a computer. However, he mentioned that as a "side remark" one could wonder if it's possible to simulate physics with a new kind of computer - a "quantum computer" which would "not [be] a Turing machine, but a machine of a different kind". As far as I know, Feynman did not suggest that such a computer could be useful for computations completely outside the domain of quantum simulation. Indeed, he was more interested in the question of whether quantum mechanics could be simulated by a classical computer.

nism to simulate a variety of the real quantum systems that people care about, such as various interactions inside molecules in quantum chemistry. Second, as a challenge to the *Extended Church Turing hypothesis* which says that every physically realizable computation device can be modeled (up to polynomial overhead) by Turing machines (or equivalently, NAND++ / NAND« programs).

Quantum chemistry is important (and in particular understanding it can be a bottleneck for designing new materials, drugs, and more), but it is still a rather niche area within the broader context of computing (and even scientific computing) applications. Hence for a while most researchers (to the extent they were aware of it), thought of quantum computers as a theoretical curiosity that has little bearing to practice, given that this theoretical "extra power" of quantum computer seemed to offer little advantage in the majority of the problems people want to solve in areas such as combinatorial optimization, machine learning, data structures, etc..

To some extent this is still true today. As far as we know, quantum computers, if built, will *not* provide exponential speed ups for 95% of the applications of computing.[7] In particular, as far as we know, quantum computers will *not* help us solve **NP** complete problems in polynomial or even sub-exponential time, though *Grover's algorithm* (Remark 22.5) does yield a quadratic advantage in many cases.

However, there is one cryptography-sized exception: In 1994 Peter Shor showed that quantum computers can solve the integer factoring and discrete logarithm in polynomial time. This result has captured the imagination of a great many people, and completely energized research into quantum computing. This is both because the hardness of these particular problems provides the foundations for securing such a huge part of our communications (and these days, our economy), as well as it was a powerful demonstration that quantum computers could turn out to be useful for problems that a-priori seemd to have nothing to do with quantum physics.

As we'll discuss later, at the moment there are several intensive efforts to construct large scale quantum computers. It seems safe to say that, as far as we know, in the next five years or so there will not be a quantum computer large enough to factor, say, a 1024 bit number, but there it is quite possible that some quantum computer will be built that is strong enough to achieve some task that is too inefficient to achieve with a non-quantum or "classical" computer (or at least requires far more resources classically than it would for this computer). When and if such a computer is built that can break reasonable parameters of Diffie Hellman, RSA and elliptic curve cryptography is anybody's guess. It could also be a "self destroying prophecy" whereby the existence of a small-scale quantum computer would

[7] This "95 percent" is a figure of speech, but not completely so. At the time of this writing, cryptocurrency mining electricity consumption is estimated to use up at least 70Twh or 0.3 percent of the world's production, which is about 2 to 5 percent of the total energy usage for the computing industry. All the current cryptocurrencies will be broken by quantum computers. Also, for many web servers the TLS protocol (which is based on the current non-lattice based systems would be completely broken by quantum computing) is responsible for about 1 percent of the CPU usage.

cause everyone to shift away to lattice-based crypto which in turn will diminish the motivation to invest the huge resources needed to build a large scale quantum computer.[8]

[8] Of course, given that we're still hearing of attacks exploiting "export grade" cryptography that was supposed to disappear in 1990's, I imagine that we'll still have products running 1024 bit RSA when everyone has a quantum laptop.

> **R** **Quantum computing and NP** Despite popular accounts of quantum computers as having variables that can take "zero and one at the same time" and therefore can "explore an exponential number of possibilities simultaneously", their true power is much more subtle and nuanced. In particular, as far as we know, quantum computers do *not* enable us to solve **NP** complete problems such as 3SAT in polynomial or even sub-exponential time. However, Grover's search algorithm does give a more modest advantage (namely, quadratic) for quantum computers over classical ones for problems in **NP**. In particular, due to Grover's search algorithm, we know that the k-SAT problem for n variables can be solved in time $O(2^{n/2}poly(n))$ on a quantum computer for every k. In contrast, the best known algorithms for k-SAT on a classical computer take roughly $2^{(1-\frac{1}{k})n}$ steps.

22.6 QUANTUM SYSTEMS

Before we talk about *quantum* computing, let us recall how we physically realize "vanilla" or *classical* computing. We model a *logical bit* that can equal 0 or a 1 by some physical system that can be in one of two states. For example, it might be a wire with high or low voltage, charged or uncharged capacitor, or even (as we saw) a pipe with or without a flow of water, or the presence or absence of a soldier crab. A *classical* system of n bits is composed of n such "basic systems", each of which can be in either a "zero" or "one" state. We can model the state of such a system by a string $s \in \{0,1\}^n$. If we perform an operation such as writing to the 17-th bit the NAND of the 3rd and 5th bits, this corresponds to applying a *local* function to s such as setting $s_{17} = 1 - s_3 \cdot s_5$.

In the *probabilistic* setting, we would model the state of the system by a *distribution*. For an individual bit, we could model it by a pair of non-negative numbers α, β such that $\alpha + \beta = 1$, where α is the probability that the bit is zero and β is the probability that the bit is one. For example, applying the *negation* (i.e., NOT) operation to this bit corresponds to mapping the pair (α, β) to (β, α) since the probability that $NOT(\sigma)$ is equal to 1 is the same as the probability that σ is equal to 0. This means that we can think of the NOT function as the linear map $N : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ such that $N \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}$ or equivalently as the matrix

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

If we think of the n-bit system as a whole, then since the n bits can take one of 2^n possible values, we model the state of the system as a vector p of 2^n probabilities. For every $s \in \{0,1\}^n$, we denote by e_s the 2^n dimensional vector that has 1 in the coordinate corresponding to s (identifying it with a number in $[2^n]$), and so can write p as $\sum_{s\in\{0,1\}^n} p_s e_s$ where p_s is the probability that the system is in the state s.

Applying the operation above of setting the 17-th bit to the NAND of the 3rd and 5th bits, corresponds to transforming the vector p to the vector Fp where $F : \mathbb{R}^{2^n} \to \mathbb{R}^{2^n}$ is the linear map that maps e_s to $e_{s_0\cdots s_{16}(1-s_3\cdot s_5)s_{18}\cdots s_{n-1}}$.[9]

[9] Since $\{e_s\}_{s\in\{0,1\}^n}$ is a *basis* for R^{2^n}, it sufficies to define the map F on vectors of this form.

Please make sure you understand why performing the operation will take a system in state p to a system in the state Fp. Understanding the evolution of probabilistic systems is a prerequisite to understanding the evolution of quantum systems.

If your linear algebra is a bit rusty, now would be a good time to review it, and in particular make sure you are comfortable with the notions of *matrices*, *vectors*, (orthogonal and orthonormal) *bases*, and *norms*.

22.6.1 Quantum amplitudes

In the quantum setting, the state of an individual bit (or "qubit", to use quantum parlance) is modeled by a pair of numbers (α, β) such that $|\alpha|^2 + |\beta|^2 = 1$. While in general these numbers can be *complex*, for the rest of this chapter, we will often assume they are *real* (though potentially negative), and hence often drop the absolute value operator. (This turns out not to make much of a difference in explanatory power.) As before, we think of α^2 as the probability that the bit equals 0 and β^2 as the probability that the bit equals 1. As we did before, we can model the NOT operation by the map $N : \mathbb{R}^2 \to \mathbb{R}^2$ where $N(\alpha, \beta) = (\beta, \alpha)$.

