Quantum computation and information: Notes for Spring 2020 TAMU class

J.M. Landsberg

Contents

Chapter 1

Classical and probabilistic computing

1.1. 2025

In January 2016, and in more detail in October, the NSA released a document warning the world that current encryption algorithms will be no longer secure as early as 2025[1]. At that point in time there may be operational quantum computers. What is all the fuss about?

The main way banks, governments, etc. communicate securely now is using the RSA cryptosystem. RSA relies on the assumption that it is difficult to factor a large number N into its prime factors. In 1994 [**Sho94**] (also see [**Sho97**]) P. Shor described an algorithm to factor numbers quickly on a "quantum computer".

Why can't we factor numbers quickly already? What is a quantum computer?

Before addressing these questions, we need to address more basic ones:

What computations *can* we do quickly on a computer? What is a classical computer and what can it do?

[1] See http://www.math.tamu.edu/~jml/CNSA-Suite-and-Quantum-Computing-FAQ.pdf

1.2. Surprising algorithms

1.2.1. Logarithms: fast multiplication of numbers. Until the 1600's, when people had to do astronomical predictions (the king was very interested in knowing his horoscope, see [**Lyo09**]), a difficult step was the multiplication of large numbers. In 1614 John Napier revolutionized computation by writing a book of lists of numbers to implement a transform that swaps multiplication for addition: the logarithm. Kepler used Naiper's book to make astronomical tables on the order of 30 times more accurate of previous tables [**Gle11**, p87].

Even in this example, there is something modern to learn: if the difficult step of a calculation (in this case taking logs and exponentiation) can be precomputed and stored in a database, it becomes essentially "free".

Exercise 1.2.1: Show that if we are allowed the binary operations $\oplus$ and AND (multiplication), we can multiply integers m, n using $O(mn)$ operations. (Expand m, n in binary.)

Exercise 1.2.2: The famous mathematician Kolmogorov conjectured that one could not multiply two n digit numbers using less than n^2 one-digit multiplications. His student Karatsuba showed in fact that one can use less than $n^{1.6}$ one-digit multiplications. Look up Karatsuba's algorithm (e.g., on Wikipedia) and use it to compute 42597×3232.

1.2.2. The DFT: Fast multiplication of polynomials. Say $a(x), b(x)$ are polynomials of degree at most d. Write $a(x) = \sum_{i=0}^{d} a_i x^i$, $b(x) = \sum_{j=0}^{d} b_j x^j$. Write $\overline{a} = (a_0, \ldots, a_d)^{\mathbf{t}}$ and similarly for other coefficients. Writing $a(x)b(x) = \sum_{k=0}^{2d} c_k x^k$, one has

$$c_k = \sum_{i+j=k} a_i b_j. \tag{1.2.1}$$

(One says $\overline{c}$ is the *convolution* of $\overline{a}$ and $\overline{b}$.) To obtain the coefficient vector $\overline{c}$ by this standard method, one needs to perform on the order of d^2 arithmetic operations (i.e., $+$'s and $*$'s). In this situation, we will write $O(d^2)$ arithmetic operations, see §1.3.1 below for the precise definition of $O(d^2)$.

Quantum algorithms will be expressed as a sequence of matrix vector multiplications, and we may do so here as well to facilitate comparisons.

To express this calculation in terms of matrix-vector multiplication, note that the vector $\overline{c}$ is the product

$$\begin{pmatrix} a_0 & 0 & & \cdots & 0 \\ a_1 & a_0 & 0 & \cdots & 0 \\ a_2 & a_1 & a_0 & \ddots & \\ \vdots & & \vdots & & \\ a_d & a_{d-1} & & \cdots & a_0 \\ 0 & a_d & a_{d-1} & \cdots & 0 \\ \vdots & & \ddots & & \\ 0 & \cdots & & 0 & a_d \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_d \end{pmatrix}.$$

Here we have broken the symmetry between $a(x)$ and $b(x)$. The symmetry will be restored momentarily.

Now we explain a trick to reduce the amount of computation, which was discovered by Schönhage and Strassen [**SS71**]. Pay attention as a variant of this trick will be critical to Shor's quantum algorithm for factoring. As with the multiplication of numbers, the key will be to do a transformation that re-organizes the input data of the two polynomials.

Up until now I have not mentioned what the coefficients of the polynomials are: they could be integers, rational numbers, real numbers or complex numbers. To simplify what comes next, I will allow complex numbers. Since $\deg(ab) \leq 2d$, instead of working in the space of all polynomials, we can work in the ring $\mathbb{C}[x]/(x^N - 1)$ of polynomials quotiented by the ideal generated by the polynomial $x^N - 1$ for any $N > 2d$. (Elements of this ring are equivalence classes of polynomials, where $x^N \equiv 1$ generates the equivalence relation.) For the moment, to fix ideas set $N = 2d + 1$, but later we will take N to be a power of two. We can then write a $(2d+1) \times (2d+1)$ matrix for $a(x)$ (allowing it now to have larger degree) as

$$\begin{pmatrix} a_0 & a_{2d} & a_{2d-1} & \cdots & a_2 & a_1 \\ a_1 & a_0 & a_{2d} & \cdots & a_3 & a_2 \\ a_2 & a_1 & a_0 & \ddots & & \\ \vdots & & \vdots & & & \\ a_d & a_{d-1} & & \cdots & a_{d+2} & a_{d+1} \\ a_{d+1} & a_d & a_{d-1} & \cdots & a_{d+3} & a_{d+2} \\ \vdots & & \ddots & & & \\ a_{2d} & \cdots & & & a_1 & a_0 \end{pmatrix}$$

and similarly for $b(x)$ (although we only need the first column of the product).

Note that the first $d+1$ columns of this matrix is our old matrix. This looks like we are making our problem more complicated. However, now that we have a square matrix, call it A, we can *diagonalize* it. Call the diagonalized matrix $\hat{A}$. Even better, and to restore symmetry, we could write the corresponding matrix for $b(x)$, and we observe that A, B are simultaneously diagonalizable. This is because $a(x)b(x) = b(x)a(x)$, in both the usual multiplication of polynomials and as elements of the $\mathbb{C}[x]/(x^N - 1)$, and if commuting matrices are diagonalizable, they are simultaneously diagonalizable. (In other words, the map $\mathbb{C}[x]/(x^N - 1) \to Mat_{N\times N}$ is a ring homomorphism, which insures the corresponding matrices will commute.) Thus to compute the matrix $\hat{C}$, we only need to perform $N = O(d)$ scalar multiplications instead of $O(d^2)$!

Exercise 1.2.3: Show that if two diagonalizable matrices commute, then they are simultaneously diagonalizable. ◎

Exercise 1.2.4: Is the same true if one has three diagonalizable commuting matrices? Give a sufficient condition that makes it true.

However, this seems like a very bad idea: the cost of a change of basis is worse than $O(d^2)$! Moreover, we will have to un-diagonalize $\hat{C}$ to get the coefficients of $c(x)$ (which are the elements of the first column of C). The punch line will be that there is a relatively cheap way to carry out the diagonalization.

Consider the linear map that sends the coefficient vector of a polynomial of degree at most N to the vector consisting of eigenvalues of the corresponding $N \times N$ matrix as above. Let $DFT_N : \mathbb{C}^N \to \mathbb{C}^N$ denote this linear map. (DFT stands for *discrete Fourier transform.*) Write $\hat{a} = DFT_N\overline{a}$ (where we have padded the coefficient vector of $a(x)$ with zeros to make it have length N), and similarly $\hat{b} = DFT_N\overline{b}$. Given $\hat{a}$ and $\hat{b}$, the vector $\hat{c}$ can be computed using N scalar multiplications as $\hat{c}_k = \hat{a}_k\hat{b}_k$. Finally $\overline{c} = {DFT_N}^{-1}\hat{c}$.

Notation. Throughout this book we set $i = \sqrt{-1}$.

Proposition 1.2.5. *The matrix representing DFT_N is given by*

$$(DFT_N)_{jk} = (e^{\frac{2\pi i}{N}})^{jk} \tag{1.2.2}$$

and its inverse is given by $({DFT_N}^{-1})_{jk} = \frac{1}{N}(e^{\frac{2\pi i}{N}})^{-jk}$. Here use index ranges $0 \leq j, k \leq N-1$.

Proof. The j-th column vector of (1.2.2) is an eigenvector for multiplication by x with eigenvalue $e^{\frac{-j2\pi i}{N}}$. To see this consider (the equivalence class of) $x(1+e^{\frac{j2\pi i}{N}}x+e^{\frac{2j2\pi i}{N}}x^2+\cdots+e^{\frac{(N-1)j2\pi i}{N}}x^{N-1})$ and notice that $e^{\frac{(N-1)j2\pi i}{N}}x^N \equiv e^{\frac{-j2\pi i}{N}}(1)$. □

Exercise 1.2.6: Show that there is an isomorphism between $\mathbb{Z}/N\mathbb{Z}$ and $\{e^{\frac{2\pi iu}{N}} | 0 \leq u \leq N-1\}$ under multiplication of complex numbers.

Exercise 1.2.7: Write computer code to verify that DFT_4 indeed diagonalizes matrices corresponding to elements of $\mathbb{C}[x]/(x^4-1)$ and use it to multiply two degree one polynomials.

Remark 1.2.8. For those familiar with representation theory, the DFT is the change of basis matrix from the standard basis of the regular functions on the cyclic group of order N, $\mathbb{Z}_N = \mathbb{Z}[x]/(x^N-1)$ to the character basis. (The roots of unity appearing in the DFT matrix are the characters of the cyclic group.) Thus using representation theory, it is straight-forward to derive the DFT matrix.

Now we come to a great discovery of Gauss in 1810 [**Gau**], rediscovered by several people, including Cooley-Tukey in 1965 [**CT65**], who are responsible for its modern implementation the revolutionized signal processing: the DFT matrix factors as a product of sparse matrices. Explicitly, if $N = 2^k$, DFT_N may be written as a product of k matrices, each with only $2N$ nonzero entries. The cost of matrix-vector multiplication of a sparse matrix with S nonzero entries is $O(S)$, so the cost of performing our DFT is $O(\log_2(N)N)$ instead of $O(N^2)$. Performing three such, plus the diagonal matrix multiplication does not change the order of this total cost.

Explicitly,

$$DFT_{2M} = \begin{pmatrix} DFT_M & \Delta_M DFT_M \\ DFT_M & -\Delta_M DFT_M \end{pmatrix} \Pi \tag{1.2.3}$$

where, setting $\omega = e^{\frac{2\pi i}{2M}}$, $\Delta_M = \mathrm{diag}(1, \omega, \omega^2, \ldots, \omega^{M-1})$ and Π is a permutation matrix corresponding to the inverse of the shuffle permutation $(1, \ldots, 2M) \mapsto (1, 3, 5, \ldots, 2M-1, 2, 4, 6, \ldots, 2M)$.

Exercise 1.2.9: Write DFT_4 as a product of two matrices, one of which has only four nonzero entries. Write DFT_8 as a product of three matrices, respectively with $8, 32$ and 16 nonzero entries. ◎)

Exercise 1.2.10: Verify Equation (1.2.3).

Exercise 1.2.11: Show that DFT_{2^k} may be factored as a product $S_1 \cdots S_k$ where the total number of nonzero entries in the k matrices is $O(k2^{k+1}) = O(\log(d)d)$, and thus multiplication of two polynomials of degree at most $d = 2^{k-1}$ may be computed using $O(k2^{k+1}) = O(\log(d)d)$ arithmetic operations. ◎

Remark 1.2.12. You may have seen Fourier transforms of periodic functions, where convolution in the original space corresponds to multiplication

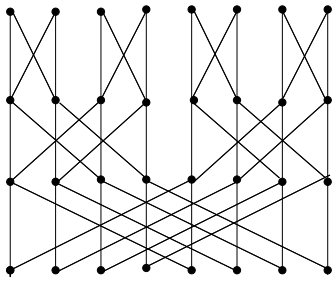

Figure 1.2.1. Graph representing product of three 8×8 matrices that gives DFT_8. Vertices in each row represent indices from 1 to 8, and edge from i to j at level $k \in \{1, 2, 3\}$ means the (i, j)-th entry of the k-th matrix is nonzero.

in the transform space. This is the analogous transform when the group is the circle.

More explicitly, write the unit circle in $\mathbb{R}^2$ as $S^1 = \{(\cos(\theta), \sin(\theta)) \mid \theta \in [0, 2\pi)\} \subset \mathbb{R}^2$. Introduce complex notation $\mathbb{C} = \mathbb{R}^2$, so $S^1 = \{e^{i\theta} \mid \theta \in [0, 2\pi)\} \subset \mathbb{C}$. Then for (e.g., continuous) functions $f(\theta)$ on the unit circle, we may write

$$f(\theta) = \sum_{n=-\infty}^{\infty} c_n e^{\frac{in\theta}{2}}, \text{ where } c_n = \frac{1}{4\pi} \int_0^{2\pi} f(\theta) e^{-\frac{in\theta}{2}} d\theta.$$

Since nonzero complex numbers form a group under multiplication, and the product of elements of length one is of length one, S^1 is naturally a group. In signal processing, we need to digitize (e.g. sound waves), so we approximate a periodic function by sampling it at say N equally spaced points on the circle, e.g., the points $e^{\frac{k2\pi i}{N}}$, $0 \leq k \leq N-1$. Note that these points form a subgroup, in fact the cyclic group of order N, $\mathbb{Z}[x]/(x^N - 1)$. Tracing through the calculation, the DFT really is the discretization of the Fourier transform on the circle, exactly what one needs in signal processing.

Aside 1.2.13. For those familiar with tensors and their ranks, the structure tensor of $\mathcal{A} = \mathbb{C}[x]/(x^N - 1)$ has minimal tensor rank N, and the DFT is a change of basis that rewrites the structure tensor $T_{\mathcal{A}} \in \mathcal{A}^* \otimes \mathcal{A}^* \otimes \mathcal{A}$ as a sum of rank one tensors.

Aside 1.2.14. One might wonder if there is an even more efficient way of computing the operation $\overline{a} \mapsto DFT_N \overline{a}$. This question, and a path to resolving it, were presented by L. Valiant in [**Val77**]. There is interesting algebraic geometry related to the question, see [**KLPSMN09, GHIL16**].

1.2.3. Integer multiplication. The same Schönhage-Strassen idea above can be used to multiply integers a, b, if $a, b < 2^{\frac{n}{2}}$, multiply them in the ring

$\mathbb{Z}/2^n\mathbb{Z}$. The result, if each addition and multiplication of bits has unit cost, is $O(n\log(n)\log(\log(n)))$.

exercises here to realize*

1.2.4. Matrix multiplication. Another surprising algorithm deals with matrix multiplication. The usual algorithm for multiplying two $n \times n$ matrices uses $O(n^3)$ arithmetic operations. Strassen [**Str69**] discovered an algorithm that uses $O(n^{2.81})$ arithmetic operations and it has been conjectured that as n grows, it becomes nearly as easy to multiply matrices as it is to add them, that is for any $\epsilon > 0$, one can multiply matrices using $O(n^{2+\epsilon})$ arithmetic operations.

1.3. Notation, probability and linear algebra

1.3.1. Big/Little O etc. notation. For functions f, g of a real variable (or integer) x:

$f(x) = O(g(x))$ if there exists a constant $C > 0$ and x_0 such that $|f(x)| \leq C|g(x)|$ for all $x \geq x_0$,

$f(x) = o(g(x))$ if $\lim_{x\to\infty} \frac{|f(x)|}{|g(x)|} = 0$,

$f(x) = \Omega(g(x))$ if there exists a constant $C > 0$ and x_0 such that $C|f(x)| \geq |g(x)|$ for all $x \geq x_0$,

$f(x) = \omega(g(x))$ if if $\lim_{x\to\infty} \frac{|g(x)|}{|f(x)|} = 0$, and

$f(x) = \Theta(g(x))$ if $f(x) = O(g(x))$ and $f(x) = \Omega(g(x))$.

We write ln for the natural logarithm and log for $\log_2$.

1.3.2. Probability. Let $\mathcal{X} = \{a_1, a_2, ...\}$ be a countable set and let $p : \mathcal{X} \to [0, 1]$ be a function such that $\sum_j p(a_j) = 1$. Such p is called a *discrete probability distribution* on $\mathcal{X}$. A function $X : \mathcal{X} \to \mathbb{R}$ is called a *discrete random variable* and it defines a probability distribution with discrete support on $\mathbb{R}$ by $p_X(z) = \sum_{j|X(a_j)=z} p(a_j)$ so $p_X(z) = 0$ if $z \notin X(\mathcal{X})$. Similarly, random variables X, Y define a probability distribution $p_{X,Y}(x, y)$ with discrete support on $\mathbb{R} \times \mathbb{R}$, and similarly n random variables define a probability distribution with discrete support on $\mathbb{R}^n$. If $f : \mathbb{R} \to \mathbb{R}$ is a function, then $f \circ X$ is also a random variable. If P is a probability distribution on $\mathcal{X} \times \mathcal{X}'$, one defines the *marginals* by $P_{\mathcal{X}}(x) = \sum_{y\in\mathcal{X}'} P(x, y)$ and $P_{\mathcal{X}'}(y) = \sum_{x\in\mathcal{X}} P(x, y)$, which are probability distributions on $\mathcal{X}$, $\mathcal{X}'$ respectively.

1.3.3. Linear algebra. Terms such as vector space, linear map etc.. will be assumed. For a vector space V, over a field $\mathbb{F}$, recall the *dual space* $V^* := \{f : V \to \mathbb{F} \mid f \text{ is linear}\}$. If $V = \mathbb{F}^n$ is the space of column vectors,

then V^* may be identified with the space of row vectors. If V is finite dimensional, there is a canonical isomorphism $V \to (V^*)^*$, so we may also think of V as the space of linear maps $V^* \to \mathbb{F}$. Following physics convention, we will usually denote elements of V by $|v\rangle$ and elements of V^* by $\langle\alpha|$, and their pairing by $\langle\alpha|v\rangle$. In bases, if

$$|v\rangle = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix},$$

we may write $\langle\alpha| = (\alpha^1 \cdots \alpha^n)$, and $\langle\alpha|v\rangle = \sum_j \alpha^j v_j$ is row-column matrix multiplication. Let $\mathrm{End}(V)$ denote the space of linear maps $V \to V$.

Define the *tensor product* $V \otimes W$ of vector spaces V and W to be the space of bi-linear maps $V^* \times W^* \to \mathbb{F}$, and more generally for a collection of vector spaces $V_1, \ldots, V_m$, $V_1 \otimes \cdots \otimes V_m$ is the space of m-linear maps $V_1^* \times \cdots \times V_m^* \to \mathbb{F}$. If we work in bases, and $\dim V_j = \mathbf{v}_j$, then $V_1 \otimes V_2$ is the space of $\mathbf{v}_1 \times \mathbf{v}_2$-matrices and $V_1 \otimes V_2 \otimes V_3$ may be visualized as the space of $\mathbf{v}_1 \times \mathbf{v}_2 \times \mathbf{v}_3$ "three dimensional matrices".

Given a linear map $f : V \to W$, we may define a second linear map $f^{\mathbf{t}} : W^* \to V^*$, by, for $\beta \in W^*$, $f^{\mathbf{t}}(\beta)(v) = \beta(f(v))$. This is the coordinate free definition of the transpose of a matrix. One may also define a bilinear map $W^* \times V \to \mathbb{C}$, by $(\beta, v) \mapsto \beta(f(v))$ which extends to a linear map $W^* \otimes V \to \mathbb{C}$. Thus we may also think of $V \otimes W$ as the set of bilinear maps $V^* \times W^* \to \mathbb{C}$. Consider a 2×3 matrix and its roles respectively as a linear map $\mathbb{C}^3 \to \mathbb{C}^2$, a linear map $\mathbb{C}^{2*} \to \mathbb{C}^{3*}$ and a bilinear map $\mathbb{C}^{2*} \times \mathbb{C}^3 \to \mathbb{C}$:

$$\begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} ax + by + cz \\ dx + ey + fz \end{pmatrix}, \quad \begin{pmatrix} s & t \end{pmatrix} \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} = \begin{pmatrix} sa + td \\ sb + te \\ sc + tf \end{pmatrix},$$

$$\begin{pmatrix} s & t \end{pmatrix} \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = sax + tdx + sby + tey + scz + tfz.$$

1.4. Classical Complexity

Classical complexity works in binary: one deals with strings of 0's and 1's. The set $\{0, 1\}$ is called a *bit*: it can encode "one bit" of information.

1.4.1. Circuits. We will mostly deal with *circuits*: Boolean circuits for classical computation, Boolean circuits with access to randomness for probabilistic computation, and quantum circuits for quantum computation.

Let $\mathbb{F}_2$ denote the field with two elements $\{0, 1\}$. A *Boolean function* is a map $f : \mathbb{F}_2^n \to \mathbb{F}_2$, or more generally $\mathbb{F}_2^n \to \mathbb{F}_2^m$. We agree on some basic

Boolean functions, whose complexity is designated as having unit cost, e.g., addition $\oplus$ (also called XOR) where $a \oplus b$ is addition in $\mathbb{F}_2$ (i.e., $0 \oplus 0 = 1 \oplus 1 = 0$ and $0 \oplus 1 = 1 \oplus 0 = 1$), $\neg$ (NOT) negation, which swaps 0 and 1, (OR) $a \vee b$ where $0 \vee 0 = 0$ and all other $a \vee b = 1$, (AND= multiplication in $\mathbb{F}_2$) $a \wedge b = ab$, where $1 \wedge 1 = 1$ and all other $a \wedge b = 0$. I will call such a collection a *(logic) gate set*.

A *Boolean circuit* is a representation of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ as a directed graph with n input edges, vertices labeled by elements of some fixed gate set, with edges going in and out, and m output edges. The *size* of a circuit is the number of edges in it.

Call a gate set a *universal gate set* if any Boolean function can be computed with a circuit whose vertices are labeled with gate set elements.[2]

Figure 1.4.1 depicts a Boolean circuit for the addition of two two digit (in binary) numbers:

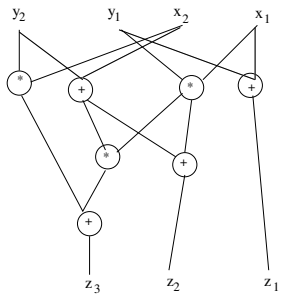

Figure 1.4.1. Circuit for $y_2y_1 + x_2x_1$

We began by saying factorization is not known to have an efficient algorithm. We can now make that precise: a classical algorithm for a task (such as factoring) is *efficient* if there exists a polynomial p, such that if the input (in the case of factoring, the number to be factored N expressed in binary) is of size M (in the case of factoring, the expression of N in binary has at most $M = \log N$ digits), then there exists a Boolean circuit of size $p(M)$ that accomplishes the task. We now rephrase this more formally:

1.4.2. P/*poly*, P and NP. Fix a universal gate set G. A natural complexity measure for a Boolean function is then the minimal size of a Boolean circuit that computes it. Let $p(n)$ be a polynomial and let $F_n : \mathbb{F}_2^n \to \mathbb{F}_2^{p(n)}$ be a sequence of functions (F_n). (We often will just have $p(n) = 1$.) We consider the the growth with n of the size of a circuit needed to compute F_n. The critical issue, according to complexity theorists, is whether or not

[2]In some of the literature a gate set is sometimes called a "basis" (despite being unrelated to bases of vector spaces) and a universal gate set is called a "complete basis".

this growth is bounded by a polynomial. If it grows like a polynomial we say the sequence (F_n) is in the class $\mathbf{P}/poly$ with respect to G.

