Quantum Computing: Lecture Notes

Ronald de Wolf

Preface from 2011

These lecture notes were formed in small chunks during my "Quantum computing" course at the University of Amsterdam, Feb-May 2011, and compiled into one text thereafter. Each chapter was covered in a lecture of 2×45 minutes, with an additional 45-minute lecture for exercises and homework. The first half of the course (Chapters 1–7) covers quantum algorithms, the second half covers quantum complexity (Chapters 8–9), stuff involving Alice and Bob (Chapters 10–13), and error-correction (Chapter 14). A 15th lecture about physical implementations and general outlook was more sketchy, and I didn't write lecture notes for it.

These chapters may also be read as a general introduction to the area of quantum computation and information from the perspective of a theoretical computer scientist. While I made an effort to make the text self-contained and consistent, it may still be somewhat rough around the edges; I hope to continue polishing and adding to it. Comments & constructive criticism are very welcome, and can be sent to `rdewolf@cwi.nl`

Those who want to read more (much more...): see the book by Nielsen and Chuang [110].

Attribution, acknowledgments, subsequent updates

Most of the material in Chapters 1–6 [chapter numbers in this paragraph are for the 2011 version] comes from the first chapter of my PhD thesis [131], with a number of additions: the lower bound for Simon, the Fourier transform, the geometric explanation of Grover. Chapter 7 is newly written for these notes, inspired by Santha's survey [117]. Chapters 8 and 9 are largely new as well. Section 3 of Chapter 8, and most of Chapter 10 are taken (with many changes) from my "quantum proofs" survey paper with Andy Drucker [52]. Chapters 11 and 12 are partly taken from my non-locality survey with Harry Buhrman, Richard Cleve, and Serge Massar [33]. Chapters 13 and 14 are new. Thanks to Giannicola Scarpa (the teaching assistant for the first two editions of this course) for useful comments on some of the chapters.

Jan'13: Updated and corrected a few things for the Feb-Mar 2013 version of this course, and included exercises for each chapter. Thanks to Harry Buhrman, Florian Speelman, and Jeroen Zuiddam for spotting some typos in the earlier version.

April'13: More updates, clarifications and corrections; moved some material from Chapter 2 to 1; changed and added some exercises. Thanks to Jouke Witteveen for useful comments.

April'14: Fixed and clarified a few more things. Thanks to Maarten Wegewijs for spotting a typo in Chapter 4.

March'15: Updated a few small things.

July'15: Updated and corrected a few small things, added more exercises. Thanks to Srinivasan Arunachalam, Carla Groenland, and Koen Groenland for useful comments.

May'16: A few more corrections, thanks to Ralph Bottesch for useful comments.

Jan'18: Many more corrections, more exercises, a new Chapter 6 about HSP (the above-mentioned chapter numbers are for the earlier version of the notes), and moved the hints about exercises to an Appendix for students who want to try the exercises first without hints. Thanks to Joran

van Apeldoorn, Srinivasan Arunachalam, Rens Baardman, Alexander Belov, Koen de Boer, Daniel Chernowitz, András Gilyén, Ronald de Haan, Leon Ingelse, Stacey Jeffery, Rafael Kiesel, Jens Klooster, Sam Kuypers, Christian Nesenberend, and Christian Schaffner for useful comments.

Jan'19: More corrections and exercises, and new chapters about Hamiltonian simulation and the HHL algorithm. I marked by '(H)' the exercises having a hint in Appendix C, and removed citations from exercises to prevent students looking up the original papers when doing the exercises (which is neither necessary nor helpful). Those references are [21, 48, 54, 31, 53, 63, 36, 16, 14, 42, 9]. Thanks to Arjan Cornelissen, Sven Cornets de Groot, Gerrit Vos, and Harm de Vries for useful comments, and to András Gilyén for much help with Chapters 9 and 10.

Contents

Chapter 1

Quantum Computing

1.1 Introduction

Today's computers—both in theory (Turing machines) and practice (HPCs, PCs, laptops, tablets, smartphones,...)—are based on classical physics. They are limited by locality (operations have only local effects) and by the classical fact that systems can be in only one state at the time. However, modern quantum physics tells us that the world behaves quite differently. A quantum system can be in a *superposition* of many different states at the same time, and can exhibit *interference* effects during the course of its evolution. Moreover, spatially separated quantum systems may be *entangled* with each other and operations may have "non-local" effects because of this.

Quantum computation is the field that investigates the computational power and other properties of computers based on quantum-mechanical principles. An important objective is to find quantum algorithms that are significantly faster than any classical algorithm solving the same problem. The field started in the early 1980s with suggestions for analog quantum computers by Yuri Manin [101] (and appendix of [102]), Richard Feynman [59, 60], and Paul Benioff [19], and reached more digital ground when in 1985 David Deutsch defined the universal quantum Turing machine [49]. The following years saw only sparse activity, notably the development of the first algorithms by Deutsch and Jozsa [51] and by Simon [123], and the development of quantum complexity theory by Bernstein and Vazirani [23]. However, interest in the field increased tremendously after Peter Shor's very surprising discovery of efficient quantum algorithms for the problems of integer factorization and discrete logarithms in 1994 [122]. Since most of current classical cryptography is based on the assumption that these two problems are computationally hard, the ability to actually build and use a quantum computer would allow us to break most current classical cryptographic systems, notably the RSA system [114, 115]. (In contrast, a *quantum* form of cryptography due to Bennett and Brassard [22] is unbreakable even for quantum computers.)

Let us mention three different motivations for studying quantum computers, from practical to more philosophical:

1. The process of miniaturization that has made current classical computers so powerful and cheap, has already reached micro-levels where quantum effects occur. Chip-makers tend to go to great lengths to suppress those quantum effects, but instead one might also try to work with them, enabling further miniaturization.

2. Making use of quantum effects allows one to speed-up certain computations enormously

(sometimes exponentially), and even enables some things that are impossible for classical computers. The main purpose of this text is to explain these things (algorithms, crypto, etc.) in detail.

3. Finally, one might state the main goal of theoretical computer science as "study the power and limitations of the strongest-possible computational devices that nature allows us." Since our current understanding of nature is quantum mechanical, theoretical computer science should be studying the power of quantum computers, not classical ones.

Before limiting ourselves to theory, let us say a few words about practice: *to what extent will quantum computers ever be built*? At this point in time, it is just too early to tell. The first small 2-qubit quantum computer was built in 1997 and in 2001 a 5-qubit quantum computer was used to successfully factor the number 15 [128]. Since then, experimental progress on a number of different technologies has been steady but slow. Currently, the largest quantum computers have a few dozen qubits.

The practical problems facing physical realizations of quantum computers seem formidable. The problems of noise and decoherence have to some extent been solved in theory by the discovery of quantum error-correcting codes and fault-tolerant computing (see, e.g., Chapter 17 in these notes or [110, Chapter 10]), but these problems are by no means solved in practice. On the other hand, we should realize that the field of physical realization of quantum computing is still in its infancy and that classical computing had to face and solve many formidable technical problems as well—interestingly, often these problems were even of the same nature as those now faced by quantum computing (e.g., noise-reduction and error-correction). Moreover, while the difficulties facing the implementation of a full quantum computer may seem daunting, more limited applications involving quantum communication have already been implemented with some success, for example teleportation (which is the process of sending qubits using entanglement and *classical* communication), and quantum cryptography is nowadays even commercially available.

Even if the theory of quantum computing never materializes to a real large-scale physical computer, quantum-mechanical computers are still an extremely interesting idea which will bear fruit in other areas than practical fast computing. On the physics side, it may improve our understanding of quantum mechanics. The emerging theory of entanglement has already done this to some extent. On the computer science side, the theory of quantum computation generalizes and enriches classical complexity theory and may help resolve some of its problems.

1.2 Quantum mechanics

Here we give a brief and abstract introduction to quantum mechanics. In short: a quantum state is a *superposition* of classical states, to which we can apply either a *measurement* or a *unitary operation*. For the required linear algebra and Dirac notation we refer to Appendix A.

1.2.1 Superposition

Consider some physical system that can be in N different, mutually exclusive classical states. Call these states $|1\rangle, |2\rangle, \ldots, |N\rangle$. Roughly, by a "classical" state we mean a state in which the system can be found if we observe it. A *pure quantum state* (usually just called *state*) $|\phi\rangle$ is a *superposition* of classical states, written

$$|\phi\rangle = \alpha_1|1\rangle + \alpha_2|2\rangle + \cdots + \alpha_N|N\rangle.$$

Here α_i is a complex number that is called the *amplitude* of $|i\rangle$ in $|\phi\rangle$ (see Appendix B for a brief explanation of complex numbers). Intuitively, a system in quantum state $|\phi\rangle$ is in *all* classical states *at the same time*! It is in state $|1\rangle$ with amplitude α_1, in state $|2\rangle$ with amplitude α_2, and so on. Mathematically, the states $|1\rangle, \ldots, |N\rangle$ form an orthonormal basis of an N-dimensional *Hilbert space* (i.e., an N-dimensional vector space equipped with an inner product) of dimension N, and a quantum state $|\phi\rangle$ is a vector in this space, usually treated as a column vector.

There are two things we can do with a quantum state: measure it or let it evolve unitarily without measuring it. We will deal with measurement first.

1.2.2 Measurement

Measurement in the computational basis

Suppose we measure state $|\phi\rangle$. We cannot "see" a superposition itself, but only classical states. Accordingly, if we measure state $|\phi\rangle$ we will see one and only one classical state $|j\rangle$. Which specific $|j\rangle$ will we see? This is not determined in advance; the only thing we can say is that we will see state $|j\rangle$ with probability $|\alpha_j|^2$, which is the squared norm of the corresponding amplitude α_j ($|a+ib| = \sqrt{a^2+b^2}$). This is known as "Born's rule." Thus observing a quantum state induces a probability distribution on the classical states, given by the squared norms of the amplitudes. This implies $\sum_{j=1}^{N} |\alpha_j|^2 = 1$, so the vector of amplitudes has (Euclidean) norm 1. If we measure $|\phi\rangle$ and see classical state $|j\rangle$ as a result, then $|\phi\rangle$ itself has "disappeared," and all that is left is $|j\rangle$. In other words, observing $|\phi\rangle$ "collapses" the quantum superposition $|\phi\rangle$ to the classical state $|j\rangle$ that we saw, and all "information" that might have been contained in the amplitudes α_i is gone. Note that the probabilities of the various measurement outcomes are exactly the same when we measure $|\phi\rangle$ or when we measure state $e^{i\theta}|\phi\rangle$; because of this we sometimes say that the "gloabal phase" $e^{i\theta}$ has no physical significance.

Projective measurement

For most of the topics in this text, the above "measurement in the computational (or standard) basis" suffices. However, somewhat more general kinds of measurement than the above are possible and sometimes useful. This will be used only sparsely in the course, so it may be skipped on a first reading. A *projective* measurement is described by projectors $P_1, \ldots, P_m$ ($m \leq N$) which sum to identity. These projectors are then pairwise orthogonal, meaning that $P_i P_j = 0$ if $i \neq j$. The projector P_j projects on some subspace V_j of the total Hilbert space V, and every state $|\phi\rangle \in V$ can be decomposed in a unique way as $|\phi\rangle = \sum_{j=1}^{m} |\phi_j\rangle$, with $|\phi_j\rangle = P_j|\phi\rangle \in V_j$. Because the projectors are orthogonal, the subspaces V_j are orthogonal as well, as are the states $|\phi_j\rangle$. When we apply this measurement to the pure state $|\phi\rangle$, then we will get outcome j with probability $\||\phi_j\rangle\|^2 = \mathrm{Tr}(P_j|\phi\rangle\langle\phi|)$ and the state will then "collapse" to the new state $|\phi_j\rangle/\||\phi_j\rangle\| = P_j|\phi\rangle/\|P_j|\phi\rangle\|$.

For example, a measurement in the standard basis (a.k.a. the *computational* basis) is the specific projective measurement where $m = N$ and $P_j = |j\rangle\langle j|$. That is, P_j projects onto the computational basis state $|j\rangle$ and the corresponding subspace V_j is the one-dimensional space spanned by $|j\rangle$. Consider the state $|\phi\rangle = \sum_{j=1}^{N} \alpha_j |j\rangle$. Note that $P_j|\phi\rangle = \alpha_j|j\rangle$, so applying our measurement to $|\phi\rangle$ will give outcome j with probability $\|\alpha_j|j\rangle\|^2 = |\alpha_j|^2$, and in that case the state collapses to $\alpha_j|j\rangle/\|\alpha_j|j\rangle\| = \frac{\alpha_j}{|\alpha_j|}|j\rangle$. The norm-1 factor $\frac{\alpha_j}{|\alpha_j|}$ may be disregarded because it has no physical significance, so we end up with the state $|j\rangle$ as we saw before.

As another example, a measurement that distinguishes between $|j\rangle$ with $j \leq N/2$ and $|j\rangle$ with $j > N/2$ corresponds to the two projectors $P_1 = \sum_{j \leq N/2} |j\rangle\langle j|$ and $P_2 = \sum_{j > N/2} |j\rangle\langle j|$. Applying this measurement to the state $|\phi\rangle = \frac{1}{\sqrt{3}}|1\rangle + \sqrt{\frac{2}{3}}|N\rangle$ will give outcome 1 with probability $\|P_1|\phi\rangle\|^2 = 1/3$, in which case the state collapses to $|1\rangle$, and will give outcome 2 with probability $\|P_2|\phi\rangle\|^2 = 2/3$, in which case the state collapses to $|N\rangle$.

Observables

A projective measurement with projectors $P_1, \ldots, P_m$ and associated distinct outcomes $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$, can be written as one matrix $M = \sum_{i=1}^m \lambda_i P_i$, which is called an *observable*. This is a succinct way of writing down the projective measurement as one matrix, and has the added advantage that the *expected value* of the outcome can be easily calculated: if we are measuring a state $|\phi\rangle$, the probability of outcome λ_i is $\|P_i|\phi\rangle\|^2 = \mathrm{Tr}(P_i|\phi\rangle\langle\phi|)$, so the expected value of the outcome is $\sum_{i=1}^m \lambda_i \mathrm{Tr}(P_i|\phi\rangle\langle\phi|) = \mathrm{Tr}(M|\phi\rangle\langle\phi|)$ (where we used that trace is linear). Note that M is Hermitian: $M = M^*$. Conversely, since every Hermitian M has a spectral decomposition $M = \sum_{i=1}^m \lambda_i P_i$, there is a one-to-one correspondence between observables and Hermitian matrices. The Pauli matrices (Appendix A.7) are examples of 2-dimensional observables, with eigenvalues ± 1.

POVM measurement

If we only care about the final probability distribution on the m outcomes, not about the resulting post-measurement state, then the most general thing we can do is a so-called *positive-operator-valued measure* (POVM). This is specified by m positive semidefinite matrices $E_1, \ldots, E_m$ that sum to identity. When measuring a state $|\phi\rangle$, the probability of outcome i is given by $\mathrm{Tr}(E_i|\phi\rangle\langle\phi|)$. A projective measurement is the special case of a POVM where the measurement elements E_i are projectors.[1]

1.2.3 Unitary evolution

Instead of measuring $|\phi\rangle$, we can also apply some operation to it, i.e., change the state to some

$$|\psi\rangle = \beta_1|1\rangle + \beta_2|2\rangle + \cdots + \beta_N|N\rangle.$$

Quantum mechanics only allows *linear* operations to be applied to quantum states. What this means is: if we view a state like $|\phi\rangle$ as an N-dimensional vector $(\alpha_1, \ldots, \alpha_N)^T$, then applying an operation that changes $|\phi\rangle$ to $|\psi\rangle$ corresponds to multiplying $|\phi\rangle$ with an $N \times N$ complex-valued matrix U:

$$U \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_N \end{pmatrix}.$$

Note that by linearity we have $|\psi\rangle = U|\phi\rangle = U\left(\sum_i \alpha_i|i\rangle\right) = \sum_i \alpha_i U|i\rangle$.

Because measuring $|\psi\rangle$ should also give a probability distribution, we have the constraint $\sum_{j=1}^N |\beta_j|^2 = 1$. This implies that the operation U must preserve the norm of vectors, and hence

[1] Note that if E_i is a projector, then $\mathrm{Tr}(E_i|\phi\rangle\langle\phi|) = \mathrm{Tr}(E_i^2|\phi\rangle\langle\phi|) = \mathrm{Tr}(E_i|\phi\rangle\langle\phi|E_i) = \|E_i|\phi\rangle\|^2$, using the fact that $E_i = E_i^2$ and the cyclic property of the trace.

must be a *unitary* transformation. A matrix U is *unitary* if its inverse U^{-1} equals its conjugate transpose U^*. This is equivalent to saying that U always maps a vector of norm 1 to a vector of norm 1. Because a unitary transformation always has an inverse, it follows that any (non-measuring) operation on quantum states must be reversible: by applying U^{-1} we can always "undo" the action of U, and nothing is lost in the process. On the other hand, a measurement is clearly non-reversible, because we cannot reconstruct $|\phi\rangle$ from the observed classical state $|j\rangle$.

1.3 Qubits and quantum memory

In classical computation, the unit of information is a *bit*, which can be 0 or 1. In *quantum* computation, this unit is a *quantum* bit (*qubit*), which is a superposition of 0 and 1. Consider a system with 2 basis states, call them $|0\rangle$ and $|1\rangle$. We identify these basis states with the vectors $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$, respectively. A single qubit can be in any superposition

$$\alpha_0|0\rangle + \alpha_1|1\rangle, \ |\alpha_0|^2 + |\alpha_1|^2 = 1.$$

Accordingly, a single qubit "lives" in the vector space $\mathbb{C}^2$. Similarly we can think of systems of more than 1 qubit, which "live" in the tensor product space of several qubit systems. For instance, a 2-qubit system has 4 basis states: $|0\rangle \otimes |0\rangle$, $|0\rangle \otimes |1\rangle$, $|1\rangle \otimes |0\rangle$, $|1\rangle \otimes |1\rangle$. Here for instance $|1\rangle \otimes |0\rangle$ means that the first qubit is in its basis state $|1\rangle$ and the second qubit is in its basis state $|0\rangle$. We will often abbreviate this to $|1\rangle|0\rangle$, $|1,0\rangle$, or even $|10\rangle$.

More generally, a register of n qubits has 2^n basis states, each of the form $|b_1\rangle \otimes |b_2\rangle \otimes \ldots \otimes |b_n\rangle$, with $b_i \in \{0,1\}$. We can abbreviate this to $|b_1 b_2 \ldots b_n\rangle$. We will often abbreviate $0 \ldots 0$ to 0^n. Since bitstrings of length n can be viewed as numbers between 0 and $2^n - 1$, we can also write the basis states as numbers $|0\rangle, |1\rangle, |2\rangle, \ldots, |2^n - 1\rangle$. A quantum register of n qubits can be in any superposition

$$\alpha_0|0\rangle + \alpha_1|1\rangle + \cdots + \alpha_{2^n-1}|2^n - 1\rangle, \ \sum_{j=0}^{2^n-1} |\alpha_j|^2 = 1.$$

If we measure this in the computational basis, we obtain the n-bit state state $|j\rangle$ with probability $|\alpha_j|^2$.

Measuring just the first qubit of a state would correspond to the projective measurement that has the two projectors $P_0 = |0\rangle\langle 0| \otimes I_{2^{n-1}}$ and $P_1 = |1\rangle\langle 1| \otimes I_{2^{n-1}}$. For example, applying this measurement to the state $\frac{1}{\sqrt{3}}|0\rangle|\phi\rangle + \sqrt{\frac{2}{3}}|1\rangle|\psi\rangle$ will give outcome 0 with probability 1/3 (the state then becomes $|0\rangle|\phi\rangle$) and outcome 1 with probability 2/3 (the state then becomes $|1\rangle|\psi\rangle$). Similarly, measuring the first n qubits of an $(n+m)$-qubit state in the computational basis corresponds to the projective measurement that has 2^n projectors $P_j = |j\rangle\langle j| \otimes I_{2^m}$ for $j \in \{0,1\}^n$.

An important property that deserves to be mentioned is *entanglement*, which refers to quantum correlations between different qubits. For instance, consider a 2-qubit register that is in the state

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle.$$

Such 2-qubit states are sometimes called *EPR-pairs* in honor of Einstein, Podolsky, and Rosen [56], who first examined such states and their seemingly paradoxical properties. Initially neither of the

two qubits has a classical value $|0\rangle$ or $|1\rangle$. However, if we measure the first qubit and observe, say, a $|0\rangle$, then the whole state collapses to $|00\rangle$. Thus observing the first qubit immediately fixes also the second, unobserved qubit to a classical value. Since the two qubits that make up the register may be far apart, this example illustrates some of the non-local effects that quantum systems can exhibit. In general, a bipartite state $|\phi\rangle$ is called *entangled* if it cannot be written as a tensor product $|\phi_A\rangle \otimes |\phi_B\rangle$ where $|\phi_A\rangle$ lives in the first space and $|\phi_B\rangle$ lives in the second.

At this point, a comparison with classical probability distributions may be helpful. Suppose we have two probability spaces, A and B, the first with 2^n possible outcomes, the second with 2^m possible outcomes. A probability distribution on the first space can be described by 2^n numbers (non-negative reals summing to 1; actually there are only $2^n - 1$ degrees of freedom here) and a distribution on the second by 2^m numbers. Accordingly, a *product* distribution on the joint space can be described by $2^n + 2^m$ numbers. However, an arbitrary (non-product) distribution on the joint space takes 2^{n+m} real numbers, since there are 2^{n+m} possible outcomes in total. Analogously, an n-qubit state $|\phi_A\rangle$ can be described by 2^n numbers (complex numbers whose squared moduli sum to 1), an m-qubit state $|\phi_B\rangle$ by 2^m numbers, and their tensor product $|\phi_A\rangle \otimes |\phi_B\rangle$ by $2^n + 2^m$ numbers. However, an arbitrary (possibly entangled) state in the joint space takes 2^{n+m} numbers, since it lives in a 2^{n+m}-dimensional space. We see that the number of parameters required to describe quantum states is the same as the number of parameters needed to describe probability distributions. Also note the analogy between statistical independence[2] of two random variables A and B and non-entanglement of the product state $|\phi_A\rangle \otimes |\phi_B\rangle$. However, despite the similarities between probabilities and amplitudes, quantum states are much more powerful than distributions, because amplitudes may have negative (or even complex) parts which can lead to *interference* effects. Amplitudes only become probabilities when we square them. The art of quantum computing is to use these special properties for interesting computational purposes.

1.4 Elementary gates

A unitary that acts on a small number of qubits (say, at most 3) is often called a *gate*, in analogy to classical logic gates like AND, OR, and NOT; more about that in the next chapter. Two simple but important 1-qubit gates are the bitflip gate X (which negates the bit, i.e., swaps $|0\rangle$ and $|1\rangle$) and the phaseflip gate Z (which puts a $-$ in front of $|1\rangle$). Represented as 2×2 unitary matrices, these are

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

Another important 1-qubit gate is the *phase gate* R_ϕ, which merely rotates the phase of the $|1\rangle$-state by an angle ϕ:

$$R_\phi|0\rangle = |0\rangle$$
$$R_\phi|1\rangle = e^{i\phi}|1\rangle$$

This corresponds to the unitary matrix

$$R_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}.$$

[2]Two random variables A and B are *independent* if their joint probability distribution can be written as a product of individual distributions for A and for B: $\Pr[A = a \wedge B = b] = \Pr[A = a] \cdot \Pr[B = b]$ for all possible values a, b.

Note that Z is a special case of this: $Z = R_\pi$, because $e^{i\pi} = -1$. The $R_{\pi/4}$-gate is often just called the T-gate.

Possibly the most important 1-qubit gate is the *Hadamard* transform, specified by:

$$H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$
$$H|1\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

As a unitary matrix, this is represented as

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

If we apply H to initial state $|0\rangle$ and then measure, we have equal probability of observing $|0\rangle$ or $|1\rangle$. Similarly, applying H to $|1\rangle$ and observing gives equal probability of $|0\rangle$ or $|1\rangle$. However, if we apply H to the superposition $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ then we obtain $|0\rangle$: the positive and negative amplitudes for $|1\rangle$ cancel out! This effect is called *interference*, and is analogous to interference patterns between light or sound waves.

An example of a *two*-qubit gate is the *controlled-not* gate CNOT. It negates the second bit of its input if the first bit is 1, and does nothing if the first bit is 0:

$$\text{CNOT}|0\rangle|b\rangle = |0\rangle|b\rangle$$
$$\text{CNOT}|1\rangle|b\rangle = |1\rangle|1-b\rangle$$

The first qubit is called the *control* qubit, the second the *target* qubit. In matrix form, this is

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

More generally, if U is some single-qubit gate (i.e., 2×2 unitary matrix), then the two-qubit controlled-U gate corresponds to the following 4×4 unitary matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{11} & U_{12} \\ 0 & 0 & U_{21} & U_{22} \end{pmatrix}.$$

1.5 Example: quantum teleportation

In the next chapter we will look in more detail at how we can use and combine such elementary gates, but as an example we will here already explain *teleportation* [20]. Suppose there are two parties, Alice and Bob. Alice has a qubit $\alpha_0|0\rangle + \alpha_1|1\rangle$ that she wants to send to Bob via a *classical* channel. Without further resources this would be impossible, but Alice also shares an EPR-pair

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

with Bob (say Alice holds the first qubit and Bob the second). Initially, their joint state is

$$(\alpha_0|0\rangle + \alpha_1|1\rangle) \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

The first two qubits belong to Alice, the third to Bob. Alice performs a CNOT on her two qubits and then a Hadamard transform on her first qubit. Their joint state can now be written as

$$\begin{array}{l} \frac{1}{2}\ |00\rangle(\alpha_0|0\rangle + \alpha_1|1\rangle) + \\ \frac{1}{2}\ |01\rangle(\alpha_0|1\rangle + \alpha_1|0\rangle) + \\ \frac{1}{2}\ |10\rangle(\alpha_0|0\rangle - \alpha_1|1\rangle) + \\ \frac{1}{2}\ \underbrace{|11\rangle}_{Alice}\ \underbrace{(\alpha_0|1\rangle - \alpha_1|0\rangle)}_{Bob}. \end{array}$$

Alice then measures her two qubits in the computational basis and sends the result (2 random classical bits ab) to Bob over a classical channel. Bob now knows which transformation he must do on his qubit in order to regain the qubit $\alpha_0|0\rangle + \alpha_1|1\rangle$. First, if $b = 1$ then he applies a bitflip (X-gate) on his qubit; second if $a = 1$ then he applies a phaseflip (Z-gate). For instance, if Alice sent $ab = 11$, then Bob knows that his qubit is $\alpha_0|1\rangle - \alpha_1|0\rangle$. A bitflip followed by a phaseflip will give him Alice's original qubit $\alpha_0|0\rangle + \alpha_1|1\rangle$. In fact, if Alice's qubit had been *entangled* with some other qubits, then teleportation preserves this entanglement: Bob then receives a qubit that is entangled in the same way as Alice's original qubit was.

Note that the qubit on Alice's side has been destroyed: teleporting moves a qubit from A to B, rather than copying it. In fact, *copying an unknown qubit is impossible* [132], see Exercise 1.

Exercises

1. (H) Prove the *quantum no-cloning theorem*: there does not exist a 2-qubit unitary U that maps
$$|\phi\rangle|0\rangle \mapsto |\phi\rangle|\phi\rangle$$
for every qubit $|\phi\rangle$.

2. Show that unitaries cannot "delete" information: there is no one-qubit unitary U that maps $|\phi\rangle \mapsto |0\rangle$ for every one-qubit state $|\phi\rangle$.

3. Compute the result of applying a Hadamard transform to both qubits of $|0\rangle \otimes |1\rangle$ in two ways (the first way using tensor product of vectors, the second using tensor product of matrices), and show that the two results are equal:
$$H|0\rangle \otimes H|1\rangle = (H \otimes H)(|0\rangle \otimes |1\rangle).$$

4. Show that a bitflip operation, preceded and followed by Hadamard transforms, equals a phaseflip operation: $HXH = Z$.

5. Show that surrounding a CNOT gate with Hadamard gates switches the role of the control-bit and target-bit of the CNOT: $(H \otimes H)\mathrm{CNOT}(H \otimes H)$ is the 2-qubit gate where the second bit controls whether the first bit is negated (i.e., flipped).

6. Prove that an EPR-pair $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ is an *entangled* state, i.e., that it cannot be written as the tensor product of two separate qubits.

7. A matrix A is *inner product-preserving* if the inner product $\langle Av|Aw\rangle$ between Av and Aw equals the inner product $\langle v|w\rangle$, for all vectors v, w. A is *norm-preserving* if $\|Av\| = \|v\|$ for all vectors v, i.e., A preserves the Euclidean length of the vector. A is *unitary* if $A^*A = AA^* = I$.

   In the following, you may assume for simplicity that the entries of the vectors and matrices are real, not complex.

   (a) Prove that A is norm-preserving if, and only if, A is inner product-preserving.

   (b) Prove that A is inner product-preserving iff $A^*A = AA^* = I$.

   (c) Conclude that A is norm-preserving iff A is unitary.

   Bonus: prove the same for *complex* instead of real vector spaces.

8. Suppose Alice and Bob are not entangled. If Alice sends a qubit to Bob, then this can give Bob at most one bit of information about Alice.[3] However, if they share an EPR-pair, they can transmit *two* classical bits by sending one qubit over the channel; this is called *superdense coding*. This exercise will show how this works.

   (a) They start with a shared EPR-pair, $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Alice has classical bits a and b. Suppose she does an X-gate on her half of the EPR-pair if $a = 1$, followed by a Z-gate if $b = 1$ (she does both if $ab = 11$, and neither if $ab = 00$). Write the resulting 2-qubit state for the four different cases that ab could take.

   (b) Suppose Alice sends her half of the state to Bob, who now has two qubits. Show that Bob can determine both a and b from his state. Write Bob's operation as a quantum circuit with Hadamard and CNOT gates, followed by a measurement in the computational basis.

9. Let $\theta \in [0, 2\pi)$, $U_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$, $|\phi\rangle = U_\theta|0\rangle$ and $|\phi^\perp\rangle = U_\theta|1\rangle$.

   (a) Show that $ZX|\phi^\perp\rangle = |\phi\rangle$.

   (b) Show that an EPR-pair, $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, can also be written as $\frac{1}{\sqrt{2}}(|\phi\rangle|\phi\rangle + |\phi^\perp\rangle|\phi^\perp\rangle)$.

   (c) Suppose Alice and Bob start with an EPR-pair. Alice applies U_θ^{-1} to her qubit and then measures it in the computational basis. What pure state does Bob have if her outcome was 0, and what pure state does he have if her outcome was 1?

   (d) Suppose Alice knows the number θ but Bob does not. Give a protocol that uses one EPR-pair and 1 classical bit of communication where Bob ends up with the qubit $|\phi\rangle$ (in contrast to general teleportation of an unknown qubit, which uses 1 EPR-pair and *2* bits of communication).

[3] This is actually a deep statement, a special case of *Holevo's theorem*. More about this may be found in Chapter 13.

Chapter 2

The Circuit Model and Deutsch-Jozsa

2.1 Quantum computation

Below we explain how a quantum computer can apply computational steps to its register of qubits. Two models exist for this: the quantum Turing machine [49, 23] and the quantum circuit model [50, 134]. These models are equivalent, in the sense that they can simulate each other in polynomial time, assuming the circuits are appropriately "uniform." We only explain the circuit model here, which is more popular among researchers.

2.1.1 Classical circuits

In classical complexity theory, a *Boolean circuit* is a finite directed acyclic graph with AND, OR, and NOT gates. It has n input nodes, which contain the n input bits ($n \geq 0$). The internal nodes are AND, OR, and NOT gates, and there are one or more designated output nodes. The initial input bits are fed into AND, OR, and NOT gates according to the circuit, and eventually the output nodes assume some value. We say that a circuit *computes* some Boolean function $f : \{0,1\}^n \to \{0,1\}^m$ if the output nodes get the right value $f(x)$ for every input $x \in \{0,1\}^n$.

A *circuit family* is a set $\mathcal{C} = \{C_n\}$ of circuits, one for each input size n. Each circuit has one output bit. Such a family *recognizes* or *decides* a *language* $L \subseteq \{0,1\}^* = \cup_{n \geq 0} \{0,1\}^n$ if, for every n and every input $x \in \{0,1\}^n$, the circuit C_n outputs 1 if $x \in L$ and outputs 0 otherwise. Such a circuit family is *uniformly polynomial* if there is a deterministic Turing machine that outputs C_n given n as input, using space logarithmic in n.[1] Note that the size (number of gates) of the circuits C_n can then grow at most polynomially with n. It is known that uniformly polynomial circuit families are equal in power to polynomial-time deterministic Turing machines: a language L can be decided by a uniformly polynomial circuit family iff $L \in \mathbf{P}$ [111, Theorem 11.5], where $\mathbf{P}$ is the class of languages decidable by polynomial-time Turing machines.

Similarly we can consider *randomized* circuits. These receive, in addition to the n input bits, also some random bits ("coin flips") as input. A randomized circuit computes a function f if it successfully outputs the right answer $f(x)$ with probability at least $2/3$ for every x (probability taken over the values of the random bits; the $2/3$ may be replaced by any $1/2 + \varepsilon$). Randomized circuits are equal in power to randomized Turing machines: a language L can be decided by a uniformly

[1] Logarithmic space implies time that's at most polynomial in n, because such a machine will have only $\text{poly}(n)$ different internal states, so it either halts after $\text{poly}(n)$ steps or cycles forever.

polynomial randomized circuit family iff $L \in \mathbf{BPP}$, where **BPP** ("Bounded-error Probabilistic Polynomial time") is the class of languages that can efficiently be recognized by randomized Turing machines with success probability at least 2/3.

2.1.2 Quantum circuits

A *quantum circuit* (also called quantum network or quantum gate array) generalizes the idea of classical circuit families, replacing the AND, OR, and NOT gates by elementary *quantum gates*. A quantum gate is a unitary transformation on a small (usually 1, 2, or 3) number of qubits. We saw a number of examples already in the previous chapter: the bitflip gate X, the phaseflip gate Z, the Hadamard gate H. The main 2-qubit gate we have seen is the controlled-NOT (CNOT) gate. Adding another control register, we get the 3-qubit *Toffoli* gate, also called controlled-controlled-not (CCNOT) gate. This negates the third bit of its input if *both* of the first two bits are 1. The Toffoli gate is important because it is complete for classical reversible computation: any classical computation can be implemented by a circuit of Toffoli gates. This is easy to see: using auxiliary wires with fixed values, Toffoli can implement AND (fix the 3rd ingoing wire to 0) and NOT (fix the 1st and 2nd ingoing wire to 1). It is known that AND and NOT-gates together suffice to implement any classical Boolean circuit, so if we can apply (or simulate) Toffoli gates, we can implement any *classical* computation in a reversible manner.

Mathematically, such elementary quantum gates can be composed into bigger unitary operations by taking tensor products (if gates are applied *in parallel* to different parts of the register), and ordinary matrix products (if gates are applied *sequentially*). We have already seen a simple example of such a circuit of elementary gates in the previous chapter, namely to implement teleportation.

For example, if we apply the Hadamard gate H to each bit in a register of n zeroes, we obtain $\frac{1}{\sqrt{2^n}} \sum_{j\in\{0,1\}^n} |j\rangle$, which is a superposition of all n-bit strings. More generally, if we apply $H^{\otimes n}$ to an initial state $|i\rangle$, with $i \in \{0,1\}^n$, we obtain

$$H^{\otimes n}|i\rangle = \frac{1}{\sqrt{2^n}} \sum_{j\in\{0,1\}^n} (-1)^{i\cdot j}|j\rangle, \tag{2.1}$$

where $i \cdot j = \sum_{k=1}^{n} i_k j_k$ denotes the inner product of the n-bit strings $i, j \in \{0,1\}^n$. For example:

$$H^{\otimes 2}|01\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{2} \sum_{j\in\{0,1\}^2} (-1)^{01\cdot j}|j\rangle.$$

Note that Hadamard happens to be its own inverse (it's unitary and Hermitian, hence $H = H^* = H^{-1}$), so applying it once more on the right-hand side of the above equation would give us back $|01\rangle$. The n-fold Hadamard transform will be very useful for the quantum algorithms explained later.

As in the classical case, a quantum circuit is a finite directed acyclic graph of input nodes, gates, and output nodes. There are n nodes that contain the input (as classical bits); in addition we may have some more input nodes that are initially $|0\rangle$ ("workspace"). The internal nodes of the quantum circuit are quantum gates that each operate on at most two or three qubits of the state. The gates in the circuit transform the initial state vector into a final state, which will generally be a superposition. We measure some dedicated output bits of this final state to (probabilistically) obtain an answer.

To draw such circuits, we typically let time progress from left to right: we start with the initial state on the left. Each qubit is pictured as a wire, and the circuit prescribes which gates are to be applied to which wires. Single-qubit gates like X and H just act on one wire, while multi-qubit gates such as the CNOT act on multiple wires simultaneously.[2] When one qubit "controls" the application of a gate to another qubit, then the controlling wire is drawn with a dot linked vertically to the gate that is applied to the target qubit. This happens for instance with the CNOT, where the applied single-qubit gate is X (sometimes drawn as '$\oplus$'). Figure 2.1 gives a simple example on two qubits, initially in basis state $|00\rangle$: first apply H to the 1st qubit, then CNOT to both qubits (with the first qubit acting as the control), and then Z to the last qubit. The resulting state is $\frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$.

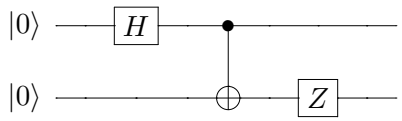

Figure 2.1: Simple circuit for turning $|00\rangle$ into an entangled state

In analogy to the classical class **BPP**, we will define **BQP** ("Bounded-error Quantum Polynomial time") as the class of languages that can efficiently be computed with success probability at least 2/3 by (a family of) quantum circuits whose size grows at most polynomially with the input length. We will study this quantum complexity class and its relation with various classical complexity classes in more detail in Chapter 12.

2.2 Universality of various sets of elementary gates

Which set of elementary gates should we allow? There are several reasonable choices.

> (1) The set of all 1-qubit operations together with the 2-qubit CNOT gate is universal, meaning that any other unitary transformation can be built from these gates.

Allowing all 1-qubit gates is not very realistic from an implementational point of view, as there are continuously many of them, and we cannot expect experimentalists to implement gates to infinite precision. However, the model is usually restricted, only allowing a small finite set of 1-qubit gates from which all other 1-qubit gates can be efficiently approximated.

> (2) The set consisting of CNOT, Hadamard, and the phase-gate $T = R_{\pi/4}$ is universal in the sense of approximation, meaning that any other unitary can be arbitrarily well approximated using circuits of only these gates. The *Solovay-Kitaev theorem* [110, Appendix 3] says that this approximation is quite efficient: we can approximate any gate on 1 or 2 qubits up to error ε using polylog$(1/\varepsilon)$ gates from our small set; in particular, simulating arbitrary gates up to exponentially small error costs only a polynomial overhead.

It is often convenient to restrict to real numbers and use an even smaller set of gates:

[2] Note that the number of wires (qubits) going into a unitary must equal the number of wires going out because a unitary is always invertible (reversible). This differs from the case of classical circuits, where non-reversible gates like AND have more wires going in than out.

(3) The set of Hadamard and Toffoli (CCNOT) is universal for all unitaries with real entries in the sense of approximation, meaning that any unitary with only real entries can be arbitrarily well approximated using circuits of only these gates.

2.3 Quantum parallelism

One uniquely quantum-mechanical effect that we can use for building quantum algorithms is *quantum parallelism*. Suppose we have a classical algorithm that computes some function $f : \{0,1\}^n \to \{0,1\}^m$. Then we can build a quantum circuit U (consisting only of Toffoli gates) that maps $|z\rangle|0\rangle \to |z\rangle|f(z)\rangle$ for every $z \in \{0,1\}^n$. Now suppose we apply U to a superposition of *all* inputs z (which is easy to build using n Hadamard transforms):

$$U\left(\frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}|z\rangle|0\rangle\right) = \frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}|z\rangle|f(z)\rangle.$$

We applied U just once, but the final superposition contains $f(z)$ for *all* 2^n input values z! However, by itself this is not very useful and does not give more than classical randomization, since observing the final superposition will give just one uniformly random $|z\rangle|f(z)\rangle$ and all other information will be lost. As we will see below, quantum parallelism needs to be combined with the effects of interference and entanglement in order to get something that is better than classical.

2.4 The early algorithms

The two main successes of quantum algorithms so far are Shor's factoring algorithm from 1994 [122] and Grover's search algorithm from 1996 [68], which will be explained in later chapters. In this section we describe some of the earlier quantum algorithms that preceded Shor's and Grover's.

Virtually all quantum algorithms work with *queries* in some form or other. We will explain this model here. It may look contrived at first, but eventually will lead smoothly to Shor's and Grover's algorithm. We should, however, emphasize that the query complexity model differs from the standard model described above, because the input is now given as a "black-box." This means that the exponential quantum-classical separations that we describe below (like Simon's) do not by themselves give exponential quantum-classical separations in the standard model.

To explain the query setting, consider an N-bit input $x = (x_0, \ldots, x_{N-1}) \in \{0,1\}^N$. Usually we will have $N = 2^n$, so that we can address bit x_i using an n-bit index i. One can think of the input as an N-bit memory which we can access at any point of our choice (a Random Access Memory). A memory access is via a so-called "black-box," which is equipped to output the bit x_i on input i. As a quantum operation, this would be the following unitary mapping on $n+1$ qubits:

$$O_x : |i,0\rangle \to |i,x_i\rangle.$$

The first n qubits of the state are called the *address bits* (or address register), while the $(n+1)$st qubit is called the *target bit*. Since this operation must be unitary, we also have to specify what happens if the initial value of the target bit is 1. Therefore we actually let O_x be the following unitary transformation:

$$O_x : |i,b\rangle \to |i,b \oplus x_i\rangle,$$

here $i \in \{0,1\}^n$, $b \in \{0,1\}$, and $\oplus$ denotes exclusive-or (addition modulo 2). In matrix representation, O_x is now a permutation matrix and hence unitary. Note that a quantum computer can apply O_x on a superposition of various i, something a classical computer cannot do. One application of this black-box is called a *query*, and counting the required number of queries to compute this or that function of x is something we will do a lot in the first half of these notes.

Given the ability to make a query of the above type, we can also make a query of the form $|i\rangle \mapsto (-1)^{x_i}|i\rangle$ by setting the target bit to the state $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = H|1\rangle$:

$$O_x\left(|i\rangle|-\rangle\right) = |i\rangle\frac{1}{\sqrt{2}}(|x_i\rangle - |1-x_i\rangle) = (-1)^{x_i}|i\rangle|-\rangle.$$

This $\pm$-kind of query puts the output variable in the phase of the state: if x_i is 1 then we get a -1 in the phase of basis state $|i\rangle$; if $x_i = 0$ then nothing happens to $|i\rangle$.[3] This "phase-oracle" is sometimes more convenient than the standard type of query. We sometimes denote the corresponding n-qubit unitary transformation by $O_{x,\pm}$.

2.4.1 Deutsch-Jozsa

Deutsch-Jozsa problem [51]:
For $N = 2^n$, we are given $x \in \{0,1\}^N$ such that either
(1) all x_i have the same value ("constant"), or
(2) $N/2$ of the x_i are 0 and $N/2$ are 1 ("balanced").
The goal is to find out whether x is constant or balanced.

The algorithm of Deutsch and Jozsa is as follows. We start in the n-qubit zero state $|0^n\rangle$, apply a Hadamard transform to each qubit, apply a query (in its $\pm$-form), apply another Hadamard to each qubit, and then measure the final state. As a unitary transformation, the algorithm would be $H^{\otimes n}O_{x,\pm}H^{\otimes n}$. We have drawn the corresponding quantum circuit in Figure 2.2 (where time again progresses from left to right). Note that the number of wires going into the query is n, not N; the basis states on this sequence of wires specify an n-bit address.

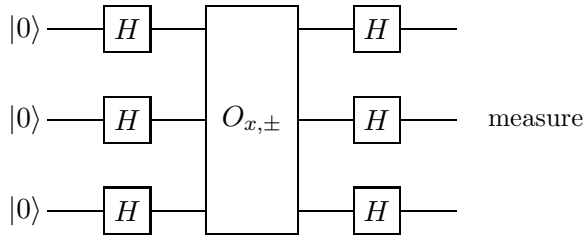

Figure 2.2: The Deutsch-Jozsa algorithm for $n = 3$

Let us follow the state through these operations. Initially we have the state $|0^n\rangle$. By Equation (2.1) on page 12, after the first Hadamard transforms we have obtained the uniform superposition of all i:

$$\frac{1}{\sqrt{2^n}}\sum_{i\in\{0,1\}^n}|i\rangle.$$

[3] Note that for $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, we have $O_x\left(|i\rangle|+\rangle\right) = |i\rangle|+\rangle$ irrespective of what x is.

The $O_{x,\pm}$-query turns this into

$$\frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} (-1)^{x_i} |i\rangle.$$

Applying the second batch of Hadamards gives (again by Equation (2.1)) the final superposition

$$\frac{1}{2^n} \sum_{i \in \{0,1\}^n} (-1)^{x_i} \sum_{j \in \{0,1\}^n} (-1)^{i \cdot j} |j\rangle,$$

where $i \cdot j = \sum_{k=1}^n i_k j_k$ as before. Since $i \cdot 0^n = 0$ for all $i \in \{0,1\}^n$, we see that the amplitude of the $|0^n\rangle$-state in the final superposition is

$$\frac{1}{2^n} \sum_{i \in \{0,1\}^n} (-1)^{x_i} = \begin{cases} 1 & \text{if } x_i = 0 \text{ for all } i, \\ -1 & \text{if } x_i = 1 \text{ for all } i, \\ 0 & \text{if } x \text{ is balanced.} \end{cases}$$

Hence the final observation will yield $|0^n\rangle$ if x is constant and will yield some other state if x is balanced. Accordingly, the Deutsch-Jozsa problem can be solved with certainty using only 1 quantum query and $O(n)$ other operations (the original solution of Deutsch and Jozsa used 2 queries, the 1-query solution is from [47]).

In contrast, it is easy to see that any *classical* deterministic algorithm needs at least $N/2 + 1$ queries: if it has made only $N/2$ queries and seen only 0s, the correct output is still undetermined. However, a classical algorithm can solve this problem efficiently if we allow a small error probability: just query x at two random positions, output "constant" if those bits are the same and "balanced" if they are different. This algorithm outputs the correct answer with probability 1 if x is constant and outputs the correct answer with probability $1/2$ if x is balanced. Thus the quantum-classical separation of this problem only holds if we consider algorithms without error probability.

2.4.2 Bernstein-Vazirani

Bernstein-Vazirani problem [23]:
For $N = 2^n$, we are given $x \in \{0,1\}^N$ with the property that there is some unknown $a \in \{0,1\}^n$ such that $x_i = (i \cdot a) \mod 2$. The goal is to find a.

The Bernstein-Vazirani algorithm is *exactly* the same as the Deutsch-Jozsa algorithm, but now the final observation miraculously yields a. Since $(-1)^{x_i} = (-1)^{(i \cdot a) \mod 2} = (-1)^{i \cdot a}$, we can write the state obtained after the query as:

$$\frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} (-1)^{x_i} |i\rangle = \frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} (-1)^{i \cdot a} |i\rangle.$$

Since Hadamard is its own inverse, applying a Hadamard to each qubit of the above state will turn it into the classical state $|a\rangle$ and hence solves the Bernstein-Vazirani problem with 1 query and $O(n)$ other operations. In contrast, any classical algorithm (even a randomized one with small error probability) needs to ask n queries for information-theoretic reasons: the final answer consists of n bits and one classical query gives at most 1 bit of information.

Bernstein and Vazirani also defined a recursive version of this problem, which can be solved exactly by a quantum algorithm in $\text{poly}(n)$ steps, but for which every classical randomized algorithm needs $n^{\Omega(\log n)}$ steps.

Exercises

1. Is the controlled-NOT operation C Hermitian? Determine C^{-1}.

2. Show that every unitary one-qubit gate with real entries can be written as a rotation matrix, possibly preceded and followed by Z-gates. In other words, show that for every 2×2 real unitary U, there exist signs $s_1, s_2, s_3 \in \{1, -1\}$ and angle $\theta \in [0, 2\pi)$ such that

$$U = s_1 \begin{pmatrix} 1 & 0 \\ 0 & s_2 \end{pmatrix} \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & s_3 \end{pmatrix}.$$

3. Construct a CNOT from two Hadamard gates and one controlled-Z (the controlled-Z gate maps $|11\rangle \mapsto -|11\rangle$ and acts like the identity on the other basis states).

4. A SWAP-gate interchanges two qubits: it maps basis state $|a, b\rangle$ to $|b, a\rangle$. Implement a SWAP-gate using a few CNOTs.

5. Let U be a 1-qubit unitary that we would like to implement in a controlled way, i.e., we want to implement a map $|c\rangle|b\rangle \mapsto |c\rangle U^c|b\rangle$ for all $c, b \in \{0, 1\}$. Suppose there exist 1-qubit unitaries A, B, and C, such that $ABC = I$ and $AXBXC = U$ (remember that X is the NOT-gate). Give a circuit that acts on two qubits and implements a controlled-U gate, using CNOTs and (uncontrolled) A, B, and C gates.

6. (H) It is possible to avoid doing any intermediate measurements in a quantum circuit, using one auxiliary qubit for each one-qubit measurement that needs to be delayed until the end of the computation. Show how.

7. (a) Give a circuit that maps $|0^n, b\rangle \mapsto |0^n, 1 - b\rangle$ for $b \in \{0, 1\}$, and that maps $|i, b\rangle \mapsto |i, b\rangle$ whenever $i \in \{0, 1\}^n \backslash \{0^n\}$. You are allowed to use every type of elementary gate mentioned in the lecture notes (incl. Toffoli gates), as well as auxiliary qubits that are initially $|0\rangle$ and that should be put back to $|0\rangle$ at the end of the computation.

   You can draw a Toffoli gate similar to a CNOT gate: a bold dot on each of the two control wires, and a '$\oplus$' on the target wire.

   (b) Suppose we can make queries of the type $|i, b\rangle \mapsto |i, b \oplus x_i\rangle$ to input $x \in \{0, 1\}^N$, with $N = 2^n$. Let x' be the input x with its first bit flipped (e.g., if $x = 0110$ then $x' = 1110$). Give a circuit that implements a query to x'. Your circuit may use one query to x.

   (c) Give a circuit that implements a query to an input x'' that is obtained from x (analogously to (b)) by setting its first bit to 0. Your circuit may use one query to x.

8. In Section 2.4 we showed that a query of the type $|i, b\rangle \mapsto |i, b \oplus x_i\rangle$ (where $i \in \{0, \ldots, N-1\}$ and $b \in \{0, 1\}$) can be used to implement a phase-query to x, i.e., one of the type $|i\rangle \mapsto (-1)^{x_i}|i\rangle$. Show that a query of the first type be implemented using controlled phase-queries to x, and possibly some auxiliary qubits and other gates. Can this also be done with *un*controlled phase-queries to x? If yes, show how. If no, explain why not.

9. Give a randomized classical algorithm (i.e., one that can flip coins during its operation) that makes only two queries to x, and decides the Deutsch-Jozsa problem with success probability at least 2/3 on every possible input. A high-level description is enough, no need to write out the classical circuit.

10. Suppose our N-bit input x satisfies the following promise:
either (1) the first $N/2$ bits of x are all 0 and the second $N/2$ bits are all 1; or (2) the number of 1s in the first half of x plus the number of 0s in the second half, equals $N/2$. Modify the Deutsch-Jozsa algorithm to efficiently distinguish these two cases (1) and (2).

11. (H) A *parity query* to input $x \in \{0,1\}^N$ corresponds to the $(N+1)$-qubit unitary map $Q_x : |y,b\rangle \mapsto |y, b \oplus (x \cdot y)\rangle$, where $x \cdot y = \sum_{i=0}^{N-1} x_i y_i \bmod 2$. For a fixed function $f : \{0,1\}^N \to \{0,1\}$, give a quantum algorithm that computes $f(x)$ using only one such query (i.e., one application of Q_x), and as many elementary gates as you want. You do not need to give the circuit in full detail, an informal description of the algorithm is good enough.

Chapter 3

Simon's Algorithm

The Deutsch-Jozsa problem showed an exponential quantum improvement over the best *deterministic* classical algorithms; the Bernstein-Vazirani problem shows a polynomial improvement over the best *randomized* classical algorithms that have error probability $\leq 1/3$. In this chapter we will combine these two features: we will see a problem where quantum computers are exponentially more efficient than bounded-error randomized algorithms.

3.1 The problem

Let $N = 2^n$, and identify the set $\{0, \ldots, N-1\}$ with $\{0,1\}^n$. Let $j \oplus s$ be the n-bit string obtained by bitwise adding the n-bit strings j and s mod 2.

Simon's problem [123]:
For $N = 2^n$, we are given $x = (x_0, \ldots, x_{N-1})$, with $x_i \in \{0,1\}^n$, with the property that there is some unknown nonzero $s \in \{0,1\}^n$ such that $x_i = x_j$ iff ($i = j$ or $i = j \oplus s$). The goal is to find s.