Following quantum tradition, instead of using e_0 and e_1 as we did above, from now on we will denote the vector $(1, 0)$ by $|0\rangle$ and the vector $(0, 1)$ by $|1\rangle$ (and moreover, think of these as column vectors). This is known as the Dirac "ket" notation. This means that NOT is the unique linear map $N : \mathbb{R}^2 \to \mathbb{R}^2$ that satisfies $N|0\rangle = |1\rangle$ and $N|1\rangle = |0\rangle$. In other words, in the quantum case, as in the probabilistic

case, NOT corresponds to the matrix

$$N = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} . \tag{22.3}$$

In classical computation, we typically think that there are only two operations that we can do on a single bit: keep it the same or negate it. In the quantum setting, a single bit operation corresponds to any linear map $OP : \mathbb{R}^2 \to \mathbb{R}^2$ that is *norm preserving* in the sense that for every α, β, if we apply OP to the vector $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ then we obtain a vector $\begin{pmatrix} \alpha' \\ \beta' \end{pmatrix}$ such that $\alpha'^2 + \beta'^2 = \alpha^2 + \beta^2$. Such a linear map OP corresponds to a unitary two by two matrix.[10] Keeping the bit the same corresponds to the matrix $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and (as we've seen) the NOT operations corresponds to the matrix $N = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. But there are other operations we can use as well. One such useful operation is the *Hadamard* operation, which corresponds to the matrix

$$H = \tfrac{1}{\sqrt{2}} \begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix} . \tag{22.4}$$

In fact it turns out that Hadamard is all that we need to add to a classical universal basis to achieve the full power of quantum computing.

[10] As we mentioned, quantum mechanics actually models states as vectors with *complex* coordinates. However, this does not make any qualitative difference to our discussion.

22.6.2 Recap

The *state* of a *quantum system* of n qubits is modeled by an 2^n dimensional vector ψ of unit norm (i.e., squares of all coordinates sums up to 1), which we write as $\psi = \sum_{x \in \{0,1\}^n} \psi_x |x\rangle$ where $|x\rangle$ is the column vector that has 0 in all coordinates except the one corresponding to x (identifying $\{0,1\}^n$ with the numbers $\{0, \ldots, 2^n - 1\}$). We use the convention that if a, b are strings of lengths k and ℓ respectively then we can write the $2^{k+\ell}$ dimensional vector with 1 in the ab-th coordinate and zero elsewhere not just as $|ab\rangle$ but also as $|a\rangle|b\rangle$. In particular, for every $x \in \{0,1\}^n$, we can write the vector $|x\rangle$ also as $|x_0\rangle|x_1\rangle \cdots |x_{n-1}\rangle$. This notation satisfies certain nice distributive laws such as $|a\rangle(|b\rangle + |b'\rangle)|c\rangle = |abc\rangle + |ab'c\rangle$.

A *quantum operation* on such a system is modeled by a $2^n \times 2^n$ *unitary matrix* U (one that satisfies $UU^\top = I$ where $U^\top$ is the *transpose* operation, or conjugate transpose for complex matrices). If the system is in state ψ and we apply to it the operation U, then the new state of the system is $U\psi$.

When we *measure* an n-qubit system in a state $\psi = \sum_{x\in\{0,1\}^n} \psi_x |x\rangle$, then we observe the value $x \in \{0,1\}^n$ with probability $|\psi_x|^2$. In this case, the system *collapses* to the state $|x\rangle$.

22.7 ANALYSIS OF BELL'S INEQUALITY (OPTIONAL)

Now that we have the notation in place, we can show a strategy for Alice and Bob to display "quantum telepathy" in Bell's Game. Recall that in the classical case, Alice and Bob can succeed in the "Bell Game" with probability at most $3/4 = 0.75$. We now show that quantum mechanics allows them to succeed with probability at least 0.8.[11]

[11] The strategy we show is not the best one. Alice and Bob can in fact succeed with probability $\cos^2(\pi/8) \sim 0.854$.

Lemma 22.2 There is a 2-qubit quantum state $\psi \in \mathbb{C}^4$ so that if Alice has access to the first qubit of ψ, can manipulate and measure it and output $a \in \{0,1\}$ and Bob has access to the second qubit of ψ and can manipulate and measure it and output $b \in \{0,1\}$ then $\Pr[a \oplus b = x \wedge y] \geq 0.8$.

Proof. Alice and Bob will start by preparing a 2-qubit quantum system in the state

$$\psi = \tfrac{1}{\sqrt{2}}|00\rangle + \tfrac{1}{\sqrt{2}}|11\rangle \tag{22.5}$$

(this state is known as an EPR pair). Alice takes the first qubit of the system to her room, and Bob takes the qubit to his room. Now, when Alice receives x if $x = 0$ she does nothing and if $x = 1$ she applies the unitary map $R_{-\pi/8}$ to her qubit where $R_\theta = \begin{pmatrix} cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$ is the unitary operation corresponding to rotation in the plane with angle θ. When Bob receives y, if $y = 0$ he does nothing and if $y = 1$ he applies the unitary map $R_{\pi/8}$ to his qubit. Then each one of them measures their qubit and sends this as their response.

Recall that to win the game Bob and Alice want their outputs to be more likely to differ if $x = y = 1$ and to be more likely to agree otherwise. We will split the analysis in one case for each of the four possible values of x and y.

Case 1: $x = y = 0$ If $x = y = 0$ then the state does not change. Because the state ψ is proportional to $|00\rangle + |11\rangle$, the measurements of Bob and Alice will always agree (if Alice measures 0 then the state collapses to $|00\rangle$ and so Bob measures 0 as well, and similarly for 1). Hence in the case $x = y = 1$, Alice and Bob always win.

Case 2: $x = 0$,$y = 1$ If $x = 0$ and $y = 1$ then after Alice measures her bit, if she gets 0 then the system collapses to the state $|00\rangle$, in which case after Bob performs his rotation, his qubit is in the state $\cos(\pi/8)|0\rangle + \sin(\pi/8)|1\rangle$. Thus, when Bob measures his qubit, he will get 0 (and hence agree with Alice) with probability $\cos^2(\pi/8) \geq 0.85$.

Similarly, if Alice gets 1 then the system collapses to $|11\rangle$, in which case after rotation Bob's qubit will be in the state $-\sin(\pi/8)|0\rangle + \cos(\pi/8)|1\rangle$ and so once again he will agree with Alice with probability $\cos^2(\pi/8)$. The analysis for **Case 3**, where $x = 1$ and $y = 0$, is symetric, and hence they will agree with probability $\cos^2(\pi/8)$ in this case as well.[12]

[12] We are using the (not too hard) observation that the result of this experiment is the same regardless of the order in which Alice and Bob apply their rotations and measurements.

