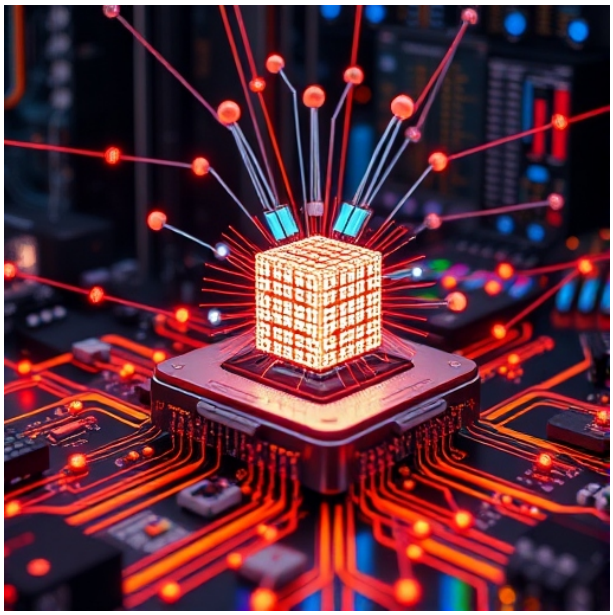
Quantum Programming Lecture Notes

Colorado School of Mines
CSCI 581

Matthew Fox
University of Colorado Boulder



Last Updated: April 16, 2025

Contents

Foreword

This is a set of lecture notes on quantum computation that was originally prepared for the Quantum Programming (CSCI 581) course at the Colorado School of Mines during the Spring 2025 semester. These notes are intended for advanced undergraduate and first-year graduate students who have taken a course in computability theory and who have had some exposure to quantum mechanics and/or quantum information theory. That said, the requisite quantum mechanics is introduced within the first few lectures, so anyone lacking a "quantum" background can in principle follow along.

These notes cover several important topics in quantum computing theory, including: the postulates of quantum mechanics, the theory of classical reversible computation in the circuit model, the theory of quantum computation in the circuit model, quantum computational complexity theory, and several important quantum algorithms, such as Grover's algorithm, Shor's algorithms for factoring and the discrete logarithm problem, and other more general hidden subgroup problems. These notes also feature lectures on Hamiltonian simulation and quantum and post-quantum cryptography.

Throughout these notes, an asterisk (*) means "Time Permitting", i.e., that the marked topic will probably not be covered in class, but that the topic is nevertheless relevant to the material at hand, and should therefore be read outside of class. Moreover, these notes consist of several exercises, discussions, and questions, and anyone reading these notes should try their hand at all of these. They are generally easy to do, at least mathematically speaking.

Please keep in mind that I wrote these notes on a tight schedule. In consequence, they are in no way a comprehensive treatment of the subject nor are they guaranteed to be error-free. Corrections by email to matthew.fox@colorado.edu are welcome.

I hope you find these notes to be a pedagogically useful resource for learning about the theory of quantum computation.

Lecture 1
A Most Incomprehensible Thing

In this first lecture, we will review the notion of computable functions, the Stern-Gerlach experiment, and a simplified version of Bell's theorem. Part of this presentation is based on David Albert's outstanding book *Quantum Mechanics and Experience*. My point is simply to present the inimitable bizarreness of the quantum world. Perhaps, as we will explore at length in this course, such weirdness will manifest into a sort of "quantum computational advantage" for computing some of the computable functions.

1.1. The Church-Turing-Deutsch Thesis

In this class, we are ultimately interested in functions $f : \mathbb{N} \to \mathbb{N}$ (or, equivalently, $f : \Sigma^* \to \Sigma^*$, where Σ is an alphabet of symbols, like $\{0, 1\}$). Let's begin by seeing how many of such functions there are.

Claim 1.1. *There are uncountably many $f : \mathbb{N} \to \mathbb{N}$.*

Proof. Suppose there are only countably many. Then, we can enumerate them: $f_1, f_2, \ldots$. Let $h(n) \coloneqq f_n(n) + 1$. Clearly $h : \mathbb{N} \to \mathbb{N}$, however $h \neq f_n$ for all n, which is a contradiction. ■

We are particularly interested in those $f : \mathbb{N} \to \mathbb{N}$ that we can "compute" in some sense. To get at this, consider the following informal idea:

Definition 1.1. Say $f : \mathbb{N} \to \mathbb{N}$ is *effectively calculable* (*EC*) iff there exists a finite, pen-and-paper procedure whereby a rote worker can deduce $f(n)$ for every $n \in \mathbb{N}$.

This, of course, is awkwardly informal. Effective calculability gets at the idea that there is some finite, mechanical (and physical!) process to evaluate f on any input. It is the purpose of the Church-Turing thesis to precisify this idea.

Thesis 1.2 (Church-Turing Thesis)**.** *A function $f : \mathbb{N} \to \mathbb{N}$ is EC iff it is computable by a Python program, i.e., iff there exists a Python program M such that $n \in \mathbb{N}$, $M(n) = f(n)$.*[1]

Note that this is not a statement that one can prove. Instead, the CTT is a postulate about computability theory that makes an hitherto informal idea (effective calculability) mathematically rigorous. From this, we can deduce some interesting consequences, such as the following.

Exercise 1.1. *Prove there is $f : \mathbb{N} \to \mathbb{N}$ that no Python program can compute. Conclude that there are uncomputable functions.*

In this class, we will take the perspective that computers are physical devices that evolve according to the laws of physics. This of course is motivated by the CTT, which instantiates this picture of someone working tirelessly with pen-and-paper, deducing on pain of irrationality $f(n)$ for any given n.

Discussion 1.1. Is computational physical? Are computers constrained by physical law?

This motivates the following, highly informal "definition" that is akin to the notion of effective calculability.

Definition 1.2. Say $f : \mathbb{N} \to \mathbb{N}$ is *physically calculable* (PC) iff there exists a finite, physical system S whose mere physical evolution computes $f(n)$ for any $n \in \mathbb{N}$.

Again, this is awkwardly informal. Moreover, this also feels quite different from the once but no longer informal notion of effective calculability, which was based on the picture of a rote worker computing with pen and paper. That said, the EC picture is nevertheless physical, so the following claim is somewhat clear:

Claim 1.3. *If $f : \mathbb{N} \to \mathbb{N}$ is computable by a Python program, then f is PC.*

Discussion 1.2.

(i) Is this reasonable?

(ii) What about the converse? Is there a PC $f : \mathbb{N} \to \mathbb{N}$ (computed, perhaps by a bunch of electrons or, more exotically, the Hawking radiation from a Schwarzschild black hole) that is *un*computable?

[1] Note that one can replace "Python program" with "deterministic Turing machine", for example, because Python is a Turing-complete programming language (as is PowerPoint, by the way).

In fact, it is generally believed that every PC function is computable. This is based on reductionist arguments that quantum mechanics (most likely) underlies everything there is in the universe and, as we will see in this course, that quantum systems cannot compute uncomputable functions. Put together, this belief constitutes its own thesis:

Thesis 1.4 (Church-Turing-Deutsch Thesis)**.** *A function $f : \mathbb{N} \to \mathbb{N}$ is PC iff it is computable by a Python program.*

Again, like the CTT, this statement is not something that one can prove. Instead, its purpose is to make mathematically precise an hitherto informal idea (physical calculability). Note also that this thesis is no longer just a postulate about computability theory, but is also a postulate about the physical world. In my opinion, this elevates the set of all computable functions to the level of a fundamental constant of nature, on a par with the speed of light c, Plank's constant $\hbar$, and Newton's gravitational constant G.

Conclusion 1.1. Quantum systems (and quantum computers in particular) cannot compute functions that no classical computer (like a Python program or deterministic Turing machine) can. In other words, for every quantum computer, there exists a classical computer that can simulate it, and vice versa.

Question 1.1. *If this is the case, then what is this course about? That is, if classical computers can simulate physical systems (and hence if classical computers can compute the same set of functions that physical systems can), then what are we doing here?*

1.2. Feynman's Vision

In 1981, the physics Nobel laureate Richard Feynman wrote a paper entitled *Simulating Physics with Computers*. There, he notes that if $|\psi\rangle \in \mathbb{C}^{2^n}$ is the state of a quantum system, then, generally speaking,

> a faithful description of $|\psi\rangle$ seems to require $\Omega(2^n)$ complex numbers.

Therefore, to accurately simulate the evolution of $|\psi\rangle$ would require storing an exponential number of parameters, so that the overall simulation will take a *very* long time (as a function of n). The quantum system, however, merely evolves its state $|\psi\rangle$ with ease. In this way, it seems that quantum systems might be able to compute a function more efficiently than any classical computer. So indeed, this

course is not about effective calculability, but *efficient* calculability. Overall, the suspicion of most folks in the quantum computing world is the following:

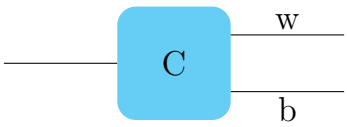
Conjecture 1.5. *There exists $f : \mathbb{N} \to \mathbb{N}$ that a quantum computer can compute in polynomial time but any classical computer takes exponential time. Note, this does not mean that we think quantum computers can solve every problem in* NP *because f need not correspond to an* NP*-complete problem (e.g. factoring integers).*

Ultimately, we expect this "quantum advantage" or "quantum speedup" to come from the inimitable bizarreness of the quantum world, some of which we will now discuss.

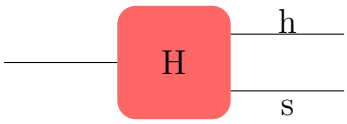
1.3. An Experimental Fact of Life

Every quantum particle (an electron or silver atom, say) appears to have an intrinsic property that we call *color* and an intrinsic property that we call *hardness*. Whenever we look to see what the color of a particle is, we only ever see it to be either white (w) or black (b). Likewise, whenever we look to see what the hardness of a particle is, we only ever see it to be either hard (h) or soft (s). For decades, no other color or hardness has ever been seen, so we are confident that these are the only possible color and hardness values.

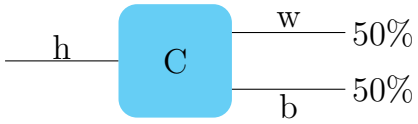
It is possible to build a *color box*, C, which resolves the color (w or b) of a particle. It acts by taking as input a particle (left), whose color can be known or unknown, and, after a short time, by ejecting as output the same particle (right) on either the top track, if the color of the particle is white, or on the bottom track, if the color of the particle is black. Diagrammatically,



Similarly, we can build a *hardness box*, H, which resolves the hardness (h or s) of an input particle akin to how a color box resolves the color of an input particle. It looks like this:

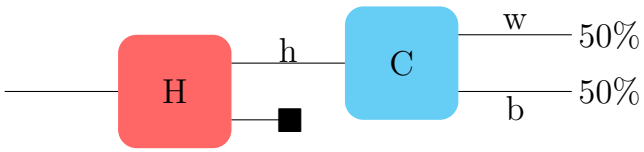


We may wonder if color and hardness are correlated. But, after many trials of feeding only hard particles into a color box, we conclude they are not because, in aggregate, 50% of the hard particles came out black and 50% came out white:



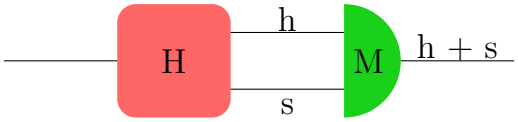
Question 1.2. *How can we do this experiment without a source of hard particles?*

Answer: this setup can be achieved by concatenating a hardness box with a color box, and then directing the hard output of the hardness box into the input of the color box:



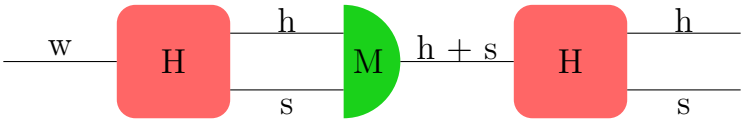
The same 50-50 statistics result if we instead feed soft particles into a color box, white particles into a hardness box, or, finally, black particles into a hardness box. These results only reaffirm our previous conclusion that hardness and color are uncorrelated.

Now consider the following apparatus:



Here, the logical-AND-looking M gate is the *commingler*—it commingles the hard and soft beams through a simple placement of mirrors (which experiments decidedly show do not affect the hardness or color of a particle).

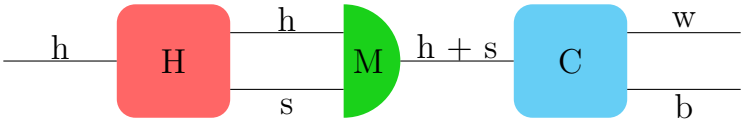
Suppose we feed many white particles into this apparatus and then measure the hardness of the h + s beam, as in:



Question 1.3. *Given what has been said so far, at the output of the first hardness box (left), what percentage of white particles do you expect to come out soft and what percentage do you expect to come out hard?*

Answer: at the output of the *first* hardness box (left), we expect 50% of the white particles to be soft and 50% to be hard. By design, the commingler M does not change hardness or color, so these statistics should hold in the h + s beam. We therefore expect 50% of the output of the *second* hardness box (right) to emerge soft and 50% to emerge hard since 50% of the input to the second hardness box is soft and 50% is hard. Indeed, this is exactly what happens.

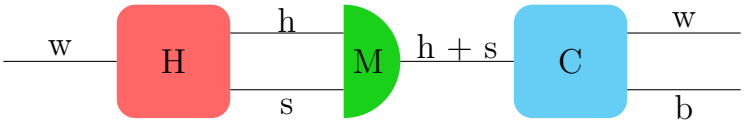
Now consider the simple variation below, where instead of inputting a white particle we input a hard particle, and instead of measuring its hardness after the commingler we measure its color:



Question 1.4. *Now what do you expect the statistics to be?*

Answer: In this case, since the input particles are all hard, we expect the hardness box to output zero soft particles. The commingler does not change the hardness, so 100% of the particles in the h + s beam should be hard. We know when we input a hard particle into a color box that the output is split 50% white and 50% black, so this is what we should expect. Indeed, this is exactly what happens.

Now consider one final variation to this experiment. Suppose into this apparatus we input a *white* particle, as opposed to a hard particle, and then, after the commingler, measure its color like in the previous experiment:

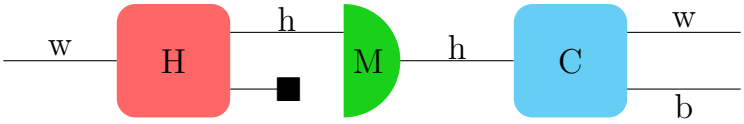


Given the reasoning we've used so far, there are two ways you might go here. On one hand, we input white particles, so it is natural to expect only white particles to emerge. On the other hand, at the output of the hardness box, we expect 50% of the white particles to be soft and 50% to be hard. The commingler M does not change hardness or color, so these statistics should hold in the h + s beam. We therefore expect 50% of the input to the color box to soft and 50% to be hard. Therefore, one might also expect 50% of the output to be white and 50% to be black.

Question 1.5. *What do you think will happen?*

Answer: In fact, 100% of the output is white. No black particles ever emerge from this apparatus.

Let's look more closely at this experimeint. Since the output of a color box is white if and only if the input is white, it must be that the h + s beam is composed of only white particles. But the h + s beam comes from the commingler, which does not change the color or hardness of the individual h and s beams. Thus, the individual h and s beams must themselves be exclusively composed of white particles. To test this hypothesis, we block off the internal soft beam:



Question 1.6. *What do you think happens?*

Answer: In fact, instead of finding that the output is 100% white particles, the output goes back to 50% white and 50% black. The same thing happens if we instead blocked off the internal hard beam.

This is (ostensibly) bad news for logic. To see why, ask the innocent question: *along which internal path (h or s) did the input white particle go?*

If it took h, then blocking the s path should have no effect. But, as we just saw, blocking the s path *does* have an effect: the output statistics change.

If it took s, then blocking the h path should have no effect. But, like in the last experiment, blocking the h path *does* have an effect: the output statistics change.

Maybe it somehow took both? Suppose, then, that when the particle is traversing the internal path we closely scrutinize the two tracks. No matter how we look, we invariably see it on only *one* of the two paths, so it makes no sense to say that it takes both!

Maybe it took neither? But that's moonshine: if it takes neither, then blocking both the h and s paths should have no effect, yet doing that yields no output at all!

Surely, then, something is amok with these color and hardness boxes and the commingler. After all, what else could be responsible for this inscrutable behavior? However, after decades of R&D into wildly different color and hardness boxes and comminglers, all of which function perfectly but via totally unrelated means, our credence that it is the experimental apparatus at fault nears zero.

Hence, with exceptionally high credence—higher credence than most other scientific exploits—we disconcertingly find ourselves with this: *particles passing through this last apparatus, to the extent that we are able to understand them, do not take the internal route h, nor the internal route s, nor both, nor neither.* These exhaust

the logical possibilities. Therefore, if this is right—and again we have exceptionally good evidence that it is—then there can be no *fact* of what internal route the particle took. In other words, despite our ability to measure the hardness of a white particle and to obtain, in every instance of measuring, a demonstrable fact of the matter of what we measure the hardness of the white particle to be, before we measure the hardness, there is no fact of the hardness of a white particle. In the philosopher David Albert's words, "asking which internal path the electron took is like asking what is the marital status of the number 5."

Something extraordinary is happening pre-measurement in this experiment, and that extraordinary thing is *quantum superposition*.

Discussion 1.3. Given what you know about quantum superpositions, how do you think about this? Is there anything in classical computing that resembles quantum superposition? In your day-to-day, is there anything that resembles a quantum superposition?

Much of quantum computing rests on our ability to create quantum superpositions, and then, in some sense, to compute a bunch of things in parallel. We will see exactly how this works in a few lectures.

1.4. Another Experimental Fact of Life

Yet another bizarre feature of quantum mechanics stems from something called *entanglement*, which we will formally define in the next lecture. In fact, this feature (called a Bell inequality violation) is arguably the most quintessentially quantum thing about quantum mechanics.

Definition 1.3. Let S be a physical system with measurable properties A, B, and C, and let

$$\begin{aligned} N_S(A,B,C) &= \#\text{ times we see } S \text{ with } A, B, \text{ and } C \\ N_S(A,B,\bar{C}) &= \#\text{ times we see } S \text{ with } A, B, \text{ and NOT } C \\ N_S(A,B) &= \#\text{ times we see } S \text{ with } A \text{ and } B, \\ &\vdots \end{aligned}$$

Example 1.1.

- S is a car, A is its speed relative to the road, B is its GPS coordinates, and C is its color.

- S is a star, A is its mass, B is its luminosity, and C is its angular momentum.
- S is an electron, A is its color (a.k.a. its x-spin), B is its hardness (a.k.a. its y-spin), and C is its z-spin.

Claim 1.6. *For all systems S with measurable properties A, B, and C,*

$$N_S(A, \bar{B}) + N_S(B, \bar{C}) \geq N_S(A, \bar{C}).$$

Proof. The right-hand side equals

$$\text{RHS} = N_S(A, \bar{C}) = N_S(A, B, \bar{C}) + N_S(A, \bar{B}, \bar{C}).$$

The left-hand side equals

$$\begin{aligned}
\text{LHS} &= N_S(A, \bar{B}) + N_S(B, \bar{C}) \\
&= N_S(A, \bar{B}, C) + \underbrace{N_S(A, \bar{B}, \bar{C}) + N_S(A, B, \bar{C})}_{\text{RHS}} + N_S(\bar{A}, B, \bar{C}) \\
&= N_S(A, \bar{B}, C) + N_S(\bar{A}, B, \bar{C}) + \text{RHS} \\
&\geq \text{RHS}.
\end{aligned}$$

■

Theorem 1.7 (A Version of Bell's Theorem)**.** *Let S be two maximally entangled electrons, e_1 and e_2, and consider the measurable properties*

- *A = the spin of e_1 is up along the z-axis,*
- *B = the spin of e_2 is up along the $(\theta, 0)$-axis,*
- *C = the spin of e_2 is up along the $(2\theta, 0)$-axis.*

Then, for sufficiently small θ,

$$N_S(A, \bar{B}) + N_S(B, \bar{C}) < N_S(A, \bar{C}).$$

What the heck is going on will have to wait. Interestingly, though, in certain models of quantum computation that exhibit a provable quantum advantage (namely, quantum shallow circuits), it is possible to directly relate their advantage to a Bell inequality violation like this.

Lecture 2
Quantum Mechanics I

Discussion 2.1. Discuss with your group what you took away from last time.

Last time, we discussed the Church-Turing-Deutsch thesis, which expresses the belief that quantum computers can probably not compute Turing uncomputable functions. However, this leaves open the possibility that quantum computers might be able to compute certain functions *faster* than any classical computer, thanks to the weirdness of quantum mechanics (such as things like superposition and Bell's theorem, the latter of which turns out to be possible because of entanglement).

In this lecture, we will study some of the postulates of quantum mechanics, their mathematical formalism, and the definition of a qubit.

2.1. States of Quantum Systems

Every physical theory has physical primitives that do not have a totally agreeable definition. For example, a "particle" is a primitive in Newtonian mechanics and in Einstein's relativity. In quantum mechanics, the physical primitive is a "quantum system". It is a "you know it when you see it" kind of thing, which exists in the real world. If you like reductionism, then most likely every physical system is a quantum system, though that position is contended by some.

Postulate 2.1.

- To every quantum system S, there corresponds a complex-valued, complete inner product space (a.k.a. a *Hilbert space*)

$$\mathcal{H} = \mathbb{C}^N,$$

  where N is the *dimension* of $\mathcal{H}$.

- We denote a generic vector in $\mathcal{H}$ using Paul Dirac's "ket notation", $|\psi\rangle$. This can be thought of as a column vector in some basis:

$$|\psi\rangle = \begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{N-1} \end{pmatrix}, \quad \psi_i \in \mathbb{C}.$$

- To every $|\psi\rangle \in \mathcal{H}$, there corresponds a dual vector $\langle\psi|$, known as a "bra vector", which formally is in the dual space of $\mathcal{H}$. The dual vector $\langle\psi|$ can be thought of as a row vector that is the conjugate transpose of $|\psi\rangle$:

$$\begin{aligned} \langle\psi| &= |\psi\rangle^\dagger \\ &\coloneqq (|\psi\rangle^*)^T \\ &= \begin{pmatrix} \psi_0^* & \psi_1^* & \cdots & \psi_{N-1}^* \end{pmatrix}. \end{aligned}$$

- The inner-product $\langle\cdot|\cdot\rangle$ on $\mathcal{H}$ is the bra-ket inner-product:

$$\begin{aligned} \langle\psi|\phi\rangle &\coloneqq \langle\psi||\phi\rangle \\ &= \begin{pmatrix} \psi_0^* & \psi_1^* & \cdots & \psi_{N-1}^* \end{pmatrix} \begin{pmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_{N-1} \end{pmatrix} \\ &= \sum_{i=0}^{N-1} \psi_i^* \phi_i. \end{aligned}$$

- The bra-ket inner-product induces a norm on $\mathcal{H}$:

$$\||\psi\rangle\| \coloneqq \sqrt{\langle\psi|\psi\rangle} = \sqrt{\sum_{i=0}^{N-1} \psi_i^* \psi_i}.$$

- A *unit-vector* $|\psi\rangle$ is a vector in $\mathcal{H}$ whose norm is one, i.e.,

$$\||\psi\rangle\| = \sqrt{\langle\psi|\psi\rangle} = 1.$$

- A *state* of the system S is a unit-vector in $\mathcal{H}$.

Fact 2.1. *If $|\psi\rangle$ and $|\phi\rangle$ are quantum states, then so is*

$$\alpha|\psi\rangle + \beta|\phi\rangle$$

for all $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$, where $|\alpha|^2 \coloneqq \alpha^*\alpha$. This is superposition.

Example 2.1. A two-state system, such as the spin of an electron, is called a *quantum bit* or *qubit* for short. Mathematically, we describe this using a two-dimensional Hilbert space

$$\mathcal{H}_{\text{qubit}} = \mathbb{C}^2 = \text{span}\left\{\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\right\}.$$

Exercise 2.1.

(1) Do you know a quantum system whose Hilbert space is $\mathbb{C}^4$? $\mathbb{C}^{2^n}$?

(2) What would you say the difference is between the states $\frac{1}{\sqrt{2}}|\psi\rangle + \frac{1}{\sqrt{2}}|\phi\rangle$ and $|\psi\rangle$ with probability 1/2 and $|\phi\rangle$ with probability 1/2?

Definition 2.1. The *computational basis of* $\mathbb{C}^N$ is the orthonormal basis

$$B(N) \coloneqq \left\{\begin{pmatrix}1\\0\\\vdots\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\\vdots\\0\\0\end{pmatrix}, \ldots, \begin{pmatrix}0\\0\\\vdots\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\\vdots\\0\\1\end{pmatrix}\right\}.$$

If $N = 2$, then we label the two computational basis states using bits:

$$|0\rangle \coloneqq \begin{pmatrix}1\\0\end{pmatrix} \quad \text{and} \quad |1\rangle \coloneqq \begin{pmatrix}0\\1\end{pmatrix}.$$

Therefore,

$$\mathcal{H}_{\text{qubit}} = \text{span}\big\{|0\rangle, |1\rangle\big\}.$$

Exercise 2.2. *Let $\{0,1\}^n$ be the set of all n-bit strings. Prove $\{0,1\}^n$ and $B(2^n)$ are bijective. Why is that interesting from an information-encoding point of view?*

2.2. Composite Systems

Postulate 2.2. Let A and B be quantum systems with Hilbert spaces $\mathcal{H}_A = \mathbb{C}^{N_A}$ and $\mathcal{H}_B = \mathbb{C}^{N_B}$, respectively. The combined quantum system $A + B$ is described by the Hilbert space

$$\mathcal{H}_{A+B} = \mathcal{H}_A \otimes \mathcal{H}_B \cong \mathbb{C}^{NM},$$

where $\otimes$ denotes the *tensor product*.

What does this mean?

Definition 2.2. Let $|a_0\rangle, |a_1\rangle, \ldots, |a_{N_A-1}\rangle$ and $|b_0\rangle, |b_1\rangle, \ldots, |b_{N_B-1}\rangle$ be bases of $\mathcal{H}_A$ and $\mathcal{H}_B$, respectively. Then,

$$\mathcal{H}_A \otimes \mathcal{H}_B = \text{span}\Big\{|a_i\rangle \otimes |b_j\rangle : i \in \{0, \ldots, N_A - 1\}, j \in \{0, \ldots, N_B - 1\}\Big\},$$

where if

$$|\psi\rangle = \begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{N_A-1} \end{pmatrix} \quad \text{and} \quad |\phi\rangle = \begin{pmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_{N_B-1} \end{pmatrix},$$

then

$$|\psi\rangle \otimes |\phi\rangle \coloneqq \begin{pmatrix} \psi_0|\phi\rangle \\ \psi_1|\phi\rangle \\ \vdots \\ \psi_{N_A-1}|\phi\rangle \end{pmatrix} = \begin{pmatrix} \psi_0 \begin{pmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_{N_B-1} \end{pmatrix} \\ \psi_1 \begin{pmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_{N_B-1} \end{pmatrix} \\ \vdots \\ \psi_{N_A-1} \begin{pmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_{N_B-1} \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \psi_0\phi_0 \\ \psi_0\phi_1 \\ \vdots \\ \psi_0\phi_{N_B-1} \\ \psi_1\phi_0 \\ \psi_1\phi_1 \\ \vdots \\ \psi_1\phi_{N_B-1} \\ \vdots \\ \psi_{N_A-1}\phi_0 \\ \psi_{N_A-1}\phi_1 \\ \vdots \\ \psi_{N_A-1}\phi_{N_B-1} \end{pmatrix}.$$

Similarly,

$$\langle\psi| \otimes \langle\phi| = \begin{pmatrix} \psi_0^*\langle\phi| & \psi_1^*\langle\phi| & \cdots & \psi_{N_A-1}^*\langle\phi| \end{pmatrix}.$$

Notation 2.1. It is often tedious to keep track of all the tensor product symbols. For this reason, I (and others) will often omit the tensor product symbol $\otimes$ when talking about composite systems:

$$|\psi\rangle|\phi\rangle \coloneqq |\psi\rangle \otimes |\phi\rangle$$
$$\langle\psi|\langle\phi| \coloneqq \langle\psi| \otimes \langle\phi|$$

Exercise 2.3.

(1) What is the Hilbert space for a quantum system composed of two qubits?

(2) Evaluate the overall state of the system for two qubits in the states:

(a) $|0\rangle$ *and* $|0\rangle$

(b) $|1\rangle$ *and* $|0\rangle$

(c) $|0\rangle$ *and* $|1\rangle$

(d) $|1\rangle$ *and* $|1\rangle$

(3) What bit strings would you say that each of these states encodes?

Conclusion 2.1. Like how $\{0,1\}$ is the "state-space" of a single bit, $\{|0\rangle, |1\rangle\}$ spans the state space of a single qubit. And like how $\{0,1\}^2 \coloneqq \{00, 01, 10, 11\}$ is the "state space" of two bits, $\{|0\rangle|0\rangle, |0\rangle|1\rangle, |1\rangle|0\rangle, |1\rangle|1\rangle\}$ spans the state space of two qubits, etc.

Fact 2.2 (HW 1)**.**

(1) The tensor product is bilinear. That is, for all $\alpha \in \mathbb{C}$ *and all* $|\psi_1\rangle, |\psi_2\rangle \in \mathcal{H}_A$ *and* $|\phi_1\rangle, |\phi_2\rangle \in \mathcal{H}_B$*:*

- $\alpha(|\psi_1\rangle \otimes |\phi_1\rangle) = (\alpha|\psi_1\rangle) \otimes |\phi_1\rangle = |\psi_1\rangle \otimes (\alpha|\phi_1\rangle)$,
- $(|\psi_1\rangle + |\psi_2\rangle) \otimes |\phi_1\rangle = |\psi_1\rangle \otimes |\phi_1\rangle + |\psi_2\rangle \otimes |\phi_1\rangle$,
- $|\psi_1\rangle \otimes (|\phi_1\rangle + |\phi_2\rangle) = |\psi_1\rangle \otimes |\phi_1\rangle + |\psi_1\rangle \otimes |\phi_2\rangle$.