Note that x, viewed as a function from $\{0, \ldots, N-1\}$ to $\{0, \ldots, N-1\}$, is a 2-to-1 function, where the 2-to-1-ness is determined by the unknown *mask* s. The queries to the input here are slightly different from before: the input $x = (x_0, \ldots, x_{N-1})$ now has variables x_i that themselves are n-bit strings, and one query gives such a string completely ($|i, 0^n\rangle \mapsto |i, x_i\rangle$). However, we can also view this problem as having $n2^n$ binary variables that we can query individually. Since we can simulate one x_i-query using only n binary queries (just query all n bits of x_i), this alternative view will not affect the number of queries very much.

3.2 The quantum algorithm

Simon's algorithm starts out very similar to Deutsch-Jozsa: start in a state of $2n$ zero qubits $|0^n\rangle|0^n\rangle$ and apply Hadamard transforms to the first n qubits, giving

$$\frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} |i\rangle|0^n\rangle.$$

At this point, the second n-qubit register still holds only zeroes. A query turns this into

$$\frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} |i\rangle |x_i\rangle.$$

Now the algorithm measures the second n-bit register (see Exercise 1; this measurement is actually not necessary, but it facilitates analysis). The measurement outcome will be some value x_i and the first register will collapse to the superposition of the two indices having that x_i-value:

$$\frac{1}{\sqrt{2}}(|i\rangle + |i \oplus s\rangle)|x_i\rangle.$$

We will now ignore the second register and apply Hadamard transforms to the first n qubits. Using Equation (2.1) and the fact that $(i \oplus s) \cdot j = (i \cdot j) \oplus (s \cdot j)$, we can write the resulting state as

$$\frac{1}{\sqrt{2^{n+1}}} \left(\sum_{j \in \{0,1\}^n} (-1)^{i \cdot j} |j\rangle + \sum_{j \in \{0,1\}^n} (-1)^{(i \oplus s) \cdot j} |j\rangle \right) =$$

$$\frac{1}{\sqrt{2^{n+1}}} \left(\sum_{j \in \{0,1\}^n} (-1)^{i \cdot j} \left(1 + (-1)^{s \cdot j}\right) |j\rangle \right).$$

Note that $|j\rangle$ has nonzero amplitude iff $s \cdot j = 0 \mod 2$. Measuring the state gives a uniformly random element from the set $\{j \mid s \cdot j = 0 \mod 2\}$. Accordingly, we get a linear equation that gives information about s. We repeat this algorithm until we have obtained $n-1$ independent linear equations involving s. The solutions to these equations will be 0^n and the correct s, which we can compute efficiently by a classical algorithm (Gaussian elimination modulo 2). This can be done by means of a classical circuit of size roughly $O(n^3)$.

Note that if the j's you have generated at some point span a space of size 2^k, for some $k < n-1$, then the probability that your next run of the algorithm produces a j that is linearly independent of the earlier ones, is $(2^{n-1} - 2^k)/2^{n-1} \geq 1/2$. Hence an expected number of $O(n)$ runs of the algorithm suffices to find $n-1$ linearly independent j's. Simon's algorithm thus finds s using an expected number of $O(n)$ x_i-queries and polynomially many other operations.

3.3 Classical algorithms for Simon's problem

3.3.1 Upper bound

Let us first sketch a classical randomized algorithm that solves Simon's problem using $O(\sqrt{2^n})$ queries, based on the so-called "birthday paradox." Our algorithm will make T randomly chosen distinct queries $i_1, \ldots, i_T$, for some T to be determined later. If there is a *collision* among those queries (i.e., $x_{i_k} = x_{i_\ell}$ for some $k \neq \ell$), then we are done, because then we know $i_k = i_\ell \oplus s$, equivalently $s = i_k \oplus i_\ell$. How large should T be such that we are likely to see a collision in case $s \neq 0^n$? (there won't be any collisions if $s = 0^n$.) There are $\binom{T}{2} = \frac{1}{2}T(T-1) \approx T^2/2$ pairs in our sequence that could be a collision, and since the indices are chosen randomly, the probability for a fixed pair to form a collision is $1/(2^n - 1)$. Hence by linearity of expectation, the *expected* number of collisions in our sequence will be roughly $T^2/2^{n+1}$. If we choose $T = \sqrt{2^{n+1}}$, we expect to have roughly 1 collision in our sequence, which is good enough to find s. Of course, an *expected value* of 1 collision does not mean that we will have at least one collision *with high probability*, but a slightly more involved calculation shows the latter statement as well.

3.3.2 Lower bound

Simon [123] proved that any classical randomized algorithm that finds s with high probability needs to make $\Omega(\sqrt{2^n})$ queries, so the above classical algorithm is essentially optimal. This was the first proven exponential separation between quantum algorithms and classical bounded-error algorithms (let us stress again that this does not prove an exponential separation in the usual circuit model, because we are counting queries rather than ordinary operations here). Simon's algorithm inspired Shor to his factoring algorithm, which we describe in Chapter 5.

We will prove the classical lower bound for a decision version of Simon's problem:

Given: input $x = (x_0, \ldots, x_{N-1})$, where $N = 2^n$ and $x_i \in \{0,1\}^n$
Promise: $\exists s \in \{0,1\}^n$ such that: $x_i = x_j$ iff ($i = j$ or $i = j \oplus s$)
Task: decide whether $s = 0^n$

Consider the input distribution μ that is defined as follows. With probability $1/2$, x is a uniformly random permutation of $\{0,1\}^n$; this corresponds to the case $s = 0^n$. With probability $1/2$, we pick a nonzero string s at random, and for each pair $(i, i \oplus s)$, we pick a unique value for $x_i = x_{i \oplus s}$ at random. If there exists a *randomized* T-query algorithm that achieves success probability p under this input distribution μ, then there also is *deterministic* T-query algorithm that achieves success probability p under μ (because the behavior of the randomized algorithm is the average over a number of deterministic algorithms). Now consider a deterministic algorithm with error $\leq 1/3$ under μ, that makes T queries to x. We want to show that $T = \Omega(\sqrt{2^n})$.

First consider the case $s = 0^n$. We can assume the algorithm never queries the same point twice. Then the T outcomes of the queries are T distinct n-bit strings, and each sequence of T strings is equally likely.

Now consider the case $s \neq 0^n$. Suppose the algorithm queries the indices $i_1, \ldots, i_T$ (this sequence depends on x) and gets outputs $x_{i_1}, \ldots, x_{i_T}$. Call a sequence of queries $i_1, \ldots, i_T$ *good* if it shows a collision (i.e., $x_{i_k} = x_{i_\ell}$ for some $k \neq \ell$), and *bad* otherwise. If the sequence of queries of the algorithm is good, then we can find s, since $i_k \oplus i_\ell = s$. On the other hand, if the sequence is bad, then each sequence of T distinct outcomes is equally likely—just as in the $s = 0^n$ case! We will now show that the probability of the bad case is very close to 1 for small T.

If $i_1, \ldots, i_{k-1}$ is bad, then we have excluded at most $\binom{k-1}{2}$ possible values of s (namely all values $i_j \oplus i_{j'}$ for $j, j' \in [k-1]$), and all other values of s are equally likely. The probability that the next query i_k makes the sequence good, is the probability that $x_{i_k} = x_{i_j}$ for some $j < k$, equivalently, that the set $S = \{i_k \oplus i_j \mid j < k\}$ happens to contain the string s. But S has only $k-1$ members, while there are $2^n - 1 - \binom{k-1}{2}$ equally likely remaining possibilities for s. This means that the probability that the sequence is still bad after query i_k is made, is very close to 1. In formulas:

$$\begin{aligned}
\Pr[i_1, \ldots, i_T \text{ is bad}] &= \prod_{k=2}^{T} \Pr[i_1, \ldots, i_k \text{ is bad} \mid i_1, \ldots, i_{k-1} \text{ is bad}] \\
&= \prod_{k=2}^{T} \left(1 - \frac{k-1}{2^n - 1 - \binom{k-1}{2}}\right) \\
&\geq 1 - \sum_{k=2}^{T} \frac{k-1}{2^n - 1 - \binom{k-1}{2}}.
\end{aligned}$$

Here we used the fact that $(1-a)(1-b) \geq 1-(a+b)$ if $a, b \geq 0$. Note that $\sum_{k=2}^{T}(k-1) = T(T-1)/2 \approx T^2/2$, and $2^n - 1 - \binom{k-1}{2} \approx 2^n$ as long as $k \ll \sqrt{2^n}$. Hence we can approximate the last formula by $1 - T^2/2^{n+1}$. Accordingly, if $T \ll \sqrt{2^n}$ then with probability nearly 1 (probability taken over the distribution μ) the algorithm's sequence of queries is bad. If it gets a bad sequence, it cannot "see" the difference between the $s = 0^n$ case and the $s \neq 0^n$ case, since both cases result in a uniformly random sequence of T distinct n-bit strings as answers to the T queries. This shows that T has to be $\Omega(\sqrt{2^n})$ in order to enable the algorithm to get a good sequence of queries with high probability.

Exercises

1. Give the projectors of the 2-register projective measurement that's apply in Simons algorithm.

2. Analyze the different steps of Simon's algorithm if $s = 0^n$ (so all x_i-values are distinct), and show that the final output j is uniformly distributed over $\{0,1\}^n$.

3. Suppose we run Simon's algorithm on the following input x (with $N = 8$ and hence $n = 3$):

   $$x_{000} = x_{111} = 000$$
   $$x_{001} = x_{110} = 001$$
   $$x_{010} = x_{101} = 010$$
   $$x_{011} = x_{100} = 011$$

   Note that x is 2-to-1 and $x_i = x_{i \oplus 111}$ for all $i \in \{0,1\}^3$, so $s = 111$.

   (a) Give the starting state of Simon's algorithm.

   (b) Give the state after the first Hadamard transforms on the first 3 qubits.

   (c) Give the state after applying the oracle.

   (d) Give the state after measuring the second register (suppose the measurement gave $|001\rangle$).

   (e) Using $H^{\otimes n}|i\rangle = \frac{1}{\sqrt{2^n}} \sum_{j \in \{0,1\}^n} (-1)^{i \cdot j} |j\rangle$, give the state after the final Hadamards.

   (f) Why does a measurement of the first 3 qubits of the final state give information about s?

   (g) Suppose the first run of the algorithm gives $j = 011$ and a second run gives $j = 101$. Show that, assuming $s \neq 000$, those two runs of the algorithm already determine s.

4. Consider the following generalization of Simon's problem: the input is $x = (x_0, \ldots, x_{N-1})$, with $N = 2^n$ and $x_i \in \{0,1\}^n$, with the property that there is some unknown *subspace* $V \subseteq \{0,1\}^n$ such that $x_i = x_j$ iff there exists a $v \in V$ such that $i = j \oplus v$. The usual definition of Simon's problem corresponds to the case where the subspace V has dimension at most 1.

   Show that one run of Simon's algorithm now produces a $j \in \{0,1\}^n$ that is orthogonal to the whole subspace (i.e., $j \cdot v = 0 \bmod 2$ for every $v \in V$).

5. (a) Suppose x is an N-bit string. What happens if we apply a Hadamard transform to each qubit of the N-qubit state $\dfrac{1}{\sqrt{2^N}} \displaystyle\sum_{y \in \{0,1\}^N} (-1)^{x \cdot y} |y\rangle$?

(b) Give a quantum algorithm that uses T queries to N-bit string x, and that maps $|y\rangle \mapsto (-1)^{x \cdot y}|y\rangle$ for every $y \in \{0,1\}^N$ that contains at most T 1s (i.e., for every y of Hamming weight $\leq T$). You can argue on a high level, no need to write out circuits in detail.

(c) (H) Give a quantum algorithm that with high probability outputs x, using at most $N/2 + 2\sqrt{N}$ queries to x.

(d) Argue that a classical algorithm needs at least $N - 1$ queries in order to have success probability at least $1/2$ of outputting the correct x.

Chapter 4

The Fourier Transform

4.1 The classical discrete Fourier transform

The Fourier transform occurs in many different versions throughout classical computing, in areas ranging from signal-processing to data compression to complexity theory.

For our purposes, the Fourier transform is going to be an $N \times N$ unitary matrix, all of whose entries have the same magnitude. For $N = 2$, it's just our familiar Hadamard transform:

$$F_2 = H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Doing something similar in 3 dimensions is impossible with real numbers: we can't give three orthogonal vectors in $\{+1, -1\}^3$. However, using *complex* numbers allows us to define the Fourier transform for any N. Let $\omega_N = e^{2\pi i/N}$ be an N-th root of unity ("root of unity" means that $\omega_N^k = 1$ for some integer k, in this case $k = N$). The rows of the matrix will be indexed by $j \in \{0, \ldots, N-1\}$ and the columns by $k \in \{0, \ldots, N-1\}$. Define the (j,k)-entry of the matrix F_N by $\frac{1}{\sqrt{N}}\omega_N^{jk}$ (the exponent jk is the usual product of two integers):

$$F_N = \frac{1}{\sqrt{N}} \begin{pmatrix} & \vdots & \\ \cdots & \omega_N^{jk} & \cdots \\ & \vdots & \end{pmatrix}$$

Note that F_N is a unitary matrix, since each column has norm 1, and any two columns (say those indexed by k and k') are orthogonal:

$$\sum_{j=0}^{N-1} \frac{1}{\sqrt{N}}(\omega_N^{jk})^* \frac{1}{\sqrt{N}}\omega_N^{jk'} = \frac{1}{N}\sum_{j=0}^{N-1} \omega_N^{j(k'-k)} = \begin{cases} 1 & \text{if } k = k' \\ 0 & \text{otherwise} \end{cases}$$

Since F_N is unitary and symmetric, the inverse $F_N^{-1} = F_N^*$ only differs from F_N by having minus signs in the exponent of the entries. For a vector $v \in \mathbb{R}^N$, the vector $\widehat{v} = F_N v$ is called the Fourier transform of v.[1] Doing the matrix-vector multiplication, its entries are given by $\widehat{v}_j =$

[1] The literature on Fourier analysis usually talks about the Fourier transform of a *function* rather than of a vector, but on finite domains that's just a notational variant of what we do here: a vector $v \in \mathbb{R}^N$ can also be viewed as a function $v : \{0, \ldots, N-1\} \to \mathbb{R}$ defined by $v(i) = v_i$. Also, in the classical literature people sometimes use the term "Fourier transform" for what we call the inverse Fourier transform.

$\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} v_k$.

4.2 The Fast Fourier Transform

The naive way of computing the Fourier transform $\widehat{v} = F_N v$ of $v \in \mathbb{R}^N$ just does the matrix-vector multiplication to compute all the entries of $\widehat{v}$. This would take $O(N)$ steps (additions and multiplications) per entry, and $O(N^2)$ steps to compute the whole vector $\widehat{v}$. However, there is a more efficient way of computing $\widehat{v}$. This algorithm is called the *Fast Fourier Transform* (FFT, due to Cooley and Tukey in 1965), and takes only $O(N \log N)$ steps. This difference between the quadratic N^2 steps and the near-linear $N \log N$ is tremendously important in practice when N is large, and is the main reason that Fourier transforms are so widely used.

We will assume $N = 2^n$, which is usually fine because we can add zeroes to our vector to make its dimension a power of 2 (but similar FFTs can be given also directly for most N that aren't a power of 2). The key to the FFT is to rewrite the entries of $\widehat{v}$ as follows:

$$\begin{aligned}
\widehat{v}_j &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} v_k \\
&= \frac{1}{\sqrt{N}} \left(\sum_{\text{even } k} \omega_N^{jk} v_k + \omega_N^j \sum_{\text{odd } k} \omega_N^{j(k-1)} v_k \right) \\
&= \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{N/2}} \sum_{\text{even } k} \omega_{N/2}^{jk/2} v_k + \omega_N^j \frac{1}{\sqrt{N/2}} \sum_{\text{odd } k} \omega_{N/2}^{j(k-1)/2} v_k \right)
\end{aligned}$$

Note that we've rewritten the entries of the N-dimensional Fourier transform $\widehat{v}$ in terms of two $N/2$-dimensional Fourier transforms, one of the even-numbered entries of v, and one of the odd-numbered entries of v.

This suggest a recursive procedure for computing $\widehat{v}$: first separately compute the Fourier transform $\widehat{v_{\text{even}}}$ of the $N/2$-dimensional vector of even-numbered entries of v and the Fourier transform $\widehat{v_{\text{odd}}}$ of the $N/2$-dimensional vector of odd-numbered entries of v, and then compute the N entries

$$\widehat{v}_j = \frac{1}{\sqrt{2}} (\widehat{v_{\text{even}}}_j + \omega_N^j \widehat{v_{\text{odd}}}_j).$$

Strictly speaking this is not well-defined, because $\widehat{v_{\text{even}}}$ and $\widehat{v_{\text{odd}}}$ are just $N/2$-dimensional vectors. However, if we define $\widehat{v_{\text{even}}}_{j+N/2} = \widehat{v_{\text{even}}}_j$ (and similarly for $\widehat{v_{\text{odd}}}$) then it all works out.

The time $T(N)$ it takes to implement F_N this way can be written recursively as $T(N) = 2T(N/2) + O(N)$, because we need to compute two $N/2$-dimensional Fourier transforms and do $O(N)$ additional operations to compute $\widehat{v}$. This recursion works out to time $T(N) = O(N \log N)$, as promised. Similarly, we have an equally efficient algorithm for the *inverse* Fourier transform $F_N^{-1} = F_N^*$, whose entries are $\frac{1}{\sqrt{N}} \omega_N^{-jk}$.

4.3 Application: multiplying two polynomials

Suppose we are given two real-valued polynomials p and q, each of degree at most d:

$$p(x) = \sum_{j=0}^{d} a_j x^j \text{ and } q(x) = \sum_{k=0}^{d} b_k x^k$$

We would like to compute the product of these two polynomials, which is

$$(p \cdot q)(x) = \left(\sum_{j=0}^{d} a_j x^j\right)\left(\sum_{k=0}^{d} b_k x^k\right) = \sum_{\ell=0}^{2d} (\underbrace{\sum_{j=0}^{2d} a_j b_{\ell-j}}_{c_\ell}) x^\ell,$$

where implicitly we set $a_j = b_j = 0$ for $j > d$ and $b_{\ell-j} = 0$ if $j > \ell$. Clearly, each coefficient c_ℓ by itself takes $O(d)$ steps (additions and multiplications) to compute, which suggests an algorithm for computing the coefficients of $p \cdot q$ that takes $O(d^2)$ steps. However, using the fast Fourier transform we can do this in $O(d \log d)$ steps, as follows.

The *convolution* of two vectors $a, b \in \mathbb{R}^N$ is a vector $a * b \in \mathbb{R}^N$ whose ℓ-th entry is defined by $(a * b)_\ell = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j b_{\ell-j \bmod N}$. Let us set $N = 2d + 1$ (the number of nonzero coefficients of $p \cdot q$) and make the above $(d+1)$-dimensional vectors of coefficients a and b N-dimensional by adding d zeroes. Then the coefficients of the polynomial $p \cdot q$ are proportional to the entries of the convolution: $c_\ell = \sqrt{N}(a * b)_\ell$. It is easy to show that the Fourier coefficients of the convolution of a and b are the products of the Fourier coefficients of a and b: for every $\ell \in \{0, \ldots, N-1\}$ we have $\left(\widehat{a * b}\right)_\ell = \widehat{a}_\ell \cdot \widehat{b}_\ell$. This immediately suggests an algorithm for computing the vector of coefficients c_ℓ: apply the FFT to a and b to get $\widehat{a}$ and $\widehat{b}$, multiply those two vectors entrywise to get $\widehat{a * b}$, apply the inverse FFT to get $a * b$, and finally multiply $a * b$ with $\sqrt{N}$ to get the vector c of the coefficients of $p \cdot q$. Since the FFTs and their inverse take $O(N \log N)$ steps, and pointwise multiplication of two N-dimensional vectors takes $O(N)$ steps, this algorithm takes $O(N \log N) = O(d \log d)$ steps.

Note that if two numbers $a_d \cdots a_1 a_0$ and $b_d \cdots b_1 b_0$ are given in decimal notation, then we can interpret their digits as coefficients of single-variate degree-d polynomials p and q, respectively: $p(x) = \sum_{j=0}^{d} a_j x^j$ and $q(x) = \sum_{k=0}^{d} b_k x^k$. The two numbers will now be $p(10)$ and $q(10)$. Their product is the evaluation of the product-polynomial $p \cdot q$ at the point $x = 10$. This suggests that we can use the above procedure (for fast multiplication of polynomials) to multiply two numbers in $O(d \log d)$ steps, which would be a lot faster than the standard $O(d^2)$ algorithm for multiplication that one learns in primary school. However, in this case we have to be careful since the steps of the above algorithm are themselves multiplications between numbers, which we cannot count at unit cost anymore if our goal is to implement a multiplication between numbers! Still, it turns out that implementing this idea carefully allows one to multiply two d-digit numbers in $O(d \log d \log \log d)$ elementary operations. This is known as the Schönhage-Strassen algorithm [118] (slightly improved further by Fürer [64]), and is one of the ingredients in Shor's algorithm in the next chapter. We'll skip the details.

4.4 The quantum Fourier transform

Since F_N is an $N \times N$ unitary matrix, we can interpret it as a quantum operation, mapping an N-dimensional vector of amplitudes to another N-dimensional vector of amplitudes. This is called the quantum Fourier transform (QFT). In case $N = 2^n$ (which is the only case we will care about), this will be an n-qubit unitary. Notice carefully that this quantum operation does something different from the classical Fourier transform: in the classical case we are given a vector v, written on a piece of paper so to say, and we compute the vector $\widehat{v} = F_N v$, and also write the result on a piece of paper. In the quantum case, we are working on *quantum states*; these are vectors of amplitudes, but

we don't have those written down anywhere—they only exist as the amplitudes in a superposition. We will see below that the QFT can be implemented by a quantum circuit using $O(n^2)$ elementary gates. This is exponentially faster than even the FFT (which takes $O(N \log N) = O(2^n n)$ steps), but it achieves something different: computing the QFT won't give us the entries of the Fourier transform written down on a piece of paper, but only as the amplitudes of the resulting state.

4.5 An efficient quantum circuit

Here we will describe the efficient circuit for the n-qubit QFT. The elementary gates we will allow ourselves are Hadamards and controlled-R_s gates, where

$$R_s = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^s} \end{pmatrix}.$$

Note that $R_1 = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, $R_2 = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$. For large s, $e^{2\pi i/2^s}$ is close to 1 and hence the R_s-gate is close to the identity-gate I. We could implement R_s-gates using Hadamards and controlled-$R_{1/2/3}$ gates, but for simplicity we will just treat each R_s as an elementary gate.

Since the QFT is linear, it suffices if our circuit implements it correctly on n-qubit basis states $|k\rangle$, i.e., it should map

$$|k\rangle \mapsto F_N|k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \omega_N^{jk} |j\rangle.$$

The key to doing this efficiently is to rewrite $F_N|k\rangle$, which turns out to be a *product state* (so F_N does not introduce entanglement when applied to a basis state $|k\rangle$). Let $|k\rangle = |k_1 \dots k_n\rangle$, k_1 being the most significant bit. Note that for integer $j = j_1 \dots j_n$, we can write $j/2^n = \sum_{\ell=1}^{n} j_\ell 2^{-\ell}$. For example, binary 0.101 is $1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} = 5/8$. We have the following sequence of equalities:

$$\begin{aligned}
F_N|k\rangle &= \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i jk/2^n} |j\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i (\sum_{\ell=1}^{n} j_\ell 2^{-\ell}) k} |j_1 \dots j_n\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \prod_{\ell=1}^{n} e^{2\pi i j_\ell k/2^\ell} |j_1 \dots j_n\rangle \\
&= \bigotimes_{\ell=1}^{n} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i k/2^\ell} |1\rangle \right).
\end{aligned}$$

Note that $e^{2\pi i k/2^\ell} = e^{2\pi i 0.k_{n-\ell+1} \dots k_n}$: the $n-\ell$ most significant bits of k don't matter for this value.

As an example, for $n = 3$ we have the 3-qubit product state

$$F_8|k_1 k_2 k_3\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.k_3}|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.k_2 k_3}|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.k_1 k_2 k_3}|1\rangle).$$

This example suggests what the circuit should be. To prepare the first qubit of the desired state $F_8|k_1 k_2 k_3\rangle$, we can just apply a Hadamard to $|k_3\rangle$, giving state $\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{k_3}|1\rangle)$ and observe that

$(-1)^{k_3} = e^{2\pi i 0.k_3}$. To prepare the second qubit of the desired state, apply a Hadamard to $|k_2\rangle$, giving $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.k_2}|1\rangle)$, and then conditioned on k_3 (*before* we apply the Hadamard to $|k_3\rangle$) apply R_2. This multiplies $|1\rangle$ with a phase $e^{2\pi i 0.0k_3}$, producing the correct qubit $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.k_2k_3}|1\rangle)$. Finally, to prepare the third qubit of the desired state, we apply a Hadamard to $|k_1\rangle$, apply R_2 conditioned on k_2 and R_3 conditioned on k_3. This produces the correct qubit $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.k_1k_2k_3}|1\rangle)$. We have now produced all three qubits of the desired state $F_8|k_1k_2k_3\rangle$, *but in the wrong order*: the first qubit should be the third and vice versa. So the final step is just to swap qubits 1 and 3. Figure 4.1 illustrates the circuit in the case $n = 3$. Here the black circles indicate the control-qubits for each of the controlled-R_s operations, and the operation at the end of the circuit swaps qubits 1 and 3. The general case works analogously: starting with $\ell = 1$, we apply a Hadamard to $|k_\ell\rangle$ and then "rotate in" the additional phases required, conditioned on the values of the later bits $k_{\ell+1} \dots k_n$. Some swap gates at the end then put the qubits in the right order.[2]

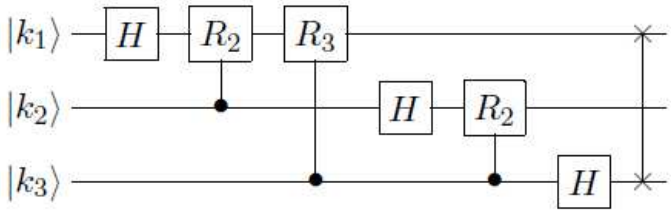

Figure 4.1: The circuit for the 3-qubit QFT

Since the circuit involves n qubits, and at most n gates are applied to each qubit, the overall circuit uses at most n^2 gates. In fact, many of those gates are phase gates R_s with $s \gg \log n$, which are very close to the identity and hence don't do much anyway. We can actually omit those from the circuit, keeping only $O(\log n)$ gates per qubit and $O(n \log n)$ gates overall. Intuitively, the overall error caused by these omissions will be small (Exercise 4 asks you to make this precise). Finally, note that by inverting the circuit (i.e., reversing the order of the gates and taking the adjoint U^* of each gate U) we obtain an equally efficient circuit for the inverse Fourier transform $F_N^{-1} = F_N^*$.

4.6 Application: phase estimation

Suppose we can apply a unitary U and we are given an eigenvector $|\psi\rangle$ of U ($U|\psi\rangle = \lambda|\psi\rangle$), and we would like to approximate the corresponding eigenvalue λ. Since U is unitary, λ must have magnitude 1, so we can write it as $\lambda = e^{2\pi i\phi}$ for some real number $\phi \in [0, 1)$; the only thing that matters is this phase ϕ. Suppose for simplicity that we know that $\phi = 0.\phi_1 \dots \phi_n$ can be written with n bits of precision. Then here's the algorithm for phase estimation:

1. Start with $|0^n\rangle|\psi\rangle$
2. For $N = 2^n$, apply F_N to the first n qubits to get $\frac{1}{\sqrt{2^n}} \sum_{j=0}^{N-1} |j\rangle|\psi\rangle$
   (in fact, $H^{\otimes n} \otimes I$ would have the same effect)

[2] We can implement a SWAP-gate using CNOTs (Exercise 2.4); CNOTs can in turn be constructed from Hadamard and controlled-R_1 (= controlled-Z) gates, which are in the set of elementary gates we allow here.

3. Apply the map $|j\rangle|\psi\rangle \mapsto |j\rangle U^j|\psi\rangle = e^{2\pi i\phi j}|j\rangle|\psi\rangle$. In other words, apply U to the second register for a number of times given by the first register.

4. Apply the inverse Fourier transform F_N^{-1} to the first n qubits and measure the result.

Note that after step 3, the first n qubits are in state $\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} e^{2\pi i\phi j}|j\rangle = F_N|2^n\phi\rangle$, hence the inverse Fourier transform is going to give us $|2^n\phi\rangle = |\phi_1 \dots \phi_n\rangle$ with probability 1.

In case ϕ cannot be written exactly with n bits of precision, one can show that this procedure still (with high probability) spits out a good n-bit approximation to ϕ. We'll omit the calculation.

Exercises

1. For $\omega = e^{2\pi i/3}$ and $F_3 = \frac{1}{\sqrt{3}}\begin{pmatrix} 1 & 1 & 1 \\ 1 & \omega & \omega^2 \\ 1 & \omega^2 & \omega \end{pmatrix}$, calculate $F_3 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ and $F_3 \begin{pmatrix} 1 \\ \omega^2 \\ \omega \end{pmatrix}$

2. Prove that the Fourier coefficients of the convolution of vectors a and b are the product of the Fourier coefficients of a and b. In other words, prove that for every $a, b \in \mathbb{R}^N$ and every $\ell \in \{0, \dots, N-1\}$ we have $\left(\widehat{a * b}\right)_\ell = \widehat{a}_\ell \cdot \widehat{b}_\ell$. Here the Fourier transform $\widehat{a}$ is defined as the vector $F_N a$, and the ℓ-entry of the convolution-vector $a * b$ is $(a*b)_\ell = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} a_j b_{\ell - j \bmod N}$.

3. (H) The *total variation distance* between two probability distributions P and Q on the same set, is defined as $d_{TVD}(P,Q) = \frac{1}{2}\sum_i |P(i) - Q(i)|$. An equivalent alternative way to definite this: $d_{TVD}(P,Q)$ is the maximum, over all events E, of $|P(E) - P(Q)|$. Hence $d_{TVD}(P,Q)$ is small iff all events have roughly the same probability under P and under Q.

   The *Euclidean distance* between two states $|\phi\rangle = \sum_i \alpha_i|i\rangle$ and $|\psi\rangle = \sum_i \beta_i|i\rangle$ is defined as $\||\phi\rangle - |\psi\rangle\| = \sqrt{\sum_i |\alpha_i - \beta_i|^2}$. Assume the two states are unit vectors with (for simplicity) real amplitudes. Suppose the Euclidean distance is small: $\||\phi\rangle - |\psi\rangle\| = \epsilon$. Show that then the probabilities resulting from a measurement (in the computational basis) on the two states are also close: the total variation distance $\frac{1}{2}\sum_i \left|\alpha_i^2 - \beta_i^2\right|$ is $\leq \epsilon$.

4. The *operator norm* of a matrix A is defined as $\|A\| = \max_{v:\|v\|=1} \|Av\|$.
   The *distance* between two matrices A and B is defined as $\|A - B\|$.

   (a) What is the distance between the 2×2 identity matrix and the phase-gate $\begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$?

   (b) What is the distance between the $2^n \times 2^n$ identity matrix I_{2^n} and $\begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \otimes I_{2^{n-1}}$?

   (c) Suppose we have a product of n-qubit unitaries $U = U_T U_{T-1} \cdots U_1$ (for instance, each U_i could be an elementary gate on a few qubits, tensored with identity on the other qubits). Suppose we drop the j-th gate from this sequence: $U' = U_T U_{T-1} \cdots U_{j+1} U_{j-1} \cdots U_1$. Show that $\|U' - U\| = \|I - U_j\|$.

   (d) (H) Now we also drop the k-th unitary: $U'' = U_T U_{T-1} \cdots U_{j+1} U_{j-1} \cdots \cdots U_{k+1} U_{k-1} \cdots U_1$. Show that $\|U'' - U\| \leq \|I - U_j\| + \|I - U_k\|$.

(e) (H) Give a quantum circuit with $O(n \log n)$ elementary gates that has distance less than $1/n$ from the Fourier transform F_{2^n}.

5. Suppose $a \in \mathbb{R}^N$ is a vector (indexed by $\ell = 0, \ldots, N-1$) which is r-periodic in the following sense: there exists an integer r such that $a_\ell = 1$ whenever ℓ is an integer multiple of r, and $a_\ell = 0$ otherwise. Compute the Fourier transform $F_N\, a$ of this vector, i.e., write down a formula for the entries of the vector $F_N a$. Assuming r divides N and $r \ll N$, write down a simple closed form for the formula for the entries. What are the entries with largest magnitude in the vector $F_N\, a$?

Chapter 5

Shor's Factoring Algorithm

5.1 Factoring

Probably the most important quantum algorithm so far is Shor's factoring algorithm [122]. It can find a factor of a composite number N in roughly $(\log N)^2$ steps, which is polynomial in the length $\log N$ of the input. On the other hand, there is no known classical (deterministic or randomized) algorithm that can factor N in polynomial time. The best known classical randomized algorithms run in time roughly

$$2^{(\log N)^\alpha},$$

where $\alpha = 1/3$ for a heuristic upper bound [91] and $\alpha = 1/2$ for a rigorous upper bound [92]. In fact, much of modern cryptography is based on the conjecture that no fast classical factoring algorithm exists [115]. All this cryptography (for example RSA) would be broken if Shor's algorithm could be physically realized. In terms of complexity classes: factoring (rather, the decision problem equivalent to it) is provably in **BQP** but is not known to be in **BPP**. If indeed factoring is not in **BPP**, then the quantum computer would be the first counterexample to the "strong" Church-Turing thesis, which states that all "reasonable" models of computation are polynomially equivalent (see [57] and [111, p.31,36]).

5.2 Reduction from factoring to period-finding

The crucial observation of Shor was that there is an efficient quantum algorithm for the problem of *period-finding* and that factoring can be reduced to this, in the sense that an efficient algorithm for period-finding implies an efficient algorithm for factoring.

We first explain the reduction. Suppose we want to find factors of the composite number $N > 1$. We may assume N is odd and not a prime power, since those cases can easily be filtered out by a classical algorithm. Now randomly choose some integer $x \in \{2, \ldots, N-1\}$ which is coprime[1] to N. If x is not coprime to N, then the greatest common divisor of x and N is a nontrivial factor of N, so then we are already done. From now on consider x and N are coprime, so x is an element of

[1]The *greatest common divisor* of two integers a and b is the largest positive integer c that divides both a and b. If $gcd(a, b) = 1$, then a and b are called *coprime*. The gcd can be computed efficiently (in time roughly linear in the number of bits of a and b) on a classical computer by *Euclid's algorithm*.

the multiplicative group $\mathbb{Z}_N^*$. Consider the sequence

$$1 = x^0 \pmod N, \quad x^1 \pmod N, \quad x^2 \pmod N, \ldots$$

This sequence will cycle after a while: there is a least $0 < r \leq N$ such that $x^r = 1 \pmod N$. This r is called the *period* of the sequence (a.k.a. the *order* of the element x in the group $\mathbb{Z}_N^*$). Assuming N is odd and not a prime power (those cases are easy to factor anyway), it can be shown that with probability $\geq 1/2$, the period r is even and $x^{r/2}+1$ and $x^{r/2}-1$ are not multiples of N.[2] In that case we have:

$$\begin{aligned} x^r &\equiv 1 \mod N \iff \\ (x^{r/2})^2 &\equiv 1 \mod N \iff \\ (x^{r/2}+1)(x^{r/2}-1) &\equiv 0 \mod N \iff \\ (x^{r/2}+1)(x^{r/2}-1) &= kN \text{ for some } k. \end{aligned}$$

Note that $k > 0$ because both $x^{r/2}+1 > 0$ and $x^{r/2}-1 > 0$ ($x > 1$). Hence $x^{r/2}+1$ or $x^{r/2}-1$ will share a factor with N. Because $x^{r/2}+1$ and $x^{r/2}-1$ are not multiples of N this factor will be $< N$, and in fact *both* these numbers will share a non-trivial factor with N. Accordingly, if we have r then we can compute the *greatest common divisors* $gcd(x^{r/2}+1, N)$ and $gcd(x^{r/2}-1, N)$, and both of these two numbers will be non-trivial factors of N. If we are unlucky we might have chosen an x that does not give a factor (which we can detect efficiently), but trying a few different random x gives a high probability of finding a factor.

Thus the problem of factoring reduces to finding the period r of the function given by modular exponentiation $f(a) = x^a \mod N$. In general, the period-finding problem can be stated as follows:

The period-finding problem:
We are given some function $f : \mathbb{N} \to \{0, \ldots, N-1\}$ with the property that there is some unknown $r \in \{0, \ldots, N-1\}$ such that $f(a) = f(b)$ iff $a = b \mod r$. The goal is to find r.

We will show below how we can solve this problem efficiently, using $O(\log \log N)$ evaluations of f and $O(\log \log N)$ quantum Fourier transforms. An evaluation of f can be viewed as analogous to the application of a query in the previous algorithms. Even a somewhat more general kind of period-finding can be solved by Shor's algorithm with very few f-evaluations, whereas any classical bounded-error algorithm would need to evaluate the function $\Omega(N^{1/3}/\sqrt{\log N})$ times in order to find the period [44].

How many steps (elementary gates) does Shor's algorithm take? For $a = N^{O(1)}$, we can compute $f(a) = x^a \mod N$ in $O((\log N)^2 \log \log N \log \log \log N)$ steps, as follows. First compute $x^2 \mod N, x^4 \mod N, x^8 \mod N, \ldots$ by repeated squaring[3] Second, write a in binary as

[2] For those familiar with basic number theory, here is a proof for the special case where $N = p_1 p_2$ is the product of two distinct primes p_1 and p_2; for the general case see [110, Theorem A4.13]. By the Chinese remainder theorem, choosing a uniformly random $x \bmod N$ is equivalent to choosing, independently and uniformly at random, an $x_1 \bmod p_1$ and an $x_2 \bmod p_2$. Let r be the period of the sequence $(x^a \mod N)_a$, and (for $i \in \{1, 2\}$) let r_i be the period of the sequence $(x_i^a \mod p_i)_a$. Because $(x_i, 1)$ generates a size-r_i subgroup of the size-r group generated by (x_1, x_2), Lagrange's Theorem implies that r_i divides r. Hence if r is odd then both r_1 and r_2 must be odd. The probability that r_i is odd is $1/2$, because the group of numbers mod p_i is cyclic and of even size, so half of its elements are squares. Hence the probability that r is odd, is at most $(1/2)^2 = 1/4$. If r is even, then $x^{r/2} \neq 1 \bmod N$, for otherwise the period would be at most $r/2$. If $x^{r/2} = -1 \bmod N$, then $x^{r/2} = -1 \bmod p_1$ and $\bmod p_2$, which has probability at most $(1/2)^2 = 1/4$. Hence $\Pr[r \text{ is odd or } x^{r/2} = 1 \text{ or } x^{r/2} = -1] \leq \Pr[r \text{ is odd}] + \Pr[x^{r/2} = -1] \leq \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$.

[3] Using the Schönhage-Strassen algorithm for fast multiplication [118, 88], see Exercise 1. We could also use the more recent improvement of Fürer [64], who removes most of the $\log \log N$ factor.

$a = \sum_{i \geq 0} a_i 2^i$ and observe that $x^a = \prod_{i \geq 0} x^{a_i 2^i} = \prod_{i : a_i = 1} x^{2^i}$. Hence multiplying together some of the numbers produced in the first step (mod N) gives us $f(a)$.

Moreover, as explained in the previous chapter, the quantum Fourier transform can be implemented using $O((\log N)^2)$ steps. Accordingly, Shor's algorithm finds a factor of N using an expected number of roughly $(\log N)^2 (\log \log N)^2 \log \log \log N$ steps, which is only slightly worse than quadratic in the length of the input.

5.3 Shor's period-finding algorithm

Now we will show how Shor's algorithm finds the period r of the function f, given a "black-box" that maps $|a\rangle|0^n\rangle \rightarrow |a\rangle|f(a)\rangle$. We can always efficiently pick some $q = 2^\ell$ such that $N^2 < q \leq 2N^2$. Then we can implement the Fourier transform F_q using $O((\log N)^2)$ gates. Let O_f denote the unitary that maps $|a\rangle|0^n\rangle \mapsto |a\rangle|f(a)\rangle$, where the first register consists of ℓ qubits, and the second of $n = \lceil \log N \rceil$ qubits.

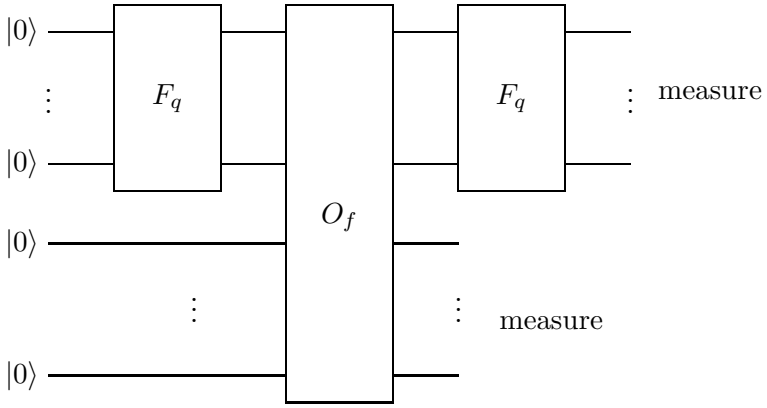

Figure 5.1: Shor's period-finding algorithm

Shor's period-finding algorithm is illustrated in Figure 5.1.[4] Start with $|0^\ell\rangle|0^n\rangle$. Apply the QFT (or just ℓ Hadamard gates) to the first register to build the uniform superposition

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|0^n\rangle.$$

The second register still consists of zeroes. Now use the "black-box" to compute $f(a)$ in quantum parallel:

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|f(a)\rangle.$$

Observing the second register gives some value $f(s)$, with $s < r$. Let m be the number of elements of $\{0, \ldots, q-1\}$ that map to the observed value $f(s)$. Because $f(a) = f(s)$ iff $a = s \mod r$, the a of the form $a = jr + s$ $(0 \leq j < m)$ are exactly the a for which $f(a) = f(s)$. Thus the first register collapses to a superposition of $|s\rangle, |r+s\rangle, |2r+s\rangle, |3r+s\rangle, \ldots$; this superposition runs until the

[4] Notice the resemblance of the basic structure (Fourier, f-evaluation, Fourier) with the basic structure of Simon's algorithm (Hadamard, query, Hadamard).

last number of the form $jr+s$ that is $< q$, let's define m to be the number of elements in this superposition, i.e., the number of integers j such that $jr+s \in \{0,\ldots,q-1\}$ (depending on s, this m will be $\lceil q/r \rceil$ or $\lfloor q/r \rfloor$). The second register collapses to the classical state $|f(s)\rangle$. We can now ignore the second register, and have in the first:

$$\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |jr+s\rangle.$$

Applying the QFT again gives

$$\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} \frac{1}{\sqrt{q}} \sum_{b=0}^{q-1} e^{2\pi i \frac{(jr+s)b}{q}} |b\rangle = \frac{1}{\sqrt{mq}} \sum_{b=0}^{q-1} e^{2\pi i \frac{sb}{q}} \left(\sum_{j=0}^{m-1} e^{2\pi i \frac{jrb}{q}} \right) |b\rangle.$$

We want to see which $|b\rangle$ have large squared amplitudes—those are the b we are likely to see if we now measure. Using that $\sum_{j=0}^{m-1} z^j = (1-z^m)/(1-z)$ for $z \neq 1$ (see Appendix B), we compute:

$$\sum_{j=0}^{m-1} e^{2\pi i \frac{jrb}{q}} = \sum_{j=0}^{m-1} \left(e^{2\pi i \frac{rb}{q}} \right)^j = \begin{cases} m & \text{if } e^{2\pi i \frac{rb}{q}} = 1 \\ \frac{1-e^{2\pi i \frac{mrb}{q}}}{1-e^{2\pi i \frac{rb}{q}}} & \text{if } e^{2\pi i \frac{rb}{q}} \neq 1 \end{cases} \tag{5.1}$$

Easy case: r divides q. Let us do an easy case first. Suppose r divides q, so the whole period "fits" an integer number of times in the domain $\{0,\ldots,q-1\}$ of f, and $m = q/r$. For the first case of Eq. (5.1), note that $e^{2\pi i rb/q} = 1$ iff rb/q is an integer iff b is a multiple of q/r. Such b will have squared amplitude equal to $(m/\sqrt{mq})^2 = m/q = 1/r$. Since there are exactly r such b, together they have all the amplitude. Thus we are left with a superposition where only the b that are integer multiples of q/r have nonzero amplitude. Observing this final superposition gives some random multiple $b = cq/r$, with c a random number $0 \leq c < r$. Thus we get a b such that

$$\frac{b}{q} = \frac{c}{r},$$

where b and q are known to the algorithm, and c and r are not. There are $\phi(r) \in \Omega(r/\log\log r)$ numbers smaller than r that are coprime to r [73, Theorem 328], so c will be coprime to r with probability $\Omega(1/\log\log r)$. Accordingly, an expected number of $O(\log\log N)$ repetitions of the procedure of this section suffices to obtain a $b = cq/r$ with c coprime to r.[5] Once we have such a b, we can obtain r as the denominator by writing b/q in lowest terms.

Hard case: r does not divide q. Because our q is a power of 2, it is actually quite likely that r does not divide q. However, the same algorithm will still yield with high probability a b which is *close* to a multiple of q/r. Note that q/r is no longer an integer, and $m = \lfloor q/r \rfloor$, possibly $+1$. All calculations up to and including Eq. (5.1) are still valid. Using $|1-e^{i\theta}| = 2|\sin(\theta/2)|$, we can rewrite the absolute value of the second case of Eq. (5.1) to

$$\frac{|1-e^{2\pi i \frac{mrb}{q}}|}{|1-e^{2\pi i \frac{rb}{q}}|} = \frac{|\sin(\pi mrb/q)|}{|\sin(\pi rb/q)|}.$$

[5] The number of required f-evaluations for period-finding can actually be reduced from $O(\log\log N)$ to $O(1)$.

The right-hand side is the ratio of two sine-functions of b, where the numerator oscillates much faster than the denominator because of the additional factor of m. Note that the denominator is close to 0 (making the ratio large) iff b is close to an integer multiple of q/r. For most of those b, the numerator won't be close to 0. Hence, roughly speaking, the ratio will be small if b is far from an integer multiple of q/r, and large for most b that are close to a multiple of q/r. Doing the calculation precisely, one can show that with high probability the measurement yields a b such that

$$\left|\frac{b}{q} - \frac{c}{r}\right| \leq \frac{1}{2q}.$$

As in the easy case, b and q are known to us while c and r are unknown.

Two distinct fractions, each with denominator $\leq N$, must be at least $1/N^2 > 1/q$ apart.[6] Therefore c/r is the *only* fraction with denominator $\leq N$ at distance $\leq 1/2q$ from b/q. Applying a classical method called "continued-fraction expansion" to b/q efficiently gives us the fraction with denominator $\leq N$ that is closest to b/q (see the next section). This fraction must be c/r. Again, with good probability c and r will be coprime, in which case writing c/r in lowest terms gives us r.

5.4 Continued fractions

Let $[a_0, a_1, a_2, \ldots]$ (finite or infinite) denote

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots}}}$$

This is called a *continued fraction* (CF). The a_i are the *partial quotients*. We assume these to be positive natural numbers ([73, p.131] calls such CF "simple"). $[a_0, \ldots, a_n]$ is the nth *convergent* of the fraction. [73, Theorem 149 & 157] gives a simple way to compute numerator and denominator of the nth convergent from the partial quotients:

If

$$\begin{array}{lll} p_0 = a_0, & p_1 = a_1 a_0 + 1, & p_n = a_n p_{n-1} + p_{n-2} \\ q_0 = 1, & q_1 = a_1, & q_n = a_n q_{n-1} + q_{n-2} \end{array}$$

then $[a_0, \ldots, a_n] = \dfrac{p_n}{q_n}$. Moreover, this fraction is in lowest terms.

Note that q_n increases at least exponentially with n ($q_n \geq 2q_{n-2}$). Given a real number x, the following "algorithm" gives a continued fraction expansion of x [73, p.135]:

$$\begin{array}{ll} a_0 := \lfloor x \rfloor, & x_1 := 1/(x - a_0) \\ a_1 := \lfloor x_1 \rfloor, & x_2 := 1/(x - a_1) \\ a_2 := \lfloor x_2 \rfloor, & x_3 := 1/(x - a_2) \\ \ldots & \end{array}$$

Informally, we just take the integer part of the number as the partial quotient and continue with the inverse of the decimal part of the number. The convergents of the CF approximate x as follows [73, Theorem 164 & 171]:

[6] Consider two fractions $z = x/y$ and $z' = x'/y'$ with $y, y' \leq N$. If $z \neq z'$ then $|xy' - x'y| \geq 1$, and hence $|z - z'| = |(xy' - x'y)/yy'| \geq 1/|yy'| \geq 1/N^2$.

If $x = [a_0, a_1, \ldots]$ then $\left| x - \frac{p_n}{q_n} \right| < \frac{1}{q_n^2}$.

Recall that q_n increases exponentially with n, so this convergence is quite fast. Moreover, p_n/q_n provides the *best* approximation of x among all fractions with denominator $\leq q_n$ [73, Theorem 181]:

If $n > 1$, $q \leq q_n$, $p/q \neq p_n/q_n$, then $\left| x - \frac{p_n}{q_n} \right| < \left| x - \frac{p}{q} \right|$.

Exercises

1. This exercise is about efficient classical implementation of modular exponentiation.

   (a) (H) Given n-bit numbers x and N (where $n = \lceil \log_2 N \rceil$), compute the whole sequence x^0 mod N, x^1 mod N, x^2 mod N, x^4 mod N, x^8 mod N,x^{16} mod $N, \ldots, x^{2^{n-1}}$ mod N, using $O(n^2 \log(n) \log\log(n))$ steps.

   (b) Suppose n-bit number a can be written as $a = a_{n-1} \ldots a_1 a_0$ in binary. Express x^a mod N as a product of the numbers computed in part (a).

   (c) Show that you can compute $f(a) = x^a$ mod N in $O(n^2 \log(n) \log\log(n))$ steps.

2. Consider the function $f(a) = 7^a$ mod 10.

   (a) What is the period r of f?

   (b) Show how Shor's algorithm find the period of f, using a Fourier transform over $q = 128$ elements. Write down all intermediate superpositions of the algorithm for this case (don't just copy the general expressions from the notes, but instantiate them with actual numbers as much as possible). You may assume you're lucky, meaning the first run of the algorithm already gives a measurement outcome $b = cq/r$ with c coprime to r.

3. (H) This exercise explains basic RSA encryption. Suppose Alice wants to allow other people to send encrypted messages to her, such that she is the only one who can decrypt them. She believes that factoring an n-bit number can't be done efficiently (efficient = in time polynomial in n). So in particular, she doesn't believe in quantum computing.

   Alice chooses two large random prime numbers, p and q, and computes their product $N = p \cdot q$ (a typical size is to have N a number of $n = 1024$ bits, which corresponds to both p and q being numbers of roughly 512 bits). She computes the so-called Euler ϕ-function: $\phi(N) = (p-1)(q-1)$; she also chooses an *encryption exponent* e, which doesn't share any nontrivial factor with $\phi(N)$ (i.e., e and $\phi(N)$ are coprime). Group theory guarantees there is an efficiently computable *decryption exponent* d such that $de = 1$ mod $\phi(N)$. The *public key* consists of e and N (Alice puts this on her homepage), while the *secret key* consists of d and N. Any number $m \in \{1, \ldots, N-1\}$ that is coprime to N, can be used as a message. There are $\phi(N)$ such m, and these numbers form a group under the operation of multiplication mod N. The number of bits $n = \lceil \log_2 N \rceil$ of N is the maximal length (in bits) of a message m and also the length (in bits) of the encryption. The encryption function is defined as $C(m) = m^e$ mod N, and the decryption function is $D(c) = c^d$ mod N.

   (a) Give a randomized algorithm by which Alice can efficiently generate the secret and public key.

(b) Show that Bob can efficiently compute the encoding $C(m)$ of the message m that he wants to send to Alice, knowing the public key but not the private key.

(c) Show that $D(C(m)) = m$ for all possible messages.

(d) Show that Alice can efficiently decrypt the encryption $C(m)$ she receives from Bob.

(e) Show that if Charlie could factor N, then he could efficiently decrypt Bob's message.

Chapter 6

Hidden Subgroup Problem

6.1 Hidden Subgroup Problem

6.1.1 Group theory reminder

A group G consists of a set of elements (which is usually denoted by G as well) and an operation $\circ : G \times G \to G$ (often written as addition or multiplication), such that

1. the operation is associative: $g \circ (h \circ k) = (g \circ h) \circ k$ for all $g, h, k \in G$;
2. there is an *identity element* $e \in G$ satisfying $e \circ g = g \circ e = g$ for every $g \in G$;
3. and every $g \in G$ has an *inverse* $g^{-1} \in G$, such that $g \circ g^{-1} = g^{-1} \circ g = e$ (if the group operation is written as addition, then g^{-1} is written as $-g$).