Exercise 1.4.1: Show that membership in $\mathbf{P}/poly$ with respect to G is independent of the choice of the finite universal gate set G.

Thus we will just say that the sequence (F_n) is in the class $\mathbf{P}/poly$.

Another way to phrase the class $\mathbf{P}/poly$ is that for each n, there exists a circuit of size $poly(n)$ that solves the problem, but the circuits are allowed to be different for each n. (In particular the description of the circuit may grow exponentially.)

The famous complexity class $\mathbf{P}$ is the standard model for feasible computations, and it is unfortunate that $\mathbf{P}/poly$, with a simple description, has a different definition than $\mathbf{P}$. The definition is less restrictive, but not so much less so, and it can be used as a substitute for $\mathbf{P}$, see [**AB09**, Chap. 6]. It is known that $\mathbf{P} \subsetneq \mathbf{P}/poly$, however membership in $\mathbf{P}/poly$ for many problems of interest is not known.

The class $\mathbf{P}$ is usually defined in terms of a different model of computation, namely *Turing machines*. We will avoid defining them, and assume the reader has at least a passing familiarity with them. A function F is in $\mathbf{P}$ if it is in $\mathbf{P}/poly$ and there exists a Turing Machine TM such that the circuits C_n computing F_n are constructed by TM in time $poly(n)$, see [**KSV02**, Thm. 2.3].

The famous class $\mathbf{NP}$ essentially consists of problems whose proposed solutions can be verified quickly, i.e., in polynomial time. For example the traveling salesman problem, where if someone claims to have a route to visit 30 cities traveling less than 2000 miles, it is easy to verify the claim by examining the route, but the only known way of finding such a route is essentially by a brute force search. Another problem in $\mathbf{NP}$ is "SAT": one is handed a Boolean circuit and wants to know if it ever outputs 1. (If it does, to convince you it does, someone just needs to hand you an input that works, and you can quickly check if it outputs 1.) SAT is $\mathbf{NP}$-complete, which means one could define $\mathbf{NP}$ to be the collection of problems that can be *reduced* (in polynomial time) to SAT, see, e.g., [**AB09**, Chap. 2]. In other words, there is a polynomial time algorithm for SAT if and only if $\mathbf{P} = \mathbf{NP}$.

1.4.3. How does a (classical) computer work? One can build mechanical devices that implement the classical gates. In our computers, logic gates are made out of electrical circuits. Input is either a 5 volt impulse for 1 and no impulse for 0. For example, the NOT gate is realized by a the following diagram **** from top to bottom, there is a voltage source, a connection to an output wire the input source, and a ground .....

the NAND gate is realized by the following diagram A voltage source is connected .....

1.4.4. Reversible classical computation. We will see that the gates of a quantum circuit (other than the measurements) must be reversible. Before quantum computing, researchers were concerned that the second law of thermodynamics would have the consequence that as computers got more powerful, they would generate too much heat (entropy) from erasing bits. One way out of this would be to have reversible computation, see [**Lan61**], so one could argue for it independent of quantum computation. Of the gates we saw, NOT is clearly reversible as $\neg\neg x = x$. At the cost of adding an extra bit, one can make addition and multiplication reversible. Consider the following gate, called to *Toffoli gate* Tof:

$$|x, y, z\rangle \mapsto |x, y, z \oplus (x * y)\rangle = |x, y, z \oplus (x \wedge y)\rangle \tag{1.4.1}$$

Note that if we send in $|x, y, 0\rangle$ we obtain $x * y$ in the third slot (register) and if we send in $|x, 1, y\rangle$ we obtain $x \oplus y$.

Exercise 1.4.2: Show that $Tof \circ Tof = \mathrm{Id}$, so Tof is indeed reversible.

The gate set $\{Tof, \neg\}$, is universal and reversible, so there is no loss in computing power restricting to reversible classical computation.

1.5. Probabilistic computing

We will develop quantum mechanics as a generalization of probability, and we will view quantum computing as a generalization of probabilistic computing.

We will want to see the improvement of quantum computing to classical computing, so we should understand the what can be computed efficiently on a computer with access to randomness. Quantum computing itself is probabilistic, so we will need to implement notions from probability. Rather than introduce both the quantum-ness and the probabilistic nature at the same time, it will be easier to digest them one at a time.

1.5.1. BPP. It might increase our computational power if we exploit randomness. (Assuming we have a method to generate random numbers - more on this later.) For example, if someone hands you a complicated expression for a polynomial, e.g., in terms of an (algebraic) circuit, it can be very difficult to determine if the polynomial is just the zero polynomial in disguise. If we test the polynomial at a point, and its evaluation is non-zero, then we know it is not the zero polynomial. If if does evaluate to zero, then we have no information. For a polynomial of degree d in one variable, it is sufficient to test $d+1$ distinct points, but as the number of variables grows,

the number of points one needs to check grows exponentially. However, if we are allowed to test at a random point and it evaluates to zero, then with high probability over finite fields and probability one over $\mathbb{Z}$, the polynomial is the zero polynomial. Over finite fields, we can make this probability as high as we want by testing on several random points.

These observations motivate the class **BPP** (short for "bounded-error probabilistic polynomial time"), where one works with a Turing machine with access to randomness, and instead of asking for a correct answer on any input in polynomial time, one asks for a correct answer with probability strictly greater than $\frac{1}{2}$ on any input in polynomial time. (So that if one runs the program enough times, one can get a correct answer on any input with probability as high as one wants.)

Another motivation for probabilistic computation is that physical computers sometimes make mistakes (e.g. short circuit, input misread), so in the real world we are never completely sure of our answers.

Remark 1.5.1. It is actually subtle to know if one has a random sequence of numbers (e.g., take the last digit of the temperature in binary or similar). For example, the first digit of the number 2^n is far from a random element of $\{1, \dots, 9\}$, see [**Ad89**, §16, Ex. 4]. It is a subtle problem to make a machine to generate random numbers for us. Fortunately, for most situations, *pseudo-random* numbers suffice, see [**AB09**, §9.2.3].

Aside 1.5.2. If we are given additional information about the polynomial, then under certain circumstances one can test if the polynomial is zero by testing a reasonable number of points. This subject PIT (polynomial identity testing) is an active area of research, see [**AB09**, §7.2.3]. For a geometric perspective see [**Lan17**, §7.7].

Probabilistic computation however *cannot* be made reversible on a classical computer, as we will see in §1.6.3.

1.6. Computation via linear algebra

(Following [**AB09**, Exercise 10.4]) We will encode computation of a circuit as a sequence of matrix-vector multiplications.

1.6.1. Tensor products of spaces. Give $\mathbb{R}^2$ basis $|0\rangle, |1\rangle$, which induces the basis $|i\rangle\otimes|j\rangle$, $i, j \in \{0, 1\}$ of $(\mathbb{R}^2)^{\otimes 2}$ and $|I\rangle := |i_1\rangle\otimes\cdots\otimes|i_N\rangle$ of $(\mathbb{R}^2)^{\otimes N}$, $i_\alpha \in \{0, 1\}$, $1 \leq \alpha \leq N$. E.g., if our bit string is 00101100, we represent it by the vector $|00101100\rangle \in \mathbb{R}^{2^8}$.

Sometimes it is also convenient to represent vectors as column vectors. There is no canonical change of basis to the basis $|I\rangle$ to the standard basis

of column vectors, we'll simply use the lexicographic one, so the vector

$$a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle \in \mathbb{R}^2 \otimes \mathbb{R}^2 \simeq \mathbb{R}^4$$

would get identified with the column vector

$$\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}.$$

and the linear map $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \mathrm{Id}_2$ would be written as the 4×4 matrix

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

1.6.2. Reversible classical computation. Say $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ can be computed by a reversible Boolean circuit C. We describe how to rephrase the computation as a sequence of restricted linear operations on a vector space containing $\mathbb{R}^{2^n}$ in anticipation of what will come in quantum computation. The restrictions will be:

(1) Each matrix must be invertible and take a vector representing a sequence of bits to a sequence of bits. Such matrices are *permutation matrices*. That is, each row and column has exactly one entry equal to one, and all other entries zero.

(2) In order to deal with finite gate sets, we will require that each matrix only alters a small number of entries. For simplicity we assume it alters at most three entries, i.e., it acts on at most $\mathbb{R}^{2^3}$ and is the identity on all other factors in the tensor product.

Each map will imitate some Boolean gate. For example, say we want to effect the Toffoli gate,

$$|x, y, z\rangle \mapsto |x, y, z \oplus (x * y)\rangle = |x, y, z \oplus (x \wedge y)\rangle$$

and act as the identity on all other basis vectors (sometimes called *registers*). Here, if the Toffoli gate is to compute $x * y$ and z will be a "workspace bit": x, y will come from the input to the problem and z will be set to 0 in the input. In the basis $|000\rangle, |001\rangle, |010\rangle, |100\rangle, |011\rangle, |101\rangle, |110\rangle, |111\rangle$, of $\mathbb{R}^8$,

the matrix is

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{1.6.1}$$

Call this matrix the *Toffoli matrix*.

The negation gate $\neg$ may be defined by the linear map $\mathbb{R}^2 \to \mathbb{R}^2$ given by the matrix

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

Exercise 1.6.1: Write matrices for $(x, y, z) \mapsto (x, y, z \oplus (x \oplus y))$ and $(x, y, z) \mapsto (x, y, z \oplus (x \vee y))$.

In summary, the computation of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ via a reversible classical circuit has input $|x, I\rangle \in \mathbb{R}^{n+s}$, where x is the input and I some initialization (often just a string of zeros), and the output is of the form $|y, J\rangle$ where the $y \in \mathbb{F}_2^m$ is the desired answer.

1.6.3. Probabilistic computation via linear algebra. If on given input, a probabilistic computation outputs 0 with probability p and 1 with probability $1 - p$, we could encode this with the vector $p|0\rangle + (1 - p)|1\rangle$, and then obtain either 0 or 1 by flipping a biased coin that gives heads with probability p.

Say $f : \mathbb{F}_2^n \to \mathbb{F}_2$ can be computed correctly with probability greater than $\frac{1}{2}$ by a Boolean circuit C that is allowed to access randomness. (In particular, we can compute f correctly with probability as close as we want to one by repeating the computation enough times.) To represent a coin flip in terms of linear algebra, introduce the matrix:

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

(Classicial) probabilistic computation cannot be made reversible. This is indicated by the fact that this matrix is not invertible.

Consider $\{0, 1\}^m \subset \mathbb{R}^{2^m}$. A probability distribution on $\{0, 1\}^m$ may be encoded as a vector in $\mathbb{R}^{2^m}$: Give $\mathbb{R}^2$ basis $|0\rangle, |1\rangle$ and $(\mathbb{R}^2)^{\otimes m} = \mathbb{R}^{2^m}$ basis $|I\rangle$ where $I \in \{0, 1\}^m$. If the probability distribution assigns probability p_I to $I \in \{0, 1\}^m$, assign to the distribution the vector $v = \sum_I p_I |I\rangle \in \mathbb{R}^{2^m}$.

For probabilistic computation via linear algebra we will require the following of our matrices:

(1) Each linear map must take vectors representing probability distributions to vectors representing probability distributions. This implies the matrices are *stochastic*: the entries are non-negative and each column sums to 1.

(2) In order to deal with finite gate sets, we will require that each linear map only alters a small number of entries. For simplicity we assume it alters at most three entries, i.e., it acts on at most $\mathbb{R}^{2^3}$ and is the identity on all other factors in the tensor product.

We will work with $\mathbb{R}^{2^{n+s+r}}$ where r is the number of times we want to access a random choice and s are "workspace bits" as before.

Exercise 1.6.2: In probabilistic algorithms we will want to choose an element uniformly at random from a a set of M elements. How can we approximately realize this choice with the above matrices? (We can realize it exactly when M is a power of two.)

A probabilistic computation of $f : \mathbb{F}^n \to \mathbb{F}^m$, viewed this way, starts with $|x0^{r+s}\rangle$, where $x \in \mathbb{F}_2^n$ is the input. One then applies a sequence of admissible stochastic linear maps to it, and ends with a vector $|v\rangle = \sum p_I|I\rangle$ that encodes a probability distribution on $\{0,1\}^{n+s+r}$. One then takes the marginal distribution on the first m copies of $\mathbb{R}^2$, i.e., write $I = (i_1, \ldots, i_{n+r+s}) = (J, L)$, where $J = (i_1, \ldots, i_m)$, and take

$$\sum_J (\sum_L p_{J,L}|J\rangle.$$

The algorithm then outputs $J \in \{0,1\}^m$ with probability $\sum_L p_{J,L}$. Note that even if our calculation was "feasible" (i.e., polynomial in n size circuit), to write out the original output vector that we truncate would be exponential in cost because of the rational coefficients. (More precisely the coefficients are rational numbers whose denominators are powers of two.) A stronger variant of this phenomenon will occur with quantum computing, where the result will be obtained with a polynomial size calculation, but one does not have access to the vector created, even using an exponential amount of computation.

To further prepare for the analogy with quantum computation, define a probabilistic bit (a *pbit*) to be an element of the set

$$\{p_0|0\rangle + p_1|1\rangle \mid p_j \in [0,1] \text{ and } p_0 + p_1 = 1\} \subset \mathbb{R}^2.$$

Note that the set of pbits is a convex set, and the basis vectors are the extremal points of this convex set.

Exercise 1.6.3: Show that if we have two problems to solve, one in $(\mathbb{R}^2)^{\otimes m}$ and another in $(\mathbb{R}^2)^{\otimes n}$, and we want to solve them simultaneously via linear algebra, then we should work in $(\mathbb{R}^2)^{\otimes m} \otimes (\mathbb{R}^2)^{\otimes n} = (\mathbb{R}^2)^{\otimes n+m}$.

1.6.4. What is known. $\mathbf{P} \subseteq \mathbf{BPP} \subseteq \mathbf{P}/Poly = \mathbf{BPP}/Poly$.

The inclusion $\mathbf{BPP} \subseteq \mathbf{P}/Poly$ is Adelman's theorem [**Adl78**]. The key observation is that "off-line" computations are not counted in the complexity assessment. So one can create, for any given n, a library of "random" a's to test on. For example, to correctly determine the primality of 32-bit numbers, it is enough to test $a = 2, 7$, and 61.

Does randomness really help? At the moment, we don't know. See [**AB09**, Chap. 20] for a discussion.

1.6.5. BQP. We are not yet in a position to define it, but the class **BQP** will be the quantum analog of **BPP**, the problems that can be solved efficiently, with high probability, on a quantum computer. Pbits will be replaced by *qubits*, which are unit vectors in $\mathbb{C}^2$ subject to an equivalence relation. The matrices will be allowed to have complex entries, they will be required to be unitary instead of stochastic, and at the end of the computation, one will not have the resulting vector in hand, but the result of a projection operator applied to it. The probability of obtaining I_0 from $\sum_I z_I|I\rangle$ will be $|z_{I_0}|^2$. There is no analog of **P** for a quantum computer as answers will always have a probability of being incorrect.

A subtlety about quantum gate sets is that the notion of a "universal quantum gate set" will have a different meaning, namely that one can approximate any unitary map arbitrarily closely by elements of the gate set, not that one can perform the map exactly. This is similar to, but stronger than, the situation in probabilistic computation, where we can only exactly achieve probability distributions with numerators that are powers of two.

1.6.6. The Church-Turing theses. The Church-Turing thesis (made explicitly by Church in [**Chu36**]) is:

Any algorithm can be realized by a Turing machine.

So far there has been no challenge to this - e.g., any computation that can be done on a quantum computer can be done with a sufficiently large Turing machine.

The quantitative (sometimes called *strong*) Church-Turing thesis [**VSD86**] is:

Any algorithmic process can be simulated efficiently by a Turing machine

or

Any algorithmic process can be simulated efficiently by a probabilistic Turing machine

Shor's algorithm challenges this thesis. On the other hand, there are experts who think that factoring could be in **P**, because unlike, say SAT or the traveling salesman problem, the problem is highly structured.

Chapter 2

Quantum mechanics for quantum computing

This chapter covers basic quantum mechanics needed for quantum computing. I present quantum mechanics as a generalization of probability and quantum computing will be viewed as a generalization of probabilistic computing.

2.1. Quantum mechanics via probability

2.1.1. A wish list. In §1.6 we saw that any $f : \{0,1\}^n \to \{0,1\}^m$ that could be computed correctly with probability say at least $\frac{2}{3}$ on any $I \in \{0,1\}^n$ with a circuit of size s and r coin flips, could be computed with the same probability via a sequence of linear operators on $(\mathbb{R}^2)^{\otimes n+r+s}$. Each linear operator was stochastic, so it took probability distributions to probability distributions, and acted on at most three registers via the action of one of the gates from the gate set used to construct the circuit. To get the output, after performing the linear operations, one throws away all but the first m entries of the output vector. The resulting vector encodes a non-normalized probability distribution, i.e., is of the form $|v\rangle = \sum_{|I|=p(n)} q_I|I\rangle$ with $q_I \geq 0$ and $\sum q_I \leq 1$. One then renormalizes, dividing each coefficient by $\sum q_I$, to obtain a vector $\sum_{|I|=m} p_I|I\rangle$ with $p_I \geq 0$ and $\sum p_I = 1$. Then the algorithm outputs I with probability p_I.

Here is a wish list for how one might want to improve upon this set-up:

(1) Allow more general kinds of linear maps to get more computing power, while keeping the maps easy to compute.

(2) Have reversible probabilistic computation: we saw that classical computatation can be made reversible, but the coin flip was not. This property is motivated by physics, where many physical theories require time reversibility.

(3) Also motivated by physics, one would like to have a continous evolution of the probability vector, more precisely, one would like the probability vector to depend on a continuous parameter t such that if $|\psi_{t_1}\rangle = X|\psi_{t_0}\rangle$, then there exist admissible matrices Y, Z such that $|\psi_{t_0+\frac{1}{2}t_1}\rangle = Y|\psi_{t_0}\rangle$ and $|\psi_{t_1}\rangle = Z|\psi_{t_0+\frac{1}{2}t_1}\rangle$ and $X = ZY$. A physicist would say "time evolution is described by a semi-group".

Let's start with wish (2). One way to make the coin flip reversible is, instead of making the probability distribution be determined by the sum of the coefficients, one could take the sum of the squares. That is, before we had $|v\rangle = \sum p_I|I\rangle$ with $p_I \geq 0$ and $\sum_I p_I = 1$ and the probability of getting output I was p_I. Instead we may take $\sum q_I|I\rangle$ where $\sum q_I^2 = 1$ and the probability of getting I is q_I^2. If we do this, there is no harm in allowing the entries of the output vectors to become negative, and one could use

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

for the coin flip applied to $|0\rangle$. The matrix H is called the *Hadamard matrix* or *Hadamard gate* in the quantum computing literature. It could just as well be called the *quantum coin flip*. The stochastic matrices (those that map the set of vectors with non-negative coefficients that sum to one to themselves), must be replaced with the matrices that map vectors of length one in the L_2-norm (the L_2-norm of $|v\rangle = \sum v_I|I\rangle$ is defined to be $|v| := \sqrt{\sum v_I^2}$), namely the *orthogonal matrices*: $O(n) = \{A \in Mat_{n\times n} \mid AA^{\mathbf{t}} = \mathrm{Id}\}$ With this change, we obtain our second wish as orthogonal matrices are invertible, they form a *group* (as $\mathrm{Id} \in O(n)$, $A \in O(n)$ implies $A^{-1} \in O(n)$ and $A, B \in O(n)$ implies $AB \in O(n)$). Moreover many operations are "continous". For example, rotation matrices are orthogonal and any rotation matrix has a square root.

Exercise 2.1.1: We have seen the matrix H without the $\frac{1}{\sqrt{2}}$ normalization before - where?

Remark 2.1.2. One indication that generalized probability may be related to quantum mechanics is that the interference patterns observed in the famous two slit experiments is manifested in generalized probability: We obtain a "random bit" by applying H to $|0\rangle$: $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. However, if we apply a second quantum coin flip to the vector, we loose the randomness

as $H^2 = \mathrm{Id}$, which, as pointed out in [**Aar13**], could be interpreted as a manifestation of interference.

Our third property will not be completely satisfied, as the matrix

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

which represents a reflection, does not have a square root in $O(2)$.

To have the third wish satisfied, we will allow ourselves vectors with *complex* entries. From now on, set $i = \sqrt{-1}$. For a complex number $z = x + iy$, let $\overline{z} = x - iy$ denote its complex conjugate and $|z|^2 = z\overline{z}$ the square of its norm. For a vector $|v\rangle = \sum z_I |I\rangle$, where $z_I \in \mathbb{C}$, defined the length of v to be $|v| := \sqrt{\sum_I |z_I|^2}$.

We now start with vectors $|v\rangle = \sum z_I |I\rangle$, where $z_I \in \mathbb{C}$ and $\sum \overline{z_I} z_I = 1$. The probability of obtaining I is now $|z_I|^2$. The set of admissible matrices is now the *unitary group*:

$$\mathbf{U}(n) := \{A \in Mat_{n\times n}(\mathbb{C}) \mid \ |Av| = |v| \ \forall |v\rangle \in \mathbb{C}^n\}.$$

Claim: $\mathbf{U}(n)$ satisfies the third wish on the list. More precisely:

Proposition 2.1.3. *For all $A \in \mathbf{U}(n)$, there exists a matrix $B \in \mathbf{U}(n)$ satisfying $B^2 = A$. In particular, fix any $\epsilon > 0$. Then any $A \in \mathbf{U}(n)$ may be written as a product of $k = k(\epsilon, A)$ unitary matrices $A = A_1 \cdots A_k$ with $|A_k - \mathrm{Id}\,| < \epsilon$ (where $|X| := \sqrt{\sum_{ij} |x_{ij}|^2}$).*

Proposition 2.1.3 is proved in §2.1.2 below.

Consider wish 1: it is an open question! However we can at least see that our generalized probabilistic computation includes our old probabilistic computation by the following easy exercise:

Exercise 2.1.4: Show that quantum coin flip H, the not ($\neg$) matrix and the Toffoli matrix (1.6.1) are unitary.

Moreover, we will see quantum algorithms that are better than any known probabilistic algorithms.

So we go from pbits, $\{p|0\rangle + q|1\rangle \mid \ p, q \geq 0 \text{ and } p + q = 1\}$ to *qubits*

$$\{\alpha|0\rangle + \beta|1\rangle \mid \alpha, \beta \in \mathbb{C} \text{ and } |\alpha|^2 + |\beta|^2 = 1\}.$$

The set of qubits, considered in terms of real parameters, looks at first like the 3-sphere S^3 in $\mathbb{R}^4 \simeq \mathbb{C}^2$. However, the probability distributions induced by $|\psi\rangle$ and $e^{i\theta}|\psi\rangle$ are the same so it is really S^3/S^1 (the Hopf fibration), i.e., the two-sphere S^2. Physicists call this S^2 the "Bloch sphere". Geometrically, it would be more natural (especially since we have already

seen the need to re-normalize in probalisitic computation) to work with projective space $\mathbb{CP}^1 \simeq S^2$ as our space of qubits, instead of a subset of $\mathbb{C}^2$. For $v = (v_1, \ldots, v_n) \in \mathbb{C}^n$, write $|v|^2 = |v_1|^2 + \cdots + |v_n|^2$. The norm induces a Hermitian inner product $\langle v|w\rangle := \overline{v_1}w_1 + \cdots + \overline{v_n}w_n$. Note the physicst convention (which I use in this book) is the reverse of the mathematician one because the product is conjugate linear in the first factor and linear in the second.