(2) For all product states $|\psi_1\rangle \otimes |\phi_1\rangle, |\psi_2\rangle \otimes |\phi_2\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$*,*

$$(\langle\psi_1| \otimes \langle\phi_1|)(|\psi_2\rangle \otimes |\phi_2\rangle) = \langle\psi_1|\psi_2\rangle \cdot \langle\phi_1|\phi_2\rangle.$$

(3) The norm of a tensor product is the product of the norms, i.e. for all $|\psi\rangle \in \mathcal{H}_A$ *and all* $|\phi\rangle \in \mathcal{H}_B$*,*

$$\||\psi\rangle \otimes |\phi\rangle\| = \||\psi\rangle\| \cdot \||\phi\rangle\|.$$

(4) The tensor product is not commutative.

Exercise 2.4. *Prove this last fact, that the tensor product is not commutative. (*Hint: *What are $|0\rangle|1\rangle$ and $|1\rangle|0\rangle$?)*

Fact 2.3 (Entanglement)**.** *By the bilinearity of the tensor product, it is sometimes possible to factor states:*

$$|\psi_1\rangle \otimes |\phi\rangle + |\psi_2\rangle \otimes |\phi\rangle = (|\psi_1\rangle + |\psi_2\rangle) \otimes |\phi\rangle.$$

We say this state is separable*, because the state can be written as a state from $\mathcal{H}_A$ times a state from $\mathcal{H}_B$. However, this is not always possible, for example:*

$$|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle.$$

This (unnormalized) state is not separable, i.e., it is entangled*.*

Definition 2.3. Let $\mathcal{H}_A$ and $\mathcal{H}_B$ be Hilbert spaces. A state $|\psi\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$ is *separable* iff there exist $|\phi\rangle_A \in \mathcal{H}_A$ and $|\phi\rangle_B \in \mathcal{H}_B$ such that $|\psi\rangle = |\phi\rangle_A \otimes |\phi\rangle_B$. We say $|\psi\rangle$ is *entangled* iff it is not separable.

Exercise 2.5. *Determine whether the following (unnormalized) states are separable or entangled:*

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

(2) $|1\rangle|0\rangle - e^{i\pi/42}|1\rangle|1\rangle$

(3) $|0\rangle|0\rangle|0\rangle - |0\rangle|0\rangle|1\rangle$

(4) $|0\rangle|0\rangle|0\rangle - |0\rangle|1\rangle|1\rangle$

(5) $|0\rangle|1\rangle|1\rangle + |1\rangle|0\rangle|0\rangle$

(6) $|0\rangle|0\rangle|0\rangle + |1\rangle|1\rangle|1\rangle$

(7) $|0\rangle|1\rangle|0\rangle + |1\rangle|1\rangle|1\rangle$*. Would you say this is more or less entangled than the state in (6)?*

In fact, there is a whole zoo of "entanglement measures" out there, which allow one to make statements like "this state is more entangled than that state, at least with respect to this measure". While there are many subtleties in entannglement theory, it is the case that for so-called pure, bipartite state (which is what we are dealing with here), there are canonical *maximally entangled states*. The four most relevant to us are the so-called "Bell states".

Definition 2.4. The four *Bell states* are the maximally entangled states

$$
\begin{aligned}
|\Phi^+\rangle &:= \frac{1}{\sqrt{2}}\left(|0\rangle|0\rangle + |1\rangle|1\rangle\right) \\
|\Phi^-\rangle &:= \frac{1}{\sqrt{2}}\left(|0\rangle|0\rangle - |1\rangle|1\rangle\right) \\
|\Psi^+\rangle &:= \frac{1}{\sqrt{2}}\left(|0\rangle|1\rangle + |1\rangle|0\rangle\right) \\
|\Psi^-\rangle &:= \frac{1}{\sqrt{2}}\left(|0\rangle|1\rangle - |1\rangle|0\rangle\right).
\end{aligned}
$$

Exercise 2.6. *Prove that the Bell states form an orthonormal basis of* $\mathbb{C}^4$.

The Bell states will come up when we prove (a variant of) Bell's theorem in the next lecture. They are also key to many quantum mechanical protocols, such as superdense coding and quantum teleportation.

Lecture 3
Quantum Mechanics II

Discussion 3.1. Discuss with your group what you took away from last time.

In the last lecture, we discussed two of four postulates of quantum mechanics, namely, how we represent states of quantum systems mathematically (unit vectors in particular Hilbert spaces) and how we describe composite quantum systems (the tensor product of Hilbert spaces).

In this lecture, we will discuss the other two postulates of quantum mechanics. In particular, we will discuss how quantum systems evolve and how they are measured. Time permitting, I will also discuss one way of thinking about this, which I like, but note that this is by no means the only way of thinking about quantum mechanics.

3.1. The Evolution of Quantum Systems

Postulate 3.1. Let S be a quantum system with Hilbert space $\mathcal{H}_S = \mathbb{C}^N$. Absent any "measurements" of the system, if at time t_1 the state of S is $|\psi(t_1)\rangle$ and if at time $t_2 \geq t_1$ the state of S is $|\psi(t_2)\rangle$, then there exists an $N \times N$ *unitary* matrix U such that

$$|\psi(t_2)\rangle = U|\psi(t_1)\rangle.$$

In other words, with no measurements, every quantum state $|\psi\rangle \in \mathcal{H}_S$ evolves in time *unitarily*. This is one form of the *Schrödinger equation*.

What does this mean?

Definition 3.1.

- An $N \times N$ complex-valued matrix U is *unitary* iff $U^{-1} = U^\dagger \coloneqq (U^*)^T$.
- $\mathrm{U}(N) \coloneqq \{N \times N \text{ unitary matrices } U\}$. With matrix multiplication, $\mathrm{U}(N)$ forms a group (in fact a Lie group) called the *unitary group of order* N.

- $\mathrm{SU}(N) \coloneqq \{N \times N \text{ unitary matrices } U \text{ with } \det U = 1\}$. With matrix multiplication, $\mathrm{SU}(N)$ also forms a (Lie) group called the *special unitary group of order* N.

Example 3.1. The following matrices are unitary:

- I_N (the $N \times N$ identity),
- the T or $\pi/8$ *gate*:

$$T \coloneqq \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, \qquad T^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\pi/4} \end{pmatrix},$$

- the *Hadamard gate*:

$$H \coloneqq \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = H^\dagger,$$

- the S or *phase gate*:

$$S \coloneqq T^2 = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \qquad S^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix},$$

- the X, Y, *and* Z *Pauli matrices*:

$$X \coloneqq \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = X^\dagger, \qquad Y \coloneqq \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = Y^\dagger, \quad \text{and} \quad Z \coloneqq \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = Z^\dagger,$$

- the SWAP *gate*

$$\mathrm{SWAP} \coloneqq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \mathrm{SWAP}^\dagger,$$

- and the CNOT *gate*

$$\mathrm{CNOT} \coloneqq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \mathrm{CNOT}^\dagger.$$

Exercise 3.1.

(i) What are $\mathrm{U}(1)$ *and* $\mathrm{SU}(1)$ *geometrically?*

(ii) Let $U \in \mathrm{U}(N)$ *and* $|\psi_1\rangle, |\phi_1\rangle \in \mathbb{C}^N$*. If* $|\psi_2\rangle = U|\psi_1\rangle$ *and* $|\phi_2\rangle = U|\phi_1\rangle$*, prove that*

$$\langle\phi_2|\psi_2\rangle = \langle\phi_1|\psi_1\rangle.$$

Conclusion 3.1. This last exercise is important. It reveals that unitary maps *preserve the inner-product of a Hilbert space.* Therefore, unitary maps are the *structure-preserving* maps on Hilbert spaces, in the same way that bijective maps are the structure-preserving maps on sets, homeomorphisms are the structure-preserving maps on topological spaces, homomorphisms are the structure-preserving maps on groups, and so forth.

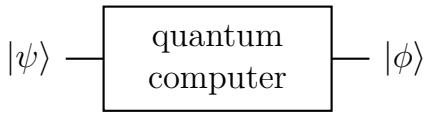
Fact 3.1.

(i) With matrix multiplication, both $\mathrm{U}(N)$ *and* $\mathrm{SU}(N)$ *are groups for all* $N \geq 1$*.*[1]

(ii) If $U \in \mathrm{U}(N)$ *and* λ *is an eigenvalue of* U*, then* $\lambda = e^{i\theta}$ *for some* $\theta \in [0, 2\pi)$*.*

3.2. Application: Quantum Computers

We will formally discuss what a quantum computer is in a few lectures. At a high level, though, a quantum computer is simply a map that takes as input a bunch of qubits prepared in a particular state $|\psi\rangle \in \mathbb{C}^{2^n}$, and outputs another state $|\phi\rangle \in \mathbb{C}^{2^n}$. Overall, a quantum computer looks like:



Therefore, a quantum computer simply evolves in time the state $|\psi\rangle$ to a different state $|\phi\rangle$. Consequently, there exists a unitary operator $U_{\mathrm{QC}} \in \mathrm{U}(2^n)$ that implements

$$|\phi\rangle = U_{\mathrm{QC}}|\psi\rangle.$$

In other words, *every quantum computer is a unitary operator.* Exactly what unitary operator will have to wait.

[1] Recall, a *group* is a pair $(G, \cdot)$, where G is a set and $\cdot : G \times G \to G$ is a binary operation, such that the operation is associative, G has an identity element, and G is closed under inverses.

3.3. Observables and Projective Measurements

Definition 3.2. An $N \times N$ complex-valued matrix M is *Hermitian* iff

$$M = M^\dagger \coloneqq (M^*)^T.$$

Claim 3.2. *Every Hermitian matrix M has only real eigenvalues.*

Proof. Let λ be an eigenvalue of M with eigenvector $|\lambda\rangle$. Then,

$$M|\lambda\rangle = \lambda|\lambda\rangle \quad \text{and} \quad \langle\lambda|M^\dagger = \langle\lambda|M = \lambda^*\langle\lambda|.$$

Thus, on one hand,

$$\langle\lambda|M|\lambda\rangle = \lambda\langle\lambda|\lambda\rangle.$$

On the other hand,

$$\langle\lambda|M|\lambda\rangle = \langle\lambda|M^\dagger|\lambda\rangle = \lambda^*\langle\lambda|\lambda\rangle.$$

By definition, every eigenvector $|\lambda\rangle \neq 0$. Therefore, $\langle\lambda|\lambda\rangle \neq 0$, so

$$\lambda = \lambda^*,$$

as desired. ■

Theorem 3.3 (Spectral Decomposition of Hermitian Operators)**.** *Let M be a Hermitian operator on $\mathbb{C}^N$ with (real) eigenvalues $\lambda_1, \ldots, \lambda_N$ and eigenvectors $|\lambda_1\rangle, \ldots, |\lambda_N\rangle$, respectively. Then,*

$$M = \sum_{i=1}^{N} \lambda_i \Pi_i,$$

where $\Pi_i \coloneqq |\lambda_i\rangle\langle\lambda_i|$ is the projector *onto the $|\lambda_i\rangle$ subspace of $\mathbb{C}^N$.*

Real numbers are what we see when we measure things in the real world. For this and other reasons, Hermitian operators are equated with "observables of quantum systems".

Postulate 3.2. Let S be a quantum system with Hilbert space $\mathcal{H}_S = \mathbb{C}^N$. To "measure" S means to apply a (not necessarily unitary) Hermitian operator M to the state of S.[2] If S is in state $|\psi\rangle$ and if $M = \sum_i \lambda_i \Pi_i$ is M's spectral decomposition, then the probability of measuring S to be in state $|\lambda_i\rangle$ is got by the *Born rule*:

$$\Pr[\lambda_i] \coloneqq \Pr[\text{state}(S) = |\lambda_i\rangle] = \langle\psi|\Pi_i|\psi\rangle.$$

[2] An example of a non-unitary Hermitian operator is the projector $|0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$.

Immediately after the measurement, the state of S "collapses" to

$$\frac{\Pi_i|\psi\rangle}{\sqrt{\Pr[\lambda_i]}}.$$

Example 3.2. Let S be a qubit in the superposition state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. We will perform a *computational basis measurement*, which means to measure S using the Z Pauli operator,[3] whose spectral decomposition is

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = (+1)|0\rangle\langle 0| + (-1)|1\rangle\langle 1|.$$

Thus, we see that the computational basis states are exactly the eigenvectors of Z, which is why this is called a "computational basis measurement".

Measuring S in the computational basis, we obtain that the state of S is $|0\rangle$ (the $+1$ eigenstate of Z) with probability

$$\begin{aligned} \Pr[\text{state}(S) = |0\rangle] &= \langle\psi|(|0\rangle\langle 0|)|\psi\rangle \\ &= |\alpha|^2. \end{aligned}$$

In this case, the state collapses to

$$\frac{(|0\rangle\langle 0|)|\psi\rangle}{|\alpha|} = e^{i\theta}|0\rangle,$$

where $e^{i\theta} = \frac{\alpha}{|\alpha|}$.

On the other hand, we obtain that the state of S is $|1\rangle$ (the -1 eigenstate of Z) with probability

$$\begin{aligned} \Pr[\text{state}(S) = |1\rangle] &= \langle\psi|(|1\rangle\langle 1|)|\psi\rangle \\ &= |\beta|^2. \end{aligned}$$

In this case, the state collapses to

$$\frac{(|1\rangle\langle 1|)|\psi\rangle}{|\beta|} = e^{i\theta'}|1\rangle,$$

where $e^{i\theta'} = \frac{\beta}{|\beta|}$.

[3] Physically, this corresponds to measuring the "z-spin" of S, which is what I meant by "color" in the first lecture.

We have now learned what the meaning of the coefficients or "amplitudes" in a quantum state vector mean. They correspond to the *probability density* of seeing the quantum system in a particular basis state.

Example 3.3. Let S be a qubit in the superposition state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. We will perform an X measurement of the system.[4] The spectral decomposition of X is

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = (+1)|+\rangle\langle+| + (-1)|-\rangle\langle-|$$

where

$$|+\rangle \coloneqq \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad (\text{“}x\text{-spin up”})$$
$$|-\rangle \coloneqq \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \quad (\text{“}x\text{-spin down”}).$$

In this basis,

$$|\psi\rangle = \frac{\alpha+\beta}{\sqrt{2}}|+\rangle + \frac{\alpha-\beta}{\sqrt{2}}|-\rangle.$$

Measuring S in the X basis, we obtain that the state of S is $|+\rangle$ (up along the x-axis, which is the $+1$ eigenstate of X) with probability

$$\begin{aligned}\Pr[\text{state}(S) = |+\rangle] &= \langle\psi|(|+\rangle\langle+|)|\psi\rangle \\ &= \frac{|\alpha+\beta|^2}{2}\end{aligned}$$

In this case, the state collapses to

$$\frac{\sqrt{2}}{|\alpha+\beta|}(|+\rangle\langle+|)|\psi\rangle = e^{i\theta}|+\rangle,$$

where $e^{i\theta} = \frac{\alpha+\beta}{|\alpha+\beta|}$.

On the other hand, we obtain that the state of S is $|-\rangle$ (down along the x-axis, which is the -1 eigenstate of X) with probability

$$\begin{aligned}\Pr[\text{state}(S) = |-\rangle] &= \langle\psi|(|-\rangle\langle-|)|\psi\rangle \\ &= \frac{|\alpha-\beta|^2}{2}\end{aligned}$$

[4] Physically, this corresponds to measuring the "x-spin" of S, which is what I meant by "hardness" in the first lecture.

In this case, the state collapses to

$$\frac{\sqrt{2}}{|\alpha-\beta|}(|-\rangle\langle-|)|\psi\rangle = e^{i\theta'}|-\rangle,$$

where $e^{i\theta'} = \frac{\alpha-\beta}{|\alpha-\beta|}$.

3.4. Distinguishing Quantum States

Measuring quantum states is the only way we can "see" them and thereby obtain classical data from them. It is natural to wonder which states are "different" in this regard, in the sense that there is some measurement that we can do to tell them apart.

Definition 3.3. Two states $|\psi\rangle, |\phi\rangle \in \mathcal{H}$ are *operationally equivalent* iff there exists $\theta \in [0, 2\pi)$ such that $|\psi\rangle = e^{i\theta}|\phi\rangle$. We shall write $|\psi\rangle \sim |\phi\rangle$ to denote this equivalence.

Discussion 3.2. Why does this make sense? If $\theta \in (0, 2\pi)$, are the (unnormalized) states

$$|0\rangle + |1\rangle \quad \text{and} \quad |0\rangle + e^{i\theta}|1\rangle$$

really operationally inequivalent? What do you remember from last year?

Theorem 3.4 (The Helstrom–Holevo Bound)**.** *Let S be a quantum system that is either in state $|\psi_a\rangle$ or $|\psi_b\rangle$. The probability of correctly inferring the state of S from a single measurement of its state is at most $\frac{1}{2}(1+\sin\theta)$, where $\theta = \arccos|\langle\psi_a|\psi_b\rangle|$.*[5] *Moreover, this bound is tight, because the Hermitian operator*

$$M = |\psi_a\rangle\langle\psi_a| - |\psi_b\rangle\langle\psi_b|$$

achieves it.

Corollary 3.5.

- *If $|\psi\rangle \sim |\phi\rangle$, then $\theta = 0$, so one cannot experimentally distinguish these states even in principle (because one gets 50-50 statistics).*
- *If $|\psi\rangle \not\sim |\phi\rangle$, then $\theta \neq 0$, so one can, in principle, eventually distinguish these states with enough measurements of M.*
- *If $\langle\psi_a|\psi_b\rangle = 0$, then $\theta = \pi/2$, so one can distinguish the states with probability 1 with just a single measurement of M.*

[5] Note that the range of arccos is $[0, \pi]$, so the range of $\arccos \circ \mathrm{abs}$ is $[0, \frac{\pi}{2}]$.

Lecture 4
Quantum Mechanics III

Discussion 4.1. Discuss with your group what you took away from last time.

In the last lecture we talked about two other postulates of quantum mechanics, both of which concern how quantum states evolve. One way was unitarily, i.e. the evolution implied by the Schrödinger equation, and the other way was non-unitarily, i.e. the evolution implied whenever a quantum system is measured.

In this lecture, we will continue this discussion, but we will generalize it to the evolution of composite systems, such as a bunch of qubits. If time permits, we will also discuss the Einstein-Podolsky-Rosen (EPR) paradox and one way (but certainly not the only way) of thinking about measurements and quantum mechanics more generally.

4.1. The Evolution of Composite Systems

Definition 4.1. Let A and B be $N_A \times M_A$- and $N_B \times M_B$-dimensional complex-valued matrices, respectively. The *tensor product of A and B*,[1] denoted $A \otimes B$, is the $N_A N_B \times M_A M_B$-dimensional complex-valued matrix

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1M_A}B \\ a_{21}B & a_{22}B & \cdots & a_{2M_A}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{N_A1}B & a_{N_A2}B & \cdots & a_{N_AM_A}B \end{pmatrix}.$$

Fact 4.1 (Useful Properties of the Tensor Product of Matrices)**.**

(i) $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$

(ii) $(A + B) \otimes C = A \otimes C + B \otimes C$

[1] Technically this is the *Kronecker product*, which is a special case of the tensor product.

(iii) $A \otimes (B + C) = A \otimes B + A \otimes C$

(iv) $(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger$

Exercise 4.1. *Using the rules above, prove the two statements below.*

(i) If $A \in \mathrm{U}(N)$ *and* $B \in \mathrm{U}(M)$*, then* $A \otimes B \in \mathrm{U}(NM)$*.*

(ii) If $A \in \mathrm{U}(N)$*,* $B \in \mathrm{U}(M)$*,* $|\psi\rangle \in \mathbb{C}^N$*, and* $|\phi\rangle \in \mathbb{C}^M$*, then*

$$(A \otimes B)(|\psi\rangle \otimes |\phi\rangle) = (A|\psi\rangle) \otimes (B|\phi\rangle).$$

Postulate 4.1. Let $S_1, \ldots, S_m$ be quantum systems with Hilbert spaces $\mathcal{H}_1 = \mathbb{C}^{N_1}, \ldots, \mathcal{H}_m = \mathbb{C}^{N_m}$, respectively, so that the composite system $S = S_1 + S_2 + \cdots + S_m$ has Hilbert space

$$\begin{aligned}
\mathcal{H}_S &= \mathcal{H}_{S_1+S_2+\cdots+S_m} \\
&= \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_m = \bigotimes_{i=1}^{m} \mathcal{H}_i \\
&= \mathbb{C}^{N_1} \otimes \mathbb{C}^{N_2} \otimes \cdots \otimes \mathbb{C}^{N_m} = \bigotimes_{i=1}^{m} \mathbb{C}^{N_i} \\
&= \mathbb{C}^{N_1 N_2 \ldots N_m}.
\end{aligned}$$

Supposing no subsystem of S is being measured, if at time t_1 the state of S is $|\psi(t_1)\rangle$ and if at time $t_2 \geq t_1$ the state of S is $|\psi(t_2)\rangle$, then there exists $U \in \mathrm{U}(N_1 N_2 \cdots N_m)$ such that

$$|\psi(t_2)\rangle = U|\psi(t_1)\rangle.$$

In other words, every quantum state $|\psi\rangle \in \mathcal{H}_S$ evolves in time unitarily.

4.2. Measuring Composite Systems

Postulate 4.2. Let S be a composite quantum system with Hilbert space $\mathcal{H}_S$, and let T be a subsystem of S with Hilbert space $\mathcal{H}_T \subseteq \mathcal{H}_S$. To "measure the subsystem T of S" means to apply a Hermitian operator M to the subspace $\mathcal{H}_T$. If S is in state $|\psi\rangle$ and if $M = \sum_i \lambda_i \Pi_i$ is M's spectral decomposition, then the probability of measuring the subsystem T of S to be in state $|\lambda_i\rangle$ is,

$$\Pr[\lambda_i] = \langle\psi|(\Pi_i \otimes I_{S \setminus T})|\psi\rangle,$$

where $I_{S\backslash T}$ denotes the identity operator on the subsystem $S\backslash T$ (i.e., the subsystem of S that you get when you remove T). Immediately after the measurement, the state of S "collapses" to

$$\frac{(\Pi_i \otimes I_{S\backslash T})|\psi\rangle}{\sqrt{\Pr[\lambda_i]}}.$$

Example 4.1 (The EPR Paradox)**.** Consider two people, Alice and Bob, each with a qubit in hand, that are spatially separated by a distance d. Suppose, further, that the joint state of their qubits is the entangled *Bell state*

$$\begin{aligned}|\Phi^+\rangle &:= \frac{1}{\sqrt{2}}\big(|0\rangle|0\rangle + |1\rangle|1\rangle\big) \\ &= \frac{1}{\sqrt{2}}\big(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle\big).\end{aligned}$$

Here, the left ket in each term refers to Alice's qubit and the right ket refers to Bob's qubit.

Suppose Alice performs a computational basis measurement of her qubit and Bob does nothing. This means that, overall, the Hermitian operator describing the measurement is

$$\begin{aligned}Z \otimes I_2 &= \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix} \otimes I_2 \\ &= \underbrace{\big(|0\rangle\langle 0| - |1\rangle\langle 1|\big)}_{\text{spectral decomp of } Z} \otimes I_2 \\ &= |0\rangle\langle 0| \otimes I_2 - |1\rangle\langle 1| \otimes I_2,\end{aligned}$$

where I_2 is the 2×2 identity matrix.

By the postulate above, the probability that Alice measures her subsystem in

state $|0\rangle$ (which corresponds to the $+1$ eigenvalue of Z) is

$$\begin{aligned}
\Pr[\text{Alice measures } |0\rangle] &= \langle\Phi^+|\Big(|0\rangle\langle 0| \otimes I_2\Big)|\Phi^+\rangle \\
&= \langle\Phi^+|\Big(|0\rangle\langle 0| \otimes I_2\Big)\frac{1}{\sqrt{2}}\Big(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle\Big) \\
&= \frac{1}{\sqrt{2}}\langle\Phi^+|\Big(|0\rangle\langle 0|0\rangle \otimes I_2|0\rangle + |0\rangle\langle 0|1\rangle \otimes I_2|1\rangle\Big) \\
&= \frac{1}{\sqrt{2}}\langle\Phi^+|\left(|0\rangle \otimes |0\rangle\right) \\
&= \frac{1}{2}\left(\langle 0| \otimes \langle 0| + \langle 1| \otimes \langle 1|\right)\left(|0\rangle \otimes |0\rangle\right) \\
&= \frac{1}{2}.
\end{aligned}$$

In this case, the overall state collapses to

$$\frac{(|0\rangle\langle 0| \otimes I_2)|\Phi^+\rangle}{\sqrt{1/2}} = |0\rangle \otimes |0\rangle.$$

On the other hand, the probability that Alice measures her subsystem in state $|1\rangle$ is

$$\Pr[\text{Alice measures } |1\rangle] = \langle\Phi^+|(|1\rangle\langle 1| \otimes I_2)|\Phi^+\rangle = \frac{1}{2}.$$

In this case, the overall state collapses to

$$\sqrt{2}(|1\rangle\langle 1| \otimes I_2)|\Phi^+\rangle = |1\rangle \otimes |1\rangle.$$

Thus, the two possible states of the composite system after Alice measures are

$$\begin{aligned}
|0\rangle \otimes |0\rangle \quad &\text{with probability } \frac{1}{2} \\
|1\rangle \otimes |1\rangle \quad &\text{with probability } \frac{1}{2}.
\end{aligned}$$

Consequently, *if* Alice measures $|0\rangle$, then she knowns with certainty that Bob, whenever he chooses to measure his side of the system, will see his qubit in state $|0\rangle$ as well. This implies that at the instant Alice measures, there becomes a definite fact of the matter of what state Bob has (either $|0\rangle$ or $|1\rangle$), and this is despite the two facts, but actually in no way in contradiction to them, that (1) Bob might be lightyears away (the distance d between Alice and Bob never showed up in this calculation) and (2) that before Alice measured, there was no fact of the matter of what Bob's state was! Thus, in some way, Alice's act of measuring her qubit *reified* Bob's qubit, and this occurred faster than the time it would take light to go from Alice and Bob. This is, in Einstein's words, "spooky action at a distance".

Discussion 4.2. How do you feel about this? Do you agree with Einstein that this is "spooky"?

Example 4.2 (Measuring all qubits of a multi-qubit system)**.** Let S be a quantum system of n qubits that is in the state[2]

$$|\psi\rangle = \sum_{x_0,x_1,\ldots,x_{n-1}\in\{0,1\}} \alpha_{x_0,x_1,\ldots,x_{n-1}} |x_0\rangle|x_1\rangle\cdots|x_{n-1}\rangle.$$

We will perform a *computational basis measurement on all n qubits*, which means to independently measure each qubit of S using the Z Pauli operator. Overall, then, we are measuring all n qubits of S using the Hermitian operator

$$Z^{\otimes n} = \bigotimes_{i=1}^{n} Z = \underbrace{Z \otimes Z \otimes \cdots \otimes Z}_{n \text{ times}},$$

which has the spectral decomposition

$$\begin{aligned} Z^{\otimes n} &= \underbrace{\big(|0\rangle\langle 0| - |1\rangle\langle 1|\big)}_{\text{spectral decomp of } Z} \otimes \cdots \otimes \underbrace{\big(|0\rangle\langle 0| - |1\rangle\langle 1|\big)}_{\text{spectral decomp of } Z} \\ &= \sum_{x_0,x_1,\ldots,x_{n-1}\in\{0,1\}} (-1)^{x_0\oplus x_1\oplus\cdots\oplus x_{n-1}} \Pi_{x_0,x_1,\ldots,x_{n-1}}, \end{aligned}$$

where $\oplus$ is addition mod 2 and

$$\Pi_{x_0,x_1,\ldots,x_{n-1}} = |x_0\rangle\langle x_0| \otimes |x_1\rangle\langle x_1| \otimes \cdots \otimes |x_{n-1}\rangle\langle x_{n-1}|$$

is the projector onto the $|x_0\rangle|x_1\rangle\cdots|x_{n-1}\rangle$ subspace of $\mathcal{H}_S$.

Now, by the postulate above, the probability that we measure the state of S to be $|y_0\rangle|y_1\rangle\cdots|y_{n-1}\rangle$ (which corresponds to the $(-1)^{y_0\oplus y_1\oplus\cdots\oplus y_{n-1}}$ eigenvalue of $Z^{\otimes n}$) is

$$\begin{aligned} \Pr[\text{state}(S) = |y_0\rangle|y_1\rangle\cdots|y_{n-1}\rangle] &= \langle\psi|\Pi_{y_0,y_1,\ldots,y_{n-1}}|\psi\rangle \\ &= \langle\psi|\Big(|y_0\rangle|y_1\rangle\cdots|y_{n-1}\rangle\Big)\Big(\langle y_0|\langle y_1|\cdots\langle y_{n-1}|\Big)|\psi\rangle \\ &= \Big|\Big(\langle y_0|\langle y_1|\cdots\langle y_{n-1}|\Big)|\psi\rangle\Big|^2 \\ &= |\alpha_{y_0,y_1,\ldots,y_{n-1}}|^2. \end{aligned}$$

[2]In this example, I do not use the tensor product symbol in the states because it crowds the notation. This will be the norm going forward, so make sure that this example makes sense!

In this case, the state of S collapses to

$$\begin{aligned}\frac{\Pi_{y_0,y_1,\ldots,y_{n-1}}|\psi\rangle}{|\alpha_{y_0,y_1,\ldots,y_{n-1}}|} &= \frac{\alpha_{y_0,y_1,\ldots,y_{n-1}}}{|\alpha_{y_0,y_1,\ldots,y_{n-1}}|}|y_0\rangle|y_1\rangle\cdots|y_{n-1}\rangle \\ &= e^{i\theta}|y_0\rangle|y_1\rangle\cdots|y_{n-1}\rangle,\end{aligned}$$

where

$$e^{i\theta} = \frac{\alpha_{y_0,y_1,\ldots,y_{n-1}}}{|\alpha_{y_0,y_1,\ldots,y_{n-1}}|}.$$

This state is operationally equivalent to $|y_0\rangle|y_1\rangle\cdots|y_{n-1}\rangle$, so the state of S has collapsed to $|y_0\rangle|y_1\rangle\cdots|y_{n-1}\rangle$.

4.3. One Way of Thinking About This*

Let us take seriously the reductionist idea that every physical system is reducible to a finite number of quantum systems. This entails that tables, chairs, as well as you and me are but a vast soup of quarks, electrons, and other Standard Model matter.