We often abbreviate $g \circ h$ to gh. The group is *Abelian* (or *commutative*) if $gh = hg$ for all $g, h \in G$. Simple examples of finite additive Abelian groups are $G = \{0,1\}^n$ with bitwise addition mod 2 as the group operation, and $G = \mathbb{Z}_N$, the "cyclic group" of integers mod N. The set $G = \mathbb{Z}_N^*$ is the multiplicative group consisting of all integers in $\{1, \ldots, N-1\}$ that are coprime to N, with multiplication mod N as the group operation.[1] An important example of a non-Abelian group is the "symmetric group" S_n, which is the group of $n!$ permutations of n elements, using composition as the group operation.

A *subgroup* H of G, denoted $H \leq G$, is a subset of G that is itself a group, i.e., it contains e and is closed under taking products and inverses. A (left) *coset* of H is a set $gH = \{gh \mid h \in H\}$, i.e., a translation of H by the element g. All cosets of H have size $|H|$, and it is easy to show that two cosets gH and $g'H$ are either equal or disjoint, so the set of cosets partitions G into equal-sized parts.[2] Note that g and g' are in the same coset of H iff $g^{-1}g' \in H$.

If $T \subseteq G$, then we use $\langle T \rangle$ to denote the set of elements of G that we can write as products of elements from T and their inverses. This $H = \langle T \rangle$ is a subgroup of G, and T is called a *generating set* of H. Note that adding one more element $t \notin \langle T \rangle$ to T at least doubles the size of the generated subgroup, because H and tH are disjoint and $H \cup tH \subseteq \langle T \cup \{t\} \rangle$. This implies that every $H \leq G$ has a generating set of size $\leq \log |H| \leq \log |G|$. We abbreviate $\langle \{\gamma\} \rangle$ to $\langle \gamma \rangle$, which is the cyclic group generated by γ; every cyclic group of size N is isomorphic to $\mathbb{Z}_N$.

[1] This group $\mathbb{Z}_N^*$ is isomorphic to the additive group $\mathbb{Z}_{\phi(N)}$, where Euler's ϕ-function counts the elements of $\{1, \ldots, N-1\}$ that are coprime to N. For example, $\mathbb{Z}_p^*$ is isomorphic to $\mathbb{Z}_{p-1}$.

[2] This also proves Lagrange's theorem for finite groups: if $H \leq G$ then $|H|$ divides $|G|$.

6.1.2 Definition and some instances of the HSP

The *Hidden Subgroup Problem* is the following:

> Given a known group G and a function $f : G \to S$ where S is some finite set.
> Suppose f has the property that there exists a subgroup $H \leq G$ such that f is constant within each coset, and distinct on different cosets: $f(g) = f(g')$ iff $gH = g'H$.
> Goal: find H.

We assume f can be computed efficiently, meaning in time polynomial in $\log |G|$ (the latter is the number of bits needed to describe an input $g \in G$ for f). Since H may be large, "finding H" typically means finding a generating set for H.

This looks like a rather abstract algebraic problem, but many important problems can be written as an instance of the HSP. We will start with some examples where G is Abelian.

Simon's problem. This is a very natural instance of HSP. Here G is the additive group $\mathbb{Z}_2^n = \{0,1\}^n$ of size 2^n, $H = \{0, s\}$ for a "hidden" $s \in \{0,1\}^n$, and f satisfies $f(x) = f(y)$ iff $x - y \in H$. Clearly, finding the generator of H (i.e., finding s) solves Simon's problem.

Period-finding. As we saw in Chapter 5, we can factor a large number N if we can solve the following: given an x that is coprime to N and associated function $f : \mathbb{Z} \to \mathbb{Z}_N^*$ by $f(a) = x^a \bmod N$, find the period r of f.[3] Since $\langle x \rangle$ is a size-r subgroup of the group $\mathbb{Z}_N^*$, the period r divides $|\mathbb{Z}_N^*| = \phi(N)$. Hence we can restrict the domain of f to $\mathbb{Z}_{\phi(N)}$.

Period-finding is an instance of the HSP as follows. Let $G = \mathbb{Z}_{\phi(N)}$ and consider its subgroup $H = \langle r \rangle$ of all multiples of r up to $\phi(N)$ (i.e., $H = r\mathbb{Z}_{\phi(N)} = \{0, r, 2r, \ldots, \phi(N) - r\}$). Note that because of its periodicity, f is constant on each coset $s + H$ of H, and distinct on different cosets. Also, f is efficiently computable by repeated squaring. Since the hidden subgroup H is generated by r, finding the generator of H solves the period-finding problem.

Discrete logarithm. Another problem often used in classical public-key cryptography is the discrete logarithm problem: given a generator γ of a cyclic multiplicative group C of size N (so $C = \{\gamma^a \mid a \in \{0, \ldots, N-1\}\}$), and $A \in C$, can we find the unique $a \in \{0, 1, \ldots, N-1\}$ such that $\gamma^a = A$? This a is called the discrete logarithm of A (w.r.t. generator γ). It is generally believed that classical computers need time roughly exponential in $\log N$ to compute a from A (and one can actually *prove* this in a model where we can only implement group operations via some "black-box"). This assumption underlies for instance the security of Diffie-Hellman key exchange (where $C = \mathbb{Z}_p^*$ for some large prime p, see Exercise 3), as well as elliptic-curve cryptography.

Discrete log is an instance of the HSP as follows. We take $G = \mathbb{Z}_N \times \mathbb{Z}_N$ and define function $f : G \to C$ by $f(x, y) = \gamma^x A^{-y}$, which is efficiently computable by repeated squaring. For group elements $g_1 = (x_1, y_1), g_2 = (x_2, y_2) \in G$ we have

$$f(g_1) = f(g_2) \iff \gamma^{x_1 - ay_1} = \gamma^{x_2 - ay_2} \iff (x_1 - x_2) = a(y_1 - y_2) \bmod N \iff g_1 - g_2 \in \langle (a, 1) \rangle.$$

Let H be the subgroup of G generated by the element $(a, 1)$, then we have an instance of the HSP. Finding the generator of the hidden subgroup H gives us a, solving the discrete log problem.

[3] This r is also known as the *order* of the element x in the group $\mathbb{Z}_N^*$, so this problem is also known as *order-finding*.

6.2 An efficient quantum algorithm if G is Abelian

In this section we show that HSPs where G (and hence H) is Abelian, and where f is efficiently computable, can be solved efficiently by a quantum algorithm. This generalizes Shor's factoring algorithm, and will also show that discrete logarithms can be computed efficiently.

6.2.1 Representation theory and the quantum Fourier transform

We start by explaining the basics of representation theory. The idea here is to replace group elements by matrices, so that linear algebra can be used as a tool in group theory. A d-dimensional *representation* of a multiplicative group G is a map $\rho : g \mapsto \rho(g)$ from G to the set of $d \times d$ invertible complex matrices, satisfying $\rho(gh) = \rho(g)\rho(h)$ for all $g, h \in G$. The latter property makes the map ρ a *homomorphism*. It need not be an *iso*morphism; for example, the constant-1 function is a trivial representation of any group. The *character* corresponding to ρ is the map $\chi_\rho : G \to \mathbb{C}$ defined by $\chi_\rho(g) = \mathrm{Tr}(\rho(g))$.

Below we restrict attention to the case where G is Abelian (and usually finite). In this case we may assume the dimension d to be 1 without loss of generality, so a representation ρ and the corresponding character χ_ρ are just the same function. Also, it is easy to see that the complex values $\chi_\rho(g)$ have modulus 1, because $|\chi_\rho(g^k)| = |\chi_\rho(g)|^k$ for all integers k. The "Basis Theorem" of group theory says that every finite Abelian group G is isomorphic to a direct product $\mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_k}$ of cyclic groups. First consider just one cyclic group $\mathbb{Z}_N$, written additively. Consider the discrete Fourier transform (Chapter 4), which is an $N \times N$ matrix. Ignoring the normalizing factor of $1/\sqrt{N}$, its kth column may be viewed as a map $\chi_k : \mathbb{Z}_N \to \mathbb{C}$ defined by $\chi_k(j) = \omega_N^{jk}$, where $\omega_N = e^{2\pi i/N}$. Note that $\chi_k(j + j') = \chi_k(j)\chi_k(j')$, so χ_k is actually a 1-dimensional representation (i.e., a character function) of $\mathbb{Z}_N$. In fact, the N characters corresponding to the N columns of the Fourier matrix are *all* the characters of $\mathbb{Z}_N$. For Abelian groups G that are (isomorphic to) a product $\mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_k}$ of cyclic groups, the $|G| = N_1 \cdots N_k$ characters are just the products of the characters of the individual cyclic groups $\mathbb{Z}_{N_j}$. Note that the characters are pairwise orthogonal.

The set of all characters of G forms a group $\widehat{G}$ with the operation of pointwise multiplication. This is called the *dual group* of G. If $H \leq G$, then the following is a subgroup of $\widehat{G}$ of size $|G|/|H|$:

$$H^\perp = \{\chi_k \mid \chi_k(h) = 1 \text{ for all } h \in H\}.$$

Let us interpret the quantum Fourier transform in terms of the characters. For $k \in \mathbb{Z}_N$, define the state whose entries are the (normalized) values of χ_k:

$$|\chi_k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \chi_k(j)|j\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \omega_N^{jk}|j\rangle.$$

With this notation, the QFT just maps the standard (computational) basis of $\mathbb{C}^N$ to the orthonormal basis corresponding to the characters:

$$F_N : |k\rangle \mapsto |\chi_k\rangle.$$

As we saw in Chapter 4, this map can be implemented by an efficient quantum circuit if N is a power of 2. The QFT corresponding to a group G that is isomorphic to $\mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_k}$ is just the tensor product of the QFTs for the individual cyclic groups. For example, the QFT corresponding to $\mathbb{Z}_2$ is the Hadamard gate H, so the QFT corresponding to $\mathbb{Z}_2^n$ is $H^{\otimes n}$ (which is of course very different from the QFT corresponding to $\mathbb{Z}_{2^n}$).

6.2.2 A general algorithm for Abelian HSP

The following is an efficient quantum algorithm for solving the HSP for some Abelian group G (written additively) and function $f : G \to S$. This algorithm, sometimes called the "standard algorithm" for HSP, was first observed by Kitaev [84] (inspired by Shor's algorithm) and worked out further by many, for instance Mosca and Ekert [106].

1. Start with $|0\rangle|0\rangle$, where the two registers have dimension $|G|$ and $|S|$, respectively.
2. Create a uniform superposition over G in the first register: $\frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle|0\rangle$.
3. Compute f in superposition: $\frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle|f(g)\rangle$.
4. Measure the second register. This yields some value $f(s)$ for unknown $s \in G$. The first register collapses to a superposition over the g with the same f-value as s (i.e., the coset $s + H$): $\frac{1}{\sqrt{|H|}} \sum_{h \in H} |s + h\rangle$.
5. Apply the QFT corresponding to G to this state, giving $\frac{1}{\sqrt{|H|}} \sum_{h \in H} |\chi_{s+h}\rangle$.
6. Measure and output the resulting g.

The key to understanding this algorithm is to observe that step 5 maps the uniform superposition over the coset $s + H$ to a uniform superposition over the labels of $H^\perp$:

$$\begin{aligned}\frac{1}{\sqrt{|H|}} \sum_{h \in H} |\chi_{s+h}\rangle &= \frac{1}{\sqrt{|H||G|}} \sum_{h \in H} \sum_{g \in G} \chi_{s+h}(g)|g\rangle \\ &= \frac{1}{\sqrt{|H||G|}} \sum_{g \in G} \chi_s(g) \sum_{h \in H} \chi_h(g)|g\rangle = \sqrt{\frac{|H|}{|G|}} \sum_{g : \chi_g \in H^\perp} \chi_s(g)|g\rangle,\end{aligned}$$

where the last equality follows from the orthogonality of characters of the group H (note that χ_g restricted to H is a character of H, and it's the constant-1 character iff $\chi_g \in H^\perp$):

$$\sum_{h \in H} \chi_h(g) = \sum_{h \in H} \chi_g(h) = \begin{cases} |H| & \text{if } \chi_g \in H^\perp \\ 0 & \text{if } \chi_g \notin H^\perp \end{cases}$$

The phases $\chi_s(g)$ do not affect the probabilities of the final measurement, since $|\chi_s(g)|^2 = 1$. The above algorithm thus samples uniformly from the (labels of) elements of $H^\perp$. Each such element $\chi_g \in H^\perp$ gives us a constraint on H because $\chi_g(h) = 1$ for all $h \in H$.[4] Generating a small number of such elements will give sufficient information to find the generators of H itself. Consider our earlier examples of Abelian HSP:

[4] This is a *linear* constraint mod N. For example, say $G = \mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}$, and $g = (g_1, g_2)$ is the label of an element of $H^\perp$. Then $1 = \chi_g(h) = \omega_{N_1}^{g_1 h_1} \omega_{N_2}^{g_2 h_2}$ for all $h = (h_1, h_2) \in H$, equivalently $g_1 h_1 N_2 + g_2 h_2 N_1 = 0 \bmod N$.

Simon's problem. Recall that $G = \mathbb{Z}_2^n = \{0,1\}^n$ and $H = \{0, s\}$ for the HSP corresponding to Simon's problem. Setting up the uniform superposition over G can be done by applying $H^{\otimes n}$ to the initial state $|0^n\rangle$ of the first register. The QFT corresponding to G is just $H^{\otimes n}$. The 2^n character functions are $\chi_g(x) = (-1)^{x \cdot g}$. The algorithm will uniformly sample from labels of elements of

$$H^\perp = \{\chi_g \mid \chi_g(h) = 1 \text{ for all } h \in H\} = \{\chi_g \mid g \cdot s = 0\}.$$

Accordingly, the algorithm samples uniformly from the $g \in \{0,1\}^n$ such that $g \cdot s = 0 \pmod 2$. Doing this an expected $O(n)$ times gives $n-1$ linearly independent equations about s, from which we can find s using Gaussian elimination.

Period-finding. For the HSP corresponding to period-finding, $G = \mathbb{Z}_{\phi(N)}$ and $H = \langle r \rangle$, and

$$H^\perp = \{\chi_b \mid e^{2\pi i b h/\phi(N)} = 1 \text{ for all } h \in H\} = \{\chi_b \mid br/\phi(N) \in \{0, \ldots, r-1\}\}.$$

Accordingly, the output of the algorithm is an integer multiple $b = c\phi(N)/r$ of $\phi(N)/r$, for uniformly random $c \in \{0, \ldots, r-1\}$.

Notice that the algorithm doesn't actually know $\phi(N)$, which creates two problems. First, of the 4 numbers $b, c, \phi(N), r$ involved in the equation $b = c\phi(N)/r$ we only know the measurement outcome b, which is not enough to compute r. Second, step 5 of the algorithm wants to do a QFT corresponding to the group $\mathbb{Z}_{\phi(N)}$ but it doesn't know $\phi(N)$ (and even if we knew $\phi(N)$, we've only seen how to efficiently do a QFT over $\mathbb{Z}_q$ when q is a power of 2). Fortunately, if we actually use the QFT over $\mathbb{Z}_q$ for q a power of 2 that is roughly N^2 (and in step 1 set up a uniform superposition over $\mathbb{Z}_q$ instead of G), then one can show that the above algorithm still works, with high probability yielding a number b that's close to an integer multiple of q/r.[5] This is basically just Shor's algorithm as described in Chapter 5.

Discrete logarithm. For the HSP corresponding to the discrete log problem, where $G = \mathbb{Z}_N \times \mathbb{Z}_N$ and $H = \langle (a, 1) \rangle$, a small calculation shows that $H^\perp = \{\chi_{(c,-ac)} \mid c \in \mathbb{Z}_N\}$ (see Exercise 2). Hence sampling from $H^\perp$ yields some label $(c, -ac) \in G$ of an element of $H^\perp$, from which we can compute the discrete logarithm a. The QFT corresponding to G is $F_N \otimes F_N$, which we don't know how to implement efficiently, but which we can replace by $F_q \otimes F_q$ for some power-of-2 q somewhat larger than N.

In the above algorithm we assumed G is a *finite* Abelian group. These techniques have been much extended to the case of infinite groups such as $G = \mathbb{Z}$ and even $\mathbb{R}^d$, to obtain efficient quantum algorithms for problems like Pell's equation [70], and computing properties in number fields [27].

6.3 General non-Abelian HSP

6.3.1 The symmetric group and the graph isomorphism problem

The Abelian HSP covers a number of interesting computational problems, including period-finding and discrete log. However, there are also some interesting computational problems that can be cast

[5] There is something to be proved here, but we will skip the details. In fact one can even use a Fourier transform for $q = O(N)$ instead of $O(N^2)$ [69]. Note that this also reduces the number of qubits used by Shor's algorithm from roughly $3 \log N$ to roughly $2 \log N$.

as an instance of HSP with a *non-Abelian* G. Unfortunately we do not have an efficient algorithm for most non-Abelian HSPs.

A good example is the *graph isomorphism* (GI) problem: given two undirected n-vertex graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, decide whether there exists a bijection taking the vertices of $\mathcal{G}_1$ to those of $\mathcal{G}_2$ that makes the two graphs equal. No efficient classical algorithm is known for GI, so it would be great if we could solve this efficiently on a quantum computer.[6]

How can we try to solve this via the HSP? Let $\mathcal{G}$ be the $2n$-vertex graph that is the disjoint union of the two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$. Let $G = S_{2n}$. Let f map $\pi \in S_{2n}$ to $\pi(\mathcal{G})$, which means that edge (i, j) becomes edge $(\pi(i), \pi(j))$. Let H be the automorphism group $\mathrm{Aut}(\mathcal{G})$ of $\mathcal{G}$, which is the set of all $\pi \in S_{2n}$ that map $\mathcal{G}$ to itself. This gives an instance of the HSP, and solving it would give us a generating set of $H = \mathrm{Aut}(\mathcal{G})$.

Assume for simplicity that each of $\mathcal{G}_1$ and $\mathcal{G}_2$ is connected. If $\mathcal{G}_1$ and $\mathcal{G}_2$ are not isomorphic, then the only automorphisms of $\mathcal{G}$ are the ones that permute vertices inside $\mathcal{G}_1$ and inside $\mathcal{G}_2$: $\mathrm{Aut}(\mathcal{G}) = \mathrm{Aut}(\mathcal{G}_1) \times \mathrm{Aut}(\mathcal{G}_2)$. However, if the two graphs are isomorphic, then $\mathrm{Aut}(\mathcal{G})$ will also contain a permutation that swaps the first n with the second n vertices. Accordingly, if we were able to find a generating set of the hidden subgroup $H = \mathrm{Aut}(\mathcal{G})$, then we can just check whether all generators are in $\mathrm{Aut}(\mathcal{G}_1) \times \mathrm{Aut}(\mathcal{G}_2)$ and decide graph isomorphism.

6.3.2 Non-Abelian QFT on coset states

One can try to design a quantum algorithm for general, non-Abelian instances of the HSP along the lines of the earlier standard algorithm: set up a uniform superposition over a random coset of H, apply the QFT corresponding to G, measure the final state, and hope that the result gives useful information about H. QFTs corresponding to non-Abelian G are much more complicated than in the Abelian case, because the representations ρ can have dimension $d > 1$ and hence do not coincide with the corresponding character χ_ρ. For completeness, let's write down the QFT anyway. Let $\widehat{G}$ denote the set of "irreducible" representations of G, and $\dim(\rho)$ be the dimension of a particular $\rho \in \widehat{G}$. We can assume without loss of generality that the $\dim(\rho) \times \dim(\rho)$ matrices $\rho(g)$ are unitary. The QFT corresponding to G is defined as follows:

$$|g\rangle \longmapsto \sum_{\rho \in \widehat{G}} \sqrt{\frac{\dim(\rho)}{|G|}} |\rho\rangle \sum_{i,j=1}^{\dim(\rho)} \rho(g)_{ij} |i, j\rangle,$$

where $|\rho\rangle$ denotes a name or label of ρ. It can be shown that this map is unitary. In particular, $|G| = \sum_{\rho \in \widehat{G}} \dim(\rho)^2$, which implies that the dimensions on the left and the right are the same, and that the right-hand state has norm 1. In many cases this QFT can still be implemented with an efficient quantum circuit, including for the symmetric group $G = S_{2n}$ that is relevant for graph isomorphism [15, 104]. However, that is not enough for an efficient algorithm: the standard algorithm does not always yield much information about the hidden $H \leq S_{2n}$ [67, 105, 71].

There are some special cases of non-Abelian HSP that can be computed efficiently, for instance for normal subgroups [72], solvable groups [129], and nil-2 groups [79].

[6] For a long time, the best algorithm for GI took time roughly $2^{\sqrt{n}}$ [13], but in a recent breakthrough Babai gave a "quasi-polynomial" algorithm, which is $n^{(\log n)^{O(1)}}$ [12]. That's not yet polynomial time, but a lot faster than before.

6.3.3 Query-efficient algorithm

While we do not have a general efficient quantum algorithm for the non-Abelian HSP, there does exist an algorithm that needs to compute f only a few times, i.e., a *query-efficient* algorithm. We will sketch this now. Consider steps 1–4 of the standard algorithm for the Abelian case. Even in the general non-Abelian case, this produces a coset state, i.e., a uniform superposition over the elements of a uniformly random coset of H. Suppose we do this m times, producing a state $|\psi_H\rangle$ which is the tensor product of m random coset states.[7] One can show that the states corresponding to different hidden subgroups are pairwise almost orthogonal: $|\langle\psi_H|\psi_{H'}\rangle|$ is exponentially small in m. The hidden subgroup H is generated by a set of $\leq \log|G|$ elements. Hence the total number of possible H that we want to distinguish is at most $\binom{|G|}{\log|G|} \leq 2^{(\log|G|)^2}$. This allows us to define a POVM measurement $\{E_H\}$ (see Section 1.2.2), with one element for each possible hidden subgroup H, such that if we measure $|\psi_H\rangle$ with this POVM, then we are likely to get the correct outcome H (see Exercise 4 for the idea). Choosing m some polynomial in $\log|G|$ suffices for this. While this POVM need not be efficiently implementable, at least the number of times we need to query the function f is only m. Ettinger et al. [58] even showed that $m = O(\log|G|)$ suffices.

For those interested in more HSP results, a good source is Childs's lecture notes [40, Chapter 4–14].

Exercises

1. Show that the Deutsch-Jozsa problem for $n = 1$ (i.e., where $f : \{0,1\} \to \{0,1\}$) is an instance of the HSP. Explicitly say what G, f, H, and $H^\perp$ are, and how sampling from $H^\perp$ allows you to solve the problem.

2. Show that for the HSP corresponding to discrete log, we indeed have $H^\perp = \{\chi_{(c,-ac)} \mid c \in \mathbb{Z}_N\}$ as claimed near the end of Section 6.2.2.

3. This exercise explains Diffie-Hellman key exchange, which is secure under the assumption that the adversary cannot efficiently compute discrete logarithms. Alice and Bob choose a public key consisting of a large prime p (say, of 1000 or 2000 bits) and generator γ of the group $\mathbb{Z}_p^*$, which has size $\phi(p) = p-1$. To agree on a shared secret key K, Alice chooses a uniformly random $a \in \{0,\ldots,p-2\}$ and sends Bob the group element $A = \gamma^a$; Bob chooses a uniformly random $b \in \{0,\ldots,p-2\}$ and sends Alice $B = \gamma^b$. Alice and Bob use $K = \gamma^{ab}$ as their secret key, which they can use for instance to encrypt messages using a one-time pad.

   (a) Show that both Alice and Bob can efficiently compute K given the communication.

   (b) Show that an adversary who can efficiently compute discrete logarithms, can compute K from the public key and the communication tapped from the channel (i.e., A, B, p and γ, but not a and b).

4. Suppose we are given an unknown state $|\psi_i\rangle$ from a known set of K states $\{|\psi_j\rangle \mid j \in [K]\}$.

   (a) Suppose the states are pairwise orthogonal: $\langle\psi_j|\psi_k\rangle = \delta_{jk}$. Give a projective measurement that determines i with probability 1.

[7] Strictly speaking we should consider the tensor product of m copies of the *mixed* state ρ_H that is the uniform average over all coset states of H.

(b) (H) Suppose the states are pairwise *almost* orthogonal: $|\langle\psi_j|\psi_k\rangle| \ll 1/K^2$ for all distinct $j, k \in [K]$. Define $E_i = \frac{2}{3}|\psi_i\rangle\langle\psi_i|$. Show that $I - \sum_{i=1}^{K} E_i$ is positive semidefinite.

(c) Under the same assumption as (b), give a POVM that determines i with success probability at least 2/3.

5. (H) Suppose we have an efficient algorithm to produce, from a given undirected n-vertex graph $\mathcal{G}$, the following n^2-qubit state, where the basis states correspond to $n \times n$ adjacency matrices:

$$a_{\mathcal{G}} \sum_{\pi \in S_n} |\pi(\mathcal{G})\rangle.$$

Here $a_{\mathcal{G}}$ is a scalar that makes the norm equal to 1. Use this procedure to efficiently decide (with high success probability) whether two given graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic or not.

Chapter 7

Grover's Search Algorithm

The second-most important quantum algorithm after Shor's, is Grover's quantum search problem from 1996 [68]. While it doesn't provide an exponential speed-up, it is much more widely applicable than Shor's algorithm.

7.1 The problem

The search problem:
For $N = 2^n$, we are given an arbitrary $x \in \{0,1\}^N$. The goal is to find an i such that $x_i = 1$ (and to output 'no solutions' if there are no such i).

This problem may be viewed as a simplification of the problem of searching an N-slot unordered database. Classically, a randomized algorithm would need $\Theta(N)$ queries to solve the search problem. Grover's algorithm solves it in $O(\sqrt{N})$ queries, and $O(\sqrt{N}\log N)$ other gates.

7.2 Grover's algorithm

Let $O_{x,\pm}|i\rangle = (-1)^{x_i}|i\rangle$ denote the $\pm$-type oracle for the input x, and R be the unitary transformation that puts a -1 in front all basis states $|i\rangle$ where $i \neq 0^n$, and that does nothing to the other basis states $|0^n\rangle$.[1] The *Grover iterate* is $\mathcal{G} = H^{\otimes n}RH^{\otimes n}O_{x,\pm}$. Note that 1 Grover iterate makes 1 query.

Grover's algorithm starts in the n-bit state $|0^n\rangle$, applies a Hadamard transformation to each qubit to get the uniform superposition $|U\rangle = \frac{1}{\sqrt{N}}\sum_i |i\rangle$ of all N indices, applies $\mathcal{G}$ to this state k times (for some k to be chosen later), and then measures the final state. Intuitively, what happens is that in each iteration some amplitude is moved from the indices of the 0-bits to the indices of the 1-bits. The algorithm stops when almost all of the amplitude is on the 1-bits, in which case a measurement of the final state will probably give the index of a 1-bit. Figure 7.1 illustrates this.

To analyze this, define the following "good" and "bad" states:

$$|G\rangle = \frac{1}{\sqrt{t}}\sum_{i:x_i=1}|i\rangle \text{ and } |B\rangle = \frac{1}{\sqrt{N-t}}\sum_{i:x_i=0}|i\rangle.$$

[1] This R is independent of the input x, and can be implemented using $O(n)$ elementary gates (see Exercise 2.7).

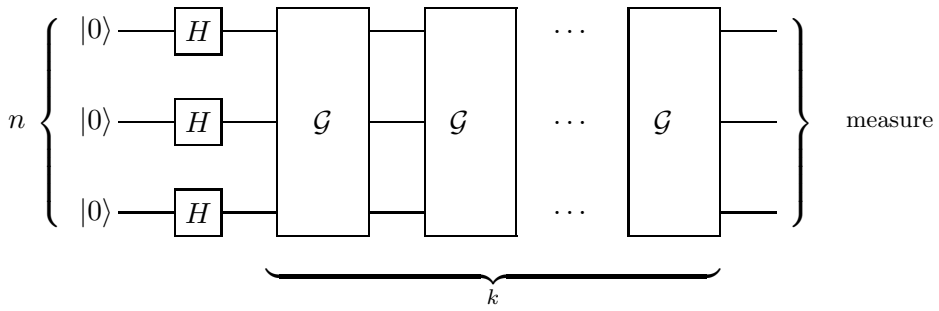

Figure 7.1: Grover's algorithm, with k Grover iterates

Then the uniform state over all indices edges can be written as

$$|U\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle = \sin(\theta)|G\rangle + \cos(\theta)|B\rangle, \quad \text{for } \theta = \arcsin(\sqrt{t/N}).$$

The Grover iterate $\mathcal{G}$ is actually the product of two *reflections*[2] (in the 2-dimensional space spanned by $|G\rangle$ and $|B\rangle$): $O_{x,\pm}$ is a reflection through $|B\rangle$, and

$$H^{\otimes n} R H^{\otimes n} = H^{\otimes n}(2|0^n\rangle\langle 0^n| - I)H^{\otimes n} = 2|U\rangle\langle U| - I$$

is a reflection through $|U\rangle$. Here is Grover's algorithm restated, assuming we know the fraction of solutions is $\varepsilon = t/N$:

1. Set up the starting state $|U\rangle = H^{\otimes n}|0\rangle$
2. Repeat the following $k = O(1/\sqrt{\varepsilon})$ times:
   (a) Reflect through $|B\rangle$ (i.e., apply $O_{x,\pm}$)
   (b) Reflect through $|U\rangle$ (i.e., apply $H^{\otimes n} R H^{\otimes n}$)
3. Measure the first register and check that the resulting i is a solution

Geometric argument: There is a fairly simple geometric argument why the algorithm works. The analysis is in the 2-dimensional real plane spanned by $|B\rangle$ and $|G\rangle$. We start with

$$|U\rangle = \sin(\theta)|G\rangle + \cos(\theta)|B\rangle.$$

The two reflections (a) and (b) increase the angle from θ to 3θ, moving us towards the good state, as illustrated in Figure 7.2.

The next two reflections (a) and (b) increase the angle with another 2θ, etc. More generally, after k applications of (a) and (b) our state has become

$$\sin((2k+1)\theta)|G\rangle + \cos((2k+1)\theta)|B\rangle.$$

[2] A reflection through a subspace V is a unitary A such that $Av = v$ for all vectors $v \in V$, and $Aw = -w$ for all w orthogonal to V. In the two reflections used in one Grover iteration, the subspace V will be 1-dimensional, corresponding to $|B\rangle$ and to $|U\rangle$, respectively.

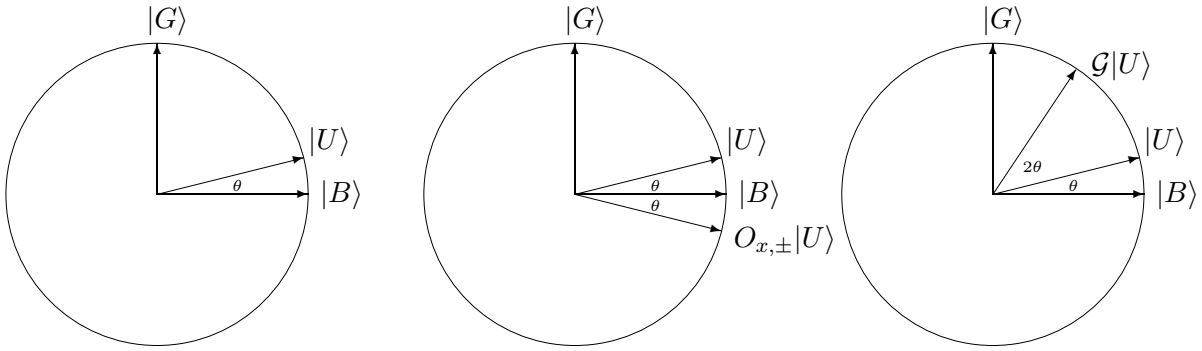

Figure 7.2: The first iteration of Grover: (left) start with $|U\rangle$, (middle) reflect through $|B\rangle$ to get $O_{x,\pm}|U\rangle$, (right) reflect through $|U\rangle$ to get $\mathcal{G}|U\rangle$

If we now measure, the probability of seeing a solution is $P_k = \sin((2k+1)\theta)^2$. We want P_k to be as close to 1 as possible. Note that if we can choose $\tilde{k} = \pi/4\theta - 1/2$, then $(2\tilde{k}+1)\theta = \pi/2$ and hence $P_{\tilde{k}} = \sin(\pi/2)^2 = 1$. An example where this works is if $t = N/4$, for then $\theta = \pi/6$ and $\tilde{k} = 1$.

Unfortunately $\tilde{k} = \pi/4\theta - 1/2$ will usually not be an integer, and we can only do an integer number of Grover iterations. However, if we choose k to be the integer closest to $\tilde{k}$, then our final state will still be close to $|G\rangle$ and the failure probability will still be small (assuming $t \ll N$):

$$\begin{aligned} 1 - P_k &= \cos((2k+1)\theta)^2 = \cos((2\tilde{k}+1)\theta + 2(k-\tilde{k})\theta)^2 \\ &= \cos(\pi/2 + 2(k-\tilde{k})\theta)^2 = \sin(2(k-\tilde{k})\theta)^2 \leq \sin(\theta)^2 = \frac{t}{N}, \end{aligned}$$

where we used $|k - \tilde{k}| \leq 1/2$. Since $\arcsin(\theta) \geq \theta$, the number of queries is $k \leq \pi/4\theta \leq \frac{\pi}{4}\sqrt{\frac{N}{t}}$.

Algebraic argument: For those who don't like geometry, here's an alternative (but equivalent) algebraic argument. Let a_k denote the amplitude of the indices of the t 1-bits after k Grover iterates, and b_k the amplitude of the indices of the 0-bits. Initially, for the uniform superposition $|U\rangle$ we have $a_0 = b_0 = 1/\sqrt{N}$. Using that $H^{\otimes n}RH^{\otimes n} = [2/N] - I$, where $[2/N]$ is the $N \times N$ matrix in which all entries are $2/N$, we find the following recursion:

$$\begin{aligned} a_{k+1} &= \frac{N-2t}{N}a_k + \frac{2(N-t)}{N}b_k \\ b_{k+1} &= \frac{-2t}{N}a_k + \frac{N-2t}{N}b_k \end{aligned}$$

The following formulas, due to Boyer et al. [28], provide a closed form for a_k and b_k (which may be verified by substituting them into the recursion). With $\theta = \arcsin(\sqrt{t/N})$ as before, define

$$\begin{aligned} a_k &= \frac{1}{\sqrt{t}}\sin((2k+1)\theta) \\ b_k &= \frac{1}{\sqrt{N-t}}\cos((2k+1)\theta) \end{aligned}$$

Accordingly, after k iterations the success probability (the sum of squares of the amplitudes of the locations of the t 1-bits) is the same as in the geometric analysis

$$P_k = t \cdot a_k^2 = (\sin((2k+1)\theta))^2.$$

Accordingly, we have a bounded-error quantum search algorithm with $O(\sqrt{N/t})$ queries, *assuming we know* t. We now list (without full proofs) a number of usuful variants of grover:

- If we know t exactly, then the algorithm can be tweaked to end up in *exactly* the good state. Roughly speaking, you can make the angle θ slightly smaller, such that $\tilde{k} = \pi/4\theta - 1/2$ becomes an integer (see Exercise 7).

- If we do *not* know t, then there is a problem: we do not know which k to use, so we do not know when to stop doing the Grover iterates. Note that if k gets too big, the success probability $P_k = (\sin((2k+1)\theta))^2$ goes down again! However, a slightly more complicated algorithm due to [28] (basically running the above algorithm with systematic different guesses for k) shows that an *expected* number of $O(\sqrt{N/t})$ queries still suffices to find a solution if there are t solutions. If there is no solution ($t = 0$), then we can easily detect that by checking x_i for the i that the algorithm outputs.

- If we know a lower bound τ on the actual (possibly unknown) number of solutions t, then the above algorithm uses an *expected* number of $O(\sqrt{N/\tau})$ queries. If we run this algorithm for up to three times its expected number of queries, then (by Markov's inequality) with probability at least 2/3 it will have found a solution. This way we can turn an *expected* runtime into a *worst-case* runtime.

- If we do not know t but would like to reduce the probability of not finding a solution to some small $\varepsilon > 0$, then we can do this using $O(\sqrt{N \log(1/\varepsilon)})$ queries (see Exercise 8).
  NB: The important part here is that the $\log(1/\varepsilon)$ is inside the square-root; usual error-reduction by $O(\log(1/\varepsilon))$ repetitions of basic Grover would give the worse upper bound of $O(\sqrt{N}\log(1/\varepsilon))$ queries.

7.3 Amplitude amplification

The analysis that worked for Grover's algorithm is actually much more generally applicable (we will also see it again in the next chapter). Let $\chi : \mathbb{Z} \to \{0,1\}$ be any Boolean function; inputs $z \in \mathbb{Z}$ satisfying $\chi(z) = 1$ are called *solutions*. Suppose we have an algorithm to check whether z is a solution. This can be written as a unitary O_χ that maps $|z\rangle \mapsto (-1)^{\chi(z)}|z\rangle$. Suppose also we have some (quantum or classical) algorithm $\mathcal{A}$ that uses no intermediate measurements and has probability p of finding a solution when applied to starting state $|0\rangle$. Classically, we would have to repeat $\mathcal{A}$ roughly $1/p$ times before we find a solution. The *amplitude amplification* algorithm below (from [30]) only needs to run $\mathcal{A}$ $O(1/\sqrt{p})$ times:

1. Setup the starting state $|U\rangle = \mathcal{A}|0\rangle$
2. Repeat the following $O(1/\sqrt{p})$ times:
   (a) Reflect through $|B\rangle$ (i.e., apply O_χ)
   (b) Reflect through $|U\rangle$ (i.e., apply $\mathcal{A}R\mathcal{A}^{-1}$)
3. Measure the first register and check that the resulting element x is marked.

Defining $\theta = \arcsin(\sqrt{p})$ and good and bad states $|G\rangle$ and $|B\rangle$ in analogy with the earlier geometric argument for Grover's algorithm, the same reasoning shows that amplitude amplification indeed finds a solution with high probability. This way, we can speed up a very large class of classical heuristic algorithms: any algorithm that has some non-trivial probability of finding a solution can be amplified to success probability nearly 1 (provided we can efficiently check solutions, i.e., implement O_χ).

Note that the Hadamard transform $H^{\otimes n}$ can be viewed as an algorithm with success probability $p = t/N$ for a search problem of size N with t solutions, because $H^{\otimes n}|0^n\rangle$ is the uniform superposition over all N locations. Hence Grover's algorithm is a special case of amplitude amplification, where $O_\chi = O_{x,\pm}$ and $\mathcal{A} = H^{\otimes n}$.

7.4 Application: satisfiability

Grover's algorithm has many applications: basically any classical algorithm that has some search-component can be improved using Grover's algorithm as a subroutine. This includes many basic computer applications such as finding shortest paths and minimum spanning trees, various other graph algorithms, etc.

We can also use it to speed up the computation of **NP**-complete problems, albeit only quadratically, not exponentially. As an example, consider the satisfiability problem: we are given a Boolean formula $\phi(i_1, \ldots, i_n)$ and want to know if it has a satisfying assignment, i.e., a setting of the bits $i_1, \ldots, i_n$ that makes $\phi(i_1, \ldots, i_n) = 1$. A classical brute-force search along all 2^n possible assignments takes time roughly 2^n.

To find a satisfying assignment faster, define the $N = 2^n$-bit input to Grover's algorithm by $x_i = \phi(i)$, where $i \in \{0,1\}^n$. For a given assignment $i = i_1 \ldots i_n$ it is easy to compute $\phi(i)$ classically in polynomial time. We can write that computation as a reversible circuit (using only Toffoli gates), corresponding to a unitary U_ϕ that maps $|i, 0, 0\rangle \mapsto |i, \phi(i), w_i\rangle$, where the third register holds some classical workspace the computation may have needed. To apply Grover we need an oracle that puts the answer in the phase and doesn't leave workspace around (as that would mess up the interference effects). Define O_x as the unitary that first applies U_ϕ, then applies a Z-gate to the second register, and then applies U_ϕ^{-1} to "clean up" the workspace again. This has the form we need for Grover: $O_{x,\pm}|i\rangle = (-1)^{x_i}|i\rangle$, where we omitted the workspace qubits, which start and end in $|0\rangle$. Now we can run Grover and find a satisfying assignment with high probability if there is one, using a number of elementary operations that is $\sqrt{2^n}$ times some polynomial factor.

Exercises

1. (a) Suppose $n = 2$, and $x = x_{00}x_{01}x_{10}x_{11} = 0001$. Give the initial, intermediate, and final superpositions in Grover's algorithm, for $k = 1$ queries. What is the success probability?

   (b) Give the final superposition for the above x after $k = 2$ iterations. What is now the success probability?

2. Show that if the number of solutions is $t = N/4$, then Grover's algorithm always finds a solution with certainty after just one query. How many queries would a classical algorithm need to find a solution with certainty if $t = N/4$? And if we allow the classical algorithm error probability 1/10?

3. (H) Let $x = x_0 \dots x_{N-1}$ be a sequence of distinct integers, where $N = 2^n$. We can query these in the usual way, i.e., we can apply unitary $O_x : |i, 0\rangle \mapsto |i, x_i\rangle$, as well as its inverse. The *minimum* of x is defined as $\min\{x_i \mid i \in \{0, \dots, N-1\}\}$. Give a quantum algorithm that finds (with probability $\geq 2/3$) an index achieving the minimum, using at most $O(\sqrt{N} \log N)$ queries to the input.

   Bonus: give a quantum algorithm that uses $O(\sqrt{N})$ queries.

4. Let $x = x_0 \dots x_{N-1}$, where $N = 2^n$ and $x_i \in \{0,1\}^n$, be an input that we can query in the usual way. We are promised that this input is 2-to-1: for each i there is exactly one other j such that $x_i = x_j$.[3] Such an (i, j)-pair is called a *collision*.

   (a) Suppose S is a uniformly randomly chosen set of $s \leq N/2$ elements of $\{0, \dots, N-1\}$. What is the probability that there exists a collision in S?

   (b) (H) Give a classical randomized algorithm that finds a collision (with probability $\geq 2/3$) using $O(\sqrt{N})$ queries to x.

   (c) (H) Give a quantum algorithm that finds a collision (with probability $\geq 2/3$) using $O(N^{1/3})$ queries.

5. Suppose we have a database with $N = 2^n$ binary slots, containing t ones (solutions) and $N - t$ zeroes. You may assume you know the number t.

   (a) Show that we can use Grover's algorithm to find the positions of *all* t ones, using an expected number of $O(t\sqrt{N})$ queries to the database. You can argue on a high level, no need to draw actual quantum circuits.

   (b) (H) Show that this can be improved to an expected number of $O(\sqrt{tN})$ queries.

6. Consider an undirected graph $G = (V, E)$, with vertex set $V = \{1, \dots, n\}$ and edge-set E. We say G is *connected* if, for every pair of vertices $i, j \in V$, there is a path between i and j in the graph. The *adjacency matrix* of G is the $n \times n$ Boolean matrix M where $M_{ij} = 1$ iff $(i, j) \in E$ (note that M is a symmetric matrix because G is undirected). Suppose we are given input graph G in the form of a unitary that allows us to query whether an edge (i, j) is present in G or not:

$$O_M : |i, j, b\rangle \mapsto |i, j, b \oplus M_{ij}\rangle.$$

   (a) Assume G is connected. Suppose we have a set A of edges which we already know to be in the graph (so $A \subseteq E$; you can think of A as given classically, you don't have to query it). Let $G_A = (V, A)$ be the subgraph induced by only these edges, and suppose G_A is not connected, so it consists of $c > 1$ connected components. Call an edge $(i, j) \in E$ "good" if it connects two of these components. Give a quantum algorithm that finds a good edge with an *expected* number of $O(n/\sqrt{c-1})$ queries to M.

   (b) Give a quantum algorithm that uses at most $O(n^{3/2})$ queries to M and decides (with success probability at least 2/3) whether G is connected or not.

   (c) Show that classical algorithms for deciding (with success probability at least 2/3) whether G is connected, need to make $\Omega(n^2)$ queries to M.

[3] The 2-to-1 inputs for Simon's algorithm are a very special case of this, where x_i equals x_j if $i = j \oplus s$ for fixed but unknown $s \in \{0,1\}^n$.

7. At the end of Section 7.2 we claimed without proof that Grover's algorithm can be tweaked to work *with probability 1* if we know the number of solutions exactly. For $N = 2^n$, this question will ask you to provide such an exact algorithm for an N-bit database $x \in \{0,1\}^N$ with a unique solution (so we are promised that there is exactly one $i \in \{0,1\}^n$ with $x_i = 1$, and our goal is to find this i).

   (a) Give the success probability of the basic version of Grover's algorithm after k iterations.

   (b) Suppose the optimal number of iterations $\tilde{k} = \frac{\pi}{4 \arcsin(1/\sqrt{N})} - \frac{1}{2}$ is not an integer. Show that if we round $\tilde{k}$ up to the nearest integer, doing $\lceil \tilde{k} \rceil$ iterations, then the algorithm will have success probability strictly less than 1.

   (c) Define a new $2N$-bit database $y \in \{0,1\}^{2N}$, indexed by $(n+1)$-bit strings $j = j_1 \dots j_n j_{n+1}$, by setting

$$y_j = \begin{cases} 1 & \text{if } x_{j_1 \dots j_n} = 1 \text{ and } j_{n+1} = 0, \\ 0 & \text{otherwise.} \end{cases}$$

   Show how you can implement the following $(n+1)$-qubit unitary

$$S_y : |j\rangle \mapsto (-1)^{y_j} |j\rangle,$$

   using one query to x (of the usual form $O_x : |i,b\rangle \mapsto |i, b \oplus x_i\rangle$) and a few elementary gates.

   (d) Let $\gamma \in [0, 2\pi)$ and let $U_\gamma = \begin{pmatrix} \cos\gamma & -\sin\gamma \\ \sin\gamma & \cos\gamma \end{pmatrix}$ be the corresponding rotation matrix. Let $\mathcal{A} = H^{\otimes n} \otimes U_\gamma$ be an $(n+1)$-qubit unitary. What is the probability (as a function of γ) that measuring the state $\mathcal{A}|0^{n+1}\rangle$ in the computational basis gives a solution $j \in \{0,1\}^{n+1}$ for y (i.e., such that $y_j = 1$)?

   (e) (H) Give a quantum algorithm that finds the unique solution in database x with probability 1 using $O(\sqrt{N})$ queries to x.

8. Given query access to $x \in \{0,1\}^N$, with unknown Hamming weight $t = |x|$, we want to find a solution, i.e., an index $i \in \{0, \dots, N-1\}$ such that $x_i = 1$. If $x = 0^N$ then our search algorithm should output "no solution."

   (a) (H) Suppose we know an integer s such that $t \in \{1, \dots, s\}$. Give a quantum algorithm that finds a solution with probability 1, using $O(\sqrt{sN})$ queries to x.

   (b) Suppose we know that $t \in \{s+1, \dots, N\}$. Give a quantum algorithm that finds a solution with probability at least $1 - 2^{-s}$, using $O(\sqrt{sN})$ queries to x.

   (c) For given $\varepsilon > 2^{-N}$, give a quantum algorithm that solves the search problem with probability $\geq 1 - \varepsilon$ using $O(\sqrt{N \log(1/\varepsilon)})$ queries, without assuming anything about t.

Chapter 8

Quantum Walk Algorithms

8.1 Classical random walks

Consider an undirected graph G with N vertices. Suppose at least an ε-fraction of the vertices are "marked," and we would like to find a marked vertex. One way to do this is with a *random walk*:

> Start at some specific vertex y of the graph.
> Repeat the following a number of times: Check if y is marked, and if not then choose one of its neighbors at random and set y to be that neighbor.

This may seem like a stupid algorithm, but it has certain advantages. For instance, it only needs space $O(\log N)$, because you only need to keep track of the current vertex y, and maybe a counter that keeps track of how many steps you've already taken.[1] Such an algorithm can for example decide whether there is a path from a specific vertex y to a specific vertex x using $O(\log N)$ space. We'd start the walk at y and only x would be marked; one can show that if there exists a path from y to x in G, then we will reach x in $\text{poly}(N)$ steps.

Let us restrict attention to d-regular graphs without self-loops, so each vertex has exactly d neighbors. A random walk on such a graph G corresponds to an $N \times N$ symmetric matrix P, where $P_{x,y} = 1/d$ if (x, y) is an edge in G, and $P_{x,y} = 0$ otherwise. This P is the normalized adjacency matrix of G. If $v \in \mathbb{R}^N$ is a vector with a 1 at position y and 0s elsewhere, then Pv is a vector whose xth entry is $(Pv)_x = 1/d$ if (x, y) is an edge, and $(Pv)_x = 0$ otherwise. In other words, Pv is the uniform probability distribution over the neighbors of y, which is what you get by taking one step of the random walk starting at y. More generally, if v is a probability distribution on the vertices, then Pv is the new probability distribution on vertices after taking one step of the random walk, and $P^k v$ is the probability distribution after taking k steps.

Suppose we start with some probability-distribution vector v (which may or may not be concentrated at one vertex y). We will assume G is connected and not bipartite. Then $P^k v$ will converge to the uniform distribution over all vertices, and the speed of convergence is determined by the "gap" between the first eigenvalue of P and all other eigenvalues. This can be seen as follows. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$ be the eigenvalues of P, ordered by size, and $v_1, \ldots, v_N$ be corresponding orthogonal eigenvectors.[2] The largest eigenvalue is $\lambda_1 = 1$, and corresponds to the

[1] Here we're assuming the neighbors of any one vertex are efficiently computable, so you don't actually need to keep the whole graph in memory. This will be true for all graphs we consider here.

[2] Analyzing graphs by looking at the eigenvalues of their adjacency matrix is called "algebraic graph theory" or "spectral graph theory," see for instance [32].

eigenvector $v_1 = u = (1/N)$ which is the uniform distribution over all vertices. One can show that our assumption that G is connected implies $\lambda_2 < 1$, and our assumption that G is not bipartite implies $\lambda_N > -1$. Hence all eigenvalues λ_i for $i \in \{2, \ldots, N\}$ will be in $(-1, 1)$; the corresponding eigenvector v_i will be orthogonal to the uniform vector u, so the sum of its entries is 0. Let $\delta > 0$ be the difference between $\lambda_1 = 1$ and $\max_{i \geq 2} |\lambda_i|$ (hence $|\lambda_i| \leq 1 - \delta$ for all $i \geq 2$). This δ is called the "spectral gap" of the graph.

Now decompose the starting distribution v as $v = \sum_{i=1}^{N} \alpha_i v_i$. Since the sum of v's entries is 1, and the sum of v_1's entries is 1, while each other eigenvector v_i ($i \geq 2$) has entries summing to 0, it follows that $\alpha_1 = 1$. Now let us see what happens if we apply the random walk for k steps, starting from v:

$$P^k v = P^k \left(\sum_i \alpha_i v_i \right) = \sum_i \alpha_i \lambda_i^k v_i = u + \sum_{i \geq 2} \alpha_i \lambda_i^k v_i.$$

Consider the squared norm of the difference between $P^k v$ and u:

$$\left\| P^k v - u \right\|^2 = \left\| \sum_{i \geq 2} \alpha_i \lambda_i^k v_i \right\|^2 = \sum_{i \geq 2} |\alpha_i|^2 |\lambda_i|^{2k} \leq \|v\|^2 (1 - \delta)^{2k}.$$

Since v is a probability distribution, we have $\|v\|^2 \leq 1$. By choosing $k = \ln(1/\eta)/\delta$, we get $\left\| P^k v - u \right\| \leq \eta$. In particular, if δ is not too small, then we get quick convergence of the random walk to the uniform distribution u, no matter which distribution v we started with.[3] Once we are close to the uniform distribution, we have probability roughly ε of hitting a marked vertex. Of course, the same happens if we just pick a vertex uniformly at random, but that may not always be an option if the graph is given implicitly.

Suppose it costs $\boldsymbol{S}$ to set up an initial state v; it costs $\boldsymbol{U}$ to update the current vertex, i.e., to perform one step of the random walk; and it costs $\boldsymbol{C}$ to check whether a given vertex is marked. "Cost" is left undefined for now, but typically it will count number of queries to some input, or number of elementary operations. Consider a classical search algorithm that starts at v, and then repeats the following until it finds a marked vertex: check if the current vertex is marked, and if not run a random walk for roughly $1/\delta$ steps to get close to the uniform distribution. Ignoring constant factors, the expected cost before this procedure finds a marked item, is on the order of

$$\boldsymbol{S} + \frac{1}{\varepsilon} \left(\boldsymbol{C} + \frac{1}{\delta} \boldsymbol{U} \right). \tag{8.1}$$

8.2 Quantum walks

We will now modify the classical random walk algorithm preceding Eq. (8.1) to a quantum algorithm, where the distribution-preserving matrix P is changed to a norm-preserving matrix $W(P)$ (i.e., a unitary). This is due to Magniez et al. [99], inspired by Szegedy [124]; our presentation is mostly based on Santha's survey paper [117], to which we refer for more details and references.