2.1.2. Exercises on the unitary group. The unitary group will play a central role in all that follows. For those readers not familiar with it, here are some basic exercises.

Exercise 2.1.5: Let $A \in \mathbf{U}(n))$ and let λ be an eigenvalue of A. Show $|\lambda| = 1$. In particular, we may write $\lambda = e^{i\theta}$ for some $\theta \in [0, 2\pi)$.

Exercise 2.1.6: Let $A \in \mathbf{U}(n))$ and let λ be an eigenvalue of A with eigenvector v. Set $v^\perp :=\}w \in \mathbf{U}(n) \mid \langle w, v\rangle = 0$. Show that $Av^\perp = v^\perp$.

Exercise 2.1.7: Notation as above. Show that A has a basis $v_1, \ldots, v_n$ of eigenvectors that are orthogonal to one another, i.e., $\langle v_i|v_j\rangle = \delta_{ij}$ for all i, j. ◎

Exercise 2.1.8: Show that a unitary matrix is diagonalizable.

Exercise 2.1.9: Show that if $A \in \mathbf{U}(n)$, then $\langle v|w\rangle = \langle Av|Aw\rangle$ for all $v, w \in \mathbb{C}^n$.

Exercise 2.1.10: Show that $\mathbf{U}(n) = \{A \in Mat_{n\times n}(\mathbb{C}) \mid \overline{A}^{\mathbf{t}}A = \mathrm{Id}\}$

Proof of Proposition 2.1.3. Let A be a unitary matrix and let $|v\rangle$ be an eigenvector for A with eigenvalue λ. Since $|v| = |Av| = |\lambda v| = |\lambda||v|$, we see $|\lambda| = 1$, i.e., $\lambda = e^{i\theta}$ for some $\theta \in \mathbb{R}$. Note that A must have a basis of eigenvectors, $|v_1\rangle, \ldots, |v_n\rangle$, as otherwise, let $|w\rangle$ be a putative generalized eigenvector, i.e., $A|w\rangle = \lambda|w\rangle + |u\rangle$. Since $|\lambda| = 1$, $|Aw| \neq |w|$.

Let $|v_1\rangle, \ldots, |v_n\rangle$ be an eigenbasis of $A \in \mathbf{U}(n)$ where $|v_j\rangle$ has eigenvalue $e^{i\theta_j}$. Let $B : \mathbb{C}^n \to \mathbb{C}^n$ be the matrix with the property that $B|v_j\rangle = e^{i\frac{\theta_j}{2}}|v_j\rangle$. Then B preserves the lengths of the eigenvectors, and thus of all vectors since the eigenvectors form an orthogonal basis by Exercise 2.1.11 below, and is therefore unitary, and clearly satisfies $B^2 = A$. □

Exercise 2.1.11: Show that if A is unitary, eigenvectors corresponding to distinct eigenvalues are orthogonal. ◎

2.2. Postulates of quantum mechanics and relevant linear algebra

Here are the standard postulates of quantum mechanics and relevant definitions from linear algebra.

2.2.1. Relevant definitions from linear algebra.

Definition 2.2.1. A *Hilbert space* $\mathcal{H}$ is a complex vector space equipped with a non-degenerate Hermitian inner-product, $h : \mathcal{H} \times \mathcal{H} \to \mathbb{C}$, where by definition h is conjugate linear in the first factor and linear in the second, $h(|v\rangle, |w\rangle) = \overline{h(|w\rangle, |v\rangle)}$, and $h(|v\rangle, |v\rangle) > 0$ for all $|v\rangle \neq 0$. (As mentioned above, this is the physicists' convention.)

Since our vector spaces will all be finite dimensional, one might ask why we need this general definition when our Hilbert spaces will just be $\mathbb{C}^n$ for some n? The reason is that our vector spaces will not always arise naturally as the space of column vectors with its standard bases. For example, spaces of linear maps and tensors will often have a natural Hilbert space structure, and if one writes these as column vectors, it is hard to keep track of the extra structure they have as linear maps or tensors.

The Hermitian inner-product h allows an identification of $\mathcal{H}$ with $\mathcal{H}^*$ by $|w\rangle \mapsto \langle w| := h(\cdot, |w\rangle)$. This identification will be used repeatedly. We write $h(|v\rangle, |w\rangle) = \langle v|w\rangle v$ and $|v| = \sqrt{\langle v|v\rangle}$ for the *length* of $|v\rangle$.

If $\mathcal{H} = \mathbb{C}^n$ with its standard basis, where $|v\rangle = (v_1, \ldots, v_n)^{\mathbf{t}}$, the *standard Hermitian innner-product* on $\mathbb{C}^n$ is $\langle w|v\rangle = \sum_{j=1}^n \overline{w}_j v_j$. We will always assume $\mathbb{C}^n$ is equipped with its standard Hermitian inner-product.

Remark 2.2.2. Generally in quantum mechanics one needs to deal with infinite dimensional Hilbert spaces, but fortunately this is not necessary in quantum computing and quantum information theory.

Definition 2.2.3. The *adjoint* of an operator $X \in \mathrm{End}(\mathcal{H})$, to be the operator $X^\dagger \in \mathrm{End}(\mathcal{H})$ such that $h(|X^\dagger v\rangle, |w\rangle) = h(|v\rangle, |Xw\rangle)$, i.e., $\langle X^\dagger v|w\rangle = \langle v|Xw\rangle$. Call X *Hermitian* if $X = X^\dagger$.

When $\mathcal{H} = \mathbb{C}^n$, so $\mathrm{End}(\mathcal{H})$ is the space of $n \times n$ matrices, then $X^\dagger = \overline{X}^{\mathbf{t}}$, where $\mathbf{t}$ denotes transpose.

Exercise 2.2.4: Show that the eigenvalues of a Hermitian matrix are real.

Remark 2.2.5. Physicists tend to use the letter H for the Hamiltonian, but since we already use H for the Hadamard matrix, I do not adopt this convention.

2.2.2. Postulate 1: State space. The first postulate describes the space one works in:

P1. Associated to any isolated physical system is a Hilbert space $\mathcal{H}$, called the *state space*. The system is completely described at a given moment by a unit vector $|\psi\rangle \in \mathcal{H}$, called its *state vector*, which is well defined up to a phase $e^{i\theta}$ with $\theta \in \mathbb{R}$. Alternatively one may work in projective space $\mathbb{P}\mathcal{H}$.

Remark 2.2.6. Note the first postulate is identical to what one gets with generalized probability.

Remark 2.2.7. This definition looks bad from Hardy's perspective. Later, in ***, we will see how to associate more than a $2n-2$-real-dimensional space of states to an n-dimensional Hilbert space. In fact, we will be able to associate an n^2-real-dimensional space of states using density operators. Moreover, the set of density operators will more closely resemble the set of probability distributions in that it will be a convex set.

2.2.3. Postulate 2: Evolution. The second postulate describes how a state vector evolves over time:

P2. The state of an isolated system evolves with time according to the *Schrödinger equation*

$$i\hbar\frac{d|\psi\rangle}{dt} = X|\psi\rangle$$

where $\hbar$ is a constant (*Planck's constant*) and X is a fixed Hermitian operator, called the *Hamiltonian* of the system.

Relation to generalized probability. Recall that in generalized probability theory, transformations are unitary. For a general Hilbert space, define the *Unitary group*

$$\mathbf{U}(\mathcal{H}) := \{U \in \mathrm{End}(\mathcal{H}) \mid |Uv| = |v| \ \forall |v\rangle \in \mathcal{H}\}.$$

When $\mathcal{H} = \mathbb{C}^n$ we have $\mathbf{U}(\mathbb{C}^n) = \mathbf{U}(n)$.

How do unitary operators from generalized probability lead to Schrödinger's equation? Recall that in generalized probability we are allowed to break up our action of an element $U \in \mathbf{U}(\mathcal{H})$ into a product of elements of $\mathbf{U}(\mathcal{H})$. More precisely, for each $\epsilon > 0$, there exists $k = k(\epsilon, U)$, such that $U = U_1 \cdots U_k$ with each U_j a distance at most ϵ from the identity. Similarly, we may find a curve from the identity to U in $\mathbf{U}(\mathcal{H})$.

Now say we have a smooth curve $U(t) \subset \mathbf{U}(\mathcal{H})$ with $U(0) = \mathrm{Id}$. Write $U'(0) = \frac{d}{dt}|_{t=0}U(t)$. Consider

$$\begin{aligned} 0 &= \frac{d}{dt}|_{t=0}\langle v|w\rangle \\ &= \frac{d}{dt}|_{t=0}\langle U(t)v|U(t)w\rangle \\ &= \langle U'(0)v|w\rangle + \langle v|U'(0)w\rangle. \end{aligned} \tag{2.2.1}$$

Remark 2.2.8. The trick of writing 0 as the derivative of a constant function is ubiquitous in differential geometry.

Thus $U'(0)$ behaves almost like a Hermitian operator, which instead satisfies $0 = \langle Xv|w\rangle - \langle v|Xw\rangle$.

Exercise 2.2.9: Show that $iU'(0)$ is Hermitian.

We are almost at Schrödinger's equation.

Let $\mathfrak{u}(\mathcal{H}) := \{X \in \mathrm{End}(\mathcal{H}) \mid \langle Xv|w\rangle + \langle v|Xw\rangle = 0 \forall v, w \in \mathcal{H}\}$. In other words, by (2.2.1), $\mathfrak{u}(\mathcal{H}) = T_{\mathrm{Id}}\mathbf{U}(\mathcal{H})$, the tangent space to the unitary group at the identity. The vector space $\mathfrak{u}(\mathcal{H})$ is called the *Lie algebra* of $\mathbf{U}(\mathcal{H})$. Note that $\mathfrak{u}(\mathcal{H})$ is a *real* vector space, not a complex one, because complex conjugation is not a complex linear map.

Thus $i\mathfrak{u}(\mathcal{H}) \subset \mathrm{End}(\mathcal{H})$ is the space of Hermitian endomorphisms.

Write $\mathfrak{u}(n) = \mathfrak{u}(\mathbb{C}^n)$.

Exercise 2.2.10: Verify independently that both $\mathfrak{u}(n)$ and the space of hermitian matrices have (real) dimension n^2.

For those familiar with dimensions of manifolds, Exercise 2.2.10 also shows $\dim \mathbf{U}(n) = n^2$.

For $X \in \mathrm{End}(\mathcal{H})$, write $X^k \in \mathrm{End}(\mathcal{H})$ for $X \cdots X$ applied k times. Write $e^X := \sum_{k=0}^{\infty} \frac{1}{k!}X^k$. This sum converges to a fixed matrix, essentially for the same reason it does in the $\dim \mathcal{H} = 1$ case.

Exercise 2.2.11: Show that the sum indeed converges, assuming the scalar case. ⊙

Proposition 2.2.12. *If X is Hermitian, then $e^{iX} \in \mathbf{U}(\mathcal{H})$.*

Exercise 2.2.13: Prove Proposition 2.2.12. ⊙

Postulate 2 implies the system will evolve unitarily, by (assuming we start at $t = 0$), $|\psi_t\rangle = U(t)|\psi_0\rangle$, where

$$U(t) = e^{\frac{-itX}{\hbar}}.$$

We conclude Postulate 2 is indeed predicted by generalized probability.

2.2.4. Postulate 3: measurements. In our first two postulates we dealt with isolated systems. In reality, no system is isolated and the whole universe is modeled by one enormous Hilbert space. In practice, parts of the system are sufficiently isolated that they can be treated as isolated systems. However, they are occasionally acted upon by the outside world, and we need a way to describe this outside interference. For our purposes, the isolated systems will be the Hilbert space attached to the input in a quantum algorithm and the outside interference will be the measurement at the end. That is, after a sequence of unitary operations one obtains a vector $|\psi\rangle = \sum z_j|j\rangle$ and as in generalized probability:

P3. If $|\psi\rangle = \sum_j z_j|j\rangle$, and a measurement is taken, the output is j with probability $|z_j|^2$.

2.2.5. Postulate 4: composite systems. A typical situation in quantum mechanics and quantum computing is that there are two or more isolated systems, say $\mathcal{H}_A, \mathcal{H}_B$ that are brought together (i.e., allowed to interact with each other) to form a larger isolated system $\mathcal{H}_{AB}$. The larger system is called the *composite system*. In classical probability, the composite space is $\{0,1\}^{N_A} \times \{0,1\}^{N_B}$. We have already seen in our generalized probability, the correct composite space is $(\mathbb{C}^2)^{\otimes N_A} \otimes (\mathbb{C}^2)^{\otimes N_B} = (\mathbb{C}^2)^{\otimes N_A + N_B}$ (Exercise 1.6.3).

P4. The state of a composite system $\mathcal{H}_{AB}$ is the tensor product of the state spaces of the component physical systems $\mathcal{H}_A, \mathcal{H}_B$: $\mathcal{H}_{AB} = \mathcal{H}_A \otimes \mathcal{H}_B$.

When dealing with composite systems, we will allow partial measurements whose outcomes are of the form $|I\rangle \otimes \phi$.

This tensor product structure gives rise to the notion of *entanglement*, which, in the next few sections, we will see accounts for phenomenon outside of our classical intuition.

Definition 2.2.14. A state $|\psi\rangle \in \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_n$ is called *separable* if it corresponds to a rank one tensor, i.e., $|\psi\rangle = |v_1\rangle \otimes \cdots \otimes |v_n\rangle$ with each $|v_j\rangle \in \mathcal{H}_j$. Otherwise it is *entangled*.

Remark 2.2.15. Later, in §**??**, we will discuss how to *derive* the postulates of quantum mechanics as a natural generalization of probability, by just insisting on having operations arbitrarily close to the identity.

2.2.6. Further Exercises. For $X, Y \in \mathrm{End}(\mathcal{H})$, let $[X,Y] := XY - YX \in \mathrm{End}(\mathcal{H})$ denote their commutator.

Exercise 2.2.16: For $X, Y \in \mathfrak{u}(\mathcal{H})$ show that $[X,Y] \in \mathfrak{u}(\mathcal{H})$, showing that $\mathfrak{u}(\mathcal{H})$ is indeed an algebra with the multiplication given by the commutator.

Exercise 2.2.17: Show that if $\mathcal{H} = \mathbb{C}^n$, then $X^\dagger = \overline{X}^{\mathbf{t}}$, where the $\mathbf{t}$ denotes transpose.

Exercise 2.2.18: Show that if $Y \in \mathrm{End}(\mathcal{H})$ is arbitrary, then $YY^\dagger$ and $Y^\dagger Y$ are Hermitian.

Exercise 2.2.19: Show that the eigenvalues of a Hermitian operator are real.

Exercise 2.2.20: Prove the *spectral decomposition theorem* for Hermitian operators: Hermitian operators are diagonalizable and the eigenspaces of a Hermitian operator M are orthogonal. In particular we may write $M = \sum_\lambda \lambda P_\lambda$ where λ are the eigenvalues of M and the P_λ are commuting projection operators: $P_\lambda P_\mu = P_\mu P_\lambda$ and $P_\lambda^2 = P_\lambda$. ⊚

Exercise 2.2.21: Show that

$$\mathbf{U}(\mathcal{H}) = \{U \in \mathrm{End}(\mathcal{H}) \mid \langle Uv|Uw\rangle = \langle v|w\rangle \ \forall |v\rangle, |w\rangle \in \mathcal{H}\},$$

and that if $U \in \mathbf{U}(\mathcal{H})$, then $U^{-1} = U^\dagger$.

Exercise 2.2.22: Show that $\mathbf{U}(2)$ acts transitively on lines in $\mathbb{C}^2$, i.e, given any nonzero $v, w \in \mathbb{C}^2$ there exists $U \in \mathbf{U}(2)$ such that $U|v\rangle = \lambda|w\rangle$ for some $\lambda \in \mathbb{C}^*$. ⊚

A *reflection* in a hyperplane $\mathbb{C}^{n-1} \subset \mathbb{C}^n$ is the linear map that, writing $|v\rangle \in \mathbb{C}^n$ as $|v\rangle = |v_1\rangle + |v_2\rangle$ with $|v_1\rangle \in \mathbb{C}^{n-1}$ and $|v_2\rangle \perp \mathbb{C}^{n-1}$, sends $|v\rangle \mapsto |v_1\rangle - |v_2\rangle$.

Exercise 2.2.23: Show that $\mathbf{U}(n)$ contains the reflections.

Exercise 2.2.24: Show that the product of two reflections is a rotation. More precisely, show that if $|v\rangle, |w\rangle$ are vectors in $\mathbb{C}^n$, the composition of a reflection in the hyperplane perpendicular to $|v\rangle$, followed by a reflection in the hyperplane perpendicular to $|w\rangle$, is a rotation in the $|v\rangle, |w\rangle$ plane by an angle equal to twice the angle between $|v\rangle$ and $|w\rangle$ (and the identity elsewhere).

2.3. Super-dense coding

In this section, we show that with a shared entangled state one can transmit two bits of classical information by transmitting a vector in just one qubit, which has led to the term "super[1]-dense coding". Super-dense coding was introduced in [**BW92**].

Physcists describe their experiments in terms of two characters, Alice and Bob. We generally follow this convention.

[1]Physicists use the word "super" in the same way American teenagers use the word "like".

Let $\mathcal{H} = \mathbb{C}^2 \otimes \mathbb{C}^2 = \mathcal{H}_A \otimes \mathcal{H}_B$, and let $|epr\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$ (called the *EPR state* in the physics literature, named after Einstein-Podosky-Rosen) Assume this state has been created, both Alice and Bob are aware of it, Alice is in possesion of (i.e., can manipulate) the first qubit, and Bob the second. This all happens before the experiment begins. They are allowed to agree on a protocol in advance. Then they are separated, but have a "quantum channel" along which they can transmit qubits. (Such will be explained in ***.)

Now say Alice wants to transmit a two classical bit message to Bob, i.e., one of $00, 01, 10, 11$. She is allowed to act on her half of $|epr\rangle$ by unitary transformations and then send it to Bob. (Later we will establish a gate set she must choose from, but it will include the gates we need below.) They agree in advance that once Bob is in possesion of it, he will act on the four-dimensional space $\mathcal{H}_A \otimes \mathcal{H}_B$ by the unitary operator that performs the following change of basis:

$$\begin{aligned} |epr\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) &\mapsto |00\rangle \\ \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) &\mapsto |01\rangle \\ \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle) &\mapsto |10\rangle \\ \frac{1}{\sqrt{2}}(|10\rangle - |01\rangle) &\mapsto |11\rangle \end{aligned}$$

and then will measure.

If Alice wants to send 00, she just does nothing as then when Bob measures he will get $|00\rangle$ with probability one. Similarly, if she wants to send 01, she acts by

$$\sigma_x := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

so Bob will be in possesion of the state $\frac{1}{\sqrt{2}}|00\rangle - |11\rangle$, so when he performs the change of basis and measures, he will get $|01\rangle$ with probability one.

Exercise 2.3.1: What are the other two matrices Alice should act by to transmit the other two-bit messages?

In summary, with preparation of an EPR state in advance, plus transmission of a single qubit, one can transmit two classical bits of information.

2.4. Quantum Teleportation

A similar phenomenon is *quantum teleportation*, where again Alice and Bob share half of an EPR state. This time Alice is in possesion of a qubit

$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, and wants to "send" $|\psi\rangle$ to Bob. However Alice only has access to a classical channel that sends bits to Bob. Can she transmit $|\psi\rangle$ to Bob, and if so, how many classical bits does she need to transmit to do so? Write the state of the system as

$$\frac{1}{\sqrt{2}}\left[\alpha|0\rangle\otimes(|00\rangle + |11\rangle) + \beta|1\rangle\otimes(|00\rangle + |11\rangle)\right]$$

where Alice can operate on the first two qubits.

Exercise 2.4.1: Show that if Alice acts on the first two qubits by $\mathrm{Id}_1 \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ then $H\otimes \mathrm{Id}_2 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \mathrm{Id}_2$ (where the subscripts on Id indicate which factor the identity acts). She obtains

$$\frac{1}{2}\left[|00\rangle\otimes(\alpha|0\rangle + \beta|1\rangle) + |01\rangle\otimes(\alpha|1\rangle + \beta|0\rangle) + |10\rangle\otimes(\alpha|0\rangle - \beta|1\rangle) + |11\rangle\otimes(\alpha|1\rangle - \beta|0\rangle)\right].$$

Notice that Bob's coefficient of Alice's $|00\rangle$ is the state ψ that is to be transmitted. Alice performs a measurement. If she has the good luck to obtain $|00\rangle$, then she knows Bob has $|\psi\rangle$ and she can tell him classically that he is in possesion of $|\psi\rangle$. But say she obtains the state $|01\rangle$: the situation is still good, she knows Bob is in possession of a state such that, if he acts on it with $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, he will obtain the state $|\psi\rangle$, so she just needs to tell him classically to apply σ_x. Since they had communicated the algorithm in the past, all Alice really needs to tell Bob in the first case is the classical message 00 and in the second case the message 01.

Exercise 2.4.2: Write out the other two different actions Bob should take depending on the possible bit pairs Alice could send him.

In summary, a shared EPR pair plus sending two classical bits of information allows one to transmit one qubit.

Remark 2.4.3. The name "teleportation" is misleading because information is transmitted at a speed slower than the speed of light.

2.5. Bell's game

The 1934 Einstein-Podosky-Rosen paper [**EPR35**] challenged quantum mechanics with the following thought experiment that they believed implied instaneous communication across distances, in violation of principles of relativity: Alice and Bob prepare $|epr\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, then travel far apart. Alice measures her bit. If she gets 0, then she can predict with certainty that Bob will get 0 in his measurement, even if his measurement is taken a

second later and they are a light year apart. (The essential property of the state is that Alice's measurement makes Bob's state classical as well.)

Ironically, this thought experiment has been made into an actual experiment designed by Bell [**Bel64**] and realized. The modern interpretation is that there is no paradox because the system does not transmit information faster than the speed of light, but rather they are acting on information that has already been shared. What follows is a version from [**CHSH69**], adapted from the presentation in [**AB09**].

To show that neither classical nor probabilistic reasoning can account for the experimental results, they are often described as a game: Alice and Bob are on the same team and Charlie is a referee. Charlie chooses $x, y \in \{0, 1\}$ at random and sends x to Alice and y to Bob. Based on this information, Alice and Bob, without communicating with each other, get to choose bits a, b and send them to Charlie. They win if $a \oplus b = x \wedge y$, i.e., either $(x, y) \neq (1, 1)$ and $a = b$ or $(x, y) = (1, 1)$ and $a \neq b$.

2.5.1. Classical version. Note that if Alice and Bob both always choose 0, they win with probability $\frac{3}{4}$.

Theorem 2.5.1. [**Bel64**] *Regardless of the classical or probabilistic strategy Alice and Bob use, they never win with probability greater than* $\frac{3}{4}$.

The idea of the proof is that one first reduces a probabilisitic strategy to a classical one, because after repeated rounds of the game, one can just adopt the most frequent choice for each possible x, y. Then there are only 2^4 possible strategies and each can be analyzed. See, e.g., [**AB09**, Thm 20.2] for more detail.