In particular, for the experiment where there is an electron e, an electron spin detector D, and a human H running the detector, there exists Hilbert spaces $\mathcal{H}_e$, $\mathcal{H}_D$, and $\mathcal{H}_H$ for the electron, detector, and human. Moreover, there are several states that the various systems can be in. For example, the electron (or rather its spin, say) can be $|0\rangle$ or $|1\rangle$, as we have discussed before. Additionally, the detector has at least three physically distinguishable states

$$|\text{ready}\rangle, \quad |\text{spin is } 0\rangle, \quad \text{and} \quad |\text{spin is } 1\rangle.$$

These correspond to the detector being in the mode "ready to measure the electron", "the detector has measured the electron to be spin 0 and is reporting it as such", and "the detector has measured the electron to be spin 1 and is reporting it as such".

Likewise, the human has at least three physically distinguishable states

$$|\text{sees "ready"}\rangle, \quad |\text{sees "spin is 0"}\rangle, \quad \text{and} \quad |\text{sees "spin is 1"}\rangle.$$

These correspond to the human seeing the detector being in mode "ready", "spin is 0", and "spin is 1", respectively.

How do these states interface with each other? Well, in the overall system which has Hilbert space $\mathcal{H}_e \otimes \mathcal{H}_D \otimes \mathcal{H}_H$, there are the obvious evolutions

$$\begin{aligned}&|0\rangle|\text{ready}\rangle|\text{sees "ready"}\rangle \mapsto |0\rangle|\text{spin is } 0\rangle|\text{sees "spin is 0"}\rangle \\ &|1\rangle|\text{ready}\rangle|\text{sees "ready"}\rangle \mapsto |0\rangle|\text{spin is } 1\rangle|\text{sees "spin is 1"}\rangle,\end{aligned}$$

which correspond to the physical experiment of measuring an electron in the spin $|0\rangle$ and $|1\rangle$ states, respectively, and the device and human having readouts of the measurements. Note, these transformations are necessarily unitary since they are automorphisms of the Hilbert space $\mathcal{H}_e \otimes \mathcal{H}_D \otimes \mathcal{H}_H$.

But now consider the experiment of measuring an electron in a spin superposition $\alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \neq 0$. Then, the overall electron, detector, and human evolution is, by the linearity of quantum mechanics,

$$(\alpha|0\rangle + \beta|1\rangle)|\text{ready}\rangle|\text{sees ``ready''}\rangle \mapsto$$
$$\alpha|0\rangle|\text{spin is } 0\rangle|\text{sees ``spin is 0''}\rangle + \beta|1\rangle|\text{spin is } 1\rangle|\text{sees ``spin is 1''}\rangle.$$

There is no collapse here, but simply a "larger", entangled superposition that now includes the detector and human.

One can continue this reasoning by including the lab, the earth, and the whole universe in the mix as well, and what one finds is that for every quantum experiment in which we "measure" a quantum system, the whole universe evolves into a big superposition of the possible outcomes of that experiment. In this way, there are "many worlds" created, each with a different outcome. This is the germ of Hugh Everett's *many-worlds interpretation of quantum mechanics*. Note that it differs from the canonical interpretation in that it predicts that for any quantum experiment, there is generically more than one measurement outcome.

Lecture 5
Qubits, Quantum Encodings, and Bell's Theorem

Discussion 5.1. Discuss with your group what you took away from last time.

Last lecture, we finished our discussion of the postulates of quantum mechanics. We also discussed the EPR paradox, which pointed toward some sort of fundamental non-locality in the structure of quantum mechanics.

In this lecture, we will transition to a more computational point of view of quantum mechanics, and discuss in detail how to encode classical information into a quantum state. We will also discuss a nice, geometric way of thinking about qubits known as the Bloch sphere. If time permits, we will discuss amplitude encodings and prove Bell's theorem.

5.1. Bits and Bit Strings

Definition 5.1.

- A *classical bit* is an element of $\{0,1\}$.
- An *n-bit string* is an element of the set

$$\{0,1\}^n \coloneqq \underbrace{\{0,1\} \times \cdots \times \{0,1\}}_{n \text{ times}}.$$

- The set of all n-bit strings is[1]

$$\{0,1\}^* \coloneqq \bigcup_{n \geq 0} \{0,1\}^n.$$

- The *length $|x|$ of a string $x \in \{0,1\}^*$* is the number of bits comprising x.

[1] Here, $*$ is known as the *Kleene star operator*, which, formally speaking, is a regular expression.

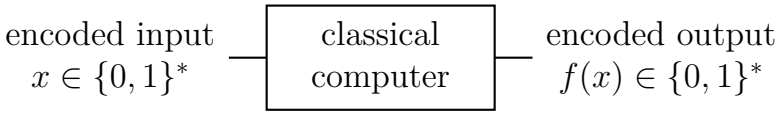
Bits are important in computation because more complicated mathematical objects (integers, graphs, Turing machines, etc.) can be *encoded* as bit strings.

Example 5.1. Given $N \in \mathbb{N}$, let $n \in \mathbb{N}$ be the number of bits you need to represent N in binary, i.e., n is the smallest positive integer such that $k \leq 2^n$.[2] Then there exist $x_0, x_1, \ldots, x_{n-1} \in \{0, 1\}$ such that

$$N = \sum_{i=0}^{n-1} x_i 2^{n-1-i}.$$

In this way, the n-bit string $x = x_0 x_1 \ldots x_{n-1}$ encodes N, where x_{n-1} is the least significant bit (LSb) and x_0 is the most significant bit (MSb). Incidentally, this is called the *MSb 0 numbering scheme*, where the "0" here specifies our indexing convention that a subscript of 0 denotes the first index.

Given a binary encoding of a mathematical object, a classical, deterministic computer (like a deterministic Turing machine, which governs complexity classes like P and PSPACE) can operate on the encoding, and hence on the mathematical object itself. In general, the procedure look something like:



The time and space usage of the classical computer is always regarded as a function of the length of the *encoded input* size $|x|$, as opposed to the "size" or "value" of the unencoded input. The reason is because the computer only ever "sees" the encoded object, not the actual, unencoded object. As the next exercise shows, this point is paramount for computational complexity.

Exercise 5.1. *Contrive a simple algorithm that finds a prime factor of a composite number $N \in \mathbb{N}$ in time $O(N)$. Is this algorithm really polynomial time?*

5.2. Basis Encodings

In quantum computing, we want to manipulate mathematical objects on a quantum computer. Because of this, we need a way to encode the mathematical object into a quantum state, so that the quantum computer can act on it. To do this, we will exploit the following observation.

[2] In general, you can show that $n = \lfloor \log_2(N-1) \rfloor + 1$.

Observation 5.1. The set $\{0,1\}$ is bijective to the computational basis over $\mathbb{C}^2$, $B(2) = \{|0\rangle, |1\rangle\}$, where the two possible bijections are

$$\begin{array}{ccc} 0 & \mapsto & |0\rangle \\ 1 & \mapsto & |1\rangle \end{array} \quad \text{and} \quad \begin{array}{ccc} 0 & \mapsto & |1\rangle \\ 1 & \mapsto & |0\rangle \end{array}.$$

Obviously, the former is the most natural, so that is what we will use in this course. This bijection implies that every bit can be encoded into a quantum state.

This observation is a particular case of the following more general observation.

Observation 5.2. For all n, $\{0,1\}^n$ is bijective to to the computational basis over $\mathbb{C}^{2^n}$,

$$B(2^n) = \left\{ \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \ldots, \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \right\}.$$

Consequently, using any of the bijections between these two sets, one can encode classical data into a computational basis state.

There are $2^n!$ bijections from $\{0,1\}^n$ to $B(2^n)$,[3] but arguably the most natural is

$$\begin{aligned} x = x_0 x_1 \ldots x_{n-1} \mapsto |x\rangle &= |x_0 x_1 x_2 \ldots x_{n-1}\rangle \\ &\coloneqq |x_0\rangle \otimes |x_1\rangle \otimes \cdots \otimes |x_{n-1}\rangle, \end{aligned}$$

where each $|x_i\rangle \in B(2)$. We will adopt this encoding convention in this course.

Exercise 5.2.

(1) Let 0^n denote the n-bit, all zero string $00\ldots0$. Write $|0^n\rangle$ in terms of $|0\rangle$.

(2) The following state is an "equal superposition" of what?

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$$

(3) Given $y \in \{0,1\}^n$, what is the probability that one obtains $|y\rangle$ if one measures all n qubits of $|\psi\rangle$ in the computational basis?

[3] *Proof:* For the first element in $\{0,1\}^n$, there are 2^n computational basis states to assign it to, for the second element in $\{0,1\}^n$, there are $2^n - 1$ computational basis states to assign it to, etc., for a total of $2^n!$ possible assignments.

Definition 5.2. Let G be a mathematical object (e.g., a graph, a number, or a Turing machine) that has a binary representation $x_G = x_0 x_1 \dots x_{n-1}$. The *basis encoding of* G is the quantum state vector

$$\begin{aligned} |G\rangle &= |x_G\rangle \\ &= |x_0 x_1 \dots x_{n-1}\rangle \\ &= |x_0\rangle \otimes |x_1\rangle \otimes \cdots \otimes |x_{n-1}\rangle. \end{aligned}$$

At the end of the day, basis encodings allow us to encode classical data into quantum states.

Example 5.2. Let N be an integer with binary expansion

$$N = \sum_{i=0}^{n-1} x_i 2^{n-1-i}.$$

Then, $x_N = x_0 x_1 \dots x_{n-1}$ encodes N in binary. The basis encoding of N is therefore

$$\begin{aligned} |N\rangle &= |x_N\rangle \\ &= |x_0\rangle|x_1\rangle \cdots |x_{n-1}\rangle. \end{aligned}$$

In this way, we have encoded the number N into a quantum system of n qubits.

5.3. Qubits and the Bloch Sphere

Qubits constitute the fundamental unit of information in quantum information theory and quantum computation. Given their importance, it is helpful to have a nice way of thinking about them. Here, we will derive a useful geometric representation of qubits called the *Bloch sphere*, which is named after the physicist Felix Bloch.

Fact 5.1. *Let S be a qubit (meaning $\mathcal{H}_S = \mathbb{C}^2$) and let $\sim$ denotes the "operationally equivalent" equivalence relation, where $|\psi\rangle \sim |\phi\rangle$ iff there exists $\theta \in [0, 2\pi)$ such that $|\psi\rangle = e^{i\theta}|\phi\rangle$. Then, the set of distinct quantum states that S can be in (i.e., the operationally inequivalent states of a qubit) is the quotient set*[4]

$$\Delta = \Big\{\alpha|0\rangle + \beta|1\rangle : \alpha, \beta \in \mathbb{C} \text{ and } |\alpha|^2 + |\beta|^2 = 1\Big\}/\sim .$$

[4]Given a set S and an equivalence relation $\sim$, the *quotient set* $S/\sim$ is the set of elements in S that are inequivalent to each other under $\sim$. That is, $S/\sim$ is the set of equivalence classes of S under $\sim$.

We would like a visual way to think about this space.

Claim 5.2. *If $|\psi\rangle \in \Delta$, then there exist unique $\theta, \phi \in [0, 2\pi)$ such that*

$$|\psi\rangle \sim \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle.$$

Proof. If $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$, then there exist unique $\sigma, \theta, \phi' \in [0, 2\pi)$ such that[5]

$$\alpha = e^{i\sigma}\cos\frac{\theta}{2} \quad \text{and} \quad \beta = e^{i\phi'}\sin\frac{\theta}{2}.$$

Therefore, if $|\psi\rangle \in \Delta$, then there exist unique $\sigma, \theta, \phi' \in [0, 2\pi)$ such that

$$\begin{aligned}
|\psi\rangle &= e^{i\sigma}\cos\frac{\theta}{2}|0\rangle + e^{i\phi'}\sin\frac{\theta}{2}|1\rangle \\
&= e^{i\sigma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle\right) \quad (\phi \coloneqq \phi' - \sigma) \\
&\sim \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle.
\end{aligned}$$

■

This representation lends itself to a rather simple geometric representation of all operationally distinct qubit states.

Claim 5.3. *There is a bijection from Δ to the unit-sphere in $\mathbb{R}^3$.*

Proof. The bijection is simply

$$|\psi\rangle \mapsto (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta),$$

where θ, ϕ are the unique numbers in $[0, 2\pi)$ such that

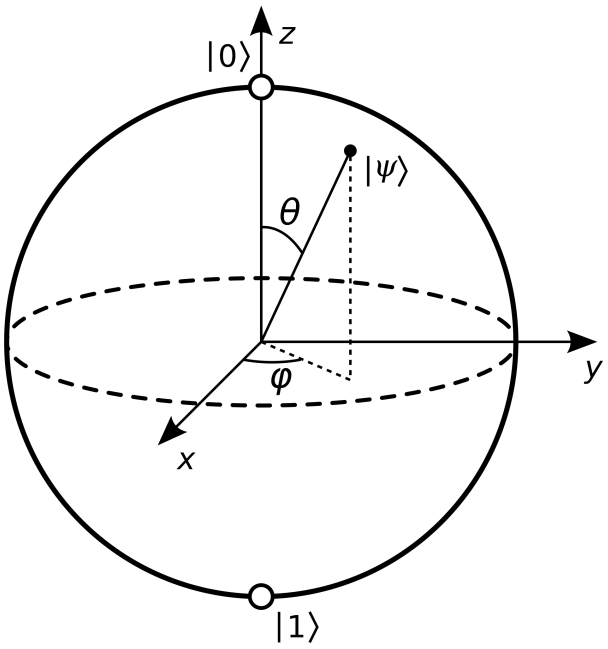
$$|\psi\rangle \sim \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle.$$

■

Exercise 5.3. *Draw the image of this bijection and label the $\mathbb{R}^3$ coordinate axes with a few states like $|0\rangle, |1\rangle, |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.*

The image of this bijection (below) is called the *Bloch sphere*:

[5] This follows from the polar representation of complex numbers.



Discussion 5.2. Can you think of some other way we might encode classical data into a quantum state?

The most obvious way to encode classical data into quantum states is to do what we did above—namely, basis encode. However, there is another way.

Definition 5.3. Let $a = (a_0, \ldots, a_{N-1})^T \in \mathbb{R}^N$ be a non-zero vector with norm $|a|$ (that may in some way encode some other mathematical object). Then, the *amplitude encoding of* a into a quantum state $|a\rangle \in \mathbb{C}^N$ is the vector

$$|a\rangle \coloneqq \frac{a_0}{|a|}|e_0\rangle + \frac{a_1}{|a|}|e_1\rangle + \cdots + \frac{a_{N-1}}{|a|}|e_{N-1}\rangle,$$

where $|e_0\rangle, \ldots, |e_{N-1}\rangle$ are the computational basis states of $\mathbb{C}^N$.

Discussion 5.3. Generally speaking, do you suspect it to be easier to basis or amplitude encode classical information? Why?

In fact, amplitude encoding is generically harder to achieve, because one needs very fine control of the unitary that will generate the encoding. For most of this course, we will therefore focus on basis encodings. That said, many important quantum algorithms, such as the HHL algorithm for solving systems of linear equations, utilize an amplitude encoding scheme.

5.5. Bell's Theorem*

Like how we can talk about the up/down spin of a qubit along the x-, y-, and z-axes, we can use the Bloch sphere to more generally talk about the (θ, ϕ)-spin of the qubit.

Definition 5.4. We say a qubit is *spin up along the* (θ, ϕ)*-axis* iff its state is (up to a global phase)

$$|\uparrow_{(\theta,\phi)}\rangle \coloneqq \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle$$

Likewise, we say a qubit is *spin down along the* (θ, ϕ)*-axis* iff its state is (up to a global phase)

$$|\downarrow_{(\theta,\phi)}\rangle \coloneqq \sin\frac{\theta}{2}|0\rangle - e^{i\phi}\cos\frac{\theta}{2}|1\rangle.$$

Exercise 5.4.

- *Prove that these states form an orthonormal basis of* $\mathbb{C}^2$*.*
- *Prove that*

$$|0\rangle \sim \cos\frac{\theta}{2}|\uparrow_{(\theta,\phi)}\rangle + \sin\frac{\theta}{2}|\downarrow_{(\theta,\phi)}\rangle$$
$$|1\rangle \sim \sin\frac{\theta}{2}|\uparrow_{(\theta,\phi)}\rangle - \cos\frac{\theta}{2}|\downarrow_{(\theta,\phi)}\rangle.$$

Now, recall from the first lecture the following definition and theorem.

Definition 5.5. Let S be a physical system with measurable properties A, B, and C, and let

$$\begin{aligned} N_S(A, B, C) &= \#\text{ times we see } S \text{ with } A, B, \text{ and } C \\ N_S(A, B, \bar{C}) &= \#\text{ times we see } S \text{ with } A, B, \text{ and NOT } C \\ N_S(A, B) &= \#\text{ times we see } S \text{ with } A \text{ and } B, \\ &\vdots \end{aligned}$$

Example 5.3.

- S is a car, A is its speed relative to the road, B is its GPS coordinates, and C is its color.

- S is a star, A is its mass, B is its luminosity, and C is its angular momentum.
- S is an electron, A is its color (a.k.a. its x-spin), B is its hardness (a.k.a. its y-spin), and C is its z-spin.

Fact 5.4. *For all systems S with measurable properties A, B, and C,*

$$N_S(A, \bar{B}) + N_S(B, \bar{C}) \geq N_S(A, \bar{C}).$$

Theorem 5.5 (A Version of Bell's Theorem)**.** *Let S be two maximally entangled electrons, e_1 and e_2, and consider the measurable properties*

- *$A =$ the spin of e_1 is up along the z-axis,*
- *$B =$ the spin of e_2 is up along the $(\theta, 0)$-axis,*
- *$C =$ the spin of e_2 is up along the $(2\theta, 0)$-axis.*

Then, for sufficiently small θ,

$$N_S(A, \bar{B}) + N_S(B, \bar{C}) < N_S(A, \bar{C}).$$

Proof. Suppose for contradiction that for all $\theta \in [0, 2\pi)$

$$N_S(A, \bar{B}) + N_S(B, \bar{C}) \geq N_S(A, \bar{C}).$$

This, of course, is what we would expect classically. The electrons e_1 and e_2 are maximally entangled, so let us suppose in particular that they are in the Bell state[6]

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle_1 |0\rangle_2 + |1\rangle_1 |1\rangle_2 \right).$$

Fact 5.6 (Verify on your own)**.** *For all θ, ϕ,*

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}} \left(|\uparrow_{(\theta,\phi)}\rangle_1 |\uparrow_{(\theta,\phi)}\rangle_2 + |\downarrow_{(\theta,\phi)}\rangle_1 |\downarrow_{(\theta,\phi)}\rangle_2 \right).$$

Therefore,

$$\begin{aligned} \Pr[B] &= \Pr[\text{state}(e_2) = |\uparrow_{(\theta,0)}\rangle] \\ &= |(I_2 \otimes \langle\uparrow_{(\theta,0)}|)|\Phi^+\rangle|^2 \\ &= \frac{1}{2}. \end{aligned}$$

[6] See Lecture 2 for the definition of this and the other three Bell states, as well as the meaning of the words "maximally entangled".

This implies

$$\Pr[\bar{B}] = 1 - \Pr[B] = \frac{1}{2},$$

so that

$$\begin{aligned} N_S(A, \bar{B}) &= M \Pr[A \mid \bar{B}] \Pr[\bar{B}] \\ &= \frac{M}{2} \Pr[A \mid \bar{B}], \end{aligned}$$

where M is the number of times we run this experiment.

Now, given $\bar{B}$, i.e., given that

$$\text{state}(e_2) = |\downarrow_{(\theta,0)}\rangle = \sin\frac{\theta}{2}|0\rangle - \cos\frac{\theta}{2}|1\rangle,$$

then $\Pr[A \mid \bar{B}] = \sin^2\frac{\theta}{2}$. Consequently,

$$N_S(A, \bar{B}) = \frac{M}{2}\sin^2\frac{\theta}{2}.$$

Similar reasoning establishes that

$$N_S(B, \bar{C}) = \frac{M}{2}\sin^2\frac{\theta}{2} \quad \text{and} \quad N_S(A, \bar{C}) = \frac{M}{2}\sin^2\theta.$$

Therefore,

$$\begin{aligned} N_S(A, \bar{B}) + N_S(B, \bar{C}) \geq N_S(A, \bar{C}) &\implies \frac{M}{2}\sin^2\frac{\theta}{2} + \frac{M}{2}\sin^2\frac{\theta}{2} \geq \frac{M}{2}\sin^2\theta \\ &\implies 2\sin^2\frac{\theta}{2} \geq \sin^2\theta. \end{aligned}$$

Is this true for all θ? Taking $0 < \theta \ll 1$ so that $\sin^2\theta \approx \theta^2$, then

$$2\frac{\theta^2}{4} \geq \theta^2 \implies \frac{1}{2} \geq 1,$$

which is a contradiction! ■

What is going on here? There are at least two fundamental assumptions that went into the statement of the theorem. One, that an electron can *simultaneously* have a definite spin about two different axes, and, two, that when we measure the spin of the electron, there is only one outcome. If you accept the many-worlds interpretation of quantum mechanics, then this second assumption is false, so you

would not expect the theorem to hold, and this result should not be surprising (its not surprising to me!). On the other hand, maybe you don't like the many-worlds interpretation. Let's see where that leaves you!

You will have to contend with the idea that an electron cannot *simultaneously* have a definite spin about two different axes. This entails that, pre-measurement, there is no fact of the matter of what the electron spin is about *any* axis. In other words, pre-measurement, it is impossible to give an accurate, local prescription of what the spin of the electron is about the $(\theta, 0)$, $(2\theta, 0)$, etc. axes, because if you could, then the inequality above would be satisfied. Therefore, it is genuinely the case (as was suggested in the EPR paradox) that any measurement of the spin will change what you see the spin to be about the other axes! Because of this, we say that there are no "local hidden variables" that describe the state of the electrons. Here, the word "local" is in reference to the fact that the two electrons in this experiment could be arbitrarily far apart. There is, therefore, a deep degree of *non-locality* in non-Everettian quantum mechanics, which is ostensibly at odds with Einstein's relativity (but in fact it is not).

Lecture 6
The Circuit Model of Classical Computation

Discussion 6.1. Discuss with your group what you took away from last time.

Last time, we saw how to encode classical data into a quantum state. We also discussed the Bloch sphere, which affords a geometric way of thinking about (operationally distinct) qubit states.

Today, we will cover the basic notions of classical circuits and formally define the terms "classical computer" and "efficient classical computer" with respect to the circuit model of classical computation.

In the next two lectures, we will refine these definitions so that, in a few lectures from now, we can formally define the terms "quantum computer" and "efficient quantum computer" with respect to the circuit model of quantum computation.

6.1. Classical Gate Sets and Circuits

Definition 6.1. A *classical gate set* is a finite set $\mathcal{B} = \{b_1, b_2, \ldots, b_t\}$ of Boolean functions $b_i : \{0,1\}^{k_i} \to \{0,1\}^{\ell_i}$, where $k_i, \ell_i \in \mathbb{N}$ are constants.

Example 6.1. $\{\text{AND}, \text{NOT}\}$, where

$$\begin{aligned} \text{AND} : \{0,1\}^2 &\to \{0,1\} \\ : (x_0, x_1) &\mapsto x_0 x_1 \end{aligned}$$

is the 2-bit logical AND gate and

$$\begin{aligned} \text{NOT} : \{0,1\} &\to \{0,1\} \\ : \quad x \quad &\mapsto 1 \oplus x \end{aligned}$$

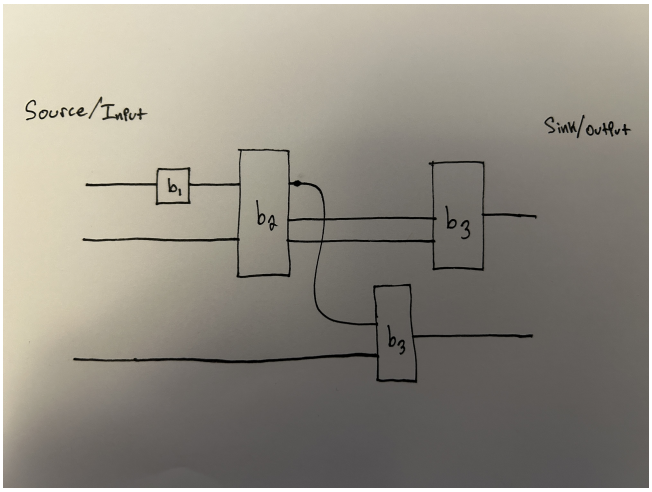
is the 1-bit logical NOT gate (and $\oplus$ is addition mod 2).

Given a classical gate set, we can build circuits over them. The following is a semi-formal definition of a "circuit". For a more formal definition, see Arora and Barak's textbook (citation in the syllabus), or any book that discusses the circuit model of classical computation.

Definition 6.2.

- Let $\mathcal{B}$ be a classical gate set. An *n-to-m-bit classical circuit C over $\mathcal{B}$* is a directed acyclic graph with n *sources* and m *sinks*. All other non-source and non-sink vertices are from $\mathcal{B}$ and are called *gates*.
- The *size* of C is the number of gates that comprise C.

Diagramatically, an example 3-to-2 bit circuit whose size equals four is:

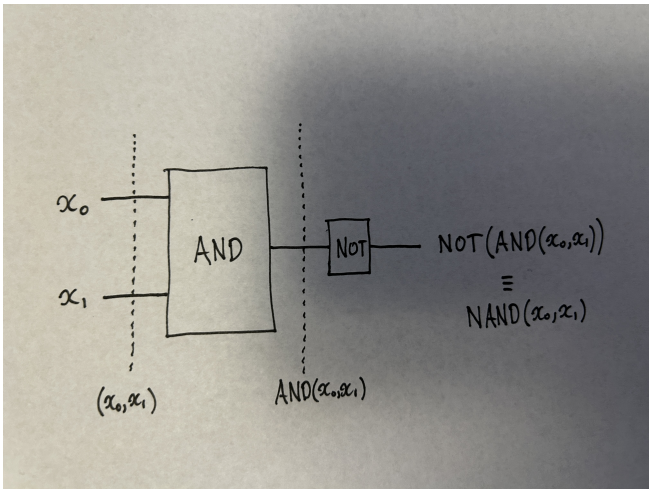


Note, here (and in all other classical or quantum circuits in this class) it is implicit that each edge or *wire* is directed from left to right.

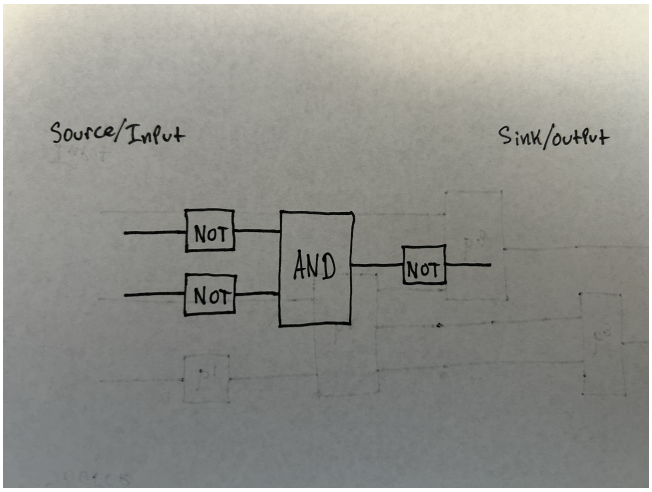
Importantly, we can use circuits to compute functions.

Definition 6.3. If C is an n-to-m bit circuit over $\mathcal{B}$ and $x \in \{0,1\}^n$, then the *output of C on input x*, denoted $C(x)$, is got by "flowing" the bits from left to right, and computing the Boolean functions from $\mathcal{B}$ as you go along (example below). We say C *computes* $f : \{0,1\}^n \to \{0,1\}^m$ iff for all $x \in \{0,1\}^n$, $C(x) = f(x)$.

Example 6.2. The following circuit computes NAND : $\{0,1\}^2 \to \{0,1\}$:



Exercise 6.1. *What function does the following circuit compute?*



6.2. Universal Classical Gate Sets

Among all possible gate sets, some are distinguished because circuits over them can compute any Boolean function.

Definition 6.4. A classical gate set $\mathcal{B}$ is *universal* (a.k.a. *functionally complete*) iff for all $f : \{0,1\}^n \to \{0,1\}^m$, there is a circuit over $\mathcal{B}$ that computes f.

Discussion 6.2. What are some examples of universal gate sets?

One ostensible example is $\{\text{AND}, \text{NOT}\}$. This is almost right.

Fact 6.1. *If $f : \{0,1\}^n \to \{0,1\}$, then there is a circuit over $\{\text{AND}, \text{NOT}\}$ that computes f.*

Proof Idea: Decompose f into conjunctive normal form (CNF), and then use De Morgan's laws to replace all ORs in the CNF with AND and NOT (in particular, $\text{OR}(x, y) = \text{NOT}(\text{AND}(\text{NOT}(x), \text{NOT}(y)))$). ■

However, $\{\text{AND}, \text{NOT}\}$ is not universal for more general Boolean functions.

Exercise 6.2. *Argue that no circuit over $\{\text{AND}, \text{NOT}\}$ computes COPY (a.k.a. FANOUT),*[1]

$$\begin{aligned} \text{COPY} : \{0,1\} &\to \{0,1\}^2 \\ : \quad x \quad &\mapsto (x, x). \end{aligned}$$

The fundamental problem with $\{\text{AND}, \text{NOT}\}$ is that neither gate can *create* bits. If we could somehow do that, then we can circumvent this issue of NOT never changing the bit length, and AND always reducing the bit length (which, incidentally, is how you solve the above exercise). As we have already seen, COPY does exactly this.