While the basis state of a classical random walk is the current vertex we are at, a basis state of a quantum walk has *two* registers, the first corresponding to the *current* vertex and the second

[3] Convergence in total variation distance can be derived from this by Cauchy-Schwarz, choosing $\eta \ll 1/\sqrt{N}$.

corresponding to the *previous* vertex. Equivalently, a basis state of a quantum walk corresponds to an *edge* of the graph.

Our resulting quantum walk algorithm for search will actually be quite analogous to Grover's algorithm. We'll call a basis state $|x\rangle|y\rangle$ "good" if x is a marked vertex, and "bad" otherwise. Define $|p_x\rangle = \sum_y \sqrt{P_{xy}}|y\rangle$ to be the uniform superposition over the neighbors of x. As for Grover, define "good" and "bad" states as the superpositions over good and bad basis states:

$$|G\rangle = \frac{1}{\sqrt{|M|}} \sum_{x \in M} |x\rangle|p_x\rangle \text{ and } |B\rangle = \frac{1}{\sqrt{N-|M|}} \sum_{x \notin M} |x\rangle|p_x\rangle,$$

where M denotes the set of marked vertices. Note that $|G\rangle$ is just the uniform superposition over all edges (x, y) where the first coordinate is marked, and $|B\rangle$ is just the uniform superposition over all edges (x, y) where the first coordinate is not marked.

If $\varepsilon = |M|/N$ and $\theta := \arcsin(\sqrt{\varepsilon})$ then the uniform state over all edges can be written as

$$|U\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle|p_x\rangle = \sin(\theta)|G\rangle + \cos(\theta)|B\rangle.$$

Here is the algorithm for searching a marked vertex if an ε-fraction is marked[4]:

1. Setup the starting state $|U\rangle$
2. Repeat the following $O(1/\sqrt{\varepsilon})$ times:
   (a) Reflect through $|B\rangle$
   (b) Reflect through $|U\rangle$
3. Measure the first register and check that the resulting vertex x is marked.

We'll explain in a moment how to implement (a) and (b). Assuming we know how to do that, the proof that this algorithm finds a marked vertex is the same as for Grover and for amplitude amplification (Chapter 7). We start with $|U\rangle = \sin(\theta)|G\rangle + \cos(\theta)|B\rangle$. The two reflections (a) and (b) increase the angle from θ to 3θ, moving us towards the good state (as for Grover, draw a 2-dimensional picture with axes $|B\rangle$ and $|G\rangle$ to see this). More generally, after k applications of (a) and (b) our state has become

$$\sin((2k+1)\theta)|G\rangle + \cos((2k+1)\theta)|B\rangle.$$

Choosing $k \approx \frac{\pi}{4\theta} = O(1/\sqrt{\varepsilon})$, we will have $\sin((2k+1)\theta) \approx 1$, at which point measuring the first register will probably yield a marked vertex x.

(a) Reflect through $|B\rangle$. Reflecting through $|B\rangle$ is relatively straightforward: we just have to "recognize" whether the first register contains a marked x, and put a -1 if so.

[4] As in Grover, if we don't know ε then we just run the algorithm repeatedly with exponentially decreasing guesses for ε (1/2, 1/4, 1/8, ...). If at the end we still haven't found a marked item, we'll conclude that probably none exists.

(b) Reflect through $|U\rangle$. This is where the quantum walk comes in. Let $\mathcal{A}$ be the subspace $\text{span}\{|x\rangle|p_x\rangle\}$ and $\mathcal{B}$ be $\text{span}\{|p_y\rangle|y\rangle\}$. Let $\text{ref}(\mathcal{A})$ denote the unitary which is a reflection through $\mathcal{A}$ (i.e., $\text{ref}(\mathcal{A})v = v$ for all vectors $v \in \mathcal{A}$, and $\text{ref}(\mathcal{A})w = -w$ for all vectors w orthogonal to $\mathcal{A}$) and $\text{ref}(\mathcal{B})$ be a reflection through $\mathcal{B}$. Define $W(P) = \text{ref}(\mathcal{B})\text{ref}(\mathcal{A})$ to be the product of these two reflections. This is the unitary analogue of P, and may be called "one step of a quantum walk." Suppose we are able to implement the following two operations (even in a controlled manner):

(1) $|x\rangle|0\rangle \mapsto |x\rangle|p_x\rangle$
(2) $|0\rangle|y\rangle \mapsto |p_y\rangle|y\rangle$

Since (1) and (2) prepare a uniform superposition over the neighbors of x and y, respectively, one can think of them as taking one classical walk step "in superposition." Note that $\text{ref}(\mathcal{A})$ can be implemented by applying the inverse of (1), putting a minus if the second register is not $|0\rangle$, and applying (1). We can similarly implement $\text{ref}(\mathcal{B})$ using (2) and its inverse. Hence we can think of $W(P) = \text{ref}(\mathcal{B})\text{ref}(\mathcal{A})$ as corresponding to *four* steps of the classical walk in superposition.

To do the reflection through $|U\rangle$, we now want to construct a unitary $R(P)$ that maps $|U\rangle \mapsto |U\rangle$ and $|\psi\rangle \mapsto -|\psi\rangle$ for all $|\psi\rangle$ that are orthogonal to $|U\rangle$ (and that are in the span of the eigenvectors of $W(P)$). We will do that by means of *phase estimation* on $W(P)$ (see Section 4.6). The eigenvalues of $W(P)$ can be related to the eigenvalues $\lambda_1, \lambda_2, \ldots$ of P as follows. Let $\theta_j \in [0, \pi/2]$ be such that $|\lambda_j| = \cos(\theta_j)$. We won't prove it here, but it turns out that the eigenvalues of $W(P)$ are of the form $e^{\pm 2i\theta_j}$. $W(P)$ has one eigenvalue-1 eigenvector, which is $|U\rangle$, corresponding to $\theta_1 = 0$. The spectral gap of P is δ. Hence all other eigenvectors of $W(P)$ correspond to an eigenvalue $e^{\pm 2i\theta_j}$ where $\theta_j \geq \sqrt{2\delta}$, because $1 - \delta \geq |\lambda_j| = \cos(\theta_j) \geq 1 - \theta_j^2/2$.

The procedure $R(P)$ will add a second auxiliary register (initially $|0\rangle$) and do phase estimation with precision $\sqrt{\delta}/2$ to detect the unique eigenvalue-1 eigenvector $|U\rangle$. This precision requires $O(1/\sqrt{\delta})$ applications of $W(P)$. Let us analyze this on some eigenvector $|w\rangle$ of $W(P)$, with corresponding eigenvalue $e^{\pm 2i\theta_j}$. Assume for simplicity that phase estimation gives (in the auxiliary second register) an estimate $\tilde{\theta}_j$ of θ_j that is within precision $\sqrt{\delta}/2$.[5] Because the nonzero θ_j are at least $\sqrt{2\delta}$, approximating them within $\sqrt{\delta}/2$ is good enough to determine whether the correct value θ_j itself is 0 or not. If $|\tilde{\theta}_j| > \sqrt{\delta}/2$, then $R(P)$ "infers" that $\theta_j \neq 0$ and puts a minus in front of the state. Finally, it reverses the phase estimation to set the auxiliary second register back to $|0\rangle$. In formulas, $R(P)$ maps

$$|w\rangle|0\rangle \overset{\text{PE}}{\mapsto} |w\rangle\left|\tilde{\theta}_j\right\rangle \mapsto (-1)^{\tilde{\theta}_j \neq 0}|w\rangle\left|\tilde{\theta}_j\right\rangle \overset{\text{PE}^{-1}}{\mapsto} (-1)^{[\tilde{\theta}_j \neq 0]}|w\rangle|0\rangle.$$

This has the desired effect: $R(P)$ maps $|U\rangle \mapsto |U\rangle$, and $|\psi\rangle \mapsto -|\psi\rangle$ for all $|\psi\rangle$ orthogonal to $|U\rangle$.

Now that we know how to implement the algorithm, let us look at its complexity. Consider the following setup, update, and checking costs:

- Setup cost $\boldsymbol{S}$: the cost of constructing $|U\rangle$
- Checking cost $\boldsymbol{C}$: the cost of the unitary map $|x\rangle|y\rangle \mapsto m_x|x\rangle|y\rangle$, where $m_x = -1$ if x is marked, and $m_x = 1$ otherwise
- Update cost $\boldsymbol{U}$: the cost of one step of the quantum walk, i.e., of $W(P)$

[5] Phase estimation will actually give a superposition over estimates $\tilde{\theta}_j$, with small but nonzero amplitudes on bad estimates, but we'll skip the technical details that are needed to deal with this.

The cost of part (a) of the algorithm is $\boldsymbol{C}$. Since $R(P)$ uses $O(1/\sqrt{\delta})$ applications of $W(P)$, and a few other gates, the cost of part (b) of the algorithm is essentially $O(\boldsymbol{U}/\sqrt{\delta})$. Ignoring constant factors, the total cost of the algorithm is then

$$\boldsymbol{S} + \frac{1}{\sqrt{\varepsilon}}\left(\boldsymbol{C} + \frac{1}{\sqrt{\delta}}\boldsymbol{U}\right). \tag{8.2}$$

Compare this with the classical cost of Eq. (8.1): quantum search square-roots both ε and δ.

8.3 Applications

There are a number of interesting quantum walk algorithms that beat the best classical algorithms. We'll give three examples here. More can be found in [117].

8.3.1 Grover search

Let us first derive a quantum algorithm for search. Suppose we have an N-bit string x of weight t, and we know $t/N \geq \varepsilon$. Consider the complete graph G on N vertices. Then the matrix P for the random walk on G has 0s on its diagonal, and its off-diagonal entries are all equal to $1/(N-1)$. This can be written as $P = \frac{1}{N-1}J - \frac{1}{N-1}I$, where J is the all-1 matrix and I is the identity. It is easy to see that $\lambda_1 = N/(N-1) - 1/(N-1) = 1$ (corresponding to the uniform vector) and all other eigenvalues are $-1/N$. Hence δ is very large here: $\delta = 1 - 1/N$. We'll mark a vertex i iff $x_i = 1$. Then, measuring cost by number of queries, a quantum walk on G will have $\boldsymbol{S} = \boldsymbol{U} = 0$ and $\boldsymbol{C} = 1$. Plugging this into Eq. (8.2), it will probably find a marked vertex in time $O(1/\sqrt{\varepsilon})$. The worst case is $\varepsilon = 1/N$, in which case we'll use $O(\sqrt{N})$ queries. Not surprisingly, we've essentially rederived Grover's algorithm.

8.3.2 Collision problem

Consider the following collision problem:

> Input: $x = x_0, \ldots, x_{n-1}$, where each x_i is an integer.[6]
> Goal: find distinct i and j such that $x_i = x_j$ if these exist, otherwise output "all elements are distinct."

The decision version of this problem (deciding if there exists at least one collision) is also known as *element distinctness*.

Consider the graph whose vertices correspond to the sets $R \subseteq \{0, \ldots, n-1\}$ of r elements. The total number of vertices is $N = \binom{n}{r}$. We'll put an edge between the vertices for R and R' iff these two sets differ in exactly two elements; in other words, you can get from R to R' by removing one element i from R and replacing it by a new element j. The resulting graph $J(n,r)$ is known as the *Johnson graph*. It is $r(n-r)$-regular, since every R has $r(n-r)$ different neighbors R'. Its spectral gap is known to be $\delta = \frac{n}{r(n-r)}$ [32, Sec. 12.3.2]; we won't prove that here, just note that if $r \ll n$, then $\delta \approx 1/r$. For each set R we also keep track of the corresponding sequence of x-values, $x_R = (x_i)_{i \in R}$. Hence the full "name" of a vertex is the pair (R, x_R).

[6] Say, all $x_i \leq n^2$ to avoid having to use too much space to store these numbers.

We'll call a vertex in $J(n,r)$ *marked* if it contains a collision, i.e., the corresponding set R contains distinct i,j such that $x_i = x_j$. In the worst case there is exactly one colliding pair i,j (more collisions only make the problem easier). The probability that i and j are both in a random r-set R, is $\varepsilon = \frac{r}{n}\frac{r-1}{n-1}$. Hence the fraction of marked vertices is at least $\varepsilon \approx (r/n)^2$.

We will now determine the setup, checking, and update costs. The setup cost (measured in terms of queries) is $\boldsymbol{S} = r+1$: we have to create a uniform superposition $|U\rangle$ over all edges R, R', and for each such basis state query all $r+1$ elements of $R \cup R'$ to add the information x_R and $x_{R'}$. Checking whether a given vertex R, x_R contains a collision doesn't take any queries because we already have x_R, hence $\boldsymbol{C} = 0$. To determine the update cost, note that mapping the second register of $|R, x_R\rangle|0\rangle$ to a superposition of all neighbors $R', x_{R'}$ requires querying (in superposition for all neighbors R') the value x_j of the element j that was added to get R'. Hence $\boldsymbol{U} = O(1)$. Plugging this into Eq. (8.2), the cost of a quantum walk algorithm for collision-finding is

$$\boldsymbol{S} + \frac{1}{\sqrt{\varepsilon}}\left(\boldsymbol{C} + \frac{1}{\sqrt{\delta}}\boldsymbol{U}\right) = O(r + n/\sqrt{r}).$$

This is $O(n^{2/3})$ if we set $r = n^{2/3}$. This $O(n^{2/3})$ turns out to be the optimal query complexity for the collision problem [2]. By some more work involving efficient data structures, the *time* complexity (= total number of elementary quantum gates) can be brought down to $n^{2/3}(\log n)^{O(1)}$ [7].

8.3.3 Finding a triangle in a graph

Consider the following triangle-finding problem:

> Input: the adjacency matrix of a graph H on n vertices.
> Goal: find vertices u, v, w that form a triangle (i.e., $(u,v), (v,w), (w,u)$ are all edges in the graph), if they exist.

We'll assume we have query access to the entries of the adjacency matrix of H, which tells us whether (u,v) is an edge or not. There are $\binom{n}{2}$ bits in this oracle, one for each potential edge of H. It is not hard to see that a classical algorithm needs $\Omega(n^2)$ queries before it can decide with good probability whether a graph contains a triangle or not. For example, take a bipartite graph consisting of 2 sets of $n/2$ vertices each, such that any pair of vertices from different sets is connected by an edge. Such a graph is triangle-free, but adding any one edge will create a triangle. A classical algorithm would have to query all those edges separately.

Let us try a quantum walk approach. Again consider the Johnson graph $J(n,r)$. Each vertex will correspond to a set $R \subseteq \{0,\ldots,n-1\}$ of r vertices, annotated with the result of querying all possible $\binom{r}{2}$ edges having both endpoints in R. We will call the vertex for set R *marked* if it contains *one edge* of a triangle. If there is at least one triangle in the graph, the fraction of marked vertices is at least $\varepsilon \approx (r/n)^2$. Getting a good upper bound for the checking cost $\boldsymbol{C}$ requires some work—namely Grover search plus another quantum walk!

The setup cost will be $\boldsymbol{S} = \binom{r}{2}$. The update cost will be $\boldsymbol{U} = 2r-2$, because if we remove one vertex i from R then we have to remove information about $r-1$ edges in H, and if we add a new j to R we have to query $r-1$ new edges in H.

Suppose we are given a set R of r vertices. How do we decide whether R contains an edge of a triangle? If we can efficiently decide, for a given u and R, whether R contains vertices v, w such that u, v, w form a triangle in H, then we could combine this with a Grover search over all n possible

vertices u of H. Given u and R, let us design a subroutine based on another quantum walk, this time on the Johnson graph $J(r, r^{2/3})$. Each vertex of this Johnson graph corresponds to a subset $R' \subseteq R$ of $r' = r^{2/3}$ vertices. Its spectral gap is $\delta' = r/r'(r-r') \approx 1/r^{2/3}$. We'll mark R' if it contains vertices v, w such that u, v, w form a triangle. If there is at least one triangle involving u and some $v, w \in R$, then the fraction of marked vertices R' in $J(r, r^{2/3})$ is at least $\varepsilon' \approx (r'/r)^2 = 1/r^{2/3}$. For this subroutine, the setup cost is $O(r^{2/3})$ (for each $v \in R$, query whether (u, v) is an edge in H); the update cost is $O(1)$ (if we replace v in R by w, then we need to "unquery" edge (u, v) and query edge (u, w)); and the checking cost is 0. Plugging this into Eq. (8.2), we can decide whether a fixed u forms a triangle with two vertices in R', using $O(r^{2/3})$ queries. Let's ignore the small error probability of the latter subroutine (it can be dealt with, but that's technical). Then we can combine it with Grover search over all n vertices u to get checking cost $\boldsymbol{C} = O(\sqrt{n}r^{2/3})$.

Plugging these $\boldsymbol{S}$, $\boldsymbol{U}$, and $\boldsymbol{C}$ into Eq. (8.2), the overall cost of a quantum walk algorithm for triangle-finding is

$$\boldsymbol{S} + \frac{1}{\sqrt{\varepsilon}}\left(\boldsymbol{C} + \frac{1}{\sqrt{\delta}}\boldsymbol{U}\right) = O\left(r^2 + \frac{n}{r}(\sqrt{n}r^{2/3} + r^{3/2})\right).$$

This is $O(n^{13/10})$ if we set $r = n^{3/5}$ [100]. The exponent 13/10 can be slightly improved further [18, 90, 80], and the current best exponent is 5/4 [89]. It is an open question what the optimal quantum query complexity for triangle-finding is; the best lower bound is only n. Also, the optimal quantum *time* complexity of this problem is still wide open.

Exercises

1. Let P be the projector on a d-dimensional subspace $V \subseteq \mathbb{R}^n$ that is spanned by orthonormal vectors $v_1, \ldots, v_d$. This means that $Pv = v$ for all $v \in V$, and $Pw = 0$ for all w that are orthogonal to V.

   (a) Show that P can be written in Dirac notation as $P = \sum_{i=1}^{d} |v_i\rangle\langle v_i|$.

   (b) Show that $R = 2P - I$ is a reflection through the subspace corresponding to P, i.e., $Rv = v$ for all v in the subspace, and $Rw = -w$ for all w that are orthogonal to the subspace.

2. Let G be a d-regular graph that is bipartite, so its vertex set $V = [N]$ can be partitioned into disjoint sets A and B, and all its edges are in $A \times B$. Give an eigenvector with eigenvalue 1 of the associated $N \times N$ normalized adjacency matrix P, and another eigenvector with eigenvalue -1.

3. This exercise is about obtaining a quantum algorithm for the collision problem with a slightly different quantum walk. Consider the problem of Section 8.3.2: we can query elements of the sequence of integers $x_0, \ldots, x_{n-1}$, and want to find distinct i and j such that $x_i = x_j$ (or report that there are no collisions). Again consider the Johnson graph $J(n, r)$, for some r to be optimized over later. Deviating from Section 8.3.2, now call a vertex R *marked* if there exist $i \in R$ and $j \in [n] \setminus R$ such that $x_i = x_j$. Show that we can find a marked vertex in this graph with high probability using $O(n^{2/3})$ queries to x. You may ignore small error probabilities, for example when using Grover's algorithm. Be explicit about what data you store about x at each vertex R.

4. (H) Let A, B, and C be $n \times n$ matrices with real entries. We'd like to decide whether or not $AB = C$. Of course, you could multiply A and B and compare the result with C, but matrix multiplication is expensive (the current best algorithm takes time roughly $O(n^{2.38})$).

   (a) Give a classical randomized algorithm that verifies whether $AB = C$ (with success probability at least $2/3$) using $O(n^2)$ steps, using the fact that matrix-vector multiplication can be done in $O(n^2)$ steps.

   (b) Show that if we have query-access to the entries of the matrices (i.e., oracles that map $i, j, 0 \mapsto i, j, A_{i,j}$ and similarly for B and C), then any classical algorithm with small error probability needs at least n^2 queries to detect a difference between AB and C.

   (c) Give a quantum walk algorithm that verifies whether $AB = C$ (with success probability at least $2/3$) using $O(n^{5/3})$ queries to matrix-entries.

5. A 3-SAT instance ϕ over n Boolean variables $x_1, \ldots, x_n$ is a formula which is the AND of a number of clauses, each of which is an OR of 3 variables or their negations. For example, $\phi(x_1, \ldots, x_4) = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_2 \vee x_3 \vee \overline{x_4})$ is a 3-SAT formula with 2 clauses. A satisfying assignment is a setting of the n variables such that $\phi(x_1, \ldots, x_n) = 1$ (i.e, TRUE). You may assume the number of clauses is at most some polynomial in n. In general it is NP-hard to find a satisfying assignment to such a formula. Brute force would try out all 2^n possible truth-assignments, but something better can be done by a classical random walk. Consider the following simple algorithm of Schöning [119], which is a classical random walk on the set of all $N = 2^n$ truth assignments:

   > Start with a uniformly random $x \in \{0,1\}^n$.
   > Repeat the following at most $3n$ times: if $\phi(x) = 1$ then STOP, else find the leftmost clause that is false, randomly choose one of its 3 variables and flip its value.

   One can show that this algorithm has probability at least $(3/4)^n$ of finding a satisfying assignment (if ϕ is satisfiable). You may assume this without proof.

   (a) Use the above to give a classical algorithm that finds a satisfying assignment with high probability in time $(4/3)^n \cdot p(n)$, where $p(n)$ is some polynomial factor (no need to use the C, U, S-framework of the chapter here; the answer is much simpler).

   (b) (H) Give a quantum algorithm that finds one (with high probability) in time $\sqrt{(4/3)^n} \cdot p(n)$.

Chapter 9

Hamiltonian Simulation

9.1 Hamiltonians

Thus far, we have viewed the dynamics of quantum systems from the perspective of *unitary transformations*: apart from measurement, the only way a quantum state (i.e., a vector of amplitudes) can change is by multiplication with a unitary matrix, for instance a two-qubit gate tensored with identities on the other qubits. But which unitary will actually occur in a given physical system? This is determined by the *Hamiltonian* of the system, which is the observable corresponding to the *total energy* in the system. Typically, this total energy is the sum of several different terms, corresponding to kinetic energy, potential energy, etc. Also typically, it is the sum of many *local* terms that each act on only a few of the particles (qubits) of the system, for example if all interactions are between pairs of particles.

One can think of the Hamiltonian H as describing the physical characteristics of the system. These do not determine the initial state $|\psi(0)\rangle$ of the system, but they do determine the *evolution* of the state in time, i.e., the state $|\psi(t)\rangle$ as a function of the time-parameter t, given initial state $|\psi(0)\rangle$. This is governed by the most important equation in quantum mechanics: the *Schrödinger equation*. It is a linear differential equation that relates the time-derivative of the current state to that state itself and to the Hamiltonian:

$$i\hbar \frac{d|\psi(t)\rangle}{dt} = H|\psi(t)\rangle.$$

Here $\hbar$ is a very small yet important physical constant: Planck's constant divided by 2π. We can set it to 1 by choosing appropriate units, and hence will ignore it from now on. In general H may itself change with t, but for simplicity we will only consider here the case where H is time-independent. Then, if we start in some state $|\psi(0)\rangle$, the solution to this differential equation is the following unitary evolution of the state:[1]

$$|\psi(t)\rangle = U|\psi(0)\rangle, \quad \text{where } U = e^{-iHt}.$$

So t time-steps of evolution induced by Hamiltonian H, corresponds to applying the unitary matrix e^{-iH} t times. Note, however, that t need not be integer here: this evolution is continuous in time, in contrast to the discrete picture one gets from the circuit model with elementary quantum gates.

[1] Applying a function, for instance $f(x) = e^{-ix}$, to a normal matrix means applying f to its eigenvalues: if A has diagonalization $V^{-1}DV$ then $f(A) = V^{-1}f(D)V$, where $f(D)$ is the diagonal matrix obtained by applying f to the diagonal entries of D. For example, if $A = \sum_j \lambda_j a_j a_j^T$ and $f(x) = e^{-ix}$, then $f(A) = \sum_j e^{-i\lambda_j} a_j a_j^T$. Note that if A is Hermitian, then e^{iA} is unitary.

In areas like quantum chemistry (i.e., the study of properties of molecules) and material sciences, it is often important to figure out how a quantum system will evolve from some given initial state, for instance a basis state.[2] This is typically hard to do on classical computers, since the number of parameters (amplitudes) is exponential in the number of particles. However, a *quantum* computer is like a universal quantum system, and should be able to efficiently simulate every efficient quantum process, in the same way that a classical universal Turing machine can efficiently simulate other (classical) physical processes.[3] In fact, this was the main reason why Feynman invented quantum computers: as a controllable quantum system that can be used to simulate other quantum systems. In order to realize that idea, we need methods to efficiently implement the unitary evolution that is induced by a given Hamiltonian. In other words, we need methods to implement $U = e^{-iHt}$ as a quantum circuit of gates (say, up to some small error ε in operator norm), and to apply this to a given initial state $|\psi\rangle$. This is known as the problem of "Hamiltonian simulation."

In this chapter we will cover several methods for Hamiltonian simulation. For simplicity we'll ignore the minus sign in Hamiltonian simulation, implementing $U = e^{iHt}$ rather than e^{-iHt}. We will also assume that our quantum system consists of n qubits. Some physical systems, for instance electron spins, naturally correspond to qubits. More complicated Hilbert spaces, for instance with basis states labeled by the positions (x, y, z coordinates) of all particles involved, can be encoded (approximately) in binary to reduce them to the case of qubits. This encoding can be done in many ways; much of the art in quantum chemistry is in how best to do this for specific systems, but we won't study that here (see [38]).

Word of warning: this chapter will be denser and more complicated than most of the other chapters in these notes. On the other hand, unlike those chapters it explains some very recent, cutting-edge results.

9.2 Method 1: Lie-Suzuki-Trotter methods

Note that an n-qubit Hamiltonian is a $2^n \times 2^n$ matrix, which is huge even for moderate n. Typically in Hamiltonian simulation we are dealing with very structured Hamiltonians that have a much shorter classical description. Suppose our Hamiltonian is of the form $H = \sum_{j=1}^{m} H_j$, where m is not too big (say, polynomial in n) and each H_j acts only on a few of the n qubits. For concreteness assume each H_j acts non-trivially on only *two* of the qubits.[4] Such a Hamiltonian is called *2-local*. Note that, for fixed t, the unitary $e^{iH_j t}$ is really just a 2-qubit gate, acting like identity on the other $n-2$ qubits; this 2-qubit gate could in turn be constructed from CNOTs and single-qubit gates.

[2] It is also very important in chemistry to be able to find out *global* properties of a given Hamiltonian like its lowest energy, a.k.a. *ground state energy*. Unfortunately that is a problem that seems to be hard to solve even for a quantum computer, even for the special case of 2-local Hamiltonians [85, 82].

[3] In Chapter 12 we will see that it is actually possible to classically simulate quantum computers (and hence quantum systems more generally) with a polynomial amount of *space*, but our best methods still use an exponential amount of *time*. If factoring a large integer is a hard problem for classical computers (which is widely believed), then Shor's efficient quantum factoring algorithm (Chapter 5) implies that it is *impossible* to simulate a quantum computer in polynomial time on a classical computer.

[4] This means H can be described efficiently by m 4×4 matrices, rather than by a $2^n \times 2^n$ matrix. A different assumption that is often made on Hamiltonians and that we will see later, is that H is *s-sparse*, meaning each of the 2^n columns has at most s nonzero entries, and we have some efficient "sparse access" to these nonzero entries. Note that if $H = \sum_j H_j$ and each H_j acts on only 2 qubits, then H is $4m$-sparse. Thus, roughly speaking, the locality assumption implies the sparsity assumption.

Our goal is to implement $U = e^{iHt} = e^{i\sum_j H_j t}$. It is now tempting to view this exponential of a sum of matrices as a product $\prod_{j=1}^m e^{iH_j t}$, which is just a product of m 2-qubit gates. If all terms H_j are diagonal, or if there is some basis in which all terms are diagonal (equivalently, if all H_j commute), then this indeed works out. However, in general matrix exponentials do not work that way: e^{A+B} need not equal $e^A e^B$ if A and B do not commute (see Exercise 1). The Lie-Suzuki-Trotter decomposition gives us a way to handle this. It uses the fact that if A and B have small operator norm, then e^{A+B} and $e^A e^B$ are *approximately* equal: $e^{A+B} = e^A e^B + E$, where the error-term E is a matrix whose operator norm is $O(\|AB\|)$.[5]

How can we use this to approximate U by a circuit $\widetilde{U}$ of 2-qubit gates? Assume H, as well as each of the terms H_j, has operator norm ≤ 1 (see Exercise 2 for why such normalization matters). First consider the simple case $m = 2$, so $H = H_1 + H_2$. We can now implement $U = e^{iHt}$ by doing a little bit of H_1, a little bit of H_2, a little bit of H_1, etc. More precisely, for every integer $r \geq 1$ of our choice, we have

$$U = e^{iHt} = (e^{iHt/r})^r = (e^{iH_1 t/r + iH_2 t/r})^r = (e^{iH_1 t/r} e^{iH_2 t/r} + E)^r.$$

Here the error-term E has norm $\|E\| = O(\|iH_1 t/r \cdot iH_2 t/r\|) = O(\|H_1 H_2\| t^2/r^2)$. Our approximating circuit will be $\widetilde{U} = (e^{iH_1 t/r} e^{iH_2 t/r})^r$. Since errors in a product of unitaries add at most linearly (see Exercise 4.4), we have approximation error $\left\|U - \widetilde{U}\right\| \leq r\|E\| = O(\|H_1 H_2\| t^2/r) = O(t^2/r)$. Choosing $r = O(t^2/\varepsilon)$, we can make this error $\leq \varepsilon$. The circuit $\widetilde{U}$ uses $2r = O(t^2/\varepsilon)$ 2-qubit gates.

The same idea works for the general case where we have $m > 2$ Hamiltonian terms:

$$U = e^{iHt} = (e^{iHt/r})^r = (e^{iH_1 t/r + \cdots + iH_m t/r})^r = (e^{iH_1 t/r} \cdots e^{iH_m t/r} + E)^r,$$

where $\|E\| = O(\|H\|^2 t^2/r^2) = O(t^2/r^2)$. Choosing $r = O(t^2/\varepsilon)$, we have an approximating circuit $\widetilde{U} = (e^{iH_1 t/r} \cdots e^{iH_m t/r})^r$ with $mr = O(mt^2/\varepsilon)$ 2-qubit gates, and error $\left\|U - \widetilde{U}\right\| \leq \varepsilon$.

This is the *first-order* Lie-Suzuki-Trotter approach to Hamiltonian simulation, due to Lloyd [93]. Its gate-complexity depends quadratically on the time t for which we want to simulate the evolution, which is not optimal. One can do fancier higher-order decompositions that make the dependence on t nearly linear, but we won't explain those here. The dependence on ε is polynomial, which can be improved as well.

9.3 Method 2: Linear combination of unitaries (LCU)

Here we will describe a method for Hamiltonian simulation whose complexity depends linearly on the time t for which we want to evolve the state, and only logarithmically on the desired error ε.

Let's start with a more general problem. Suppose we have a $2^n \times 2^n$ matrix M and an n-qubit state $|\psi\rangle$, and we would like to prepare the state $M|\psi\rangle/\|M|\psi\rangle\|$. Here M need not be unitary, but suppose we can write M as a linear combination of unitaries:[6]

$$M = \sum_{j=1}^m \alpha_j V_j,$$

[5] A non-rigorous but reasonably convincing way to see this is to approximate term e^M by its first-order Taylor series $I + M$, which is a good approximation if M has small norm. Then $e^A e^B - e^{A+B} \approx (I+A)(I+B) - (I+A+B) = AB$.

[6] In fact *every* M can be written in such a way, because the 4^n n-qubit Pauli matrices (each of which is unitary) form a basis for the linear space of all $2^n \times 2^n$ matrices. See Appendix A.7.

with the α_j being nonnegative reals (we can always absorb complex phases into the V_j). Let $\|\alpha\|_1 = \sum_j \alpha_j$, and let W be a unitary acting on $\lceil \log m \rceil$ qubits that maps $|0\rangle \mapsto \frac{1}{\sqrt{\|\alpha\|_1}} \sum_j \sqrt{\alpha_j}|j\rangle$. Suppose each V_j is an "easy" unitary, for instance a 2-qubit gate tensored with identity on the other $n-2$ qubits, or a small circuit. Also suppose we can implement these unitaries in a controlled way: we have access to a 2-register unitary $V = \sum_{j=1}^m |j\rangle\langle j| \otimes V_j$. This maps $|j\rangle|\phi\rangle \mapsto |j\rangle V_j|\phi\rangle$, and we can think of the first register as "selecting" which unitary V_j to apply to the second register.[7]

We want to use V and W to implement M on a given state $|\psi\rangle$. Consider the following algorithm:

1. Start with two-register state $|0\rangle|\psi\rangle$, where the first register has $\lceil \log m \rceil$ qubits.
2. Apply W to the first register.
3. Apply V to the whole state.
4. Apply W^{-1} to the first register.

A small calculation (see Exercise 4) shows that the resulting state can be written as

$$\frac{1}{\|\alpha\|_1}|0\rangle M|\psi\rangle + \sqrt{1 - \frac{1}{\|\alpha\|_1^2}}|\phi\rangle, \tag{9.1}$$

where $|\phi\rangle$ is some other state that we don't care about, but that has no support on basis states starting with $|0\rangle$. If we were to measure the first register, the probability of outcome 0 is $p = \|M|\psi\rangle\|^2/\|\alpha\|_1^2$. In case of that measurement outcome, the second register would collapse to the normalized version of $M|\psi\rangle$, as desired. The success probability p may be small, but we could use $O(1/\sqrt{p}) = O(\|\alpha\|_1/\|M|\psi\rangle\|)$ rounds of amplitude amplification to amplify the part of the state that starts with $|0\rangle$. Thus we would prepare (the normalized version of) $M|\psi\rangle$ in the second register. Unfortunately this usage of amplitude amplification assumes the ability to implement a unitary (as well as its inverse) to prepare $|\psi\rangle$ from a known initial state, say $|0\rangle$. Regular amplitude amplification won't work if we just have one copy of the state $|\psi\rangle$ available, which is the typical situation for instance in Hamiltonian simulation. However, Exercise 6 gives us a variant called *oblivious* amplitude amplification, which circumvents this problem: it works even with just one copy of $|\psi\rangle$, as long as M is proportional to a unitary (or close to that).

9.3.1 Hamiltonian simulation via LCU

Recall that our goal is to efficiently implement the unitary e^{iHt} that is induced by a given Hamiltonian H, normalized so that $\|H\| \leq 1$. The following approach is due to Berry et al. [24, 25, 26]. Suppose, somewhat paradoxically, that we can write out the *Hermitian* matrix H as a linear combination of *unitaries*: $H = \sum_j \alpha_j V_j$. For example, if H is the sum of m 2-local terms like before, then every 2-local term can be written as the sum of at most 16 n-qubit Pauli matrices (each of which is unitary and acts non-trivially on only two qubits). Thus we would decompose H as a sum of at most $16m$ unitaries, each acting non-trivially on only two of the n qubits. The sum of coefficients $\|\alpha\|_1$ will be $O(m)$.

[7] In the literature, this V is often called "select-V." One might expect the cost of V to be not much higher than the costliest V_j, just like the cost of a classical "if A then B, else C" statement is not much bigger than the largest of the costs of B and C. However, if we measure circuit size, then the cost of V could be roughly the *sum* of the costs of the V_js because circuits for each V_j should be "included" in the circuit for V.

Using the Taylor series $e^x = \sum_{k=0}^{\infty} x^k/k!$, we can write the unitary we want to implement exactly as

$$e^{iHt} = \sum_{k=0}^{\infty} \frac{(iHt)^k}{k!} = \sum_{k=0}^{\infty} \frac{(it)^k}{k!} \left(\sum_{j\in[m]} \alpha_j V_j \right)^k = \sum_{k=0}^{\infty} \frac{(it)^k}{k!} \sum_{j_1,\ldots,j_k\in[m]} \alpha_{j_1} \cdots \alpha_{j_k} V_{j_1} \cdots V_{j_k}. \tag{9.2}$$

Note that if each V_j is easy to implement and k is not too big, then the unitary $V_{j_1} \cdots V_{j_k}$ is also not too hard to implement. Exercise 7 shows that if we truncate the Taylor series at $k = O(t + \log(1/\varepsilon))$, dropping the terms of higher order, then the induced error (i.e., the dropped part) has operator norm at most ε. Accordingly, we can take the part of the right-hand side of Eq. (9.2) for $k = O(t + \log(1/\varepsilon))$ and then use the linear combination of unitaries approach to approximately implement e^{iHt}. The unitaries in this decomposition are of the form $V_{j_1,\ldots,j_k} = i^k V_{j_1} \cdots V_{j_k}$; let $\mathcal{V} = \sum_{j_1,\ldots,j_k} |j_1,\ldots,j_k\rangle\langle j_1,\ldots,j_k| \otimes V_{j_1,\ldots,j_k}$ denote the controlled operation of the $V_{j_1,\ldots,j_k}$ unitaries, each of which involves k V_j's. The corresponding nonnegative coefficients in this decomposition are

$$\beta_{j_1,\ldots,j_k} = \frac{t^k}{k!} \alpha_{j_1} \cdots \alpha_{j_k}, \qquad \text{for } k \leq O(t + \log(1/\varepsilon)).$$

These β-coefficients add up to

$$\|\beta\|_1 = \sum_{k=0}^{O(t+\log(1/\varepsilon))} \frac{t^k}{k!} \alpha_{j_1} \cdots \alpha_{j_k} \leq \sum_{k=0}^{\infty} \frac{t^k}{k!} \alpha_{j_1} \cdots \alpha_{j_k} = \sum_{k=0}^{\infty} \frac{(t\|\alpha\|_1)^k}{k!} = e^{t\|\alpha\|_1},$$

so straightforward application of the LCU method with oblivious amplitude amplification uses $O(\|\beta\|_1) = O(e^{t\|\alpha_1\|})$ applications of $\mathcal{V}$ and $\mathcal{V}^{-1}$.

The logarithmic error-dependence of the complexity of the above method is excellent. The exponential dependence on $t\|\alpha_1\|$ is quite terrible for large t, but not too bad for very small t. So what we'll do if we want to do a simulation for large t, is to divide that t into $b = t\|\alpha\|_1$ blocks of time $\tau = 1/\|\alpha\|_1$ each, run the above algorithm for time τ with error $\varepsilon' = \varepsilon/b$, and then glue b time-τ simulations together. This will simulate $(e^{iH\tau})^b = e^{iHt}$, with error $\leq b\varepsilon' = \varepsilon$. The cost of each time-$\tau$ simulation is $O(e^{\tau\|\alpha\|_1}) = O(1)$ applications of $\mathcal{V}$ and $\mathcal{V}^{-1}$, each of which involves $O(\tau + \log(1/\varepsilon')) = O(\log(t\|\alpha\|_1/\varepsilon))$ applications of the V_j's. The overall cost will be b times that, since we'll run b subsequent time-τ simulations in order to implement a time-t simulation.

To give a more concrete example, consider again the special case where $H = \sum_i H_i$ consists of 2-local terms, so the unitaries V_j in the induced linear combination of unitaries $H = \sum_{j=1}^m \alpha_j V_j$ only act nontrivially on 2 qubits each. Then we approximate the time-τ unitary $e^{iH\tau}$ by a linear combination of unitaries

$$M = \sum_{k=0}^{O(\tau+\log(1/\varepsilon'))} \sum_{j_1,\ldots,j_k\in[m]} \beta_{j_1,\ldots,j_k} V_{j_1,\ldots,j_k}, \tag{9.3}$$

where each $V_{j_1,\ldots,j_k}$ is a product of $k = O(\tau + \log(1/\varepsilon')) = O(\log(t\|\alpha\|_1/\varepsilon))$ 2-qubit gates. We can implement this using the linear combination of unitaries approach, and repeat this $b = t\|\alpha\|_1$ times. The cost of the unitary W is typically relatively small (see Exercise 5), so we can ε-approximate the unitary e^{iHt} using a circuit of roughly $O(t\|\alpha\|_1 \log(t\|\alpha\|_1/\varepsilon)) = O(mt \log(mt/\varepsilon))$ applications of $\mathcal{V}$ and $\mathcal{V}^{-1}$, and slightly more other 2-qubit gates. Note the linear dependence of the cost on the evolution-time t, and the logarithmic dependence on the error ε, which is much better than Lie-Suzuki-Trotter methods.

9.4 Method 3: Transforming block-encoded matrices

In this section we'll describe a recent approach that is very general and flexible. Suppose A is an n-qubit matrix with operator norm $\|A\| \leq 1$, and we know how to implement an $(n+1)$-qubit unitary

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix}. \tag{9.4}$$

The '$\cdot$'s are unspecified $2^n \times 2^n$-dimensional matrices, the only constraint on which is that U is unitary. Such a U is called a unitary *block-encoding* of A. Note that

$$U : |0\rangle|\psi\rangle \mapsto |0\rangle A|\psi\rangle + |1\rangle|\phi\rangle,$$

where we can't say much about the (subnormalized) state $|\phi\rangle$. Written more technically, the defining property of such a block-encoding is $(\langle 0| \otimes I)U(|0\rangle \otimes I) = A$, where the first register is one qubit. More generally we can define an *a-qubit* block-encoding of A, which is an $(a+n)$-qubit unitary U with the property that $(\langle 0^a| \otimes I)U(|0^a\rangle \otimes I) = A$.

Example 1: LCU does block-encoding. From Eq. (9.1) we can see that LCU (without the final amplitude amplifcation) implements a $\lceil \log m \rceil$-qubit block-encoding of the matrix $A = M/\|\alpha\|_1$.

Example 2: Block-encoding a sparse Hermitian matrix. Let A be a $2^n \times 2^n$ Hermitian matrix of operator norm $\|A\| \leq 1$ that is s-sparse, so each row and column of A have at most s nonzero entries (for simplicity assume *exactly* s nonzero entries). Since this matrix A is still an exponentially large object, we have to be careful how we can access such sparse matrices. First, we assume we can query the entries of A in the usual way: we have an oracle

$$O_A : |i,j\rangle|0\rangle \mapsto |i,j\rangle|A_{ij}\rangle,$$

where we assume the last register has sufficiently many qubits to write down the complex entry A_{ij} either exactly or with sufficient precision. Of course, since A is sparse, A_{ij} will actually be 0 for most (i,j). Let $\nu(j,\ell) \in \{0,\ldots,N-1\}$ denote the location of the ℓth nonzero entry of the jth column of A; so the s nonzero entries in the jth column are at positions $\nu(j,1),\ldots,\nu(j,s)$. We also assume we have another oracle that allows us to find these locations:

$$O_{A,loc} : |j,\ell\rangle \mapsto |j,\nu(j,\ell)\rangle.$$

We also assume we can run O_A^{-1} and $O_{A,loc}^{-1}$. Together these assumption are called having "sparse access" to A.

We will now show how to implement a block-encoding of the matrix A/s. Exercise 8 shows how we can implement two $(2n+1)$-qubit unitaries that create superpositions over the locations of the nonzero entries in the jth column and ith row of A, respectively:

$$W_1 : |0\rangle|0^n\rangle|j\rangle \mapsto \frac{1}{\sqrt{s}}|0\rangle \sum_{k:A_{kj}\neq 0} |k,j\rangle, \qquad W_3 : |0\rangle|0^n\rangle|i\rangle \mapsto \frac{1}{\sqrt{s}} \sum_{\ell:A_{i\ell}\neq 0} |0\rangle|i,\ell\rangle,$$

using one $O_{A,loc}$-query and a few other A-independent gates. We can also implement the following unitary using one query to each of O_A and O_A^{-1}, and a few other A-independent gates (and some auxiliary qubits that start and end in $|0\rangle$):

$$W_2 : |0\rangle|k,j\rangle \mapsto A_{kj}|0\rangle|k,j\rangle + \sqrt{1-|A_{kj}|^2}|1\rangle|k,j\rangle.$$

By going through the action on initial state $|0^{n+1}j\rangle$ step-by-step (see Exercise 8), one can show that the $(0^{n+1}i, 0^{n+1}j)$-entry of $U = W_3^{-1}W_2W_1$ is exactly A_{ij}/s. In other words, U is an $(n+a)$-qubit block-encoding of the matrix A/s for some a (this depends on how many ancilla qubits are actually used).

How can we use a given block-encoding U of A? Suppose that for some function $f : \mathbb{R} \to \mathbb{R}$ we want to implement a unitary V that looks like

$$V = \begin{pmatrix} f(A) & \cdot \\ \cdot & \cdot \end{pmatrix},$$

using a small number of applications of the block-encoding of A. Here we don't care what submatrices sit at the '$\cdot$' entries of U or V, as long as the upper-left block of V is $f(A)$ and V as a whole is unitary.

For example, in Hamiltonian simulation A would be the Hamiltonian H and $f(x)$ would be e^{ixt}, so that we are effectively implementing $f(H) = e^{iHt}$, as is the goal in Hamiltonian simulation. In the HHL algorithm in the next chapter, $f(x)$ will be $1/x$, so that we effectively implement A^{-1}.

It turns out that we can implement a good approximation of V efficiently if we have a *low-degree polynomial* P approximating f. The idea is that we can let P act on the eigenvalues of A, thus transforming a block-encoding of A into one of $P(A)$. We state without proof the following theorem by Gilyén et al. [65, follows from Theorem 59], which extends work of Low et al. [96, 97, 95, 98].

Theorem 1 *Let $P : [-1,1] \to \{c \in \mathbb{C} \mid |c| \leq 1/4\}$ be a degree-d polynomial, and let U be a unitary a-qubit block-encoding of Hermitian matrix A. We can implement a unitary $O(a)$-qubit block-encoding V of $P(A)$ using d applications of U and U^{-1}, one controlled application of U, and $O(ad)$ other 2-qubit gates.*

This theorem can be generalized to a powerful technique called "singular-value transformation" [65], where A can be an arbitrary matrix, non-Hermitian and even non-square.

9.4.1 Hamiltonian simulation via transforming block-encoded matrices

Let's see how we can use Theorem 1 for Hamiltonian simulation for a given sparse Hamiltonian H. We again approximate the function $f(x) = e^{ixt}$ using a degree-$d = O(t + \log(1/\varepsilon))$ polynomial P which is the first d terms of the Taylor series of f (see Exercise 7), divided by 4 to ensure that its range satisfies the condition of Theorem 1. If H is s-sparse and we have sparse access to it, then Example 2 of Section 9.4 shows how to efficiently implement a block-encoding U of the scaled Hamiltonian H/s, using $O(1)$ queries to H and $O(n)$ other gates. Note that evolving Hamiltonian H for time t is the same as evolving H/s for time st. Theorem 1 now gives us a block-encoding V of $P(H) \approx \frac{1}{4}e^{iHt}$. This V invokes U and U^{-1} $O(st + \log(1/\varepsilon))$ times, and maps:

$$V : |0\rangle|\psi\rangle \mapsto |0\rangle P(H)|\psi\rangle + |\phi\rangle,$$

where $|\phi\rangle$ has no support on basis states starting with $|0\rangle$. Since $P(H) \approx \frac{1}{4}e^{iHt}$ is essentially proportional to a unitary, we can now apply $O(1)$ rounds of oblivious amplitude amplification to boost the factor $\frac{1}{4}$ to essentially 1, using only one copy of $|\psi\rangle$.

This implements the desired unitary e^{iHt} on one copy of $|\psi\rangle$, up to small error. The complexity of ε-precise Hamiltonian simulation of an s-sparse Hamiltonian H of operator norm ≤ 1, then becomes $O(st + \log(1/\varepsilon))$ queries to H and $O(n(st + \log(1/\varepsilon)))$ 2-qubit gates.

Exercises

1. Compute the following five 2×2 unitaries: e^{iX}, e^{iZ}, $e^{iX}e^{iZ}$, $e^{iZ}e^{iX}$, and $e^{i(X+Z)}$. Here X and Z are the usual Pauli matrices.

2. Suppose we want to implement a certain unitary U, and we can do that by switching on a Hamiltonian H for some time t: $U = e^{-iHt}$. Now suppose H' is another Hamiltonian, with 100 times as much energy as H: $H' = 100H$. Show that using H' we can implement U a 100 times faster than with H.
   NB: This exercise shows why some kind of normalization of the Hamiltonian is needed if we want to talk about the time it takes to implement something. We can always "speed up" a computation by a factor k if we can multiply our Hamiltonian with a factor k.

3. Consider the simple case of the linear-combination-of-unitaries trick where $m = 2$ and $M = U_1 + U_2$. Describe the unitaries V and W, and track the initial state $|0\rangle|\psi\rangle$ through the 4-step algorithm in Section 9.3.

4. (H) Give a calculation to justify that the 4-step algorithm in Section 9.3 indeed produces a state of the form of Eq. (9.1).

5. Let $v \in [-1,1]^N$ be a vector with real entries, of dimension $N = 2^n$, indexed by $i \in \{0,1\}^n$. Suppose we can query the entries of this vector by a unitary that maps
$$O_v : |i\rangle|0^p\rangle \mapsto |i\rangle|v_i\rangle,$$
so where the binary representation of the ith entry of v is written into the second register. We assume this second register has p qubits, and the numbers v_i can all be written exactly with p bits of precision (it doesn't matter how, but for concreteness say that the first bit indicates the sign of the number, followed by the $p-1$ most significant bits after the decimal dot). Our goal is to prepare the n-qubit quantum state
$$|\psi\rangle = \frac{1}{\|v\|} \sum_{i \in \{0,1\}^n} v_i |i\rangle.$$

   (a) Show how you can implement the following 3-register map (where the third register is one qubit) using one application of O_v and one of O_v^{-1}, and some v-independent unitaries (you don't need to draw detailed circuits for these unitaries, nor worry about how to write those in terms of elementary gates).
   $$|i\rangle|0^p\rangle|0\rangle \mapsto |i\rangle|0^p\rangle(v_i|0\rangle + \sqrt{1-v_i^2}|1\rangle).$$

   (b) Suppose you apply the map of (a) to a uniform superposition over all $i \in \{0,1\}^n$. Write the resulting state, and calculate the probability that measuring the last qubit in the computational basis gives outcome 0.

   (c) What is the resulting 3-register state if the previous measurement gave outcome 0?

   (d) Assume you know $\|v\|$ exactly. Give an algorithm that prepares $|\psi\rangle$ exactly, using $O\left(\frac{\sqrt{N}}{\|v\|}\right)$ applications of O_v and O_v^{-1}, and some v-independent unitaries.

6. (H) This exercise explains *oblivious* amplitude amplification.
Let M be an n-qubit unitary and U be an $(a+n)$-qubit unitary, such that for all n-qubit $|\psi\rangle$:

$$U|0^a\rangle|\psi\rangle = \sin(\theta)|0^a\rangle M|\psi\rangle + \cos(\theta)\left|\Phi^\perp\right\rangle,$$

where θ is some angle that's independent of $|\psi\rangle$, while $\left|\Phi^\perp\right\rangle$ depends on $|\psi\rangle$ and has no support on basis states starting with 0^a. We start from $|\Psi\rangle = |0^a\rangle|\psi\rangle$ and our goal is to prepare $|\Phi\rangle = |0^a\rangle M|\psi\rangle$. If θ is close to $\pi/2$, then we can just apply U and measure the first register; we'll see 0^a with probability $\sin(\theta)^2 \approx 1$ and in that case end up with the desired state $|\Phi\rangle$. But suppose θ is quite small. Here we will see how we can amplify the angle θ to roughly $\pi/2$, without assuming a unitary to prepare $|\Psi\rangle$.

   (a) Let $\mathcal{S}$ be the 2-dimensional space spanned by $|\Phi\rangle$ and $\left|\Phi^\perp\right\rangle$. Let $R = (2|0^a\rangle\langle 0^a| - I) \otimes I$ be a unitary that puts a '$-$' in front of every basis state that doesn't start with 0^a. Show that R, restricted to $\mathcal{S}$, is a reflection through $|\Phi\rangle$.

   (b) Define $\left|\Psi^\perp\right\rangle = U^{-1}\left(\cos(\theta)|\Phi\rangle - \sin(\theta)\left|\Phi^\perp\right\rangle\right)$. Show $U|\Psi\rangle$ and $U\left|\Psi^\perp\right\rangle$ are orthogonal.
   One can also show with a bit more work [24, Lemma 3.7] that $\left|\Psi^\perp\right\rangle$ has no support on basis states starting with 0^a. You may assume this fact without proof in the remainder of this exercise.

   (c) Show that URU^{-1}, restricted to $\mathcal{S}$, is a reflection through $U|\Psi\rangle$.

   (d) Show that $(URU^{-1}R)^k U|0^a\rangle|\psi\rangle = \sin((2k+1)\theta)|\Phi\rangle + \cos((2k+1)\theta)\left|\Phi^\perp\right\rangle$.

   (e) How large should we take k in order to end up with (approximately) the state $|\Phi\rangle$?
   NB: If you know θ exactly, then you can even exactly prepare $|\Phi\rangle$ (along the lines of Exercise 7.7) but you don't need to show that.

7. (H) Show that you can choose a sufficiently large constant c such for all Hermitian H with operator norm $\|H\| \leq 1$, we have

$$\left\| e^{iHt} - \sum_{k=0}^{c(t+\log(1/\varepsilon))-1} \frac{(iHt)^k}{k!} \right\| = \left\| \sum_{k=c(t+\log(1/\varepsilon))}^{\infty} \frac{(iHt)^k}{k!} \right\| \leq \varepsilon.$$

8. This exercise looks at the details of block-encoding an s-sparse matrix A with $\|A\| \leq 1$ from Section 9.4. Consider the various unitaries defined there.