2.5.2. Quantum version. Alice and Bob prepare $|epr\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$ in advace, and Alice takes the first qubit and Bob the second. When Alice gets x from Charlie, if $x = 1$, she applies a rotation by $\frac{\pi}{8}$ to her quibit, and if $x = 0$ she does nothing. When Bob gets y from Charlie, he applies a rotation by $-\frac{\pi}{8}$ to his qubit if $y = 1$ and if $y = 0$ he does nothing. (The order these rotations are applied does not matter because the operators on $(\mathbb{C}^2)^{\otimes 2}$ commute.) Both of them measure their respective qubits (again, the order will not matter) and send the values obtained to Charlie.

Theorem 2.5.2. *With this strategy, Alice and Bob win with probability at least* $\frac{4}{5}$.

Proof. If $(x, y) = (0, 0)$, then they are measuring $|epr\rangle$, so the measurement either yields 0 for both or 1 for both and $0 \oplus 0 = 1 \oplus 1 = 0 = 0 \wedge 0$, so they always win in this case.

If $(x, y) = (1, 0)$, then they are measuring

$$\frac{1}{\sqrt{2}}\left(\cos(\frac{\pi}{8})|00\rangle + \sin(\frac{\pi}{8})|10\rangle - \sin(\frac{\pi}{8})|01\rangle + \cos(\frac{\pi}{8})|11\rangle\right),$$

and the outputs are equal with probability $(\frac{1}{2} + \frac{1}{2})\cos^2(\frac{\pi}{8}) \geq \frac{17}{20}$, and similarly if $(x, y) = (0, 1)$.

If $(x, y) = (1, 1)$, then they are measuring

$$\begin{aligned}
&\frac{1}{\sqrt{2}}[\cos(\frac{\pi}{8})(\cos(-\frac{\pi}{8})|00\rangle + \sin(-\frac{\pi}{8})|01\rangle) + \cos(\frac{\pi}{8})(-\sin(-\frac{\pi}{8})|00\rangle + \cos(-\frac{\pi}{8})|01\rangle)\\
&\quad + \sin(\frac{\pi}{8})(\cos(-\frac{\pi}{8})|10\rangle + \sin(-\frac{\pi}{8})|11\rangle) + \sin(\frac{\pi}{8})(-\sin(-\frac{\pi}{8})|10\rangle + \cos(-\frac{\pi}{8})|11\rangle)\\
&- \sin(\frac{\pi}{8})(\cos(-\frac{\pi}{8})|00\rangle + \sin(-\frac{\pi}{8})|01\rangle) - \sin(\frac{\pi}{8})(-\sin(-\frac{\pi}{8})|00\rangle + \cos(-\frac{\pi}{8})|01\rangle)\\
&\quad + \cos(\frac{\pi}{8})(\cos(-\frac{\pi}{8})|10\rangle + \sin(-\frac{\pi}{8})|11\rangle) + \cos(\frac{\pi}{8})(-\sin(-\frac{\pi}{8})|10\rangle + \cos(-\frac{\pi}{8})|11\rangle)]\\
&\quad = \frac{1}{2}[|00\rangle + |01\rangle + |10\rangle + |11\rangle],
\end{aligned}$$

so they win with probability $\frac{1}{2}$, as all coefficients have the same norm.

In sum, the overall chance of winning is at least $\frac{1}{4}(1) + \frac{1}{4}(\frac{17}{20}) + \frac{1}{4}(\frac{17}{20}) + \frac{1}{4}(\frac{1}{2}) = \frac{4}{5}$. □

Exercise 2.5.3: Show that this strategy can be improved. What is its limit? ◎

Chapter 3

Algorithms

This chapter covers the basics of quantum computing, and the standard quantum algorithms. We begin with a probabilistic algorithm, the Miller-Rabin primality test, as ideas from its proof appear in Shor's algorithm. We next present the algorithms of Grover and Simons. We then discuss admissible quantum gates, and then, after considerable preliminaries, present Shor's algorithm. For those not familiar with basic facts regarding groups and rings, I suggest starting with the Appendix §3.8.

3.1. Primality testing

In this section we discuss the problem: Given an integer N, determine if N is prime. Of course one could just test if any of the numbers $2, \ldots, \lfloor\sqrt{N}\rfloor$ divide N, but this is not efficient to compute (it is not polynomial time). Remarkably, there was no efficient algorithm known until the 1970's when an efficient probabilistic algorithm was developed.

Recall what I mean by this: an algorithm such that, given input an integer N and access to randomness, correctly outputs whether N is composite or prime with probability $> \frac{1}{2}$.

I present the famous *Miller-Rabin test* [**Rab80**]. and its proof because parts of the proof will be used for Shor's algorithm.

Let $\mathbb{Z}/N\mathbb{Z}$ denote the ring of integers mod N. Write $m \bmod N$ for the equivalence class of m. Just as with the problem of multiplying polynomials, where we arrived at an efficient computation working in a quotient of the polynomial ring, here we will obtain our efficient algorithm by working in a quotient of the ring of integers.

Recall from Exercise 1.2.1 that the naïve algorithm for computing a^k uses $O(a^k)$ operations, and one can do a little better using Karatsuba's algorithm (Exercise 1.2.2). The key to the efficiency is the following exercise:

Exercise 3.1.1: Prove that for $k \in \{0, \ldots, N-1\}$, $a^k \bmod N$ can be computed by a classical circuit of size $O(N^3)$ (where bit addition and bit multiplication have unit cost). Show that with a little more work, one can get $O(N^2 \log(N))$. What about if one uses Shönhage-Strassen (§1.2.3)? ⊙

The *Chinese remainder theorem* asserts that, for primes p, q, there is a ring isomorphism $\mathbb{Z}/pq\mathbb{Z} \simeq \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$.

Exercise 3.1.2: Verify the map $m \bmod pq \mapsto (m \bmod p, m \bmod q)$ defines a ring isomorphism.

More generally if $N = p_1^{a_1} \cdots p_k^{a_k}$ with p_j distinct primes, there is a ring isomorphism $(\mathbb{Z}/N\mathbb{Z}) = (\mathbb{Z}/p_1^{a_1}\mathbb{Z}) \times \cdots \times (\mathbb{Z}/p_k^{a_k}\mathbb{Z})$. For a ring R, let R^* denote its invertible elements under multiplication, which form a group. We also have $(\mathbb{Z}/N\mathbb{Z})^* = (\mathbb{Z}/p_1^{a_1}\mathbb{Z})^* \times \cdots \times (\mathbb{Z}/p_k^{a_k}\mathbb{Z})^*$.

Here is a warm-up: an efficient test to see if N is prime, that works for "most" N.

Recall that if p is prime, then the multiplicative group $(\mathbb{Z}/p\mathbb{Z})^*$ is a cyclic group of order $p-1$. As a consequence, if $x \not\equiv 0 \bmod p$ then $x^{p-1} \equiv 1 = x^0 \bmod p$ (the little Fermat theorem). In other words, if we find x such that $x^{N-1} \not\equiv 1 \bmod N$, then we know N is composite.

Exercise 3.1.3: Show that if p is prime, then for all integers m, n, $(m+n)^p \equiv m^p + n^p \bmod p$.

Exercise 3.1.4: Prove the little Fermat theorem. ⊙

Call the following probabilistic algorithm the *Fermat test*: Choose a uniformly at random from $\{2, \ldots, N-1\}$ and compute $a^{N-1} \bmod N$. It will be clear that for this and the algorithm that follows, the tests will always report that N is prime when it is prime, so say N is composite. Under what circumstances do we correctly determine compositeness with probability at least $\frac{1}{2}$? Consider the following two cases:

(1) $\gcd(a, N) = d \neq 1$. (This occurs with low probability.) Then the test detects that N is composite as in this situation $a \equiv 0 \bmod d$, and hence $a^{N-1} \not\equiv 1 \bmod N$.

(2) $\gcd(a, N) = 1$, so $a \in (\mathbb{Z}/N\mathbb{Z})^*$.

We need to determine what happens in the second case.

Lemma 3.1.5. *If there exists $a \in (\mathbb{Z}/N\mathbb{Z})^*$, such that $a^{N-1} \not\equiv 1 \bmod N$, then the Fermat test detects the compositeness of N with probability $\geq \frac{1}{2}$.*

Before giving the proof, introduce the following groups associated to an abelian group G: For any natural number m, consider the group homomorphism $\phi_m : G \to G$, $\phi_m(x) = x^m$, and let

$$G^{(m)} = \text{Image}\,\phi_m \text{ and } G_{(m)} = \ker \phi_m, \tag{3.1.1}$$

both of which are abelian groups. Note that if a is uniformly distributed over an abelian group G then a^m is uniformly distributed over $G^{(m)}$ defined in (3.1.1).

In this language, the a's for which the Fermat test fails are those in $(\mathbb{Z}/N\mathbb{Z})^*_{(N-1)}$.

Proof. The hypothesis is that $(\mathbb{Z}/N\mathbb{Z})^*_{(N-1)} \neq (\mathbb{Z}/N\mathbb{Z})^*$ and we need to show most elements of $(\mathbb{Z}/N\mathbb{Z})^*_{(N-1)}$ are not in $(\mathbb{Z}/N\mathbb{Z})^*$. Since $N > 3$, the quotient $(\mathbb{Z}/N\mathbb{Z})^*/(\mathbb{Z}/N\mathbb{Z})^*_{(N-1)}$ has cardinality at least 2. Thus $a^{N-1} \not\equiv 1 \bmod N$ for at least half of the elements of $(\mathbb{Z}/N\mathbb{Z})^*$ and we conclude. □

It is possible that $a^{N-1} \equiv 1 \bmod N$ for all $a \in (\mathbb{Z}/N\mathbb{Z})^*$. So we will need an additional test to apply when the Fermat test fails to get our desired algorithm.

Exercise 3.1.6: Show that $N = 561 = 3 * 11 * 17$ is such that $a^{N-1} \equiv 1 \bmod N$ for all $a \in (\mathbb{Z}/N\mathbb{Z})^*$.

The second test uses the following proposition:

Proposition 3.1.7. *If there exists a natural number b such that $b^2 \equiv 1 \bmod N$ and $b \not\equiv \pm 1 \bmod N$, then N is composite with nontrivial factors in common with both $b+1$ and $b-1$.*

Proof. In this case $b^2 - 1 = (b-1)(b+1)$ is a multiple of N but $b-1, b+1$ are not, so N must have nontrivial factors in common with both $b+1$ and $b-1$. □

Here is the Miller-Rabin algorithm: to avoid trivialities, assume N is odd.

Choose $a \in \{2, \ldots, N-2\}$ uniformly at random.

Step 1: Test if $a^{N-1} \not\equiv 1 \bmod N$. If so, then N is composite by the Little Fermat theorem and one concludes. Otherwise go to step 2:

Step 2: Let 2^k be the largest power of 2 that divides $N-1$ and write $N - 1 = 2^k l$. Compute the sequence $a^l, a^{2l}, a^{4l}, \ldots, a^{2^k l}$, all $\bmod N$. If this sequence contains a 1 preceded by anything except ± 1, i.e., if there exists j such that $a^{2^j l} \not\equiv \pm 1 \bmod N$ and $(a^{2^j l})^2 \equiv 1 \bmod N$, then N is composite by Proposition 3.1.7. Otherwise the algorithm replies "N is prime".

One can check that the total circuit size of this algorithm is $O(\log(N)^3)$. The only subtlety is that taking exponentially many powers of a would violate this size, but we are only taking powers mod N.

Proposition 3.1.8. *The Miller-Rabin algorithm succeeds on any input with probability at least* $\frac{1}{2}$.

Proof. It is clear that if N is prime, the algorithm always indicates that it is prime, so assume N is composite and odd. Start the algorithm, get some $a \in \{2, \ldots, N-2\}$ chosen uniformly at random. If $\gcd(a, N) > 1$, then step 1 shows that N is composite, so assume this does not happen, which implies a is uniformly distributed over $(\mathbb{Z}/N\mathbb{Z})^*$. (This last assertion holds because for any group homomorphism of finite groups $f : G \to H$, all fibers have the same cardinality. We will use this repeatedly in what follows.)

In order for step 1 to work with probability at least $\frac{1}{2}$, it is enough that there is one $x \in \{2, \ldots, N-1\}$ such that $x^{N-1} \not\equiv 1 \bmod N$. The following exercise shows that if N is a power of an odd prime, then there is such an x.

Exercise 3.1.9: Show that if $N = p^c$ for some odd prime p, and some $c \geq 2$, then taking $a = p^c + 1 - p^{c-1}$, then $a^{N-1} \not\equiv 1 \bmod N$ because $a^{N-1} \equiv p^{c-1} + 1 \bmod N$ and $p^{c-1} + 1 \not\equiv 1 \bmod N$.

By Exercise 3.1.9, we may assume $N = uv$, where u, v are odd, $u, v > 1$, and $\gcd(u, v) = 1$. By the Chinese remainder theorem $(\mathbb{Z}/N\mathbb{Z})^* \simeq (\mathbb{Z}/u\mathbb{Z})^* \times (\mathbb{Z}/v\mathbb{Z})^*$.

Now $(\mathbb{Z}/N\mathbb{Z})^{*(m)} \simeq (\mathbb{Z}/u\mathbb{Z})^{*(m)} \times (\mathbb{Z}/v\mathbb{Z})^{*(m)}$. If either $(\mathbb{Z}/u\mathbb{Z})^{*(N-1)}$ or $(\mathbb{Z}/v\mathbb{Z})^{*(N-1)}$ is non-trivial, step 1 will detect compositeness with probability at least $\frac{1}{2}$, so assume both are trivial. To apply step 2, we need to consider the powers $a^{2^j l} \bmod N$ and show there exists j such that $a^{2^j l} \not\equiv \pm 1 \bmod N$ and $(a^{2^j l})^2 = a^{2^{j+1} l} \equiv 1 \bmod N$ with probability at least $\frac{1}{2}$.

Let j_0 be the largest value such that $(\mathbb{Z}/N\mathbb{Z})^{*(2^{j_0} l)} \neq \{1\}$ and $(\mathbb{Z}/N\mathbb{Z})^{*(2^{j_0+1} l)} = \{1\}$. Use the isomorphism $(\mathbb{Z}/N\mathbb{Z})^{*(2^{j_0} l)} \simeq (\mathbb{Z}/u\mathbb{Z})^{*(2^{j_0} l)} \times (\mathbb{Z}/v\mathbb{Z})^{*(2^{j_0} l)}$: both the factors cannot be trivial by assumption. If one of the two factors is trivial, we could only fail if $a^{2^{j_0} l}$ maps to $(1, 1)$, but this will happen for the nontrivial factor with probability at most $\frac{1}{2}$. Now assume both factors are nontrivial, say of cardinalities c_u, c_v. In this case, the image of $a^{2^{j_0} l}$ in the first factor is 1 with probability $\frac{1}{c_u}$, and is 1 in the second factor with probability $\frac{1}{c_v}$, and these events are independent (again by the Chinese remainder theorem). Thus the probability $a^{2^{j_0} l} \equiv 1 \bmod N$ is $\frac{1}{c_u c_v}$. For similar reasons the the probability $a^{2^{j_0} l} \equiv -1 \bmod N$ is either $\frac{1}{c_u c_v}$ or zero. Thus the probability of failure is at most $\frac{2}{c_u c_v} \leq \frac{1}{2}$. $\square$

To get an algorithm that works with probability greater than $\frac{1}{2}$, apply the test twice.

But this is not the end of the story:

Theorem 3.1.10. [**AKS04**] *Primality testing is in* **P**.

The core of the proof is a variant of the little Fermat theorem: Let a, N be relatively prime integers with $N > 2$, Then N is prime if and only if $(x+a)^N \equiv x^N + a \bmod N$. Here the identity is to be interpreted in the ring of polynomials with coefficients in $\mathbb{Z}/N\mathbb{Z}$.

Exercise 3.1.11: Prove the assertion. ◎

The bulk of the work is reducing the number of coefficients one needs to check in the expansion of the left hand side.

The lesson to be drawn here is that we should not make any assumptions regarding the difficulty of a problem until we have a proof.

3.2. Grover's search algorithm

The problem: given $F_n : \mathbb{F}_2^n \to \mathbb{F}_2$, computable by a $poly(n)$-size classical circuit, find a such that $F_n(a) = 1$ if such a exists.

Compare this with a brute force search, which requires a circuit of size $poly(n)2^n$. No classical or probabilistic algorithm is known that does better than $poly(n)2^n$. Note that it also gives a size $poly(n)2^{\frac{n}{2}}$ probabilistic solution to the **NP**-complete problem SAT (it is stronger, as it not only determines existence of a solution, but finds it).

Grover found a quantum circuit of size $poly(n)2^{\frac{n}{2}}$ that solves this problem (with high probability).

We will present the algorithm for the following simplified version where one is promised there exists exactly one solution. All essential ideas of the general case are here.

Problem: given $F_n : \mathbb{F}_2^n \to \mathbb{F}_2$, computable by a $poly(n)$-size classical circuit, and the information that F has exactly one solution a, find a.

The idea of the algorithm is to start with a vector equidistant from all possible solutions, and then to incrementally rotate it towards a. What is strange for our classical intuition is that we will be able to rotate towards the solution without knowing what it is, and similarly, we won't "see" the rotation matrix either.

We work in $(\mathbb{C}^2)^{\otimes n+1+s}$ where $s = s(n)$ is the size of the classical circuit needed to compute F_n. We suppress reference to the s "workspace bits" in what follows.

The first step is to construct such a starting vector:

The following vector is the average of all the classical (observable) states:

$$|av\rangle := \frac{1}{2^{\frac{n}{2}}} \sum_{I\in\{0,1\}^n} |I\rangle. \tag{3.2.1}$$

To prepare $|av\rangle$, note that $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, so applying $H^{\otimes n}$ to $|0\cdots 0\rangle$ transforms it to $|av\rangle$.

The cost of this is n gates, as $H^{\otimes n}$ is the composition of $H\otimes \mathrm{Id}_{2,\ldots,n}$, $\mathrm{Id}_1 \otimes H \otimes \mathrm{Id}_{3,\ldots,n}$, ... , $\mathrm{Id}_{1,\ldots,n-1}\otimes H$.

Since $|av\rangle$ is equidistant from all possible solution vectors, we have $\langle av|a\rangle = \frac{1}{2^{\frac{n}{2}}}$. We want to rotate $|av\rangle$ towards the unknown a. Recall that $\cos(\angle(|v\rangle,|w\rangle)) = \frac{\langle v|w\rangle}{|v||w|}$. Write the angle between av and a as $\frac{\pi}{2}-\theta$, so $\sin(\theta) = \frac{1}{2^{\frac{n}{2}}}$.

Recall from Exercise 2.2.24, that a rotation is a product of two reflections. In order to perform the rotation R that moves $|av\rangle$ towards $|a\rangle$, we first reflect across the hyperplane orthogonal to $|a\rangle$ (i.e., we send $|a\rangle$ to $-|a\rangle$ and the hyperplane perpendicular to it is fixed), and then across the hyperplane orthogonal to $|av\rangle$.

Consider the map

$$|xy\rangle \mapsto |x(y\oplus F(x))\rangle \tag{3.2.2}$$

defined on basis vectors and extended linearly. To execute this, we use the s workspace bits corresponding to y to effect s reversible classical gates. We initially set $y=0$ so that the image is $|x0\rangle$ for $x\neq a$, and $|x1\rangle$ when $x=a$. At this point our vector is

$$\frac{1}{2^{\frac{n}{2}}}\left(\sum_{I\neq a}|I\rangle\otimes|0\rangle + |a\rangle\otimes|1\rangle\right).$$

Next apply the quantum gate $\mathrm{Id}\otimes\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$ which sends $|x0\rangle\mapsto|x0\rangle$, and $|x1\rangle\mapsto -|x1\rangle$. Finally apply the map $|xy\rangle\mapsto|x(y\oplus F(x))\rangle$ again.

Thus $|a0\rangle\mapsto -|a0\rangle$ and all other basis vectors $|b0\rangle$ are mapped to themselves, which is what we desired. That is, we now have the vector

$$\frac{1}{2^{\frac{n}{2}}}\left(\sum_{I\neq a}|I\rangle\otimes|0\rangle - |a\rangle\otimes|0\rangle\right).$$

Next we need to reflect around $|av\rangle$. It is easy to reflect around a classical state, so first perform the map $H^{\otimes n}$ that sends $|av\rangle$ to $|0\cdots 0\rangle$ (recall that $H = H^{-1}$), then reflect in the hyperplane perpendicular to $|0\cdots 0\rangle$ using the Boolean function $g:\mathbb{F}_2^n\to\mathbb{F}_2$ that outputs 1 if and only if its input is

$(0,\dots,0)$, in the role of F for our previous reflection, then apply Hadamard again so the resulting reflection is about $|av\rangle$.

The composition of these two reflections is the desired R.

Exercise 3.2.1: Approximately determine the probability that a measurement of $R|av\rangle$ will produce $|a\rangle$. (Use the approximation that for θ small, $\sin(\theta) \simeq \theta$.) ⊚

As mentioned above, the vector $R|av\rangle$ is not useful, but if we instead compose this map with itself $O(\frac{1}{\theta})$ times, we obtain a vector much closer to $|a\rangle$.

Another way to keep track of the maps is to note the first reflection is $\mathrm{Id} -2|a\rangle\langle a|$ and the second is $\mathrm{Id} -2|av\rangle\langle av|$, so, e.g., the rotation is

$$\begin{aligned}
&\mathrm{Id} -2|a\rangle\langle a| - 2|av\rangle\langle av| + 4|av\rangle\langle av||a\rangle\langle a| \\
&= \mathrm{Id} -2|a\rangle\langle a| - 2|av\rangle\langle av| + 4\langle av|a\rangle|av\rangle\langle a| \\
&= \mathrm{Id} -2|a\rangle\langle a| - 2|av\rangle\langle av| + \frac{4}{2^{\frac{n}{2}}}|av\rangle\langle a|.
\end{aligned}$$

Exercise 3.2.2: Using the above expression, exactly determine the probability that a measurement of $R|av\rangle$ will produce $|a\rangle$.

Exercise 3.2.3: Show that applying the procedure $2^{\frac{n}{2}}$ times, one obtains a vector such that the probability of it being in state $|a\rangle$ after a measurement is close to 1.

3.3. Simons' algorithm

The problem: given $f = f_n : \mathbb{F}_2^n \to \mathbb{F}_2^n$, computable by a Boolean circuit of size polynomial in n, such that there exists $a \in \mathbb{F}_2^n$ satisfying for all $x, y \in \mathbb{F}_2^n$, $f(x) = f(y)$ if and only if $x = y \oplus a$, find a. For simplicity of exposition, assume we know $a \neq (0,\dots,0)$ as well.

Simons gives a $poly(n)$ size quantum circuit that obtains the solution.

Remark 3.3.1. Although this problem may look unnatural, the resulting algorithm inspired Shor's algorithm and its generalizations, and it fits into a larger framework of problems that allow for an exponential quantum speedup over known probabilistic algorithms.

Remark 3.3.2. This problem is expected to be hard on a classical computer. Consider the following variant where F is allowed to be difficult to compute, but we are handed a black box that will compute it for us at unit cost. If a and F are chosen at random subject to the condition that $F(x) = F(y)$ if and only if $x = y \oplus a$, then classically one would need to use the black box $2^{\frac{n}{2}}$ times before having any information at all, as with fewer

calls, it is likely that one never gets the same answer twice. On the other hand, Simons' algorithm still gives a $poly(n)$-size solution in this setting.