Case 4: $x = y = 1$ For the case that $x = 1$ and $y = 1$, after both Alice and Bob perform their rotations, the state will be proportional to

$$R_{-\pi/8}|0\rangle R_{\pi/8}|0\rangle + R_{-\pi/8}|1\rangle R_{\pi/8}|1\rangle \ . \tag{22.6}$$

Opening up the coefficients and using $\cos(-x) = \cos(x)$ and $\sin(-x) = -\sin(x)$, we can see this is proportional to

$$\cos^2(\pi/8)|00\rangle + \cos(\pi/8)\sin(\pi/8)|01\rangle - \sin(\pi/8)\cos(\pi/8)|10\rangle + \sin^2(\pi/8)|11\rangle - \sin^2(\pi/8)|00\rangle + \sin(\pi/8)\cos(\pi/8)|01\rangle - \cos(\pi/8)s \tag{22.7}$$

using the trigonometric identities $2\sin(\alpha)\cos(\alpha) = \sin(2\alpha)$ and $\cos(\alpha) - \sin^2(\alpha) = \cos(2\alpha)$, we see that the probability of getting any one of $|00\rangle, |10\rangle, |01\rangle, |11\rangle$ is proportional to $\cos(\pi/4) = \sin(\pi/4) = \frac{1}{\sqrt{2}}$. Hence all four options for (a, b) are equally likely, which mean that in this case $a = b$ with probability 0.5.

Taking all the four cases together, we see that the overall probability of winning the game is at least $\frac{1}{4} \cdot 1 + \frac{1}{2} \cdot 0.85 + \frac{1}{4} \cdot 0.5 = 0.8$. ■

R **Quantum vs probabilistic strategies** It is instructive to understand what is it about quantum mechanics that enabled this gain in Bell's Inequality. For this, consider the following analogous probabilistic strategy for Alice and Bob. They agree that each one of them output 0 if he or she get 0 as input and outputs 1 with probability p if they get 1 as input. In this case one can see that their success probability would be $\frac{1}{4} \cdot 1 + \frac{1}{2}(1-p) + \frac{1}{4}[2p(1-p)] = 0.75 - 0.5p^2 \leq 0.75$. The quantum strategy we described above can be thought of as a variant of the probabilistic strategy for parameter p set to $\sin^2(\pi/8) = 0.15$. But in the case $x = y = 1$, instead of disagreeing only with probability $2p(1-p) = 1/4$, because we can use these negative probabilities in the quantum world and rotate the state in opposite directions, and hence the probability of disagreement ends up being $\sin^2(\pi/4) = 0.5$.

22.8 QUANTUM COMPUTATION

Recall that in the classical setting, we modeled computation as obtained by a sequence of *basic operations*. We had two types of computa-

tional models:

- *Non uniform models of computation* such as Boolean circuits and NAND program, where a finite function $f : \{0,1\}^n \rightarrow \{0,1\}$ is computable in size T if it can be expressed as a combination of T basic operations (gates in a circuit or lines in a NAND program)

- *Uniform models of computation* such as Turing machines and NAND++ programs, where an infinite function $F : \{0,1\}^* \rightarrow \{0,1\}$ is computable in time $T(n)$ if there is a single algorithm that on input $x \in \{0,1\}^n$ evaluates $F(x)$ using at most $T(n)$ basic steps.

When considering *efficient computation*, we defined the class **P** to consist of all infinite functions $F : \{0,1\}^* \rightarrow \{0,1\}$ that can be computed by a Turing machine or NAND++ program in time $p(n)$ for some polynomial $p(\cdot)$. We defined the class $\mathbf{P}_{/\mathbf{poly}}$ to consists of all infinite functions $F : \{0,1\}^* \rightarrow \{0,1\}$ such that for every n, the restriction F_n of F to $\{0,1\}^n$ can be computed by a Boolean circuit or NAND program of size at most $p(n)$ for some polynomial $p(\cdot)$.

We will do the same for *quantum computation*, focusing mostly on the *non uniform* setting of quantum circuits, since that is simpler, and already illustrates the important differences with classical computing.

22.8.1 Quantum circuits

A *quantum circuit* is analogous to a Boolean circuit, and can be described as a directed acyclic graph. One crucial difference that the *out degree* of every vertex in a quantum circuit is at most one. This is because we cannot "reuse" quantum states without *measuring* them (which collapses their "probabilities"). Therefore, we cannot use the same qubit as input for two different gates.[13] Another more technical difference is that to express our operations as unitary matrices, we will need to make sure all our gates are *reversible*. This is not hard to ensure. For example, in the quantum context, instead of thinking of $NAND$ as a (non reversible) map from $\{0,1\}^2$ to $\{0,1\}$, we will think of it as the reversible map on *three* qubits that maps a, b, c to $a, b, c \oplus NAND(a,b)$ (i.e., flip the last bit if $NAND$ of the first two bits is 1). Equivalently, the NAND operation corresponds to the 8×8 unitary matrix U_{NAND} such that (identifying $\{0,1\}^3$ with $[8]$) for every $a, b, c \in \{0,1\}$, if $|abc\rangle$ is the basis element with 1 in the abc-th coordinate and zero elsewhere, then $U_{NAND}|abc\rangle = |ab(c \oplus NAND(a,b))\rangle$.[14] If we order the rows and columns as $000, 001, 010, \ldots, 111$, then U_{NAND} can be written as the

[13] This is known as the No Cloning Theorem.

[14] Readers familiar with quantum computing should note that U_{NAND} is a close variant of the so called Toffoli gate and so QNAND programs correspond to quantum circuits with the Hadamard and Toffoli gates.

following matrix:

$$U_{NAND} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{22.8}$$

If we have an n qubit system, then for $i, j, k \in [n]$, we will denote by $U_{NAND}^{i,j,k}$ as the $2^n \times 2^n$ unitary matrix that corresponds to applying U_{NAND} to the i-th, j-th, and k-th bits, leaving the others intact. That is, for every $v = \sum_{x\in\{0,1\}^n} v_x|x\rangle$, $U_{NAND}^{i,j,k}v = \sum_{x\in\{0,1\}^n} v_x|x_0 \cdots x_{k-1}(x_k \oplus NAND(x_i, x_j))x_{k+1} \cdots x_{n-1}\rangle$.

As mentioned above, we will also use the *Hadamard* or HAD operation, A *quantum circuit* is obtained by applying a sequence of U_{NAND} and HAD gates, which correspond to the matrix

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix} . \tag{22.9}$$

Another way to write define H is that for $b \in \{0, 1\}$, $H|b\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}(-1)^b|1\rangle$. We define HAD^i to be the $2^n \times 2^n$ unitary matrix that applies HAD to the i-th qubit and leaves the others intact. Using the ket notation, we can write this as

$$HAD^i \sum_{x\in\{0,1\}^n} v_x|x\rangle = \frac{1}{\sqrt{2}} \sum_{x\in\{0,1\}^n} |x_0 \cdots x_{i-1}\rangle \left(|0\rangle + (-1)^{x_i}|1\rangle\right) |x_i \cdots x_{n-1}\rangle . \tag{22.10}$$

A *quantum circuit* is obtained by composing these basic operations on some m qubits. If $m \geq n$, we use a circuit to compute a function $f : \{0, 1\}^n \to \{0, 1\}$:

- On input x, we initialize the system to hold $x_0, \ldots, x_{n-1}$ in the first n qubits, and initialize all remaining $m - n$ qubits to zero.
- We execute each elementary operation one by one.
- At the end of the computation, we *measure* the system, and output the result of the last qubit (i.e. the qubit in location $m - 1$).[15]
- We say that the circuit *computes* f, if the probability that this output equals $f(x)$ is at least $2/3$. Note that this probability is obtained by summing up the squares of the amplitudes of all coordinates in the final state of the system corresponding to vectors $|y\rangle$ where $y_{m-1} = f(x)$.