   (a) Show how to implement W_1 using an $O_{A,loc}$-query and a few other A-independent gates. (Note that the same method allows to implement W_3.)

   (b) Show how to implement W_2 using an O_A-query, an O_A^{-1}-query, and a few other A-independent gates (you may use auxiliary qubits as long as those start and end in $|0\rangle$).

   (c) Show that the $(0^{n+1}i, 0^{n+1}j)$-entry of $W_3^{-1}W_1$ is exactly $1/s$ if $A_{ij} \neq 0$, and is 0 if $A_{ij} = 0$.

   (d) Show that the $(0^{n+1}i, 0^{n+1}j)$-entry of $W_3^{-1}W_2W_1$ is exactly A_{ij}/s.

Chapter 10

The HHL Algorithm

10.1 The linear-systems problem

In this chapter we present the Harrow-Hassidim-Lloyd (HHL [74]) algorithm for solving large systems of linear equations. Such a system is given by an $N \times N$ matrix A with real or complex entries, and an N-dimensional nonzero vector b. Assume for simplicity that $N = 2^n$. The linear-system problem is

LSP: find an N-dimensional vector x such that $Ax = b$.

Solving large systems of linear equations is extremely important in many computational problems in industry, in science, in optimization, in machine learning, etc. In many applications it suffices to find a vector $\tilde{x}$ that is close to the actual solution x.

We will assume A is invertible (equivalently, has rank N) in order to guarantee the existence of a unique solution vector x, which is then just $A^{-1}b$. This assumption is just for simplicity: if A does not have full rank, then the methods below would still allow to invert it on its support, replacing A^{-1} by the "Moore-Penrose pseudoinverse."

The HHL algorithm can solve "well-behaved" large linear systems very fast (under certain assumptions), but in a weak sense: instead of outputting the solution vector x, its goal is to output the n-qubit state

$$|x\rangle := \frac{1}{\|x\|} \sum_{i=0}^{N-1} x_i |i\rangle,$$

or some other n-qubit state close to $|x\rangle$. This is called the *quantum* linear-system problem:

QLSP: find an n-qubit state $|\tilde{x}\rangle$ such that $\||x\rangle - |\tilde{x}\rangle\| \leq \varepsilon$ and $Ax = b$.

Note that the QLSP is an inherently quantum problem, since the goal is to produce an n-qubit state whose amplitude-vector (up to normalization and up to ε-error) is a solution to the linear system. In general this is not as useful as just having the N-dimensional vector x written out on a piece of paper, but in some cases where we only want some partial information about x, it may suffice to just (approximately) construct $|x\rangle$.

We will assume without loss of generality that A is Hermitian (see Exercise 1). Let us state the more restrictive assumptions that will make the linear system "well-behaved" and suitable for the HHL algorithm:

1. We have a unitary that can prepare the vector b as an n-qubit quantum state $|b\rangle = \frac{1}{\|b\|}\sum_i b_i|i\rangle$ using a circuit of B 2-qubit gates. We also assume for simplicity that $\|b\| = 1$.

2. The matrix A is *s-sparse* and we have sparse access to it, like in Section 9.4. Such sparsity is not essential to the algorithm, and could be replaced by other properties that enable an efficient block-encoding of A.

3. The matrix A is *well-conditioned*: the ratio between its largest and smallest singular value is at most some κ.[1] For simplicity, assume the smallest singular value is $\geq 1/\kappa$ while the largest is ≤ 1. In other words, all eigenvalues of A lie in the interval $[-1,-1/\kappa] \cup [1/\kappa, 1]$. The smaller the "condition number" κ is, the better it will be for the algorithm. Let's assume our algorithm knows κ, or at least knows a reasonable upper bound on κ.

10.2 The basic HHL algorithm for linear systems

Let us start with some intuition. The solution vector x that we are looking for is $A^{-1}b$, so we would like to apply A^{-1} to b. If A has spectral decomposition $A = \sum_{j=0}^{N-1} \lambda_j a_j a_j^T$, then the map A^{-1} is the same as the map $a_j \mapsto \frac{1}{\lambda_j} a_j$: we just want to multiply the eigenvector a_j with the scalar $1/\lambda_j$. The vector b can also be written as a linear combination of the eigenvectors a_j: $b = \sum_j \beta_j a_j$ (we don't need to know the coefficients β_j for what follows). We want to apply A^{-1} to b to obtain $A^{-1}b = \sum_j \beta_j \frac{1}{\lambda_j} a_j$, normalized, as an n-qubit quantum state.

Unfortunately the maps A and A^{-1} are not unitary (unless $|\lambda_j| = 1$ for all j), so we cannot just apply A^{-1} as a quantum operation to state $|b\rangle$ to get state $|x\rangle$. Fortunately $U = e^{iA} = \sum_j e^{i\lambda_j} a_j a_j^T$ *is* unitary, and has the same eigenvectors as A and A^{-1}. We can implement U and powers of U by Hamiltonian simulation, and then use phase estimation (Section 4.6) to estimate the λ_j associated with eigenvector $|a_j\rangle$ with some small approximation error (for this sketch, assume for simplicity that the error is 0). Conditioned on our estimate of λ_j we can then rotate an auxiliary $|0\rangle$-qubit to $\frac{1}{\kappa\lambda_j}|0\rangle + \sqrt{1 - \frac{1}{(\kappa\lambda_j)^2}}|1\rangle$ (this is a valid state because $|\kappa\lambda_j| \geq 1$). Next we undo the phase estimation to set the register that contained the estimate back to $|0\rangle$. Suppressing the auxiliary qubits containing the temporary results of the phase estimation, we have now unitarily mapped

$$|a_j\rangle|0\rangle \mapsto |a_j\rangle \left(\frac{1}{\kappa\lambda_j}|0\rangle + \sqrt{1 - \frac{1}{(\kappa\lambda_j)^2}}|1\rangle \right).$$

If we prepare a copy of $|b\rangle|0\rangle = \sum_j \beta_j |a_j\rangle|0\rangle$ and apply the above unitary map to it, then we obtain

$$\sum_j \beta_j |a_j\rangle \left(\frac{1}{\kappa\lambda_j}|0\rangle + \sqrt{1 - \frac{1}{(\kappa\lambda_j)^2}}|1\rangle \right) = \frac{1}{\kappa} \underbrace{\sum_j \beta_j \frac{1}{\lambda_j} |a_j\rangle}_{\propto |x\rangle} |0\rangle + |\phi\rangle|1\rangle,$$

[1] Note that the assumption that A is invertible is equivalent to assuming $\kappa < \infty$. We can think of the stronger assumption that κ is small, as the assumption that A is invertible in a stable or robust way, so that small errors either in b or in our computational steps don't lead to massive errors in the solution vector x.

where we don't care about the (subnormalized) state $|\phi\rangle$. Note that because $\sum_j |\beta_j/\lambda_j|^2 \geq \sum_j |\beta_j|^2 = 1$, the norm of the part of the state ending in qubit $|0\rangle$ is at least $1/\kappa$. Accordingly, we can now apply $O(\kappa)$ rounds of amplitude amplification to amplify this part of the state to have amplitude essentially 1. This prepares state $|x\rangle$, as intended.

This rough sketch (which Exercise 2 asks you to make more precise) is the basic idea of HHL. It leads to an algorithm that produces a state $|\tilde{x}\rangle$ that is ε-close to $|x\rangle$, using roughly $\kappa^2 s/\varepsilon$ queries to H and roughly $\kappa s(\kappa n/\varepsilon + B)$ other 2-qubit gates.

10.3 Improving the complexity of the HHL agorithm

The complexity of the above basic HHL algorithm can be improved further. Gilyén et al. [65] used the singular-value transformation technique of Section 9.4 to implement A^{-1}, improving on an LCU construction due to Childs et al. [41]. For this we need a low-degree polynomial to approximate the function $f(x) = 1/x$. Childs et al. [41, Lemmas 17-19] started from the following polynomial of degree $D = 2b - 1$ for $b = O(\kappa^2 \log(\kappa/\varepsilon))$:

$$\frac{1 - (1 - x^2)^b}{x}.$$

This is indeed a polynomial because all terms in the numerator have degree ≥ 1, so we can divide out the x of the denominator. Since $(1 - x^2)^b$ is close to 0 (unless $|x|$ is small), this polynomial is indeed close to $1/x$ (unless $|x|$ is small, but we won't care because we'll apply this to a matrix whose eigenvalues aren't close to 0). More precisely, this polynomial is $\varepsilon/2$-close to $1/x$ whenever x lies in the interval $D_\kappa = [-1, -1/\kappa] \cup [1/\kappa, 1]$. Its range on this domain is $[-\kappa, -1] \cup [1, \kappa]$ (ignoring the small ε for simplicity). Like every degree-D polynomial, f can be written exactly as a sum of the first $D + 1$ Chebyshev polynomials of the first kind.[2] Childs et al. show that the coefficients in this sum decrease quickly for larger degree, and that dropping the Chebyshev polynomials of degree higher than $d = O(\kappa \log(\kappa/\varepsilon))$ incurs only small error $\varepsilon/2$. The resulting degree-d polynomial p ε-approximates $1/x$ on the interval D_κ, and its largest value (in absolute value) on this domain is κ. Now define the polynomial $P = p/(4\kappa)$. This has the same degree d as p, but a range $[-1/4, 1/4]$ that fits the assumption of Theorem 1 of Chapter 9 (there's a trick to ensure the values of P are within that range even for x very close to 0).

As we saw in Section 9.4, we can implement a block-encoding of the s-sparse matrix A/s using $O(1)$ sparse-access queries to A and $O(n)$ other gates. Using a factor $O(s)$ more work, we can turn this into a block-encoding of A itself (alternatively, we could directly invert the matrix A/s, whose singular values are $\geq 1/(\kappa s)$). We now apply Theorem 1 with this block-encoding of A, and the polynomial $P = p/(4\kappa)$, of degree $d = O(\kappa \log(\kappa/\varepsilon))$. Note that all eigenvalues of A lie in the interval D_κ, where $p(x) \approx 1/x$, hence $p(A) \approx A^{-1}$ and $P(A) \approx \frac{1}{4\kappa} A^{-1}$. Theorem 1 then gives us a block-encoding of $P(A)$, at the expense of running the block-encoding of A $O(d)$ times. Using $O(\kappa)$ rounds of amplitude amplification on top of this, we can get rid of the $1/(4\kappa)$ factor and end up with essentially the state $A^{-1}|b\rangle$, normalized.[3] This gives a quantum algorithm that

[2] These univariate polynomials are defined recursively as follows: $T_0(x) = 1$, $T_1(x) = x$, and $T_{d+1} = 2xT_d(x) - T_{d-1}(x)$. Note that T_d has degree d, and maps $[-1, 1]$ to $[-1, 1]$. The polynomials $T_0, \ldots, T_D$ are linearly independent (even orthonormal in a certain way) and hence span the set of all univariate polynomials of degree $\leq D$.

[3] Note that we need to assume a unitary to prepare $|b\rangle$ here, having just one copy of $|b\rangle$ is not enough. We cannot use *oblivious* amplitude amplification because that assumes we have a block-encoding of a matrix that is proportional to a unitary (or close to that), which A^{-1} is not.

solves the QLSP using $O(d\kappa s) = O(\kappa^2 s \log(\kappa/\varepsilon))$ queries to A, and $O(\kappa s(\kappa n \log(\kappa/\varepsilon) + B))$ 2-qubit gates. Note that compared to basic HHL, the dependence on $1/\varepsilon$ has been improved from linear to logarithmic. The dependence on κ can also be further improved, from quadratic to linear, using a technique called "variable-time amplitude amplification" [8, 41, 39] that we won't explain here.

The HHL algorithm can in some cases solve the QLSP exponentially faster than classical algorithms can solve the LSP. In particular, if the sparsity s, the condition number κ, and the cost B of preparing $|b\rangle$ are all $\leq \text{polylog}(N)$, and the allowed error is $\varepsilon \geq 2^{-\text{polylog}(N)}$, then this improved version of the HHL algorithm uses $\text{polylog}(N)$ queries and gates to solve (in a quantum way) an N-dimensional linear system.

Exercises

1. Suppose we are given an arbitrary invertible $N \times N$ matrix A and an N-dimensional vector b. Give a *Hermitian* $2N \times 2N$ matrix A' and $2N$-dimensional vector b' (based on A and b, respectively), such that a solution x to the linear system $Ax = b$ can be read off from a solution to the system $A'x' = b'$.

2. This exercise asks you to add more details to the sketch of the basic HHL algorithm given at the start of Section 10.2. For simplicity we will only count queries, not gates.

   (a) Use Hamiltonian simulation and phase estimation to implement the following unitary map:

   $$|a_j\rangle|0\rangle \mapsto |a_j\rangle\left|\widetilde{\lambda_j}\right\rangle,$$

   where $\left|\widetilde{\lambda_j}\right\rangle$ is a superposition over estimates of λ_j, which (if measured) gives with probability ≥ 0.99 an estimator $\ell \in [-1, 1]$ such that $|\lambda_j - \ell| \leq \varepsilon/\kappa$. Your implementation is allowed to use $O(\kappa s/\varepsilon)$ queries to the sparse matrix A. You may invoke the best Hamiltonian simulator for sparse matrices from Section 9.4.

   (b) Show the basic HHL algorithm can be implemented using $O(\kappa^2 s/\varepsilon)$ sparse-access queries to A. To make your life easier, you may assume that $\left|\widetilde{\lambda_j}\right\rangle$ is just one basis state, so one estimator which is close to λ_j rather than a superposition over estimators (and hence the success probability 0.99 is actually 1). You may also assume the amplitude amplification at the end works perfectly.

Chapter 11

Quantum Query Lower Bounds

11.1 Introduction

Almost all the algorithms we have seen so far in this course worked in the *query* model. Here the goal is to compute some function $f : \{0,1\}^N \to \{0,1\}$ on a given input $x \in \{0,1\}^N$. The distinguishing feature of the query model is the way x is accessed: x is not given explicitly, but is stored in a random access memory, and we're being charged unit cost for each *query* that we make to this memory. Informally, a query asks for and receives the i-th element x_i of the input. Formally, we model a query unitarily as the following 2-register quantum operation O_x, where the first register is N-dimensional and the second is 2-dimensional[1]:

$$O_x : |i, b\rangle \mapsto |i, b \oplus x_i\rangle.$$

In particular, $|i, 0\rangle \mapsto |i, x_i\rangle$. This only states what O_x does on basis states, but by linearity this determines the full unitary. Note that a quantum algorithm can apply O_x to a superposition of basis states, gaining some sort of access to several input bits x_i at the same time.

A T-query quantum algorithm starts in a fixed state, say the all-0 state $|0 \dots 0\rangle$, and then interleaves fixed unitary transformations $U_0, U_1, \dots, U_T$ with queries. The algorithm's fixed unitaries may act on a workspace-register, in addition to the two registers on which O_x acts. In this case we implicitly extend O_x by tensoring it with the identity operation on this extra register, so it maps

$$O_x : |i, b, w\rangle \mapsto |i, b \oplus x_i, w\rangle.$$

Hence the final state of the algorithm can be written as the following matrix-vector product:

$$U_T O_x U_{T-1} O_x \cdots O_x U_1 O_x U_0 |0 \dots 0\rangle.$$

This state depends on the input x only via the T queries. The output of the algorithm is obtained by a measurement of the final state. For instance, if the output is Boolean, the algorithm could just measure the final state in the computational basis and output the first bit of the result.

The query complexity of some function f is now the minimal number of queries needed for an algorithm that outputs the correct value $f(x)$ for every x in the domain of f (with error probability

[1] If the input x consists of non-binary items x_i (as is the case for instance with the input for Simon's algorithm) then those can be simulated by querying individual bits of each x_i.

at most 1/3, say). Note that we just count queries to measure the complexity of the algorithm[2], while the intermediate fixed unitaries are treated as costless.

In many cases, the overall computation time of quantum query algorithms (as measured by the total number of elementary gates, say) is not much bigger than the query complexity. This justifies analyzing the latter as a proxy for the former. This is the model in which essentially all the quantum algorithm we've seen work: Deutsch-Jozsa, Simon, Grover, the various random walk algorithms. Even the period-finding algorithm that is the quantum core of Shor's algorithm works because it needs only few queries to the periodic function.

11.2 The polynomial method

From quantum query algorithms to polynomials. An *N-variate multilinear polynomial* p is a function $p : \mathbb{C}^N \to \mathbb{C}$ that can be written as

$$p(x_0, \ldots, x_{N-1}) = \sum_{S \subseteq \{0,\ldots,N-1\}} a_S \prod_{i \in S} x_i,$$

for some complex numbers a_S. The *degree* of p is $deg(p) = \max\{|S| : a_S \neq 0\}$. It is easy to show that every function $f : \{0,1\}^N \to \mathbb{C}$ has a unique representation as such a polynomial; $deg(f)$ is defined as the degree of that polynomial (see Exercise 1). For example, the 2-bit AND function is $p(x_0, x_1) = x_0 x_1$, and the 2-bit Parity function is $p(x_0, x_1) = x_0 + x_1 - 2x_0 x_1$. Both polynomials have degree 2. Sometimes a lower degree suffices for a polynomial to *approximate* the function. For example, $p(x_0, x_1) = \frac{1}{3}(x_0 + x_1)$ approximates the 2-bit AND function up to error 1/3 for all inputs, using degree 1.

A very useful property of T-query algorithms is that the amplitudes of their final state are degree-T N-variate polynomials of x [61, 16]. More precisely: consider a T-query algorithm with input $x \in \{0,1\}^N$ acting on an m-qubit space. Then its final state can be written

$$\sum_{z \in \{0,1\}^m} \alpha_z(x)|z\rangle,$$

where each α_z is a multilinear complex-valued polynomial in x of degree at most T.

Proof. The proof is by induction on T. The base case ($T = 0$) trivially holds: the algorithm's state $U_0|0 \ldots 0\rangle$ is independent of x, so its amplitudes are constants.

For the induction step, suppose we have already done T queries. Then by the induction hypothesis the state after U_T can be written as

$$\sum_{z \in \{0,1\}^m} \alpha_z(x)|z\rangle,$$

where each α_z is a multilinear polynomial in x of degree at most T. Each basis state $|z\rangle = |i, b, w\rangle$ consists of 3 registers: the two registers $|i, b\rangle$ of the query, and a workspace register containing basis state $|w\rangle$. The algorithm now makes another query O_x followed by a unitary U_{T+1}. The query

[2]Clearly, N queries always suffice since we can just query each of the N input bits separately, thus learning x completely, and then look up and output whatever the correct value is for that input.

swaps basis states $|i,0,w\rangle$ and $|i,1,w\rangle$ if $x_i = 1$, and doesn't do anything to these basis states if $x_i = 0$. This changes amplitudes as follows:

$$\alpha_{i,0,w}(x)|i,0,w\rangle + \alpha_{i,1,w}(x)|i,1,w\rangle \mapsto$$
$$((1-x_i)\alpha_{i,0,w}(x) + x_i\alpha_{i,1,w}(x))|i,0,w\rangle + (x_i\alpha_{i,0,w}(x) + (1-x_i)\alpha_{i,1,w}(x))|i,1,w\rangle.$$

Now the new amplitudes are of the form $(1-x_i)\alpha_{i,0,w}(x)+x_i\alpha_{i,1,w}(x)$ or $x_i\alpha_{i,0,w}(x)+(1-x_i)\alpha_{i,1,w}(x)$. The new amplitudes are still polynomials in $x_0,\ldots,x_{N-1}$. Their degree is at most 1 more than the degree of the old amplitudes, so at most $T+1$. Finally, since U_{T+1} is a linear map that is independent of x, it does not increase the degree of the amplitudes further (the amplitudes after U_{T+1} are linear combinations of the amplitudes before U_{T+1}). This concludes the induction step.

Note that this construction could introduce degrees higher than 1, e.g., terms of the form x_i^2. However, our inputs x_i are 0/1-valued, so we have $x_i^k = x_i$ for all integers $k \geq 1$. Accordingly, we can reduce higher degrees to 1, making the polynomials multilinear without increasing degree. □

Suppose our algorithm acts on an m-qubit state. If we measure the first qubit of the final state and output the resulting bit, then the probability of output 1 is given by

$$p(x) = \sum_{z\in\{1\}\times\{0,1\}^{m-1}} |\alpha_z(x)|^2.$$

This is a real-valued polynomial of x of degree at most $2T$, because $|\alpha_z(x)|^2$ is the sum of the squares of the real and imaginary parts of the amplitude $\alpha_z(x)$, each of which is a polynomial of degree $\leq T$. Note that if the algorithm computes f with error $\leq 1/3$, then p is an approximating polynomial for f: if $f(x) = 0$ then $p(x) \in [0,1/3]$ and if $f(x) = 1$ then $p(x) \in [2/3,1]$. This gives a method to lower bound the minimal number of queries needed to compute f: if one can show that every polynomial that approximates f has degree at least d, then every quantum algorithm computing f with error $\leq 1/3$ must use at least $d/2$ queries.

Applications of the polynomial method. For our examples we will restrict attention to *symmetric* functions.[3] Those are the ones where the function value $f(x)$ only depends on the Hamming weight (number of 1s) in the input x. Examples are N-bit OR, AND, Parity, Majority, etc.

Suppose we have a polynomial $p(x_0,\ldots,x_{N-1})$ that approximates f with error $\leq 1/3$. Then it is easy to see that a polynomial that averages over all permutations π of the N input bits $x_0,\ldots,x_{N-1}$:

$$q(x) = \frac{1}{N!}\sum_{\pi\in S_N} p(\pi(x)),$$

still approximates f. As it turns out, we can define a single-variate polynomial $r(z)$ of the same degree as q, such that $q(x) = r(|x|)$.[4] This r is defined on all real numbers, and we know something

[3] One can also use the polynomial method for non-symmetric functions, for instance to prove a tight lower bound of $\Omega(N^{2/3})$ queries for the general problem of collision-finding; this matches the quantum walk algorithm of Section 8.3.2. However, that lower bound proof is substantially more complicated and we won't give it here.

[4] To see why this is the case, note that for every degree i, all degree-i monomials in the symmetrized polynomial q have the same coefficient a_i. Moreover, on input $x \in \{0,1\}^N$ of Hamming weight z, exactly $\binom{z}{i}$ of the degree-i monomials are 1, while the others are 0. Hence $q(x) = \sum_{i=0}^d a_i\binom{|x|}{i}$. Since $\binom{z}{d} = z(z-1)\cdots(z-d+1)/d!$ is a single-variate polynomial in z of degree d, we can define $r(z) = \sum_{i=0}^d a_i\binom{z}{i}$.

about its behavior on integer points $\{0,\ldots,N\}$. Thus it suffices to lower bound the degree of single-variate polynomials with the appropriate behavior.

For an important example, consider the N-bit OR function. Grover's algorithm can find an i such that $x_i = 1$ (if such an i exists) and hence can compute the OR function with error probability $\leq 1/3$ using $O(\sqrt{N})$ queries. By the above reasoning, any T-query quantum algorithm that computes the OR with error $\leq 1/3$ induces a single-variate polynomial r satisfying

$$r(0) \in [0,1/3], \text{ and } r(t) \in [2/3,1] \text{ for all integers } t \in \{1,\ldots,N\}.$$

This polynomial $r(x)$ "jumps" between $x = 0$ and $x = 1$ (i.e., it has a derivative $r'(x) \geq 1/3$ for some $x \in [0,1]$), while it remains fairly constant on the domain $\{1,\ldots,N\}$. By a classical theorem from approximation theory (proved independently around the same time by Ehlich and Zeller [55], and by Rivlin and Cheney [116]), such polynomials must have degree $d \geq \Omega(\sqrt{N})$. Hence $T \geq \Omega(\sqrt{N})$ as well. Accordingly, Grover's algorithm is optimal (up to a constant factor) in terms of number of queries.

What about *exact* algorithms for OR? Could we tweak Grover's algorithm so that it always finds a solution with probability 1 (if one exists), using $O(\sqrt{N})$ queries? This turns out not to be the case: a T-query exact algorithm for OR induces a polynomial r of degree $\leq 2T$ that satisfies

$$r(0) = 0, \text{ and } r(t) = 1 \text{ for all integers } t \in \{1,\ldots,N\}.$$

It is not hard to see that such a polynomial needs degree at least N: observe that $r(x) - 1$ is a non-constant polynomial with at least N roots.[5] Hence $T \geq N/2$. Accordingly, Grover cannot be made exact without losing the square-root speed-up!

Using the polynomial method, one can in fact show for *every symmetric function* f that is defined on all 2^N inputs, that quantum algorithms cannot provide a more-than-quadratic speed-up over classical algorithms. More generally, for *every* function f (symmetric or non-symmetric) that is defined on all inputs[6], quantum algorithms cannot provide a more-than-6th-root speed-up over classical algorithms (see Exercise 10).

11.3 The quantum adversary method

The polynomial method has a strength which is also a weakness: it applies even to a stronger (and less physically meaningful) model of computation where we allow *any linear transformation* on the state space, not just unitary ones. As a result, it does not always provide the strongest possible lower bound for quantum query algorithms.

Ambainis [5, 6] provided an alternative method for quantum lower bounds, the *quantum adversary*. This exploits unitarity in a crucial way and in certain cases yields a provably better bound than the polynomial method [6]. We will present a very simple version of the adversary method here. Stronger versions may be found in [78, 77]; the latter one actually gives optimal bounds for every Boolean function [113]! Recall that a quantum query algorithm is a sequence

$$U_T O_x U_{T-1} O_x \cdots O_x U_1 O_x U_0,$$

[5] A "root" is an x such that $r(x) = 0$. It is a well-known fact from algebra that every univariate non-constant polynomial of degree d has at most d roots (over any field). Note that this is not true for multivariate polynomials; for example the polynomial $x_0 \cdots x_{N-1}$ has $2^N - 1$ roots in $\{0,1\}^N$ but degree only N

[6] Note that this doesn't include functions where the input has to satisfy a certain promise, such as Deutsch-Jozsa and Simon's problem.

applied to the fixed starting state $|0\dots0\rangle$, where the basic "query transformation" O_x depends on the input x, and $U_0, U_1, \dots, U_T$ are arbitrary unitaries that don't depend on x. Consider the evolution of our quantum state under all possible choices of x; formally, we let $|\psi_x^t\rangle$ denote the state at time t (i.e., after applying O_x for the t-th time) under input x. In particular, $|\psi_x^0\rangle = |0\dots0\rangle$ for all x (and $\langle\psi_x^0|\psi_y^0\rangle = 1$ for each x, y). Now if the algorithm computes the Boolean function f with success probability $2/3$ on every input, then the final measurement must accept every $x \in f^{-1}(0)$ with probability $\leq 1/3$, and accept every $y \in f^{-1}(1)$ with probability $\geq 2/3$. It is not hard to verify that this implies $|\langle\psi_x^T|\psi_y^T\rangle| \leq \frac{17}{18}$.[7] This suggests that we find a $R \subseteq f^{-1}(0) \times f^{-1}(1)$ of hard-to-distinguish (x, y)-pairs, and consider the *progress measure*

$$S_t = \sum_{(x,y)\in R} |\langle\psi_x^t|\psi_y^t\rangle|$$

as a function of t. By our observations, initially we have $S_0 = |R|$, and in the end we must have $S_T \leq \frac{17}{18}|R|$. Also, crucially, the progress measure is *unaffected* by each application of a unitary U_t, since each U_t is independent of the input and unitary transformations preserve inner products.

If we can determine an upper bound Δ on the change $|S_{t+1} - S_t|$ in the progress measure at each step, we can conclude that the number T of queries is at least $\frac{|R|}{18\Delta}$. Ambainis proved the following. Suppose that

(i) each $x \in f^{-1}(0)$ appearing in R, appears at least m_0 times in pairs (x, y) in R;

(ii) each $y \in f^{-1}(1)$ appearing in R, appears at least m_1 times in pairs (x, y) in R;

(iii) for each $x \in f^{-1}(0)$ and $i \in \{0, \dots, N-1\}$, there are at most ℓ_0 inputs $y \in f^{-1}(1)$ such that $(x, y) \in R$ and $x_i \neq y_i$;

(iv) for each $y \in f^{-1}(1)$ and $i \in \{0, \dots, N-1\}$, there are at most ℓ_1 inputs $x \in f^{-1}(0)$ such that $(x, y) \in R$ and $x_i \neq y_i$.

Then for all $t \geq 0$, $|S_{t+1} - S_t| = O\left(\sqrt{\frac{\ell_0}{m_0} \cdot \frac{\ell_1}{m_1}} \cdot |R|\right)$, and therefore

$$T = \Omega\left(\sqrt{\frac{m_0}{\ell_0} \cdot \frac{m_1}{\ell_1}}\right). \tag{11.1}$$

Intuitively, conditions (i)-(iv) imply that $|S_{t+1} - S_t|$ is small relative to $|R|$ by bounding the "distinguishing ability" of any query. The art in applying this technique lies in choosing the relation R carefully to maximize this quantity, i.e., make m_0 and/or m_1 large, while keeping ℓ_0 and ℓ_1 small.

Note that for the N-bit OR function this method easily gives the optimal $\Omega(\sqrt{N})$ lower bound, as follows. Choose $R = \{(x, y) : x = 0^N, y \text{ has Hamming weight } 1\}$. Then $m_0 = N$ while $m_1 = \ell_0 = \ell_1 = 1$. Plugging this into Eq. (11.1) gives the right bound.

Let us give another application, a lower bound that is much harder to prove using the polynomial method. Suppose $f : \{0,1\}^N \to \{0,1\}$ is a 2-level AND-OR tree, with $N = k^2$ input bits: f is the

[7] Remember Exercise 3 from Chapter 4 for states $|\phi\rangle$ and $|\psi\rangle$: if $\|\phi - \psi\| = \varepsilon$ then the total variation distance between the probability distributions you get from measuring $|\phi\rangle$ and $|\psi\rangle$, respectively, is at most ε. Hence, if we know there is a two-outcome measurement that accepts $|\phi\rangle$ with probability $\leq 1/3$ and accepts $|\psi\rangle$ with probability $\geq 2/3$, then we must have total variation distance at least $1/3$ and hence $\varepsilon \geq 1/3$. Assume for simplicity that the inner product $\langle\phi|\psi\rangle$ is real. Via the equation $\varepsilon^2 = \|\phi - \psi\|^2 = 2 - 2\langle\phi|\psi\rangle$, this translates into an upper bound $|\langle\phi|\psi\rangle| \leq 1 - \varepsilon^2/2 \leq 17/18$ (this upper bound can be improved to $2\sqrt{2}/3$ with more careful analysis).

AND of k ORs, each of which has its own set of k inputs bits. By carefully doing 2 levels of Grover search (search for a subtree which is 0^k), one can construct a quantum algorithm that computes f with small error probability and $O(\sqrt{k}\cdot\sqrt{k}) = O(\sqrt{N})$ queries. It was long an open problem to give a matching lower bound on the approximate degree, and this was proved only in 2013 [120, 37]. In contrast, the adversary method gives the optimal lower bound on the quantum query complexity quite easily: choose the relation R as follows

> R consists of those pairs (x, y) where
> x has one subtree with input 0^k and the other $k-1$ subtrees have an arbitrary k-bit input of Hamming weight 1 (note $f(x) = 0$)
> y is obtained from x by changing one of the bits of the 0^k-subtree to 1 (note $f(y) = 1$).

Then $m_0 = m_1 = k$ and $\ell_0 = \ell_1 = 1$, and we get a lower bound of $\Omega\left(\sqrt{\frac{m_0 m_1}{\ell_0 \ell_1}}\right) = \Omega(k) = \Omega(\sqrt{N})$.

Exercises

1. Consider a function $f : \{0,1\}^N \to \mathbb{R}$. Show that this function can be represented by an N-variate multilinear polynomial of degree $\leq N$, and that this representation is unique.

2. Consider a 2-bit input $x = x_0x_1$ with phase-oracle $O_{x,\pm} : |i\rangle \mapsto (-1)^{x_i}|i\rangle$. Write out the final state of the following 1-query quantum algorithm: $HO_{x,\pm}H|0\rangle$. Give a degree-2 polynomial $p(x_0, x_1)$ that equals the probability that this algorithm outputs 1 on input x. What function does this algorithm compute?

3. Consider polynomial $p(x_0, x_1) = 0.3 + 0.4x_0 + 0.5x_1$, which approximates the 2-bit OR function. Write down the symmetrized polynomial $q(x_0, x_1) = \frac{1}{2}(p(x_0, x_1) + p(x_1, x_0))$. Give a single-variate polynomial r such that $q(x) = r(|x|)$ for all $x \in \{0,1\}^2$.

4. (H) Let f be the N-bit Parity function, which is 1 if its input $x \in \{0,1\}^N$ has odd Hamming weight, and 0 if the input has even Hamming weight (assume N is an even number).

   (a) Give a quantum algorithm that computes Parity with success probability 1 on every input x, using $N/2$ queries.

   (b) Show that this is optimal, even for quantum algorithms that have error probability $\leq 1/3$ on every input

5. Suppose we have a T-query quantum algorithm that computes the N-bit AND function with success probability 1 on all inputs $x \in \{0,1\}^N$. In Section 11.2 we showed that such an algorithm has $T \geq N/2$ (we showed it for OR, but the same argument works for AND). Improve this lower bound to $T \geq N$.

6. Consider the following 3-bit function $f : \{0,1\}^3 \to \{0,1\}$:
   $f(x_0, x_1, x_2) = 1$ if $x_0 = x_1 = x_2$, and $f(x_0, x_1, x_2) = 0$ otherwise

   (a) How many queries does a classical deterministic algorithm need to compute f? Explain your answer.

   (b) Give a quantum algorithm that computes f with success probability 1 using 2 queries.

(c) (H) Show that 2 queries is optimal: there is no quantum algorithm that computes f with success probability 1 using only 1 query.

7. Let f be the N-bit Majority function, which is 1 if its input $x \in \{0,1\}^N$ has Hamming weight $> N/2$, and 0 if the input has Hamming weight $\leq N/2$ (assume N is even).

   (a) Prove that $deg(f) \geq N/2$. What does this imply for the query complexity of exact quantum algorithms that compute majority?

   (b) (H) Use the adversary method to show that every bounded-error quantum algorithm for computing Majority, needs $\Omega(N)$ queries. Be explicit about what relation R you're using, and about the values of the parameters m_0, m_1, ℓ_0, ℓ_1.

8. Let k be an odd natural number, $N = k^2$, and define the Boolean function $f : \{0,1\}^N \to \{0,1\}$ as the k-bit majority of k separate k-bit OR functions. In other words, the N-bit input is $x = x^{(1)} \dots x^{(k)}$ with $x^{(i)} \in \{0,1\}^k$ for each $i \in [k]$, and $f(x)$ is the majority value of the k bits $\mathrm{OR}(x^{(1)}), \dots, \mathrm{OR}(x^{(k)})$. Use the adversary method to prove that computing this f with error probability $\leq 1/3$ requires $\Omega(N^{3/4})$ quantum queries. Be explicit about what relation R you're using, and about the values of the parameters m_0, m_1, ℓ_0, ℓ_1.

9. (H) Consider the *sorting* problem: there are N numbers $a_1, \dots, a_N$ and we want to sort these. We can only access the numbers by making *comparisons*. A comparison is similar to a black-box query: it takes 2 indices i, j as input and outputs whether $a_i < a_j$ or not. The output of a sorting algorithm should be the list of N indices, sorted in increasing order. It is known that for classical computers, $N \log_2(N) + O(N)$ comparisons are necessary and sufficient for sorting. Prove that a quantum algorithm needs at least $\Omega(N)$ comparisons for sorting, even if it is allowed an error probability $\leq 1/3$.

10. Consider a total Boolean function $f : \{0,1\}^N \to \{0,1\}$. Given an input $x \in \{0,1\}^N$ and subset $B \subseteq \{0, \dots, N-1\}$ of indices of variables, let x^B denote the N-bit input obtained from x by flipping all bits x_i whose index i is in B. The *block sensitivity* $bs(f,x)$ of f at input x, is the maximal integer k such that there exist disjoint sets $B_1, \dots, B_k$ satisfying $f(x) \neq f(x^{B_i})$ for all $i \in [k]$. The *block sensitivity* $bs(f)$ of f is $\max_x bs(f,x)$.

    (a) (H) Show that the bounded-error quantum query complexity of f is $\Omega(\sqrt{bs(f)})$.

    (b) It is known that for every total Boolean function f, there is a classical deterministic algorithm that computes it using $O(bs(f)^3)$ many queries. What can you conclude from this and part (a) about the relation between deterministic and quantum query complexity for total functions?

Chapter 12

Quantum Complexity Theory

12.1 Most functions need exponentially many gates

As we have seen, quantum computers seem to provide enormous speed-ups for problems like factoring, and square-root speed-ups for various search-related problems. Could they be used to speed up almost all problems, at least by some amount? Here we will show that this is not the case: as it turns out, quantum computers are not significantly better than classical computers for most computational problems.

Consider the problem of computing a Boolean function $f : \{0,1\}^n \to \{0,1\}$ by means of a quantum circuit. Ideally, most such functions would be computable by efficient quantum circuits (i.e., using at most poly(n) elementary gates). Instead, we will show by means of a simple counting argument that almost all such functions f have circuit complexity nearly 2^n. This is a variant of a well-known counting argument for classical Boolean circuits due to Shannon.

Let us fix some finite set of elementary gates, for instance the Shor basis $\{H, T, \text{CNOT}\}$ or $\{H, \text{Toffoli}\}$. Suppose this set has k types of gates, of maximal fanout 3. Let us try to count the number of distinct circuits that have at most C elementary gates. For simplicity we include the initial qubits (the n input bits as well as workspace qubits, which are initially $|0\rangle$) as a $(k+1)$st type among those C gates. First we need to choose which type of elementary gate each of the C gates is; this can be done in $(k+1)^C$ ways. Now every gate has at most 3 ingoing and 3 outgoing wires. For each of its 3 outgoing wires we can choose an ingoing wire into one of the gates in the following level; this can be done in at most $(3C)^3$ ways. Hence the total number of circuits with up to C elementary gates is at most $(k+1)^C(3C)^{3C} = C^{O(C)}$. We are clearly overcounting here, but that's OK because we want an upper bound on the number of circuits.

We'll say that a specific circuit computes a Boolean function $f : \{0,1\}^n \to \{0,1\}$ if for every input $x \in \{0,1\}^n$, a measurement of the first qubit of the final state (obtained by applying the circuit to initial state $|x,0\rangle$) gives value $f(x)$ with probability at least $2/3$. Each of our $C^{O(C)}$ circuits can compute at most one f (in fact some of those circuits don't compute any Boolean function at all). Accordingly, with C gates we can compute at most $C^{O(C)}$ distinct Boolean functions $f : \{0,1\}^n \to \{0,1\}$. Hence even if we just want to be able to compute 1% of all 2^{2^n} Boolean functions, then we already need

$$C^{O(C)} \geq \frac{1}{100} 2^{2^n}, \text{ which implies } C \geq \Omega(2^n/n).$$

Accordingly, very few computational problems will be efficiently solvable on a quantum computer.

Below we will try to classify those using the tools of complexity theory.

12.2 Classical and quantum complexity classes

A "complexity class" is a set of decision problems (a.k.a. "languages") that all have similar complexity in some sense, for instance the ones that can be solved with polynomial time or polynomial space. Let us first mention some of the main classical complexity classes:

- **P**. The class of problems that can be solved by classical deterministic computers using polynomial time.
- **BPP**. The problems that can be solved by classical randomized computers using polynomial time (and with error probability $\leq 1/3$ on every input).
- **NP**. The problems where the 'yes'-instances can be verified in polynomial time if some prover gives us a polynomial-length "witness." Some problems in this class are **NP**-*complete*, meaning that any other problem in **NP** can be reduced to it in polynomial time. Hence the **NP**-complete problems are the hardest problems in **NP**. An example is the problem of satisfiability: we can verify that a given n-variable Boolean formula is satisfiable if a prover gives us a satisfying assignment, so it is in **NP**, but one can even show that is **NP**-complete. Other examples are integer linear programming, travelling salesman, graph-colorability, etc.
- **PSPACE**. The problems that can be solved by classical deterministic computers using polynomial space.

We can consider quantum analogues of all such classes, an enterprise that was started by Bernstein and Vazirani [23]:

- **EQP**. The class of problems that can be solved exactly by quantum computers using polynomial time. This class depends on the set of elementary gates one allows, and is not so interesting.
- **BQP**. The problems that can be solved by quantum computers using polynomial time (and with error probability $\leq 1/3$ on every input). This class is the accepted formalization of "efficiently solvable by quantum computers."
- "quantum **NP**". In analogy with the above definition of **NP**, one could define quantum **NP** as the class of problems where the 'yes'-instances can be verified efficiently if some prover gives us a "quantum witness" of a polynomial number of qubits. For every 'yes'-instance there should be a quantum witness that passes the verification with probability 1, while for 'no'-instances every quantum witness should be rejected with probability 1. This class is again dependent on the elementary gates one allows, and not so interesting. Allowing error probability $\leq 1/3$ on every input, we get a class called **QMA** ("quantum Merlin-Arthur"). This is a more robust and more interesting quantum version of **NP**; unfortunately we don't have time to study it in this course.
- **QPSPACE**. The problems that can be solved by quantum computers using polynomial space. This turns out to be the same as classical **PSPACE**.

As explained in Appendix B, in all the above cases the error probability 1/3 can be reduced efficiently to much smaller constant $\varepsilon > 0$: just run the computation $O(\log(1/\varepsilon))$ times and take the majority of the answers given by these runs.

We should be a bit careful about what we mean by a "polynomial time [or space] quantum algorithm." Our model for computation has been quantum circuits, and we need a separate quantum circuit for each new input length. So a quantum algorithm of time $p(n)$ would correspond to a *family* of quantum circuits $\{C_n\}$, where C_n is the circuit that is used for inputs of length n; it should have at most $p(n)$ elementary gates.[1]

In the next section we will prove that **BQP** $\subseteq$ **PSPACE**. We have **BPP** $\subseteq$ **BQP**, because a **BPP**-machine on a fixed input length n can be written as a polynomial-size reversible circuit (i.e., consisting of Toffoli gates) that starts from a state that involves some coin flips. Quantum computers can generate those coin flips using Hadamard transforms, then run the reversible circuit, and measure the final answer bit. It is believed that **BQP** contains problems that aren't in **BPP**, for example factoring large integers: this problem (or rather the decision-version thereof) is in **BQP** because of Shor's algorithm, and is generally believed not to be in **BPP**. Thus we have the following sequence of inclusions:

$$\mathbf{P} \subseteq \mathbf{BPP} \subseteq \mathbf{BQP} \subseteq \mathbf{PSPACE}.$$

It is generally believed that **P** = **BPP**, while the other inclusions are believed to be strict. Note that a *proof* that **BQP** is strictly greater than **BPP** (for instance, a proof that factoring cannot be solved efficiently by classical computers) would imply that **P** $\neq$ **PSPACE**, solving what has been one of the main open problems in computers science since the 1960s. Hence such a proof—if it exists at all—will probably be very hard.

What about the relation between **BQP** and **NP**? It's generally believed that **NP**-complete problems are probably not in **BQP**. The main evidence for this is the lower bound for Grover search: a quantum brute-force search on all 2^n possible assignments to an n-variable formula gives a square-root speed-up, but not more. This is of course not a proof, since there might be clever, non-brute-force methods to solve satisfiability. However, neither in the classical nor in the quantum case do we know clever methods that solve the general satisfiability problem much faster than brute-force search.

Finally, there could also be problems in **BQP** that are not in **NP**, so it may well be that **BQP** and **NP** are incomparable. Much more can be said about quantum complexity classes; see for instance Watrous's survey [130].

12.3 Classically simulating quantum computers in polynomial space

When Richard Feynman first came up with quantum computers [59], he motivated them by

> "the full description of quantum mechanics for a large system with R particles is given by a function $q(x_1, x_2, \ldots, x_R, t)$ which we call the amplitude to find the particles $x_1, \ldots, x_R$ [RdW: think of x_i as one qubit], and therefore, because it has too many variables, it

[1] To avoid smuggling loads of hard-to-compute information into this definition (e.g., C_n could contain information about whether the nth Turing machine halts or not), we will require this family to be efficiently describable: there should be a classical Turing machine which, on input n and j, outputs (in time polynomial in n) the jth elementary gate of C_n, with information about where its incoming and outcoming wires go.

> cannot be simulated with a normal computer with a number of elements proportional to R or proportional to N." [...]
> "Can a quantum system be probabilistically simulated by a classical (probabilistic, I'd assume) universal computer? In other words, a computer which will give the same probabilities as the quantum system does. If you take the computer to be the classical kind I've described so far (not the quantum kind described in the last section) and there are no changes in any laws, and there's no hocus-pocus, the answer is certainly, No!"

The suggestion to devise a *quantum* computer to simulate quantum physics is of course a brilliant one, but the main motivation is not quite accurate. As it turns out, it is not necessary to keep track of all (exponentially many) amplitudes in the state to classically simulate a quantum system. Here we will show that it can actually be simulated efficiently in terms of *space* [23], though not necessarily in terms of *time*.

Consider a circuit with $T = \text{poly}(n)$ gates that acts on S qubits. Assume for simplicity that all gates are either the 1-qubit Hadamard or the 3-qubit Toffoli gate (as mentioned before, these two gates suffice for universal quantum computation), and that the classical output (0 or 1) of the algorithm is determined by a measurement of the first qubit of the final state. Without loss of generality $S \leq 3T$, because T Toffoli gates won't affect more than $3T$ qubits. Let U_j be the unitary that applies the jth gate to its (1 or 3) qubits, and applies identity to all other qubits. The entries of this matrix are of a simple form (0, $1/\sqrt{2}$, or $-1/\sqrt{2}$ for Hadamard; 0 or 1 for Toffoli) and easy to compute. Let $|i_0\rangle = |x\rangle|0^{S-n}\rangle$ be the starting state, where $x \in \{0,1\}^n$ is the classical input, and the second register contains the workspace qubits the algorithm uses. The final state will be

$$|\psi_x\rangle = U_T U_{T-1} \cdots U_2 U_1 |i_0\rangle.$$

The amplitude of basis state $|i_T\rangle$ in this final state is

$$\langle i_T|\psi_x\rangle = \langle i_T|U_T U_{T-1} U_{T-2} \cdots U_2 U_1|i_0\rangle.$$

Inserting an identity matrix $I = \sum_{i\in\{0,1\}^S} |i\rangle\langle i|$ between the gates, we can rewrite this as[2]

$$\begin{aligned}\langle i_T|\psi_x\rangle &= \langle i_T|U_T \left(\sum_{i_{T-1}} |i_{T-1}\rangle\langle i_{T-1}|\right) U_{T-1} \left(\sum_{i_{T-2}} |i_{T-2}\rangle\langle i_{T-2}|\right) U_{T-2} \cdots U_2 \left(\sum_{i_1} |i_1\rangle\langle i_1|\right) U_1|x,0\rangle \\ &= \sum_{i_{T-1},\ldots,i_1} \prod_{j=1}^{T} \langle i_j|U_j|i_{j-1}\rangle.\end{aligned}$$

The number $\langle i_j|U_j|i_{j-1}\rangle$ is just one entry of the matrix U_j and hence easy to calculate. Then $\prod_{j=1}^{T} \langle i_j|U_j|i_{j-1}\rangle$ is also easy to compute, in polynomial space (and polynomial time). If ℓ of the T gates are Hadamards, then each such number is either 0 or $\pm 1/\sqrt{2^\ell}$.

Adding up $\prod_{j=1}^{T} \langle i_j|U_j|i_{j-1}\rangle$ for all $i_{T-1},\ldots,i_1$ is also easy to do in polynomial space if we reuse space for each new $i_{T-1},\ldots,i_1$. Hence the amplitude $\langle i_T|\psi_x\rangle$ can be computed exactly using polynomial space.[3] We assume that the **BQP** machine's answer is obtained by measuring

[2] For the physicists: this is very similar to a path integral.

[3] Of course, the calculation will take exponential *time*, because there are $2^{S(T-1)}$ different sequences $i_{T-1},\ldots,i_1$ that we need to go over sequentially.

the first qubit of the final state. Then its acceptance probability is the sum of squares of all amplitudes of basis states starting with a 1: $\sum_{i_T:(i_T)_1=1} |\langle i_T|\psi_x\rangle|^2$. Since we can compute each $\langle i_T|\psi_x\rangle$ in polynomial space, the acceptance probability of a **BQP**-circuit on classical input x can be computed in polynomial space.

Exercises

1. (H) The following problem is a decision version of the factoring problem:

   Given positive integers N and k, decide if N has a prime factor $p \in \{k, \ldots, N-1\}$.

   Show that if you can solve this decision problem efficiently (i.e., in time polynomial in the input length $n = \lceil \log N \rceil$), then you can also find the prime factors of N efficiently.

2. (a) Let U be an S-qubit unitary which applies a Hadamard gate to the kth qubit, and identity gates to the other $S-1$ qubits. Let $i, j \in \{0,1\}^S$. Show an efficient way (i.e., using time polynomial in S) to calculate the matrix-entry $U_{i,j} = \langle i|U|j\rangle$ (note: even though U is a tensor product of 2×2 matrices, it's still a $2^S \times 2^S$ matrix, so calculating U completely isn't efficient).

   (b) Let U be an S-qubit unitary which applies a CNOT gate to the kth and ℓth qubits, and identity gates to the other $S-2$ qubits. Let $i, j \in \{0,1\}^S$. Show an efficient way to calculate the matrix-entry $U_{i,j} = \langle i|U|j\rangle$.

3. (H) Consider a circuit C with $T = \text{poly}(n)$ elementary gates (only Hadamards and Toffolis) acting on $S = \text{poly}(n)$ qubits. Suppose this circuit computes $f : \{0,1\}^n \to \{0,1\}$ with bounded error probability: for every $x \in \{0,1\}^n$, when we start with basis state $|x, 0^{S-n}\rangle$, run the circuit and measure the first qubit, then the result equals $f(x)$ with probability at least $2/3$.

   (a) Consider the following quantum algorithm: start with basis state $|x, 0^{S-n}\rangle$, run the above circuit C without the final measurement, apply a Z gate to the first qubit, and reverse the circuit C. Denote the resulting final state by $|\psi_x\rangle$. Show that if $f(x) = 0$ then the amplitude of basis state $|x, 0^{S-n}\rangle$ in $|\psi_x\rangle$ is in the interval $[1/3, 1]$, while if $f(x) = 1$ then the amplitude of $|x, 0^{S-n}\rangle$ in $|\psi_x\rangle$ is in $[-1, -1/3]$.

   (b) **PP** is the class of computational decision problems that can be solved by classical randomized polynomial-time computers with success probability $> 1/2$ (however, the success probability could be exponentially close to $1/2$, i.e., **PP** is **BPP** without the 'B' for bounded-error). Show that **BQP** $\subseteq$ **PP**.

Chapter 13

Quantum Encodings, with a Non-Quantum Application

13.1 Mixed states and general measurements

So far, we have restricted our states to so-called *pure states*: unit vectors of amplitudes. In the classical world we often have uncertainty about the state of a system, which can be expressed by viewing the state as a random variable that has a certain probability distribution over the set of basis states. Similarly we can define a *mixed* quantum state as a probability distribution (or "mixture") over pure states. While pure states are written as vectors, it is most convenient to write mixed states as *density matrices*. A pure state $|\phi\rangle$ corresponds to the density matrix $|\phi\rangle\langle\phi|$, which is the outer product of the vector $|\phi\rangle$ with itself. For example, the pure state $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ corresponds to the density matrix

$$|\phi\rangle\langle\phi| = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \cdot (\alpha^* \quad \beta^*) = \begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2 \end{pmatrix}.$$

A mixed state that is in pure states $|\phi_1\rangle, \ldots, |\phi_m\rangle$ with probabilities $p_1, \ldots, p_m$, respectively, corresponds to the density matrix $\rho = \sum_{i=1}^m p_i |\phi_i\rangle\langle\phi_i|$. This ρ is sometimes called a "mixture" of the states $|\phi_1\rangle, \ldots, |\phi_m\rangle$.[1] The set of density matrices is exactly the set of positive semidefinite (PSD) matrices of trace 1. A mixed state is pure if, and only if, it has rank 1.