Work in $(\mathbb{C}^2)^{\otimes 2n+s}$, where s is the size of a reversible Boolean circuit needed to compute F. We suppress reference to the s workspace qubits in what follows. As with Grover's algorithm, we will construct a vector that "sees" the answer a, but we will not be able to see the vector, so instead we manipulate it to get information about the solution. Also as before, first prepare $|av\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{x\in\{0,1\}^n} |x\rangle$. Then apply the operation $|xz\rangle \mapsto |x(z \oplus F(x))\rangle$ to $|av\rangle \otimes |0^n\rangle$, to obtain

$$\frac{1}{2^{\frac{n}{2}}} \sum_{x\in\{0,1\}^n} |x\rangle \otimes |F(x)\rangle$$

Now measure the second n bits of the register to put the second n qubits into some classical state z_0 to get a vector that is a constant multiple of:

$$\sum_{\{x|F(x)=z_0\}} |x\rangle \otimes |z_0\rangle.$$

Say $F(x_0) = z_0$, then (assuming $a \neq 0^n$) our sum collapses to

$$\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus a\rangle) \otimes |z_0\rangle.$$

We want to manipulate this vector to gain information about a.

For $x, y \in \mathbb{F}_2^n$, let $x \cdot y := \bigoplus_{j=1}^n x_j * y_j \in \mathbb{F}_2$ denote their inner product. Now perform the Hadamard operation on the first n bits again.

Exercise 3.3.3: Show that for $x \in \mathbb{F}_2^n$, $H^{\otimes n}|x\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{y\in\mathbb{F}_2^n} (-1)^{x\cdot y}|y\rangle$.

We obtain a constant times the vector

$$\sum_{y\in\mathbb{F}_2^n} \left((-1)^{x_0\cdot y} + (-1)^{(x_0\oplus a)\cdot y}\right) |y f(x_0)\rangle \tag{3.3.1}$$

Since $(x \oplus a) \cdot y = x \cdot y \oplus a \cdot y$, the term in the summand for any given y is non-zero if and only if $a \cdot y = 0$.

Thus when we measure the vector, the first n bits consists of some $y \in \mathbb{F}_2^n$ that is orthogonal to a with respect to the inner-product on $\mathbb{F}_2^n$ (and is chosen uniformly at random among such). So we may restrict our search for a to the hyperplane in $\mathbb{F}_2^n$ perpendicular to y. If we continue, with good luck, we could find a after $n-1$ runs of the algorithm. However, we have no guarantee we do not end up with linearly dependent y's. If we run the algorithm more than $2n$ times, then there will be $n-1$ independent hyperplanes with high probability.

Exercise 3.3.4: Prove that for any $a \in \mathbb{F}_2^n$, if vectors $y_1, \ldots, y_{2n}$ are chosen uniformly at random subject to $a \cdot y_j = 0$ for every j, then with high probability a subset of $n-1$ of them are linearly independent. ◎

3.4. Quantum gate sets

Previously we only needed the quantum gates corresponding to the Toffoli gate, $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, and the Hadamard matrix H. For Shor's algorithm one needs more general gates In this section we discuss admissible quantum gate sets. There are two issues we need to deal with: locality and approximation. We will first see, in §3.4.1 that locality is easily dealt with. Approximation is more subtle. We will first need to a way to measure how close an approximation is to the desired gate set. This will be done via the *operator norm* defined in §3.4.2. I introduce the "standard quantum gate set" in §3.4.4. Before that, in §3.4.3, I briefly discuss "controlled gates" which encode classical quantum interaction.

3.4.1. Locality. We would like to execute arbitrary unitary operations but we expect to only be able implement local quantum gates, as it is likely that one can only create *entanglement* in a laboratory on qubits that are physically close to one another. Thanks to the following unitary version of the classical Cartan-Dieudonné theorem, this issue is not a problem:

Lemma 3.4.1. *Any $U \in \mathbf{U}(n)$ may be written as a product of at most $\binom{n}{2}$ elements, each of which acts on some $\mathbb{C}\{e_i, e_{i+1}\}$ as an element of $\mathbf{U}(2)$ and is the identity on the span of $e_1, \ldots, e_{i-1}, e_{i+2}, \ldots, e_n$, where $e_1, \ldots, e_n$ is the standard basis of $\mathbb{C}^n$.*

Proof. Let $\mathbf{U}(2)_i \subset \mathbf{U}(n)$ be the copy of $\mathbf{U}(2)$ acting only on $\mathbb{C}\{e_i, e_{i+1}\}$. By Exercise 2.2.22, $\mathbf{U}(2)$ acts transitively on lines in $\mathbb{C}^2$. Moreover, it can send any $|v\rangle \in \mathbb{C}^2$ to $\begin{pmatrix} |v| \\ 0 \end{pmatrix}$. Thus for any unit vector $|\psi\rangle \in \mathbb{C}^n$, there exist $U_j \in \mathbf{U}(2)_j$ such that $U_1 \cdots U_n |\psi\rangle = e_1$. Write the columns of U^{-1} as $|\psi_1\rangle, \ldots, |\psi_n\rangle$ where $|\psi_j| = 1$. We may find $U_{1,1}, \ldots, U_{1,n-1}$ such that $U_{1,1} \cdots U_{1,n-1} |\psi_1\rangle = e_1$. Note that their effect on the remaining columns will make them orthogonal to e_1. Next we may find $U_{2,2}, \ldots, U_{2,n-1}$ with $U_{2,j} \in \mathbf{U}(2)_j$ such that their product applied to $U_{1,1} \cdots U_{1,n-1} |\psi_2\rangle$ is e_2. Continuing, we obtain $U = U_{n-1,n-1} U_{n-2,n-2} U_{n-3,n-2} \cdots U_{2,2} \cdots U_{2,n-1} U_{1,1} \cdots U_{1,n-1}$. □

3.4.2. Approximation. Ideally one would like to work with a universal gate set as in the classical case, but that will not be possible with a finite (or even countable) set of gates. This is similar to the situation in classical

computing where we could only create probability distributions with coefficients of the form $m/2^k$, where $m, k \in \mathbb{Z}_{\geq 0}$. If we start with a finite gate set, we cannot generate the entire unitary group. This results in quantum computation in general being not only probabilistic, as we have already discussed, but also *approximate*. This has not been an issue so far, but will be when we study Shor's algorithm.

We need to prove we can get arbitrarily "close" to any $U \in \mathbf{U}(\mathcal{H})$ with polynomially many elements from our finite gate set.

First we need to make the meaning of "close" precise, i.e., we need to choose a norm on $\mathrm{End}(\mathcal{H})$. Instead of working with a distance function on $\mathbf{U}(\mathcal{H})$, we work with a norm on $\mathrm{End}(\mathcal{H})$ because we will need to measure the norm of the difference of two unitary operators, which in general is not unitary. Moreover, $\mathrm{End}(\mathcal{H})$ has the advantage of being a linear space where distances are easier to work with.

Let V be a vector space. A *norm* on V is a function $||\ \cdot\ || : V \to \mathbb{R}_{\geq 0}$ satisfying, for all $v, w \in V$ and all $c \in \mathbb{C}$:

$$\begin{aligned} &||v|| \geq 0 \text{ with equality iff } v = 0, \\ &||v + w|| \leq ||v|| + ||w||, \\ &||cv|| = |c|\,||v||. \end{aligned}$$

In our case, we have $V = \mathrm{End}(\mathcal{H})$, and we take a norm that reflects this additional structure of our vector space as a space of operators, called the *operator norm*. It is particularly convenient for unitary operators.

Definition 3.4.2. For $X \in \mathrm{End}(\mathcal{H})$, define the *operator norm* of X,

$$||X|| := \sup_{|\xi\rangle \neq 0} \frac{|X|\xi\rangle|}{|\xi|}$$

where $|X|\psi\rangle|$ is the usual Hermitian norm of the vector $X|\psi\rangle$.

Exercise 3.4.3: Verify that the operator norm is indeed a norm.

Exercise 3.4.4: Prove that when $\mathcal{H}$ is finite dimensional, that $||X||^2$ is the largest eigenvalue of $X^\dagger X$.

For $X \in \mathrm{End}(\mathcal{H}_A), Z \in \mathrm{End}(\mathcal{H}_B)$, we may consider $X \otimes Z \in \mathrm{End}(\mathcal{H}_A \otimes \mathcal{H}_B)$. The operator norm has the following additional properties:

$$\begin{aligned} &||XY|| \leq ||X||\,||Y||, \\ &||X^\dagger|| = ||X||, \\ &||X \otimes Z|| = ||X||\,||Z||, \\ &||U|| = 1\ \forall U \in \mathbf{U}(\mathcal{H}). \end{aligned}$$

Exercise 3.4.5: Verify these additional properties.

Definition 3.4.6. An operator $\tilde{U} \in \mathbf{U}(\mathcal{H})$ *approximates* $U \in \mathbf{U}(\mathcal{H})$ *with precision* δ if $||\tilde{U} - U|| \leq \delta$.

Exercise 3.4.7: Show that $\tilde{U} \in \mathbf{U}(\mathcal{H})$ approximates $U \in \mathbf{U}(\mathcal{H})$ with precision δ if and only if $\tilde{U}^{-1}$ approximates U^{-1} with precision δ.

We will not be concerned with a single unitary transformation, but the product of many such, so we need to examine how errors grow under composition of maps. The key property of the operator norm is that for unitary transformations, errors accumulate linearly:

Proposition 3.4.8. *Say* $U = U_L \cdots U_2 U_1$ *with* $U, U_j \in \mathbf{U}(\mathcal{H})$*, and that* U_j *is approximated by* $\tilde{U}_j \in \mathbf{U}(\mathcal{H})$ *with precision* δ_j*. Then* $\tilde{U} := \tilde{U}_L \cdots \tilde{U}_2 \tilde{U}_1$ *approximates* U *with precision* $\sum_{j=1}^{L} \delta_j$*.*

Proof. By induction, it will be sufficient to prove the case $L = 2$. We have

$$\begin{aligned} ||\tilde{U}_2\tilde{U}_1 - U_2U_1|| &= ||\tilde{U}_2(\tilde{U}_1 - U_1) + (\tilde{U}_2 - U_2)U_1|| \\ &\leq ||\tilde{U}_2(\tilde{U}_1 - U_1)|| + ||(\tilde{U}_2 - U_2)U_1|| \\ &\leq ||\tilde{U}_2||\,||\tilde{U}_1 - U_1|| + ||\tilde{U}_2 - U_2||\,||U_1|| \\ &\leq ||\tilde{U}_1 - U_1|| + ||\tilde{U}_2 - U_2||. \end{aligned}$$

□

This linear accumulation of errors allows for good approximation as we will see in Theorem 3.4.12 below.

3.4.3. Notation for classically controlled quantum gates. In Simon's algorithm, we used classical gates to decide if a quantum gate (a reflection) was applied. We now introduce notation for such classical "controls" in a quantum circuit.

For $U \in \mathbf{U}(n)$, introduce *k-controlled* U, $\Lambda^k(U) : \mathbb{C}^k \otimes \mathbb{C}^n \to \mathbb{C}^k \otimes \mathbb{C}^n$ by

$$\Lambda^k(U)(|x_1, \ldots, x_k\rangle \otimes |\xi\rangle) = \begin{cases} |x_1, \ldots, x_k\rangle \otimes |\xi\rangle & \text{if } x_1 \cdots x_k = 0 \\ |x_1, \ldots, x_k\rangle \otimes U|\xi\rangle & \text{if } x_1 \cdots x_k = 1. \end{cases}$$

When acting by these controlling bits, we will allow violation of strict locality for the controlling bits (they will be the "last" s bits, as was the situation with Grover's and Simons' algorithms). This is physically acceptable, because it will correspond to interfering with the quantum system from "outside".

Introduce the notation (taken from physics) $\sigma_x := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

Exercise 3.4.9: Show that $\Lambda^1(\sigma_x)|ab\rangle = |a, a \oplus b\rangle$.

Exercise 3.4.10: Show the Toffoli gate is $\Lambda^2(\sigma_x)$.

It is exactly in implementing these classical-quantum gates that we will allow violation of strict locality- the Hilbert spaces corresponding to the classical bits need not be adjacent to the Hilbert spaces corresponding to the quantum bits in these gates.

3.4.4. The standard quantum gate set.

Definition 3.4.11. The quantum gate set

$$H := \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \text{ (the Hadamard gate),} \tag{3.4.1}$$

$$K := \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}, \tag{3.4.2}$$

$$K^{-1}, \tag{3.4.3}$$

$$\Lambda^1(\sigma_x) \text{ where } \sigma_x = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}, \tag{3.4.4}$$

$$\Lambda^2(\sigma_x) \text{ the Toffoli gate} \tag{3.4.5}$$

is called *standard*.

Theorem 3.4.12. *Any $U \in \mathbf{U}(n)$ can be realized with precision δ by a $poly(\log(\frac{1}{\delta}))$-size circuit over the standard basis, using workspace bits (i.e., working with elements of $\mathbf{U}(n+s)$). Moreover, there exists a polynomial size algorithm constructing the circuit.*

Theorem 3.4.12 justifies the assertion that one can achieve good approximate quantum algorithms from a fixed gate set. For the proof, see [**KSV02**, Thm. 13.5].

3.5. Shor's algorithm

Shor's algorithm involves a classical part and a quantum part. The quantum part is: given a randomly chosen $a \in (\mathbb{Z}/N\mathbb{Z})^*$, find the order of a in $(\mathbb{Z}/N\mathbb{Z})^*$, that is the smallest r, such that $a^r \equiv 1 \bmod N$. The classical part uses the output of the quantum algorithm to find a factor of N. The size of the quantum circuit will be $poly(\log(N))$. Since there are at most $\log(N)$ factors of N, it will still be $poly(\log(N))$-operations to factor N completely.

The quantum part will hinge on i) there being "enough" prime numbers less than N, and ii) the ability to "closely" approximate a rational number by other rational numbers. After explaining the classical part, I address these two issues and then give the quantum part.

3.5.1. The classical part. Recall Proposition 3.1.7 which gave a way to find a factor of N, namely if there exists b such that $b^2 \equiv 1 \bmod N$ and $b \not\equiv \pm 1 \bmod N$, then both $b+1, b-1$ are factors of N.

The quantum part produces an a and its order r.

The classical part will consist in showing that with probability at least $\frac{1}{4}$, r will be even and $a^{\frac{r}{2}} \not\equiv \pm 1 \bmod N$. Applying Proposition 3.1.7 with $b = a^{\frac{r}{2}}$ we obtain our factor. (To get success with probability greater than $\frac{1}{2}$, just run the whole algorithm three times.)

First we treat the special case that $N = p_1 p_2$ where p_1, p_2 are distinct primes. Here we will succeed with probability at least $\frac{1}{2}$. Recall by the Chinese remainder theorem that $(\mathbb{Z}/N\mathbb{Z})^* \simeq (\mathbb{Z}/p_1\mathbb{Z})^* \times (\mathbb{Z}/p_2\mathbb{Z})^*$. Write r_j for the order of a in $(\mathbb{Z}/p_j\mathbb{Z})^*$, so $r = \operatorname{lcm}(r_1, r_2)$. Write $r_j = 2^{s_j} r'_j$. Without loss of generality, assume $s := s_2 \geq s_1$, so $r = 2^s r'$ with r' odd.

The first bad case is when $s = 0$. We'll see this occurs with low probability later.

The second bad case is when $a^{\frac{r}{2}} \equiv \pm 1 \bmod N$. Claim: this will not happen if $s_1 < s$. To see this, note that in this case r_1 divides $\frac{r}{2}$, so $a_1^{\frac{r}{2}} \equiv 1 \bmod p_1$. But r_2 does not $\frac{r}{2}$, so $a_2^{\frac{r}{2}} \not\equiv 1 \bmod p_2$. Since under the Chinese remainder theorem isomorphism $1 \mapsto (1,1)$ and $-1 \mapsto (-1,-1)$ we see $a^{\frac{r}{2}} \not\equiv \pm 1 \bmod N$.

It remains to show that $s_1 = s_2$ happens with probability at most $\frac{1}{2}$.

Exercise 3.5.1: Show that the group $(\mathbb{Z}/p\mathbb{Z})^*$ is cyclic of order $p-1$. ⊙

Let g_j be a generator of $(\mathbb{Z}/p\mathbb{Z})^*$. Our choice of random a may be then considered as a choice of a random element of $[p_1 - 1] \times [p_2 - 1]$, where (u, v) corresponds to the element $(g_1^u \bmod p_1, g_2^v \bmod p_2)$. Thus the test succeeds if either u is odd and v is even or vice versa. The odds of this are $\frac{1}{4} + \frac{1}{4} = \frac{1}{2}$ and we conclude.

The case $N = p_1 \cdots p_q$ is identical. The case $N = p^\alpha$ should really be checked before: there is a small (see (3.5.1)) list of prime numbers less than $\sqrt{N}$ and one just needs to check their powers. The case $N = p_1^{\alpha_1} \cdots p_k^{\alpha_k}$ is similar to the case $N = p_1 \cdots p_q$. In the notation above, the algorithm fails if and only if $s_1 = s_2 = \cdots = s_k$. The only new ingredient needed is the following exercise:

Exercise 3.5.2: Show that for p prime, $|(\mathbb{Z}/p^\alpha\mathbb{Z})^*| = p^\alpha - p^{\alpha-1}$.

Write $p^\alpha - p^{\alpha-1} = 2^t q$ where q is odd and note that $t > 0$. Let g be a generator of $(\mathbb{Z}/p^\alpha\mathbb{Z})^*$. Write the order of a as $2^{s_a} r'_a$ where r'_a is odd.

$$\#\{a \in (\mathbb{Z}/p^\alpha\mathbb{Z})^* \mid s_a = s\} = \#\{\text{powers } g^{2^{t-s}m} \text{ with } m \text{ odd}\}$$
$$= \left\{ \begin{array}{cc} q & s = 0 \\ (2^s - 2^{s-1})q & s = 1, \ldots, t \\ 0 & s > t \end{array} \right\}.$$

All these numbers are at most $\frac{1}{2}(p^\alpha - p^{\alpha-1})$ because $t > 0$, so they have probability at most $\frac{1}{2}$ of being selected, so we conclude the probability of failure is at most $\frac{1}{2^{k-1}}$.

3.5.2. Preliminaries I: the number of primes. Let $\pi(x)$ denote the number of prime numbers that are less than or equal to x. The *prime number theorem* states $\pi(x) \sim \frac{x}{\ln(x)}$, more precisely,

$$\lim_{x\to\infty} \frac{\pi(x)}{x/\ln(x)} = 1. \tag{3.5.1}$$

The proof is not so simple, but for our purposes, the following result will suffice:

Theorem 3.5.3. *For all $x \geq 2$,*

$$\pi(x) \geq \frac{x}{2\log(x)}.$$

(In [**?**] it is shown that $\frac{x}{\log x} < \pi(x)$ for $x \geq 17$.) We give the proof after several preliminary lemmas. Let $v_p(m)$ denote the largest power of p that divides m. Thus $m = \Pi_{p\leq m} p^{v_p(m)}$.

Lemma 3.5.4.

$$v_p(n!) = \sum_{m\geq 1} \lfloor \frac{n}{p^m} \rfloor.$$

Proof. Among $1, 2, \ldots, n$, exactly $\lfloor \frac{n}{p} \rfloor$ are multiples of p, contributing $\lfloor \frac{n}{p} \rfloor$ to the summation, exactly $\lfloor \frac{n}{p^2} \rfloor$ are multiples of p^2, contributing an additional $\lfloor \frac{n}{p^2} \rfloor$ to the summation (and thus multiples of p^2 are counted two times). Continuing, one gets the result. □

Recall that $\binom{m}{q} = \frac{m!}{q!(m-q)!}$. Note that

$$v_p(\binom{2n}{n}) = v_p((2n)!) - 2v_p(n!) = \sum_{m\geq 1} \lfloor \frac{2n}{p^m} \rfloor - 2\lfloor \frac{n}{p^m} \rfloor.$$

Exercise 3.5.5: Show that for all $x \in \mathbb{R}$, $\lfloor 2x \rfloor - 2\lfloor x \rfloor \in \{0, 1\}$. ⊙

Note that if $\frac{2n}{p^m} < 1$, i.e., $m > \frac{\log(2n)}{\log(p)}$, then $\lfloor \frac{2n}{p^m} \rfloor - 2\lfloor \frac{n}{p^m} \rfloor = 0$, and thus

$$v_p(\binom{2n}{n}) \leq \frac{\log(2n)}{\log(p)}. \tag{3.5.2}$$

Stirling's formula allows one to estimate $n!$:

$$\ln(n!) = n\ln(n) - n + O(\ln(n)), \tag{3.5.3}$$

$$\log(n!) = n\log(n) - \log(e)n + O(\log(n)). \tag{3.5.4}$$

It is often proved using a contour integral of the Gamma function (see, e.g., [**Ahl78**, §5.2.5]). To see why it is plausible, write $\ln(n!) = \ln(1) + \cdots + \ln(n)$. This quantity may be estimated by

$$\int_1^n \ln(x)dx = [x\ln(x) - x]_1^n = n\ln n - n + 1,$$

giving intuition to (3.5.3).

Exercise 3.5.6: Show that $\binom{2n}{n} \geq \frac{2^{2n}}{2n}$. ⊚

Lemma 3.5.7. $\pi(2n) \geq \frac{2n}{\log(2n)} - 1$.

Proof. By Exercise 3.5.6

$$\begin{aligned} 2n\log(2) - \log(2n) &\leq \log\binom{2n}{n} \\ &= \log(\Pi p^{v_p(\binom{2n}{n})}) \\ &\leq \sum_{p \leq 2n} \lfloor \frac{\log(2n)}{\log(p)} \rfloor \log(p) \quad \text{by (3.5.2)} \\ &\leq \sum_{p \leq 2n} \log(2n) \\ &= \pi(2n)\log(2n). \end{aligned}$$

Thus

$$\pi(2n) \geq \frac{2n}{\log(2n)} - 1. \tag{3.5.5}$$

□

Proof of Theorem 3.5.3. For $x \leq 16$, one can check the result by hand. So assume $x > 16$ and take n such that $16 \leq 2n \leq x < 2n+2$. Then

$$\begin{aligned}\pi(x) \geq \pi(2n) \geq \frac{2n}{\log(2n)} - 1 =& \frac{1}{2\log(2n)}[4n - \frac{1}{2\log(2n)}] \\ &\geq \frac{1}{2\log(x)}[4n - \frac{1}{2\log(2n)}] \\ &\geq \frac{x}{2\log(x)}.\end{aligned}$$

The last line holds because we have $x \leq 2n+2 \leq 4n - \frac{1}{2\log(2n)}$, since $n \geq 8$. □

We will use Theorem 3.5.3 in the form:

Corollary 3.5.8. *For every natural number r, there are $\Omega(\frac{r}{\log r})$ numbers in $\{1, \ldots, \frac{r}{10}\}$ relatively prime to r.*

Proof. r has at most $\log(r)$ prime factors and there are at least $\frac{r/10}{2\log(r/10)} = \frac{r}{20(\log(r)-\log(10))} > \frac{4}{20\log(r)}$ prime numbers less than $\frac{r}{10}$. □

3.5.3. Preliminaries II: Continued Fractions. We will need to approximate real numbers by sequences of rational numbers. A first idea would simply be to use the decimal expansion. The following scheme has better convergence properties for our purposes. (E.g., consider the decimal expansion of π compared with the results below.) Given $\alpha \in \mathbb{R}$, consider the expansion

$$\alpha = \alpha_0 + \cfrac{1}{\alpha_1 + \cfrac{1}{\alpha_2 + \cfrac{1}{\alpha_3 + \cfrac{1}{\alpha_4 + \cdots}}}}$$

where $\alpha_0 = \lfloor \alpha \rfloor$,

$$\alpha_1 = \frac{1}{\alpha - \alpha_0} - \lfloor \frac{1}{\alpha - \alpha_0} \rfloor$$

and in general, α_k is the integer part of the reciprocal of the error term in the previous estimate. Write $\frac{p_n}{q_n} = [\alpha_0, \ldots, \alpha_n]$ for the rational number obtained after the n-th step, which is an approximation to the real number α.