[15] For simplicity we restrict attention to functions with a single bit of output, though the definition of quantum circuits naturally extends to circuits with multiple outputs.

Formally this is defined as follows:

Definition 22.3 — Quantum circuit. A *quantum circuit of m inputs and s gates* over the $\{U_{NAND}, HAD\}$ basis is a sequence of s unitary $2^n \times 2^n$ matrices $U_0, \ldots, U_{s-1}$ such that each matrix U_ℓ is either of the form $NAND^{i,j,k}$ for $i, j, k \in [n]$ or HAD^i for $i \in [n]$.

A quantum circuit *computes* a function $f : \{0,1\}^n \to \{0,1\}$ if the following is true for every $x \in \{0,1\}^n$:

Let v be the vector

$$v = U_{s-1}U_{s-2}\cdots U_1 U_0 |x0^{m-n}\rangle \tag{22.11}$$

and write v as $\sum_{y\in\{0,1\}^m} v_y|y\rangle$. Then

$$\sum_{y\in\{0,1\}^m \text{ s.t. } y_{m-1}=f(x)} |v_y|^2 \geq \frac{2}{3} . \tag{22.12}$$

P Please stop here and see that this definition makes sense to you.

Once we have the notion of quantum circuits, we can define the quantum analog of $\mathbf{P}_{/\mathbf{poly}}$ (i.e., define the class of functions computable by *polynomial size quantum circuits*) as follows:

Definition 22.4 — $\mathbf{BQP}_{/\mathbf{poly}}$. Let $F : \{0,1\}^* \to \{0,1\}$. We say that $F \in \mathbf{BQP}_{/\mathbf{poly}}$ if there exists some polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for every $n \in \mathbb{N}$, if F_n is the restriction of F to inputs of length n, then there is a quantum circuit of size at most $p(n)$ that computes F_n.

R **The obviously exponential fallacy** A priori it might seem "obvious" that quantum computing is exponentially powerful, since to perform a quantum computation on n bits we need to maintain the 2^n dimensional state vector and apply $2^n \times 2^n$ matrices to it. Indeed popular descriptions of quantum computing (too) often say something along the lines that the difference between quantum and classical computer is that a classical bit can either be zero or one while a qubit can be in both states at once, and so in many qubits a quantum computer can perform exponentially many computations at once.

Depending on how you interpret it, this description is either false or would apply equally well to *proba-*

bilistic computation, even though we've already seen that every randomized algorithm can be simulated by a similar-sized circuit, and in fact we conjecture that **BPP** $=$ **P**.

Moreover, this "obvious" approach for simulating a quantum computation will take not just exponential time but *exponential space* as well, while can be shown that using a simple recursive formula one can calculate the final quantum state using *polynomial space* (in physics this is known as "Feynman path integrals"). So, the exponentially long vector description by itself does not imply that quantum computers are exponentially powerful. Indeed, we cannot *prove* that they are (i.e., as far as we know, every QNAND program could be simulated by a NAND program with polynomial overhead), but we do have some problems (integer factoring most prominently) for which they do provide exponential speedup over the currently best *known* classical (deterministic or probabilistic) algorithms.

22.8.2 QNAND programs (optional)

Just like in the classical case, there is an equivalence between circuits and straightline programs, and so we can define the programming language QNAND that is the quantum analog of our NAND programming language. To do so, we only add a single operation: `HAD(foo)` which applies the single-bit operation H to the variable `foo`. We also use the following interpretation to make **NAND** reversible: `foo = NAND(bar,blah)` means that we modify `foo` to be the XOR of its original value and the NAND of `bar` and `blah`. (In other words, apply the 8 by 8 unitary transformation U_{NAND} defined above to the three qubits corresponding to `foo`, `bar` and `blah`.) If `foo` is initialized to zero then this makes no difference.

If P is a QNAND program with n input variables, ℓ workspace variables, and m output variables, then running it on the input $x \in \{0,1\}^n$ corresponds to setting up a system with $n + m + \ell$ qubits and performing the following process:

1. We initialize the input variables **X**`[0]` ... **X**`[`$n-1$`]` to $x_0, \ldots, x_{n-1}$ and all other variables to 0.

2. We execute the program line by line, applying the corresponding physical operation H or U_{NAND} to the qubits that are referred to by the line.

3. We *measure* the output variables **Y**`[0]`, ..., **Y**`[`$m-1$`]` and output the result (if there is more than one output then we measure more variables).

22.8.3 Uniform computation

Just as in the classical case, we can define *uniform* computational models. For example, we can define the *QNAND++ programming language* to be QNAND augmented with loops and arrays just like NAND++ is obtained from NAND. Using this we can define the class **BQP** which is the uniform analog of $\mathbf{BQP}_{/\mathbf{poly}}$. Just as in the classical setting it holds that $\mathbf{BPP} \subseteq \mathbf{P}_{/\mathbf{poly}}$, in the quantum setting it can be shown that $\mathbf{BQP} \subseteq \mathbf{BQP}_{/\mathbf{poly}}$. Just like the classical case, we can also use Quantum Turing Machines instead of QNAND++ to define **BQP**.

Yet another way to define **BQP** is the following: a function $F : \{0,1\}^* \rightarrow \{0,1\}$ is in **BQP** if **(1)** $F \in \mathbf{BQP}_{/\mathbf{poly}}$ and **(2)** moreover for every n, the quantum circuit that verifies this can be generated by a *classical polynomial time NAND++ program* (or, equivalently, a polynomial-time Turing machine).[16] We use this definition here, though an equivalent one can be made using QNAND++ or quantum Turing machines:

[16] This is analogous to the alternative characterization of **P** that appears in Exercise 12.4.

Definition 22.5 — The class BQP. Let $F : \{0,1\}^* \rightarrow \{0,1\}$. We say that $F \in \mathbf{BQP}$ if there exists a polynomial time NAND++ program P such that for every n, $P(1^n)$ is the description of a quantum circuit C_n that computes the restriction of F to $\{0,1\}^n$.

P One way to verify that you've understood these definitions it to see that you can prove **(1)** $\mathbf{P} \subseteq \mathbf{BQP}$ and in fact the stronger statement $\mathbf{BPP} \subseteq \mathbf{BQP}$, **(2)** $\mathbf{BQP} \subseteq \mathbf{EXP}$, and **(3)** For every **NP**-complete function F, if $F \in \mathbf{BQP}$ then $\mathbf{NP} \subseteq \mathbf{BQP}$. See Exercise 22.1.

The relation between **NP** and **BQP** is not known (see also Remark 22.5). It is widely believed that $\mathbf{NP} \not\subseteq \mathbf{BQP}$, but there is no consensus whether or not $\mathbf{BQP} \subseteq \mathbf{NP}$. It is possible that these two classes are *incomparable*, in the sense that $\mathbf{NP} \not\subseteq \mathbf{BQP}$ (and in particular no **NP**-complete function belongs to **BQP**) but also $\mathbf{BQP} \not\subseteq \mathbf{NP}$ (and there are some interesting candidates for such problems).