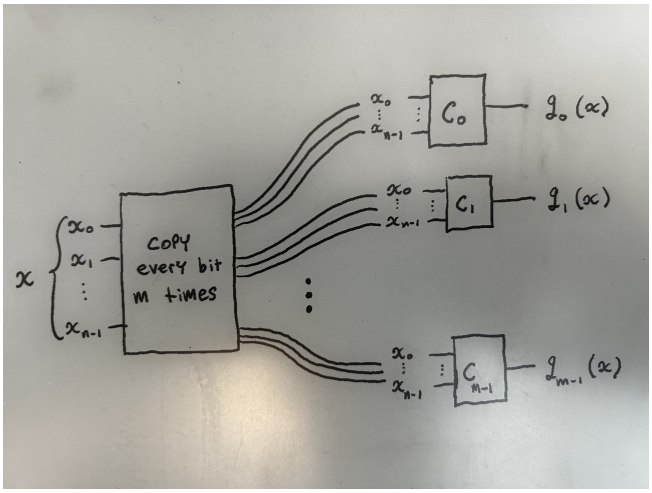
Claim 6.2. *$\{\text{AND}, \text{NOT}, \text{COPY}\}$ is universal.*

Proof. If $f : \{0,1\}^n \to \{0,1\}^m$, then there exist $g_0, g_1, \ldots, g_{m-1} : \{0,1\}^n \to \{0,1\}$ such that for all $x \in \{0,1\}^n$,

$$f(x) = (g_0(x), g_1(x), \ldots, g_{m-1}(x)),$$

Since every $g_i : \{0,1\}^n \to \{0,1\}$, there is a circuit C_i over $\{\text{AND}, \text{NOT}\}$ that computes it. Now use COPY to distribute the initial bits to each C_i, and then you compute f, as in the following circuit:

[1] In fact, it is not hard to prove the following, more general claim: If $f : \{0,1\}^n \to \{0,1\}^m$, where $n < m$, then no circuit over $\{\text{AND}, \text{NOT}\}$ computes f.



■

6.3. The Circuit Model of Classical Computation

In computation, we are ultimately interested in functions $f : \{0,1\}^* \to \{0,1\}^*$,[2] i.e., functions where the input size is *not fixed*. For example, we are interested in functions like PRIME : $\{0,1\}^* \to \{0,1\}$, which on *any* input x, outputs 1 if x encodes a prime and 0 otherwise. To this end, we want a way for a circuit to compute a function whose input size can vary.

Definition 6.5. A *circuit family C over a gate set $\mathcal{B}$* is a collection $C = \{C_n : n \in \mathbb{N}\}$ of n-to-m circuits C_n over $\mathcal{B}$, where, generally, m is a function of n. On input $x \in \{0,1\}^*$, the *output of C* is $C(x) \coloneqq C_{|x|}(x)$. We say the family C *computes* $f : \{0,1\}^* \to \{0,1\}^*$ iff for all $x \in \{0,1\}^*$, $C(x) = C_{|x|}(x) = f(x)$.

Interestingly, circuit families can compute "hard" functions.

Fact 6.3 (HW2)**.** *There exists a circuit family over* $\{\text{AND}, \text{NOT}, \text{COPY}\}$ *that solves the (uncomputable) halting problem.*

This implies that circuit families are too powerful to be a good model of computation.[3] To reduce their power to a more reasonable level, we will impose what is known as a "uniformity" condition on the circuit family. The most basic type of uniformity condition enforces that in a circuit family $C = \{C_n\}$, the map $n \mapsto C_n$

[2]Equivalently, functions $f : \mathbb{N} \to \mathbb{N}$, which is equivalent because $\{0,1\}^*$ and $\mathbb{N}$ are bijective sets.
[3]If you are interested in the computational complexity of such families, I encourage you to Google the complexity class $\mathsf{P/poly}$.

is computable. This implies that there is a Python program A such that for all $n \in \mathbb{N}$, $A(n)$ outputs a description of the circuit C_n. Interestingly, by imposing this condition, uniform circuit families exactly characterize the set of computable functions.

Fact 6.4. *$f : \{0,1\}^* \to \{0,1\}^*$ is computable on a deterministic Turing machine iff there exists a circuit family $C = \{C_n : n \in \mathbb{N}\}$ over a universal gate set $\mathcal{B}$ such that:*

(i) C computes f,

(ii) the map $n \mapsto C_n$ is computable, i.e., the family C is uniform*.*

Because of this, uniform circuit families constitute a valid model of computation. This means we can define the term "classical computer" in terms of them. Altogether, the following definition constitutes the *circuit model of classical computation*.

Definition 6.6. A *deterministic classical computer* is a pair $(C, \mathcal{B})$, where C is a uniform circuit family over a universal gate set $\mathcal{B}$.

6.4. Efficient Deterministic Classical Computers

In this class, however, we are ultimately interested in the set of functions that we can compute *efficiently* (i.e., in *polynomial time*).[4] Therefore, going forward, we will need a precise definition of what an *efficient* classical computer is in the circuit model of classical computation. Thankfully, there is a simple fact that helps motivate this definition.

Fact 6.5. *$f : \{0,1\}^* \to \{0,1\}^*$ is computable in polynomial time on a deterministic Turing machine iff there exists a circuit family $C = \{C_n : n \in \mathbb{N}\}$ over a universal gate set $\mathcal{B}$ such that:*

(i) C computes f,

(ii) the map $n \mapsto C_n$ is computable in polynomial time on a deterministic Turing machine, i.e., the family C is P-uniform *(a.k.a.* polynomial time uniform*),*

(iii) C is polynomial size*, i.e., there is a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for all n, the size of C_n is at most $p(n)$.*

[4] The conflation of "efficient" with "polynomial time" is known as the *Cobham-Edmonds thesis*.

Given this, it is natural to define an efficient deterministic classical computer in the circuit model of classical computation as follows.

Definition 6.7. An *efficient (a.k.a. polynomial time) deterministic classical computer* is a pair $(C, \mathcal{B})$, where C is a polynomial size, P-uniform circuit family over a universal gate set $\mathcal{B}$.

We will discuss efficient deterministic classical computers and the associated complexity class P in more depth when we discuss computational complexity in a few lectures.

Lecture 7
Randomized Computation

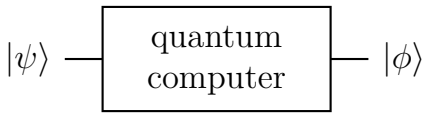
Discussion 7.1. Discuss with your group what you took away from last time.

Last time, we introduced the circuit model of classical computation. In particular, we saw that a function $f : \{0,1\}^* \to \{0,1\}^*$ is computable iff there exists a uniform circuit family over the universal gate set $\{\text{AND}, \text{NOT}, \text{COPY}\}$ that computes f. Additionally, we defined what an *efficient*, deterministic classical computer is in terms of the circuit model.

In this lecture, we will generalize these definitions to include randomness. This is important, because as we will discuss below, quantum computers are inherently probabilistic, so to compare quantum computation to classical computation, we need to make classical computers probabilistic as well.

7.1. Quantum Computers are Probabilistic

Recall our basic picture of a quantum computer:



Here, the quantum computer is a unitary operator, and on input $|\psi\rangle$ (which will in general be some computational basis state $|x\rangle$), it outputs a state $|\phi\rangle$, which can be written in the computational basis:

$$|\phi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle.$$

Therefore, in order to obtain any classical information from this state, we will have to measure it in the computational basis. But this implies that when measuring the output of a quantum computer, the bit string we get as output, say y, is only

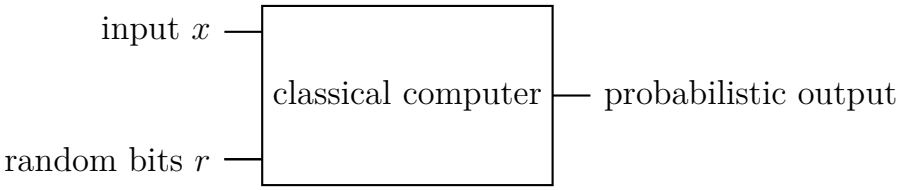
got with a particular probability, namely $|\alpha_y|^2$. Thus, in order to fairly compare quantum computers to classical computers, we have to make classical computers probabilistic as well.

Discussion 7.2. Intuitively speaking, do you think adding randomness to a classical computer will make it more or less powerful in terms of the functions it can compute? What about in terms of *how quickly* it can compute?

7.2. Probabilistic Classical Circuits

So far, our definition of a classical computer is *deterministic*. This means that on every input x, the computer outputs a particular string with probability one. However, the most powerful notion of practical, classical computer exploits randomness. This is necessarily more powerful, because randomness can only add power, as determinism is a type of randomness.

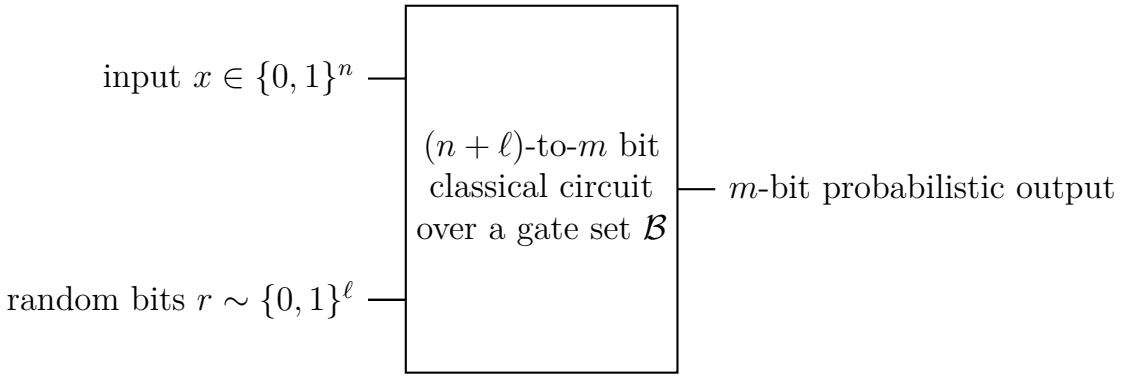
To get a randomized or probabilistic model of computation from the classical circuit model is actually very simple, and everything we have learned so far still applies. The only difference is the addition of "random bits", which are a certain number of bits that are drawn uniformly at random from $\{0,1\}$. Pictorially, a *probabilistic* (a.k.a. *randomized*) classical computer looks like:



In order to formally define the notion of a probabilistic computer, we must first discuss *probabilistic circuits*.

Definition 7.1.

- An *n-to-m bit probabilistic circuit C with $\ell \geq 0$ random bits over a gate set $\mathcal{B}$* is an $(n+\ell)$-to-m bit (deterministic) circuit over $\mathcal{B}$.
- On input $x \in \{0,1\}^n$, the *output of C* is $C(x,r)$, where $r \sim \{0,1\}^\ell$, i.e., r is an ℓ-bit string that is drawn uniformly at random from $\{0,1\}^\ell$. Pictorially,



- On input x, *the probability that* C *outputs* $y \in \{0,1\}^m$ is

$$\Pr_{r \sim \{0,1\}^\ell}[C(x,r) = y].$$

- We say C *computes* $f : \{0,1\}^n \to \{0,1\}^m$ iff for all $x \in \{0,1\}^n$,[1]

$$\Pr_{r \sim \{0,1\}^\ell}[C(x,r) = f(x)] \geq \frac{2}{3}.$$

Of course, this implies that C will sometimes err and output something other than $f(x)$. This fact is what makes probabilistic computation more powerful, at least ostensibly.[2]

Discussion 7.3. Any idea why the 2/3 here is somewhat arbitrary?

7.3. Probabilistic Classical Computers

We will now define the probabilistic analogue of a deterministic classical computer and the probabilistic analogue of an *efficient* (a.k.a. *polynomial time*) deterministic classical computer. Here, like in deterministic classical computers, we will use the notion of a circuit family. Because of this, it is important to impose a uniformity

[1] This type of randomized computation is a *Monte Carlo computation*, as opposed to a *Las Vegas computation*. In particular, we are allowing for the possibility that our circuit outputs an incorrect output, but that its "runtime" is always fixed. This is in contrast to Las Vegas algorithms, where the output is always correct, but the runtime is a random variable, so it can vary wildly. If you are interested in this, a good place to start is in understanding the differences between the complexity classes ZPP and BPP. (We will discuss BPP when we talk about quantum computational complexity theory in a few lectures from now).

[2] I write "ostensibly" here because of the *belief* that P = BPP (indeed, this is not known).

condition so that the families cannot compute uncomputable functions (such as the halting problem).

Definition 7.2.

- A *probabilistic classical computer* is a tuple $(C, \mathcal{B}, s)$, where C is a uniform circuit family over $\mathcal{B}$ and $s : \mathbb{N} \to \mathbb{N}$ is a computable function.
- We say $(C, \mathcal{B}, s)$ is *efficient* (a.k.a. *polynomial time*) iff C is a polynomial size, P-uniform circuit family and $s : \mathbb{N} \to \mathbb{N}$ is computable in polynomial time on a deterministic Turing machine.
- On input $x \in \{0,1\}^*$, the *output of* $(C, \mathcal{B}, s)$ is $C(x) = C_{|x|+|r|}(x, r)$, where $r \sim \{0,1\}^{s(|x|)}$.
- We say $(C, \mathcal{B}, s)$ *computes* $f : \{0,1\}^* \to \{0,1\}^*$ iff for all $x \in \{0,1\}^*$,

$$\Pr_{r \sim \{0,1\}^{s(|x|)}} \left[C_{|x|+|r|}(x, r) = f(x) \right] \geq \frac{2}{3}.$$

Observation 7.1. In this definition, the purpose of s is to specify the number of random bits to use in the computation for inputs x of a particular size. For example, if no random bits are to be used for all inputs, then $s(n) = 0$ for all n. This way, no circuit in the circuit family $\{C_n\}$ uses randomness. On the other hand, if we want inputs x of size n to use n^{42} random bits, then $s(n) = n^{42}$.

Exercise 7.1. *Argue that if $f : \{0,1\}^* \to \{0,1\}^*$ is computable by a deterministic classical computer, then it is computable by a probabilistic classical computer.*

Therefore, with regard to the functions they can compute, deterministic classical computers are at least as powerful as probabilistic classical computers. Importantly, this claim also goes the other way, which proves that probabilistic classical computers cannot compute "more" than their deterministic counterparts.

Claim 7.1. *$f : \{0,1\}^* \to \{0,1\}^*$ is computable by a deterministic classical computer iff it is computable by a probabilistic classical computer.*

Proof Sketch.* You proved the forward direction above. For the other direction, suppose f is computable by a probabilistic computer $(C, \mathcal{B}, s)$. It suffices to contrive a deterministic computer $(C', \mathcal{B})$ that computes f. Roughly put, the construction is as follows:

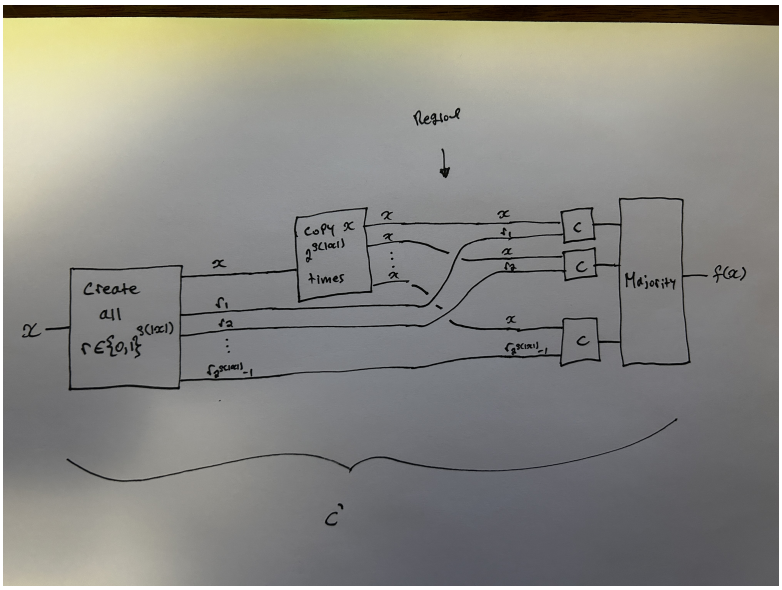
(i) On input $x \in \{0,1\}^n$ (so that $|x| = n$, C' creates all the strings in $\{0,1\}^{s(n)}$, namely, $r_0, r_1, \ldots, r_{2^{s(n)}-1}$.

(ii) C' copies x $2^s(n)$ many times.

(iii) With the $2^{s(n)}$ many pairs $(x, r_0), \ldots, (x, r_{2^{s(n)}-1})$, C' computes $C_{n+|r_i|}(x, r_i)$ for each.

(iv) Finally, C' computes the majority output of all of these circuits. Since $(C, \mathcal{B}, s)$ computes f, $\frac{2}{3}s(n)$-many of the $C_{n+|r_i|}(x, r_i)$ outputs equal $f(x)$. Therefore, the majority outputs $f(x)$.

Pictorially, this construction is as follows:



■

Therefore, randomness cannot help to compute otherwise uncomputable functions. Instead, all randomness can possibly do is expedite a deterministic computation. However, this is not known, and in fact it is generally believed to be false.

Open Problem 7.2. *If $f : \{0,1\}^* \to \{0,1\}^*$ is computable on an efficient probabilistic classical computer, is f computable on an efficient deterministic classical computer?*

We will discuss the complexity classes P and BPP, which are behind this conjecture, in more depth in a few lectures.

7.4. Probability Amplification

Lemma 7.3 (HW 2)**.** *Let C be an n-to-m bit probabilistic circuit with ℓ random bits that computes $f : \{0,1\}^n \to \{0,1\}^m$. If C is run t times on the same input $x \in \{0,1\}^n$ (but $r \sim \{0,1\}^\ell$ is drawn independently on each run), then the probability that the majority of the outputs is $f(x)$ is at least $1 - e^{-\frac{t}{18}}$.*

In particular, if on input x, we simply run the circuit $18|x|^k$ times, where $k \in \mathbb{N}$, then the probability of success satisfies

$$\Pr[\text{majority is } f(x)] \geq 1 - 2^{-|x|^k}.$$

The function 2^{-n^k} is the canonical example of a *negligible function*, which appears all over the place in computational complexity and cryptography. This is so because for any polynomial function $\text{poly} : \mathbb{N} \to \mathbb{N}$, $2^{-n^k} < 1/\text{poly}(n)$ for large enough n.

Altogether, this is to say that just by running a circuit polynomially many times (so that the whole operation is still a polynomial time operation!), then we can get the success probability of computing the function *exponentially* close to one. This is ultimately why the $2/3$ doesn't actually matter, and why in particular the same analysis works if we instead replaced $2/3$ with $1/2 + \delta$ for any fixed $\delta > 0$. We will now put this statement more formally.

Claim 7.4 (Probability Amplification)**.** *If f is computable by a probabilistic classical computer $(C, \mathcal{B}, s)$ of size $T(n)$, then for all $k \in \mathbb{N}$, there is a probabilistic classical computer $(C', \mathcal{B}, s')$ of size $\text{poly}(n, T(n))$ such that for all $x \in \{0,1\}^*$,*

$$\Pr_{r \sim \{0,1\}^{s'(|x|)}} \left[C'_{|x|+|r|}(x, r) = f(x) \right] \geq 1 - 2^{-|x|^k}.$$

In particular, if f is computable by an efficient probabilistic classical computer $(C, \mathcal{B}, s)$, then for all $k \in \mathbb{N}$, there exists an efficient probabilistic classical computer $(C', \mathcal{B}, s')$ such that for all $x \in \{0,1\}^*$, the previous equation holds.

Proof Sketch. The construction of $(C', \mathcal{B}, s')$ is as follows.

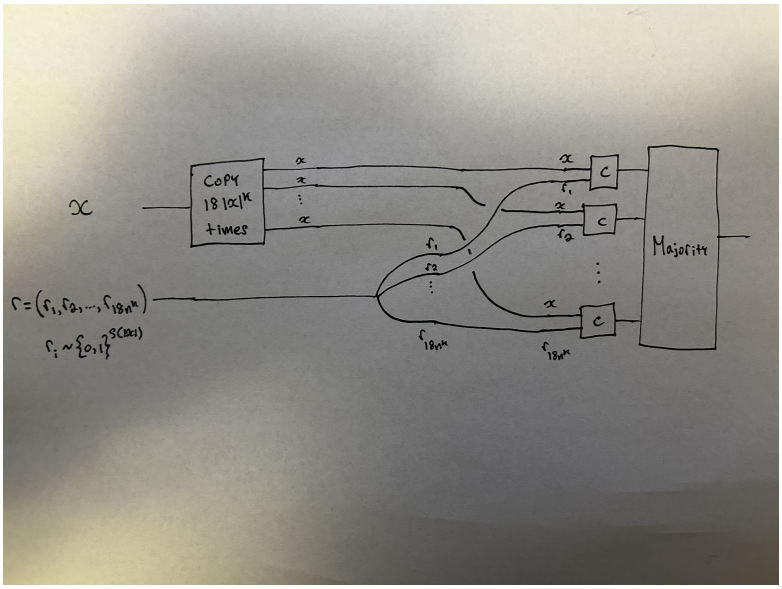
(i) Put $s'(n) = 18n^k s(n)$, so that $r \sim \{0,1\}^{s'(n)}$ can be written as an $18n^k$ component tuple $r = (r_1, r_2, \ldots, r_{18n^k})$, where $r_i \sim \{0,1\}^{s(n)}$.

(ii) On input $x \in \{0,1\}^n$ (so that $|x| = n$), C' copies x $18n^k$ times, and then pairs each x with a unique r_i. This requires $O(n^{k+1})$ COPY gates (because we copy each of the n bits of x $18n^k$ times).

(iii) C' then computes each $C_{n+|r_i|}(x, r_i)$. This takes $O(T(n)n^k)$ gates (one for each of the $18n^k$ pairs (x, r_i)).

(iv) Finally, C' computes the majority output of all of these circuits and returns that value. This can be done with $\text{poly}(n)$ gates (because one can compute the majority function in polynomial time on a deterministic Turing machine using, for example, the Boyer-Moore majority vote algorithm[3]).

By the lemma above, C' computes $f(x)$ with probability at least $1-e^{-|x|^k} \geq 1-2^{-|x|^k}$. Moreover, C' has size $\text{poly}(|x|, T(|x|))$. Pictorially, this construction looks as follows:



■

Consequently, the probability with which an efficient probabilistic classical computer computes a function can be made exponentially close to one.

[3]An alternate "trick" is to note that if the outputs are lexicographically ordered (which can be achieved using your favorite sorting algorithm), then the majority of the strings equals the median value. This gives another polynomial time way to compute the majority.

Lecture 8
Reversible Computation

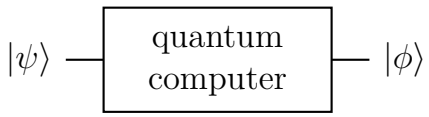
Discussion 8.1. Discuss with your group what you took away from last time.

Last lecture, we discussed probabilistic classical computers in the circuit model of classical computation. We motivated this study because quantum computers are inherently probabilistic, so if we want a fair comparison between classical and quantum computers, we better look at a *probabilistic* model of classical computation.

In this last lecture on the classical circuit model, we will frame the circuit model in such a way so that in a few lectures from now we can generalize it to define a circuit model of *quantum* computation. To do this requires making the circuit model of classical computation *reversible*.

8.1. Quantum Computers are Reversible

As we've said several times now, a quantum computer is ultimately just a map from one quantum state of one dimension to another quantum state of the same dimension:



On pain of violating the unitarity of quantum mechanics, therefore, a quantum computer is a *unitary* operator U_{QC} that implements the transformation

$$|\phi\rangle = U_{\text{QC}}|\psi\rangle.$$

One rather bizarre thing to glean from this is the following fact.

Fact 8.1. *Every quantum computation is reversible (because $U_{\text{QC}}^{-1} = U_{\text{QC}}^{\dagger}$ exists).*

This is in contrast to classical computers, which are over gate sets that are generically irreversible. Thus, given that whatever quantum computers are, they must in general be reversible, it is an interesting exercise to see if every *classical* computer can be made reversible. If not, then this would suggest that the functions that classical and quantum computers can compute are fundamentally different.

8.2. Reversible Gate Sets and Garbage Bits

Definition 8.1. $f : \{0,1\}^n \to \{0,1\}^m$ is *reversible* iff $f^{-1} : \{0,1\}^m \to \{0,1\}^n$ exists. This is possible iff f is a bijection, so $n = m$.

Question 8.1. *Are the following functions reversible or irreversible?*

- AND
- OR
- $\oplus$ *(XOR, addition mod 2)*
- NOT
- COPY

Thus, many important Boolean functions are not reversible. Fortunately, there is a simple way to make them reversible.

Claim 8.2. *Let* $\oplus$ *be bitwise* XOR.[1] *For all* $f : \{0,1\}^n \to \{0,1\}^m$ *(including irreversible* f*), the function*

$$\begin{array}{rccc} F : \{0,1\}^n \times \{0,1\}^m & \to & \{0,1\}^n \times \{0,1\}^m \\ : \qquad (x,a) & \mapsto & (x, a \oplus f(x)) \end{array}$$

is reversible for all x, a*. Moreover,* $F(x, 0^m)$ *computes* $f(x)$*.*

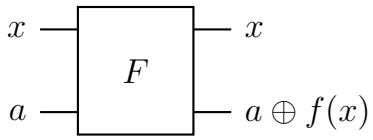
Proof. The inverse of F is F itself (involution!) because

$$F(F(x,a)) = F(x, a \oplus f(x)) = (x, a \oplus f(x) \oplus f(x)) = (x,a).$$

To compute f, compute $F(x, 0^m) = (x, f(x))$ and look at the last m bits. ■

[1] That is, for $x, y \in \{0,1\}^n$ with components $x_0, \ldots, x_{n-1}$ and $y_0, \ldots, y_{n-1}$, respectively, $x \oplus y := (x_0 \oplus y_0) \ldots (x_{n-1} \oplus y_{n-1}) \in \{0,1\}^n$, where here on the right side, $\oplus$ is the usual, 2-bit XOR.

Pictorially, F looks like

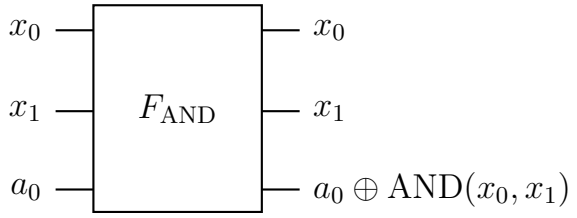


Definition 8.2. The additional string a is called the *ancilla string*. The bits that compose a are *ancilla bits*, and together they are *ancillae*.

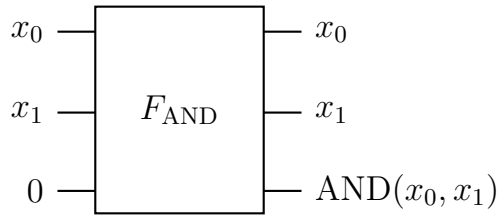
Example 8.1. To compute $\text{AND} : \{0,1\}^2 \to \{0,1\}$ reversibly, construct

$$\begin{aligned} F_{\text{AND}} : \{0,1\}^2 \times \{0,1\} &\to \{0,1\}^2 \times \{0,1\} \\ : \quad (x_0, x_1, a_0) &\mapsto (x_0, x_1, a_0 \oplus \text{AND}(x_0, x_1)). \end{aligned}$$

Pictorially,



Therefore, $F_{\text{AND}}(x_0, x_1, 0) = (x_0, x_1, \text{AND}(x_0, x_1))$, i.e.,

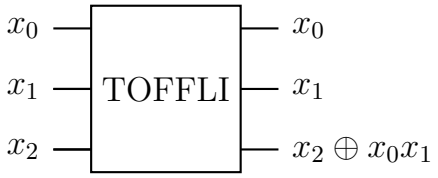


Unfortunately, the reversible counterparts of AND, NOT, and COPY are not the most natural functions (despite forming a universal, *reversible* gate set). Rather, one of the more natural and useful reversible gates is the TOFFLI *gate*.

Definition 8.3. The TOFFLI *gate* is the Boolean function

$$\begin{aligned} \text{TOFFLI} : \quad \{0,1\}^3 &\to \{0,1\}^3 \\ : (x_0, x_1, x_2) &\mapsto (x_0, x_1, x_2 \oplus x_0 x_1). \end{aligned}$$

Pictorially,

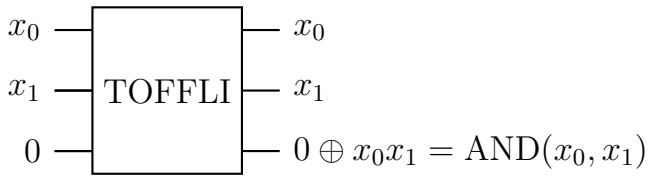


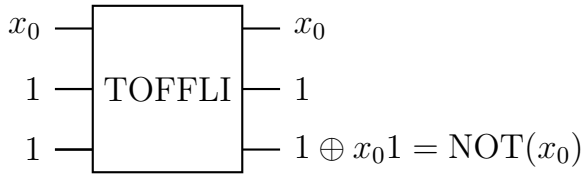
Exercise 8.1. *Prove* TOFFLI *is reversible.*

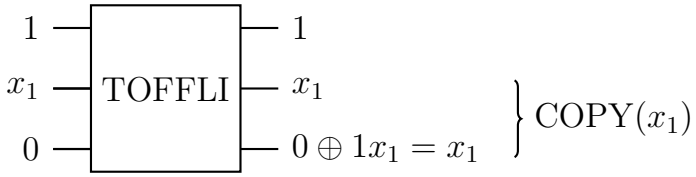
Importantly, {TOFFLI} is universal provided that the correct ancillae are used.

Claim 8.3. *With two ancilla bits,* TOFFLI *can compute* AND*,* NOT*, and* COPY*. Therefore,* {TOFFLI} *is a universal, reversible gate set.*

Proof. The following three circuits establish this claim:







■

Shortly, we will use these three identities to make every classical circuit reversible. For now, however, notice, that in the output of each of these circuits, there is the function we wanted to compute, e.g., $\mathrm{AND}(x_0, x_1) = x_0 x_1$ in the first diagram, but also additional bits, e.g., the bits x_0 and x_1 in the first diagram. These extra bits are an inevitable consequence of the reversible nature of the computation. They are examples of *garbage bits*.

Definition 8.4. A *garbage bit* is any bit in the output of a reversible computation that is not a bit in the output of the function being computed.

Classically, garbage bits are no big deal, because they can simply be reset to any value by using the reversible NOT gate. However, in quantum mechanics, garbage qubits are a huge deal, because they can be entangled with other parts of the computation, and so if they are not dealt with properly (i.e., if they are not somehow unentangled from the rest of the computation), then they can drastically change what the quantum computer is computing. To mend this issue, we will begin by seeing what a reversible circuit looks like in general.