For example, taking the continued fraction expansion of π, one obtains $\frac{22}{7} = [3, 7] \sim 3.142$, $\frac{333}{106} = [3, 7, 15] \sim 3.1415$, $\frac{355}{113} = [3, 5, 15, 1] \sim 3.1415926$.

If α is rational, we will see momentarily that the algorithm reproduces α. For example $\frac{11}{9} = 1 + \frac{2}{9}$, so $\alpha_0 = 1$. Since $\frac{9}{2} = 1 + \frac{1}{4}$, we have $\alpha_1 = 4$, and since $2 = 2 + 0$, we have $\alpha_2 = 2$ and $\frac{11}{9} = [1, 4, 2]$.

Proposition 3.5.9. *If the continued fraction expansion of $\alpha \in \mathbb{R}_{>0}$ does not converge at the n-th step, then after the n-th step one obtains $[\alpha_0, \ldots, \alpha_n] = \frac{a}{b} \in \mathbb{Q}$ with $b \geq 2^{\frac{n}{2}}$.*

Proof. Write $[\alpha_0, \ldots, \alpha_n] = \frac{p_n}{q_n}$. Then $p_0 = \alpha_0$, $q_0 = 1$, $p_1 = 1 + p_0\alpha_1$, $q_1 = \alpha_1$ and

$$\begin{aligned} p_k &= \alpha_k p_{k-1} + p_{k-2} \\ q_k &= \alpha_k q_{k-1} + q_{k-2}. \end{aligned} \tag{3.5.6}$$

Exercise 3.5.10: Verify the equalities (3.5.6).

Since $\alpha_j > 0$, we have $p_j \geq 2p_{j-2}$ and $q_j \geq 2q_{j-2}$, so $p_{n-1}, q_{n-1} \geq 2^{\lfloor \frac{n}{2} \rfloor}$, proving the lower bound on b. □

Theorem 3.5.11. *Let $\alpha, \frac{s}{r} \in \mathbb{Q}$ be such that $|\frac{s}{r} - \alpha| \leq \frac{1}{2r^2}$. Then $\frac{s}{r}$ appears in the continued fraction expansion for α. In particular, the expansion for $\alpha = \frac{a}{b}$ terminates after $2\log(b)$ steps.*

Theorem 3.5.11 says that if two rational numbers are close to each other, then their continued fraction expansions start out the same.

Proof. Take the continued fraction expansion of $\frac{s}{r} = [\beta_0, \ldots, \beta_n]$, and write $\frac{p_j}{q_j} = [\beta_0, \ldots, \beta_j]$. Define δ by the equation

$$\alpha = \frac{s}{r} + \frac{\delta}{2r^2}.$$

Note that δ is a measure of the failure of $\frac{p_n}{q_n} = \frac{s}{r}$ to be equal to α. The hypothesis implies $|\delta| < 1$. Claim:

$$\alpha = \frac{\lambda p_n + p_{n-1}}{\lambda q_n + q_{n-1}}$$

for

$$\lambda = 2\left(\frac{q_n p_{n-1} - p_n q_{n-1}}{\delta}\right) - \frac{q_{n-1}}{q_n}.$$

Exercise 3.5.12: Prove the claim.

Slightly abusing notation (as $\lambda \notin \mathbb{Z}$) $\alpha = [\beta_0, \ldots, \beta_n, \lambda]$. Since $\lambda \in \mathbb{Q}$, we may write $\lambda = [\gamma_0, \ldots, \gamma_m]$, so $\alpha = [\beta_0, \ldots, \beta_n, \gamma_0, \ldots, \gamma_m]$ and $\frac{s}{r}$ appears at the n-th step. □

In summary: Given $\alpha \in \mathbb{R}$ and $N \in \mathbb{Z}$, the continued fraction algorithm, in $poly(\log(N))$ steps finds $\frac{a}{b} \in \mathbb{Q}$ such that $b \leq 16N$ and $\frac{a}{b}$ approximates α better than any other rational number with denominator at most b.

3.5.4. Preliminaries III: the quantum Discrete Fourier Transform. Recall the DFT for $\mathbb{Z}/M\mathbb{Z}$, which we may write in vector notation, for $j \in \mathbb{Z}/M\mathbb{Z}$, as

$$|j\rangle \mapsto \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} \omega^{jk}|k\rangle$$

where $\omega = e^{\frac{2\pi i}{M}}$. Also recall: the DFT is a unitary change of basis such that in the new basis, multiplication in $\mathbb{Z}/M\mathbb{Z}$ is given by a diagonal matrix, and the classical FFT writes the DFT as a product of $O(\log(M))$ sparse matrices (each with $M << M^2$ nonzero entries), for a total cost of $O(\log(M)M) < O(M^2)$ arithmetic operations to execute.

Write $M = 2^m \sim N^3$. We show that the DFT can be written as a product of $O(m^3) = O(\log(M)^3)$ controlled local unitary operators. We will thus be able to produce an approximation of the output vector by a sequence of $poly(m)$ unitary operators from our gate set.

It will be convenient to express j in binary and view $\mathbb{C}^M = (\mathbb{C}^2)^{\otimes m}$, i.e., write

$$|j\rangle = |j_1\rangle \otimes \cdots \otimes |j_m\rangle$$

where $j = j_1 2^{m-1} + j_2 2^{m-2} + \cdots + j_m 2^0$ and $j_i \in \{0, 1\}$. Write the DFT as

$$\begin{aligned}
&|j_1\rangle \otimes \cdots \otimes |j_m\rangle \\
&\mapsto \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} \omega^{jk}|k\rangle \\
&= \frac{1}{\sqrt{M}} \sum_{k_i \in \{0,1\}} \omega^{j(\sum_{l=1}^m k_l 2^{m-l})}|k_1\rangle \otimes \cdots \otimes |k_m\rangle \\
&= \frac{1}{\sqrt{M}} \sum_{k_i \in \{0,1\}} \bigotimes_{l=1}^m \left[\omega^{jk_l 2^{m-l}}|k_l\rangle\right] \\
&= \frac{1}{\sqrt{M}} \sum_{k_i \in \{0,1\}} \bigotimes_{l=1}^m \left[\omega^{(j_1 2^{2m-1-l} + \cdots + j_m 2^{m-l})k_l}|k_l\rangle\right]
\end{aligned}$$

(3.5.7)

$$\begin{aligned}
&= \frac{1}{2^{\frac{m}{2}}}(|0\rangle + \omega^{j_m 2^{-1}}|1\rangle) \otimes (|0\rangle + \omega^{j_{m-1}2^{-1} + j_m 2^{-2}}|1\rangle) \otimes (|0\rangle + \omega^{j_{m-2}2^{-1} + j_{m-1}2^{-2} + j_m 2^{-3}}|1\rangle) \\
&\otimes \cdots \otimes (|0\rangle + \omega^{j_1 2^{-1} + j_2 2^{-2} + \cdots + j_m 2^{-m}}|1\rangle)
\end{aligned}$$

where for the last line if $2m - s - l > m$, i.e., $s + l < m$, there is no contribution with j_s because $\omega^{2^m} = 1$, and we multiplied all terms by $1 = \omega^{-2^m}$ to have negative exponents.

It will be notationally more convenient to write the quantum circuit for this vector with the order of factors reversed. I.e., we describe a quantum

circuit that produces

(3.5.8)
$$\frac{1}{\sqrt{2}}(|0\rangle + \omega^{j_1 2^{-1}+j_2 2^{-2}\cdots+j_m 2^{-m}}|1\rangle)\otimes\cdots\otimes\frac{1}{\sqrt{2}}(|0\rangle + \omega^{j_{m-2}2^{-1}+j_{m-1}2^{-2}+j_m 2^{-3}}|1\rangle)$$
$$\otimes\frac{1}{\sqrt{2}}(|0\rangle + \omega^{j_{m-1}2^{-1}+j_m 2^{-2}}|1\rangle)\otimes\frac{1}{\sqrt{2}}(|0\rangle + \omega^{j_m 2^{-1}}|1\rangle).$$

Set

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & \omega^{2^{-k}} \end{pmatrix}, \tag{3.5.9}$$

then (3.5.8) is obtained from $|j_1\rangle\otimes\cdots\otimes|j_m\rangle$ as follows: first apply H to $(\mathbb{C}^2)_1$.

Exercise 3.5.13: Show that if $\omega = e^{\frac{2\pi i}{2^m}}$, then $\omega^{2^{-1}} = -1$.

Since $\omega^{2^{-1}} = -1$, at this point the first factor becomes $\frac{1}{\sqrt{2}}(|0\rangle+\omega^{j_1 2^{-1}}|1\rangle)$. Next apply $\Lambda^1 R_2$ to $(\mathbb{C}^2)_2\otimes(\mathbb{C}^2)_1$, so the first factor becomes $\frac{1}{\sqrt{2}}((|0\rangle + \omega^{j_1 2^{-1}+j_2 2^{-1}}|1\rangle))$. Continue, applying $\Lambda^1 R_j$ to $(\mathbb{C}^2)_j\otimes(\mathbb{C}^2)_1$ for $j = 3,\dots,m$. Note that at this point only the $(\mathbb{C}^2)_1$-term has been altered, as all the other factors only acted as controlling qubits.

From now on we leave the $(\mathbb{C}^2)_1$-slot alone. Next apply H to $(\mathbb{C}^2)_2$, followed by R_1. Then apply $\Lambda^1 R_{j-1}$ to $(\mathbb{C}^2)_j\otimes(\mathbb{C}^2)_2$ for $j = 3,\dots,m$.

Next apply H followed by R_2 to $(\mathbb{C}^2)_3$, then $\Lambda^1 R_{j-2}$ to $(\mathbb{C}^2)_j\otimes(\mathbb{C}^2)_3$ for $j = 4,\dots,m$. Continue, until finally one just applies H to $(\mathbb{C}^2)_m$. Finally to obtain the DFT, reverse the orders of the factors (a classical operation).

In practice, one has to fix a quantum gate set in advance, so in general we will have to approximate the transformations R_k from elements of our gate set, so we will only approximate the DFT.

3.5.5. The order finding algorithm. We are given a and N with $\gcd(a,N) = 1$ and want to find the order of a in $(\mathbb{Z}/N\mathbb{Z})^*$. Set $m = \lceil 3\log(N)\rceil$, $M = 2^m \sim N^3$ and $n = \lceil\log(N)\rceil$. Work in $(\mathbb{C}^2)^{\otimes(m+n)}$. (Here and in what follows, I ignore the additional bits needed to execute the classical reversible computations.) Initialize the state to $|0^{m+n}\rangle$. Apply the quantum Fourier transform to the first m qubits to obtain

$$\frac{1}{\sqrt{M}}\sum_{x\in\mathbb{Z}/M\mathbb{Z}}|x\rangle\otimes|0^n\rangle.$$

Use our (polynomial in n) reversible powering mod N routine $|x\rangle\otimes|y\rangle \mapsto |x\rangle\otimes|y \oplus a^x \bmod N\rangle$ to obtain

$$\frac{1}{\sqrt{M}} \sum_{x\in\mathbb{Z}/M\mathbb{Z}} |x\rangle\otimes|a^x \bmod N\rangle.$$

Here and in what follows, $a^x \bmod N$ is to be considered as an element of $\{0,\dots,N-1\}$ expressed in binary, so indeed $|a^x \bmod N\rangle \in (\mathbb{C}^2)^{\otimes n}$. Now perform a measurement on the last n qubits to obtain some $y_0 \in \{0,1\}^n$ that appears in the image of the map $x \mapsto a^x \bmod N$. The resulting vector is

$$\frac{1}{\sqrt{\#\{x \mid a^x \bmod N = y_0\}}} \sum_{x|a^x \bmod N = y_0} |x\rangle\otimes|y_0\rangle.$$

Let x_0 be the smallest natural number such that $a^{x_0} \equiv y_0 \bmod N$. Then we may rewrite the output vector as

$$\frac{1}{\sqrt{K}} \sum_{\ell=0}^{K-1} |x_0 + \ell r\rangle\otimes|y_0\rangle, \tag{3.5.10}$$

where

$$K = \lfloor \frac{M-(1+x_0)}{r} \rfloor \tag{3.5.11}$$

and r is the period.

Compare (3.5.10) with the step in Simons' algorithm giving rise to (3.3.1). As with Simons' algorithm, we now have a vector that "sees" r that we will manipulate to get information about r.

Apply the quantum Fourier transform to the first m qubits again to obtain

$$\frac{1}{\sqrt{M}\sqrt{K}} \sum_{x\in\mathbb{Z}/M\mathbb{Z}} \sum_{\ell=0}^{K-1} \omega^{(x_0+\ell r)x} |x\rangle\otimes|y_0\rangle.$$

Finally measure the first m qubits to obtain some $x \in \mathbb{Z}/M\mathbb{Z}$.

How will this be useful? We claim that with sufficiently high probability, the algorithm will produce an x such that $\frac{x}{M}$ will be close to some fraction with denominator r, say $\frac{b}{r}$, more precisely that $|\frac{b}{r} - \frac{x}{M}| < \frac{1}{2r^2}$. When we take the partial fraction decomposition of $\frac{x}{M}$, by Theorem 3.5.11 the term $\frac{b}{r}$ will appear, so we test the denominators q_j and take the first one such that $a^{q_j} \equiv 1 \bmod N$.

We need to show there are "enough" such good x each with sufficiently high probability of being chosen that we succeed. Since we may repeat the experiment $poly(\log N)$ times while still being in polynomial time, if G is the number of good x and P the probability of drawing any given good x,

we need $GP = \Omega(\frac{1}{\log N})$. We will show $G = \Omega(\frac{r}{\log r})$ and $P = \Omega(\frac{1}{r})$, and since $r < N$, this will suffice.

The probability of drawing any given $|x\rangle$ is

$$\begin{aligned}&\frac{1}{MK}|\omega^{x_0 x} + \omega^{(x_0+r)x} + \omega^{(x_0+2r)x} + \cdots + \omega^{(x_0+(K-1))rx}|^2\\ =&\frac{1}{MK}|1 + \omega^{rx} + \omega^{2rx} + \cdots + \omega^{(K-1)rx}|^2\end{aligned}$$

To fix ideas, first consider the highly improbable case $M = rc$ for some natural number c. Write $\eta = \omega^{rx}$. Note that $\eta^c = (\omega^{rc})^x = 1$. If $c \nmid x$, then $1 + \eta + \eta^2 + \cdots + \eta^{c-1} = 0$ and the probability of drawing such x is zero, but if $c \mid x$, each power equals 1 as $\omega^{rjx} = \omega^{\frac{M}{c}jx} = (\omega^M)^{j\frac{x}{c}} = 1$ as $\frac{x}{c} \in \mathbb{Z}$, and thus we will draw $x = cb$ for a random $b \in \{0, \ldots, r-1\}$ so $\frac{x}{M} = \frac{cb}{rc} = \frac{b}{r}$ and all such x's are equally likely. Thus in this case $P = 1$ and $G = r$.

Now for the general case: the idea is similar- we will prove that the values of x that are most likely to be drawn will be such that xr is *nearly* divisible by M. Recall that we also need $\gcd(\lfloor\frac{xr}{M}\rfloor, r) = 1$. We first show $G = \Omega(\frac{r}{\log(r)})$.

Lemma 3.5.14. *There exist $\Omega(\frac{r}{\log(r)})$ elements $x \in \mathbb{Z}/M\mathbb{Z}$ satisfying:*

i) $0 \leq xr \bmod M < \frac{r}{10}$, *and*

ii) $\gcd(\lfloor\frac{xr}{M}\rfloor, r) = 1$.

Proof. Consider the case r is odd, i.e., $\gcd(r, M) = 1$. Then the map $x \mapsto rx \bmod M$ is a permutation on $(\mathbb{Z}/M\mathbb{Z})^*$. By Corollary 3.5.8 there are $\Omega(\frac{r}{\log(r)})$ elements x such that $xr \bmod M < \frac{r}{10}$ and $\gcd(x, r) = 1$.

Since $xr \bmod M = rx - \lfloor\frac{rx}{M}\rfloor M$, we have $\gcd(\lfloor\frac{rx}{M}\rfloor, r) = 1$ as otherwise $rx \bmod M$ would also have a factor in common with r.

When r is even, write $d = \gcd(r, M)$, set $r' = \frac{r}{d}$ and $M' = \frac{M}{d}$.

Exercise 3.5.15: Apply the same argument to show there exists $\Omega(\frac{r}{d\log(r)})$ x's in $\mathbb{Z}/M'\mathbb{Z}$ satisfying the condition. Finally show that for all $c \in \mathbb{N}$, $x + cM$ also satisfies the condition.

□

It remains to show $P = \Omega(\frac{1}{r})$:

Lemma 3.5.16. *If x is such that $0 < xr \bmod M < \frac{r}{10}$, then the probability of drawing x in Shor's algorithm is $\Omega(\frac{1}{r})$.*

Proof. The probability of drawing x is

$$\frac{1}{KM}|\sum_{\ell=0}^{K-1}\omega^{\ell rx}|^2.$$

Recalling the sum of a geometric series, this is

$$\frac{1}{KM}|\frac{1-\omega^{rKx}}{1-\omega^{rx}}|^2 = \frac{1}{KM}|\frac{1-(\cos(\frac{rKx}{M})+i\sin(\frac{rKx}{M}))}{1-(\cos(\frac{rx}{M})+i\sin(\frac{rx}{M}))}|^2.$$

Using that $\frac{rKx}{M}, \frac{rx}{M}$ are both small so their cosines are close to 1, this is close to

$$\frac{1}{KM}|\frac{\sin(\frac{rKx}{M})}{\sin(\frac{rx}{M})}|^2$$

and again using that these quantities are small, this is close to

$$\frac{1}{KM}|\frac{\frac{rKx}{M}}{\frac{rx}{M}}|^2 = \frac{1}{KM}|K|^2 = \frac{K}{M}$$

Since $\frac{K}{M} > \frac{1}{2r}$ we conclude. □

3.5.6. Putting it all together. The first assertion of Lemma 3.5.14 implies

$$|xr - cM| < \frac{r}{10}$$

for $c = \lceil \frac{xr}{M} \rceil$, i.e.,

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

Since $M \sim N^3$ and $r < N$, Theorem 3.5.11 applies and we conclude that Shor's algorithm will succeed in polynomial time with high probability.

3.6. A unified perspective on quantum algorithms: the hidden subgroup problem

Given G: a discrete group with a specific representation of its elements in binary, an explicit function $f : G \to \mathbb{F}_2^n$, and the knowledge that there exists a subgroup $G' \subset G$ such that $f(x) = f(y)$ if and only if $xy^{-1} \in G'$, find G'.

For abelian groups, it is sufficient to solve the problem for $G = \mathbb{Z}^{\oplus k}$ as all abelian groups are quotients of some $\mathbb{Z}^{\oplus 2k}$.

Simons algorithm is the case $G = \mathbb{Z}_2^{\oplus m}$. The DFT_2 matrix is

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

and G' is the subgroup generated by $a \in \mathbb{Z}_2^{\oplus m}$.

Shor's algorithm is the case $G = \mathbb{Z}$ and f is the function $x \mapsto a^x \bmod N$. Note that with Shor, as we did with multiplying polynomials, we restricted to $\mathbb{Z}/M\mathbb{Z}$ for M sufficiently large.

Both are solved via the DFT for the finite group G.

Another example, closely related to order finding, is as follows: the *discrete logarithm* of a number a at base $\zeta = e^{\frac{2\pi i}{N}}$ is the smallest positive integer s such that $\zeta^s = a$. Consider the function $f : \mathbb{Z}^2 \to \mathbb{Z}_N$ given by $f(x_1, x_2) = \zeta^{x_1} a^{x_2} \bmod N$. Take $G = \mathbb{Z}^2$, $G' = \{(x_1, x_2) \in \mathbb{Z}^2 \mid \zeta^{x_1} a^{x_2} \equiv 1 \bmod N\}$. Given G', just find an element of the form $(s, -1) \in G'$, and s is the discrete logarithm of a at base ζ.

3.7. What is a quantum computer?

Currently there are three approaches towards building a quantum computer: adiabatic quantum optimization (D Wave), digital (IBM), topological (currently science fiction).

The main problem of quantum computing is on the one hand, one wants an isolated system to be resistant to outside noise, on the other hand one needs to be able to manipulate it.

3.7.1. D Wave's computers. Claims of $2,000$ qubits. Machine only designed for quadratic optimization. Yet to have an advantage over a classical computer. Internal workings of machine kept private. Principle quantum cousin of MRI. If they can scale up significantly, despite these drawbacks, they will beat classical computers.

3.7.2. IBM's digital quantum computer. It really exists and is quantum, and the world is free to examine it. 53 qubits as of 2020.

3.7.3. Topological quantum computing. technology not yet there...

3.8. Appendix: review of basic information on groups and rings

Let S be a set, a *binary operation* on S is a map $f : S \times S \to S$. One often writes $f(x, y) = x * y$.

A *group* is a set G with a binary operation such that 1) for all $x, y, z \in G$, $x * (y * z) = (x * y) * z)$ (associativity), 2) there exists and identity element $e \in G$ such that $e * x = x * e = x$ for all $x \in G$ and 3) for all $x \in G$, there exists an inverse $x^{-1} \in G$ such that $x * x^{-1} = x^{-1} * x = e$.

Examples: $(G, *) = (\mathbb{Z}, +)$, $\mathbf{U}(n)$ with operation matrix multiplication. Let $\mathbb{Z}/N\mathbb{Z}$ denote the set of equivalence classes in the integers defined by

remainder under division by N. Then $(\mathbb{Z}/N\mathbb{Z})^*$, the set of elements of $\mathbb{Z}/N\mathbb{Z}$ with multiplicative inverses is a group, where the binary operation is multiplication inherited from multiplication of the integers: $[x] * [y] = [x * y]$ (one must verify this is well defined).

Non-examples: the set of stochastic matrices with operation matrix multiplication, the nonzero integers with operation multiplication.

A group G is *abelian* if $x * y = y * x$ for all $x, y \in G$. $(\mathbb{Z}, +)$, $(\mathbb{Z}/N\mathbb{Z})^*$ are abelian groups, $\mathbf{U}(n)$ is not, when $n > 1$.

An abelian group is *cyclic* if it is generated by a single element.

A *ring* is a set R with two binary operations, often denoted $+$ and $*$, such that 1) $(R, +)$ is an abelian group, 2) $*$ is associative and has an identity, often denoted 1 or 1_R, 3) (compatibility) for all $x, y, z \in R$, $(x + y) * z = x * z + y * z$ and $z * (x + y) = z * x + z * y$.

Examples $(\mathbb{Z}, +, *)$, $(\mathbb{Z}/N\mathbb{Z}, +, *)$ (where both operations are inherited from the operations on $\mathbb{Z}$), $\mathbb{Z}[x]$: the polynomials in one variable with integer coefficients.