It can be shown that $QNANDEVAL$ (evaluating a quantum circuit on an input) is computable by a polynomial size QNAND program, and moreover this program can even be generated *uniformly* and hence $QNANDEVAL$ is in **BQP**. This allows us to "port" many of the results of classical computational complexity into the quantum realm as well.

R **Restricting attention to circuits** Because the non uniform model is a little cleaner to work with, in the

rest of this chapter we mostly restrict attention to this model, though all the algorithms we discuss can be implemented in uniform computation as well.

22.9 PHYSICALLY REALIZING QUANTUM COMPUTATION

To realize quantum computation one needs to create a system with n independent binary states (i.e., "qubits"), and be able to manipulate small subsets of two or three of these qubits to change their state. While by the way we defined operations above it might seem that one needs to be able to perform arbitrary unitary operations on these two or three qubits, it turns out that there several choices for *universal sets* - a small constant number of gates that generate all others. The biggest challenge is how to keep the system from being measured and *collapsing* to a single classical combination of states. This is sometimes known as the *coherence time* of the system. The threshold theorem says that there is some absolute constant level of errors τ so that if errors are created at every gate at rate smaller than τ then we can recover from those and perform arbitrary long computations. (Of course there are different ways to model the errors and so there are actually several threshold *theorems* corresponding to various noise models).

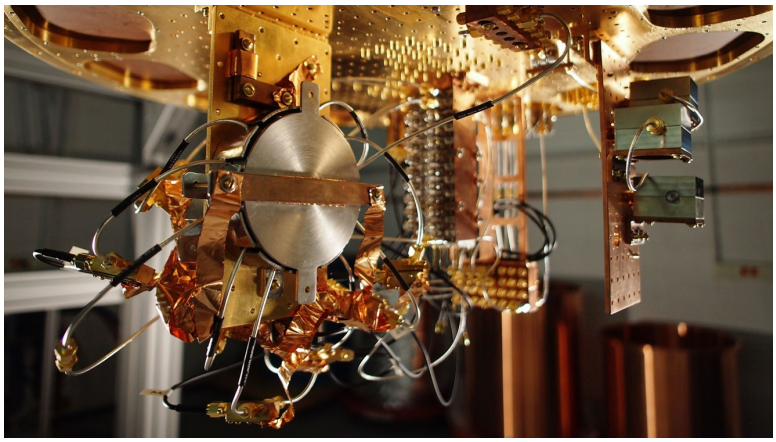

Figure 22.3: Superconducting quantum computer prototype at Google. Image credit: Google / MIT Technology Review.

There have been several proposals to build quantum computers:

- Superconducting quantum computers use superconducting electric circuits to do quantum computation. This is the direction where there has been most recent progress towards "beating" classical computers (see Fig. 22.3).
- Trapped ion quantum computers Use the states of an ion to simulate a qubit. People have made some recent advances on these

computers too. While it's not at all clear that's the right measuring yard, the current best implementation of Shor's algorithm (for factoring 15) is done using an ion-trap computer.

- Topological quantum computers use a different technology. Topological qubits are more stable by design and hence error correction is less of an issue, but constructing them is extremely challenging.

These approaches are not mutually exclusive and it could be that ultimately quantum computers are built by combining all of them together. In the near future, it seems that we will not be able to achieve full fledged large scale universal quantum computers, but rather more restricted machines, sometimes called "Noisy Intermediate-Scale Quantum Computers" or "NISQ". See this article by John Preskil for some of the progress and applications of such more restricted devices.

22.10 SHOR'S ALGORITHM: HEARING THE SHAPE OF PRIME FACTORS

Bell's Inequality is a powerful demonstration that there is something very strange going on with quantum mechanics. But could this "strangeness" be of any use to solve computational problems not directly related to quantum systems? A priori, one could guess the answer is *no*. In 1994 Peter Shor showed that one would be wrong:

Theorem 22.6 — Shor's Algorithm. There is a polynomial-time quantum algorithm that on input an integer M (represented in base two), outputs the prime factorization of M.

Another way to state Theorem 22.6 is that if we define $FACTORING : \{0,1\}^* \to \{0,1\}$ to be the function that on input a pair of numbers (M, X) outputs 1 if and only if M has a factor P such that $2 \leq P \leq X$, then $FACTORING$ is in **BQP**. This is an exponential improvement over the best known classical algorithms, which take roughly $2^{\tilde{O}(n^{1/3})}$ time, where the $\tilde{O}$ notation hides factors that are polylogarithmic in n. While we will not prove Theorem 22.6 in this chapter, we will sketch some of the ideas behind the proof.

22.10.1 Period finding

At the heart of Shor's Theorem is an efficient quantum algorithm for finding *periods* of a given function. For example, a function $f : \mathbb{R} \to \mathbb{R}$ is *periodic* if there is some $h > 0$ such that $f(x + h) = f(x)$ for every x (e.g., see Fig. 22.4).

Musical notes yield one type of periodic function. When you pull on a string on a musical instrument, it vibrates in a repeating pattern. Hence, if we plot the speed of the string (and so also the speed of the

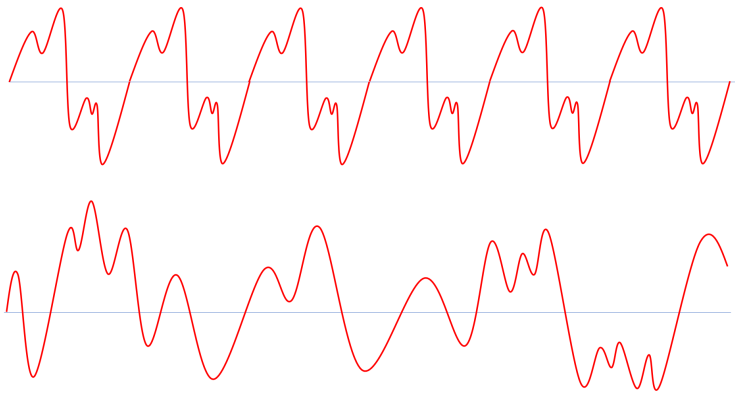

Figure 22.4: Top: A periodic function. Bottom: An a-periodic function.

air around it) as a function of time, it will some periodic function. The length of the period is known as the *wave length* of the note. The *frequency* is the number of times the function repeats itself within a unit of time. For example, the "Middle C" note has a frequency of 261.63 Hertz, which means its period is $1/(261.63)$ seconds.

If we play a *chord* by playing several notes at once, we get a more complex periodic function obtained by combining the functions of the individual notes (see Fig. 22.5). The human ear contains many small hairs, each of which is sensitive to a narrow band of frequencies. Hence when we hear the sound corresponding to a chord, the hairs in our ears actually separate it out to the components corresponding to each frequency.

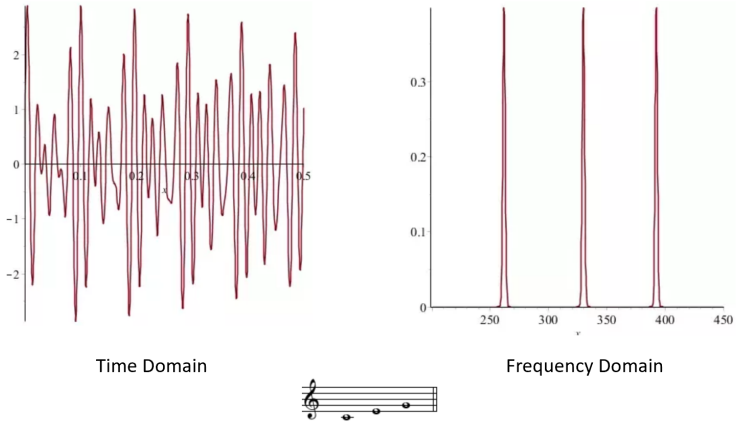

Figure 22.5: Left: The air-pressure when playing a "C Major" chord as a function of time. Right: The coefficients of the Fourier transform of the same function, we can see that it is the sum of three freuencies corresponding to the C, E and G notes (261.63, 329.63 and 392 Hertz respectively). Credit: Bjarke Mønsted's Quora answer.