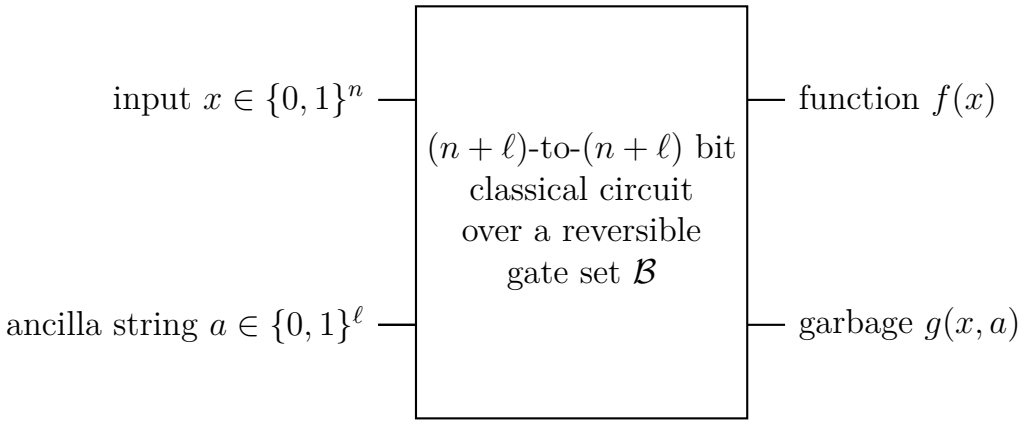
8.3. Reversible Circuits

Definition 8.5.

- An *n-bit reversible deterministic circuit C with $\ell \geq 0$ ancilla bits* is an $(n+\ell)$-to-$(n+\ell)$ bit deterministic circuit over a reversible gate set $\mathcal{B}$.
- On input $x \in \{0,1\}^n$, the *output of C* is $C(x,a)$, where a is an ℓ-bit *ancilla string* that generally depends on n.
- We say C *computes* $f : \{0,1\}^n \to \{0,1\}^m$ iff there exists an ancilla string $a \in \{0,1\}^\ell$ such that for all $x \in \{0,1\}^n$,

$$C(x,a) = C_{|x|+|a|}(x,a) = (f(x), g(x,a)).$$

Here, $g : \{0,1\}^{n+\ell} \to \{0,1\}^{n+\ell-m}$ is a *garbage function*, which groups together all the garbage bits in the output of the circuit. Pictorially,



Using the fact that TOFFLI can simulate AND, NOT, and COPY, it holds that every classical circuit can be made reversible.

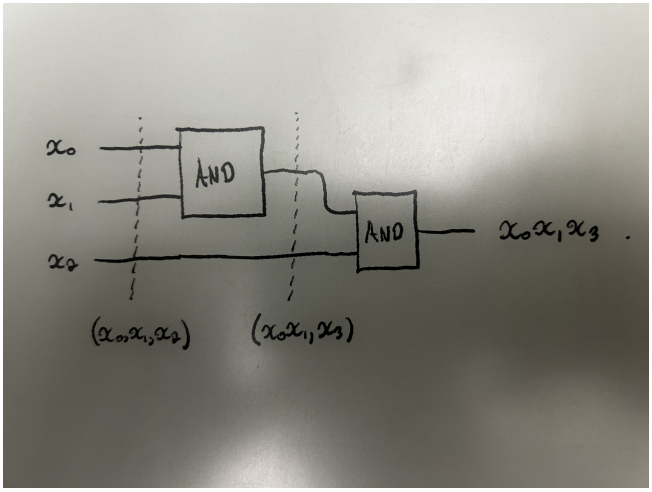
Claim 8.4. *Let C be an n-to-m-bit circuit over $\{\text{AND}, \text{NOT}, \text{COPY}\}$ of size T that computes $f : \{0,1\}^n \to \{0,1\}^m$. There is an n-bit reversible circuit C' over $\{\text{TOFFLI}\}$ of size T with $\ell = O(T)$ ancilla bits that computes f.*

Proof Sketch. Replace each gate in C with TOFFLI and the one or two ancilla bits it takes to simulate the gate being replaced. This new circuit, C', has T gates, at most $2T$ ancillae, computes f, and is reversible (as each TOFFLI is reversible). ■

Exercise 8.2. *Consider the three-bit* AND*,*

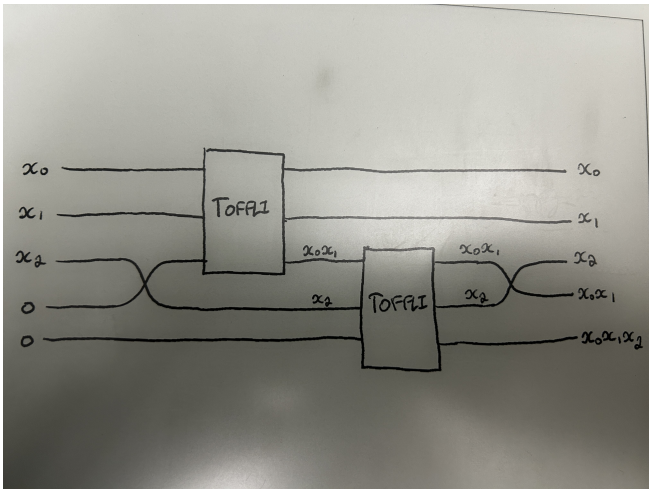
$$\text{AND}(x_0, x_1, x_2) = x_0 x_1 x_2.$$

An irreversible circuit over $\{\text{AND}, \text{NOT}, \text{COPY}\}$ *that computes this is:*



Using TOFFLI*, make this circuit reversible and find the garbage function.*

Answer: To make this circuit reversible, we insert TOFFLI for each AND, along with the appropriate ancilla, which, for each AND, is the bit 0. Overall, the reversible version of this circuit looks like:



In the output of this circuit, there is the function we wanted to compute—namely, $\text{AND}(x_0, x_1, x_2) = x_0x_1x_2$—but also the garbage bits x_0, x_1, x_2, and x_0x_1. Therefore, the garbage function is

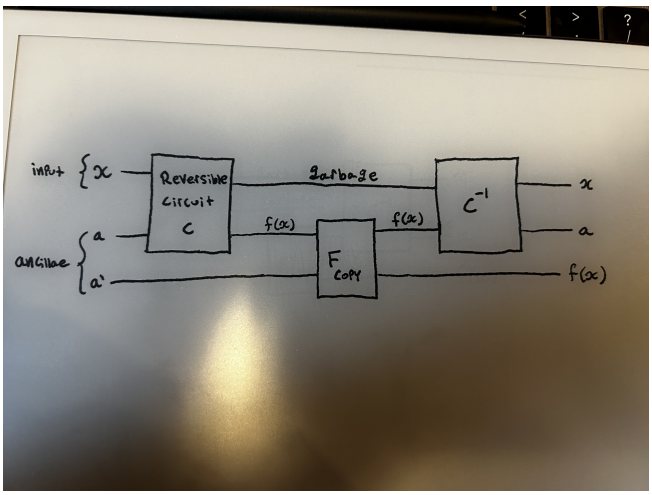
$$g(x_0, x_1, x_2, 0, 0) = (x_0, x_1, x_2, x_0x_1).$$

Interestingly, we can actually reset all of these garbage bits to their original values in the input of the circuit by "uncomputing" them.

8.4. Uncomputation

It would be nice if we could eliminate the garbage bits in any computation and only have the functional output appear. Of course, this is impossible in a reversible circuit, because the input and output size in any reversible circuit must be the same. Instead, in reversible computation the best we can do is "uncompute" the garbage bits, which is also sometimes called "Bennett's trick" after physicist Charles Bennett. The idea is to exploit the reversibility of the circuit to reset all garbage bits to their original values.

Definition 8.6. Let C be a reversible circuit that computes the function $f : \{0,1\}^n \to \{0,1\}^m$. The *uncomputed* version of C is the circuit



It is not difficult to see that if C is an n-bit reversible circuit over {TOFFLI} that has size T, uses ℓ ancillae, and computes $f : \{0,1\}^n \to \{0,1\}^m$, then the uncomputed version of C has size $O(m+T)$, uses $O(m+\ell)$ ancillae, computes f, and is reversible.

8.5. Reversible Classical Computers*

We can now define what a reversible classical computer is in the classical circuit model of computation, which is basically the same as a deterministic classical computer, but with the addition of an "ancilla function".

Definition 8.7.

- A *reversible deterministic classical computer* is a tuple $(C, \mathcal{B}, a)$, where C is a uniform circuit family over a reversible gate set $\mathcal{B}$ and $a : \mathbb{N} \to \{0,1\}^*$ is a computable function.
- We say $(C, \mathcal{B}, a)$ is *efficient* (a.k.a. *polynomial time*) iff C is a polynomial size, P-uniform circuit family and a is computable in polynomial time on a deterministic Turing machine.
- On input $x \in \{0,1\}^*$, the *output of* $(C, \mathcal{B}, a)$ is
$$C(x) = C_{|x|+|a(|x|)|}(x, a(|x|)).$$
- We say $(C, \mathcal{B}, a)$ *computes* $f : \{0,1\}^* \to \{0,1\}^*$ iff for all $x \in \{0,1\}^*$,
$$C(x) = C_{|x|+|a(|x|)|}(x, a(|x|)) = (f(x), g(x, a(|x|))).$$

Here, $g : \{0,1\}^* \to \{0,1\}^*$ is the garbage function (i.e., the function that specifies the garbage bits that are outputted in the computation).

Observation 8.1.

- In this definition, the purpose of a is to specify the ancilla string to use in the computation for inputs x of a particular size. For example, if a circuit family only requires the ancilla string 01, then $a(n) = 01$ for all n (so that every circuit in the circuit family $\{C_n\}$ uses the same ancilla string 01). On the other hand, if we want inputs x of size n to use a particular ancilla string $x^\star$ of size n^{42}, then $a(n) = x^\star$. Also, we require a to be computable (or efficiently computable), in order for it to not act what is known as an *advice string*, which would allow the computation to compute uncomputable functions (in a similar way to how non-uniform circuit families can compute uncomputable functions).
- If NOT is part of the gate set $\mathcal{B}$, then one can always make the ancilla string an all zero string (because every n-bit string can be made by NOTing the string 0^n appropriately). Indeed, in the quantum setting, it is common to use the state $|0^n\rangle$ as the ancilla qubit string.
- Finally, using Bennett's trick (uncomputation), one can make the garbage function of any reversible computer output part of the input x or the ancilla string, so that these values can be reused later. This will be particularly important in the quantum circuit model. Importantly, this does not add a significant complexity overhead. In particular, if the original circuit is polynomial size, then so is the uncomputed version.

Using the ideas in this lecture, it is not difficult to prove the following theorem.

Theorem 8.5. *$f : \{0,1\}^* \to \{0,1\}^*$ is computable on a (efficient) deterministic classical computer iff it is computable on a (efficient) reversible deterministic classical computer.*

We note that with all these definitions, it is not difficult to now define a reversible, *probabilistic* classical computer. To do this, one does the "obvious" thing and inserts a random string into the reversible classical computer, and applies the definitions we learned in the last lecture. Using this modification, one can easily prove the probabilistic analogue of the above theorem.

8.6. Landauer's Principle*

There is an interesting thermodynamical interest in reversible computation, which has to do with something called *Landauer's principle*.

Landauer's Principle: If a system (e.g., a computer) is operating at a temperature T, then the energy needed to erase one bit of information is $k_B T \ln 2$, where $k_B \approx 1.38 \times 10^{-23}$ J/K is the Boltzmann constant.

A very rough argument for this is as follows. Consider a system at temperature T that can be in two states (e.g., a bit). Then, its entropy is $k_B T \ln 2$. If you remove the two states of the system (i.e., erase the bit), then the entropy is zero. So, $k_B T \ln 2$ worth of energy is dissipated into the environment.

Because of this, reversible classical computation presents the possibility of doing highly energy efficient computation. We will not talk more about this in this course, but exploring this connection would make a great final project if you are interested.

Lecture 9
Quantum Gates

Discussion 9.1. Discuss with your group what you took away from last time.

In the last lecture, we discussed reversible computation in the context of the circuit model of classical computation. In this lecture, we will begin our study of the circuit model of *quantum* computation, which by its very nature is reversible. The quantum circuit model will eventually lead us to a formal definition of a quantum computer in the next lecture. Recall how quantum computers are ultimately just a unitary operator. In this lecture, we will begin discussing the unitary operators that quantum computers are (or, rather, what unitary operators make them up).

9.1. Single-Qubit Gates

In the classical circuit model, Boolean logic gates are the basic building blocks of classical circuits. Similarly, *quantum gates* are the basic building blocks of quantum circuits. The most basic gate is known as a *single-qubit gate*.

Definition 9.1. A *single-qubit gate* U is a unitary operator in $\mathrm{U}(2)$.

Example 9.1. Extremely important examples of single-qubit gates include:

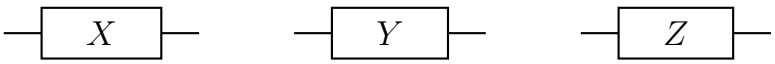
- the *Hadamard gate*,

$$H = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad \text{—}\boxed{H}\text{—}$$

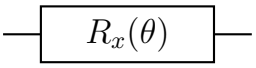
- the *Pauli gates*,

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \text{and} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

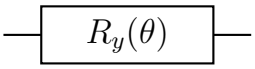
  which have the respective circuit diagrams:



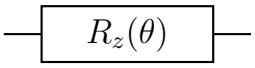
- the *x-rotation gate:*

$$R_x(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, \qquad$$ 

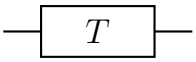
- the *y-rotation gate*:

$$R_y(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, \qquad$$ 

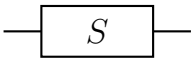
- the *z-rotation gate*:

$$R_z(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}, \qquad$$ 

- the T *gate* (a.k.a. the $\frac{\pi}{8}$ *gate*):

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, \qquad$$ 

- and the S *gate* (a.k.a. the $\frac{\pi}{4}$ *gate* or *phase gate*):

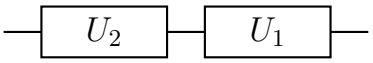
$$S = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{pmatrix} \qquad$$ 

Exercise 9.1. *Prove that X acts as a logical NOT gate when acting on the computational basis states $|0\rangle$ and $|1\rangle$.*

There are two ways to compose single-qubit gates (and in fact any more general gate) in a quantum circuit.

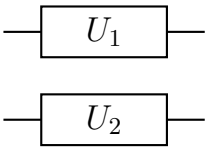
Definition 9.2. Let U_1 and U_2 be single-qubit gates.

- The *sequential composition of U_1 and U_2* is the matrix product U_1U_2. Diagramatically,

$U_1U_2 \;=$ 

Notice how the product order is flipped in the circuit (because, like in classical circuits, we imagine time as flowing from left to right).

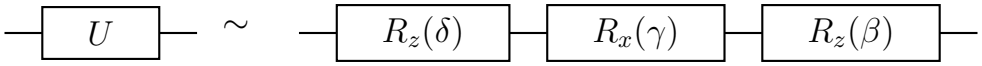
- The *parallel composition of* U_1 *and* U_2 is the tensor product $U_1 \otimes U_2$. Diagramatically,

$U_1 \otimes U_2 \;=$ 

Fact 9.1 (NC Theorem 4.1 and Exercise 4.10)**.** *For all* $U \in \mathrm{U}(2)$*, there exists* $\alpha, \beta, \gamma, \delta \in [0, 2\pi)$ *such that*

$$U = e^{i\alpha} R_z(\beta) R_x(\gamma) R_z(\delta).$$

This is called the x-z Euler decomposition of U*. Diagrammatically,*



(Again, note the order in the circuit, which is opposite to the matrix product in the statement of the fact!)

9.2. Non-Entangling Multi-Qubit Gates

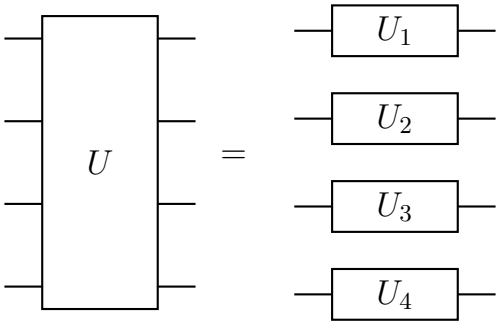
Definition 9.3. A *multi-qubit gate* is a unitary $U \in \mathrm{U}(2^n)$ for some $n > 1$.

There are several types of multi-qubit gates. We will explore these types in this and the next lecture.

Definition 9.4. A *product gate* $U \in \mathrm{U}(2^n)$ is any unitary that can be decomposed into a tensor product of single-qubit gates:

$$U = U_1 \otimes U_2 \otimes \cdots \otimes U_n, \qquad U_i \in \mathrm{U}(2).$$

For $n = 4$, this looks diagramatically as follows:



Discussion 9.2. Let $U \in \mathrm{U}(2^n)$ be a product gate. If $|\psi\rangle \in \mathbb{C}^{2^n}$ is not entangled, then $U|\psi\rangle$ is not entangled. Why?

Therefore, product gates do not create entanglement.

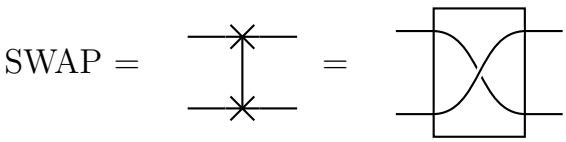
Definition 9.5. The *SWAP gate* is the 2-qubit gate

$$\mathrm{SWAP} \coloneqq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Given $x, y \in \{0, 1\}$, its action on the computational basis state $|x\rangle|y\rangle$ is what you'd expect:

$$\mathrm{SWAP}|x\rangle|y\rangle = |y\rangle|x\rangle.$$

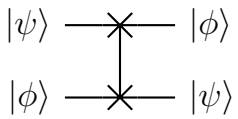
Diagrammatically, there are two ways to represent the SWAP gate:



We will use the former notation.

Importantly, this "swapping" behavior of SWAP holds for any product state.

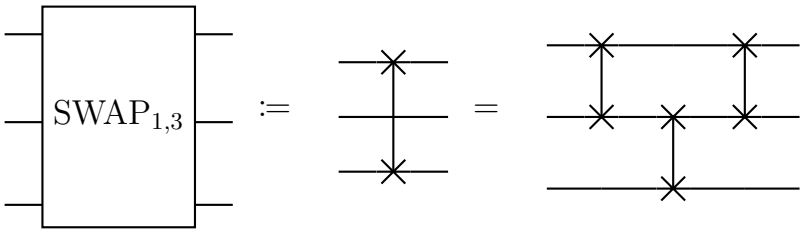
Exercise 9.2. *Prove that for all* $|\psi\rangle, |\phi\rangle \in \mathbb{C}^2$*,*



i.e.,

$$\mathrm{SWAP}|\psi\rangle \otimes |\phi\rangle = |\phi\rangle \otimes |\psi\rangle.$$

Discussion 9.3. Suppose I want to SWAP the first and third registers of a circuit. Call this operation the $\mathrm{SWAP}_{1,3}$ gate. I claim the following is true:

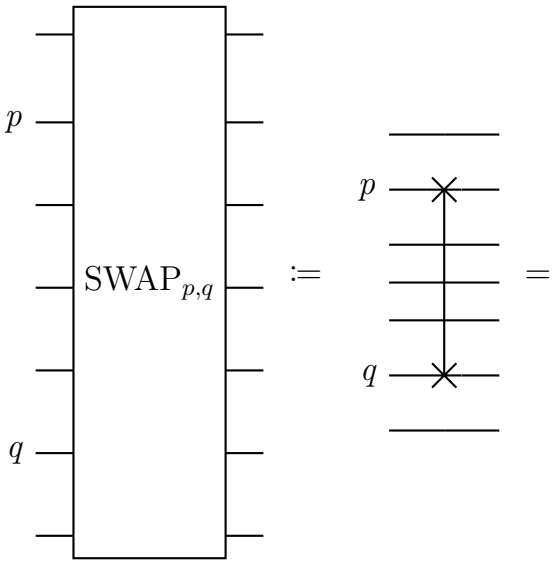


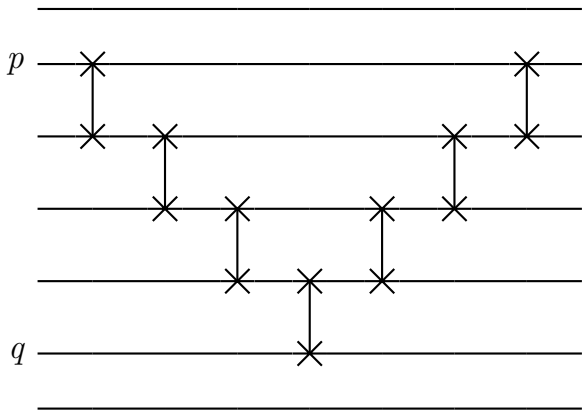
i.e.,

$$\mathrm{SWAP}_{1,3} = (\mathrm{SWAP} \otimes I_2)(I_2 \otimes \mathrm{SWAP})(\mathrm{SWAP} \otimes I_2).$$

Why?

Fact 9.2. *In general, it is possible to swap the p and q lines in a large quantum circuit (i.e., implement $\mathrm{SWAP}_{p,q}$) using a matrix and tensor product of* SWAP *and 2×2 identity gates:*





Discussion 9.4. If $|\psi\rangle \in \mathbb{C}^{2^n}$ is not entangled, then $\mathrm{SWAP}_{p,q}|\psi\rangle$ is not entangled. Why?

Therefore, SWAP, and no amount of swapping, can create entanglement.[1]

9.3. Entangling Multi-Qubit Gates

Definition 9.6. A multi-qubit gate $U \in \mathrm{U}(2^n)$ is *entangling* iff there exist a non-entangled (a.k.a. separable) state $|\psi\rangle \in \mathbb{C}^{2^n}$ such that $U|\psi\rangle$ is entangled.

Fact 9.3. *$U \in \mathrm{U}(2^n)$ is entangling iff it cannot be decomposed into a matrix and tensor product of single-qubit gates and* SWAP *gates.*[2]

Among the most important type of entangling gate is the controlled gate.

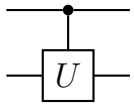
Definition 9.7. Let U be any single-qubit operator. The *(1-qubit) controlled-U gate* is the 2-qubit unitary operator

$$\begin{aligned}\Lambda^1(U) &\coloneqq |0\rangle\langle 0| \otimes I_2 + |1\rangle\langle 1| \otimes U \\ &= \begin{pmatrix} I_2 & 0 \\ 0 & U \end{pmatrix} \\ &= I_2 \oplus U.\end{aligned}$$

Diagrammatically,

[1] However, you can *transfer* entanglement with SWAP. Look up "entanglement swapping" if you are interested.

[2] The proof of this is rather involved, but the idea is to show that the normalizer of $\mathrm{SU}(2)^{\otimes n}$ in $\mathrm{SU}(2^n)$ is the semidirect product of $\mathrm{SU}(2)^{\otimes n}$ with the symmetric group S_n (the matrix representation of which is generated by SWAP). See this stack exchange post for more details.

$\Lambda^1(U) =$ 

The qubit on the line with the dot in the above diagram is called the *control qubit* and the qubit on the line with U is called the *target qubit*.

As we will shortly see, controlled operations allow us to conditionally apply a unitary operation.

Example 9.2.

- $\Lambda^1(X)$ (a.k.a. the *controlled-X* or *controlled-NOT gate*),

$$\text{CNOT} \coloneqq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = I_2 \oplus X$$

- $\Lambda^1(Z)$ (a.k.a. the *controlled-Z gate*),

$$\text{CZ} \coloneqq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} = I_2 \oplus Z.$$

Claim 9.4. *Let U be a single-qubit unitary and $x, y \in \{0, 1\}$. Then,*

$$\Lambda^1(U)|x\rangle|y\rangle = |x\rangle \otimes (U|y\rangle)$$

if and only if $x = 1$.

Proof. The action of $\Lambda^1(U)$ on the computational basis states is:

$$\begin{aligned} \Lambda^1(U)|0\rangle|0\rangle &= |0\rangle|0\rangle \\ \Lambda^1(U)|0\rangle|1\rangle &= |0\rangle|1\rangle \\ \Lambda^1(U)|1\rangle|0\rangle &= |1\rangle \otimes (U|0\rangle) \\ \Lambda^1(U)|1\rangle|1\rangle &= |1\rangle \otimes (U|1\rangle). \end{aligned}$$

■

Therefore, a controlled-U operation allows us to conditionally apply a gate, as claimed above. We condition on the control qubit and, supposing the condition is met (namely, the state of the control if $|1\rangle$), then the target qubit is affected by U. Controlled operations appear in almost every interesting quantum algorithm we know.

Exercise 9.3.

(i) Prove that if $x \in \{0, 1\}$*, then*

$$\text{CNOT}|x\rangle|0\rangle = |x\rangle|x\rangle.$$

Conclude that CNOT *together with the "ancilla qubit"* $|0\rangle$ *implements* COPY*.*

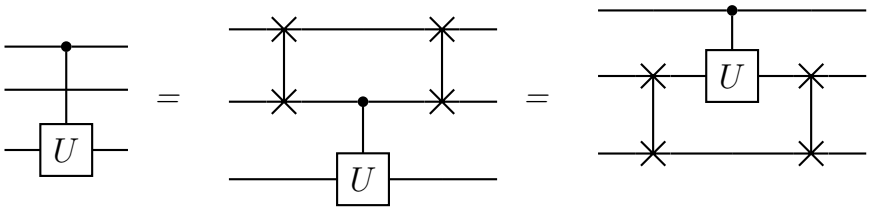
(ii) Prove that

$$\text{CNOT}|+\rangle|0\rangle = |\Phi^+\rangle.$$

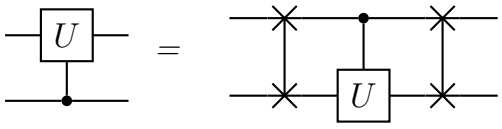
Conclude that CNOT *is entangling.*

Fact 9.5.

- *With* SWAP*, the lines to which a controlled operation is applied can be rearranged, e.g.,*



- *This trick also applies to any multi-qubit operation. For example, the control and target of a controlled operation can be interchanged,*

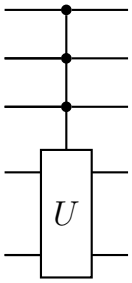


More generally, one can speak of "higher dimensional" controlled operations.

Definition 9.8. Let $U \in \mathrm{U}(2^n)$. The *m-qubit controlled-U operation* is the $2^{n+m} \times 2^{n+m}$ unitary matrix

$$\begin{aligned}\Lambda^m(U) &\coloneqq \left(\sum_{x \in \{0,1\}^m \setminus \{1^m\}} |x\rangle\langle x|\right) \otimes I_{2^n} + |1^m\rangle\langle 1^m| \otimes U \\ &= \begin{pmatrix} I_{2^n(2^m-1)} & 0 \\ 0 & U \end{pmatrix} \\ &= I_{2^n(2^m-1)} \oplus U.\end{aligned}$$

Diagrammatically, (for $m = 3$ control qubits and $n = 2$ target qubits):

$\Lambda^3(U) =$ 

Indeed, $\Lambda^m(U)$ like the 1-qubit controlled gates from before, except generalized to m controls. You should try to prove the following fact if you are skeptical of this claim.

Fact 9.6. *Let $U \in \mathrm{U}(2^m)$, $x \in \{0,1\}^m$, and $y \in \{0,1\}^n$. Then,*

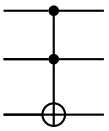
$$\Lambda^m(U)|x\rangle|y\rangle = |x\rangle \otimes (U|y\rangle)$$

if and only if $x = 1^m$.

Example 9.3. The main higher dimensional controlled operation we will use in this course is the three-qubit $\Lambda^2(X)$ gate, which is also called the *controlled-controlled-NOT* gate:

$$\begin{aligned}\mathrm{CCNOT} &\coloneqq (|00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10|) \otimes I_2 + |11\rangle\langle 11| \otimes X \\ &= \begin{pmatrix} I_6 & 0 \\ 0 & X \end{pmatrix} \\ &= I_6 \oplus X.\end{aligned}$$

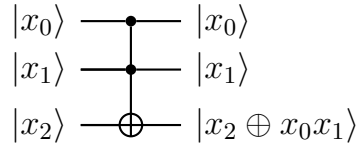
Diagrammatically,

CCNOT = 

Exercise 9.4. *For $x_0, x_1, x_2 \in \{0, 1\}$, prove that*

$$\text{CCNOT}|x_0x_1x_2\rangle = |x_0\rangle|x_1\rangle|x_2 \oplus x_0x_1\rangle,$$

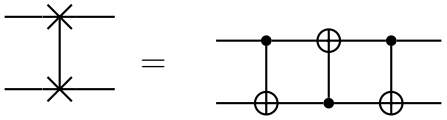
that is,



Does this look familiar? Perhaps now you have some idea for how to prove that quantum computers (whatever they are) can compute anything that a classical computer can!

We'll end with an important identity that relates SWAP to CNOT. This makes a good (albeit simple) exercise.

Fact 9.7. SWAP *decomposes into the following product of* CNOT *gates:*



Next time, we will discuss the circuit model of quantum computation and formally define a quantum computer.

Lecture 10
The Circuit Model of Quantum Computation

Discussion 10.1. Discuss with your group what you took away from last time.

Last time, we discussed quantum gates, which are the "Boolean gates of quantum circuits". In this lecture, we will see exactly how this analogy works, and we will define a quantum computer with respect to the circuit model of quantum computation.

10.1. Quantum Circuits

We have now seen a myriad of different quantum gates. In the quantum circuit model, quantum gates take the role of the fundamental Boolean functions like AND and NOT.

Definition 10.1. A finite set of quantum gates $\mathcal{G}$ is called a *quantum gate set* (or just *gate set* if the "quantum" context is clear).

Example 10.1.

- The *Clifford gate set*, $\mathcal{G}_{\text{Clifford}} \coloneqq \{H, S, \text{CNOT}\}$.
- The *Clifford + T gate set*, $\mathcal{G}_{\text{Clifford}+T} \coloneqq \{T, H, S, \text{CNOT}\}$.

Definition 10.2. A *quantum register* (or just *register* if the "quantum" context is clear) is a collection of qubits.

Given a gate set, we can define quantum circuits over it. Since quantum circuits are reversible, quantum circuits necessarily use *ancilla qubits* which comprise the an *ancilla register*.

Definition 10.3.

- An *n-qubit quantum circuit Q over a gate set $\mathcal{G}$ with $\ell \geq 0$ ancilla qubits* is an operator in $\mathrm{U}(2^{n+\ell})$ that admits the matrix product decomposition

$$Q = U_d U_{d-1} \cdots U_1,$$

  where each $U_j \in \mathrm{U}(2^{n+\ell})$ admits the tensor product decomposition

$$U_j = \bigotimes_{k=1}^{m_j} g_k, \qquad g_k \in \mathcal{G} \cup \{I_2\},$$

  where $m_j \leq n + \ell$ (as U_j is an $n + \ell$ qubit unitary).