Let G, H be groups. A map $f : G \to H$ is a *group homomorphism* if $f(x * y) = f(x) * f(y)$ (where the first $*$ is in G and the second in H) for all $x, y \in G$. Let R, S be rings. A map $f : R \to S$ is a *ring homomorphism* if $f(1_R) = 1_S$, $f(x + y) = f(x) + f(y)$ and $f(x * y) = f(x) * f(y)$ for all $x, y \in R$.

Chapter 4

Classical information theory

We have been referring to the classical unit of information as a *bit* (short for "binary digit"), an element of $\{0, 1\}$. The discovery/invention of the bit by Tukey and its development by Shannon [**Sha48**] was one of the great scientific achievements of the twentieth century, as it changed the way we view information, giving it an abstract formalism that is discussed in this chapter. Instead of reading this chapter, we suggest just reading Shannon's classic article, as it is extremely well written, with carefully chosen examples.

The basic question is: Given a physical channel (e.g., telegraph wire), what is the maximal rate of transmission of information, tolerating a small amount of error? I begin with toy examples, leading up to Shannon's two fundamental theorems on channel capacity.

4.1. Data compression: noiseless channels

4.1.1. A toy problem. (Following [**BCHW16**]) A source emits symbols x from an alphabet $\mathcal{X}$ that we want to store efficiently so we try to encode x in a small number of bits, to say $y \in \mathcal{Y}$ in a way that we can decode it later to recover x.

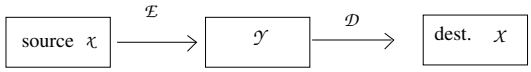

Figure 4.1.1. Message from source encoded into bits then decoded

The symbols from $\mathcal{X}$ may be emitted with varying frequencies. Let $p = P_{\mathcal{X}}$ denote the associated probability distribution. We want to determine the minimum possible size for $\mathcal{Y}$. Since we are dealing in bits, it will be convenient to use the logarithms of cardinalities, so define $\mathrm{Cap}(P_{\mathcal{X}}) := \min \log |\mathcal{Y}|$.

Consider the case $\mathcal{X} = \{a, b, c, d\}$ where $p(a) = 0.1$, $p(b) = 0$, $p(c) = 0.4$ and $p(d) = 0.5$. We can clearly get away with $\mathcal{Y}$ having cardinality 3, e.g., for the encoder, send a, b to 1, c to 2 and d to 3, then for the decoder, send 1 to a, 2 to c and 3 to d. In general, we can always throw away symbols with probability zero. On the other hand, we cannot map two distinct symbols that do occur to the same symbol, as there would be no way to distinguish them when decoding. Thus $\mathrm{Cap}(p) = \log \mathrm{supp}(p)$, where $\mathrm{supp}(p) = \#\{x \in \mathcal{X} \mid \Pr(x) > 0\}$.

Now say we are willing to tolerate a small error. In addition to throwing out the symbols that do not appear, we may also discard the largest set of symbols whose total probability is smaller than ϵ. Call the corresponding quantity $\mathrm{Cap}^{\epsilon}(p)$. In the example above, if we take $\epsilon > 0.1$, we can lower storage cost, taking $|\mathcal{Y}| = 2$.

Recall that a probability distribution p on $\mathcal{X}$ must satisfy $\sum_{x\in\mathcal{X}} \Pr(x) = 1$. We relax this to *non-normalized* probability distributions, q, where $q(x) \geq 0$ for all $x \in \mathcal{X}$ and $\sum_{x\in\mathcal{X}} q(x) \leq 1$. Define $\mathrm{Cap}^{\epsilon}(p) = \min \log \mathrm{supp}(q)$, where the min is taken over all non-normalized probability distributions q satisfying $q(x) \leq p(x)$ for all $x \in \mathcal{X}$ and $\sum_{x\in\mathcal{X}} q(x) \geq 1 - \epsilon$.

4.1.2. The case of interest. Now say the transmission is not a single symbol, but a string of n symbols, so we seek an encoder $\mathcal{E} : \mathcal{X}^n \to \mathcal{Y}(n)$, where $\mathcal{Y}(n)$ is a set that varies with n, and decoder $\mathcal{D} : \mathcal{Y}(n) \to \mathcal{X}^n$, and we want to minimize $|\mathcal{Y}(n)|$, with a tolerance of error that goes to zero for n going to infinity. In practice one wants to send information through a communication channel (e.g. telegraph wire). The channel can only send a limited number of bits per second, and we want to maximize the amount of information we can send per second. Define $\mathrm{Rate}(p) := \lim_{\epsilon\to 0} \lim_{n\to\infty} \frac{1}{n} \mathrm{Cap}^{\epsilon}(p^n)$. It now remains to compute the right hand side of the expression. To be able to do so, we review some probability.

4.1.3. Expectation and the law of large numbers. The *expectation* (or *average*) of a random variable on a countable set $\mathcal{X}$ equipped with a probability distribution p is

$$E[X] = \sum_{a_j\in\mathcal{X}} X(a_j)p(a_j). \tag{4.1.1}$$

Random variables X, Y are said to be *independent* if for all x, y, $p_{X,Y}(x, y) = p_X(x)p_Y(y)$. They are *identically distributed* if they define the same probability distributions. We write $X_1, \ldots, X_n$ are *iid* if they are independent and identically distributed.

For example, if $\mathcal{X} = \{H, T\}$ are the possible outcomes of flipping a biased coin which lands heads (H) with probability p and tails with probability $1 - p$, and $X(H) = 1$, $X(T) = -1$, then $E[X] = 2p - 1$, which is zero if the coin is fair. We will often be concerned with repeating an experiment many times. A typical situation is to define random variables X_j where X_j is 1 if the outcome of the j-th toss is heads and $X_j = -1$ if the outcome of the j-th toss is tails. Then the X_j are iid.

Note that $E[X] \in [-\infty, \infty]$. The *law of large numbers* implies that the name "expectation" is reasonable, that is, if one makes repeated experiments (e.g., as with the coin flips above) and averages the outcomes, the averages limit towards the expectation.

More precisely, the *weak law of large numbers* states that for $X, X_1, X_2, ...$ independent identically distributed random variables, and for any $\epsilon > 0$,

$$\lim_{n\to\infty} \Pr\left(\left|\frac{X_1 + \cdots + X_n}{n} - E[X]\right| \geq \epsilon\right) = 0. \tag{4.1.2}$$

We may rephrase this as for all $\epsilon, \delta > 0$, there exists an n_0 such that for all $n \geq n_0$

$$\Pr\left(\left|\frac{X_1 + \cdots + X_n}{n} - E[X]\right| \geq \epsilon\right) < \delta. \tag{4.1.3}$$

The *strong law of large numbers* states moreover that

$$\Pr\left(\lim_{n\to\infty} \frac{X_1 + \cdots + X_n}{n} = E[X]\right) = 1. \tag{4.1.4}$$

Here and throughout $\Pr(Z)$ denotes the probability of the event Z occurring with respect to some understood distribution.

However, individual outcomes can be far from the expectation. A first measurement of how far one can expect to be from the expectation is the *variance*: The variance of X is

$$\mathrm{var}(X) = E[(X - E[X])^2] \tag{4.1.5}$$

$$= E[X^2] - E[X]^2 \tag{4.1.6}$$

Exercise 4.1.1: Verify that (4.1.6)=(4.1.5).

Often one deals with the square-root of the variance, called the *standard deviation*, $\sigma(X) = \sqrt{\mathrm{var}(X)}$.

4.1.4. Estimating $|\mathcal{Y}(n)|$. Say $|\mathcal{X}| = d$. Define a map $wt : \mathcal{X}^n \to \mathbb{R}^d$ by $\overline{x}^n \mapsto (c_1, \ldots, c_d)$, where c_j is the number of times j occurs in the string. Then $E[wt(\overline{x}^n)] = (np_1, \ldots, np_d)$. The weak law of large numbers (4.1.2) states that for any $\epsilon > 0$,

$$\lim_{n\to\infty} \Pr[||\frac{1}{n}(wt(\overline{x}^n) - E[wt(\overline{x}^n))]||_1 > \epsilon] = 0$$

where for $f : \mathcal{Z} \to \mathbb{R}^d$, define $||f||_1 = \sum_{z\in\mathcal{Z}} |f(z)|$. In our case, $\mathcal{Z} = \mathcal{X}^n$.

To minimize $|\mathcal{Y}(n)|$ while having error going to zero as $n \to \infty$, we throw out all strings $\overline{x}^n$ with $||\frac{1}{n}(wt(\overline{x}^n) - E[wt(\overline{x}^n))]||_1 > \epsilon$, so $\mathcal{Y}(n)$ will have size

$$\begin{aligned} |\mathcal{Y}(n)| &= \#\{\overline{x}^n \mid ||\frac{1}{n}(wt(\overline{x}^n) - E[wt(\overline{x}^n))]||_1 < \epsilon\} \\ &= \sum_{\substack{\text{weights } \overline{c}| \\ \sum c_i = n \text{ and } ||\frac{1}{n}(\overline{c} - E[wt(\overline{x}^n))]||_1 < \epsilon}} \binom{n}{\overline{c}}. \end{aligned}$$

If ϵ is small, the multinomial coefficients appearing will all be very close to

$$\binom{n}{np_1, \ldots, np_d} \tag{4.1.7}$$

and the number of $\overline{x}^n$ of a given weight grows like a polynomial in n, so for what follows, we can take the crude approximation (which will be justified later)

$$|\mathcal{Y}(n)| \leq poly(n)\binom{n}{np_1, \ldots, np_d}. \tag{4.1.8}$$

We now need to approximate (4.1.7)

4.1.5. Approximating binomial coefficients. Let x_j be iid random variables. The string $x_1 \cdots x_n =: \overline{x}^n$ is iid. Say $\mathcal{X} = \{1, \ldots, d\}$ with $\Pr(j) = p_j$. The probability of any given string occurring depends only on the number of 1's 2's etc.. in the string and not on their order. A string with c_j j's occurs with probability $p_1^{c_1} \cdots p_d^{c_d}$. (Note that $c_1 + \cdots + c_d = n$.) The number of strings with this probability is

$$\binom{n}{c_1, \ldots, c_d} := \frac{n!}{c_1! \cdots c_d!}$$

and we will need to estimate this quantity.

Recall that Stirling's formula (3.5.4) implies $\log(n!) = n\log(n) - \log(e)n + O(\log(n))$.

In particular, for $0 < \beta < 1$ such that $\beta n \in \mathbb{Z}$,

$$\log \binom{n}{\beta n} = \log \frac{n!}{(\beta n)!((1-\beta)n)!} \tag{4.1.9}$$
$$= n[-\beta \log(\beta) - (1-\beta)\log(1-\beta)] + O(\log(n))$$

Let $H(\beta) = -\beta \log(\beta) - (1-\beta)\log(1-\beta)$ and more generally:

Definition 4.1.2. For a probability distribution $\overline{p} = (p_1, \ldots, p_d)$, the *Shannon entropy* of $\overline{p}$. is

$$H(\overline{p}) = -\sum_{i=1}^{d} p_i \log(p_i).$$

Shannon entropy will play a central role in information theory.

Exercise 4.1.3: Show that for the multinomial coefficient

$$\binom{n}{p_1 n, \ldots, p_d n} = \frac{n!}{(p_1 n)! \cdots (p_d n)!},$$

where $p_1 + \cdots p_n = 1$, we have

$$\log \binom{n}{p_1 n, \ldots, p_d n} = nH(\overline{p}) + O(\log(n)). \tag{4.1.10}$$

4.1.6. Shannon's noiseless channel theorem, first version. When we take logarithms, the right hand side of (4.1.8) becomes $nH(\overline{p}) + O(\log(n))$. Thus

$$\frac{1}{n} \log |\mathcal{Y}(n)| \leq H(\overline{p}) + o(1)$$

and we see $\mathrm{Rate}(\overline{p}) \leq H(\overline{p})$.

Theorem 4.1.4. [**Sha48**] $\mathrm{Rate}(\overline{p}) = H(\overline{p})$

We give a proof in §4.3. Since $H(\overline{p})$ is a fundamental quantity, we first discuss some of its properties.

4.2. Entropy, i.e., uncertainty

The entropy is a measure of the uncertainty of an outcome. For example, consider the case $d = 2$, with probabilities $p, 1-p$. The graph of $H(p)$ is

graph here

Note that it has a unique maximum when $p = \frac{1}{2}$ (situation of maximal uncertainty) and is 0 in the two certain cases.

We arrived at H while approximating logs of multinomial coefficients. Say we had not yet discovered it but were looking for a function h on probability distributions with the following properties:

(1) h is continuous in the p_i.

(2) If $p_i = \frac{1}{d}$ for all i and we increase d, h is monotonically increasing.

(3) h is additive with respect to breaking an event into a sequence of conditional events: Write $\mathcal{X} = A_1 \sqcup \cdots \sqcup A_k$ and assume $p(A_j) > 0$ for all j. Write $\overline{p}_A = (p(A_1), \ldots, p(A_k))$ Then $h(\overline{p}_{\mathcal{X}}) = h(\overline{p}_A) + \sum_{i=1}^{k} p(A_i) h(\overline{p}_{\mathcal{X}|A_i})$.

For example, if $\mathcal{X} = \{1, 2, 3\}$, with $p_1 = \frac{1}{2}$, $p_2 = \frac{1}{3}$, $p_3 = \frac{1}{6}$, we can choose one of the three at the outset, so we have $H(\frac{1}{2}, \frac{1}{3}, \frac{1}{6})$, or we can first decide between the sets $\{1\}$ and $\{2, 3\}$, both of which have probability $\frac{1}{2}$ and then if we choose the second, decide between 2 and 3, the first with probability $\frac{2}{3}$, the second with probability $\frac{1}{3}$. Thus we require the equality $H(\frac{1}{2}, \frac{1}{3}, \frac{1}{6}) = H(\frac{1}{2}, \frac{1}{2}) + \frac{1}{2}(0) + \frac{1}{2}H(\frac{2}{3}, \frac{1}{3})$.

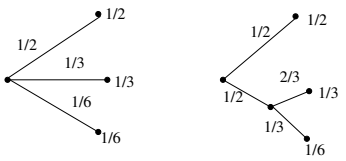

Figure 4.2.1. a choice of three outcomes viewed as a choice of two followed by a second choice of two

Theorem 4.2.1. [**Sha48**] *The only function h satisfying 1,2,3 is, up to a constant, the entropy.*

Proof. Set $A(d) = h(\frac{1}{d}, \ldots, \frac{1}{d})$. By 3, $A(s^m) = mA(s)$ (recall that $\log(s^m) = m \log(s)$).

To see this, consider the example $s = 2, m = 3$:

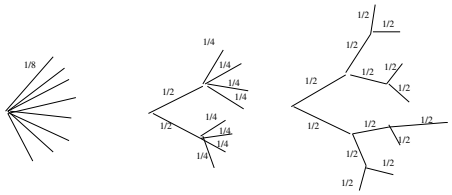

Figure 4.2.2. repeated application of (3) gives $A((\frac{1}{2})^3) = A(\frac{1}{2}) + \frac{1}{2}A(\frac{1}{4}) + \frac{1}{2}A(\frac{1}{4}) = 3A(\frac{1}{2})$

Exercise 4.2.2: Show that $A(s) = C \log(s)$ for some constant C.

Now say we have a choice of D equally likely outcomes, which we break up as $D = \sum_{i=1}^{d} d_i$. Write $p_i = \frac{d_i}{D}$, and assume the d_i are natural numbers.

By property (3), $C\log(D) = H(p_1,\ldots,p_d) + C\sum_i p_i \log d_i$, i.e.,

$$\begin{aligned} H(p_1,\ldots,p_d) &= -C[\sum_i p_i \log(d_i) - \log(D)] \\ &= -C[\sum_i p_i \log(d_i) - \sum_i p_i \log(D)] \\ &= -C\sum_i p_i \log(\frac{d_i}{D}) \\ &= -C\sum_i p_i \log(p_i). \end{aligned}$$

Finally use property 1 to extend by continuity to all probability distributions. □

Here are some further properties of entropy that are easily verified:

(1) $0 \le H(\overline{p}) \le \log(d)$ with $H(\overline{p}) = 0$ if and only if the outcome is certain, i.e., $\overline{p} = (0,\ldots,0,1,0,\ldots,0)$, and with $H(\overline{p}) = \log(d)$ if and only if $\overline{p} = (\frac{1}{d},\ldots,\frac{1}{d})$.

(2) Say our space is $\mathcal{X}\times\mathcal{Y}$, so $H(\overline{p}_{\mathcal{X}\times\mathcal{Y}}) = -\sum_{i,j} p_{\mathcal{X}\times\mathcal{Y}}(i,j)\log p_{\mathcal{X}\times\mathcal{Y}}(i,j)$. Then we can recover $H(\overline{p}_{\mathcal{X}})$ as:

$$H(\overline{p}_{\mathcal{X}}) = -\sum_{i,j} p_{\mathcal{X}\times\mathcal{Y}}(i,j)\log(\sum_k p_{\mathcal{X}\times\mathcal{Y}}(i,k)).$$

In particular, $H(\overline{p}_{\mathcal{X}\times\mathcal{Y}}) \le H(\overline{p}_{\mathcal{X}}) + H(\overline{p}_{\mathcal{Y}})$, with equality if and only if $\mathcal{X},\mathcal{Y}$ are independent, i.e., $p_{\mathcal{X}\times\mathcal{Y}}(i,j) = p_{\mathcal{X}}(i)p_{\mathcal{Y}}(j)$. (The uncertainty of $p_{\mathcal{X}\times\mathcal{Y}}$ is at most the sum of the uncertainties of the marginal distributions $p_{\mathcal{X}}$ and $p_{\mathcal{Y}}$.)

(3) A modification of the p_i towards equalization increases H. More precisely:

Exercise 4.2.3: Let A be a doubly stochastic matrix, i.e., the entries of A are non-negative and the column and row sums of A are one. Show that $H(A\overline{p}) \ge H(\overline{p})$ and unless A is a permutation matrix, that there exists $\overline{p}$ where the inequality is strict.

4.3. Shannon's noiseless channel theorem

Theorem 4.3.1. [**Sha48**] *Given $\epsilon > 0, \delta > 0$, there exists n_0 such that for all $n \ge n_0$, there is a decomposition $\mathcal{X}^{\times n} = \mathcal{X}^n_{\epsilon-typ} \sqcup \mathcal{X}^n_{\delta-small}$ where*

(1) $\Pr(\mathcal{X}^n_{\delta-small}) < \delta$, *and*

(2) $\forall \overline{x} \in \mathcal{X}^n_{\epsilon-typ}$,

$$|\frac{1}{n}\log(\Pr(\overline{x})) - H(\overline{p})| < \epsilon,$$

(3)
$$(1-\delta)2^{n(H(\overline{p})-\epsilon)} \leq |\mathcal{X}^n_{\epsilon-typ}| \leq 2^{n(H(\overline{p})-\epsilon)}.$$

The set $\mathcal{X}^n_{\epsilon-typ}$ will play the role of $\mathcal{Y}(n)$ from §4.1. Informally, the probability of not being ϵ-typical is small, if ϵ-typical, the probability is close to the expectation, and if the entropy is large, most sequences are ϵ-typical, and if it is small, there are few such.

Proof. The strong law of large numbers (4.1.4) means that given iid random variables X_j, for all $\epsilon, \delta > 0$, there exists n_0 such that for all $n \geq n_0$ such that
$$P\left(|\ \frac{X_1 + \cdots + X_n}{n} - E[X]\ | > \epsilon\right) < \delta$$
Let $\mathcal{X}^n_{\delta-small}$ be all events $\overline{x} = (x_1, \ldots, x_n)$ where $|\frac{1}{n}(x_1 + \cdots + x_n) - E[X]| \geq \epsilon$, and $\mathcal{X}^n_{\epsilon-typ}$ the events $\overline{x}$ with $|\frac{1}{n}(x_1 + \cdots + x_n) - E[X]| < \epsilon$. In our case $X = -\log(p(\overline{x}))$ and $E[-\log(p(\overline{x}))] = H(\overline{p})$. Note that (3) is a quantitative version of (4.1.8). To prove it, note that the second condition implies that for all $\overline{x} \in \mathcal{X}^n_{\epsilon-typ}$, and $n > \frac{1}{\epsilon}$,

$$\Pr(\overline{x}) \leq 2^{-n(H(\overline{p})-\epsilon)}. \tag{4.3.1}$$

On the other hand
$$1 - \delta \leq \Pr(\mathcal{X}^n_{\epsilon-typ}) \leq 1$$
i.e.,
$$1 - \delta \leq \sum_{\overline{x} \in \mathcal{X}^n_{\epsilon-typ}} \Pr(\overline{x}) \leq 1$$
so plugging in (4.3.1) we conclude.

□

It will be useful to give two variants of Theorem 4.3.1.

Theorem 4.3.2. [**Sha48**] *Fix $q \in (0,1)$. Let $\mathrm{num}(q)$ denote the minimum cardinality of a subset S of $\mathcal{X}^{\times n}$ such that $\Pr(S) \geq q$. Then*
$$\lim_{n\to\infty} \frac{\log(\mathrm{num}(q))}{n} = H(\overline{p}).$$

Note that the right hand side is independent of q, thus for large n, the ratio is nearly independent of q and n. Theorem 4.3.2 will be a consequence of:

Theorem 4.3.3. [**Sha48**] *Let $R < H(\overline{p})$ and let $S(n) \subset \mathcal{X}^{\times n}$ be a sequence of subsets with $|S(n)| \leq 2^{nR}$, i.e., $\frac{1}{n}\log|S(n)| \leq R$. Then for any $\eta > 0$, there exists n_0 such that for all $n > n_0$, $\Pr(S(n)) < \eta$.*

In other words, *any* subset of size less than capacity can only accumulate a small amount of the probability.

Proof of Theorem 4.3.3. Write $S(n) = S(n)_{\delta-small} \sqcup S(n)_{\epsilon-typ}$, where $S(n)_{\delta-small} = S(n) \cap \mathcal{X}^n_{\delta-small}$ and $S(n)_{\epsilon-typ} = S(n) \cap \mathcal{X}^n_{\epsilon-typ}$. Then $|S(n)_{\epsilon-typ}| < |S(n)| < 2^{nR}$. By (4.3.1) each element of $S(n)_{\epsilon-typ}$ has probability at most $2^{-n(H(\overline{p})-\epsilon)}$, so $\Pr(S(n)_{\epsilon-typ}) \leq 2^{-n(H(\overline{p})-\epsilon)}2^{nR}$. So just take, $\epsilon < H(\overline{p}) - R$, and n_0, δ such that $2^{-n_0(H(\overline{p})-R-\epsilon)} + \delta < \eta$. □

Theorem 4.3.4. [**Sha48**] *Let a source $\mathcal{X}$ have entropy $H(\overline{p})$ (bits per symbol) and let a channel have capacity C (bits per second). Then for any $\epsilon > 0$, it is possible to encode the output of the source to transmit at the rate $\frac{C}{H(\overline{p})} - \epsilon$ symbols/sec., and it is not possible to reliably transmit at an average rate greater than $\frac{C}{H(\overline{p})}$.*

The idea of the proof is clear: just transmit all the ϵ-typical sequences and discard the others.