It turns out that (essentially) *every* periodic function $f : \mathbb{R} \rightarrow \mathbb{R}$ can

be decomposed into a sum of simple *wave* functions (namely functions of the form $x \mapsto \sin(\theta x)$ or $x \mapsto \cos(\theta x)$). This is known as the Fourier Transform (see Fig. 22.6). The Fourier transform makes it easy to compute the period of a given function: it will simply be the least common multiple of the periods of the constituent waves.

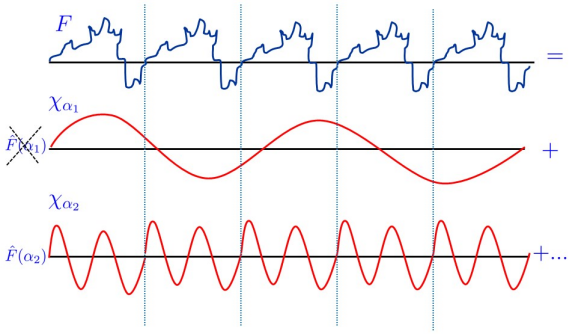

Figure 22.6: If f is a periodic function then when we represent it in the Fourier transform, we expect the coefficients corresponding to wavelengths that do not evenly divide the period to be very small, as they would tend to "cancel out".

22.10.2 Shor's Algorithm: A bird's eye view

On input a an integer M, Shor's algorithm outputs the prime factorization of M in time that is polynomial in $\log M$. The main steps in the algorithm are the following:

Step 1: Reduce to period finding. The first step in the algorithm is to pick a random $A \in \{0,1 \ldots, M-1\}$ and define the function $F_A : \{0,1\}^m \to \{0,1\}^m$ as $F_A(x) = A^x(\mod M)$ where we identify the string $x \in \{0,1\}^m$ with an integer using the binary representation, and similarly represent the integer $A^x(\mod M)$ as a string. (We will choose m to be some polynomial in m and so in particular $\{0,1\}^m$ is a large enough set to represent all the numbers in $\{0,1,\ldots,M-1\}$).

Some not-too-hard calculations (which we leave as **??**) show that: **(1)** The function F_A is *periodic* (i.e., there is some integer p_A such that $F_A(x+p_A) = F_A(x)$ for almost[17] every x) and more importantly **(2)** If we can recover the period p_A of F_A for several randomly chosen A's, then we can recover the factorization of M. Hence, factoring M reduces to finding out the period of the function F_A.

[17] We'll ignore this "almost" qualifier in the discussion below. It causes some annoying, yet ultimately manageable, technical issues in the full-fledged algorithm.

Step 2: Period finding via the Quantum Fourier Transform. Using a simple trick known as "repeated squaring", it is possible to compute the map $x \mapsto F_A(x)$ in time polynomial in m, which means we can also compute this map using a polynomial number of NAND gates,and so in particular we can generate in polynomial quantum

time a quantum state ρ that is (up to normalization) equal to

$$\sum_{x \in \{0,1\}^m} |x\rangle |F_A(x)\rangle \quad . \tag{22.13}$$

In particular, if we were to *measure* the state ρ, we would get a random pair of the form (x, y) where $y = F_A(x)$. So far, this is not at all impressive. After all, we did not need the power of quantum computing to generate such pairs: we could simply generate a random x and then compute $F_A(x)$.

Another way to describe the state ρ is that the coefficient of $|x\rangle|y\rangle$ in ρ is proportional to $f_{A,y}(x)$ where $f_{A,y} : \{0,1\}^m \to \mathbb{R}$ is the function such that

$$A_{A,y}(x) = \begin{cases} 1 & y = A^x(\mod M) \\ 0 & \text{otherwise} \end{cases} \quad . \tag{22.14}$$

The magic of Shor's algorithm comes from a procedure known as the *Quantum Fourier Transform*. It allows to change the state ρ into the state $\hat{\rho}$ where the coefficient of $|x\rangle|y\rangle$ is now proportional to the *x-th Fourier coefficient* of $f_{A,y}$. In other words, if we measure the state $\hat{\rho}$, we will obtain a pair (x, y) such that the probability of choosing x is proportional to the square of the weight of the *frequency* x in the representation of the function $f_{A,y}$. Since for every y, the function $f_{A,y}$ has the period p_A, it can be shown that the frequency x will be (almost[18]) a multiple of p_A. If we make several such samples $y_0, \ldots, y_k$ and obtain the frequencies $x_1, \ldots, x_k$, then the true period p_A divides all of them, and it can be shown that it is going to be in fact the *greatest common divisor* (g.c.d.) of all these frequencies: a value which can be computed in polynomial time.

As mentioned above, we can recover the factorization of M from the periods of $F_{A_0}, \ldots, F_{A_t}$ for some randomly chosen $A_0, \ldots, A_t$ in $\{0, \ldots, M-1\}$ and t which is polynomial in $\log M$.

[18] The "almost" qualifier again appears because the original function was only "almost" periodic, but it turns out this can be handled by using an "approximate greatest common divisor" algorithm instead of a standard g.c.d. below. The latter can be obtained using a tool known as the continued fraction representation of a number.

Quantum Fourier Transform Despite its name, the Quantum Fourier Transform does *not* actually give a way to compute the Fourier Transform of a function $f : \{0,1\}^m \to \mathbb{R}$. This would be impossible to do in time polynomial in m, as simply writing down the Fourier Transform would require 2^m coefficients. Rather the Quantum Fourier Transform gives a *quantum state* where the amplitude corresponding to an element (think: frequency) h is equal to the corresponding Fourier coefficient. This allows to sample from a distribution where h is drawn with probability proportional to the square of its Fourier coefficient. This is not the same as computing the Fourier transform, but is good enough for recovering the period.

22.11 QUANTUM FOURIER TRANSFORM (ADVANCED, OPTIONAL)

The above description of Shor's algorithm skipped over the implementation of the main quantum ingredient: the *Quantum Fourier Transform* algorithm. In this section we discuss the ideas behind this algorithm. We will be rather brief and imprecise. Remark 22.4 and Section 22.13 contain references to sources of more information about this topic.

To understand the Quantum Fourier Transform, we need to better understand the Fourier Transform itself. In particular, we will need to understand how it applies not just to functions whose input is a real number but to functions whose domain can be any arbitrary commutative *group*. Therefore we now take a short detour to (very basic) *group theory*, and define the notion of periodic functions over groups.