Applying a unitary U to a pure state $|\phi\rangle$ gives pure state $U|\phi\rangle$. Written in terms of rank-1 density matrices, this corresponds to the map

$$|\phi\rangle\langle\phi| \mapsto U|\phi\rangle\langle\phi|U^*.$$

By linearity, this actually tells us how a unitary acts on an arbitrary mixed state:

$$\rho \mapsto U\rho U^*.$$

What about measurements? Recall from Section 1.2.2 that an m-outcome projective measurement corresponds to m orthogonal projectors $P_1, \ldots, P_m$ that satisfy $\sum_{i=1}^m P_i = I$. When applying this measurement to a mixed state ρ, the probability to see outcome i is given by $p_i = \mathrm{Tr}(P_i\rho)$. If we

[1] Note that applying the probabilities p_i to the vectors $|\phi_i\rangle$ (rather than to the matrices $|\phi_i\rangle\langle\phi_i|$) does not make sense in general, because $\sum_{i=1}^m p_i|\phi_i\rangle$ need not be a unit vector.

get outcome i, then the state collapses to $P_i \rho P_i / p_i$ (the division by p_i renormalizes the state to have trace 1). This may look weird, but let's recover our familiar measurement in the computational basis in this framework. Suppose we measure a state $|\phi\rangle = \sum_{j=1}^{d} \alpha_j |j\rangle$ using d projectors $P_i = |i\rangle\langle i|$ (note that $\sum_i P_i$ is the identity on the d-dimensional space). The probability to get outcome i is given by $p_i = \mathrm{Tr}(P_i|\phi\rangle\langle\phi|) = |\langle i|\phi\rangle|^2 = |\alpha_i|^2$. If we get outcome i then the state collapses to $P_i|\phi\rangle\langle\phi|P_i/p_i = \alpha_i|i\rangle\langle i|\alpha_i^*/p_i = |i\rangle\langle i|$. This is exactly the measurement in the computational basis as we have used it until now. Similarly, a measurement of the first register of a two-register state corresponds to projectors $P_i = |i\rangle\langle i| \otimes I$, where i goes over all basis states of the first register.

If we only care about the final probability distribution on the m outcomes, not about the resulting state, then the most general thing we can do is a POVM. This is specified by m positive semidefinite matrices $E_1, \ldots, E_m$ satisfying $\sum_{i=1}^{m} E_i = I$. When measuring a state ρ, the probability of outcome i is given by $\mathrm{Tr}(E_i \rho)$.

13.2 Quantum encodings and their limits

Quantum information theory studies the quantum generalizations of familiar notions from classical information theory such as Shannon entropy, mutual information, channel capacities, etc. Here we will discuss a few quantum information-theoretic results that all have the same flavor: they say that a low-dimensional quantum state (i.e., a small number of qubits) cannot contain too much *accessible* information.

Holevo's Theorem: The mother of all such results is Holevo's theorem from 1973 [76], which predates the area of quantum computing by several decades. Its proper technical statement is in terms of a quantum generalization of mutual information, but the following consequence of it (derived by Cleve et al. [46]) about two communicating parties, suffices for our purposes.

Theorem 2 (Holevo, CDNT) *Suppose Alice wants to communicate some classical string x to Bob.*

- *If Alice sends Bob m qubits, and they did not share any prior entanglement, then Bob receives at most m bits of information about x.*
- *If Alice sends Bob m qubits, and they did share some prior entangled state, then Bob receives at most $2m$ bits of information about x.*
- *If Alice sends Bob m classical bits, and they did share some prior entangled state, then Bob receives at most m bits of information about x.*

This theorem is slightly imprecisely stated here, but the intuition should be clear: if Bob makes any measurement on his state after the communication, then the mutual information between his classical outcome and Alice's x, is bounded by m or $2m$. In particular, the first part of the theorem says that if we encode some classical random variable X in an m-qubit state[2], then no measurement on the quantum state can give more than m bits of information about X. If we encoded the classical information in an *m-bit* system instead of an m-qubit system this would be a trivial statement,

[2]Via an encoding map $x \mapsto \rho_x$; we generally use capital letters like X to denote random variables, lower case like x to denote specific values.

but the proof of Holevo's theorem is quite non-trivial. Thus we see that an m-qubit state, despite somehow "containing" 2^m complex amplitudes, is no better than m classical bits for the purpose of storing or transmitting information. Prior entanglement can improve this by a factor of 2 because of superdense coding (see Exercise 1.8), but no more than that.

Low-dimensional encodings: Here we provide a "poor man's version" of Holevo's theorem due to Nayak [107, Theorem 2.4.2], which has a simple proof and often suffices for applications. Suppose we have a classical random variable X, uniformly distributed over $[N] = \{1, \ldots, N\}$.[3] Let $x \mapsto \rho_x$ be some encoding of $[N]$, where ρ_x is a mixed state in a d-dimensional space. Let $E_1, \ldots, E_N$ be the POVM operators applied for decoding; these sum to the d-dimensional identity operator. Then the probability of correct decoding in case $X = x$, is

$$p_x = \mathrm{Tr}(E_x \rho_x) \leq \mathrm{Tr}(E_x).$$

The sum of these success probabilities is at most

$$\sum_{x=1}^{N} p_x \leq \sum_{x=1}^{N} \mathrm{Tr}(E_x) = \mathrm{Tr}\left(\sum_{x=1}^{N} E_x\right) = \mathrm{Tr}(I) = d. \tag{13.1}$$

In other words, if we are encoding one of N classical values in a d-dimensional quantum state, then any measurement to decode the encoded classical value has average success probability at most d/N (uniformly averaged over all N values that we can encode). For example, if we encode n uniformly random bits into m qubits, we will have $N = 2^n$, $d = 2^m$, and the average success probability of decoding is at most $2^m/2^n$.

Random access codes: The previous two results dealt with the situation where we encoded a classical random variable X in some quantum system, and would like to recover the original value X by an appropriate measurement on that quantum system. However, suppose $X = X_1 \ldots X_n$ is a string of n bits, uniformly distributed and encoded by a map $x \mapsto \rho_x$, and it suffices for us if we are able to decode individual bits X_i from this with some probability $p > 1/2$. More precisely, for each $i \in [n]$ there should exist a measurement $\{M_i, I - M_i\}$ allowing us to recover x_i: for each $x \in \{0,1\}^n$ we should have $\mathrm{Tr}(M_i \rho_x) \geq p$ if $x_i = 1$ and $\mathrm{Tr}(M_i \rho_x) \leq 1 - p$ if $x_i = 0$. An encoding satisfying this is called a *quantum random access code*, since it allows us to choose which bit of X we would like to access. Note that the measurement to recover x_i can change the state ρ_x, so generally we may not be able to decode more than one bit of x (also, we cannot copy ρ_x because of the no-cloning theorem, see Exercise 1.1).

An encoding that allows us to recover (with high success probability) an n-bit string requires about n qubits by Holevo. Random access codes only allow us to recover *each* of the n bits. Can they be much shorter? In small cases they can be: for instance, one can encode two classical bits into one qubit, in such a way that each of the two bits can be recovered with success probability 85% from that qubit (see Exercise 2). However, Nayak [107] proved that asymptotically quantum random access codes cannot be much shorter than classical.

Theorem 3 (Nayak) *Let $x \mapsto \rho_x$ be a quantum random access encoding of n-bit strings into m-qubit states such that, for each $i \in [n]$, we can decode X_i from $|\phi_X\rangle$ with success probability p*

[3] NB: unlike in most of these lecture notes, N need not equal 2^n in this chapter!

(averaged over a uniform choice of x and the measurement randomness). Then $m \geq (1 - H(p))n$, where $H(p) = -p \log p - (1-p)\log(1-p)$ is the binary entropy function.

The intuition of the proof is quite simple: since the quantum state allows us to predict the bit X_i with probability p_i, it reduces the "uncertainty" about X_i from 1 bit to $H(p_i)$ bits. Hence it contains at least $1 - H(p_i)$ bits of information about X_i. Since all n X_i's are independent, the state has to contain at least $\sum_{i=1}^{n}(1 - H(p_i))$ bits of information about X in total.

13.3 Lower bounds on locally decodable codes

Here we will give an application of quantum information theory to a *classical* problem.[4]

The development of error-correcting codes is one of the success stories of science in the second half of the 20th century. Such codes are eminently practical, and are widely used to protect information stored on discs, communication over channels, etc. From a theoretical perspective, there exist codes that are nearly optimal in a number of different respects simultaneously: they have constant rate, can protect against a constant noise-rate, and have linear-time encoding and decoding procedures. We refer to Trevisan's survey [126] for a complexity-oriented discussion of codes and their applications.

One drawback of ordinary error-correcting codes is that we cannot efficiently decode small parts of the encoded information. If we want to learn, say, the first bit of the encoded message then we usually still need to decode the whole encoded string. This is relevant in situations where we have encoded a very large string (say, a library of books, or a large database), but are only interested in recovering small pieces of it at any given time. Dividing the data into small blocks and encoding each block separately will not work: small chunks will be efficiently decodable but not error-correcting, since a tiny fraction of well-placed noise could wipe out the encoding of one chunk completely. There exist, however, error-correcting codes that are *locally decodable*, in the sense that we can efficiently recover individual bits of the encoded string.

Definition 1 *$C : \{0,1\}^n \to \{0,1\}^N$ is a (q, δ, ε)-*locally decodable code *(LDC) if there is a classical randomized decoding algorithm A such that*

1. *A makes at most q queries to an N-bit string y.*
2. *For all x and i, and all $y \in \{0,1\}^N$ with Hamming distance $d(C(x), y) \leq \delta N$ we have $\Pr[A^y(i) = x_i] \geq 1/2 + \varepsilon$.*

The notation $A^y(i)$ reflects that the decoder A has two different types of input. On the one hand there is the (possibly corrupted) codeword y, to which the decoder has oracle access and from which it can read at most q bits of its choice. On the other hand there is the index i of the bit that needs to be recovered, which is known fully to the decoder.

The main question about LDCs is the tradeoff between the codelength N and the number of queries q (which is a proxy for the decoding-time). This tradeoff is still not very well understood. The only case where we know the answer is the case of $q = 2$ queries (1-query LDCs don't exist once n is sufficiently large [81]). For $q = 2$ there is the Hadamard code: given $x \in \{0,1\}^n$, define a codeword of length $N = 2^n$ by writing down the bits $x \cdot z \bmod 2$, for all $z \in \{0,1\}^n$. One can

[4] There is a growing number of such applications of quantum tools to non-quantum problems. See [52] for a survey.

decode x_i with 2 queries as follows: choose $z \in \{0,1\}^n$ uniformly at random and query the (possibly corrupted) codeword at indices z and $z \oplus e_i$, where the latter denotes the string obtained from z by flipping its i-th bit. Individually, each of these two indices is uniformly distributed. Hence for each of them, the probability that the returned bit is corrupted is at most δ. By the union bound, with probability at least $1-2\delta$, both queries return the uncorrupted values. Adding these two bits mod 2 gives the correct answer:

$$C(x)_z \oplus C(x)_{z \oplus e_i} = (x \cdot z) \oplus (x \cdot (z \oplus e_i)) = x \cdot e_i = x_i.$$

Thus the Hadamard code is a $(2, \delta, 1/2 - 2\delta)$-LDC of exponential length.

The only superpolynomial *lower bound* known on the length of LDCs is for the case of 2 queries: there one needs an exponential codelength and hence the Hadamard code is essentially optimal. This is shown via a *quantum* argument [83]—despite the fact that the result is a purely classical result, about classical codes and classical decoders. The easiest way to present this argument is to assume the following fact, which states a kind of "normal form" for the decoder.

Fact 1 (Katz & Trevisan [81] + folklore) *For every (q, δ, ε)-LDC $C : \{0,1\}^n \to \{0,1\}^N$, and for each $i \in [n]$, there exists a set $\mathcal{M}_i$ of $\Omega(\delta\varepsilon N/q^2)$ disjoint tuples, each of at most q indices from $[N]$, and a bit $a_{i,t}$ for each tuple $t \in \mathcal{M}_i$, such that the following holds:*

$$\Pr_{x \in \{0,1\}^n} \left[x_i = a_{i,t} \oplus \sum_{j \in t} C(x)_j \right] \geq 1/2 + \Omega(\varepsilon/2^q), \tag{13.2}$$

where the probability is taken uniformly over x. Hence to decode x_i from $C(x)$, the decoder can just query the indices in a randomly chosen tuple t from $\mathcal{M}_i$, outputting the sum of those q bits and $a_{i,t}$.

Note that the above decoder for the Hadamard code is already of this form, with $\mathcal{M}_i = \{(z, z \oplus e_i)\}$. We omit the proof of Fact 1. It uses purely classical ideas and is not hard.

Now suppose $C : \{0,1\}^n \to \{0,1\}^N$ is a $(2, \delta, \varepsilon)$-LDC. We want to show that the codelength N must be exponentially large in n. Our strategy is to show that the following N-dimensional quantum encoding is in fact a quantum random access code for x (with some success probability $p > 1/2$):

$$x \mapsto |\phi_x\rangle = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} (-1)^{C(x)_j} |j\rangle.$$

Theorem 3 then implies that the number of qubits of this state (which is $\lceil \log N \rceil$) is at least $(1 - H(p))n = \Omega(n)$, and we are done.

Suppose we want to recover x_i from $|\phi_x\rangle$. We'll do this by a sequence of two measurements, as follows. We turn each $\mathcal{M}_i$ from Fact 1 into a projective measurement: for each pair $(j,k) \in \mathcal{M}_i$ form the projector $P_{jk} = |j\rangle\langle j| + |k\rangle\langle k|$, and let $P_{rest} = \sum_{j \notin \cup_{t \in M_i} t} |j\rangle\langle j|$ be the projector on the remaining indices. These $|\mathcal{M}_i| + 1$ projectors sum to the N-dimensional identity matrix, so they form a valid projective measurement. Applying this to $|\phi_x\rangle$ gives outcome (j,k) with probability $\|P_{jk}|\phi_x\rangle\|^2 = 2/N$ for each $(j,k) \in \mathcal{M}_i$. There are $|\mathcal{M}_i| = \Omega(\delta\varepsilon N)$ different (j,k)-pairs in $\mathcal{M}_i$, so the probability to see one of those as outcome of the measurement, is $|\mathcal{M}_i| \cdot 2/N = \Omega(\delta\varepsilon)$. With the remaining probability $r = 1 - \Omega(\delta\varepsilon)$, we'll get "rest" as outcome of the measurement. In the

latter case we didn't get anything useful from the measurement, so we'll just output a fair coin flip as our guess for x_i (then the output will equal x_i with probability exactly 1/2). In case we got one of the (j,k) as measurement outcome, the state has collapsed to the following useful superposition:

$$\frac{1}{\sqrt{2}}\left((-1)^{C(x)_j}|j\rangle + (-1)^{C(x)_k}|k\rangle\right) = \frac{(-1)^{C(x)_j}}{\sqrt{2}}\left(|j\rangle + (-1)^{C(x)_j \oplus C(x)_k}|k\rangle\right)$$

We know what j and k are, because it is the outcome of the measurement on $|\phi_x\rangle$. Doing a 2-outcome measurement in the basis $\frac{1}{\sqrt{2}}(|j\rangle \pm |k\rangle)$ now gives us the value $C(x)_j \oplus C(x)_k$ with probability 1. By Eq. (13.2), if we add the bit $a_{i,(j,k)}$ to this, we get x_i with probability at least $1/2 + \Omega(\varepsilon)$. The success probability of recovering x_i, averaged over all x, is

$$p \geq \frac{1}{2}r + \left(\frac{1}{2} + \Omega(\varepsilon)\right)(1-r) = \frac{1}{2} + \Omega(\delta\varepsilon^2).$$

Thus we have constructed a random access code that encodes n bits into $\log N$ qubits, and has success probability at least p. Applying Theorem 3 and using that $1 - H(1/2+\eta) = \Theta(\eta^2)$ for $\eta \in [0, 1/2]$, we obtain the following:

Theorem 4 *If $C : \{0,1\}^n \to \{0,1\}^N$ is a $(2,\delta,\varepsilon)$-locally decodable code, then $N \geq 2^{\Omega(\delta^2\varepsilon^4 n)}$.*

Exercises

1. (a) Give the density matrix that corresponds to a 50-50 mixture of $|0\rangle$ and $|1\rangle$.
   (b) Give the density matrix that corresponds to a 50-50 mixture of $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

2. (a) (H) Give a quantum random access code that encodes 2 classical bits into 1 qubit, such that each of the two classical bits can be recovered from the quantum encoding with success probability $p \geq 0.85$.
   (b) Prove an upper bound of $1/2 + O(1/\sqrt{n})$ on the success probability p for a random access code that encodes n classical bits into 1 qubit.

3. (H) Teleportation transfers an arbitrary unknown qubit from Alice to Bob, using 1 EPR-pair and 2 classical bits of communication from Alice to Bob (see Section 1.5). Prove that these 2 bits of communication are necessary, i.e., you cannot teleport an arbitrary unknown qubit using 1 EPR-pair and only 1 classical bit of communication.

4. Consider the Hadamard code C that encodes $n = 2$ bits x_1x_2 into a codeword of $N = 4$ bits.
   (a) Give the 4-bit codeword $C(11)$.
   (b) What are the states $|\phi_x\rangle$ that arise as quantum random access code when we apply the LDC lower bound proof of Section 13.3 to C?
   (c) What is the measurement used for recovering x_2 from $|\phi_x\rangle$ at the end of that proof? You may either describe this as a sequence of two projective measurements, or as one (combined) projective measurement.

Chapter 14

Quantum Communication Complexity

Communication complexity was first introduced by Yao [133], and has been studied extensively in the area of theoretical computer science and has deep connections with seemingly unrelated areas, such as VLSI design, circuit lower bounds, lower bounds on branching programs, size of data structures, and bounds on the length of logical proof systems, to name just a few.

14.1 Classical communication complexity

First we sketch the setting for classical communication complexity. Alice and Bob want to compute some function $f : \mathcal{D} \to \{0,1\}$, where $\mathcal{D} \subseteq X \times Y$.[1] Alice receives input $x \in X$, Bob receives input $y \in Y$, with $(x,y) \in \mathcal{D}$. A typical situation, illustrated in Fig. 14.1, is where $X = Y = \{0,1\}^n$, so both Alice and Bob receive an n-bit input string. As the value $f(x,y)$ will generally depend on both x and y, some communication between Alice and Bob is required in order for them to be able to compute $f(x,y)$. We are interested in the *minimal* amount of communication they need.

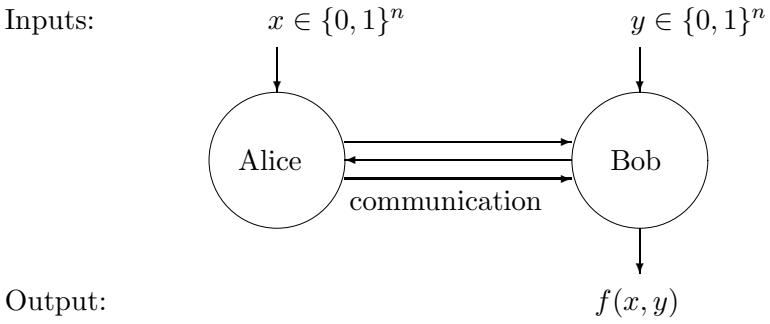

Figure 14.1: Alice and Bob solving a communication complexity problem

A communication *protocol* is a distributed algorithm where first Alice does some individual computation, and then sends a message (of one or more bits) to Bob, then Bob does some computation and sends a message to Alice, etc. Each message is called a *round*. After one or more rounds the protocol terminates and one of the parties (let's say Bob) outputs some value that should be

[1] If the domain $\mathcal{D}$ equals $X \times Y$ then f is called a *total* function, otherwise it is called a *partial* or *promise* function.

$f(x,y)$. The *cost* of a protocol is the total number of bits communicated on the worst-case input. A *deterministic* protocol for f always has to output the right value $f(x,y)$ for all $(x,y) \in \mathcal{D}$. In a *bounded-error* protocol, Alice and Bob may flip coins and the protocol has to output the right value $f(x,y)$ with probability $\geq 2/3$ for all $(x,y) \in \mathcal{D}$. We could either allow Alice and Bob to toss coins individually (local randomness, or "private coin") or jointly (shared randomness, or "public coin"). A public coin can simulate a private coin and is potentially more powerful. However, Newman's theorem [108] says that having a public coin can save at most $O(\log n)$ bits of communication, compared to a protocol with a private coin.

To illustrate the power of randomness, let us give a simple yet efficient bounded-error protocol for the equality problem, where the goal for Alice is to determine whether her n-bit input is the same as Bob's or not: $f(x,y) = 1$ if $x = y$, and $f(x,y) = 0$ otherwise. Alice and Bob jointly toss a random string $r \in \{0,1\}^n$. Alice sends the bit $a = x \cdot r$ to Bob (where '$\cdot$' is inner product mod 2). Bob computes $b = y \cdot r$ and compares this with a. If $x = y$ then $a = b$, but if $x \neq y$ then $a \neq b$ with probability 1/2. Repeating this a few times, Alice and Bob can decide equality with small error probability using $O(n)$ public coin flips and a constant amount of communication. This protocol uses public coins, but note that Newman's theorem implies that there exists an $O(\log n)$-bit protocol that uses a private coin (see Exercise 6 for an explicit protocol). Note that the correct output of the equality function depends on all n bits of x, but Bob does not need to learn all n bits of x in order to be able to decide equality with high success probability. In contrast, one can show that *deterministic* protocols for the equality problem need n bits of communication, so then Alice might as well just send x to Bob.

14.2 The quantum question

Now what happens if we give Alice and Bob a quantum computer and allow them to send each other qubits and/or to make use of EPR-pairs that they share at the start of the protocol?

Formally speaking, we can model a quantum protocol as follows. The total state consists of 3 parts: Alice's private space, the channel, and Bob's private space. The starting state is $|x\rangle|0\rangle|y\rangle$: Alice gets x, the channel is initialized to 0, and Bob gets y. Now Alice applies a unitary transformation to her space and the channel. This corresponds to her private computation as well as to putting a message on the channel (the length of this message is the number of channel-qubits affected by Alice's operation). Then Bob applies a unitary transformation to his space and the channel, etc. At the end of the protocol Alice or Bob makes a measurement to determine the output of the protocol. This model was introduced by Yao [134].

In the second model, introduced by Cleve and Buhrman [45], Alice and Bob share an unlimited number of EPR-pairs at the start of the protocol, but now they communicate via a *classical* channel: the channel has to be in a classical state throughout the protocol. We only count the communication, not the number of EPR-pairs used. Protocols of this kind can simulate protocols of the first kind with only a factor 2 overhead: using teleportation, the parties can send each other a qubit using an EPR-pair and two classical bits of communication. Hence the qubit-protocols that we describe below also immediately yield protocols that work with entanglement and a classical channel. Note that an EPR-pair can simulate a public coin toss: if Alice and Bob each measure their half of the pair of qubits, they get the same random bit.

The third variant combines the strengths of the other two: here Alice and Bob start out with an unlimited number of EPR-pairs *and* they are allowed to communicate qubits. This third kind

of communication complexity is in fact equivalent to the second, up to a factor of 2, again by teleportation.

Before continuing to study this model, we first have to face an important question: *is there anything to be gained here?* At first sight, the following argument seems to rule out any significant gain. Suppose that in the classical world k bits have to be communicated in order to compute f. Since Holevo's theorem says that k qubits cannot contain more information than k classical bits, it seems that the quantum communication complexity should be roughly k qubits as well (maybe $k/2$ to account for superdense coding, but not less). Surprisingly (and fortunately for us), this argument is false, and quantum communication can sometimes be much less than classical communication complexity. The information-theoretic argument via Holevo's theorem fails, because Alice and Bob do not need to communicate the information in the k bits of the classical protocol; they are only interested in the value $f(x, y)$, which is just 1 bit. Below we will go over four of the main examples that have so far been found of differences between quantum and classical communication complexity.

14.3 Example 1: Distributed Deutsch-Jozsa

The first impressively large gaps between quantum and classical communication complexity were exhibited by Buhrman, Cleve, and Wigderson [35]. Their protocols are distributed versions of known quantum query algorithms, like the Deutsch-Jozsa and Grover algorithms. Let us start with the first one. It is actually explained most easily in a direct way, without reference to the Deutsch-Jozsa algorithm (though that is where the idea came from). The problem is a promise version of the equality problem. Suppose the n-bit inputs x and y are restricted to the following case:

Distributed Deutsch-Jozsa: either $x = y$, or x and y differ in exactly $n/2$ positions

Note that this promise only makes sense if n is an even number, otherwise $n/2$ would not be integer. In fact it will be convenient to assume n a power of 2. Here is a simple quantum protocol to solve this promise version of equality using only $\log n$ qubits:

1. Alice sends Bob the $\log n$-qubit state $\frac{1}{\sqrt{n}} \sum_{i=1}^{n} (-1)^{x_i} |i\rangle$, which she can prepare unitarily from x and $\log n$ $|0\rangle$-qubits.

2. Bob applies the unitary map $|i\rangle \mapsto (-1)^{y_i} |i\rangle$ to the state, applies a Hadamard transform to each qubit (for this it is convenient to view i as a $\log n$-bit string), and measures the resulting $\log n$-qubit state.

3. Bob outputs 1 if the measurement gave $\left|0^{\log n}\right\rangle$ and outputs 0 otherwise.

It is clear that this protocol only communicates $\log n$ qubits, but why does it work? Note that the state that Bob measures is

$$H^{\otimes \log n} \left(\frac{1}{\sqrt{n}} \sum_{i=1}^{n} (-1)^{x_i + y_i} |i\rangle \right) = \frac{1}{n} \sum_{i=1}^{n} (-1)^{x_i + y_i} \sum_{j \in \{0,1\}^{\log n}} (-1)^{i \cdot j} |j\rangle$$

This superposition looks rather unwieldy, but consider the amplitude of the $\left|0^{\log n}\right\rangle$ basis state. It is $\frac{1}{n} \sum_{i=1}^{n} (-1)^{x_i + y_i}$, which is 1 if $x = y$ and 0 otherwise because the promise now guarantees that x and y differ in exactly $n/2$ of the bits! Hence Bob will always give the correct answer.

What about efficient *classical* protocols (without entanglement) for this problem? Proving lower bounds on communication complexity often requires a very technical combinatorial analysis. Buhrman, Cleve, and Wigderson used a deep combinatorial result from [62] to prove that every classical errorless protocol for this problem needs to send at least $0.007n$ bits.

This $\log n$-qubits-vs-$0.007n$-bits example was the first exponentially large separation of quantum and classical communication complexity. Notice, however, that the difference disappears if we move to the *bounded-error* setting, allowing the protocol to have some small error probability. We can use the randomized protocol for equality discussed above or even simpler: Alice can just send a few (i, x_i) pairs to Bob, who then compares the x_i's with his y_i's. If $x = y$ he will not see a difference, but if x and y differ in $n/2$ positions, then Bob will probably detect this. Hence $O(\log n)$ classical bits of communication suffice in the bounded-error setting, in sharp contrast to the errorless setting.

14.4 Example 2: The Intersection problem

Now consider the Intersection function, which is 1 if $x_i = y_i = 1$ for at least one i. Buhrman, Cleve, and Wigderson [35] also presented an efficient quantum protocol for this, based on Grover's search algorithm (Chapter 7). We can solve Intersection if we can solve the following search problem: find some i such that $x_i = y_i = 1$, if such an i exists.[2] We want to find a solution to the search problem on the string $z = x \wedge y$ (which is the bit-wise AND of x and y), since $z_i = 1$ whenever both $x_i = 1$ and $y_i = 1$. The idea is now to let Alice run Grover's algorithm to search for such a solution. Clearly, she can prepare the uniform starting state herself. She can also apply the unitaries H and R herself. The only thing where she needs Bob's help, is in implementing $O_{z,\pm}$. This they do as follows. Whenever Alice wants to apply $O_{z,\pm}$ to a state

$$|\phi\rangle = \sum_{i=1}^{n} \alpha_i |i\rangle,$$

she tags on her x_i in an extra qubit and sends Bob the state

$$\sum_{i=1}^{n} \alpha_i |i\rangle|x_i\rangle.$$

Bob applies the unitary map

$$|i\rangle|x_i\rangle \mapsto (-1)^{x_i \wedge y_i} |i\rangle|x_i\rangle$$

and sends back the result. Alice sets the last qubit back to $|0\rangle$ (which she can do unitarily because she has x), and now she has the state $O_{z,\pm}|\phi\rangle$! Thus we can simulate $O_{z,\pm}$ using 2 messages of $\log(n) + 1$ qubits each. Thus Alice and Bob can run Grover's algorithm to find an intersection, using $O(\sqrt{n})$ messages of $O(\log n)$ qubits each, for total communication of $O(\sqrt{n} \log n)$ qubits. Later Aaronson and Ambainis [1] gave a more complicated protocol that uses $O(\sqrt{n})$ qubits of communication.

What about lower bounds? It is a well-known result of classical communication complexity that classical bounded-error protocols for the Intersection problem need about n bits of communication.

[2] This is sometimes called the *appointment-scheduling problem*: view x and y as Alice's and Bob's agendas, respectively, with a 1 at the ith bit indicating that timeslot i is available. Then the goal is to find a timeslot where Alice and Bob are both available, so they can schedule an appointment.

Thus we have a quadratic quantum-classical separation for this problem. Could there be a quantum protocol that uses much less than $\sqrt{n}$ qubits of communication? This question was open for quite a few years after [35] appeared, until finally Razborov [112] showed that any bounded-error quantum protocol for Intersection needs to communicate about $\sqrt{n}$ qubits.

14.5 Example 3: The vector-in-subspace problem

Notice the contrast between the examples of the last two sections. For the Distributed Deutsch-Jozsa problem we get an *exponential* quantum-classical separation, but the separation only holds if we require the classical protocol to be errorless. On the other hand, the gap for the disjointness function is only *quadratic*, but it holds even if we allow classical protocols to have some error probability.

Here is a function where the quantum-classical separation has both features: the quantum protocol is exponentially better than the classical protocol, even if the latter is allowed some error:

> Alice receives a unit vector $v \in \mathbb{R}^m$
> Bob receives two m-dimensional projectors P_0 and P_1 such that $P_0 + P_1 = I$
> Promise: either $P_0 v = v$ or $P_1 v = v$.
> Question: which of the two?

As stated, this is a problem with continuous input, but it can be discretized in a natural way by approximating each real number by $O(\log m)$ bits. Alice and Bob's input is now $n = O(m^2 \log m)$ bits long. There is a simple yet efficient 1-round quantum protocol for this problem: Alice views v as a $\log m$-qubit state and sends this to Bob; Bob measures with operators P_0 and P_1, and outputs the result. This takes only $\log m = O(\log n)$ qubits of communication.

The efficiency of this protocol comes from the fact that an m-dimensional unit vector can be "compressed" or "represented" as a $\log m$-qubit state. Similar compression is not possible with classical bits, which suggests that any classical protocol will have to send the vector v more or less literally and hence will require a lot of communication. This turns out to be true, but the proof is quite hard [86]. It shows that any bounded-error protocol needs to send $\Omega(m^{1/3})$ bits.

14.6 Example 4: Quantum fingerprinting

The examples of the previous section were either exponential quantum improvements for promise problems (Deutsch-Jozsa and vector-in-subspace) or polynomial improvements for total problems (disjointness). We will now give an exponential improvement for the total problem of equality-testing, but in a restricted setting called the *simultaneous message passing* (SMP) model. Alice and Bob receive n-bit input x and y, respectively. They do not have any shared resources like shared randomness or an entangled state, but they do have local randomness. They don't communicate with each other directly, but instead send a single message to a third party, called the Referee. The Referee, upon receiving message m_A from Alice and m_B from Bob, should output the value $f(x, y)$. The goal is to compute $f(x, y)$ with a minimal amount of communication from Alice and Bob to the Referee.

We will see that for the equality problem there is an exponential savings in communication when qubits are used instead of classical bits. Classically, the problem of the bounded-error communication complexity of equality in the SMP model was first raised by Yao [133], and was open

for almost twenty years until Newman and Szegedy [109] exhibited a lower bound of $\Omega(\sqrt{n})$ bits. This is tight, since Ambainis [4] constructed a bounded-error protocol for this problem where the messages are $O(\sqrt{n})$ bits long (see Exercise 5). In contrast, in the quantum setting this problem can be solved with very little communication: only $O(\log n)$ qubits suffice [34].

The quantum trick is to associate each $x \in \{0,1\}^n$ with a short quantum state $|\phi_x\rangle$, called the *quantum fingerprint* of x. Just like with physical fingerprints, the idea is that a quantum fingerprint is a small object that doesn't contain very much information about the object x, but that suffices for testing if the fingerprinted object equals some other fingerprinted object. As we will see below, we can do such testing if the fingerprints are pairwise almost orthogonal. More precisely, an (n, m, ε)-*quantum fingerprinting scheme* maps n-bit string x to m-qubit state $|\phi_x\rangle$ with the property that for all distinct $x, y \in \{0,1\}^n$, we have $|\langle\phi_x|\phi_y\rangle| \leq \varepsilon$.

We will now show how to obtain a specific $(n, m, 0.02)$-quantum fingerprinting scheme from an error-correcting code $C : \{0,1\}^n \to \{0,1\}^N$ where $m = \log N \approx \log n$. There exist codes where $N = O(n)$ and any two codewords $C(x)$ and $C(y)$ have Hamming distance close to $N/2$, say $d(C(x), C(y)) \in [0.49N, 0.51N]$ (we won't prove this here, but for instance a random linear code will work). Define the quantum fingerprint of x as follows:

$$|\phi_x\rangle = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} (-1)^{C(x)_j} |j\rangle.$$

This is a unit vector in an N-dimensional space, so it corresponds to only $\lceil \log N \rceil = \log n + O(1)$ qubits. For distinct x and y, the corresponding fingerprints will have small inner product:

$$\langle\phi_x|\phi_y\rangle = \frac{1}{N} \sum_{j=1}^{N} (-1)^{C(x)_j + C(y)_j} = \frac{N - 2d(C(x), C(y))}{N} \in [-0.02, 0.02].$$

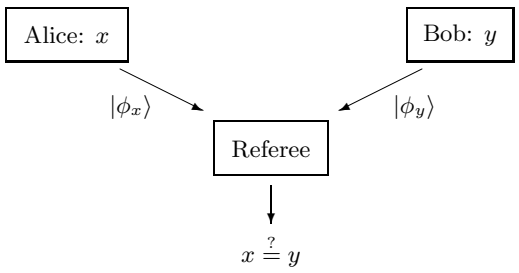

Figure 14.2: Quantum fingerprinting protocol for the equality problem

The quantum protocol is very simple (see Figure 14.2): Alice and Bob send quantum fingerprints of x and y to the Referee, respectively. The referee now has to determine whether $x = y$ (which corresponds to $\langle\phi_x|\phi_y\rangle = 1$) or $x \neq y$ (which corresponds to $\langle\phi_x|\phi_y\rangle \in [-0.02, 0.02]$). The following test (Figure 14.3), sometimes called the *SWAP-test*, accomplishes this with small error probability.

This circuit first applies a Hadamard transform to a qubit that is initially $|0\rangle$, then SWAPs the other two registers conditioned on the value of the first qubit being $|1\rangle$, then applies another Hadamard transform to the first qubit and measures it. Here SWAP is the operation that swaps the two registers: $|\phi_x\rangle|\phi_y\rangle \mapsto |\phi_y\rangle|\phi_x\rangle$. The Referee receives $|\phi_x\rangle$ from Alice and $|\phi_y\rangle$ from Bob and applies the test to these two states. An easy calculation reveals that the outcome of the measurement

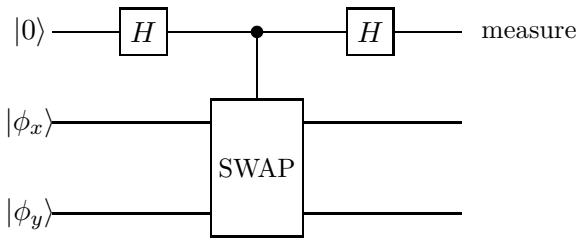

Figure 14.3: Quantum circuit to test if $|\phi_x\rangle = |\phi_y\rangle$ or $|\langle\phi_x|\phi_y\rangle|$ is small

is 1 with probability $(1-|\langle\phi_x|\phi_y\rangle|^2)/2$. Hence if $|\phi_x\rangle = |\phi_y\rangle$ then we observe a 1 with probability 0, but if $|\langle\phi_x|\phi_y\rangle|$ is close to 0 then we observe a 1 with probability close to 1/2. Repeating this procedure with several individual fingerprints can make the error probability arbitrarily close to 0.

Exercises

1. (H) Prove that classical deterministic protocols with one message (from Alice to Bob), need to send n bits to solve the equality problem.

2. (a) (H) Show that if $|\phi\rangle$ and $|\psi\rangle$ are non-orthogonal states (i.e., $\langle\phi|\psi\rangle \neq 0$), then there is no two-outcome projective measurement that perfectly distinguishes these two states, in the sense that applying the measurement on $|\phi\rangle$ always gives a different outcome from applying the same measurement to $|\psi\rangle$.

   (b) Prove that quantum protocols with one message (from Alice to Bob), need to send at least n qubits to solve the equality problem (on n-bit inputs) with success probability 1 on every input.

   (c) (H) Prove that quantum protocols with one message (from Alice to Bob), need to send at least $\log n$ qubits to solve the distributed Deutsch-Jozsa problem with success probability 1 on every input.

3. (H) Consider *one-round* quantum communication complexity. Alice gets input $x \in \{0,1\}^n$, Bob gets input $y \in \{0,1\}^n$, and they want to compute some Boolean function $f(x,y)$ of their inputs. Assume that all rows of the communication matrix are different, i.e., for all x and x' there is a y such that $f(x,y) \neq f(x',y)$. They are allowed only one round of communication: Alice sends a quantum message to Bob and Bob must then be able to give the right answer with probability 1. Prove that Alice needs to send n qubits to Bob for this. You may assume that Alice's messages are pure states (this is without loss of generality).

4. (H) The *disjointness problem* of communication complexity is the following function: Alice receives an $x \in \{0,1\}^n$, Bob receives $y \in \{0,1\}^n$, and $f(x,y) = 0$ if there is an i such that $x_i = y_i = 1$, and $f(x,y) = 1$ otherwise (i.e., f says whether x and y represent disjoint subsets of $[n]$). Suppose there exists an m-qubit one-way protocol that solves this problem, so where Alice sends Bob m qubits and then Bob outputs $f(x,y)$ with probability at least 2/3. Prove the lower bound $m = \Omega(n)$ on the number of qubits sent.

5. Consider an error-correcting code $C : \{0,1\}^n \to \{0,1\}^N$ where $N = O(n)$, N is a square, and any two distinct codewords are at Hamming distance $d(C(x), C(y)) \in [0.49N, 0.51N]$ (such codes exist, but you don't have to prove that).

   (a) View the codeword $C(x)$ as a $\sqrt{N} \times \sqrt{N}$ matrix. Show that if you choose a row uniformly at random and choose a column uniformly at random, then the unique index i where these row and column intersect, is uniformly distributed over $i \in \{1, \ldots, N\}$.

   (b) (H) Give a classical bounded-error SMP-protocol for the equality problem where Alice and Bob each send $O(\sqrt{n})$ bits to the Referee.

6. Alice and Bob want to solve the equality problem on n-bit inputs x and y (i.e., decide whether $x = y$). They do not share randomness or entanglement but can use local (private) randomness.

   (a) (H) Fix a prime number $p \in [3n, 6n]$, then the set $\mathbb{F}_p$ of integers modulo p is a finite field (i.e., it has a well-defined addition and multiplication). For $x = (x_0, \ldots, x_{n-1}) \in \{0,1\}^n$, define the univariate polynomial $P_x : \mathbb{F}_p \to \mathbb{F}_p$ of degree $< n$ as $P_x(t) = \sum_{i=0}^{n-1} x_i t^i$ (note that the n bits of x are used as coefficients here, not as the argument of the polynomial). Show that for distinct n-bit strings x and y, we have $\Pr_{t \in \mathbb{F}_p}[P_x(t) = P_y(t)] \leq 1/3$, where the probability is taken over a uniformly random $t \in \mathbb{F}_p$.

   (b) Use (a) to give a classical communication protocol where Alice sends an $O(\log n)$-bit message to Bob, and Bob can decide whether $x = y$ with success probability $\geq 2/3$.

   (c) Use (a) to give a quantum fingerprinting scheme $x \mapsto |\phi_x\rangle$, where quantum state $|\phi_x\rangle$ has $O(\log n)$ qubits, and $|\langle\phi_x|\phi_y\rangle| \in [0, 1/3]$ for all distinct n-bit strings x and y (prove the latter property explicitly, it's not enough to write down only the states).

7. Suppose Alice and Bob each have n-bit agendas, and they know that for exactly 25% of the timeslots they are both free. Give a quantum protocol that finds such a timeslot with probability 1, using only $O(\log n)$ qubits of communication.

8. The inner product problem in communication complexity is the function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ defined by $f(x,y) = \sum_{i=1}^n x_i y_i \bmod 2$. Suppose there exists a quantum protocol P for Alice and Bob that uses q qubits of communication (possibly using multiple messages between Alice and Bob) and computes the inner product function with success probability 1 (on every possible inputs x, y). The protocol does not assume any shared entangled state at the start.

   (a) Give a quantum protocol that uses $2q$ qubits of communication and implements the $2n$-qubit map $|x\rangle_A|y\rangle_B \mapsto (-1)^{x \cdot y}|x\rangle_A|y\rangle_B$ (possibly with some auxiliary qubits for each of Alice and Bob; these should start and end in state $|0\rangle$).

   (b) (H) Give a quantum protocol where Alice transmits x to Bob using $2q$ qubits of communication.

   (c) Derive a lower bound on q from (b) and Holevo's theorem.

9. Consider the following problem in communication complexity. Alice's input has two parts: a unit vector $v \in \mathbb{R}^m$ and two orthogonal projectors P_0 and P_1. Bob's input is an $m \times m$ unitary U. They are promised that the vector Uv either lies in the subspace corresponding to

P_0 (i.e., $P_0 U v = v$) or in the subspace corresponding to P_1 (i.e., $P_1 U v = v$), and the problem for Alice and Bob is to find out which of these two cases holds.

(a) Give a quantum protocol that uses two messages of $O(\log m)$ qubits (one message from Alice to Bob and one from Bob to Alice) to solve this problem with success probability 1.

(b) (H) Show that there exists a constant $c > 0$ such that classical protocols need to send $\Omega(m^c)$ bits of communication to solve this problem with error probability $\leq 1/3$, even when they are allowed to send many messages.

10. (H) Consider the following communication complexity problem, called the "Hidden Matching Problem." Alice's input is some $x \in \{0,1\}^n$. Bob's input is a matching M, i.e., a partition of $\{1, \ldots, n\}$ into $n/2$ disjoint pairs (assume n is a power of 2). Their goal is that Bob outputs a pair $(i, j) \in M$ together with the parity $x_i \oplus x_j$ of the two bits indexed by that pair. It doesn't matter which pair $(i, j) \in M$ Bob outputs, as long as the additional bit of output equals the parity of the two indexed bits of x. Show that they can solve this problem with success probability 1 using only a message of $\log n$ qubits from Alice to Bob (and no communication from Bob to Alice).

11. (a) Suppose you have a state $\frac{1}{\sqrt{2}}(|0\rangle|\phi\rangle + |1\rangle|\psi\rangle)$, where $|\phi\rangle$ and $|\psi\rangle$ are quantum states with real amplitudes. Suppose you apply a Hadamard gate to its first qubit and then measure that first qubit. Show that the probability of measurement outcome 0 is $\frac{1}{2}(1 + \langle\phi|\psi\rangle)$.

(b) Suppose H is a subgroup of a finite group G, and $g \in G$ some element. Show (1) if $g \in H$ then the cosets $g \circ H$ and H are equal
and (2) if $g \notin H$ then the cosets $g \circ H$ and H are disjoint.

(c) Suppose you are given quantum state $|\psi_H\rangle = \frac{1}{\sqrt{H}} \sum_{h \in H} |h\rangle$ (for an unknown $H \leq G$), and an element $g \in G$. You may assume you have a unitary A available that implements the group operation, $A : |g, h\rangle \mapsto |g, g \circ h\rangle$, and you may also apply a controlled version of A. Give an algorithm that acts on $|\psi_H\rangle$ and possibly some auxiliary qubits, and that outputs 0 with probability 1 if $g \in H$, and outputs 0 with probability $\leq 1/2$ if $g \notin H$.

(d) (H) Consider the following communication complexity problem. Alice and Bob both know a finite group G, Alice gets as input some subgroup $H \leq G$ (for instance in the form of a generating set for H) and Bob gets input $g \in G$. Give a one-way quantum protocol where Alice sends to Bob a message of $O(\log |G|)$ qubits, and then Bob decides with success probability $\geq 2/3$ whether $g \in H$.

Chapter 15

Entanglement and Non-Locality

15.1 Quantum non-locality

Entangled states are those that cannot be written as a tensor product of separate states. The most famous one is the EPR-pair:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

Suppose Alice has the first qubit of the pair, and Bob has the second. If Alice measures her qubit in the computational basis and gets outcome $b \in \{0, 1\}$, then the state collapses to $|bb\rangle$. Similarly, if Alice measures her qubit in some other basis, this will collapse the joint state (including Bob's qubit) to some state that depends on her measurement basis as well as its outcome. Somehow Alice's action seems to have an *instantaneous* effect on Bob's side—even if the two qubits are light-years apart! This was a great bother to Einstein, whose theory of relativity posits that information and causation cannot travel faster than the speed of light. Einstein called such effects of entanglement "spooky action at a distance" (in German: "spukhafte Fernwirkungen"), and viewed it as a fundamental problem for quantum mechanics [56]. In his view, quantum mechanics should be replaced by some "local realist" physical theory that would still have the same predictive power as quantum mechanics. Here "local" means that information and causation act locally, not faster than light, and "realistic" means that physical systems have definite, well-defined properties (even if those properties may be unknown to us).

Note that the above experiment where Alice measures her half of the EPR-pair doesn't actually violate locality: no information is transfered from Alice and Bob. From Bob's perspective there is no difference between the situation where Alice measured and the situation where she didn't.[1] For this experiment, a shared coin flip between Alice and Bob is a local realist physical model that has exactly the same observable consequences as measuring the qubits of the EPR-pair in the computational basis: a 50-50 distribution on outcomes $|00\rangle$ and $|11\rangle$. This shared-coin-flip model is *local* because no information is transfered between Alice and Bob, and it's *realist* because the coin flip has a definite outcome (even if that outcome is unknown to Alice and Bob before they measure).

Given this example, one might hope (and Einstein expected) that any kind of behavior that comes from entangled states can be replaced by some local realist physical model. This way, quantum mechanics could be replaced by an alternative physical theory with less counter-intuitive

[1] In fact, one can show that entanglement can never replace communication.

behavior. Surprisingly, in the 1960s, John Bell [17] devised entanglement-based experiments whose behavior *cannot be reproduced by any local realist theory*. In other words, we can let Alice and Bob do certain measurements on an entangled state, and the resulting distributions on their outputs predicted by quantum mechanics, *cannot be obtained* from any local realist theory. This phenomenon is known as "quantum non-locality." It could of course be that the quantum mechanical predictions of the resulting correlations are just wrong. However, in the early 1980s, such experiments were actually done by Aspect and others [11], and they gave the outcomes that quantum mechanics predicted.[2] Note that such experiments don't *prove* quantum mechanics, but they *disprove* any local realist physical theory.[3]

Such experiments, which realize correlations that are *provably impossible* to realize with local realist models, are among the deepest and most philosophical results of 20th century physics: the commonsense idea of local realism is most probably false! Since Bell's seminal work, the concept of quantum non-locality has been extensively studied, by physicists, philosophers, and more recently by computer scientists.

In the next sections we review some interesting examples. The two-party setting of these examples is illustrated in Fig. 15.1: Alice receives input x and Bob receives input y, and they produce outputs a and b, respectively, that have to be correlated in a certain way (which depends on the game). They are not allowed to communicate. In physics language, we could assume they are "space-like separated," which means that they are so far apart that they cannot influence each other during the course of the experiment (assuming information doesn't travel faster than the speed of light). In the classical scenario they are allowed to share a random variable. Physicists would call this the "local hidden variable" that gives properties their definite value (that value may be unknown to the experimenter). This setting captures all local realist models. In the quantum model Alice and Bob are allowed to share entangled states, such as EPR-pairs. The goal is to show that entanglement-based strategies can do things that local realist strategies cannot.

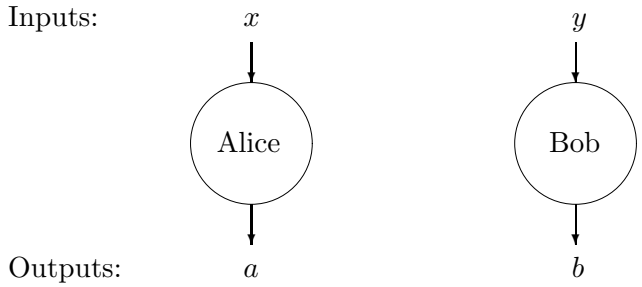

Figure 15.1: The non-locality scenario involving two parties: Alice and Bob receive inputs x and y, respectively, and are required to produce outputs a and b that satisfy certain conditions. Once the inputs are received, no communication is permitted between the parties.

[2]Modulo some technical "loopholes" due to imperfect photon sources, measurement devices, Alice and Bob not being sufficiently far apart etc. These are still hotly debated, but most people accept that Aspect's and later experiments are convincing, and kill any hope of a complete local-realist explanation of nature. Recently [75] an experiment was done that simultaneously closed the two most important loopholes.

[3]Despite its name, non-locality doesn't disprove locality, but rather disproves the conjunction of locality and realism—at least one of the two assumptions has to fail.

15.2 CHSH: Clauser-Horne-Shimony-Holt

In the CHSH game [43] Alice and Bob receive input bits x and y, and their goal is to output bits a and b, respectively, such that

$$a \oplus b = x \wedge y, \tag{15.1}$$

('$\wedge$' is logical AND; '$\oplus$' is parity, i.e. addition mod 2) or, failing that, to satisfy this condition with as high a probability as possible.

First consider the case of classical *deterministic* strategies, so without any randomness. For these, Alice's output bit depends solely on her input bit x, and similarly for Bob. Let a_0 be the bit that Alice outputs if her input is $x = 0$, and a_1 the bit she outputs if $x = 1$. Let b_0, b_1 be the outputs Bob gives on inputs $y = 0$ and $y = 1$, respectively. These four bits completely characterize any deterministic strategy. Condition (15.1) becomes

$$\begin{aligned} a_0 \oplus b_0 &= 0, \\ a_0 \oplus b_1 &= 0, \\ a_1 \oplus b_0 &= 0, \\ a_1 \oplus b_1 &= 1. \end{aligned} \tag{15.2}$$

It is impossible to satisfy all four equations simultaneously, since summing them modulo 2 yields $0 = 1$. Therefore it is impossible to satisfy Condition (15.1) perfectly. Since a probabilistic strategy (where Alice and Bob share randomness) is a probability distribution over deterministic strategies, it follows that no probabilistic strategy can have success probability better than 3/4 on every possible input (the 3/4 can be achieved simultaneously for every input, see Exercise 3).[4]

Now consider the same problem but where Alice and Bob are supplied with a shared two-qubit system initialized to the entangled state

$$\tfrac{1}{\sqrt{2}}(|00\rangle - |11\rangle).$$

Such a state can easily be obtained from an EPR-pair, for instance if Alice applies a Z to her qubit. Now the parties can produce outputs that satisfy Condition (15.1) with probability $\cos(\pi/8)^2 \approx 0.85$ (higher than what is possible in the classical case), as follows. Recall the unitary operation that rotates the qubit by angle θ: $R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$. If $x = 0$ then Alice applies $R(-\pi/16)$ to her qubit; and if $x = 1$ she applies $R(3\pi/16)$. Then Alice measures her qubit in the computational basis and outputs the resulting bit a. Bob's procedure is the same, depending on his input bit y. It is straightforward to calculate that if Alice rotates by θ_A and Bob rotates by θ_B, the state becomes

$$\frac{1}{\sqrt{2}}\left(\cos(\theta_A + \theta_B)(|00\rangle - |11\rangle) + \sin(\theta_A + \theta_B)(|01\rangle + |10\rangle)\right).$$

After the measurements, the probability that $a \oplus b = 0$ is $\cos(\theta_A + \theta_B)^2$. Note that if $x \wedge y = 0$ then $\theta_A + \theta_B = \pm\pi/8$, while if $x \wedge y = 1$ then $\theta_A + \theta_B = 3\pi/8$. Hence Condition 15.1 is satisfied with probability $\cos(\pi/8)^2$ for all four input possibilities, showing that quantum entanglement allows Alice and Bob to win the game with a probability that's higher than what the best classical strategy can achieve. Tsirelson [42] showed that $\cos(\pi/8)^2$ is the best that quantum strategies can do for CHSH, even if they are allowed to use much more entanglement than one EPR-pair (see Exercise 5).

[4] Such statements, upper bounding the optimal success probability of classical strategies for a specific game, are known as *Bell inequalities*. This specific one is called the *CHSH inequality*.

15.3 Magic square game

Is there a game where the quantum protocol *always* succeeds, while the best classical success probability is bounded below 1? A particularly elegant example is the following *magic square game* [10]. Consider the problem of labeling the entries of a 3×3 matrix with bits so that the parity of each row is even, whereas the parity of each column is odd. This is clearly impossible: if the parity of each row is even then the sum of the 9 bits is 0 mod 2, but if the parity of each column is odd then the sum of the 9 bits is 1 mod 2. The two matrices

0	0	0
0	0	0
1	1	0

0	0	0
0	0	0
1	1	1

each satisfy five out of the six constraints. For the first matrix, all rows have even parity, but only the first two columns have odd parity. For the second matrix, the first two rows have even parity, and all columns have odd parity.