- Here, d is the *depth of Q* (the "number of layers" of Q), and the number of non-identity gates that makeup Q is the *size of Q*.

- Given an input $x \in \{0,1\}^n$ and an ancilla string $a \in \{0,1\}^\ell$, the *output of Q* is the quantum state

$$Q(x,a) = Q|x\rangle|a\rangle = \sum_{z \in \{0,1\}^{n+\ell}} \alpha_z |z\rangle.$$

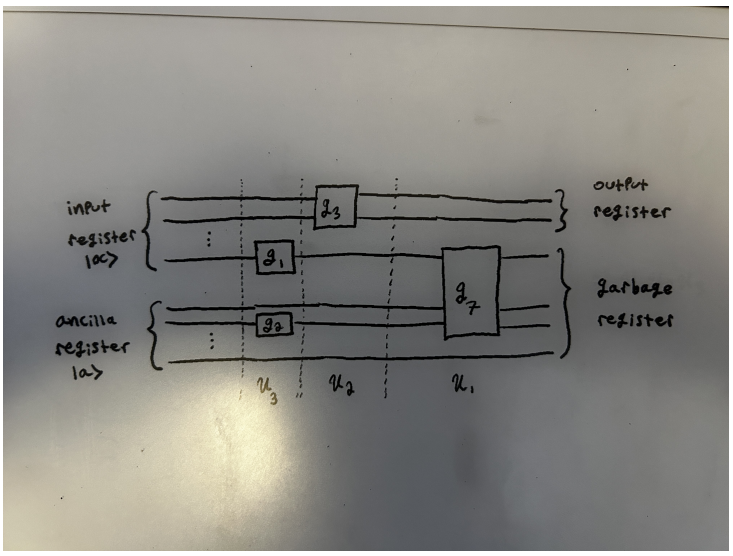
- Given an input $x \in \{0,1\}^n$ and an ancilla string $a \in \{0,1\}^\ell$, the *probability that Q outputs $y \in \{0,1\}^m$*, denoted $\Pr\left[Q^{(m)}(x,a) = |y\rangle\right]$, is the probability that a computational basis measurement of the first m qubits of $Q(x,a)$ is $|y\rangle$. In particular, with $|\psi\rangle = Q(x,a)$,

$$\Pr\left[Q^{(m)}(x,a) = |y\rangle\right] \coloneqq \langle\psi|\Pi_y \otimes_{I_{2^{n+\ell-m}}} |\psi\rangle.$$

- We say *Q computes $f : \{0,1\}^n \to \{0,1\}^m$* iff there exists $a \in \{0,1\}^\ell$ such that for all $x \in \{0,1\}^n$,

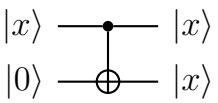
$$\Pr\left[Q^{(m)}(x,a) = |f(x)\rangle\right] \geq \frac{2}{3}.$$

Pictorially, every quantum circuit can be represented as a directed acyclic graph, where the directedness is from left-to-right, e.g.,



In this image, we have also labeled the *input register* (the collection of qubits that comprise the input), the ancilla register (the collection of qubits that comprise the ancillae), the *output register* (the collection of qubits in the output of the computation that we want to measure), and the *garbage register* (the collection of qubits in the output of the computation that we disregard).

Example 10.2. The following is a 1-qubit quantum circuit Q over $\{\text{CNOT}\}$ with $\ell = 1$ ancilla qubits (initialized to $|0\rangle$) that computes $\text{COPY} : x \mapsto (x, x)$:



This circuit has size 1, depth 1, and no garbage register. On input $x \in \{0,1\}$, the probability that Q outputs $y = y_0 y_1 \in \{0,1\}^2$ is

$$\begin{aligned}
\Pr\left[Q^{(2)}(x,0) = |y\rangle\right] &= \langle x|\langle x|\Pi_y|x\rangle|x\rangle \\
&= \langle x|\langle x|(|y_0\rangle\langle y_0| \otimes |y_1\rangle\langle y_1|)|x\rangle|x\rangle \\
&= \langle x|y_0\rangle\langle y_0|x\rangle\langle x|y_1\rangle\langle y_1|x\rangle \\
&= \begin{cases} 1 & \text{if } x = y_0 = y_1 \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

Therefore,

$$\Pr\left[Q^{(2)}(x,0) = |\text{COPY}(x)\rangle\right] = 1.$$

In this case, we say that Q computes COPY *exactly*.

Claim 10.1. *Let Q be an n-qubit quantum circuit with $\ell \geq 0$ ancillas. Then, for all $x \in \{0,1\}^n$, all $\ell \in \{0,1\}^\ell$, and all $y \in \{0,1\}^m$,*

$$\Pr\left[Q^{(m)}(x,a) = |y\rangle\right] = \sum_{w \in \{0,1\}^{n+\ell-m}} |\alpha_{y.w}|^2,$$

where . denotes string concatenation. Therefore, the probability that Q outputs $y \in \{0,1\}^m$ is a marginalized *probability, where the marginalization is over the other $n+\ell-m$-bit garbage strings in the output.*

Exercise 10.1. *Prove this. As a hint, use the fact that*

$$\begin{aligned}
|\psi\rangle &= Q(x,a) \\
&= \sum_{z \in \{0,1\}^{n+\ell}} \alpha_z |z\rangle \\
&= \sum_{u \in \{0,1\}^m} \sum_{w \in \{0,1\}^{n+\ell-m}} \alpha_{u.w} |u\rangle|w\rangle.
\end{aligned}$$

and then compute $\Pr\left[Q^{(m)}(x,a) = |y\rangle\right] \coloneqq \langle\psi|\Pi_y \otimes I_{2^{n+\ell-m}}|\psi\rangle$.

Answer:

$$\begin{aligned}
\Pr\left[Q^{(m)}(x,a) = |y\rangle\right] &\coloneqq \langle\psi|\Pi_y \otimes I_{2^{n+\ell-m}}|\psi\rangle \\
&= \left(\sum_{u,w} \alpha^*_{u.w} \langle u|\langle w|\right) \underbrace{|y\rangle\langle y|}_{\Pi_y} \otimes I_{2^{n+\ell-m}} \left(\sum_{u',w'} \alpha_{u'.w'} |u'\rangle|w'\rangle\right) \\
&= \left(\sum_{u,w} \alpha^*_{u.w} \langle u|y\rangle\langle w|\right) \left(\sum_{u',w'} \alpha_{u'.w'} \langle y|u'\rangle|w'\rangle\right) \\
&= \left(\sum_{w} \alpha^*_{y.w} \langle w|\right) \left(\sum_{w'} \alpha_{y.w'} |w'\rangle\right) \\
&= \sum_{w,w'} \alpha^*_{y.w} \alpha_{y.w'} \langle w|w'\rangle \\
&= \sum_{w} \alpha^*_{y.w} \alpha_{y.w} \\
&= \sum_{w \in \{0,1\}^{n+\ell-m}} |\alpha_{y.w}|^2.
\end{aligned}$$

10.2. The Circuit Model of Quantum Computation

Like in the classical circuit model, to compute functions with arbitrary input sizes we need a notion of a circuit *family*. Uniform families then define a quantum

computer (as non-uniform families can compute uncomputable functions, as we saw with deterministic classical circuits).

Definition 10.4.

- A *quantum computer* is a tuple $(Q, \mathcal{G}, a)$, where Q is a *uniform* family of quantum circuits over a gate set $\mathcal{G}$, and $a : \mathbb{N} \to \{0,1\}^*$ is a computable function.
- We say $(Q, \mathcal{G}, a)$ is *efficient* (a.k.a. *polynomial time*) iff Q is a polynomial size, P-uniform circuit family and a is computable in polynomial time on a deterministic Turing machine.
- On input $x \in \{0,1\}^*$, the *output of* $(Q, \mathcal{G}, a)$ is the quantum state

$$Q(x) = Q_{|x|+|a(|x|)|}(x, a(|x|)).$$

- We say $(Q, \mathcal{G}, a)$ *computes* $f : \{0,1\}^* \to \{0,1\}^*$ iff for all $x \in \{0,1\}^*$,

$$\Pr\left[Q^{(m)}(x) = |f(x)\rangle\right] \geq \frac{2}{3}.$$

  Here, m is the number of bits of the string $f(x)$, which is in general a function of the input size $|x|$. In other words, m is the number of qubits that comprise the output register of the quantum computer.

Observation 10.1.

- In this definition, the purpose of a is to specify the ancilla string to use in the computation for inputs x of a particular size. For example, if a circuit family only requires the ancilla string 01, then $a(n) = 01$ for all n (so that every circuit in the circuit family $\{Q_n\}$ uses the same ancilla state $|01\rangle$). On the other hand, if we want inputs x of size n to use a particular ancilla string $x^\star$ of size n^{42}, then $a(n) = x^\star$.
- If X is part of the gate set $\mathcal{G}$, then one can always make the ancilla qubits initialized to $|0\rangle$ (because every n-qubit computational basis state $|x\rangle$ can be realized by Xing the state $|0^n\rangle$ appropriately).

Importantly, quantum computers cannot computer functions that are classically uncomputable.

Fact 10.2 (HW3). *$f : \{0,1\}^* \to \{0,1\}^*$ is computable by a probabilistic classical computer iff f is computable by a quantum computer.*

On HW3, you will also show that *efficient* quantum computers can do everything that *efficient* classical computers can.

Fact 10.3 (HW3). *If $f : \{0,1\}^* \to \{0,1\}^*$ is computable by an efficient probabilistic classical computer, then f is computable by an efficient quantum computer.*

However, it is expected that efficient quantum computers can compute functions that no efficient classical computer can.

Conjecture 10.4. *There exists $f : \{0,1\}^* \to \{0,1\}^*$ computable by an efficient quantum computer but not computable by any efficient probabilistic classical computer.*

We will discuss this conjecture in more detail in two lectures from now when we talk about quantum computational complexity and the BPP versus BQP conjecture. It is also part of the point of this class to give some plausibility to this conjecture (by showing, for example, that there is an efficient quantum factoring algorithm, despite there not being any known efficient classical factoring algorithm).

Discussion 10.2. Given that efficient quantum computers can compute anything that efficient classical computers can, argue that efficient quantum computers can amplify the 2/3 acceptance probability to be exponentially close to one. Conclude that the probability amplification theorem holds for quantum computers.

10.3. Quantum Uncomputation*

Suppose $f : \{0,1\}^n \to \{0,1\}^m$ is *exactly* computable by an n-qubit quantum circuit Q with ℓ ancillas. This means that there is $a \in \{0,1\}^\ell$ such that for all $x \in \{0,1\}^n$,

$$\Pr\left[Q^{(m)}(x,a) = |f(x)\rangle\right] = 1.$$

Overall, then, the quantum state $Q(x,a)$ is necessarily a product state of the form

$$Q(x,a) \coloneqq Q|x\rangle|a\rangle = |f(x)\rangle|g(x,a)\rangle,$$

where $|g(x,a)\rangle$ is the $n+\ell-m$ qubit *garbage state*, i.e., the state of the garbage register.

This function $f : \{0,1\}^n \to \{0,1\}^m$ has 2^n possible inputs, and often times in quantum computing we are interested in evaluating f at all of these inputs. Of

course, classically this takes 2^n steps. Quantumly, however, it is possible to do this in a single step. The idea (which is sometimes called *quantum parallelism*) is to input the all zero string 0^n, and then apply a layer of Hadamards. This way, the overall state that is inputted to Q above is the equal superposition

$$\left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle\right)\otimes|a\rangle.$$

By linearity, then, the action of Q on this state is

$$Q\left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle\right)\otimes|a\rangle=\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|f(x)\rangle|g(x,a)\rangle.$$

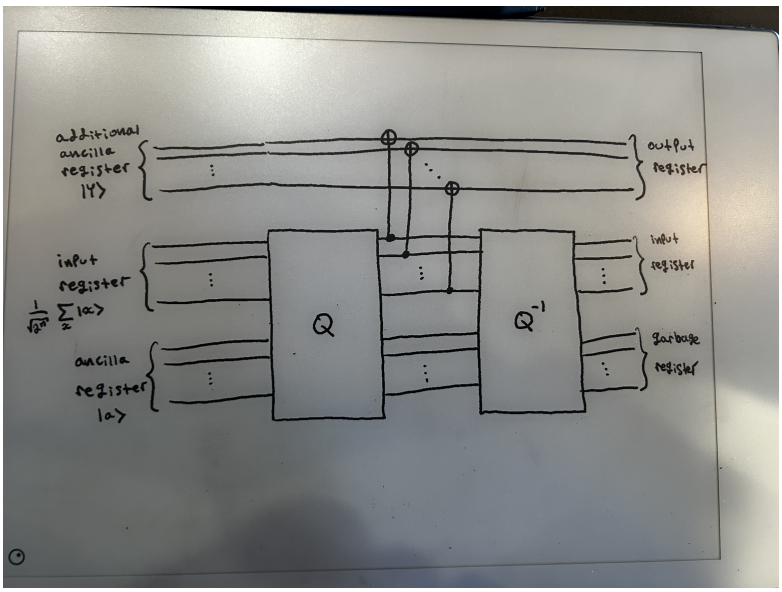
By inserting this layer of Hadamards and then applying Q, we have now evaluated f at each possible input! However, the cost is that the output and garbage registers are entangled (and of course when we measure this state, we will only see f evaluated at one input). On the surface, this is bad news, because now if the garbage register is measured (which, if we just ignore the garbage register, happens thanks to something called the *Principle of Implicit Measurement*), then the state will collapse to whatever that measurement reveals, and we lose the superposition of f evaluated at every input.

To mend this issue, we can apply Bennett's trick and uncompute the state, which in this context means to copy the output register to a new register, and then uncompute the result. However, the no-cloning theorem of quantum mechanics prevents us from copying any old quantum state. Nevertheless, there is a simple trick that works, and it relies on the fact that we *know* what the state we want to copy looks like.

Recall that the CNOT gate is such that for all $x, y \in \{0,1\}$,

$$\text{CNOT}|x\rangle|y\rangle=|x\rangle|y\oplus x\rangle.$$

Therefore, if $y=0$, then CNOT copies the value of the computational basis state to the second qubit, provided, again, that the second qubit is initialized to $|0\rangle$. By inserting an additional ancilla string initialized to $|0^m\rangle$, we can therefore copy the m qubits in the output register of a quantum circuit into a different register, and then uncompute the other registers. More generally, we can insert an additional ancilla string initialized to $|y\rangle$, $y\in\{0,1\}^m$, and create the state $|y\oplus f(x)\rangle$. This only works, though, if we know what the states in the register we are copying are (which means we know their amplitudes, which we do—they are all $\sqrt{2^{-n}}$). Pictorially,



With this construction, it is not hard to prove the following fact.

Fact 10.5. *Let Q be an n-qubit quantum circuit with ℓ ancilla qubits that computes $f : \{0,1\}^n \to \{0,1\}^m$ exactly. Let Q' be its uncomputed counterpart, which has $\ell + m$ ancilla qubits, and is depicted pictorially above. Then,*

$$Q'\left(|y\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \otimes |a\rangle\right) = \left(\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |y \oplus f(x)\rangle|x\rangle\right) \otimes |a\rangle.$$

Uncomputation, therefore, allows us to completely reset the original ancilla register to its original value (and thereby unentangle it from the computation). Of course, here the input register remains entangled with the output, but that is manageable (and not always a bad thing, because we might need to reuse the input later in the computation anyway). We will see explicit examples of uncomputation in the quantum algorithms that we study later in this course.

Lecture 11
Universal Gate Sets and Quantum Compilation

Discussion 11.1. Discuss with your group what you took away from last time.

Last time, we introduced quantum gates sets, quantum circuits, and we formally defined a quantum computer in the quantum circuit model. In this lecture, we will introduce the notion of a *universal* gate set, which is the quantum analogue of a universal gate set in classical computation (a.k.a. a functionally complete set). To do this will require some new mathematics, which we now discuss.

11.1. The Operator Norm and Generating Sets

Definition 11.1. Let M be a complex-valued square matrix. The *operator norm of* M, denoted $\|M\|_{\text{op}}$, is[1]

$$\|M\|_{\text{op}} \coloneqq \max_{\||\psi\rangle\|=1} \|M|\psi\rangle\|.$$

Roughly speaking, the operator norm of M is the maximum factor by which M can scale a unit vector.

Example 11.1. Take the $N \times N$ identity matrix I_N:

$$\begin{aligned}\|I_N\|_{\text{op}} &= \max_{\||\psi\rangle\|=1} \|I_N|\psi\rangle\| \\ &= \max_{\||\psi\rangle\|=1} \||\psi\rangle\| \\ &= \max_{\||\psi\rangle\|=1} 1 \\ &= 1.\end{aligned}$$

Intuitively, this makes sense because the identity matrix does not scale any vector.

[1]Technically, this should be a *supremum*, but we need not worry about that subtlety here.

Exercise 11.1. *Prove that for all* $U \in \mathrm{U}(N)$*,*

$$\|U\|_{\mathrm{op}} = 1.$$

Importantly, the operator norm is indeed a norm, so it induces a *metric* or *distance function* among all pairs of $N \times N$ complex-valued matrices. This allows us to formalize a notion of "closeness" among the unitary matrices.

Definition 11.2. Let U and V be $N \times N$ complex-valued matrices. The *operator distance between* U *and* V is the real number

$$d_{\mathrm{op}}(U, V) \coloneqq \|U - V\|_{\mathrm{op}}.$$

Shortly, we will use the operator distance to define a universal quantum gate set. It will allow us to say if a gate set is able to generate a unitary that is "close" to the target unitary we are trying to implement. To do this formally, however, requires one more mathematical notion.

Definition 11.3. Let $\mathcal{G}$ be a finite subset of $\mathrm{U}(N)$.

(i) The *set generated by* $\mathcal{G}$ is the set

$$\langle \mathcal{G} \rangle \coloneqq \{p : p \text{ is a finite matrix product of elements from } \mathcal{G}\}.$$

For example,

$$\langle H \rangle = \{H^0, H^1, H^2, H^3, \dots\} = \{I_2, H\}.$$

(ii) We say $\langle \mathcal{G} \rangle$ is *dense in* $\mathrm{U}(N)$ iff for all $\epsilon > 0$ and all $U \in \mathrm{U}(N)$, there exists $p \in \langle \mathcal{G} \rangle$ such that $d_{\mathrm{op}}(U, p) < \epsilon$. In other words, $\langle \mathcal{G} \rangle$ is dense in $\mathrm{U}(N)$ iff a finite matrix product p of the elements in $\mathcal{G}$ can approximate any unitary arbitrarily well (with respect to the operator distance).

Exercise 11.2.

(i) Find $\langle S \rangle$*, where* $S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$*. Is* $\langle S \rangle$ *dense in* $\mathrm{U}(2)$*?*

(ii) Could a finite *set* $\mathcal{G}$ *generate* $\mathrm{U}(N)$*? That is, could* $\langle \mathcal{G} \rangle = \mathrm{U}(N)$*?*

Consequently, unlike Boolean gates where exact synthesis of any Boolean function is always possible, the best we can hope for in quantum computing is that our gate set $\mathcal{G}$ can *approximate* any unitary arbitrarily well. Indeed, as we will now discuss, this is the usual notion of universality that is employed.

11.2. Universality and the Clifford + T Gate Set

Definition 11.4. A gate set $\mathcal{G}$ is a *universal* iff for all $\epsilon > 0$ and all $U \in \mathrm{U}(2^n)$, there exists an n-qubit circuit Q over $\mathcal{G}$ (with no ancillae) such that $d_{\mathrm{op}}(U, Q) < \epsilon$.

Interestingly, there is a nice sufficient condition for universality, which some even cite as the definition of universality. It is "nice" in the sense that to get universality, it suffices to have small, single-qubit gates that are universal, together with at least one entangling gate (such as CNOT).

Fact 11.1. *If $\mathcal{G}$ is a gate set that contains an entangling gate E and $\langle \mathcal{G} \cap \mathrm{U}(2) \rangle$ is dense in $\mathrm{U}(2)$, then $\mathcal{G}$ is universal.*

We will not prove this fact, but it ultimately follows from the fact that any unitary $U \in \mathrm{U}(2^n)$ can be decomposed into a circuit consisting of just single-qubit gates and an entangling gate E of your choice. See Nielsen and Chuang's *Quantum Computing and Quantum Information* for details.

Many important universal gate sets involve the Clifford gate set, $\mathcal{G}_{\mathrm{Clifford}}$, which we introduced last lecture,

$$\mathcal{G}_{\mathrm{Clifford}} \coloneqq \{H, S, \mathrm{CNOT}\}.$$

Interestingly, however, $\mathcal{G}_{\mathrm{Clifford}}$ is not universal.

Claim 11.2. *$\mathcal{G}_{\mathrm{Clifford}}$ is not universal.*

Proof.* Suppose $\mathcal{G}_{\mathrm{Clifford}}$ is universal. Then, for all $0 < \epsilon < 0.16$, there exists $Q \in \langle H, S \rangle$ such that $d_{\mathrm{op}}(U, Q) < \epsilon$, where U is such that $U|0\rangle = \frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle$. To obtain a contradiction, first note that for all $\theta \in [0, 2\pi)$,

$$e^{i\theta} U|0\rangle \notin \mathcal{S} \coloneqq \Big\{|0\rangle, |1\rangle, |+\rangle, |-\rangle, |+i\rangle, |-i\rangle\Big\},$$

where $|\pm i\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm i|1\rangle)$. Now note that all the states in $\mathcal{S}$ can be realized by some $Q \in \langle H, S \rangle$:[2]

$$\begin{aligned} |0\rangle &= I_2|0\rangle & |1\rangle &= HS^2H|0\rangle \\ |+\rangle &= H|0\rangle & |-\rangle &= S^2H|0\rangle \\ |+i\rangle &= SH|0\rangle & |-i\rangle &= S^3H|0\rangle \end{aligned}$$

[2]Incidentally, the states in $\mathcal{S}$ are known as *stabilizer states*.

Finally, note the action of H and S on the states in $\mathcal{S}$:

$$\begin{aligned}
H|0\rangle &= |+\rangle & S|0\rangle &= |0\rangle \\
H|1\rangle &= |-\rangle & S|1\rangle &= i|1\rangle \sim |1\rangle \\
H|+\rangle &= |0\rangle & S|+\rangle &= |+i\rangle \\
H|-\rangle &= |1\rangle & S|-\rangle &= S|-i\rangle \\
H|+i\rangle &= e^{i\pi/4}|-i\rangle \sim |-i\rangle & S|+i\rangle &= |-\rangle \\
H|-i\rangle &= e^{-i\pi/4}|+i\rangle \sim |+i\rangle & S|-i\rangle &= |+\rangle.
\end{aligned}$$

Consequently, up to a global phase, no combination of H and S can map a state in $\mathcal{S}$ to a state outside of $\mathcal{S}$. Therefore, for all $Q \in \langle H, S\rangle$,

$$\begin{aligned}
d_{\text{op}}(U, Q) &= \max_{\||\psi\rangle\|=1} \|(U-Q)|\psi\rangle\| \\
&\geq \|(U-Q)|0\rangle\| \\
&\geq \min_{|\phi\rangle\in\mathcal{S}} \|U|0\rangle - |\phi\rangle\| \\
&= \left\| \frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle - |+\rangle \right\| \\
&= \sqrt{2 - \sqrt{2 + \frac{4\sqrt{2}}{3}}} \\
&\approx 0.169 \\
&> \epsilon.
\end{aligned}$$

This is a contradiction. ■

Despite not being universal, there remains a great deal of interest in circuits over $\mathcal{G}_{\text{Clifford}}$ (a.k.a. *Clifford circuits*) largely because of their underlying group theory (in particular their relationship to the *Pauli group*), their use in quantum error correction, and because of the Gottesman–Knill theorem, which proves that Clifford circuits can be simulated on a classical computer.

Theorem 11.3 (Gottesman–Knill Theorem)**.** *If $f : \{0,1\}^* \to \{0,1\}^*$ is computable by an efficient quantum computer over $\mathcal{G}_{\text{Clifford}}$, then f is computable by an efficient probabilistic classical computer.*

Therefore, Clifford circuits cannot offer any sort of "quantum computational advantage", because every function they compute can be computed efficiently

classically. Interestingly, however, if we add the $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$ gate, then the situation changes drastically.

Fact 11.4. *The Clifford + T gate set,*

$$\mathcal{G}_{\text{Clifford}+T} \coloneqq \mathcal{G}_{\text{Clifford}} \cup \{T\} = \{T, H, S, \text{CNOT}\},$$

is universal.

Proof Idea.

(1) Show that $THTH = R_{\hat{n}}(\eta\pi)$, where $R_{\hat{n}}(\eta\pi)$ is a rotation about an axis $\hat{n}$ of the Bloch sphere by $\eta\pi$ radians, where (crucially) η is *irrational*.

(2) Show that $HTHT = R_{\hat{m}}(\rho\pi)$, where $R_{\hat{m}}(\rho\pi)$ is a rotation about an axis $\hat{m}$ of the Bloch sphere, where ρ is irrational and $\hat{m}$ is linearly independent of $\hat{n}$.

(3) Conclude that for all $\beta, \gamma \in [0, 2\pi)$, there exist $k, \ell \in \mathbb{N}$ such that $\beta \approx k\eta\pi \pmod{2\pi}$ and $\gamma \approx \ell\rho\pi \pmod{2\pi}$ so that $(THTH)^k = R_{\hat{n}}(k\eta\pi) \approx R_{\hat{n}}(\beta)$ and $(HTHT)^\ell = R_{\hat{m}}(\ell\rho\pi) \approx R_{\hat{m}}(\gamma)$.

(4) Use the generalized Euler decomposition $U = e^{i\alpha} R_{\hat{n}}(\beta) R_{\hat{m}}(\gamma) R_{\hat{n}}(\delta)$ to approximate any $U \in \mathrm{U}(2)$ (up to the immaterial global phase $e^{i\alpha}$). Note that this decomposition exists because $\hat{n}$ and $\hat{m}$ are linearly independent.

■

Interestingly, there is a more general result here.

Fact 11.5. *For all $k \geq 1$, if $U \in \mathrm{U}(2^k)$ is not a Clifford circuit, then $\mathcal{G}_{\text{Clifford}+U} \coloneqq \mathcal{G}_{\text{Clifford}} \cup \{U\}$ is universal.*

The proof of this fact is well beyond the scope of this course.[3]

11.3. Quantum Compilation

Suppose we are given a description of a unitary $U \in \mathrm{U}(N)$ in terms of single-qubit gates and an entangling gate, say CNOT. By Fact 11.1, there exists a circuit Q

[3] But if you're interested, see this stack exchange post, which summarizes the proof and contains the relevant references.

over a universal gate set $\mathcal{G}$ that approximate U to any accuracy $\epsilon > 0$ we like. This statement, however, is an *existence* statement, it tells us nothing about how to actually *find* or *compile* Q, given the gate set $\mathcal{G}$, the target unitary U, and the error parameter ϵ.

Compiling an approximating circuit Q is known as *quantum compilation*. Among the most fundamental results in quantum computing is the *Solovay-Kitaev theorem*, which proves that quantum compilation is actually not that hard. Below, we present a somewhat newer version of this result due to Bouland and Giurgica-Tiron, which (unlike the original Solovay-Kitaev theorem) does not require the set $\mathcal{G}$ to be closed under inverses.

Theorem 11.6 (Inverse-Free Solovay-Kitaev Theorem[4])**.** *Fix $N \geq 2$ and let $\mathcal{G}$ be a finite subset of $\mathrm{U}(N)$ whose generating set $\langle \mathcal{G} \rangle$ is dense in $\mathrm{U}(N)$. There exists a constant $c > 0$ such that for all $\epsilon > 0$ and all $U \in \mathrm{U}(N)$, there are $O(\log^c(1/\epsilon))$ gates from $\mathcal{G}$ whose matrix product p satisfies $d_{\mathrm{op}}(U, p) < \epsilon$. Moreover, there is a deterministic classical algorithm to find p in a time that is polylogarithmic in the parameter $1/\epsilon$.*

The Solovay-Kitaev theorem allows us to address the problem of whether the gate set $\mathcal{G}_{\mathrm{Clifford}+T}$ is somehow "better" than any other universal gate set $\mathcal{G}$, or if all universal gate sets are equally "good". Classically, every gate set is just as good as every other, because each can exactly implement the other, so it makes no difference which you use. But quantumly, we are only every *approximating* unitaries, so perhaps there is one gate set that approximates "better" than any other. The Solovay-Kitaev theorem, however, shows that this is false, and that all universal gates sets are essentially just as good as any other.

Corollary 11.7. *Let Q be an n-qubit circuit over a universal gate set $\mathcal{G}$ with $\ell \geq 0$ ancilla qubits and size T. Then, for all universal gate sets $\mathcal{G}'$, there exists a constant $c > 0$ such that for all $\epsilon > 0$, there exists an n-qubit circuit Q' over $\mathcal{G}'$ with ℓ ancilla qubits and size T' such that:*

(i) $T' = O(T \log^c(nT/\epsilon))$,

(ii) $d_{\mathrm{op}}(Q, Q') < \epsilon$.

[4] Actually, the statement of the Solovay-Kitaev theorem and its inverse-free version are both with respect to $\mathrm{SU}(N)$, not $\mathrm{U}(N)$. Recall that $\mathrm{SU}(N)$ is the subset of $\mathrm{U}(N)$ where the matrices have unit determinant. This restriction, however, is actually without loss of generality, because for all $U \in \mathrm{U}(N)$, there exists $\theta \in [0, 2\pi)$ and $U' \in \mathrm{SU}(N)$ such that $U' = e^{i\theta}U$. Since global phases like this don't affect the output statistics of a quantum circuit, it suffices to reason at the level of $\mathrm{SU}(N)$.

Moreover, there is a deterministic classical algorithm to find Q' in a time that is polylogarithmic in T/ϵ.

Proof Idea.* Let $\mathcal{G} = \{g_1, g_2, \ldots, g_r\}$ and $\mathcal{G}' = \{h_1, h_2, \ldots, h_t\}$. Then,

$$Q = U_d U_{d-1} \ldots U_1,$$

where $U_i = g_{i_1} \otimes g_{i_2} \otimes \cdots \otimes g_{i_m}$, $m \leq n$, and here each g_i is in $\mathcal{G} \cup \{I_2\}$. The essential steps of the proof are now as follows.