> (R) **Group theory** While we define the concepts we use, some background in group or number theory might be quite helpful for fully understanding this section. We will not use anything more than the basic properties of finite Abelian groups. Specifically we use the following notions:
>
> - A finite *group* $\mathbb{G}$ can be thought of as simply a set of elements and some *binary operation* $\star$ on these elements (i.e., if $g, h \in \mathbb{G}$ then $g \star h$ is an element of $\mathbb{G}$ as well).
> - The operation $\star$ satisfies the sort of properties that a product operation does, namely, it is *associative* (i.e., $(g \star h) \star f = g \star (h \star f)$) and there is some element 1 such that $g \star 1 = g$ for all g, and for every $g \in \mathbb{G}$ there exists an element g^{-1} such that $g \star g^{-1} = 1$.
> - A group is called *commutative* (also known as *Abelian*) if $g \star h = h \star g$ for all $g, h \in \mathbb{G}$.

The Fourier transform is a deep and vast topic, on which we will barely touch upon here. Over the real numbers, the Fourier transform of a function f is obtained by expressing f in the form $\sum \hat{f}(\alpha)\chi_\alpha$ where the χ_α's are "wave functions" (e.g. sines and cosines). However, it turns out that the same notion exists for *every* Abelian group $\mathbb{G}$. Specifically, for every such group $\mathbb{G}$, if f is a function mapping $\mathbb{G}$ to $\mathbb{C}$, then we can write f as

$$f = \sum_{g \in \mathbb{G}} \hat{f}(g)\chi_g \quad , \tag{22.15}$$

where the χ_g's are functions mapping $\mathbb{G}$ to $\mathbb{C}$ that are analogs of

the "wave functions" for the group $\mathbb{G}$ and for every $g \in \mathbb{G}$, $\hat{f}(g)$ is a complex number known as the *Fourier coefficient of f corresponding to g*.[19] The representation Eq. (22.15) is known as the *Fourier expansion* or *Fourier transform* of f, the numbers $(\hat{f}(g))_{g\in\mathbb{G}}$ are known as the *Fourier coefficients* of f and the functions $(\chi_g)_{g\in\mathbb{G}}$ are known as the *Fourier characters*. The central property of the Fourier characters is that they are *homomorphisms* of the group into the complex numbers, in the sense that for every $x, x' \in \mathbb{G}$, $\chi_g(x \star x') = \chi_g(x)\chi_g(x')$, where $\star$ is the group operation. One corollary of this property is that if $\chi_g(h) = 1$ then χ_g is *h periodic* in the sense that $\chi_g(x \star h) = \chi_g(x)$ for every x. It turns out that if f is periodic with minimal period h, then the only Fourier characters that have non zero coefficients in the expression Eq. (22.15) are those that are h periodic as well. This can be used to recover the period of f from its Fourier expansion.

[19] The equation Eq. (22.15) means that if we think of f as a $|\mathbb{G}|$ dimensional vector over the complex numbers, then we can write this vector as a sum (with certain coefficients) of the vectors $\{\chi_g\}_{g\in\mathbb{G}}$.

22.11.1 Quantum Fourier Transform over the Boolean Cube: Simon's Algorithm

We now describe the simplest setting of the Quantum Fourier Transform: the group $\{0,1\}^n$ with the XOR operation, which we'll denote by $(\{0,1\}^n, \oplus)$. It can be shown that the Fourier transform over $(\{0,1\}^n, \oplus)$ corresponds to expressing $f : \{0,1\}^n \to \mathbb{C}$ as

$$f = \sum_{y\in\{0,1\}} \hat{f}(y)\chi_y \tag{22.16}$$

where $\chi_y : \{0,1\}^n \to \mathbb{C}$ is defined as $\chi_y(x) = (-1)^{\sum_i y_i x_i}$ and $\hat{f}(y) = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} f(x)(-1)^{\sum_i y_i x_i}$.

The Quantum Fourier Transform over $(\{0,1\}^n, \oplus)$ is actually qutie simple:

Theorem 22.7 — QFT Over the Boolean Cube. Let $\rho = \sum_{x\in\{0,1\}^n} f(x)|x\rangle$ be a quantum state where $f : \{0,1\}^n \to \mathbb{C}$ is some function satisfying $\sum_{x\in\{0,1\}^n} |f(x)|^2 = 1$. Then we can use n gates to transform ρ to the state

$$\sum_{y\in\{0,1\}^n} \hat{f}(y)|y\rangle \tag{22.17}$$

where $f = \sum_y \hat{f}(y)\chi_y$ and $\chi_y : \{0,1\}^n \to \mathbb{C}$ is the function $\chi_y(x) = -1^{\sum x_i y_i}$.

Proof Idea: The idea behind the proof is that the *Hadamard* operation corresponds to the *Fourier transform* over the group $\{0,1\}^n$ (with the XOR operations). To show this, we just need to do the calculations. ⋆

Proof of Theorem 22.7. We can express the Hadamard operation HAD as follows:

$$HAD|a\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle + (-1)^a|1\rangle) \; . \tag{22.18}$$

We are given the state

$$\rho = \sum_{x\in\{0,1\}^n} f(x)|x\rangle \; . \tag{22.19}$$

Now suppose that we apply the HAD operation to each of the n qubits. We can see that we get the state

$$2^{-n/2} \sum_{x\in\{0,1\}^n} f(x) \prod_{i=0}^{n-1} (|0\rangle + (-1)^{x_i}|1\rangle) \; . \tag{22.20}$$

We can now use the distributive law and open up a term of the form

$$f(x)(|0\rangle + (-1)^{x_0}|1\rangle) \cdots (|0\rangle + (-1)^{x_{n-1}}|1\rangle) \tag{22.21}$$

to the following sum over 2^n terms:[20]

$$f(x) \sum_{y\in\{0,1\}^n} (-1)^{\sum y_i x_i}|y\rangle \; . \tag{22.22}$$

[20] If you find this confusing, try to work out why $(|0\rangle + (-1)^{x_0}|1\rangle)(|0\rangle + (-1)^{x_1}|1\rangle)(|0\rangle + (-1)^{x_2}|1\rangle)$ is the same as the sum over 2^3 terms $|000\rangle + (-1)^{x_2}|001\rangle + \cdots + (-1)^{x_0+x_1+x_2}|111\rangle$.

But by changing the order of summations, we see that the final state is

$$\sum_{y\in\{0,1\}^n} 2^{-n/2} \Big(\sum_{x\in\{0,1\}^n} f(x)(-1)^{\sum x_i y_i} \Big) |y\rangle \tag{22.23}$$

which exactly corresponds to $\hat{\rho}$. ■

R From Fourier to Period finding: Simon's Algorithm

Using Theorem 22.7 it is not hard to get an algorithm that can recover a string $h^* \in \{0,1\}^n$ given a circuit that computes a function $F : \{0,1\}^n \to \{0,1\}^*$ that is h^* *periodic* in the sense that $F(x) = F(x')$ for distinct x, x' if and only if $x' = x \oplus h^*$. The key observation is that if we compute the state $\sum_{x\in\{0,1\}^n} |x\rangle|F(x)\rangle$, and perform the Quantum Fourier transform on the first n qubits, then we would get a state such that the only basis elements with nonzero coefficients would be of the form $|y\rangle$ where

$$\sum y_i h_i^* = 0 (\mod 2) \tag{22.24}$$

So, by measuring the state, we can obtain a sample of a random y satisfying Eq. (22.24). But since Eq. (22.24) is a *linear* equation modulo 2 about the unknown n variables $h_0^*, \ldots, h_{n-1}^*$, if we repeat this

procedure to get n such equations, we will have at least as many equations as variables and (it can be shown that) this will suffice to recover h^*.