Consider the game where Alice receives $x \in \{1, 2, 3\}$ as input (specifying the number of a row), and Bob receives $y \in \{1, 2, 3\}$ as input (specifying the number of a column). Their goal is to each produce 3-bit outputs, $a_1a_2a_3$ for Alice and $b_1b_2b_3$ for Bob, such that

1. They satisfy the row/column parity constraints: $a_1 \oplus a_2 \oplus a_3 = 0$ and $b_1 \oplus b_2 \oplus b_3 = 1$.

2. They are consistent where the row intersects the column: $a_y = b_x$.

As usual, Alice and Bob are forbidden from communicating once the game starts, so Alice does not know y and Bob does not know x. We shall show the best classical strategy has success probability 8/9, while there is a quantum strategy that always succeeds.

An example of a deterministic strategy that attains success probability 8/9 (when the input xy is uniformly distributed) is where Alice plays according to the rows of the first matrix above and Bob plays according the columns of the second matrix above. This succeeds in all cases, except where $x = y = 3$. To see why this is optimal, note that for any other classical strategy, it is possible to represent it as two matrices as above but with different entries. Alice plays according to the rows of the first matrix and Bob plays according to the columns of the second matrix. We can assume that the rows of Alice's matrix all have even parity; if she outputs a row with odd parity then they immediately lose, regardless of Bob's output. Similarly, we can assume that all columns of Bob's matrix have odd parity.[5] Considering such a pair of matrices, the players lose at each entry where they differ. There must be such an entry, since otherwise it would be possible to have all rows even and all columns odd with one matrix. Thus, when the input xy is chosen uniformly from $\{1, 2, 3\} \times \{1, 2, 3\}$, the success probability of any classical strategy is at most 8/9.

We now give the *quantum* strategy for this game. Let I, X, Y, Z be the 2×2 Pauli matrices from Appendix A.7. Each is a one-qubit *observable* (see Section 1.2.2) with eigenvalues in $\{+1, -1\}$. That is, each can be written as $P_+ - P_-$ where P_+ and P_- are orthogonal projectors that sum to identity, and hence define a two-outcome measurement with outcomes $+1$ and -1. For example, $Z = |0\rangle\langle 0| - |1\rangle\langle 1|$, corresponding to a measurement in the computational basis (with $|b\rangle$ corresponding to outcome $(-1)^b$). And $X = |+\rangle\langle +| - |-\rangle\langle -|$, corresponding to a measurement in the Hadamard basis. The Pauli matrices are self-inverse, they anti-commute unless one of them is I (e.g., $XY = -YX$),

[5] In fact, the game can be simplified so that Alice and Bob each output just two bits, since the parity constraint determines the third bit.

and $X = iZY$, $Y = iXZ$, and $Z = iYX$. Consider the following table, where each entry is a tensor product of two Paulis:

$X \otimes X$	$Y \otimes Z$	$Z \otimes Y$
$Y \otimes Y$	$Z \otimes X$	$X \otimes Z$
$Z \otimes Z$	$X \otimes Y$	$Y \otimes X$

Because $(P_+ - P_-) \otimes (Q_+ - Q_-) = (P_+ \otimes Q_+ + P_- \otimes Q_-) - (P_+ \otimes Q_- + P_- \otimes Q_+)$, each such product is itself a $\{+1, -1\}$-valued observable. Hence each product of Pauli matrices corresponds to a measurement on a two-qubit space, with outcomes $+1$ and -1.

Note that the observables along each row commute and their product is $I \otimes I$, and the observables along each column commute and their product is $-I \otimes I$. This implies that for any two-qubit state, performing the three measurements along any row results in three $\{+1, -1\}$-valued bits whose product is $+1$. Also, performing the three measurements along any column results in three $\{+1, -1\}$-valued bits whose product is -1.

We can now describe the quantum protocol. It uses two pairs of entangled qubits, each of which is in initial state

$$\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$

(again, such states can be obtained from EPR-pairs by local operations). Alice, on input x, applies three two-qubit measurements corresponding to the observables in row x of the above table. For each measurement, if the result is $+1$ then she outputs 0, and if the result is -1 then she outputs 1. Similarly, Bob, on input y, applies the measurements corresponding to the observables in column y, and converts the ± 1-outcomes into bits.

We have already established that Alice and Bob's output bits satisfy the required parity constraints. It remains to show that Alice and Bob's output bits agree at the point where the row meets the column. For that measurement, Alice and Bob are measuring with respect to the same observable in the above table. Because all the observables in each row and in each column commute, we may assume that the place where they intersect is the first observable applied. Those bits are obtained by Alice and Bob each measuring $\frac{1}{2}(|01\rangle - |10\rangle)(|01\rangle - |10\rangle)$ with respect to the observable in entry (x, y) of the table. To show that their measurements will agree for all cases of xy, we consider the individual Pauli measurements on the individual entangled pairs of the form $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$. Let a' and b' denote the outcomes of the first measurement (in terms of bits), and a'' and b'' denote the outcomes of the second. Since the measurement associated with the tensor product of two observables is operationally equivalent to measuring each individual observable and taking the product of the results, we have $a_y = a' \oplus a''$ and $b_x = b' \oplus b''$. It is straightforward to verify that if the same measurement from $\{I, X, Y, Z\}$ is applied to each qubit of $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$ then the outcomes will be distinct: $a' \oplus b' = 1$ and $a'' \oplus b'' = 1$. We now have $a_y = b_x$, because

$$a_y \oplus b_x = (a' \oplus a'') \oplus (b' \oplus b'') = (a' \oplus b') \oplus (a'' \oplus b'') = 1 \oplus 1 = 0. \tag{15.3}$$

15.4 A non-local version of distributed Deutsch-Jozsa

The previous two examples used small amounts of entanglement: one EPR-pair for CHSH, two EPR-pairs for magic square. In both cases we could show that classical protocols need at least *some* communication if they want to achieve the same as what entanglement-based protocols can

achieve. We will now give a non-locality game that's parametrized by a number n, and where Alice and Bob's quantum strategy uses $\log n$ EPR-pairs [29]. The advantage is that we can show that classical protocols for this game need *much* classical communication rather than at least some nonzero amount.

> **Non-local DJ problem:** Alice and Bob receive n-bit inputs x and y that satisfy the DJ promise: either $x = y$, or x and y differ in exactly $n/2$ positions. The task is for Alice and Bob to provide outputs $a, b \in \{0,1\}^{\log n}$ such that if $x = y$ then $a = b$, and if x and y differ in exactly $n/2$ positions then $a \neq b$.

They achieve this as follows

1. Alice and Bob share $\log n$ EPR-pairs, i.e., the maximally entangled state $\frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} |i\rangle|i\rangle$.[6]

2. They both apply locally a conditional phase to obtain: $\frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} (-1)^{x_i} |i\rangle (-1)^{y_i} |i\rangle$.

3. They both apply a Hadamard transform, obtaining

$$\frac{1}{n\sqrt{n}} \sum_{i=0}^{n-1} (-1)^{x_i+y_i} \sum_{a \in \{0,1\}^{\log n}} (-1)^{i \cdot a} |a\rangle \sum_{b \in \{0,1\}^{\log n}} (-1)^{i \cdot b} |b\rangle$$
$$= \frac{1}{n\sqrt{n}} \sum_{a,b \in \{0,1\}^{\log n}} \left(\sum_{i=0}^{n-1} (-1)^{x_i+y_i+i \cdot (a \oplus b)} \right) |a\rangle|b\rangle.$$

4. They measure in the computational basis and output the results a and b, respectively.

For every a, the probability that both Alice and Bob obtain the same result a is:

$$\left| \frac{1}{n\sqrt{n}} \sum_{i=0}^{n-1} (-1)^{x_i+y_i} \right|^2,$$

which is $1/n$ if $x = y$, and 0 otherwise. This solves the problem perfectly using prior entanglement.

What about classical protocols? Suppose there is a classical protocol that uses C bits of communication. If they ran this protocol, and then Alice communicated her output a to Bob (using an additional $\log n$ bits), he could solve the distributed Deutsch-Jozsa problem since he could then check whether $a = b$ or $a \neq b$. But we know that solving the distributed Deutsch-Jozsa problem requires at least $0.007n$ bits of communication. Hence $C + \log n \geq 0.007n$, so $C \geq 0.007n - \log n$. Thus we have a non-locality problem that can be solved perfectly if Alice and Bob share $\log n$ EPR-pairs, while classically it needs not just *some* communication, but actually a *lot* of communication.

[6] Note that k EPR-pairs $\left(\frac{1}{\sqrt{2}}(|0\rangle_A|0\rangle_B + |1\rangle_A|1\rangle_B)\right)^{\otimes k}$ can also be written as $\frac{1}{\sqrt{2^k}} \sum_{i \in \{0,1\}^k} |i\rangle_A |i\rangle_B$ if we reorder the qubits, putting Alice's k qubits on the left and Bob's on the right. While these two ways of writing the state strictly speaking correspond to two different vectors of amplitudes, they still represent the same bipartite physical state, and we will typically view them as equal.

Exercises

1. Suppose Alice and Bob share an EPR-pair $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

   (a) Let U be a unitary with real entries. Show that the following two states are the same:
   (1) the state obtained if Alice applies U to her qubit of the EPR-pair;
   (2) the state obtained if Bob applies the transpose U^T to his qubit of the EPR-pair.

   (b) (H) What state do you get if each of Alice and Bob applies a Hadamard transform to their qubit of the EPR-pair?

2. Alice and Bob share an EPR-pair, $\frac{1}{\sqrt{2}}(|00\rangle+|11\rangle)$. Suppose they each measure their qubit with an X-observable (which corresponds to a particular projective measurement with possible outcomes $+1, -1$).

   (a) Show that Alice's measurement outcome is uniformly distributed, so 50% probability of outcome $+1$ and 50% probability of outcome -1.

   (b) (H) Show that Alice's and Bob's measurement outcomes are always equal.

   (c) Suppose we view $X \otimes X$ as one 2-qubit observable (with possible outcomes $+1, -1$) instead of two 1-qubit observables. What is the probability distribution on the two possible outcomes?

3. (H) Give a classical strategy using shared randomness for the CHSH game, such that Alice and Bob win the game with probability at least 3/4 for every possible input x, y (note the order of quantification: the same strategy has to work for every x, y).

4. "Mermin's game" is the following. Consider three space-like separated players: Alice, Bob, and Charlie. Alice receives input bit x, Bob receives input bit y, and Charlie receives input bit z. The input satisfies the promise that $x \oplus y \oplus z = 0$. The goal of the players is to output bits a, b, c, respectively, such that $a \oplus b \oplus c = \mathrm{OR}(x, y, z)$. In other words, the outputs should sum to 0 (mod 2) if $x = y = z = 0$, and should sum to 1 (mod 2) if $x + y + z = 2$.

   (a) Show that every classical deterministic strategy will fail on at least one of the 4 allowed inputs.

   (b) Show that every classical randomized strategy has success probability at most 3/4 under the uniform distribution on the four allowed inputs xyz.

   (c) Suppose the players share the following entangled 3-qubit state:

   $$\frac{1}{2}(|000\rangle - |011\rangle - |101\rangle - |110\rangle).$$

   Suppose each player does the following: if his/her input bit is 1, apply H to his/her qubit, otherwise do nothing. Describe the resulting 3-qubit superposition.

   (d) Using (c), give a quantum strategy that wins the above game with probability 1 on every possible input.

5. (H) This question examines how well the best quantum protocol can do for CHSH (resulting in the so-called "Tsirelson bound"). Consider a protocol where Alice and Bob share a $2k$-qubit state $|\psi\rangle = |\psi\rangle_{AB}$ with k qubits for Alice and k for Bob (the state can be arbitrary

and need not consist of EPR-pairs). Alice has two possible ± 1-valued observables[7] A_0 and A_1, and Bob has two possible ± 1-valued observables B_0 and B_1. Each of these observables acts on k qubits. On inputs $x \in \{0,1\}$ and $y \in \{0,1\}$, respectively, Alice measures her half of $|\psi\rangle$ with A_x and outputs the resulting sign $a \in \{+1,-1\}$, and Bob measures his half of $|\psi\rangle$ with B_y and outputs the resulting sign b. Note that we treat the output bits as signs instead of 0/1 now. However, the winning condition is the same: the AND of the input bits should equal the parity (XOR) of the output bits. So Alice and Bob *win* the game if $(-1)^{xy} = ab$.

(a) Show that the expected value of the product ab on inputs x, y is $\langle\psi|A_x \otimes B_y|\psi\rangle$ (this is the same as $\mathrm{Tr}\left[(A_x \otimes B_y)|\psi\rangle\langle\psi|\right]$).

(b) Define $2k$-qubit operator $C = A_0 \otimes B_0 + A_0 \otimes B_1 + A_1 \otimes B_0 - A_1 \otimes B_1$. Show that the winning probability of the protocol (averaged over all 4 inputs pairs x, y) is $\frac{1}{2} + \frac{1}{8}\langle\psi|C|\psi\rangle$.

(c) Show that $C^2 = 4I + (A_0A_1 - A_1A_0) \otimes (B_1B_0 - B_0B_1)$, where I is the $2k$-qubit identity matrix.

(d) Show that $\langle\psi|C|\psi\rangle \leq \sqrt{8}$.

(e) What can you conclude about the best-possible winning probability among all possible quantum protocols for CHSH?

[7] Remember that a ± 1-valued observable A can be written as $A = P - Q$, where P and Q are projectors on two orthogonal subspaces such that $P + Q = I$. This corresponds to a two-outcome measurement specified by projectors P and Q with outcomes $+1$ and -1, respectively.

Chapter 16

Quantum Cryptography

16.1 Quantum key distribution

One of the most basic tasks of cryptography is to allow Alice to send a message to Bob (whom she trusts) over a public channel, without allowing a third party Eve (for "eavesdropper") to get any information about M from tapping the channel. Suppose Alice wants to send message $M \in \{0,1\}^n$ to Bob. The goal here is not minimal communication, but *secrecy*. This is often done by public-key cryptography such as RSA. Such schemes, however, are only computationally secure, not information-theoretically secure: all the information about the private key can be computed from the public key, it just appears to take a lot of time to compute it—assuming of course that problems like factoring are classically hard, and that nobody builds a quantum computer...

In contrast, the following "one-time pad" scheme is information-theoretically secure. If Alice and Bob share a *secret key* $K \in \{0,1\}^n$ then Alice can send $C = M \oplus K$ over the channel. By adding K to what he received, Bob learns M. On the other hand, if Eve didn't know anything about K then she learns nothing about M from tapping the message $M \oplus K$ that goes over the channel. How can we make Alice and Bob share a secret key? In the classical world this is impossible, but with quantum communication it can be done!

Below we describe the famous BB84 quantum key distribution protocol of Bennett and Brassard [22]. Consider two possible bases: basis 0 is the computational basis $\{|0\rangle, |1\rangle\}$, and basis 1 is the Hadamard basis $\{|+\rangle, |-\rangle\}$. The main property of quantum mechanics that we'll use, is that if a bit b is encoded in an unknown basis, then Eve cannot get information about b without disturbing the state, and the latter can be detected by Alice and Bob.[1]

1. Alice chooses n random bits $a_1, \ldots, a_n$ and n random bases $b_1, \ldots, b_n$. She sends a_i to Bob in basis b_i over the public quantum channel. For example, if $a_i = 0$ and $b_i = 1$ then the ith qubit that she sends is in state $|+\rangle$.

2. Bob chooses random bases $b'_1, \ldots, b'_n$ and measures the qubits he received in those bases, yielding bits $a'_1, \ldots, a'_n$.

[1] Quantum key distribution might in fact better be called "quantum eavesdropper detection." There are some more assumptions underlying BB84 that should be made explicit: we assume that the classical channel used in steps 3–5 is "authenticated," meaning that Alice and Bob know they are talking to each other, and Eve can listen but not change the bits sent over the classical channel (in contrast to the qubits sent during step 1 of the protocol, which Eve is allowed to manipulate in any way she wants).

3. Bob sends Alice all b'_i, and Alice sends Bob all b_i. Note that for roughly $n/2$ of the i's, Alice and Bob used the same basis $b_i = b'_i$. For those i's Bob should have $a'_i = a_i$ (if there was no noise and Eve didn't tamper with the ith qubit on the channel). Both Alice and Bob know for which i's this holds. Let's call these roughly $n/2$ positions the "shared string."

4. Alice randomly selects $n/4$ locations in the shared string, and sends Bob those locations as well as the values a_i at those locations. Bob then checks whether they have the same bits in those positions. If the fraction of errors is bigger than some number p, then they suspect some eavesdropper was tampering with the channel, and they abort.[2]

5. If the test is passed, then they discard the $n/4$ test-bits, and have roughly $n/4$ bits left in their shared string. This is called the "raw key." Now they do some classical postprocessing on the raw key: "information reconciliation" to ensure they end up with exactly the same shared string, and "privacy amplification" to ensure that Eve has negligible information about that shared string.[3]

The communication is n qubits in step 1, $2n$ bits in step 3, $O(n)$ bits in step 4, and $O(n)$ bits in step 5. So the required amount of communication is linear in the length of the shared secret key that Alice and Bob end up with.

It's quite hard to formally *prove* that this protocol yields (with high probability) a shared key about which Eve has negligible information. In fact it took more than 12 years before BB84 was finally proven secure [103, 94]. The main reason it works is that when the qubits that encode $a_1, \ldots, a_n$ are going over the public channel, Eve doesn't know yet in which bases $b_1, \ldots, b_n$ these are encoded (she will learn the b_i later from tapping the classical communication in step 3, but at that point this information is not of much use to her anymore). She could try to get as much information as she can about $a_1, \ldots, a_n$ by some measurement, but there's an *information-vs-disturbance tradeoff*: the more information Eve learns about $a_1, \ldots, a_n$ by measuring the qubits, the more she will disturb the state, and the more likely it is that Alice and Bob will detect her presence in step 4.

We won't go into the full proof details here, just illustrate the information-disturbance tradeoff for the case where Eve individually attacks the qubits encoding each bit in step 1 of the protocol.[4] In Fig. 16.1 we give the four possible states for one BB84-qubit. If Alice wants to send $a_i = 0$, then she sends a uniform mixture of $|0\rangle$ and $|+\rangle$ across the channel; if Alice wants to send $a_i = 1$ she sends a uniform mixture of $|1\rangle$ and $|-\rangle$. Suppose Eve tries to learn a_i from the qubit on the channel. The best way for her to do this is to measure in the orthonormal basis corresponding to state $\cos(\pi/8)|0\rangle + \sin(\pi/8)|1\rangle$ and $-\sin(\pi/8)|0\rangle + \cos(\pi/8)|1\rangle$. Note that the first state is halfway between the two encodings of 0, and the second state is halfway between the two encodings of 1 (remember that $|-\rangle$ and $-|-\rangle$ are physically indistinguishable). This will give her the value of a_i with probability $\cos(\pi/8)^2 \approx 0.85$ (remember the 2-to-1 quantum random access code from Exercise 2 of Chapter 13). However, this measurement will change the state of the qubit by an angle of at least $\pi/8$, so if Bob now measures the qubit he receives in the same basis as Alice, his

[2] The number p can for instance be set to the natural error-rate that the quantum channel would have if there were no eavesdropper.

[3] This can be done for instance by something called the "leftover hash lemma."

[4] The more complicated situation where Eve does an n-qubit measurement on all qubits of step 1 simultaneously can be reduced to the case of individual-qubit measurements by something called the *quantum De Finetti theorem*, but we won't go into the details here.

probability of recovering the incorrect value of a_i is at least $\sin(\pi/8)^2 \approx 0.15$ (if Bob measured in a different basis than Alice, then the result will be discarded anyway). If this i is among the test-bits Alice and Bob use in step 4 of the protocol (which happens with probability 1/2), then they will detect an error. Eve can of course try a less disturbing measurement to reduce the probability of being detected, but such a measurement will also have a lower probability of telling her a_i.

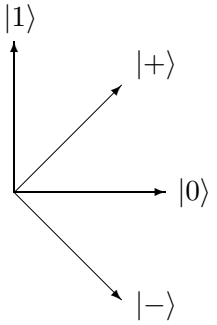

Figure 16.1: The four possible states in BB84 encoding: $|0\rangle$ and $|+\rangle$ are two different encodings of 0, and $|1\rangle$ and $|-\rangle$ are two different encodings of 1.

16.2 Reduced density matrices and the Schmidt decomposition

Suppose Alice and Bob share some pure state $|\phi\rangle$. If this state is entangled, it cannot be written as a tensor product $|\phi_A\rangle \otimes |\phi_B\rangle$ of separate pure states for Alice and Bob. Still, there is a way to describe Alice's local state as a mixed state, by *tracing out* Bob's part. Formally, if $A \otimes B$ is a product matrix then $\mathrm{Tr}_B(A \otimes B) = A \cdot \mathrm{Tr}(B)$. By extending this linearly to matrices that are not of product form, the operation Tr_B is well-defined on all mixed states (note that Tr_B removes Bob's part of the state, leaving just Alice's part of the state). If ρ_{AB} is some bipartite state (mixed or pure, entangled or not), then $\rho_A = \mathrm{Tr}_B(\rho_{AB})$ is Alice's local density matrix. This describes all the information she has. For example, for an EPR-pair $|\phi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, the corresponding density matrix is

$$\begin{aligned}\rho_{AB} &= \frac{1}{2}(|00\rangle\langle 00| + |00\rangle\langle 11| + |11\rangle\langle 00| + |11\rangle\langle 11|) \\ &= \frac{1}{2}(|0\rangle\langle 0| \otimes |0\rangle\langle 0| + |0\rangle\langle 1| \otimes |0\rangle\langle 1| + |1\rangle\langle 0| \otimes |1\rangle\langle 0| + |1\rangle\langle 1| \otimes |1\rangle\langle 1|),\end{aligned}$$

and since $\mathrm{Tr}(|a\rangle\langle b|) = 1$ if $a = b$ and $\mathrm{Tr}(|a\rangle\langle b|) = 0$ if $|a\rangle$ and $|b\rangle$ are orthogonal, we have

$$\rho_A = \mathrm{Tr}_B(\rho_{AB}) = \frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|).$$

In other words, Alice's local state is the same as a random coin flip! Similarly we can compute Bob's local state by tracing out Alice's part of the space: $\rho_B = \mathrm{Tr}_A(\rho_{AB})$.

The Schmidt decomposition is a very useful way to write bipartite pure states, and allows us to easily calculate the local density matrices of Alice and Bob. It says the following: for every bipartite state $|\phi\rangle$ there is an orthonormal basis $|a_1\rangle, \ldots, |a_d\rangle$ for Alice's space, an orthonormal basis $|b_1\rangle, \ldots, |b_d\rangle$ for Bob's, and nonnegative reals $\lambda_1, \ldots, \lambda_d$ whose squares sum to 1, such that

$$|\phi\rangle = \sum_{i=1}^{d} \lambda_i |a_i\rangle |b_i\rangle. \tag{16.1}$$

The number of nonzero λ_i's is called the *Schmidt rank* of the state. For example, an EPR-pair has Schmidt coefficients $\lambda_1 = \lambda_2 = 1/\sqrt{2}$ and hence has Schmidt rank 2.

The existence of the Schmidt decomposition is shown as follows. Let $\rho_A = \mathrm{Tr}_B(|\phi\rangle\langle\phi|)$ be Alice's local density matrix. This is Hermitian, so it has a spectral decomposition $\rho_A = \sum_i \mu_i |a_i\rangle\langle a_i|$ with orthonormal eigenvectors $|a_i\rangle$ and nonnegative real eigenvalues μ_i. Note $\sum_i \mu_i = \mathrm{Tr}(\rho_A) = 1$. Since the $\{|a_i\rangle\}$ form an orthonormal set, we can extend it to an orthonormal basis for Alice's d-dimensional space (adding additional orthonormal $|a_i\rangle$ and $\mu_i = 0$). Hence there are c_{ij} such that

$$|\phi\rangle = \sum_{i,j=1}^{d} \sqrt{\mu_i} c_{ij} |a_i\rangle|j\rangle,$$

where the $|j\rangle$ are the computational basis states for Bob's space. Define $\lambda_i = \sqrt{\mu_i}$ and $|b_i\rangle = \sum_j c_{ij}|j\rangle$. This gives the decomposition of $|\phi\rangle$ of Eq. (16.1). It only remains to show that $\{|b_i\rangle\}$ is an orthonormal set, which we do as follows. The density matrix version of Eq. (16.1) is

$$|\phi\rangle\langle\phi| = \sum_{i,j=1}^{d} \lambda_i \lambda_j |a_i\rangle\langle a_j| \otimes |b_i\rangle\langle b_j|.$$

We know that if we trace out the B-part from $|\phi\rangle\langle\phi|$, then we should get $\rho_A = \sum_i \lambda_i^2 |a_i\rangle\langle a_i|$, but that can only happen if $\langle b_j|b_i\rangle = \mathrm{Tr}(|b_i\rangle\langle b_j|) = 1$ for $i = j$ and $\langle b_j|b_i\rangle = 0$ for $i \neq j$. Hence the $|b_i\rangle$ form an orthonormal set. Note that from Eq. (16.1) it easily follows that Bob's local density matrix is $\rho_B = \sum_i \lambda_i^2 |b_i\rangle\langle b_i|$.

16.3 The impossibility of perfect bit commitment

Key distribution is just one of the many tasks cryptographers would like to solve. Another important primitive is *bit commitment*. In this scenario there is no eavesdropper, but Alice and Bob don't trust each other. Suppose Alice has a bit b which for the time being she doesn't want to reveal to Bob, though she would like to somehow convince Bob that she has already made up her mind about b and won't change its value later. A protocol for bit commitment comes in two stages, each of which may involve several rounds of communication:

1. In the "commit" phase Alice gives Bob a state which is supposed to commit her to the value of b (without informing Bob about the value of b).

2. In the "reveal" phase Alice sends b to Bob, and possibly some other information to allow him to check that this is indeed the same value b that Alice committed to before.

A protocol is *binding* if Alice can't change her mind, meaning she can't get Bob to "open" $1 - b$. A protocol is *concealing* if Bob cannot get any information about b before the "reveal phase."[5]

A good protocol for bit commitment would be a very useful building block for many other cryptographic applications. For instance, it would allow Alice and Bob (who still don't trust each other) to jointly flip a fair coin. Maybe they're going through a divorce, and need to decide who

[5] A good metaphor to think about this: in the commit phase Alice locks b inside a safe which she sends to Bob. This commits her to the value of b, since the safe is no longer in her hands. During the reveal phase she sends Bob the key to the safe, who can then open it and learn b.

gets to keep their joint car. Alice can't just flip the coin by herself because Bob doesn't trust her to do this honestly, and vice versa. Instead, Alice would pick a random coin b and commit to it. Bob would then pick a random coin c and send it to Alice. Alice then reveals b, and the outcome of the coin flip is defined to be $b \oplus c$. As long as at least one of the two parties follows this protocol, the result will be a fair coin flip.

Perfect coin flipping (and hence also perfect bit commitment) are known to be impossible in the classical world. After BB84 there was some hope that perfect bit commitment (and hence also perfect coin flipping) would be possible in the quantum world, and there were some seemingly-secure proposals for quantum protocols to achieve this. Unfortunately it turns out that there is *no* quantum protocol for bit commitment that is both perfectly binding and perfectly concealing.

To show that a protocol for perfect bit commitment is impossible, consider the joint pure state $|\phi_b\rangle$ that Alice and Bob would have if Alice wants to commit to bit-value b, and they both honestly followed the protocol.[6] If the protocol is perfectly concealing, then the reduced density matrix on Bob's side should be independent of b, i.e., $\mathrm{Tr}_A(|\phi_0\rangle\langle\phi_0|) = \mathrm{Tr}_A(|\phi_1\rangle\langle\phi_1|)$. The way we constructed the Schmidt decomposition in the previous section now implies that there exist Schmidt decompositions of $|\phi_0\rangle$ and $|\phi_1\rangle$ with the same λ_i's and the same b_i's: there exist orthonormal bases $\{a_i\}$ and $\{a'_i\}$ such that

$$|\phi_0\rangle = \sum_{i=1}^{d} \lambda_i |a_i\rangle|b_i\rangle \quad \text{and} \quad |\phi_1\rangle = \sum_{i=1}^{d} \lambda_i |a'_i\rangle|b_i\rangle$$

Now Alice can locally switch from $|\phi_0\rangle$ to $|\phi_1\rangle$ by just applying on her part of the state the map $|a_i\rangle \mapsto |a'_i\rangle$. Alice's map is unitary because it takes one orthonormal basis to another orthonormal basis. But then the protocol is not binding at all: Alice can still freely change her mind about the value of b after the "commit" phase is over! Accordingly, if a quantum protocol for bit commitment is perfectly concealing, it cannot be binding at all.

16.4 More quantum cryptography

Quantum cryptography is by now a pretty large subset of the area of quantum information and computation. Here we just briefly mention a few other topics in quantum crypto:

- There are quantum protocols for bit commitment that are partially concealing and partially binding—something which is still impossible in the classical world. A primitive called "weak coin flipping" can be implemented almost perfectly in the quantum world, and cannot be implemented at all in the classical world.

- Under assumptions on the fraction of dishonest players among a set of k parties, it is possible to implement *secure multi-party quantum computation*. This is a primitive that allows the players to compute any function of their k inputs, without revealing more information to player i than can be inferred from i's input plus the function value.

- One can actually do nearly perfect bit commitment, coin flipping, etc., assuming the dishonest party has *bounded quantum storage*, meaning that it can't keep large quantum states coherent

[6] The assumption that the state is pure rather than mixed is without loss of generality.

for longer times. At the present state of quantum technology this is a very reasonable assumption (though a breakthrough in physical realization of quantum computers would wipe out this approach).

- In *device-independent* cryptography, Alice and Bob want to solve certain cryptographic tasks like key distribution or randomness generation without trusting their own devices (for instance because they don't trust the vendor of their apparatuses). Roughly speaking, the idea here is to use Bell-inequality violations to prove the presence of entanglement, and then use this entanglement for cryptographic purposes. Even if Alice or Bob's apparatuses have been tampered with, they can still only violate things like the CHSH inequality if they actually share an entangled state.

- Experimentally it is much easier to realize quantum key distribution than general quantum computation, because you basically just need to prepare qubits (usually photons) in either the computational or the Hadamard basis, send them across a channel (usually an optical fibre, but sometimes free space), and measure them in either the computational or the Hadamard basis. Many sophisticated experiments have already been done. Somewhat surprisingly, you can already commercially buy quantum key distribution machinery. Unfortunately the implementations are typically not perfect (for instance, we don't have perfect photon counters), and once in a while another loophole is exposed in the implementation, which the vendor then tries to patch, etc.

Exercises

1. Here we will consider in more detail the information-disturbance tradeoff for measuring a qubit in one of the four BB84 states (each of which occurs with probability 25%).

   (a) Suppose Eve measures the qubit in the orthonormal basis given by $\cos(\theta)|0\rangle + \sin(\theta)|1\rangle$ and $\sin(\theta)|0\rangle - \cos(\theta)|1\rangle$, for some parameter $\theta \in [0, \pi/4]$. The first basis vector corresponds to output 0, the second to output 1. For each of the four possible BB84 states, give the probabilities of outcome 0 and outcome 1 (so your answer should consist of 8 numbers, each of which is a function of θ).

   (b) What is the average probability that Eve's measurement outcome equals the encoded bit a_i, as function of θ? (average taken both over the uniform distribution over the four BB84 states, and over the probabilities calculated in part (a))

   (c) What is the average absolute value of the angle by which the state is changed conditioned on Eve's outcome being the encoded bit a_i? Again, the answer should be a function of θ.

2. (a) What is the Schmidt rank of the state $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$?

   (b) Suppose Alice and Bob share k EPR-pairs. What is the Schmidt rank of their joint state?

   (c) Prove that a pure state $|\phi\rangle$ is entangled if, and only if, its Schmidt rank is greater than 1.

3. Give the Schmidt decomposition of the state $\frac{1}{2}(|0,0\rangle + |0,1\rangle + |1,1\rangle + |1,2\rangle)$.

4. Consider a density matrix ρ on Alice's Hilbert space. A bipartite pure state $|\psi\rangle_{AB}$ is called a *purification* of ρ, if $\rho = \mathrm{Tr}_B(|\psi\rangle\langle\psi|)$. The B-register in $|\psi\rangle_{AB}$ is called the *purifying register*.

   (a) Show that an EPR-pair is a purification of the 1-qubit mixed state $\rho = I/2$.

   (b) Show that if ρ is a density matrix of rank r, then there exists a purification of ρ where the purifying register has at most $\lceil \log_2 r \rceil$ qubits.

   (c) Show that if $|\psi\rangle_{AB}$ and $|\psi'\rangle_{AB}$ are purifications of the same ρ, then there exists a unitary U on Bob's space such that $|\psi'\rangle_{AB} = (I \otimes U)|\psi\rangle_{AB}$.

5. Suppose Alice has a 1-qubit state ρ.

   (a) Suppose Alice chooses a uniformly random Pauli matrix (see Appendix A.7) and applies it to ρ. What is the resulting density matrix?

   (b) Suppose Alice and Bob shared a secret 2-bit string ab. How can Alice send ρ to Bob over a public quantum channel, without leaking any information to Eve, in such a way that Bob can recover ρ?

6. (H) Prove that Alice cannot give information to Bob by doing a unitary operation on her part of an entangled pure state.

7. Suppose Alice sends *two* n-bit messages M_1 and M_2 with the one-time pad scheme, reusing the *same* n-bit key K. Show that Eve can now get some information about M_1, M_2 from tapping the classical channel.

Chapter 17

Error-Correction and Fault-Tolerance

17.1 Introduction

When Shor's algorithm had just appeared in 1994, most people (especially physicists) were extremely skeptical about the prospects of actually building a quantum computer. In their view, it would be impossible to avoid errors when manipulating small quantum systems, and such errors would very quickly overwhelm the computation, rendering it no more useful than classical computation. However, in the few years that followed, the theory of quantum error-correction and fault-tolerant computation was developed. This shows, roughly speaking, that if the error-rate per operation can be brought down to something reasonably small (say 1%), then under some reasonable assumptions we can actually do near-perfect quantum computing for as long as we want. Below we give a succinct and somewhat sketchy introduction to this important but complex area, just explaining the main ideas. See the surveys by Gottesman [66] and Terhal [125] for much more (in particular the "surface code").

17.2 Classical error-correction

In the early days of classical computing, errors were all over the place: memory-errors, errors in bits sent over a channel, incorrectly applied instructions, etc.[1] Nowadays hardware is much more reliable, but we also have much better "software solutions" for errors, in particular error-correcting codes. Such codes take a string of data and encode it in a larger string (the "codeword"), adding a lot of redundancy so that a small fraction of errors on the codeword won't be able to reduce the information about the encoded data.

The simplest example is of course the repetition code. If we want to protect a bit b, we could repeat it three times:

$$b \mapsto bbb.$$

If we want to decode the encoded bit b from the (possibly corrupted) 3-bit codeword, we just take the majority value of the 3 bits.

Consider a very simple noise model: every bit is flipped (independently of the other bits) with probability p. Then initially, before applying the code, b has probability p to be flipped. But if we apply the repetition code, the probability that the majority-value of the three bits is different

[1] The name "bugs" actually comes from insects getting stuck inside the computer and causing errors.

from b, is the probability of 2 or 3 bitflips, which is $3p^2(1-p)+p^3 < 3p^2$. Hence the error-rate has been reduced from p to less than $3p^2$. If the initial error-rate p_0 was $< 1/3$, then the new error-rate $p_1 < 3p_0^2$ is less than p_0 and we have made progress: the error-rate on the encoded bit is smaller than before. If we'd like it to be even smaller, we could concatenate the code with itself, i.e., repeat each of the three bits in the code three times, so the codelength becomes 9. This would give error-rate $p_2 = 3p_1^2(1-p_1)+p_1^3 < 3p_1^2 < 27p_0^4$, giving a further improvement. As we can see, as long as the initial error-rate p was at most $1/3$, we can reduce the error-rate to whatever we want: k levels of concatenation encode one "logical bit" into 3^k "physical bits," but the error-rate for each logical bit has been reduced to $\frac{1}{3}(3p_0)^{2^k}$. This is a very good thing: if the initial error is below the threshold of $1/3$, then k levels of concatenation increases the number of bits exponentially (in k), but reduces the error-rate *double-exponentially fast*!

Typically, already a small choice of k gets the error-rate down to negligible levels. For example, suppose we want to protect some polynomial (in some n) number of bits for some polynomial number of time-steps, and our physical error-rate is some fixed $p_0 < 1/3$. Choosing $k = 2\log\log n$ levels of concatenation already suffices for this, because then $p_k \leq \frac{1}{3}(3p_0)^{2^k} \sim 2^{-(\log n)^2} = n^{-\log n}$ goes to 0 faster than any polynomial. With this choice of k, each logical bit would be encoded in $3^k = (\log n)^{2\log_2(3)}$ physical bits, so we only increase the number of bits by a polylogarithmic factor.

17.3 Quantum errors

The need for error-correction is far greater for quantum computers than for classical computers, because "quantum hardware" is much more fragile than classical hardware. Unfortunately, error-correction is also substantially more difficult in the quantum world, for several reasons:

- The classical solution of just repeating a state is not available in general in the quantum world, because of the no-cloning theorem.
- The classical world has basically only bitflip-errors, while the quantum world is continuous and hence has infinitely many different possible errors.
- Measurements that test whether a state is correct can collapse the state, losing information.

Depending on the specific model of errors that one adopts, it is possible to deal with all of these issues. We will consider the following simple error model. Consider quantum circuits with S qubits, and T time-steps; in each time-step, several gates on disjoint sets of qubits may be applied in parallel. After each time-step, at each qubit, independently from the other qubits, some unitary error hits that qubit with probability p. Note that we assume the gates themselves to operate perfectly; this is just a convenient technical assumption, since a perfect gate followed by errors on its outgoing qubits is the same as an imperfect gate.

Let's investigate what kind of (unitary) errors we could get on one qubit. Consider the four Pauli matrices from Appendix A.7:

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

These have an interpretation as possible errors: I corresponds to no-error, X is a bitflip-error, Z is a phaseflip-error, and $Y = iXZ$ is a phaseflip-error followed by a bitflip-error (and a global phase

of i, which doesn't matter). These four matrices span the space of all possible 2×2 matrices, so every possible error-operation E on a qubit is some linear combination of the 4 Pauli matrices. More generally, every $2^k \times 2^k$ matrix can be written uniquely as a linear combinations of matrices that each are the tensor product of k Pauli matrices.

Consider for example the error which puts a small phase ϕ on $|1\rangle$:

$$E = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} = e^{i\phi/2}\cos(\phi/2)I - ie^{i\phi/2}\sin(\phi/2)Z.$$

Note that for small ϕ most of the weight in this linear combination sits on I, which corresponds to the fact that E is close to I. The sum of squared moduli of the two coefficients is 1 in this case. That's not a coincidence: whenever we write a unitary as a linear combination of Pauli matrices, the sum of squares of the coefficients will be 1 (see Exercise 1).

The fact that all one-qubit errors are linear combinations of I, X, Y, Z, together with the linearity of quantum mechanics, implies that if we can correct bitflip-errors, phaseflip-errors, and their product, then we can correct *all possible* unitary errors on a qubit.[2] So typically, quantum error-correcting codes are designed to correct bitflip and phaseflip-errors (their product is then typically also correctable), and all other possible errors are then also handled without further work.

Our noise model does not explicitly consider errors on multiple qubits that are not a product of errors on individual qubits. However, even such a joint error on, say, k qubits simultaneously can still be written as a linear combination of products of k Pauli matrices. So also here the main observation applies: if we can just correct bitflip and phaseflip-errors on individual qubits, then we can correct all possible errors!

17.4 Quantum error-correcting codes

Quantum error-correcting codes encode a number of "logical qubits" into a larger number of "physical qubits," in such a way that errors on some number of its qubits can be corrected. The first and simplest is Peter Shor's 9-qubit code [121], which encodes 1 logical qubit into 9 physical qubits, and can correct an error on any one of the 9 physical qubits. Here are the codewords for the two logical basis states:

$$|0\rangle \mapsto |\overline{0}\rangle = \frac{1}{\sqrt{8}}(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)$$

$$|1\rangle \mapsto |\overline{1}\rangle = \frac{1}{\sqrt{8}}(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)$$

These two quantum codewords $|\overline{0}\rangle$ and $|\overline{1}\rangle$ span a 2-dimensional space $\{\alpha|\overline{0}\rangle + \beta|\overline{1}\rangle\}$. This 2-dimensional subspace of the overall 2^9-dimensional space is called the "codespace."

Suppose an error happens on one of these 9 qubits. We would like to have a procedure that maps the resulting state back to the codespace. By linearity, it suffices if we can do this for the basis states $|\overline{0}\rangle$ and $|\overline{1}\rangle$. First consider bitflip and phaseflip-errors.

[2] We can even correct the *non-unitary* errors that arise from undesired interaction between qubits of our circuit with the environment, but we won't talk about such errors here.

Detecting a bitflip-error. If a bitflip-error occurs on one the first 3 qubits, we can detect its location by noting which of the 3 positions is the minority bit. We can do this for each of the three 3-qubit blocks. Hence there is a unitary that writes down in 4 auxiliary qubits (which are all initially $|0\rangle$) a number $e_b \in \{0, 1, \ldots, 9\}$. Here $e_b = 0$ means that no bitflip-error was detected, and $e_b \in \{1, \ldots, 9\}$ means that a bitflip-error was detected on qubit number e_b. Note that we don't specify what should happen if more than one bitflip-error occurred.

Detecting a phaseflip-error. To detect a phaseflip-error, we can consider the relative phase for each of the three blocks $|000\rangle \pm |111\rangle$, and if they are not all the same, unitarily write down in 2 more auxiliary qubits (again, initially $|0\rangle$) a number $e_p \in \{0, 1, 2, 3\}$. Here $e_p = 0$ means that no phaseflip-error was detected, and $e_p \in \{1, 2, 3\}$ means that a phaseflip-error was detected in the e_pth block.[3]

Together the above two procedures form one unitary U (i.e., one circuit) that acts on $9+4+2 = 15$ qubits, and that "writes down" both e_b and e_p in auxiliary qubits. For example, suppose we have the state $\big|\overline{0}\big\rangle$. If X_i denotes a bitflip-error on the ith qubit and Z_j denotes a phaseflip-error on the jth qubit (let j' denote the number of the block in which qubit j lies). Then after these errors our state is $X_i Z_j \big|\overline{0}\big\rangle$. After fresh auxiliary qubits $\big|0^4\big\rangle\big|0^2\big\rangle$ are added, U maps

$$X_i Z_j \big|\overline{0}\big\rangle\big|0^4\big\rangle\big|0^2\big\rangle \mapsto X_i Z_j \big|\overline{0}\big\rangle|i\rangle\big|j'\big\rangle.$$

Together, $e_b = i$ and $e_p = j'$ form the "error syndrome"; this tells us which error occurred where. The error-correction procedure can now measure this syndrome in the computational basis, and take corrective action depending on the classical outcomes e_b and e_p: apply an X to qubit e_b (or no X if $e_b = 0$), and apply a Z to one qubit in the e_p-th block (or no Z if $e_p = 0$). The case of a Y-error on the ith qubit corresponds to the case where $i = j$ (i.e., the ith qubit is hit by both a phaseflip and a bitflip); our procedure still works in this case. Hence we can perfectly correct one Pauli-error on any one of the 9 codeword qubits.

As we argued before, the ability to correct Pauli-errors suffices to correct all possible errors. Let's see in more detail how this works. Consider for instance some 9-qubit unitary error E. Assume it can be decomposed as a linear combination of 9-qubit products of Paulis, each having at most one bitflip-error and one phaseflip-error:

$$E = \sum_{i,j} \alpha_{ij} X_i Z_j.$$

Suppose this error occurs on $\big|\overline{0}\big\rangle$:

$$E\big|\overline{0}\big\rangle = \sum_{i,j} \alpha_{ij} X_i Z_j \big|\overline{0}\big\rangle.$$

If we now add auxiliary qubits $\big|0^4\big\rangle\big|0^2\big\rangle$ and apply the above unitary U, then we go into a superposition of error syndromes:

$$U(E \otimes I^{\otimes 6})\big|\overline{0}\big\rangle\big|0^4\big\rangle\big|0^2\big\rangle = \sum_{i,j} \alpha_{ij} X_i Z_j \big|\overline{0}\big\rangle|i\rangle\big|j'\big\rangle.$$

[3] Note that we are not discovering on which of the 9 qubits the phaseflip-error happened (in contrast to the case of bitflips), but that's OK: we can correct it nonetheless.

Measuring the auxiliary qubits will now probabilistically give us one of the syndromes $|i\rangle|j'\rangle$, and collapse the state to

$$X_i Z_j \big|\overline{0}\big\rangle |i\rangle \big|j'\big\rangle.$$

In a way, this measurement of the syndrome "discretizes" the continuously many possible errors to the finite set of Pauli-errors. Once the syndrome has been measured, we can apply a corrective X and/or Z to the first 9 qubits to undo the specific error corresponding to the specific syndrome we got as outcome of our measurement.

So now we can correct an error on one qubit. To achieve this, however, we have substantially increased the number of locations where such an error could occur: the number of qubits has gone from 1 to 9 (even to 15 if we count the auxiliary qubits as well), and we need a number of time-steps to compute and measure the syndrome, and to correct a detected error. Hence this procedure only gains us something if the error-rate p is so small that the probability of 2 or more errors on the larger encoded system is smaller than the probability of 1 error in the unencoded qubit. We will get back to this issue below, when talking about the threshold theorem. Note also that each new application of the correction-procedure need a new, fresh 6-qubit register initialized to $\big|0^4\big\rangle\big|0^2\big\rangle$. After one run of the error-correction procedure these auxiliary qubits will contain the measured error syndrome, and we can just discard this. In a way, error correction acts like a refrigerator: a fridge pumps heat out of its system and dumps it into the environment, and error-correction pumps noise out of its system and dumps it in the environment in the form of the discarded auxiliary qubits.

The above 9-qubit code is just one example of a quantum error-correcting code. Better codes exist, and a lot of work has gone into simultaneously optimizing the different parameters: we want to encode a large number of logical qubits into a not-much-larger number of physical qubits, while being able to correct as many errors as possible. The shortest code that encodes one logical qubit and protects against one error, has five physical qubits. There are also asymptotically good quantum error-correcting codes, which encode k logical qubits into $O(k)$ physical qubits and can correct errors on a constant fraction of the physical qubits (rather than just an error on one of the qubits).

17.5 Fault-tolerant quantum computation

Encoding a quantum state in a quantum error-correcting code to protect it against noise is good, but not enough: we also need to be able to do *operations* on the encoded qubits (Hadamards, CNOTs, etc.). One way is to decode the logical qubits, do the operation on them, and then re-encode them. This, however, is a recipe for disaster: if an error occurs between the decoding and subsequent encoding, we're unprotected. Accordingly, we need to be able to do operations on the logical qubits *while they are encoded*. Additionally, we need operations for regular stages of error-correction, i.e., measuring the syndrome and correcting. These operations may also introduce errors.[4]

There is a 7-qubit code (due to Andrew Steane) which is used often because it has nice properties: a Hadamard on the logical qubit corresponds to $H^{\otimes 7}$ on the physical qubits, and a CNOT between two logical qubits corresponds to applying CNOTs between the 7 pairs of the two blocks of physical qubits (i.e., between the 1st qubit of one block and the 1st qubit of the other block, etc.). Adding the gate that maps $|b\rangle \mapsto e^{ib\pi/4}|b\rangle$ to this suffices for universal quantum computation;

[4] It's like being inside a leaky boat on the open seas, using a leaky bucket to scoop out water all the time to prevent the boat from filling up with water and sinking. It's doable, but not easy.

unfortunately, implementing this gate fault-tolerantly takes a lot more work, and we won't go into that here (see Exercise 7, though).

When designing schemes for fault-tolerant computing, it is very important to ensure that errors do not spread too quickly. Consider for instance a CNOT: if its control-bit is erroneous, then after doing the CNOT also its target bit will be erroneous. The trick is to keep this under control in such a way that regular phases of error-correction don't get overwhelmed by the errors. In addition, we need to be able to fault-tolerantly prepare states, and measure logical qubits in the computational basis. We won't go into the (many) further details of fault-tolerant quantum computing (see Exercise 7 for one approach, and [66] for more).

17.6 Concatenated codes and the threshold theorem

The idea to concatenate a code with itself, described at the end of Section 17.2, also applies to quantum codes. Suppose we have some code that encodes one qubit into C qubits, that can correct one error on one of its C qubits, and uses D time-steps per stage of error-correcting (each time-step may involve a number of elementary gates in parallel). Instead of only 1, we now have CD locations where an error could occur! Assuming error-rate p per-qubit-per-timestep, the probability for the code to fail on a specific logical qubit at a specific time (i.e., to have *more than 1* physical error on its CD locations) is $p' = \sum_{i=2}^{CD} \binom{CD}{i} p^i$. If p is a sufficiently small constant, then this sum is dominated by the term for $i = 2$, and we have $p' \approx (CD)^2 p^2$. Hence if the initial error-rate p is below some magical constant $\approx 1/CD$, then each level of error-correction reduces the error-rate.

More generally, suppose we concatenate this code k times with itself. Then every "logical qubit" gets encoded into C^k qubits, but (by the same calculation as in Section 17.2) the error-rate for each logical qubit gets reduced to $O((CDp)^{2^k})$. Suppose we want to be able to "survive" $T = \text{poly}(n)$ time-steps without any error on the logical qubits; that is what we would need to run an efficient quantum algorithm on faulty quantum hardware. Then it suffices if we reduce the error rate to $\ll 1/T$, for which $k = O(\log \log T)$ levels of concatenation are enough. These layers of error-correction increase the number of qubits and the computation time by a factor which is exponential in k, but that is still only a polylogarithmic overhead, since $2^{O(\log \log T)} = (\log T)^{O(1)}$.

The above sketch (when implemented precisely) gives us the famous "threshold theorem" [3, 87]: if the initial error-rate of the quantum hardware can be brought down below some magical constant (known as the "fault-tolerance threshold"), then we can use software-solutions like quantum error-correcting codes and fault-tolerant computing to ensure that we can quantum compute for long periods of time without serious errors. Much research has gone into finding the best value for this fault-tolerance threshold. The more efficient our basic quantum error-correcting codes are (i.e., the smaller C and D), the higher (= better) the value of the threshold is. Currently the best rigorous estimates for the threshold are around 0.1%, but there is numerical evidence that even a few percent would be tolerable. This is actually one of the most important results in the area of quantum computing, and is the main answer to the skeptics mentioned at the start of the chapter: as long as experimentalists manage to implement basic operations within a few percent of error in a scalable way, then we should be able to build large-scale quantum computers.[5] Currently there seems to be no fundamental reason why we cannot do this; it is, however, an extremely hard engineering problem.

[5] This is of course assuming our simple model of independent noise on each physical qubit is not too far off; if the noise can be correlated in devious ways it becomes much harder (though often still possible) to protect against.

Exercises

1. (H) Let E be an arbitrary 1-qubit unitary. We know that it can be written as
$$E = \alpha_0 I + \alpha_1 X + \alpha_2 Y + \alpha_3 Z,$$
for some complex coefficients α_i. Show that $\sum_{i=0}^{3} |\alpha_i|^2 = 1$.

2. (a) Write the 1-qubit Hadamard transform H as a linear combination of the four Pauli matrices.
   (b) Suppose an H-error happens on the first qubit of $\alpha|\overline{0}\rangle + \beta|\overline{1}\rangle$ using the 9-qubit code. Give the various steps in the error-correction procedure that corrects this error.

3. Give a quantum circuit for the encoding of Shor's 9-qubit code, i.e., a circuit that maps $|00^8\rangle \mapsto |\overline{0}\rangle$ and $|10^8\rangle \mapsto |\overline{1}\rangle$. Explain why the circuit works.

4. Shor's 9-qubit code allows to *correct* a bit flip and/or a phase flip on one of its 9 qubits. Below we give a 4-qubit code which allows to *detect* a bitflip and/or a phaseflip. By this we mean that after the detection procedure we either have the original uncorrupted state back, or we know that an error occurred (though we do not know which one). The logical 0 and 1 are encoded as:
$$|\overline{0}\rangle = \tfrac{1}{2}(|00\rangle + |11\rangle) \otimes (|00\rangle + |11\rangle)$$
$$|\overline{1}\rangle = \tfrac{1}{2}(|00\rangle - |11\rangle) \otimes (|00\rangle - |11\rangle)$$
   (a) Give a procedure (either as a circuit or as sufficiently-detailed pseudo-code) that detects a bitflip error on one of the 4 qubits of $\alpha|\overline{0}\rangle + \beta|\overline{1}\rangle$.
   (b) Give a procedure (either as a circuit or as sufficiently-detailed pseudo-code) that detects a phaseflip error on one of the 4 qubits of $\alpha|\overline{0}\rangle + \beta|\overline{1}\rangle$.
   (c) Does that mean that we can now detect *any* unitary 1-qubit error on one of the 4 qubits? Explain your answer.

5. (H) Show that there cannot be a quantum code that encodes one logical qubit into $2k$ physical qubits while being able to correct errors on up to k of the physical qubits.