(1) Using the Solovay-Kitaev theorem, one can approximate each $g_i \in \mathcal{G}$ by some product p_i of gates in $\mathcal{G}'$, so that for all i, $d_{\text{op}}(g_i, p_i) < \epsilon/(nT)$. This incurs a depth cost of $O(\log^c(nT/\epsilon))$.

(2) Let $U_i' = p_{i_1} \otimes p_{i_2} \otimes \cdots \otimes p_{i_m}$ and show that[5]

$$\begin{aligned} d_{\text{op}}(U_i, U_i') &= d_{\text{op}}(g_{i_1} \otimes g_{i_2} \otimes \cdots \otimes g_{i_m}, p_{i_1} \otimes p_{i_2} \otimes \cdots \otimes p_{i_m}) \\ &\leq \sum_{k=1}^{m} d_{\text{op}}(h_{i_k}, p_{i_k}). \end{aligned}$$

Then,

$$\begin{aligned} d_{\text{op}}(U_i, U_i') &\leq \sum_{k=1}^{m} d_{\text{op}}(h_{i_k}, p_{i_k}) \\ &< \sum_{k=1}^{m} \frac{\epsilon}{nT} \\ &= m\left(\frac{\epsilon}{nT}\right) \\ &\leq n\left(\frac{\epsilon}{nT}\right) \\ &= \frac{\epsilon}{T}. \end{aligned}$$

[5]This ultimately follows from induction, the fact that $\|A \otimes B - C \otimes B\|_{\text{op}} = \|A - C\|_{\text{op}}$, and the fact that by the triangle inequality,

$$\begin{aligned} \|A \otimes B - C \otimes D\|_{\text{op}} &= \|A \otimes B - A \otimes D + A \otimes D - C \otimes D\| \\ &\leq \|A \otimes B - A \otimes D\|_{\text{op}} + \|A \otimes D - C \otimes D\|_{\text{op}} \\ &= \|B - C\|_{\text{op}} + \|A - D\|_{\text{op}}. \end{aligned}$$

(3) Show that[6]

$$d_{\text{op}}(U_d U_{d-1} \dots U_1, U'_d U'_{d-1} \dots U'_1) \leq \sum_{i=1}^{d} d_{\text{op}}(U_i, U'_i).$$

Then,

$$\begin{aligned} d_{\text{op}}(U_d U_{d-1} \dots U_1, U'_d U'_{d-1} \dots U'_1) &\leq \sum_{i=1}^{d} d_{\text{op}}(U_i, U'_i) \\ &< \sum_{i=1}^{d} \frac{\epsilon}{T} \\ &= d\frac{\epsilon}{T} \\ &\leq T\frac{\epsilon}{T} \\ &= \epsilon, \end{aligned}$$

where we've used the fact that $d \leq T$.

(4) Let $Q' = U'_d U'_{d-1} \dots U'_1$, and conclude that $d_{\text{op}}(Q, Q') < \epsilon$. Moreover, note that Q' can be found in polylogarithmic time in the parameter T/ϵ since each gate can be found in this time and we know from the structure of Q how to compose the gates. Also, note that the depth of Q' is not d, but $O(d \log^c(Tn/\epsilon))$, because to approximate each of the d matrices U'_i incurs depth $O(\log^c(Tn/\epsilon))$ by the Solovay-Kitaev theorem. Similarly, the overall size of Q' is $T' = O(T \log^c(Tn/\epsilon))$, since we approximate each of the T gates in Q using $O(\log^c(Tn/\epsilon))$ gates from $\mathcal{G}'$.

■

Ultimately, this result implies that efficient quantum computers over a universal gate set $\mathcal{G}$ can compute the same set of functions that efficient quantum computers over a different universal gate set $\mathcal{G}'$ can. I encourage you to think about this if this is not clear to you.

[6] This ultimately follows from induction, the fact that $\|B(A-C)\|_{\text{op}} = \|B\|_{\text{op}}\|A-C\|_{\text{op}}$, and the fact that by the triangle inequality,

$$\begin{aligned} \|AB - CD\|_{\text{op}} &= \|AB - CB + CB - CD\|_{\text{op}} \\ &\leq \|AB - CB\|_{\text{op}} + \|CB - CD\|_{\text{op}} \\ &= \|A - C\|_{\text{op}}\|B\|_{\text{op}} + \|B - D\|_{\text{op}}\|C\|_{\text{op}}. \end{aligned}$$

11.4. Computational Universality*

The way we have defined universality was motivated by the definition used in classical circuits. Namely, a finite set of gates is universal if and only if circuits over that set can approximate any other gate not in the set. However, at the end of the day, in a quantum computation all you really care about are the output statistics of measuring a certain state. For this reason, one could define universality with respect to a more statistical definition. Indeed, this is another notion of universality known as *computational universality*. To define this precisely, however, requires the following mathematical notion.

Definition 11.5. Let p and q be probability distributions supported on $\{0,1\}^n$. The *total variational distance (TVD) between p and q* is the quantity

$$d_{\mathrm{TVD}}(p,q) = \frac{1}{2} \sum_{y \in \{0,1\}^n} |p(y) - q(y)| \,.$$

It is not difficult to reason that if $d_{\mathrm{TVD}}(p,q)$ is small, then the distributions p and q are similar. Thus, the TVD is a measure of how close two probability distributions are.

Definition 11.6.

- Given $U \in \mathrm{U}(2^n)$, let

  $$p_U(x,y) = \langle x|U^\dagger \Pi_y U|x\rangle = |\langle y|U|x\rangle|^2$$

  be the probability that on input $|x\rangle$, U outputs $|y\rangle$.

- A gate set $\mathcal{G}$ is *computationally universal* iff for all $\epsilon > 0$ and all $U \in \mathrm{U}(2^n)$, there is a circuit Q over $\mathcal{G}$ such that for all $x \in \{0,1\}^n$,

  $$d_{\mathrm{TVD}}\big(p_U(x,\cdot), p_V(x,\cdot)\big) = \frac{1}{2} \sum_{y \in \{0,1\}^n} |p_U(x,y) - p_V(x,y)| < \epsilon.$$

In other words, $\mathcal{G}$ is computationally universal if and only if circuits over $\mathcal{G}$ can approximate the output statistics (as measured in the computational basis) of any target unitary U arbitrarily well.

It is natural to wonder how universality as we defined it and computational universality relate.

Fact 11.8 (HW3)**.** *If $\mathcal{G}$ is universal, then $\mathcal{G}$ is computationally universal.*

Interestingly, however, reasoning the other way is more difficult, because there exist unitaries U and V that are close in TVD, but far in operator norm. For example, with

$$U = I_{2^n} \quad \text{and} \quad V = \sum_{i=1}^{2^n} (-1)^{\delta_{0,i}} |i\rangle\langle i|,$$

where $\delta_{i,j}$ is the Kronecker delta, it is easy to show that for all $x \in \{0,1\}^n$, $d_{\mathrm{TVD}}(p_U(x,\cdot), p_V(x,\cdot)) = 0$ but that $d_{\mathrm{op}}(U,V) = 2$. Therefore, just because two unitaries have similar output statistics does not mean that they are close in operator norm. In fact, as you will argue in a second, computational universality does not imply universality.

An important example of a computationally universal gate set is due to Shi.

Theorem 11.9 (Shi, quant-ph/0205115)**.** $\{\mathrm{CCNOT}, H\}$ *is computationally universal.*

Given this, it is actually easy to prove that $\{\mathrm{CCNOT}, H\}$ is not universal.

Exercise 11.3. *Argue that $\{\mathrm{CCNOT}, H\}$ is not a universal gate set. (Hint: Think $\sqrt{-1}$.)*

A fascinating corollary of this discussion is the following, which I've stated informally (but of course you could formalize exactly what this means).

Corollary 11.10. *Complex numbers are not necessary for quantum computation.*

We will not talk about computational universality any more in this course (save a mention on HW3). If you are interested in this, though, then understanding Shi's paper and the above corollary would make a great final project.

Lecture 12
Quantum Computational Complexity Theory

Discussion 12.1. Discuss with your group what you took away from last time.

Last time, we discussed *universal* quantum gates sets and the Solovay-Kitaev theorem. That lecture marked the end of our discussing the formal underpinnings of quantum computation. This lecture serves to set the stage for the second half of this course, which is on quantum algorithms. To facilitate these forthcoming algorithmic discussions, in this lecture we will introduce the quantum analogue of the complexity class BPP, which is the probabilistic analogue of P.

12.1. Languages and Decision Problems

Complexity classes like P and NP are collections of *languages*.

Definition 12.1.

- A *language* L is a subset of $\{0,1\}^*$.
- The *indicator function of* L is the function $\chi_L : \{0,1\}^* \to \{0,1\}$, where for all $x \in \{0,1\}^*$,
$$x \in L \iff \chi_L(x) = 1.$$

Evidently, languages correspond to yes/no questions, a.k.a. *decision problems*, because asking "is x in L" is equivalent to evaluating $\chi_L(x)$. For this reason, we say that a (quantum or classical) computer C *decides* L iff C computes χ_L.

Example 12.1.

(i) $L_{\text{palindrome}} = \{x \in \{0,1\}^* : x \text{ is a palindrome}\}$

(ii) $L_{\text{prime}} = \{x \in \{0,1\}^* : x \text{ encodes a prime number}\}$

(iii) $L_{\text{factoring}} = \{x.y \in \{0,1\}^* : x \text{ has a non-trivial divisor that is at most } y\}$[1]

We will now discuss certain collections of languages, which are based on the types of computers that can decide them.

12.2. P, BPP, and Friends

Definition 12.2. The class P ("polynomial time") consists of all languages L for which there exists an efficient deterministic classical computer $(C, \mathcal{B})$ such that for all $x \in \{0,1\}^*$, $C_{|x|}(x) = \chi_L(x)$.

Question 12.1. *Is* $L_{palindrome} \in \mathsf{P}$*?*

Interestingly, L_{prime} is also in P, although this is not obvious.

Theorem 12.1 (Agrawal–Kayal–Saxena)**.** $L_{prime} \in \mathsf{P}$.

The class P consists of the languages L that are decidable by efficient *deterministic* classical computers. However, as we have said many times before, determinism is a special case of randomness, so the most powerful, practical notion of classical computation will exploit randomness. This brings us to the class BPP.

Definition 12.3. The class BPP ("bounded-error probabilistic polynomial time") consists of all languages L for which there exists an efficient probabilistic classical computer $(C, \mathcal{B}, s)$ such that for all $x \in \{0,1\}^*$,

$$\Pr_{r \sim \{0,1\}^{s(|x|)}} \left[C_{|x|+|r|}(x, r) = \chi_L(x) \right] \geq \frac{2}{3}.$$

Using the probability amplification theorem, it is possible to boost the 2/3 bound to be exponentially close to 1 without changing the class.

Question 12.2. *Is* $\mathsf{P} \subseteq \mathsf{BPP}$*?*

Most suspect that P = BPP because of countless "derandomization" results, which "take the randomness" out of an efficient probabilistic algorithm so that it becomes an efficient deterministic algorithm. That said, one problem that we know is in BPP but (as far as we know) not in P is called *polynomial identity testing*, which you can read about on your own time.

[1] Indeed, if you have an algorithm for this, and you know that N (which has binary expansion x) has a divisor d such that $1 < d \leq M \leq N$ (where M has binary representation y), then by using this algorithm and binary search, you can find d.

One of the major outstanding questions in theoretical computer science is if you can factor a number in polynomial time on an efficient probabilistic classical computer. We think this is impossible, and because of that, for decades we have based our cryptography on the following conjecture.

Conjecture 12.2. $L_{factoring} \notin \mathsf{BPP}$.

Let us now transition to another important classical complexity class.

Definition 12.4. The class NP ("non-deterministic polynomial time") consists of all languages L for which there exists a polynomial $s : \mathbb{N} \to \mathbb{N}$ and an efficient deterministic classical computer $(C, \mathcal{B})$ such that for all $x \in \{0,1\}^*$,

$$\begin{aligned} x \in L &\implies \exists r \in \{0,1\}^{s(|x|)} : C_{|x|+|r|}(x,r) = 1 \\ x \notin L &\implies \forall r \in \{0,1\}^{s(|x|)} : C_{|x|+|r|}(x,r) = 0. \end{aligned}$$

Here, r is called a *witness*, *certificate*, or *proof*.

In essence, NP consists of the languages L for which it is easy to verify a given "proof". P, on the other hand, consists of the languages L that are easy to "prove". Thus, saying $\mathsf{P} = \mathsf{NP}$ is like saying that finding a proof to a statement is just as easy as verifying that a given proof is correct. Of course, that seems unlikely, given our experiences in proving mathematical statements!

Interestingly, there is a simple relationship between NP and efficient probabilistic classical computers.

Fact 12.3. *$L \in \mathsf{NP}$ iff there exists an efficient probabilistic classical computer $(C, \mathcal{B}, s)$ such that for all $x \in \{0,1\}^*$,*

$$\begin{aligned} x \in L &\implies \Pr_{r \sim \{0,1\}^{s(|x|)}} \left[C_{|x|+|r|}(x,r) = 1 \right] > 0 \\ x \notin L &\implies \Pr_{r \sim \{0,1\}^{s(|x|)}} \left[C_{|x|+|r|}(x,r) = 0 \right] = 1. \end{aligned}$$

Exercise 12.1.

- *Is* $\mathsf{P} \subseteq \mathsf{NP}$*?*
- *Is* $L_{factoring} \in \mathsf{NP}$*?*
- *Is* $\mathsf{BPP} \subseteq \mathsf{NP}$*?*

In fact, it is an open question if $\mathsf{BPP} \subseteq \mathsf{NP}$. This is largely because in BPP, a computation can fail for both $x \in L$ and $x \notin L$, whereas for NP, the computation cannot fail for $x \notin L$ (in complexity words, there is not "two-sided bounded error" in NP, but there is in BPP). Nevertheless, most suspect that $\mathsf{BPP} \subseteq \mathsf{NP}$, because most suspect that $\mathsf{P} = \mathsf{BPP}$ (and we already know that $\mathsf{P} \subseteq \mathsf{NP}$).

We will now define two classes with respect to the Turing machine model of computation, but of course these could be phrased with respect to the circuit model. If you're confused by the notion of a "deterministic Turing machine", it is fine to think of it as a "deterministic Python program".

Definition 12.5. The class $\mathsf{EXPTIME}$ ("exponential time") consists of all languages L for which there is a deterministic Turing machine that decides L in $O(2^{n^k})$ time.

Exercise 12.2.

(i) *Is* $\mathsf{NP} \subseteq \mathsf{EXPTIME}$*?*

(ii) *Is* $\mathsf{BPP} \subseteq \mathsf{PSPACE}$*?*

So far, all these classes have concerned the amount of *time* or *computational steps* of the underlying computer. However, there is another resource that is important in computation, namely, *space* or *memory*. Like P, then, we can talk about computations that can be performed with polynomial space.

Definition 12.6. The class PSPACE ("polynomial space") consists of all languages L for which there is a deterministic Turing machine that decides L in $O(n^k)$ space.

Exercise 12.3.

(i) *Is* $\mathsf{PSPACE} \subseteq \mathsf{EXPTIME}$*?*

(ii) *Is* $\mathsf{NP} \subseteq \mathsf{PSPACE}$*?*

(iii) *Is* $\mathsf{BPP} \subseteq \mathsf{PSPACE}$*?*[2]

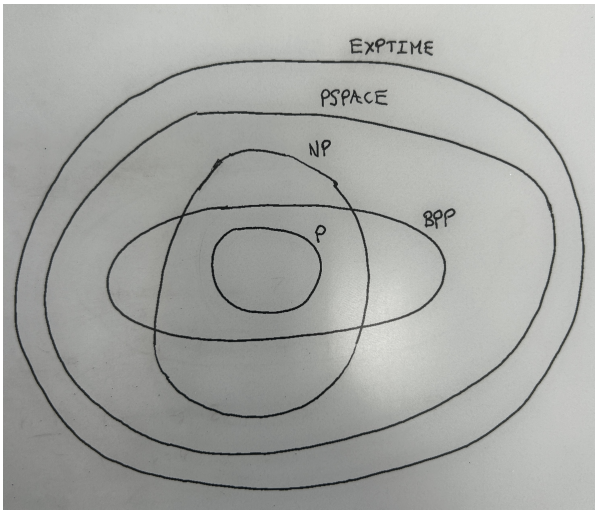
Given these five classes, we can now state a myriad of open questions about how they relate.

Open Problem 12.4.

[2] In fact, it follows from something called the *Time Hierarchy Theorem* that $\mathsf{P} \neq \mathsf{EXPTIME}$. Therefore, one of the following statements is necessarily true, we just don't know which one (although we expect them all to be true): $\mathsf{P} \neq \mathsf{NP}$, $\mathsf{BPP} \neq \mathsf{PSPACE}$, $\mathsf{NP} \neq \mathsf{PSPACE}$, $\mathsf{PSPACE} \neq \mathsf{EXPTIME}$.

- *Is* $\mathsf{P} = \mathsf{NP}$*? (Literally a million dollar question!)*
- *Is* $\mathsf{P} = \mathsf{BPP}$*?*
- *Is* $\mathsf{BPP} \subseteq \mathsf{NP}$*?*
- *Is* $\mathsf{NP} = \mathsf{PSPACE}$*?*
- *Is* $\mathsf{PSPACE} = \mathsf{EXPTIME}$*?*

Solving any of these problems would cement your name in the history of computer science. Overall, at least for these five classes, this is what we currently know:



This concludes our brief tour of some of the most important classical complexity classes. We will now discuss what is unequivocally the most important *quantum* complexity class, BQP, which, as we shall see, is the quantum analogue of BPP.

12.3. BQP

Definition 12.7. Let $\mathcal{G}$ be a gate set. The class $\mathsf{BQP}(\mathcal{G})$ ("bounded-error quantum polynomial time over the gate set $\mathcal{G}$") consists of all languages L for which there exists an efficient quantum computer $(Q, \mathcal{G}, a)$ such that for all $x \in \{0,1\}^*$,

$$\Pr\left[Q^{(1)}(x) = |\chi_L(x)\rangle\right] \geq \frac{2}{3}.$$

Importantly, if $\mathcal{G}$ is universal, then the class is robust to changes in the gate set.

Fact 12.5 (HW4)**.** *For all universal gate sets $\mathcal{G}$ and $\mathcal{G}'$,* $\mathsf{BQP}(\mathcal{G}) = \mathsf{BQP}(\mathcal{G}')$.

Because of this fact, the following is a well-defined definition.

Definition 12.8. The class BQP ("bounded-error quantum polynomial time") is $\mathsf{BQP}(\mathcal{G}_{\text{Clifford}+T})$.

Like for BPP, the $2/3$ in the definition of BQP can be amplified to be exponentially close to one using the probability amplification theorem. Note, however, that this is not always the case for $\mathsf{BQP}(\mathcal{G})$ for non-universal $\mathcal{G}$.

Question 12.3. *Is* $\mathsf{BPP} \subseteq \mathsf{BQP}$*?*

We will see in a few lectures from now that you can factor integers on an efficient quantum computer.

Theorem 12.6 (Shor)**.** $L_{factoring} \in \mathsf{BQP}$.

Therefore, if $L_{\text{factoring}} \notin \mathsf{BPP}$, then $\mathsf{BPP} \neq \mathsf{BQP}$. Moreover, because of this fact, quantum computers threaten many cryptographic schemes that secure sensitive data (e.g., your bank account information). At the end of the day, the overwhelming belief in quantum computing circles is the following conjecture:

Conjecture 12.7. $\mathsf{BPP} \neq \mathsf{BQP}$.

Similar to how it is not known how BPP and NP relate, it is not known how BQP and NP relate.

Open Problem 12.8.

- *Is* $\mathsf{BQP} \subseteq \mathsf{NP}$*?*
- *Is* $\mathsf{NP} \subseteq \mathsf{BQP}$*?*

The way we have defined $\mathsf{BQP}(\mathcal{G})$ opens up an interesting question. What if we deliberately consider *non*-universal $\mathcal{G}$? Could $\mathsf{BQP}(\mathcal{G})$ over a non-universal $\mathcal{G}$ ever equal BQP? Interestingly, the answer is yes.

Fact 12.9. *If $\mathcal{G}$ is computationally universal, then* $\mathsf{BQP}(\mathcal{G}) = \mathsf{BQP}$.[3]

For weaker choices of $\mathcal{G}$, however, the situation is often different.

Theorem 12.10 (Gottesman–Knill–Aaronson)**.** $\mathsf{BQP}(\mathcal{G}_{\text{Clifford}}) = \oplus\mathsf{L} \subseteq \mathsf{P}$.

Beyond BQP, one can define quantum analogues of many other classical complexity classes. For example, the class QMA is the quantum analogue of NP. Exploring this class (or any other quantum complexity class) would make a great final project.

[3] You have all the tools you need to prove this fact for yourself.

12.4. The Limits of BQP

We already know that efficient quantum computers are *at least* as powerful as efficient probabilistic classical computers (i.e., $\mathsf{BPP} \subseteq \mathsf{BQP}$). But just how powerful are efficient quantum computers? In other words, what class *contains* BQP, so that we can *upper-bound* the power of efficient quantum computers? A very plain upper-bound is the following.

Exercise 12.4. *Argue that* $\mathsf{BQP} \subseteq \mathsf{EXPTIME}$.

Therefore, efficient quantum computers cannot compute functions that are not computable in exponential time on a deterministic Turing machine. In fact, we can refine this to show that efficient quantum computers cannot compute functions that are not computable in *polynomial space* on a deterministic Turing machine. To do this, we will need the following claim.

Claim 12.11. *Let* $U \in \mathrm{U}(2^n)$ *be such that*

$$U = \bigotimes_{k=1}^{m} g_k, \qquad g_k \in \mathcal{G}_{\text{Clifford}+T} \cup \{I_2\},$$

where $m \leq n$*. Then, for all* $x, y \in \{0,1\}^n$*, the matrix element* $\langle y|U|x\rangle$ *is exactly computable in* $\mathrm{poly}(n)$ *space on a deterministic Turing machine.*

Proof. By definition,

$$\begin{aligned}\langle y|U|x\rangle &= \langle y| \bigotimes_{k=1}^{m} g_k |x\rangle \\ &= \langle y_{[1]}|g_1|x_{[1]}\rangle \otimes \langle y_{[2]}|g_2|x_{[2]}\rangle \otimes \cdots \otimes \langle y_{[m]}|g_m|x_{[m]}\rangle \\ &= \prod_{i=1}^{m} \langle y_{[i]}|g_i|x_{[i]}\rangle,\end{aligned}$$

where $x_{[i]}$ is either a 1- or 2-bit substring of x, depending on if g_i is a 1- or 2-qubit unitary, respectively (and similarly for $y_{[i]}$). The product $\langle y_{[i]}|g_i|x_{[i]}\rangle$ is at most a matrix multiplication involving a 1×4 dimensional row vector $\langle y_{[i]}|$ times a 4×4 dimensional matrix g_i times a 4×1 dimensional ket vector $|x_{[i]}\rangle$. The bit representation of each of these elements is constant with respect to n. Therefore, the individual product can be done in constant space, and so one can evaluate all n products in $O(n)$ space (there are at most n products, as $m \leq n$). Multiplying the results together then incurs $\mathrm{poly}(n)$ space, as desired. ∎

Using this result, we can now prove the main result of this lecture.

Claim 12.12. $\mathsf{BQP} \subseteq \mathsf{PSPACE}$.

Proof. Let $L \in \mathsf{BQP}$. Then, there is an efficient quantum computer $(Q, \mathcal{G}_{\text{Clifford}+T}, a)$ such that for all $x \in \{0,1\}^*$,

$$\Pr\left[Q^{(1)}_{|x|+|a(|x|)|}(x, a(|x|)) = |\chi_L(x)\rangle\right] \geq \frac{2}{3}.$$

To simplify notation, put $n = |x|$ and $\ell = |a(|x|)|$. Then, $Q_{|x|+|a(|x|)|} = Q_{n+\ell}$ and

$$\Pr\left[Q^{(1)}_{n+\ell}(x, a(n)) = |1\rangle\right] = \sum_{y \in \{0,1\}^{n+\ell-1}} \langle x.a(n)|Q_{n+\ell}|1.y\rangle,$$

where . denotes string concatenation.

We will show that for all $x \in \{0,1\}^*$, one can compute this probability in polynomial space in $n = |x|$ on a deterministic Turing machine T. Therefore, after computing it, we will program T to accept (i.e., output 1) if the probability is at least $2/3$, and reject (i.e., output 0) otherwise. This implies $L \in \mathsf{PSPACE}$.

To compute the probability in polynomial space, consider the following representation of $Q_{n+\ell}$:

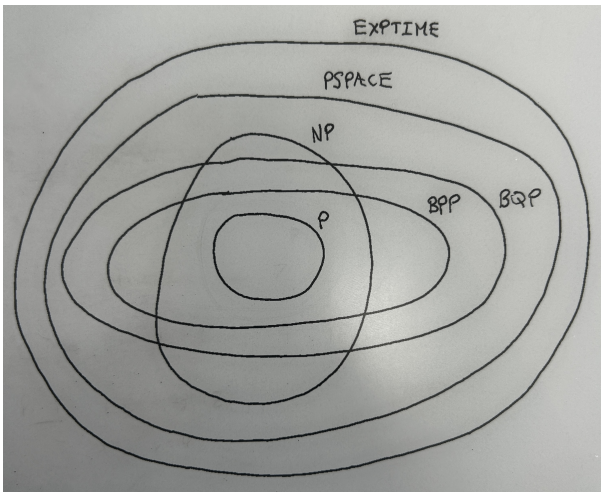
$$\begin{aligned}
Q_{n+\ell} &= U_d U_{d-1} \dots U_1 \\
&= U_d I_{2^{n+\ell}} U_{d-1} I_{2^{n+\ell}} \dots I_{2^{n+\ell}} U_1 \\
&= U_d \left(\sum_{w_0 \in \{0,1\}^{n+\ell}} |w_0\rangle\langle x_0| \right) U_{d-1} \left(\sum_{w_1 \in \{0,1\}^{n+\ell}} |w_1\rangle\langle w_1| \right) \dots \left(\sum_{w_d \in \{0,1\}^{n+\ell}} |w_d\rangle\langle x_d| \right) U_1 \\
&= \sum_{w_0, w_1, \dots, w_d \in \{0,1\}^{n+\ell}} U_d|w_0\rangle\langle w_0|U_{d-1}|w_1\rangle\langle w_1| \cdots |w_d\rangle\langle w_d|U_1.
\end{aligned}$$

Consequently,

$$\begin{aligned}
&\langle x.a(n)|Q_{n+\ell}|1.y\rangle \\
&\qquad = \sum_{w_0, w_1, \dots, w_d \in \{0,1\}^{n+\ell}} \Big(\langle x.a(n)|U_d|w_0\rangle\Big)\Big(\langle w_0|U_{d-1}|w_1\rangle\Big) \cdots \Big(\langle w_d|U_1|1.y\rangle\Big).
\end{aligned}$$

By the previous claim, each term in parenthesis can be computed in polynomial space, so, by keeping a counter, the overall sum can be computed in polynomial space as well. Summing over all $y \in \{0,1\}^{n+\ell-1}$ can then be done in polynomial space as well. It follows, therefore, that we can compute the probability from before in polynomial space, which implies that $L \in \mathsf{PSPACE}$, as desired. ∎

In conclusion, efficient quantum computers cannot compute functions that are not computable in polynomial space on a deterministic Turing machine. In fact, there are better upper-bounds on BQP (such as $\mathsf{BQP} \subseteq \mathsf{PP}$), and exploring those would make a great final project. The following diagram summarizes the relationship between the various complexity classes that talked about in this lecture.



This concludes our study of the structural complexity of quantum computing. In the next several lectures, we will see explicit problems that are in BQP (or a related, oracle-based class) that are believed to not be in BPP (or a related, oracle-based class). It turns out all these problems are solvable on an efficient quantum computer basically because quantum computers can implement a particular unitary operator rather efficiently: the quantum Fourier transform. We will learn more about this in a few lectures. Next time, we will study the *unstructured search problem*, which is the context of Lov Grover's eponymous algorithm.

Lecture 13
Grover's Algorithm

Discussion 13.1. Discuss with your group what you took away from last time.

Last time, we discussed quantum computational complexity theory. There, we introduced many complexity classes, including BPP and BQP. These classes consist of the languages that are decidable on efficient classical and quantum computer, respectively. The belief that underlies most of quantum computing is that $\mathsf{BPP} \neq \mathsf{BQP}$, and in this lecture we will begin our study of quantum algorithms that support this conjecture.

13.1. Oracles and the Query Complexity Paradigm

Definition 13.1. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a (not necessarily computable!) function. We call a (classical or quantum) operation $\mathcal{O}_f$ that computes f an *oracle for* f. Quantumly, $\mathcal{O}_f$ is a unitary operation for each input size n, so it is defined to act in the canonical, reversible way on computational basis states:

$$\mathcal{O}_f : |x\rangle|a\rangle \mapsto |x\rangle|a \oplus f(x)\rangle.$$

Notice, in this definition we have not specified how difficult it is to compute f. This is on purpose, because it may, in fact, be very difficult (or impossible!) to compute f. This is the paradigm of *query complexity*, in which we assume we have some black box function (the oracle), and we do not care at all about how difficult it is to implement the black box. Instead, what we care about is minimizing the number of calls to the black box, i.e., to the oracle (or subroutine, if you like) $\mathcal{O}_f$. As we will see, in many problems, quantum computers require provably fewer queries to f than any classical computer, basically because of the quantum parallelism idea we discussed in a previous lecture.

13.2. Unstructured Search

Grover's algorithm solves the following problem, which is called *unstructured search*.

The Unstructured Search Problem

Input: A function $f : \{0,1\}^n \to \{0,1\}$ as an oracle such that

$$\begin{cases} f(x) = 1 & \text{if } x = x^\star \in \{0,1\}^n \\ f(x) = 0 & \text{otherwise.} \end{cases}$$

Output: $x^\star$ (the "needle in the haystack") by querying f.

Exercise 13.1. *Argue that, on average, every classical probabilistic algorithm that solves unstructured search by querying f must make $\Omega(N)$ queries.*

As we will now see, however, it is possible to do much better quantumly.