This result is known as Simon's Algorithm, and it preceded and inspired Shor's algorithm.

R **From Simon to Shor (advanced, optional)** Theorem 22.7 seemed to really use the special bit-wise structure of the group $\{0,1\}^n$, and so one could wonder if it can be extended to other groups. However, it turns out this can be the case. In particular, the key step in Shor's algorithm is to implement the Fourier transform for the group $\mathbb{Z}_L$ which is the set of numbers $\{0,\ldots,L-1\}$ with the operation being addition modulo L. In this case it turns out that the Fourier characters are the functions $\chi_y(x) = \omega^{yx}$ where $\omega = e^{2\pi i/L}$ (i here denotes the complex number $\sqrt{-1}$). The y-th Fourier coefficient of a function $f:\mathbb{Z}_L \to \mathbb{C}$ is

$$\hat{f}(y) = \frac{1}{\sqrt{L}} \sum_{x\in\mathbb{Z}_L} f(x)\omega^{xy} \,. \tag{22.25}$$

The key to implementing the Quantum Fourier Transform for such groups is to use the same recursive equations that enable the classical Fast Fourier Transform (FFT) algorithm. Specifically, consider the case that $L = 2^\ell$. We can separate the sum over x in Eq. (22.25) to the terms corresponding to even x's (of the form $x = 2z$) and odd x's (of the form $x = 2z+1$) too obtain

$$\hat{f}(y) = \frac{1}{\sqrt{L}} \sum_{z\in\mathbb{Z}_{L/2}} f(2z)(\omega^2)^{yz} + \frac{1}{\sqrt{L}} \sum_{z\in\mathbb{Z}_{L/2}} f(2z+1)(\omega^2)^{yz}\omega^y \tag{22.26}$$

which reduces computing the Fourier transform of f over the group $\mathbb{Z}_{2^\ell}$ to computing the Fourier transform of the functions f_{even} and f_{odd} (corresponding to the applying f to only the even and odd vectors respectively) which have $2^{\ell-1}$ inputs that we can identify with the group $\mathbb{Z}_{2^{\ell-1}}$.

This observation is usually used to obtain a fast (e.g. $O(L\log L)$) time to compute the Fourier transform in a classical setting, but it can be used to obtain a quantum circuit of $poly(\log L)$ gates to transform a state of the form $\sum_{x\in\mathbb{Z}_L} f(x)|x\rangle$ to a state of the form $\sum_{y\in\mathbb{Z}_L} \hat{f}(y)|y\rangle$.

Lecture Recap

- The state of an n-qubit quantum system can be modeled as a 2^n dimensional vector
- An operation on the state corresponds to applying a unitary matrix to this vector.
- Quantum circuits are obtained by composing basic operations such as HAD and U_{NAND}.
- We can use quantum circuits to define the classes $\mathbf{BQP}_{/\mathbf{poly}}$ and **BQP** which are the quantum analogs of $\mathbf{P}_{/\mathbf{poly}}$ and **BPP** respectively.
- There are some problems for which the best known quantum algorithm is *exponentially faster* than the best known, but quantum computing is not a panacea. In particular, as far as we know, quantum computers could still require exponential time to solve **NP**-complete problems such as SAT.

22.12 EXERCISES

Disclaimer Most of the exercises have been written in the summer of 2018 and haven't yet been fully debugged. While I would prefer people do not post online solutions to the exercises, I would greatly appreciate if you let me know of any bugs. You can do so by posting a GitHub issue about the exercise, and optionally complement this with an email to me with more details about the attempted solution.

Exercise 22.1 — Quantum and classical complexity class relations. Prove the following relations between quantum complexity classes and classical ones:

1. $\mathbf{P}_{/\mathbf{poly}} \subseteq \mathbf{BQP}_{/\mathbf{poly}}$.[21]
2. $\mathbf{P} \subseteq \mathbf{BQP}$.[22]
3. $\mathbf{BPP} \subseteq \mathbf{BQP}$.[23]
4. $\mathbf{BQP} \subseteq \mathbf{EXP}$.[24]
5. If $SAT \in \mathbf{BQP}$ then $\mathbf{NP} \subseteq \mathbf{BQP}$.[25]

[21] *Hint:* You can use U_{NAND} to simulate NAND gates.

[22] *Hint:* Use the alternative characterization of **P** as in Exercise 12.4.

[23] *Hint:* You can use the HAD gate to simulate a coin toss.

[24] *Hint:* In exponential time simulating quantum computation boils down to matrix multiplication.

[25] *Hint:* If a reduction can be implemented in **P** it can be implemented in **BQP** as well.

Exercise 22.2 — Discrete logarithm from order finding. Show a probabilistic polynomial time classical algorithm that given an Abelian finite

group $\mathbb{G}$ (in the form of an algorithm that computes the group operation), a *generator* g for the group, and an element $h \in \mathbb{G}$, as well access to a black box that on input $f \in \mathbb{G}$ outputs the *order* of f (the smallest a such that $f^a = 1$), computes the *discrete logarithm* of h with respect to g. That is the algorithm should output a number x such that $g^x = h$. See footnote for hint.[26] ■

[26] We are given $h = g^x$ and need to recover x. TO do so we can compute the order of various elements of the form $h^a g^b$. The order of such an element is a number c satisfying $c(xa + b) = 0 \pmod{|\mathbb{G}|}$. With a few random examples we will get a non trivial equation on x (where c is not zero modulo $|\mathbb{G}|$) and then we can use our knowledge of a, b, c to recover x.

22.13 BIBLIOGRAPHICAL NOTES

Chapters 9 and 10 in the book *Quantum Computing Since Democritus* give an informal but highly informative introduction to the topics of this lecture and much more. Shor's and Simon's algorithms are also covered in Chapter 10 of my book with Arora on computational complexity.

There are many excellent videos available online covering some of these materials. The Fourier transform is covered in this videos of Dr. Chris Geoscience, Clare Zhang and Vi Hart. More specifically to quantum computing, the videos of Umesh Vazirani on the Quantum Fourier Transform and Kelsey Houston-Edwards on Shor's Algorithm are very recommended.

Chapter 10 in Avi Wigderson's book gives a high level overview of quantum computing. Andrew Childs' lecture notes on quantum algorithms, as well as the lecture notes of Umesh Vazirani, John Preskill, and John Watrous

Regarding quantum mechanics in general, this video illustrates the double slit experiment, this Scientific American video is a nice exposition of Bell's Theorem. This talk and panel moderated by Brian Greene discusses some of the philosophical and technical issues around quantum mechanics and its so called "measurement problem". The Feynmann lecture on the Fourier Transform and quantum mechanics in general are very much worth reading.

The Fast Fourier Transform, used as a component in Shor's algorithm, is one of the most useful algorithms across many applications areas. The stories of its discovery by Gauss in trying to calculate asteroid orbits and rediscovery by Tukey during the cold war are fascinating as well.

22.14 FURTHER EXPLORATIONS

Some topics related to this chapter that might be accessible to advanced students include: (to be completed)

22.15 ACKNOWLEDGEMENTS

Thanks to Scott Aaronson for many helpful comments about this chapter.

VI
APPENDICES