6. Suppose we have a qubit whose density matrix is $\rho = \alpha_I I + \alpha_X X + \alpha_Y Y + \alpha_Z Z$, where $\alpha_I, \alpha_X, \alpha_Y, \alpha_Z$ are real coefficients and I, X, Y, Z are the Pauli matrices.
   (a) Show that $\alpha_I = 1/2$.
   (b) Depolarizing noise (of strength $p \in [0,1]$) acts on a qubit as follows: with probability $1-p$ nothing happens to the qubit, and with probability p the qubit is replaced by the "completely mixed state" of a qubit, whose density matrix is $I/2$.
   Show that depolarizing noise on the above qubit doesn't change the coefficient α_I, but shrinks each of $\alpha_X, \alpha_Y, \alpha_Z$ by a factor of $1-p$.

7. Suppose we have a qubit $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ to which we would like to apply a $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$ gate, but for some reason we cannot. However, we have a second qubit available in state $\frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$, and we can apply a CNOT gate and an $S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$ gate.

(a) What state do we get if we apply a CNOT to the first and second qubit?

(b) Suppose we measure the second qubit. What are the probabilities of outcomes 0 and 1, respectively?

(c) Suppose the measurement yields 0. Show how we can get $T|\phi\rangle$ in the first qubit.

(d) Suppose the measurement yields 1. Show how we can get $T|\phi\rangle$ in the first qubit, up to an (irrelevant) global phase.

NB: This way of implementing the T-gate is very helpful in fault-tolerant computing, where often CNOT and S are easy to do on encoded states but T is not. What this exercise shows is that we can prepare (encodings of) the so-called "magic state" $\frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$ beforehand (offline, assuming we can store them until we need them in our computation), and use those to indirectly implement a T-gate using only CNOT and S-gates.

Appendix A

Some Useful Linear Algebra

In this appendix we sketch some useful parts of linear algebra, most of which will be used somewhere or other in these notes.

A.1 Some terminology and notation

We use $V = \mathbb{C}^d$ to denote the d-dimensional complex vector space, which is the set of all column vectors of d complex numbers. We assume familiarity with the basic rules of matrix addition and multiplication. A set of vectors $v_1, \ldots, v_m \in V$ is *linearly independent* if the only way to get $\sum_{i=1}^m a_i v_i$ equal to the zero-vector 0, is to set $a_1 = \cdots = a_m = 0$. A *basis* for V is a set of vectors $v_1, \ldots, v_d$ such that every vector $v \in V$ can be written as a linear combination of those basis vectors $v = \sum_{i=1}^d a_i v_i$. One can show that a basis is linearly independent.

We use A_{ij} for the (i,j)-entry of a matrix A and A^T for its *transpose*, which has $A^T_{ij} = A_{ji}$. We use I_d to denote the $d \times d$ identity matrix, which has 1s on its diagonal and 0s elsewhere; we usually omit the subscript d when the dimension is clear from context. If A is square and there is a matrix B such that $AB = BA = I$, then we use A^{-1} to denote this B, which is called the *inverse* of A (and is unique if it exists). Note that $(AB)^{-1} = B^{-1}A^{-1}$. If A is a matrix (not necessarily square), then A^* denotes its *conjugate transpose*: the matrix obtained by transposing A and taking the complex conjugates of all entries. Note that $(AB)^* = B^*A^*$. Physicists often write $A^\dagger$ instead of A^*.

For vectors v, w, we use $\langle v|w\rangle = v^*w = \sum_i v_i^* w_i$ for their *inner product*. The combination of the vector space V with this inner product is called a *Hilbert space*. Two vectors v, w are *orthogonal* if $\langle v|w\rangle = 0$. The inner product induces a vector *norm* $\|v\| = \sqrt{\langle v|v\rangle} = \sqrt{\sum_i |v_i|^2}$. The norm in turn induces a *distance* $\|v - w\|$ between vectors v and w. Note that distance and inner product are closely related:

$$\|v - w\|^2 = \langle v - w|v - w\rangle = \|v\|^2 + \|w\|^2 - \langle v|w\rangle - \langle w|v\rangle.$$

In particular, for unit vectors v and w the real part of their inner product equals $1 - \frac{1}{2}\|v - w\|^2$. Hence unit vectors that are close together have an inner product close to 1, and vice versa. The *Cauchy-Schwarz inequality* gives $|\langle v|w\rangle| \leq \|v\| \cdot \|w\|$ (see Appendix B).

A set $\{v_i\}$ of vectors is called an *orthogonal* set if all vectors are pairwise orthogonal: $\langle v_i|v_j\rangle = 0$ if $i \neq j$. If additionally the vectors all have norm 1, then the set is called *orthonormal*. The *outer product* of v and w is the matrix vw^*. Below we will restrict attention to square matrices, unless

explicitly mentioned otherwise. The complex number λ is an *eigenvalue* of square matrix A if there is some nonzero vector v (called an *eigenvector*) such that $Av = \lambda v$.

A.2 Unitary matrices

A matrix A is *unitary* if $A^{-1} = A^*$. The following conditions are equivalent:

1. A is unitary
2. A preserves inner product: $\langle Av|Aw\rangle = \langle v|w\rangle$ for all v, w
3. A preserves norm: $\|Av\| = \|v\|$ for all v
4. $\|Av\| = 1$ if $\|v\| = 1$

(1) implies (2) because if A is unitary then $A^*A = I$, and hence $\langle Av|Aw\rangle = (v^*A^*)Aw = \langle v|w\rangle$. (2) implies (1) as follows: if A is not unitary then $A^*A \neq I$, so then there is a w such that $A^*Aw \neq w$ and, hence, a v such that $\langle v|w\rangle \neq \langle v|A^*Aw\rangle = \langle Av|Aw\rangle$, contradicting (2). Clearly (2) implies (3). Moreover, it is easy to show that (3) implies (2) using the following identity:

$$\|v + w\|^2 = \|v\|^2 + \|w\|^2 + \langle v|w\rangle + \langle w|v\rangle.$$

The equivalence of (3) and (4) is obvious. Note that by (4), the eigenvalues of a unitary matrix have absolute value 1.

A.3 Diagonalization and singular values

Matrices A and B are *similar* if there is an invertible matrix S such that $A = SBS^{-1}$. Note that if $Av = \lambda v$, then $BS^{-1}v = \lambda S^{-1}v$, so similar matrices have the same eigenvalues. *Schur's lemma* states that every matrix A is similar to an upper triangular matrix: $A = U^{-1}TU$ for some unitary U and upper triangular T. Since similar matrices have the same eigenvalues and the eigenvalues of an upper triangular matrix are exactly its diagonal entries, the eigenvalues of A form the diagonal of T.

A matrix D is *diagonal* if $D_{ij} = 0$ whenever $i \neq j$. Let S be some matrix satisfying $AS = SD$ for some diagonal matrix D. Let v_i be the ith column of S and λ_i be the ith entry on the diagonal of D, then

$$\underbrace{\begin{pmatrix} \vdots & & \vdots \\ Av_1 & \cdots & Av_d \\ \vdots & & \vdots \end{pmatrix}}_{AS} = \underbrace{\begin{pmatrix} \vdots & & \vdots \\ \lambda_1 v_1 & \cdots & \lambda_d v_d \\ \vdots & & \vdots \end{pmatrix}}_{SD},$$

and we see that v_i is an eigenvector of A associated with eigenvalue λ_i. Conversely, if $v_1, \ldots, v_d$ are eigenvectors of A with eigenvalues $\lambda_1, \ldots, \lambda_d$, then we have $AS = SD$, where S has the v_i as columns and D is the diagonal matrix of λ_i. We call a square matrix A *diagonalizable* if it is similar to some diagonal matrix D: $A = SDS^{-1}$. This D then has A's eigenvalues λ_i on its diagonal, some of which may be zero. Note that A is diagonalizable iff it has a linearly independent set of d eigenvectors. These eigenvectors will form the columns of S, giving $AS = SD$, and linear independence ensures

that S has an inverse, giving $A = SDS^{-1}$. A matrix A is *unitarily* diagonalizable iff it can be diagonalized via a unitary matrix U: $A = UDU^{-1}$. If the columns of U are the vectors u_i, and the diagonal entries of D are λ_i, then we can also write $A = \sum_i \lambda_i u_i u_i^*$; this is sometimes called the *spectral decomposition* of A. By the same argument as before, A will be unitarily diagonalizable iff it has an orthonormal set of d eigenvectors.

A matrix A is *normal* if it commutes with its conjugate transpose ($A^*A = AA^*$). For example, unitary matrices are normal. If A is normal and $A = U^{-1}TU$ for some upper triangular T (which must exist because of Schur's lemma), then $T = UAU^{-1}$ and $T^* = UA^*U^{-1}$, so $TT^* = UAA^*U^{-1} = UA^*AU^{-1} = T^*T$. Hence T is normal and upper triangular, which implies (with a little work) that T is diagonal. This shows that normal matrices are unitarily diagonalizable. Conversely, if A is diagonalizable as $U^{-1}DU$, then $AA^* = U^{-1}DD^*U = U^{-1}D^*DU = A^*A$, so then A is normal. Thus a matrix is normal iff it is unitarily diagonalizable. If A is not normal, it may still be diagonalizable via a non-unitary S, for example:

$$\underbrace{\begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}}_{S} \cdot \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}}_{D} \cdot \underbrace{\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}}_{S^{-1}}.$$

If $A = UDU^{-1}$ then $A^* = UD^*U^{-1}$, so the eigenvalues of A^* are the complex conjugates of the eigenvalues of A.

An important class of normal (and hence unitarily diagonalizable) matrices are the *Hermitian* matrices, which are the ones satisfying $A = A^*$. Note that the previous paragraph implies that the eigenvalues of Hermitian matrices are real. A Hermitian matrix is called *positive definite* (resp. *positive semidefinite*) if all its eigenvalues are positive (resp. non-negative). If all eigenvalues are 0 or 1, then A is called a *projection* (or *projection matrix* or *projector*). This is equivalent to requiring $A^2 = A$.

Not all matrices are diagonalizable, for instance $A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$. However, every matrix A has a *singular value decomposition*, as follows. It is easy to see that the matrix A^*A has the same eigenvectors as A and that its eigenvalues are the squared absolute values of the eigenvalues of A. Since A^*A is Hermitian and hence normal, we have $A^*A = UDU^{-1}$ for some U and some non-negative real diagonal matrix D. The entries of $\Sigma = \sqrt{D}$ are called the *singular values* of A. Every A has a singular value decomposition $A = U\Sigma V^{-1}$, with U, V unitary. This implies that A can be written as $A = \sum_i \lambda_i u_i v_i^*$, with u_i the columns of U and v_i the columns of V.

A.4 Trace

The *trace* of a matrix A is the sum of its diagonal entries: $\mathrm{Tr}(A) = \sum_i A_{ii}$. Some important and easily verified properties of $\mathrm{Tr}(A)$ are:

- $\mathrm{Tr}(A + B) = \mathrm{Tr}(A) + \mathrm{Tr}(B)$
- $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$ (the "cyclic property" of the trace)
- $\mathrm{Tr}(A)$ is the sum of the eigenvalues of A
  (This follows from Schur and the previous item: $\mathrm{Tr}(A) = \mathrm{Tr}(UTU^{-1}) = \mathrm{Tr}(U^{-1}UT) = \mathrm{Tr}(T) = \sum_i \lambda_i$)

A.5 Tensor products

If $A = (A_{ij})$ is an $m \times n$ matrix and B an $m' \times n'$ matrix, then their *tensor* product (a.k.a. *Kronecker* product) is the $mm' \times nn'$ matrix

$$A \otimes B = \begin{pmatrix} A_{11}B & \cdots & A_{1n}B \\ A_{21}B & \cdots & A_{2n}B \\ & \ddots & \\ A_{m1}B & \cdots & A_{mn}B \end{pmatrix}.$$

For example:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \end{pmatrix}.$$

The following properties of the tensor product are easily verified:

- $c(A \otimes B) = (cA) \otimes B = A \otimes (cB)$ for all scalars c
- $(A \otimes B)^* = A^* \otimes B^*$ (and similarly for inverse and transpose)
- $A \otimes (B + C) = (A \otimes B) + (A \otimes C)$
- $A \otimes (B \otimes C) = (A \otimes B) \otimes C$
- $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$

Different vector spaces can also be combined using tensor products. If V and V' are vectors spaces of dimension d and d' with basis $\{v_1, \ldots, v_d\}$ and $\{v'_1, \ldots, v'_{d'}\}$, respectively, then their tensor product space is the $d \cdot d'$-dimensional space $W = V \otimes V'$ spanned by $\{v_i \otimes v'_j \mid 1 \leq i \leq d, 1 \leq j \leq d'\}$. Applying a linear operation A to V and B to V' corresponds to applying the tensor product $A \otimes B$ to the tensor product space W.

A.6 Rank

The *rank* of a matrix A (over a field $\mathbb{F}$) is the size of the largest linearly independent set of rows of A (linear independence taken over $\mathbb{F}$). Unless mentioned otherwise, we take $\mathbb{F}$ to be the field of real numbers. We say that A has *full rank* if its rank equals its dimension. The following properties are all easy to show:

- $\text{rank}(A) = \text{rank}(A^*)$
- $\text{rank}(A)$ equals the number of nonzero eigenvalues of A (counting multiplicity)
- $\text{rank}(A + B) \leq \text{rank}(A) + \text{rank}(B)$
- $\text{rank}(AB) \leq \min\{\text{rank}(A), \text{rank}(B)\}$
- $\text{rank}(A \otimes B) = \text{rank}(A) \cdot \text{rank}(B)$
- A has an inverse iff A has full rank

A.7 The Pauli matrices

The four *Pauli matrices* are:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},\ X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},\ \text{and } Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{A.1}$$

Note that each Pauli matrix P is both unitary and Hermitian, and hence self-inverse: $P^{-1} = P$. This implies that their eigenvalues are in $\{-1, 1\}$. Define the *Hilbert-Schmidt* inner product on the space of $d \times d$ matrices as $\langle A, B\rangle = \frac{1}{d}\mathrm{Tr}(A^*B)$. With respect to this inner product (for $d = 2$), the four Pauli matrices form an orthonormal set. This implies that every complex 2×2 matrix A can be written as a linear combination of the Pauli matrices:

$$A = \alpha_I I + \alpha_X X + \alpha_Y Y + \alpha_Z Z,$$

with complex coefficients $\alpha_P = \langle A, P\rangle$. If A is Hermitian, then these coefficients will be real.

We can also consider the *n-qubit* Paulis, which are n-fold tensor products of the above 2×2 Paulis. For example $X \otimes Z \otimes I \otimes Y \otimes Z$ is a 5-qubit Pauli. There are 4^n n-qubit Paulis, since we have 4 possibilities for each of the n tensor factors, and these 4^n matrices form an orthonormal set w.r.t. Hilbert-Schmidt inner product. Accordingly, every $2^n \times 2^n$ matrix A can be written uniquely as a linear combination of the 4^n n-qubit Paulis. Again, if A is Hermitian, then the 4^n coefficients will be real.

A.8 Dirac notation

Physicists often write their linear algebra in *Dirac notation*, and we will follow that custom for denoting quantum states. In this notation we write $|v\rangle = v$ and $\langle v| = v^*$. The first is called a *ket*, the second a *bra*. Note that

- $\langle v|w\rangle = \langle v||w\rangle$
- If A is unitarily diagonalizable, then $A = \sum_i \lambda_i |v_i\rangle\langle v_i|$ for some orthonormal set of eigenvectors $\{v_i\}$
- $|v\rangle\langle v| \otimes |w\rangle\langle w| = (|v\rangle \otimes |w\rangle)(\langle v| \otimes \langle w|)$

Appendix B

Some other Useful Math and CS

Here we collect various basic but useful facts and definitions needed in parts of the lecture notes.

B.1 Some equalities and inequalities

- A *complex number* is of the form $c = a + bi$, where $a, b \in \mathbb{R}$, and i is the imaginary unit, which satisfies $i^2 = -1$. Such a c can also be written as $c = re^{i\phi}$ where $r = |c| = \sqrt{a^2 + b^2}$ is the *magnitude* of c, and $\phi \in [0, 2\pi)$ is the angle that c makes with the positive horizontal axis when we view it as a point (a, b) in the plane. Note that complex numbers of magnitude 1 lie on the unit circle in this plane. We can also write those as $e^{i\phi} = \cos(\phi) + i\sin(\phi)$. The complex conjugate c^* is $a - ib$, equivalently $c^* = re^{-i\phi}$.

- The Cauchy-Schwarz inequality: for $a = (a_1, \ldots, a_n) \in \mathbb{R}^n$ and $b = (b_1, \ldots, b_n) \in \mathbb{R}^n$

$$\sum_{i=1}^{n} a_i b_i \leq \sqrt{\sum_{i=1}^{n} a_i^2} \sqrt{\sum_{i=1}^{n} b_i^2}.$$

  Equivalently, written in terms of inner products and norms of vectors: $|\langle a, b\rangle| \leq \|a\| \cdot \|b\|$. Proof: for every real λ we have $0 \leq \langle a - \lambda b, a - \lambda b\rangle = \|a\|^2 + \lambda^2\|b\|^2 - 2\lambda\langle a, b\rangle$. Now set $\lambda = \|a\|/\|b\|$ and rearrange (a slightly more complicated proof works if $a, b \in \mathbb{C}^n$).

- $\sum_{j=0}^{m-1} z^j = \begin{cases} m & \text{if } z = 1 \\ \frac{1-z^m}{1-z} & \text{if } z \neq 1 \end{cases}$
  Proof: The case $z = 1$ is obvious; for the case $z \neq 1$, observe $(1-z)(\sum_{j=0}^{m-1} z^j) = \sum_{j=0}^{m-1} z^j - \sum_{j=1}^{m} z^j = 1 - z^m$. For example, if $z = e^{2\pi ir/N}$ is a root of unity, with r an integer in $\{1, \ldots, N-1\}$, then $\sum_{j=0}^{N-1} z^j = \frac{1-e^{2\pi ir}}{1-e^{2\pi ir/N}}$.

- The above ratio can be rewritten using the identity $|1 - e^{i\theta}| = 2|\sin(\theta/2)|$; this identity can be seen by drawing the numbers 1 and $e^{i\theta}$ as vectors from the origin in the complex plane, and dividing their angle θ in two. Some other useful trigonometric identities: $\cos(\theta)^2 + \sin(\theta)^2 = 1$, $\sin(2\theta) = 2\sin(\theta)\cos(\theta)$.

- $1 + x \leq e^x$ for all real numbers x (positive as well as negative).

- If $\varepsilon_j \in [0,1]$ then $1 - \sum_{j=1}^{k} \varepsilon_j \leq \prod_{j=1}^{k} (1-\varepsilon_j) \leq e^{-\sum_{j=1}^{k} \varepsilon_j}$.

  Proof: The upper bound comes from the preceding item. The lower bound follows easily by induction, using the fact that $(1-\varepsilon_1)(1-\varepsilon_2) = 1 - \varepsilon_1 - \varepsilon_2 + \varepsilon_1\varepsilon_2 \geq 1 - \varepsilon_1 - \varepsilon_2$.

B.2 Algorithms and probabilities

- When we do not care about constant factors, we'll often use big-Oh notation: $T(n) = O(f(n))$ means there exist constants $c, d \geq 0$ such that for all integers n, we have $T(n) \leq cf(n) + d$. Similarly, big-Omega notation is used for lower bounds: $T(n) = \Omega(f(n))$ means there exist constants $c, d \geq 0$ such that $T(n) \geq cf(n) - d$ for all n. $T(n) = \Theta(f(n))$ means that simultaneously $T(n) = O(f(n))$ and $T(n) = \Omega(f(n))$. Such notation is often used to write upper and/or lower bounds on the running time of algorithms as a function of their input length n.

- For $N = 2^n$, we can identify the integers $\{0, \ldots, N-1\}$ with their n-bit binary representations as follows: the bitstring $x = x_{n-1} \ldots x_1 x_0 \in \{0,1\}^n$ corresponds to the integer $\sum_{i=0}^{n-1} x_i 2^i$. The leftmost bit x_{n-1} is called the *most significant bit* of x, and the rightmost bit x_0 is its *least significant* bit. For example, if $n = 3$ then the bitstring $x = x_2 x_1 x_0 = 101$ corresponds to the integer $x_2 \cdot 4 + x_1 \cdot 2 + x_0 \cdot 1 = 4 + 1 = 5$. The integer 0 corresponds to the bitstring 0^n (the value of n should be clear from context).

- Three basic upper bounds on the tails of probability distributions:

  Markov: if X is a nonnegative random variable with expectation μ, then $\Pr[X \geq k\mu] \leq 1/k$.
  Proof: Since X is nonnegative, $\mu \geq \Pr[X \geq k\mu] \cdot k\mu$.

  Chebyshev: if X is a random variable with expectation μ and standard deviation σ, then $\Pr[|X - \mu| \geq k\sigma] \leq 1/k^2$.
  Proof: Apply Markov to the random variable $|X - \mu|^2$, whose expectation is σ^2.

  Chernoff/Hoeffding: if $X = \sum_{i=1}^{n} X_i$ is the sum of n independent, identically distributed random variables $X_i \in \{0,1\}$, each with expectation $\Pr[X_i = 1] = p$, then X has expectation $\mu = np$, and exponentially decreasing tail bound $\Pr[|X - \mu| \geq \alpha n] \leq 2e^{-2\alpha^2 n}$.
  Proof idea: For all parameters λ, we have $\Pr[X - \mu \geq t] = \Pr[e^{\lambda X} \geq e^{\lambda(t+\mu)}]$. Upper bound the latter probability by applying Markov to the random variable $e^{\lambda X}$. This is a product of n independent random variables $e^{\lambda X_i}$, so its expectation is easy to analyze. Then choose λ to minimize the upper bound.

- A *randomized* algorithm is a classical algorithm that can flip random coins during its operation, meaning its behavior is partially determined by chance and its output is not a deterministic function of its input. One can think of a randomized algorithm as a probability distribution over deterministic algorithms (one deterministic algorithm for each setting of the coins).

- When we say a (randomized) algorithm has error probability $\leq 1/3$, this typically means *in the worst case*: for every possible input, the algorithm produces the correct answer with probability $\geq 2/3$, where the probability is taken over the random coin flips during its operation. Such statements do not refer to "most" inputs under some input distribution unless stated explicitly.

- If a (classical or quantum) algorithm produces the correct answer in *expected* running time T, then we can convert that into an algorithm with *worst-case* running time $3T$ and error probability $\leq 1/3$, as follows. Run the original algorithm for $3T$ steps, and just cut it off if it hasn't terminated by itself. The probability of non-termination within $3T$ steps is at most $1/3$ by Markov's inequality. Hence with probability $\geq 2/3$ we will have the correct answer.

- If a (classical or quantum) algorithm with 0/1-outputs has error probability $\leq 1/3$, then we can cheaply reduce this error probability to small $\varepsilon > 0$, as follows. Choose odd $n = O(\log(1/\varepsilon))$ such that $2e^{-2\alpha^2 n} \leq \varepsilon$ for $\alpha = 1/6$. Run the original algorithm n times and output the majority among the n output bits. The probability that this majority is wrong (i.e., that the number of correct output bits is more than αn below its expectation), is at most ε by the Chernoff bound. Hence we output the correct answer with probability $\geq 1 - \varepsilon$.

Appendix C

Hints for Exercises

Chapter 1

1. Consider what U has to do when $|\phi\rangle = |0\rangle$, when $|\phi\rangle = |1\rangle$, and when $|\phi\rangle$ is a superposition of these two.

Chapter 2

6. Instead of measuring the qubit, apply a CNOT that "copies" it to a new $|0\rangle$-qubit, which is then left alone until the end of the computation. Analyze what happens.
11. Use Bernstein-Vazirani.

Chapter 3

5.c. Approximate the state of part (a) using the subroutine of part (b), and see what happens if you apply Hadamards to the approximate state. Use the fact that $\frac{1}{2^N}\sum_{w=0}^{N/2+2\sqrt{N}}\binom{n}{w}$ is nearly 1.

Chapter 4

3. Use $|\alpha_i^2 - \beta_i^2| = |\alpha_i - \beta_i| \cdot |\alpha_i + \beta_i|$ and the Cauchy-Schwarz inequality.
4.d. Use triangle inequality.
4.e. Drop all phase-gates with small angles $\phi < 1/n^3$ from the $O(n^2)$-gate circuit for F_{2^n} explained in Section 4.5. Calculate how many gates are left in the circuit, and analyze the distance between the unitaries corresponding to the new circuit and the original circuit.

Chapter 5

1.a. The Schönhage-Strassen algorithm computes the product of two n-bit integers mod N, in $O(n\log(n)\log\log(n))$ steps (you can just invoke this algorithm as a black-box in your solution, you don't need to explain ow it works).
3.a. The prime number theorem implies that roughly $N/\ln N$ of the numbers between 1 and N are prime; also there is an efficient classical algorithm to test if a given number is prime. Be explicit in how many bits your primes p and q will have.
3.b. Use Exercise 1.

3.c. The set of all possible messages forms a group of size $\phi(N)$. Euler's Theorem says that in any group G, we have $a^{|G|} = 1$ for all $a \in G$ (here '1' is the identity element in the group).

Chapter 6

4.b. Show that $\|M\|^2 = \left\|M^2\right\| \leq \frac{2}{3}\|M\|+$ a small constant. 5. Use the SWAP test from Section 14.6.

Chapter 7

3. Start with $m = x_i$ for a random i, and repeatedly use Grover's algorithm to find an index j such that $x_j < m$ and update $m = x_j$. Continue this until you can find no element smaller than m, and analyze the number of queries of this algorithm. You are allowed to argue about this algorithm on a high level (i.e., things like "use Grover to search for a j such that..." are OK), no need to write out complete circuits.
4.b. What is the probability in (a) if $s = \sqrt{N}$?
4.c. Choose a set S of size $s = N^{1/3}$, and classically query all its elements. First check if S contains a collision. If yes, then you're done. If not, then use Grover to find a $j \notin S$ that collides with an $i \in S$.
5.b. Recall that if there are i solutions, Grover's algorithm finds a solution using an expected number of $O(\sqrt{N/i})$ queries.
7.e. Choose γ in (d) such that applying $\lceil \tilde{k} \rceil$ rounds of amplitude amplification to $\mathcal{A}$ results in a solution for y with probability 1.
8.a. Try running the exact version of Grover (see end of Section 7.2) with different guesses for what the actual t is.

Chapter 8

4.a. Choose a uniformly random vector $v \in \{0,1\}^n$, calculate ABv and Cv, and check whether these two vectors are the same.
4.b. Consider the case $A = I$.
4.c. Modify the algorithm for collision-finding: use a random walk on the Johnson graph $J(n, r)$, where each vertex corresponds to a set $R \subseteq [n]$, and that vertex is marked if there are $i, j \in R$ such that $(AB)_{i,j} \neq C_{i,j}$.
5.b. View the $3n$-step random walk algorithm as a deterministic algorithm with an additional input $r \in \{0,1\}^n \times \{1,2,3\}^{3n}$, where the first n bits determine x, and the last $3n$ entries determine which variable of the leftmost false clauses will be flipped in the $3n$ steps of the random walk. Use Grover search on the space of all such r (no need to write out complete circuits here).

Chapter 9

4. Calculate the subnormalized second-register state $(\langle 0| \otimes I)(W^{-1} \otimes I)V(W \otimes I)|0\rangle|\psi\rangle$.
6.d. Like in the analysis of Grover's algorithm and regular amplitude amplification (Chapter 7), the product of two reflections on $\mathcal{S}$ is a rotation of $\mathcal{S}$.
7. Use triangle inequality, $\|H\| \leq 1$, and the fact that $k! \geq (k/e)^k$.

Chapter 10

No hints, sorry!

Chapter 11

4.a. Use Exercise 2.
4.b. Show that the symmetrized approximate polynomial r induced by the algorithm has degree at least N.
6.c. Use the result of Exercise 5 for $N = 2$.
7.b. When defining the relation R, consider that the hardest task for this algorithm is to distinguish inputs of weight $N/2$ from inputs of weight $N/2 + 1$.
9. Show how you can use sorting to solve the Majority-problem and then use the lower bound from Exercise 7 to get an $\Omega(N)$ lower bound on sorting. (It is actually known that sorting takes $\Omega(N \log N)$ comparisons even on a quantum computer, but you don't have to show that.)
10.a. Reduce the $bs(f)$-bit OR function (restricted to inputs of weight 0 or 1) to f and invoke the lower bound that we know for OR.

Chapter 12

1. Use binary search, running the algorithm with different choices of k to "zoom in" on the largest prime factor.
3.a. Write $|\psi_x\rangle = \alpha|0\rangle|\phi_0\rangle + \beta|1\rangle|\phi_1\rangle$, and consider the inner product between $(Z \otimes I)|\psi_x\rangle$ and $|\psi_x\rangle$.
3.b. Use part (a). Analyze the amplitude of $\left|x, 0^{S-n}\right\rangle$ in the final state $|\psi_x\rangle$, using ideas from the proof of **BQP** $\subseteq$ **PSPACE** in Section 12.3. Note that in contrast to that proof, you cannot use more than polynomial time for this exercise.

Chapter 13

2.a. It suffices to use pure states with real amplitudes as encoding. Try to "spread out" the 4 encodings $|\phi_{00}\rangle$, $|\phi_{01}\rangle$, $|\phi_{10}\rangle$, $|\phi_{11}\rangle$ in the 2-dimensional real plane as well as possible.
3. Use the fact that 1 classical bit of communication can only send 1 bit of information, no matter how much entanglement Alice and Bob share. Combine this fact with superdense coding.

Chapter 14

1. Argue that if Alice sends the same message for distinct inputs x and x', then Bob doesn't know what to output if his input is $y = x$.
2.a. Argue that if P is a projector then we can't have both $P|\phi\rangle = |\phi\rangle$ and $P|\psi\rangle = 0$.
2.c. Observe that among Alice's possible n-bit inputs are the n codewords of the Hadamard code that encodes $\log n$ bits (see Section 13.3); each pair of distinct Hadamard codewords is at Hamming distance exactly $n/2$. Use part (a) to argue that Alice needs to send pairwise orthogonal states for those n inputs, and hence her message-space must have dimension at least n.
3. Use the fact that 2 non-orthogonal states cannot be distinguished perfectly (Exercise 2), and that a set of 2^n vectors that are pairwise orthogonal must have dimension 2^n.
4. Invoke the quantum random access lower bound, Theorem 3 of Section 13.2.
5.b. Let Alice send a random row of $C(x)$ (with the row-index) and let Bob send a random column

of $C(y)$ (with the column-index).
6.a. Two distinct polynomials, each of degree $\leq d$, are equal on at most d points of the domain $\mathbb{F}_p$.
8.b. Run the protocol of part (a) on an initial state where Bob has a well-chosen superposition over many $|y\rangle$.
9.b. You can derive this from one of the communication lower bounds mentioned in this chapter, you don't need to prove this from scratch.
10. The matching M induces a projective measurement that Bob can do on the message he receives.
11.d. Alice could send a uniform superposition over all $h \in H$.

Chapter 15

1.b. You could write this out, but you can also get the answer almost immediately from part (a) and the fact that $H^T = H^{-1}$.

2.b. It's helpful here to write the EPR-pair in the basis $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.
3. For every fixed input x, y, there is a classical strategy that gives a wrong output only on that input, and that gives a correct output on all other possible inputs. Use the shared randomness to randomly choose one of those deterministic strategies.
5.b. Argue that $\frac{1}{4}\langle\psi|C|\psi\rangle = \Pr[\text{win}] - \Pr[\text{lose}]$.
5.c. Use that A_x^2 and B_y^2 are the k-qubit identity matrix.
5.d. You may use that $(\langle\psi|C|\psi\rangle)^2 \leq \langle\psi|C^2|\psi\rangle$ and $\langle\psi|D \otimes E|\psi\rangle \leq \langle\psi|D \otimes I|\psi\rangle \cdot \langle\psi|I \otimes E|\psi\rangle$ for Hermitian matrices D and E (these inequalities follow from Cauchy-Schwarz, but you don't have to show that).
5.e. $\cos(\pi/8)^2 = \frac{1}{2} + \frac{1}{\sqrt{8}}$.

Chapter 16

6. Show that a unitary on Alice's side of the state won't change Bob's local density matrix ρ_B.

Chapter 17

1. Compute the trace $\text{Tr}(E^*E)$ in two ways, and use the fact that $\text{Tr}(AB) = 0$ if A and B are distinct Paulis, and $\text{Tr}(AB) = \text{Tr}(I) = 2$ if A and B are the same Pauli.
5. Given an unknown qubit $\alpha|0\rangle + \beta|1\rangle$ encoded using this code, you could split the $2k$ qubits into two sets of k qubits each, and use each to recover a copy of the unknown qubit.

Bibliography

[1] S. Aaronson and A. Ambainis. Quantum search of spatial regions. *Theory of Computing*, 1(1):47–79, 2005. Earlier version in FOCS'03. quant-ph/0303041.

[2] S. Aaronson and Y. Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM*, 51(4):595–605, 2004.

[3] D. Aharonov and M. Ben-Or. Fault tolerant quantum computation with constant error. *SIAM Journal on Computing*, 38(4):1207–1282, 2008. Earlier version in STOC'97. quant-ph/9611025.

[4] A. Ambainis. Communication complexity in a 3-computer model. *Algorithmica*, 16(3):298–301, 1996.

[5] A. Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64(4):750–767, 2002. Earlier version in STOC'00. quant-ph/0002066.

[6] A. Ambainis. Polynomial degree vs. quantum query complexity. *Journal of Computer and System Sciences*, 72(2):220–238, 2006. Earlier version in FOCS'03. quant-ph/0305028.

[7] A. Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007. Earlier version in FOCS'04. quant-ph/0311001.

[8] A. Ambainis. Quantum search with variable times. In *Proceedings of 25th Annual Symposium on Theoretical Aspects of Computer Science (STACS'08)*, pages 49–61, 2008. arXiv:1010.4458.

[9] A. Ambainis, M. Mosca, A. Tapp, and R. de Wolf. Private quantum channels. In *Proceedings of 41st IEEE FOCS*, pages 547–553, 2000. quant-ph/0003101.

[10] P. K. Aravind. A simple demonstration of Bell's theorem involving two observers and no probabilities or inequalities. quant-ph/0206070, 2002.

[11] A. Aspect, Ph. Grangier, and G. Roger. Experimental tests of realistic local theories via Bell's theorem. *Physical Review Letters*, 47:460, 1981.

[12] L. Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of 48th ACM STOC*, pages 684–697, 2016. arXiv:1512.03547.

[13] L. Babai and E. M. Luks. Canonical labeling of graphs. In *Proceedings of 15th ACM STOC*, pages 171–183, 1983.

[14] Z. Bar-Yossef, T. S. Jayram, and I. Kerenidis. Exponential separation of quantum and classical one-way communication complexity. *SIAM Journal on Computing*, 38(1):366–384, 2008. Earlier version in STOC'04.

[15] R. Beals. Quantum computation of Fourier transforms over symmetric groups. In *Proceedings of 29th ACM STOC*, pages 48–53, 1997.

[16] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. Earlier version in FOCS'98. quant-ph/9802049.

[17] J. S. Bell. On the Einstein-Podolsky-Rosen paradox. *Physics*, 1:195–200, 1964.

[18] A. Belovs. Span programs for functions with constant-sized 1-certificates. In *Proceedings of 43rd ACM STOC*, pages 77–84, 2012. arXiv:1105.4024.

[19] P. A. Benioff. Quantum mechanical Hamiltonian models of Turing machines. *Journal of Statistical Physics*, 29(3):515–546, 1982.

[20] C. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. Wootters. Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Physical Review Letters*, 70:1895–1899, 1993.

[21] C. Bennett and S. Wiesner. Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states. *Physical Review Letters*, 69:2881–2884, 1992.

[22] C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of the IEEE International Conference on Computers, Systems and Signal Processing*, pages 175–179, 1984.

[23] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997. Earlier version in STOC'93.

[24] D. Berry, A. Childs, R. Cleve, R. Kothari, and R. Somma. Exponential improvement in precision for simulating sparse Hamiltonians. In *Proceedings of 46th ACM STOC*, pages 283–292, 2014. arXiv:1312.1414.

[25] D. Berry, A. Childs, R. Cleve, R. Kothari, and R. Somma. Simulating Hamiltonian dynamics with a truncated Taylor series. *Physical Review Letters*, 114:090502, 2015. arXiv:1412.4687.

[26] D. Berry, A. Childs, and R. Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *Proceedings of 56th IEEE FOCS*, pages 792–809, 2015. arXiv:1501.01715.

[27] J-F. Biasse and F. Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In *Proceedings of 27th ACM-SIAM SODA*, pages 893–902, 2016.

[28] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4–5):493–505, 1998. Earlier version in Physcomp'96. quant-ph/9605034.

[29] G. Brassard, R. Cleve, and A. Tapp. The cost of exactly simulating quantum entanglement with classical communication. *Physical Review Letters*, 83(9):1874–1877, 1999. quant-ph/9901035.

[30] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *AMS Contemporary Mathematics Series*, pages 53–74. 2002. quant-ph/0005055.

[31] G. Brassard, P. Høyer, and A. Tapp. Quantum algorithm for the collision problem. *ACM SIGACT News (Cryptology Column)*, 28:14–19, 1997. quant-ph/9705002.

[32] A. E. Brouwer and W. H. Haemers. *Spectra of Graphs*. Springer, 2012.

[33] H. Buhrman, R. Cleve, S. Massar, and R. de Wolf. Non-locality and communication complexity. *Reviews of Modern Physics*, 82:665–698, 2010. arXiv:0907.3584.

[34] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf. Quantum fingerprinting. *Physical Review Letters*, 87(16), September 26, 2001. quant-ph/0102001.

[35] H. Buhrman, R. Cleve, and A. Wigderson. Quantum vs. classical communication and computation. In *Proceedings of 30th ACM STOC*, pages 63–68, 1998. quant-ph/9802040.

[36] H. Buhrman and R. Špalek. Quantum verification of matrix products. In *Proceedings of 17th ACM-SIAM SODA*, pages 880–889, 2006. quant-ph/0409035.

[37] M. Bun and J. Thaler. Dual lower bounds for approximate degree and Markov-Bernstein inequalities. In *Proceedings of 40th ICALP*, volume 7965 of *Lecture Notes in Computer Science*, pages 303–314, 2013.

[38] Y. Cao, J. Romero, J. Olson, M. Degroote, P. Johnson, M. Kieferová, I. Kivlichan, T. Menke, B. Peropadre, N. Sawaya, S. Sim, L. Veis, and A. Aspuru-Guzik. Quantum chemistry in the age of quantum computing, 24 Dec 2018. arXiv:1812.09976.

[39] S. Chakraborty, A. Gilyén, and S. Jeffery. The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation, 5 Apr 2018. arXiv:1804.01973.

[40] A. Childs. Lecture notes on quantum algorithms. Technical report, University of Maryland, 2017. Available at https://cs.umd.edu/ amchilds/qa/.

[41] A. Childs, R. Kothari, and R. Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017. arXiv:1511.02306.

[42] B. S. Cirel'son. Quantum generalizations of Bell's inequality. *Letters in Mathematical Physics*, 4(2):93–100, 1980.

[43] J. F. Clauser, M. A. Horne, A. Shimony, and R. A. Holt. Proposed experiment to test local hidden-variable theories. *Physical Review Letters*, 23(15):880–884, 1969.

[44] R. Cleve. The query complexity of order-finding. In *Proceedings of 15th IEEE Conference on Computational Complexity*, pages 54–59, 2000. quant-ph/9911124.

[45] R. Cleve and H. Buhrman. Substituting quantum entanglement for communication. *Physical Review A*, 56(2):1201–1204, 1997. quant-ph/9704026.

[46] R. Cleve, W. van Dam, M. Nielsen, and A. Tapp. Quantum entanglement and the communication complexity of the inner product function. In *Proceedings of 1st NASA QCQC conference*, volume 1509 of *Lecture Notes in Computer Science*, pages 61–74. Springer, 1998. quant-ph/9708019.

[47] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. In *Proceedings of the Royal Society of London*, volume A454, pages 339–354, 1998. quant-ph/9708016.

[48] W. van Dam. Quantum oracle interrogation: Getting all information for almost half the price. In *Proceedings of 39th IEEE FOCS*, pages 362–367, 1998. quant-ph/9805006.

[49] D. Deutsch. Quantum theory, the Church-Turing principle, and the universal quantum Turing machine. In *Proceedings of the Royal Society of London*, volume A400, pages 97–117, 1985.

[50] D. Deutsch. Quantum computational networks. In *Proceedings of the Royal Society of London*, volume A425, 1989.

[51] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. In *Proceedings of the Royal Society of London*, volume A439, pages 553–558, 1992.

[52] A. Drucker and R. de Wolf. Quantum proofs for classical theorems. *Theory of Computing*, 2011. ToC Library, Graduate Surveys 2. arXiv:0910.3376.

[53] C. Dürr, M. Heiligman, P. Høyer, and M. Mhalla. Quantum query complexity of some graph problems. *SIAM Journal on Computing*, 35(6):1310–1328, 2006. Earlier version in ICALP'04. quant-ph/0401091.

[54] C. Dürr and P. Høyer. A quantum algorithm for finding the minimum. quant-ph/9607014, 18 Jul 1996.

[55] H. Ehlich and K. Zeller. Schwankung von Polynomen zwischen Gitterpunkten. *Mathematische Zeitschrift*, 86:41–44, 1964.

[56] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical Review*, 47:777–780, 1935.

[57] P. van Emde Boas. Machine models and simulations. In van Leeuwen [127], pages 1–66.

[58] M. Ettinger, P. Høyer, and M. Knill. The quantum query complexity of the hidden subgroup problem is polynomial. *Information Processing Letters*, 91(1):43–48, 2004. quant-ph/0401083.

[59] R. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6/7):467–488, 1982.

[60] R. Feynman. Quantum mechanical computers. *Optics News*, 11:11–20, 1985.

[61] L. Fortnow and J. Rogers. Complexity limitations on quantum computation. *Journal of Computer and System Sciences*, 59(2):240–252, 1999. Earlier version in Complexity'98. Also cs.CC/9811023.

[62] P. Frankl and V. Rödl. Forbidden intersections. *Transactions of the American Mathematical Society*, 300(1):259–286, 1987.

[63] R. Freivalds. Probabilistic machines can use less running time. In *IFIP Congress*, pages 839–842, 1977.

[64] M. Fürer. Faster integer multiplication. *SIAM Journal on Computing*, 39(3):979–1005, 2009. Earlier version in STOC'07.

[65] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics, 5 Jun 2018. arXiv:1806.01838.

[66] D. Gottesman. An introduction to quantum error correction and fault-tolerant quantum computation. arXiv:0904.2557, 16 Apr 2009.

[67] M. Grigni, L. Schulman, M. Vazirani, and U. Vazirani. Quantum mechanical algorithms for the nonabelian hidden subgroup problem. *Combinatorica*, 24(1):137–154, 2004. Earlier version in STOC'01.

[68] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of 28th ACM STOC*, pages 212–219, 1996. quant-ph/9605043.

[69] L. Hales and S. Hallgren. An improved quantum Fourier transform algorithm and applications. In *Proceedings of 41st IEEE FOCS*, pages 515–525, 2000.

[70] S. Hallgren. Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem. *Journal of the ACM*, 54(1):653–658, 2007. Earlier version in STOC'02.

[71] S. Hallgren, C. Moore, M. Roetteler, A. Russell, and P. Sen. Limitations of quantum coset states for graph isomorphism. *Journal of the ACM*, 57(6):34, 2010. Earlier version in STOC'06.

[72] S. Hallgren, A. Russell, and A. Ta-Shma. The hidden subgroup problem and quantum computation using group representations. *SIAM Journal on Computing*, 32(4):916–934, 2003. Earlier version in STOC'00.

[73] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, New York, fifth edition, 1979.

[74] A. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for solving linear systems of equations. *Physical Review Letters*, 103(15):150502, 2009. arXiv:0811.3171.

[75] B. Hensen, H. Bernien, A. E. Dréau, A. Reiserer, N. Kalb, M. S. Blok, J. Ruitenberg, R. F. L. Vermeulen, R. N. Schouten, C. Abellán, W. Amaya, V. Pruneri, M. W. Mitchell, M. Markham, D. J. Twitchen, D. Elkouss, S. Wehner, T. H. Taminiau, and R. Hanson. Loophole-free Bell inequality violation using electron spins separated by 1.3 kilometres. *Nature*, 526, 29 October 2015.

[76] A. S. Holevo. Bounds for the quantity of information transmitted by a quantum communication channel. *Problemy Peredachi Informatsii*, 9(3):3–11, 1973. English translation in *Problems of Information Transmission*, 9:177–183, 1973.

[77] P. Høyer, T. Lee, and R. Špalek. Negative weights make adversaries stronger. In *Proceedings of 39th ACM STOC*, pages 526–535, 2007. quant-ph/0611054.

[78] P. Høyer and R. Špalek. Lower bounds on quantum query complexity. *Bulletin of the EATCS*, 87:78–103, October 2005.

[79] G. Ivanyos, L. Sanselme, and M. Santha. An efficient quantum algorithm for the hidden subgroup problem in nil-2 groups. *Algorithmica*, 62(1–2):480–498, 2012. arXiv:0707.1260.

[80] S. Jeffery, R. Kothari, and F. Magniez. Nested quantum walks with quantum data structures. In *Proceedings of 24th ACM-SIAM SODA*, pages 1474–1485, 2013. arXiv:1210.1199.

[81] J. Katz and L. Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of 32nd ACM STOC*, pages 80–86, 2000.

[82] J. Kempe, A. Yu. Kitaev, and O. Regev. The complexity of the local Hamiltonian problem. *SIAM Journal on Computing*, 35(5):1070–1097, 2006. Earlier version in FSTTCS'04. quant-ph/0406180.

[83] I. Kerenidis and R. de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. *Journal of Computer and System Sciences*, 69(3):395–420, 2004. Earlier version in STOC'03. quant-ph/0208062.

[84] A. Yu. Kitaev. Quantum measurements and the Abelian stabilizer problem. quant-ph/9511026, 12 Nov 1995.

[85] A. Yu. Kitaev. Quantum NP, January 1999. Talk given at AQIP'99 conference, DePaul University, Chicago.

[86] B. Klartag and O. Regev. Quantum one-way communication is exponentially stronger than classical communication. In *Proceedings of 43rd ACM STOC*, 2011. arXiv:1009.3640.

[87] M. Knill, R. Laflamme, and W. Zurek. Threshold accuracy for quantum computation. quant-ph/9610011, 15 Oct 1996.

[88] D. E. Knuth. *The Art of Computer Programming. Volume 2: Seminumerical Algorithms*. Addison-Wesley, third edition, 1997.

[89] F. Le Gall. Improved quantum algorithm for triangle finding via combinatorial arguments. In *Proceedings of 55th IEEE FOCS*, pages 216–225, 2014. arXiv:1407.0085.

[90] T. Lee, F. Magniez, and M. Santha. Improved quantum query algorithms for triangle finding and associativity testing. *Algorithmica*, 77(2):459–486, 2017. arXiv:1210.1014.

[91] A. K. Lenstra and H. W. Lenstra, Jr. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer, 1993.

[92] H. W. Lenstra, Jr. and C. Pomerance. A rigorous time bound for factoring integers. *Journal of the American Mathematical Society*, 5:483–516, 1992.

[93] S. Lloyd. Universal quantum simulators. *Science*, 273:1073–1078, 1996.

[94] H-K. Lo and H. F. Chau. Unconditional security of quantum key distribution over arbitrarily long distances. quant-ph/9803006, 3 Mar 1998.

[95] G. H. Low and I. L. Chuang. Hamiltonian simulation by uniform spectral amplification. arXiv:1707.05391, 17 Jul 2017.

[96] G. H. Low and I. L. Chuang. Hamiltonian simulation by qubitization. arXiv:1610.06546, 20 Oct 2016.

[97] G. H. Low and I. L. Chuang. Optimal Hamiltonian simulation by quantum signal processing. *Physical Review Letters*, 118(1):010501, 2017. arXiv:1606.02685.

[98] G. H. Low, T. J. Yoder, and I. L. Chuang. Methodology of resonant equiangular composite quantum gates. *Physical Review X*, 6(4):041067, 2016. arXiv:1603.03996.

[99] F. Magniez, A. Nayak, J. Roland, and M. Santha. Search via quantum walk. *SIAM Journal on Computing*, 40(1):142–164, 2011. Earlier version in STOC'07. quant-ph/0608026.

[100] F. Magniez, M. Santha, and M. Szegedy. Quantum algorithms for the triangle problem. In *Proceedings of 16th ACM-SIAM SODA*, pages 1109–1117, 2005. quant-ph/0310134.

[101] Y. Manin. Vychislimoe i nevychislimoe (computable and noncomputable). *Soviet Radio*, pages 13–15, 1980. In Russian.

[102] Y. Manin. Classical computing, quantum computing, and Shor's factoring algorithm. quant-ph/9903008, 2 Mar 1999.

[103] D. Mayers. Unconditional security in quantum cryptography. quant-ph/9802025, 10 Feb 1998.

[104] C. Moore, D. N. Rockmore, and A. Russell. Generic quantum Fourier transforms. *ACM Transactions on Algorithms*, 2(4):707–723, 2006. quant-ph/0304064.

[105] C. Moore, A. Russell, and L. Schulman. The symmetric group defies strong Fourier sampling. *SIAM Journal on Computing*, 37(6):1842–1864, 2008. quant-ph/0501056+66. Earlier version in FOCS'05.

[106] M. Mosca and A. Ekert. The hidden subgroup problem and eigenvalue estimation on a quantum computer. In *Proceedings of 1st NASA QCQC conference*, volume 1509 of *Lecture Notes in Computer Science*, pages 174–188. Springer, 1998. quant-ph/9903071.

[107] A. Nayak. Optimal lower bounds for quantum automata and random access codes. In *Proceedings of 40th IEEE FOCS*, pages 369–376, 1999. quant-ph/9904093.

[108] I. Newman. Private vs. common random bits in communication complexity. *Information Processing Letters*, 39(2):67–71, 1991.

[109] I. Newman and M. Szegedy. Public vs. private coin flips in one round communication games. In *Proceedings of 28th ACM STOC*, pages 561–570, 1996.

[110] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[111] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[112] A. Razborov. Quantum communication complexity of symmetric predicates. *Izvestiya of the Russian Academy of Sciences, mathematics*, 67(1):159–176, 2003. quant-ph/0204025.

[113] B. Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every Boolean function. In *Proceedings of 50th IEEE FOCS*, pages 544–551, 2009.

[114] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[115] R. L. Rivest. Cryptography. In van Leeuwen [127], pages 717–755.

[116] T. J. Rivlin and E. W. Cheney. A comparison of uniform approximations on an interval and a finite subset thereof. *SIAM Journal on Numerical Analysis*, 3(2):311–320, 1966.

[117] M. Santha. Quantum walk based search algorithms. In *Proceedings of 5th TAMC*, pages 31–46, 2008. arXiv/0808.0059.

[118] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7:281–292, 1971.

[119] U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proceedings of 40th IEEE FOCS*, pages 410–414, 1999.

[120] A. Sherstov. Approximating the AND-OR tree. *Theory of Computing*, 9(20):653–663, 2013.

[121] P. W. Shor. Scheme for reducing decoherence in quantum memory. *Physical Review A*, 52:2493, 1995.

[122] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. Earlier version in FOCS'94. quant-ph/9508027.

[123] D. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997. Earlier version in FOCS'94.

[124] M. Szegedy. Quantum speed-up of Markov chain based algorithms. In *Proceedings of 45th IEEE FOCS*, pages 32–41, 2004. quant-ph/0401053.

[125] B. M. Terhal. Quantum error correction for quantum memories. *Reviews of Modern Physics*, 87:307, 2015. arXiv:1302.3428.

[126] L. Trevisan. Some applications of coding theory in computational complexity. *Quaderni di Matematica*, 13:347–424, 2004.

[127] J. van Leeuwen, editor. *Handbook of Theoretical Computer Science. Volume A: Algorithms and Complexity*. MIT Press, Cambridge, MA, 1990.

[128] L. Vandersypen, M. Steffen, G. Breyta, C. Yannoni, R. Cleve, and I. Chuang. Experimental realization of an order-finding algorithm with an NMR quantum computer. *Physical Review Letters*, 85(25):5452–5455, 2000. quant-ph/0007017.

[129] J. Watrous. Quantum algorithms for solvable groups. In *Proceedings of 33rd ACM STOC*, pages 60–67, 2001.

[130] J. Watrous. Quantum computational complexity. In *Encyclopedia of Complexity and System Science*. Springer, 2009. arXiv:0804.3401.

[131] R. de Wolf. *Quantum Computing and Communication Complexity*. PhD thesis, University of Amsterdam, 2001.

[132] W. K. Wootters and W. H. Zurek. A single quantum cannot be copied. *Nature*, 299:802–803, 1982.

[133] A. C-C. Yao. Some complexity questions related to distributive computing. In *Proceedings of 11th ACM STOC*, pages 209–213, 1979.

[134] A. C-C. Yao. Quantum circuit complexity. In *Proceedings of 34th IEEE FOCS*, pages 352–360, 1993.