4.4. Transmission over noisy channels

Say we transmit symbols x and receive symbols y over a channel subject to noise, so we may or may not have $y = x$. Intuitively, if the noise is small, with some redundancy we should be able to communicate accurate messages most of the time. Let Rate denote the maximal possible rate of transmission. In a noiseless channel this is just $H(p_{\mathcal{X}})$, but now we must subtract off something to account for the uncertainty that, upon receiving y, that it was the signal sent. This something will be the *conditional entropy*, $H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})$ defined below. The punch line will be:

Given a channel with noise and symbols sent according to $p_{\mathcal{X}}$, and noise given by $p_{\mathcal{Y}}$, the maximum rate of transmission is $H(\overline{p}_{\mathcal{X}}) - H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})$.

Given a channel, with noise its maximal capacity for transmission is $\max_{q_{\mathcal{X}}} (H(q_{\mathcal{X}}) - H(q_{\mathcal{X}}|p_{\mathcal{Y}}))$.

Note that the second result is trivial in the noiseless case - one just takes a uniform distribution for the symbols. What is going on here is that if some symbols are more susceptible to corruption than others, the uniform distribution will no longer be optimal.

4.4.1. Conditional entropy. Recall the conditional probability of i occurring given knowledge that j occurs (assuming $\Pr(j) > 0$): $\Pr_{\mathcal{X}|\mathcal{Y}}(i|j) = \frac{\Pr_{\mathcal{X},\mathcal{Y}}(i,j)}{\Pr_{\mathcal{Y}}(j)}$ (also recall $\Pr_{\mathcal{Y}}(j) = \sum_i \Pr_{\mathcal{X},\mathcal{Y}}(i,j)$). Define the conditional entropy

$$H(\overline{p}_{\mathcal{Y}}|\overline{p}_{\mathcal{X}}) := -\sum_{i,j} \Pr_{\mathcal{X},\mathcal{Y}}(i,j) \log \Pr_{\mathcal{Y}|\mathcal{X}}(j|i).$$

Note that

$$H(\overline{p}_{\mathcal{Y}}|\overline{p}_{\mathcal{X}}) = H(\overline{p}_{\mathcal{X},\mathcal{Y}}) - H(\overline{p}_{\mathcal{X}}) \tag{4.4.1}$$

or equivalently $H(\overline{p}_{\mathcal{X},\mathcal{Y}}) = H(\overline{p}_{\mathcal{X}}) + H(\overline{p}_{\mathcal{Y}}|\overline{p}_{\mathcal{X}})$, the uncertainty of $p_{\mathcal{X},\mathcal{Y}}$ is the uncertainty of $p_{\mathcal{X}}$ plus the uncertainty of $p_{\mathcal{Y}}$ given $p_{\mathcal{X}}$.

In particular, we have $H(\overline{p}_{\mathcal{X}})+H(\overline{p}_{\mathcal{Y}}) \geq H(\overline{p}_{\mathcal{X},\mathcal{Y}}) = H(\overline{p}_{\mathcal{X}})+H(\overline{p}_{\mathcal{Y}}|\overline{p}_{\mathcal{X}})$, i.e.,

$$H(\overline{p}_{\mathcal{Y}}) \geq H(\overline{p}_{\mathcal{Y}}|\overline{p}_{\mathcal{X}}),$$

i.e., with extra knowledge, our uncertainty about $p_{\mathcal{Y}}$ cannot increase, and decreases unless $p_{\mathcal{X}}$ and $p_{\mathcal{Y}}$ are independent).

4.4.2. Examples. We first give an example where $p_{\mathcal{X}}$ is fixed. Say 0's and 1's are transmitted with each having a probability of error .01, so $H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}}) = -[.99\log(.99) + .01\log(.01)] \sim .81$ bits/symbol, and we can transmit 1000 bits/second.

Exercise 4.4.1: Verify the above assertion about $H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})$.

The above discussion predicts a transmission rate of $1000 - 81 = 919$ bits/second. If the probability of error goes up to $\frac{1}{2}$, then $H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}}) = 1$ and our discussion predicts a rate of $1000 - 1000 = 0$ bits/second, which agrees with our intuition that what is received is just noise.

Here is an example where we choose $p_{\mathcal{X}}$ to optimize capacity. Now say $\mathcal{X} = \{a, b, c\}$ where the symbol a is never effected by noise, b has probability p being transmitted correctly and $1 - p$ of being flipped to c, and c has probability p being transmitted correctly and $1 - p$ of being flipped to b.

Let p_a be the probability that a is transmitted, let p_b be the probability that b is transmitted and p_c the probability that c is transmitted: we get to choose these. Given the symmetry of the situation, we should set $p_b = p_c =: p_{b,c}$.

Thus

$$\begin{aligned} H(\overline{p}_{\mathcal{X}}) &= -p_a \log p_a - 2p_{b,c} \log p_{b,c} \\ H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}}) &= 2p_{b,c} H(p, 1-p) \end{aligned}$$

We want to maximize $H(\overline{p}_{\mathcal{X}}) - H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})$ by a good choice of $p_a, p_{b,c}$, subject to the constraint that $p_a + 2p_{b,c} = 1$. We have

$$H(\overline{p}_{\mathcal{X}}) - H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}}) = -p_a \log p_a - 2p_{b,c} \log p_{b,c} - 2p_{b,c} H(p, 1-p).$$

Viewed as a function of $p_a, p_{b,c}$, we have a standard maximization problem from calculus.

Exercise 4.4.2: Differentiating and imposing the constraint, show that it is optimal to take

$$p_a = \frac{e^{H(p,1-p)}}{e^{H(p,1-p)} + 2}, \quad p_{b,c} = \frac{1}{e^{H(p,1-p)} + 2}. \tag{4.4.2}$$

Using (4.4.2), we obtain

$$H(\overline{p}_{\mathcal{X}}) - H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}}) = \log \frac{e^{H(p,1-p)}+2}{e^{H(p,1-p)}}.$$

For example, when $p = 1$, i.e., no errors, we obtain $\log(3)$, as we have a noiseless channel. When $p = \frac{1}{2}$, we obtain $\log(2)$, as the second and third symbols are indistinguishable, so we essentially have two channels. In general, the capacity is between these two, and the first channel used somewhere between the same amount and twice as often as the other two channels.

4.4.3. Warm up: Communication with the help of a correction channel. Before giving the proof of Shannon's theorem, here is a warm-up problem giving intuition into the conditional entropy $H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})$, which Shannon calls the "equivocation", as the amount of extra information that must be supplied to correct errors on a noisy channel.

Consider the following picture ***

In the transmission from the source to the receiver, an observer is allowed to see what is transmitted and what the receiver gets. The observer is then allowed to send correction data to allow the receiver to correct the message. The question is how much information must the observer send to the receiver to enable correction, i.e., how much capacity does the correction channel need to correct errors (assume the correction channel is not subject to noise).

Theorem 4.4.3. [**Sha48**] *If* Cap(*correction channel*) $\geq H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})$*, then in the scheme above, all but an arbitrarily small fraction of the errors can be corrected.*

If Cap(*correction channel*) $< H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})$*, then in the scheme above, reliable error correction is not possible.*

Proof. Assume we are in the first case, say $\overline{y}^n$ is received when $\overline{x}^n$ is sent over t seconds. By the same argument as in Theorem 4.3.1(3) applied to $\#\{\overline{x}^n \mid \Pr_{\mathcal{X}|\mathcal{Y}}(\overline{x}^n, \overline{y}^n) \geq 1-\delta\}$, there exist on the order of $2^{tH(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})}$ possible $\overline{x}^n$'s that could have reasonably produced $\overline{y}^n$. Thus we need to send $tH(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})$ bits each t seconds, which is possible with an ϵ-tolerance of error on a channel of capacity $H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})$.

To prepare for the second case:

Exercise 4.4.4: For any random variables x, y, z, determining probability distributions $p_{\mathcal{X}}, p_{\mathcal{Y}}, p_{\mathcal{Z}}$, show that

$$H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}}, \overline{p}_{\mathcal{Z}}) \geq H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}}) - H(\overline{p}_{\mathcal{Z}}|\overline{p}_{\mathcal{Y}}) \geq H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}}) - H(\overline{p}_{\mathcal{Z}}).$$

Now let x be the output of the source, y the received signal, and z the signal sent over the correction channel. We see $H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}}, \overline{p}_{\mathcal{Z}}) > 0$ so we cannot recover x reliably by the noiseless theorem. □

Note that we have three interpretations of the rate:

$$\begin{aligned}\text{Rate} &= H(\overline{p}_{\mathcal{X}}) - H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}}) \text{ the information sent minus the uncertainty of what sent}\\ &= H(\overline{p}_{\mathcal{Y}}) - H(\overline{p}_{\mathcal{Y}}|\overline{p}_{\mathcal{X}}) \text{ the information received minus the noise}\\ &= H(\overline{p}_{\mathcal{X}}) + H(\overline{p}_{\mathcal{Y}}) - H(\overline{p}_{\mathcal{X}\times\mathcal{Y}}) \text{ the sum of the information sent and received minus the joint entropy, essentially the bits/sec. common to } x, y\end{aligned}$$

The Rate is also called the *mutual information*.

4.4.4. Capacity of a noisy channel. Define the *capacity* of a noisy channel to be the maximum rate over all possible probability distributions on the source:

$$\text{Cap} := \max_{q_{\mathcal{X}}} \left(H(q_{\mathcal{X}}) - H(q_{\mathcal{X}}|p_{\mathcal{Y}})\right).$$

Theorem 4.4.5. [**Sha48**] *Let a discrete channel have capacity* Cap *and entropy per second* H*. If* $H <$ Cap*, then there exists an encoding* $\overline{p}$ *of the source such that information can be transmitted over the channel with an arbitrarily small frequency of errors* $H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})$ ***check**. If* $H >$ Cap*, then there exists an encoding* $\overline{p}$ *such that the equivocation is less than* $H - \text{Cap} + \epsilon$ *for any* $\epsilon > 0$*, and there does not exist any* $\overline{p}$ *with equivocation less than* $H -$ Cap*.*

The basic idea is the same as the noiseless case, however there is a novel feature that now occurs frequently in complexity theory arguments - that instead of producing an algorithm to find the efficient encoding, Shannon showed that a random choice of encoding will work. More on this after the proof.

Proof. Split the transmitter and receiver $\mathcal{X}^{\times n}$ and $\mathcal{Y}^{\times n}$ into the union of ϵ-typical δ-small subsets. For a high probability message $\overline{y}^n$ in $\mathcal{Y}^n_{\epsilon-typ}$, there are roughly $2^{H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})t}$ "reasonable"$\overline{x}^n$'s, i.e., elements of $\mathcal{X}^n_{\epsilon-typ}$, that could have been sent for $\overline{y}^n$ to be received, i.e., roughly $2^{H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})t}$ elements of $\mathcal{X}^n_{\epsilon-typ}$. On the other hand, if some $\overline{x}^n \in \mathcal{X}^n_{\epsilon-typ}$ is sent, there are about $2^{H(\overline{p}_{\mathcal{Y}}|\overline{p}_{\mathcal{X}})t}$ elements of $\mathcal{Y}^n_{\epsilon-typ}$ that could be received.

pic*

Every t seconds we have 2^{tR} high probability messages. Say $\overline{y}^n$ is observed, we want to know the probability that more than one message in $\mathcal{X}^n_{\epsilon-typ}$ could arrive as $\overline{y}^n$, based on our choice of distribution.

Take a *random* encoding of $\mathcal{X}^n_{\epsilon-typ}$, so we have 2^{tR} messages distributed at random among $2^{tH(p_{\mathcal{X}})}$ points. The probability of a particular message

being received as $\overline{y}^n$ is $\frac{2^{tR}}{2^{tH(p_{\mathcal{X}})}} = 2^{t(R-H(p_{\mathcal{X}}))}$. The probability that no $\overline{x}^n \in \mathcal{X}^n_{\epsilon-typ}$ (other than $\overline{y}^n$) is sent to $\overline{y}^n$ is

$$[1 - 2^{t(R-H(p_{\mathcal{X}}))}]^{2^{tH(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})}}. \tag{4.4.3}$$

Now say $R <$ Cap, and write $R - H(p_{\mathcal{X}}) = -H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}}) - \eta$ for some $\eta > 0$, so (4.4.3) becomes

$$[1 - 2^{-tH(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})-t\eta}]^{2^{tH(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})}}.$$

This limits to 1 as $t \to \infty$.

To prove the second assertion, just send Cap bits/sec. of x's generated and throw away the rest. This gives $H(\overline{p}_{\mathcal{X}}|\overline{p}_{\mathcal{Y}})$ equal to $H(p_{\mathcal{X}}) -$ Cap plus the ϵ from the first case.

□

Exercise 4.4.6: Verify that, for $H \geq 0$, $\eta > 0$, $\lim_{t\to\infty}[1-2^{-tH-t\eta}]^{2^{tH}} = 1$. ◎

Note that the phrase "Take a random encoding" is not constructive, as it gives no recipe how to do so.

After presenting the proof, Shannon remarks: "An attempt to obtain a good approximation to ideal coding by following the method of the proof is generally impractical. ... Probably this is no accident but is related to the difficulty of giving an explicit construction for a good approximation to a random sequence". To our knowledge, this is the first time that the difficulty of "finding hay in a haystack" (phrase due to Howard Karloff) is mentioned in print. This problem is central to complexity: for example, Valiant's algebraic version of $\mathbf{P} \neq \mathbf{NP}$ can be phrased as the problem of finding a sequence of explicit polynomials that are difficult to compute, while it is known that a random sequence is indeed difficult to compute. (According to A. Wigderson, the difficulty of writing down random objects was explicitly discussed by Erdös, in the context of random graphs, at least as early as 1947, in relation to his seminar paper [**Erd47**]. This paper, along with [**Sha48**] gave rise to the now ubiquitous probabilistic method in complexity theory.

Hints and Answers to Selected Exercises

Chapter 1.

1.2.3 First do the case that one of the $n \times n$ matrices has distinct eigenvalues, then do the general case.

1.2.9 see Figure 1.2.1.

1.2.11 Write

$$\begin{pmatrix} DFT_M & \Delta_M DFT_M \\ DFT_M & -\Delta_M DFT_M \end{pmatrix} = \begin{pmatrix} \mathrm{Id}_M & \Delta_M \\ -\mathrm{Id}_M & -\Delta_M \end{pmatrix} \begin{pmatrix} DFT_M & 0 \\ 0 & DFT_M \end{pmatrix}.$$

Also note that a $k \times k$ permutation matrix has k nonzero entries, and the product of two permutation matrices is a permutation matrix.

?? Choose the first four rows and last four columns. One obtains a 4×4 matrix M' and the associated tensor T', so $\mathbf{R}(T_{aft,3}) \geq 8 + \mathbf{R}(T')$. Iterating the method twice yields $\mathbf{R}(T_{aft,3}) \geq 8 + 4 + 2 + 1 = 15$.

Chapter 2.

2.1.7 Write $A_1 := A|_{v_1^\perp}$ (the restriction of A to $v_1^\perp$). Note that A_1 must have an eigenvector and that it is unitary. Repeat $n - 1$ times.

2.1.11 Consider $\langle v_i | v_j \rangle = \langle Av_i | Av_j \rangle$.

2.2.11 It is sufficient to work in bases, i.e., with matrices. First prove the case X is diagonal, then the case X is diagonalizable, then either write $X = X_s + X_n$ as the sum of a diagonlizable matrix and a nilpotent matrix or argue that the diagonalizable matrices is a dense open subset in the space of all matrices.

2.2.13 First consider the case X is diagonal, and use that the eigenvalues of a Hermitian matrix are real.

2.2.20 Differentiate $U(t)v(t)$ where $v(t)$ is an eigenvalue of $U(t)$, and $U(0) = \mathrm{Id}$.

2.2.22 It suffices to do the case $w = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

2.5.3 $\frac{7}{8}$.

Chapter 3.

3.1.4 Use induction: the case $x = 1$ is easy, and use exercise 3.1.3 to prove the induction step.

3.1.11 Consider the binomial coefficients in the expansion of $(x + a)^N$.

3.1.1 First do the case $k = 2^\ell$. Then show the general case by using the binary expansion of k.

3.5.1 Use Exercise 1.2.6.

3.5.5 Write $x = \lfloor x \rfloor + \{x\}$. If $\{x\} < \frac{1}{2}$, then $2x = 2\lfloor x \rfloor + \{2x\}$ and therefore $\lfloor 2x \rfloor = 2\lfloor x \rfloor$. If If $\{x\} \geq \frac{1}{2}$, then $\lfloor 2x \rfloor = \lfloor (2\lfloor x \rfloor) \rfloor + 1$.

3.5.6 Use (4.1.10).

Chapter 4.

4.4.6 Prove an upper and a lower bound for the quantity.

??

$$\begin{aligned} \Pr(\mathrm{span}\{\mathcal{M}_1, \mathcal{M}_2\}) &= 1 \\ \Pr(\mathcal{M}_1) &= |\alpha|^2 \\ \Pr(\mathcal{M}_2) &= \frac{1}{2}|\alpha + \beta|^2 \\ \Pr(\mathcal{M}_1 \cap \mathcal{M}_2) &= 0 \end{aligned}$$

?? Use that any $X \in \mathrm{End}(\mathcal{H})$ may be uniquely written as a sum of a Hermitian and an anti-Hermitian (i.e., i times a Hermitian) operator.

?? Use the Cauchy-Schwartz inequality, in the form $|\operatorname{trace}(AB\rho)|^2 \leq \operatorname{trace}((A\rho)^2)\operatorname{trace}((B\rho)^2)$.

?? Use the polar or singular value decomposition.

Chapter 5.

?? Note that $\mathit{transpose} \otimes \mathrm{Id}(\sum_{i,j} |ii\rangle\langle jj|) = \sum_{i,j} |ji\rangle\langle ij|$.

?? Use Theorem **??** and the Cauchy-Schwartz inequality.

?? Write $\rho_{AB} = \sum_{ij} \lambda_{i,j} |v_i\rangle\langle v_i| \otimes |w_j\rangle\langle w_j|$, the eigenbasis decomposition.

Bibliography

[Aar13] Scott Aaronson, *Quantum computing since Democritus*, Cambridge University Press, Cambridge, 2013. MR 3058839

[AB09] Sanjeev Arora and Boaz Barak, *Computational complexity*, Cambridge University Press, Cambridge, 2009, A modern approach. MR 2500087 (2010i:68001)

[Ad89] V. I. Arnol′ d, *Mathematical methods of classical mechanics*, Graduate Texts in Mathematics, vol. 60, Springer-Verlag, New York, [1989?], Translated from the 1974 Russian original by K. Vogtmann and A. Weinstein, Corrected reprint of the second (1989) edition. MR 1345386

[Adl78] Leonard Adleman, *Two theorems on random polynomial time*, 19th Annual Symposium on Foundations of Computer Science (Ann Arbor, Mich., 1978), IEEE, Long Beach, Calif., 1978, pp. 75–83. MR 539832

[Ahl78] Lars V. Ahlfors, *Complex analysis*, third ed., McGraw-Hill Book Co., New York, 1978, An introduction to the theory of analytic functions of one complex variable, International Series in Pure and Applied Mathematics. MR 510197

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena, *PRIMES is in P*, Ann. of Math. (2) **160** (2004), no. 2, 781–793. MR 2123939

[BCHW16] F. G. S. L. Brandao, M. Christandl, A. W. Harrow, and M. Walter, *The Mathematics of Entanglement*, ArXiv e-prints (2016).

[Bel64] J.S. Bell, *On the einstein-podolsky-rosen paradox*, Physics **1** (1964), 195–200.

[BW92] Charles H. Bennett and Stephen J. Wiesner, *Communication via one- and two-particle operators on einstein-podolsky-rosen states*, Phys. Rev. Lett. **69** (1992), 2881–2884.

[CHSH69] John F. Clauser, Michael A. Horne, Abner Shimony, and Richard A. Holt, *Proposed experiment to test local hidden-variable theories*, Phys. Rev. Lett. **23** (1969), 880–884.

[Chu36] Alonzo Church, *An Unsolvable Problem of Elementary Number Theory*, Amer. J. Math. **58** (1936), no. 2, 345–363. MR 1507159

[CT65] James W. Cooley and John W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp. **19** (1965), 297–301. MR 0178586

[EPR35] A. Einstein, B. Podolsky, and N. Rosen, *Can quantum-mechanical description of physical reality be considered complete?*, Phys. Rev. **47** (1935), 777–780.

[Erd47] P. Erdös, *Some remarks on the theory of graphs*, Bull. Amer. Math. Soc. **53** (1947), 292–294. MR 0019911

[Gau] C. F. Gauss, *Nachlass: Theoria interpolationis methodo nova tractata, gauss, werke, band 3*.

[GHIL16] Fulvio Gesmundo, Jonathan D. Hauenstein, Christian Ikenmeyer, and J. M. Landsberg, *Complexity of linear circuits and geometry*, Found. Comput. Math. **16** (2016), no. 3, 599–635. MR 3494506

[Gle11] James Gleick, *The information*, Pantheon Books, 2011.

[KLPSMN09] Abhinav Kumar, Satyanarayana V. Lokam, Vijay M. Patankar, and Jayalal Sarma M. N., *Using elimination theory to construct rigid matrices*, Foundations of software technology and theoretical computer science—FSTTCS 2009, LIPIcs. Leibniz Int. Proc. Inform., vol. 4, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2009, pp. 299–310. MR 2870721

[KSV02] A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi, *Classical and quantum computation*, Graduate Studies in Mathematics, vol. 47, American Mathematical Society, Providence, RI, 2002, Translated from the 1999 Russian original by Lester J. Senechal. MR 1907291

[Lan61] R. Landauer, *Irreversibility and heat generation in the computing process*, IBM Journal of Research and Development **5** (1961), 183–191.

[Lan17] J.M. Landsberg, *Geometry and complexity theory*, Cambridge studies in advanced mathematics, vol. 169, Cambridge Univ. Press, 2017.

[Lyo09] Jonathan Lyons, *The house of wisdom*, Bloomsbury Press, 2009.

[Rab80] Michael O. Rabin, *Probabilistic algorithm for testing primality*, J. Number Theory **12** (1980), no. 1, 128–138. MR 566880

[Sha48] C. E. Shannon, *A mathematical theory of communication*, Bell System Tech. J. **27** (1948), 379–423, 623–656. MR 0026286

[Sho94] Peter W. Shor, *Algorithms for quantum computation: discrete logarithms and factoring*, 35th Annual Symposium on Foundations of Computer Science (Santa Fe, NM, 1994), IEEE Comput. Soc. Press, Los Alamitos, CA, 1994, pp. 124–134. MR 1489242

[Sho97] ———, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput. **26** (1997), no. 5, 1484–1509. MR 1471990

[SS71] A. Schönhage and V. Strassen, *Schnelle Multiplikation grosser Zahlen*, Computing (Arch. Elektron. Rechnen) **7** (1971), 281–292. MR 292344

[Str69] Volker Strassen, *Gaussian elimination is not optimal*, Numer. Math. **13** (1969), 354–356. MR 40 #2223

[Val77] Leslie G. Valiant, *Graph-theoretic arguments in low-level complexity*, Mathematical foundations of computer science (Proc. Sixth Sympos., Tatranská Lomnica, 1977), Springer, Berlin, 1977, pp. 162–176. Lecture Notes in Comput. Sci., Vol. 53. MR 0660702 (58 #32067)

[VSD86] A Vergis, K Steiglitz, and B Dickinson, *The complexity of analog computation*, Math. Comput. Simul. **28** (1986), no. 2, 91–113.