13.3. Grover's Algorithm

To solve unstructured search on a quantum computer, first recall the quantum oracle we have for f:

$$\mathcal{O}_f|x\rangle|a\rangle = |x\rangle|a \oplus f(x)\rangle.$$

Importantly, by using superposition, we can change the effect this oracle has on our quantum states.

Exercise 13.2. *Prove that for all $x \in \{0,1\}^n$, $\mathcal{O}_f|x\rangle|-\rangle = (-1)^{f(x)}|x\rangle|-\rangle$.*[1]

[1] This is an example of something called *phase kickback*.

Answer:

$$\begin{aligned}
\mathcal{O}_f|x\rangle|-\rangle &= \mathcal{O}_f|x\rangle\frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) \\
&= \frac{1}{\sqrt{2}}\left(\mathcal{O}_f|x\rangle|0\rangle - \mathcal{O}_f|x\rangle|1\rangle\right) \\
&= \frac{1}{\sqrt{2}}\left(|x\rangle|f(x)\rangle - |x\rangle|1\oplus f(x)\rangle\right) \\
&= |x\rangle\frac{1}{\sqrt{2}}\left(|f(x)\rangle - |1\oplus f(x)\rangle\right) \\
&= \begin{cases} |x\rangle|-\rangle & \text{if } f(x)=0 \\ -|x\rangle|-\rangle & \text{if } f(x)=1. \end{cases} \\
&= (-1)^{f(x)}|x\rangle|-\rangle.
\end{aligned}$$

For this reason, we can think of the oracle as only acting on the input register

$$\mathcal{O}_f : |x\rangle \mapsto (-1)^{f(x)}|x\rangle.$$

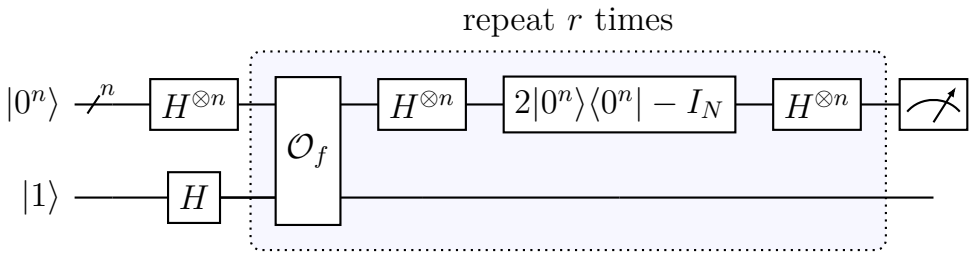
This is because the ancilla register is not entangled with the input register (so we can just ignore the ancilla in the analysis of the computation). Indeed, we will do this in the following analysis (though we will include the ancilla in the set up to Grover's algorithm). Context should make it clear which version of the oracle we mean.

We now introduce Grover's algorithm for solving the unstructured search problem.

Grover's Algorithm:

1. Prepare $n+1$ qubits in the state $|0^n\rangle|1\rangle$.
2. Apply $H^{\otimes n+1}$.
3. Perform the following *Grover iteration* r times:
   (i) apply $\mathcal{O}_f$,
   (ii) apply $H^{\otimes n}$ on the first n qubits,
   (iii) apply the *Grover diffusion operator* $2|0^n\rangle\langle 0^n| - I_N$ on the first n qubits,
   (iv) apply $H^{\otimes n}$ on the first n qubits.
4. Measure the first n qubits in the computational basis.

As a circuit, Grover's algorithm is simply:



Here, r measures the number of times that we call the oracle $\mathcal{O}_f$. We will see that $r = O(\sqrt{N})$ suffices for the output to be $|x^\star\rangle$ with probability exponentially close to one. Note, also, that a single Grover iteration (step 3) corresponds to applying what is sometimes called the *Grover operator*,[2]

$$\begin{aligned} G &\coloneqq H^{\otimes n}(2|0^n\rangle\langle 0^n| - I_N)H^{\otimes n}\mathcal{O}_f \\ &= (2|\Phi\rangle\langle\Phi| - I_N)\mathcal{O}_f, \end{aligned}$$

where

$$|\Phi\rangle = H^{\otimes n}|0^n\rangle = \frac{1}{\sqrt{N}} \sum_{x\in\{0,1\}^n} |x\rangle.$$

In this lecture, we will ultimately prove the following theorem.

Theorem 13.1. *Using a quantum computer (namely, that specified by Grover's algorithm), it is possible to solve the unstructured search problem with probability $1 - O(2^{-n})$ using only $O(\sqrt{N}) = O(2^{n/2})$ queries to f. This is quadratically fewer than the classical lower bound.*

13.4. Correctness of Grover's Algorithm

To prove that Grover's algorithm does what it says, we will begin by building some geometric intuition behind the algorithm, which will assist in its analysis. To do this, we will start with a simple exercise.

Exercise 13.3. *Let $|\psi_i\rangle$ be the state in Grover's algorithm at step i.*

- *Find $|\psi_1\rangle$ and $|\psi_2\rangle$*
- *If*

$$|\alpha\rangle = \frac{1}{\sqrt{N-1}} \sum_{\substack{x\in\{0,1\}^n \\ x\neq x^\star}} |x\rangle,$$

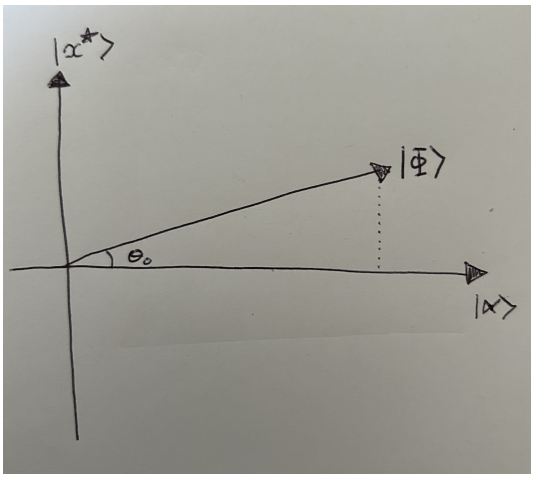
[2]Here, we are using the "phase" interpretation of the oracle $\mathcal{O}_f$, in which it ignores the ancilla.

prove that

$$|\Phi\rangle = \sqrt{\frac{N-1}{N}}|\alpha\rangle + \frac{1}{\sqrt{N}}|x^\star\rangle.$$

Corollary 13.2. *$|\Phi\rangle$ is a two dimensional vector in the two-dimensional subspace of $\mathbb{C}^{2^n}$ that is spanned by $|\alpha\rangle$ and $|x^\star\rangle$.*

Pictorially,



Here, $\theta_0 \in [0, 2\pi)$ is such that

$$\cos\theta_0 = \sqrt{\frac{N-1}{N}}$$
$$\sin\theta_0 = \frac{1}{\sqrt{N}}$$

so that, by the previous exercise,

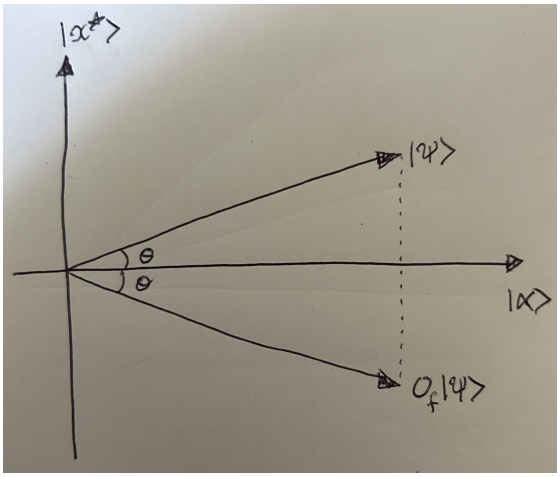
$$|\Phi\rangle = \cos\theta_0|\alpha\rangle + \sin\theta_0|x^\star\rangle.$$

In what follows, we will analyze geometrically what happens in step 3 of Grover's algorithm, and deduce the number of iterations we need to do to find $|x^\star\rangle$ with high probability. We start by understanding how the oracle acts in this two dimensional representation.

Claim 13.3. *If $|\psi\rangle = \cos\theta|\alpha\rangle + \sin\theta|x^\star\rangle$, then*

$$\mathcal{O}_f|\psi\rangle = \cos\theta|\alpha\rangle - \sin\theta|x^\star\rangle.$$

Pictorially,



Proof. By definition,

$$\begin{aligned}\mathcal{O}_f|\alpha\rangle &= |\alpha\rangle \\ \mathcal{O}_f|x^\star\rangle &= -|x^\star\rangle,\end{aligned}$$

so the result follows from linearity. ■

We will now prove one more geometric fact, but this time about the Grover operator G.

Claim 13.4. *Let $G = (2|\Phi\rangle\langle\Phi| - I_N)\mathcal{O}_f$ be the Grover operator. Then,*

$$\begin{aligned}G|x^\star\rangle &= \cos\omega|x^\star\rangle - \sin\omega|\alpha\rangle \\ G|\alpha\rangle &= \sin\omega|x^\star\rangle + \cos\omega|\alpha\rangle,\end{aligned}$$

where

$$\begin{aligned}\sin\omega &= \frac{2\sqrt{N-1}}{N} \\ \cos\omega &= \sqrt{1-\sin^2\theta} = 1 - \frac{2}{N}.\end{aligned}$$

Proof. We find:

$$\begin{aligned}G|x^\star\rangle &= -(2|\Phi\rangle\langle\Phi| - I_N)|x^\star\rangle && \text{by } \mathcal{O}_f|x^\star\rangle = -|x^\star\rangle \\ &= -2|\psi\rangle\langle\Phi|x^\star\rangle + |x^\star\rangle \\ &= -\frac{2}{\sqrt{N}}|\Phi\rangle + |x^\star\rangle && \text{by } \langle\Phi|x^\star\rangle = \frac{1}{\sqrt{N}}.\end{aligned}$$

Now using the formula for $|\Phi\rangle$ from before,

$$\begin{aligned}
G|x^\star\rangle &= -\frac{2}{\sqrt{N}}|\Phi\rangle + |x^\star\rangle \\
&= -\frac{2}{\sqrt{N}}\left(\sqrt{\frac{N-1}{N}}|\alpha\rangle + \frac{1}{\sqrt{N}}|x^\star\rangle\right) + |x^\star\rangle \\
&= -\frac{2\sqrt{N-1}}{N}|\alpha\rangle - \frac{2}{N}|x^\star\rangle + |x^\star\rangle \\
&= -\frac{2\sqrt{N-1}}{N}|\alpha\rangle + \left(1 - \frac{2}{N}\right)|x^\star\rangle \\
&= -\sin\omega|\alpha\rangle + \cos\omega|x^\star\rangle.
\end{aligned}$$

Similarly, since

$$\begin{aligned}
\langle\Phi|\alpha\rangle &= \left(\frac{1}{\sqrt{N}}\sum_{x\in\{0,1\}^n}\langle x|\right)\left(\frac{1}{\sqrt{N-1}}\sum_{x'\neq x^\star}|x'\rangle\right) \\
&= \frac{1}{\sqrt{N(N-1)}}\sum_{\substack{x\in\{0,1\}^n \\ x'\neq x^\star}}\langle x|x'\rangle \\
&= \frac{1}{\sqrt{N(N-1)}}\sum_{x\neq x^\star}\langle x|x\rangle \\
&= \frac{N-1}{\sqrt{N(N-1)}} \\
&= \sqrt{\frac{N-1}{N}},
\end{aligned}$$

it holds that

$$\begin{aligned}
G|\alpha\rangle &= (2|\Phi\rangle\langle\Phi| - I_N)|\alpha\rangle \qquad \text{by } \mathcal{O}_f|\alpha\rangle = |\alpha\rangle \\
&= 2|\Phi\rangle\langle\Phi|\alpha\rangle - |\alpha\rangle \\
&= 2\sqrt{\frac{N-1}{N}}|\Phi\rangle - |\alpha\rangle.
\end{aligned}$$

Now using the formula for $|\Phi\rangle$ from before,

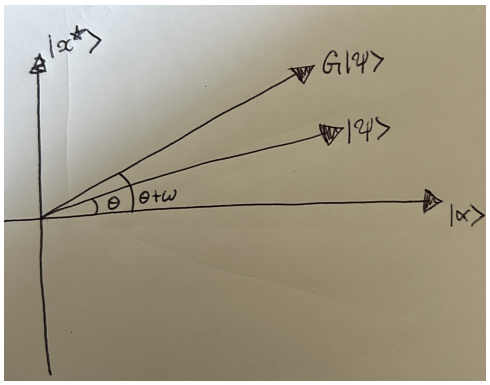
$$\begin{aligned}
G|\alpha\rangle &= 2\sqrt{\frac{N-1}{N}}|\Phi\rangle - |\alpha\rangle \\
&= 2\sqrt{\frac{N-1}{N}}\left(\sqrt{\frac{N-1}{N}}|\alpha\rangle + \frac{1}{\sqrt{N}}|x^\star\rangle\right) - |\alpha\rangle \\
&= \left(2\frac{N-1}{N} - 1\right)|\alpha\rangle + \frac{2\sqrt{N-1}}{N}|x^\star\rangle \\
&= \left(2 - \frac{2}{N} - 1\right)|\alpha\rangle + \sin\omega|x^\star\rangle \\
&= \left(1 - \frac{2}{N}\right)|\alpha\rangle + \sin\omega|x^\star\rangle \\
&= \cos\omega|\alpha\rangle + \sin\omega|x^\star\rangle.
\end{aligned}$$

■

Corollary 13.5. *If* $|\psi\rangle = \cos\theta|\alpha\rangle + \sin\theta|x^\star\rangle$, *then*

$$G|\psi\rangle = \cos(\theta+\omega)|\alpha\rangle + \sin(\theta+\omega)|x^\star\rangle$$

Pictorially,



Proof. This follows from the previous result and elementary trigonometry:

$$\begin{aligned}
G|\psi\rangle &= G(\cos\theta|\alpha\rangle + \sin\theta|x^\star\rangle) \\
&= \cos\theta(G|\alpha\rangle) + \sin\theta(G|x^\star\rangle) \\
&= \cos\theta(\sin\omega|x^\star\rangle + \cos\omega|\alpha\rangle) + \sin\theta(\cos\omega|x^\star\rangle - \sin\omega|\alpha\rangle) \\
&= (\cos\theta\cos\omega - \sin\theta\sin\omega)|\alpha\rangle + (\cos\theta\sin\omega + \sin\theta\cos\omega)|x^\star\rangle \\
&= \cos(\theta+\omega)|\alpha\rangle + \sin(\theta+\omega)|x^\star\rangle.
\end{aligned}$$

■

Therefore, the Grover operation rotates a given state closer to the target state $|x^\star\rangle$. We can use this to our advantage to now find $|x^\star\rangle$ with high probability.

Claim 13.6. *If* $r = \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rceil$,[3] *then the probability that we measure* $G^r|\Phi\rangle$ *to be in state* $|x^\star\rangle$ *is* $1 - O(1/N) = 1 - O(2^{-n})$.

Proof. Since

$$|\Phi\rangle = \cos\theta_0|\alpha\rangle + \sin\theta_0|x^\star\rangle,$$

by the previous corollary

$$G^r|\Phi\rangle = \cos(\theta_0 + r\omega)|\alpha\rangle + \sin(\theta_0 + r\omega)|x^\star\rangle.$$

We want r such that $\cos(\theta_0 + r\omega) = 0$, so that when we measure the state, we will measure $|x^\star\rangle$. Since

$$\cos(\theta_0 + r\omega) = 0 \text{ if } \theta_0 + r\omega = \frac{\pi}{2},$$

choose

$$r = \left\lfloor \frac{\frac{\pi}{2} - \theta_0}{\omega} \right\rceil.$$

Using the Taylor series

$$\arcsin(x) = x + \frac{x^3}{6} + \cdots + \frac{(2n)!}{2^{2n}(n!)^2}\frac{x^{2n+1}}{2n+1} + \cdots,$$

it follow that

$$\begin{aligned}\theta_0 &= \arcsin\frac{1}{\sqrt{N}} \\ &= \frac{1}{\sqrt{N}} + O\left(\frac{1}{N^{3/2}}\right)\end{aligned}$$

and

$$\begin{aligned}\omega &= \arcsin\frac{2\sqrt{N-1}}{N} \\ &= \frac{2\sqrt{N-1}}{N} + O\left(\frac{1}{N^{3/2}}\right) \\ &= \frac{2}{\sqrt{N}} + O\left(\frac{1}{N}\right) \\ &= \frac{2}{\sqrt{N}}\left(1 + O\left(\frac{1}{\sqrt{N}}\right)\right).\end{aligned}$$

[3]Here, $\lfloor x \rceil$ denotes the nearest integer to x.

Consequently,[4]

$$\begin{aligned} r &= \left\lfloor \frac{\frac{\pi}{2} - \theta_0}{\omega} \right\rceil \\ &= \left\lfloor \frac{\frac{\pi}{2} - \frac{1}{\sqrt{N}} + O\left(\frac{1}{N^{3/2}}\right)}{\frac{2}{\sqrt{N}}\left(1 + O\left(\frac{1}{\sqrt{N}}\right)\right)} \right\rceil \\ &= \left\lfloor \frac{\frac{\pi}{2}\sqrt{N} - 1 + O\left(\frac{1}{\sqrt{N}}\right)}{2\left(1 + O\left(\frac{1}{\sqrt{N}}\right)\right)} \right\rceil \\ &= \left\lfloor \frac{\pi\sqrt{N}}{4} - \frac{1}{2} + O\left(\frac{1}{\sqrt{N}}\right) \right\rceil \end{aligned}$$

Therefore, with $r = \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rceil$,[5]

$$\begin{aligned} \Pr\left[\text{measure } G^r|\Phi\rangle \text{ in state } |x^\star\rangle\right] &= \sin^2(\theta_0 + r\omega) \\ &= \sin^2\left(\underbrace{\frac{1}{\sqrt{N}}}_{\theta_0} + \underbrace{\left(\frac{\pi}{4}\sqrt{N}\right)}_{r}\underbrace{\frac{2}{\sqrt{N}}}_{\omega} + \underbrace{O\left(\frac{1}{\sqrt{N}}\right)}_{\text{other terms}}\right) \\ &= \sin^2\left(\frac{\pi}{2} + O\left(\frac{1}{\sqrt{N}}\right)\right) \\ &= \cos^2\left(O\left(\frac{1}{\sqrt{N}}\right)\right) \\ &= \left(1 + O\left(\frac{1}{N}\right)\right)^2 \\ &= 1 + O\left(\frac{1}{N}\right). \end{aligned}$$

■

Of course, here the probability must be at most one, so the $O(\frac{1}{N})$ term is necessarily negative. Altogether, then, we have proven Theorem 13.1, as desired.

[4]Here, we use the fact that if $|x| < 1$, then $\frac{1}{1+x} = 1 - x + x^2 - x^3 + \cdots$.
[5]Here, we use the fact that $\sin\left(\frac{\pi}{2} + x\right) = \cos x$ and that $\cos x = 1 + O(x^2)$.

13.5. Generalizations and Quantum Optimality*

In the discussion above, we assumed there was only one choice of $x \in \{0,1\}^n$ such that $f(x) = 1$ (namely, $x = x^\star$), but in fact Grover's algorithm easily generalizes to the case of many possible "correct" x. The proof of the following fact uses essentially the same argument above, and it makes a good exercise to modify Grover's algorithm to prove this fact.

Fact 13.7. *Let $f : \{0,1\}^n \to \{0,1\}$ be such that for all $x \in S \subseteq \{0,1\}^n$, $f(x) = 1$, and for all $x \in \{0,1\}^n \backslash S$, $f(x) = 1$. If $M = |S| < N = 2^n$ and M is known, then Grover's algorithm can find $x \in S$ with probability $1 - O(2^{-n})$ using $O(\sqrt{N/M})$ queries to f.*

Interestingly, if $M = N/2$, then Grover's algorithm provides no speedup (do you see why?). If $M > N/2$, then Grover's algorithm performs *worse* than classical search algorithms. In general, then, you should use a classical algorithm if $M \geq N/2$.

Of course, we might not know M from the start. In that case, there is an algorithm called the *quantum counting algorithm*, which allows one to first find M, and then to find a correct solution $x \in S$.

Among the many generalizations of Grover, perhaps the most useful is known as *amplitude amplification*. This was the basic idea in the proof, in which we separated an equal superposition of all strings into those strings that are not what we want to measure (e.g., all $x \in \{0,1\}^n \backslash \{x^\star\}$) plus those strings that we want to measure (e.g., $x^\star$). What Grover allowed us to do is amplify the amplitude on the $x^\star$ string, so that when we measure, we are more likely than not to see $|x^\star\rangle$. Indeed, this is the germ of amplitude amplification, and you are encouraged to read more about it if this stuff interests you.

Finally, it is natural to wonder if there is any quantum algorithm that could outperform Grover in the unstructured search problem. It turns out, however, that this is false.

Fact 13.8. *Any quantum algorithm that solves the unstructured search problem with probability at least $1/2 + \delta$, for any $\delta > 0$, must make $\Omega(\sqrt{N/M})$ queries to f.*

In the context of the complexity class NP, this lower bound has important consequences.

13.6. Grover and NP*

On the surface, it may seem like Grover's algorithm could give quantum computers an edge when it comes to NP-complete problems like k-SAT.[6] Indeed, they do have an edge, but unfortunately not a considerable one (thanks to the Grover lower bound above).

Fact 13.9. *Using Grover's algorithm, a quantum computer can decide k-SAT with probability $1 - O(2^{-n})$ in time $\Theta(\sqrt{2^n})$.*

Of course, this is still exponential in n, so Grover's algorithm does not afford quantum computers the ability to solve NP-complete problems, and hence it does not follow from Grover's algorithm alone that $\mathsf{NP} \subseteq \mathsf{BQP}$.

That said, there is an interesting, theoretical application of Grover to something called the *strong exponential time hypothesis* (SETH). To understand this, let us first say what the *exponential time hypothesis* (ETH) is.

Conjecture 13.10 (Exponential Time Hypothesis)**.** *For all k, there exists $s_k > 0$ such that k-SAT cannot be solved in time less than $2^{s_k n}$ on a deterministic classical computer.*

Even with Grover, we expect the ETH to hold, albeit with different choices of s_k (namely, $s'_k = s_k/2$, do you see why?). That said, there is a strong form of the ETH that fails in a quantum world thanks to Grover.

Conjecture 13.11 (Strong Exponential Time Hypothesis)**.** *If the ETH is true, then $\lim_{k\to\infty} s_k = 1$.*

Therefore, SETH basically says that not only is the ETH true, but, in essence, that there are really only two options for each s_k, either $s_k = 1 - \Theta(1)/k$ or $s_k = 1 - \Theta(1)/\log k$. Using Grover's algorithm, however, it is possible (and straightforward) to show that SETH is false in a quantum world.

Exercise 13.4. *Argue that Grover's algorithm violates SETH.*

Still, SETH is a well-defined conjecture when restricting to classical computers, and there is in fact a quantum SETH that says you can't do better than what you get when you use Grover's algorithm. These are interesting complexity theoretic ideas that could make captivating final projects if complexity theory is your thing.

[6] If you are unfamiliar with this problem, I recommend reading the relevant parts of Arora and Barak's textbook *Computational Complexity: A Modern Approach*.

Lecture 14
Simon's Algorithm

Discussion 14.1. Discuss with your group what you took away from last time.

Last time, we saw Grover's algorithm, which solves the unstructured search problem quadratically faster than any classical algorithm. In this lecture, we will see our second quantum algorithm that provably outperforms every classical algorithm. In particular, we will study Simon's algorithm, which affords an *exponential* speedup over the best possible classical algorithm (in the query complexity paradigm).

14.1. An Aside about H

Next lecture, we will learn that the Hadamard transformation is "the quantum Fourier transform (QFT) over the group $\mathbb{Z}_2$". This is an incredibly important fact, as the QFT is *the* transformation that underlies all the fast quantum algorithms that we know. Hence, it is no surprise that H appears in Grover's algorithm and, as we shall shortly see, Simon's algorithm as well. Before discussing any sort of Fourier analysis, consider the following question (a generalization of which you will prove on HW4).

Question 14.1. *Is the following true? For all $x \in \{0,1\}$,*

$$H|x\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + (-1)^x|1\rangle\big).$$

Importantly, this generalizes in a nice way:

Fact 14.1 (HW4)**.** *For all $x \in \{0,1\}^n$,*

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y}|y\rangle,$$

where

$$\begin{aligned} x \cdot y &\coloneqq \sum_{i=0}^{n-1} x_i y_i \pmod 2 \\ &= x_0 y_0 \oplus x_1 y_1 \oplus \cdots \oplus x_{n-1} y_{n-1}. \end{aligned}$$

Notice how this looks like an inner product (a.k.a. a dot product) on a vector space. Indeed, this is the case, and the exact vector space is

$$\mathbb{F}_2^n \coloneqq \left\{ b_1 \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \oplus b_2 \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \oplus \cdots \oplus b_{n-1} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{pmatrix} \oplus b_n \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} : b_1, b_2, \ldots, b_n \in \{0,1\} \right\},$$

where $\oplus$ is element-wise addition mod 2 (and multiplication is also done mod 2).

Exercise 14.1. *Prove that* $|\mathbb{F}_2^n| = 2^n$.

These facts will come in handy shortly.

14.2. Simon's Problem

Below is Simon's problem. We note that while this problem is quite artificial (as it was designed to be a problem that a classical computer is bad at solving), it is surprising just how quickly a quantum computer can solve it (as we will see in the next section).

Simon's Problem

Input: $f : \{0,1\}^n \to \{0,1\}^n$ for which there is $s \in \{0,1\}^n \backslash \{0^n\}$ such that for all $x, y \in \{0,1\}^n$, $f(x) = f(y)$ iff $y = x$ or $y = x \oplus s$.

Output: s (the "secret string") by querying f.

Fact 14.2 (HW4)**.** *Any probabilistic classical computer that finds s with probability at least $1/2$ requires $\Omega(2^{n/2})$ queries to f.*[1]

[1] This is an example of a *birthday bound*. See HW4 for a justification of this terminology.

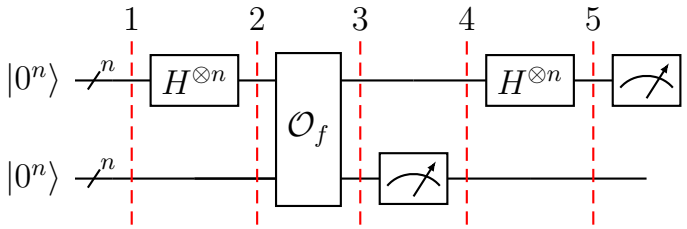
14.3. Simon's Algorithm

We will now see that we can solve Simon's problem on a quantum computer with probability $1-2^{-O(n)}$ using only *polynomially* many queries to f, thus demonstrating our first quantum *exponential* speedup. The algorithm that does this is know as *Simon's algorithm*, after computer scientist Daniel Simon.

Simon's Algorithm

1. Prepare $2n$ qubits in the state $|0^n\rangle|0^n\rangle$.
2. Apply $H^{\otimes n}$ to the first n qubits.
3. Apply the oracle $\mathcal{O}_f$ to all $2n$ qubits.
4. Measure the last n qubits.
5. Apply $H^{\otimes n}$ to the first n qubits.
6. Measure the first n qubits.
7. Repeat steps 1 – 6 $O(n)$ times, and then use Gaussian elimination to find s.

As a circuit, the quantum part of Simon's algorithm is simply:



Altogether, in this lecture we will establish the following theorem.

Theorem 14.3. *Using a quantum computer (namely, that specified by Simon's algorithm), it is possible to solve Simon's problem with probability at least* $1-2^{-O(n)}$ *using* $O(n)$ *queries to* f *and polynomial time classical post-processing. This is exponentially fewer queries than any probabilistic classical algorithm.*[2]

[2] In complexity theory, Simon's problem demonstrates what is known as an "oracle separation" between BPP and BQP. In complexity lingo, we summarize this by saying that "Simon's problem exhibits an oracle relative to which BPP $\neq$ BQP".

14.4. Correctness of Simon's Algorithm

Let us start with an elementary exercise.

Exercise 14.2. *Let* $S \coloneqq \{z \in \{0,1\}^n : z \cdot s = 0\}$*, where* $s \neq 0^n$*. Prove* $|S| = 2^{n-1}$*.*

Claim 14.4.

- *If* $z \notin S$*, then Simon's algorithm outputs* z *with probability zero.*
- *If* $z \in S$*, then Simon's algorithm outputs* z *with probability* $1/|S| = 1/2^{n-1}$*.*

Proof. Let $|\psi_i\rangle$ be the state in Simon's algorithm after step i. Then,

$$\begin{aligned}
|\psi_1\rangle &= |0^n\rangle|0^n\rangle \\
|\psi_2\rangle &= \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle|0^n\rangle \\
|\psi_3\rangle &= \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle|f(x)\rangle.
\end{aligned}$$

Measuring the last n qubits collapses the last n qubits into some state $|f(y)\rangle$. In this case, the overall $2n$ qubit state collapses to (up to normalization)

$$|\psi_4\rangle = \sum_{\substack{x\in\{0,1\}^n \\ f(x)=f(y)}} |x\rangle|f(y)\rangle.$$

Since, by the definition of f, $f(x) = f(y)$ iff $x = y$ or $x = y \oplus s$, this state is simply

$$|\psi_4\rangle = \frac{1}{\sqrt{2}}\big(|y\rangle + |y \oplus s\rangle\big)|f(y)\rangle,$$

where we've inserted the correct normalization factor. Consequently, after applying Hadamards and using the identity from before,

$$\begin{aligned}
|\psi_5\rangle &= \frac{1}{\sqrt{2}}\big(H^{\otimes n}|y\rangle + H^{\otimes n}|y \oplus s\rangle\big)|f(y)\rangle \\
&= \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{2^n}} \sum_{z\in\{0,1\}^n} \left[(-1)^{y\cdot z} + (-1)^{(y\oplus s)\cdot z}\right] |z\rangle \right) |f(y)\rangle \\
&= \frac{1}{\sqrt{2^{n+1}}} \sum_{z\in\{0,1\}^n} \left((-1)^{y\cdot z} + (-1)^{(y\oplus s)\cdot z}\right) |z\rangle|f(y)\rangle.
\end{aligned}$$

We now measure the first n qubit. The probability that we measure the first n qubits to be in the state $|z\rangle$ is

$$\Pr\left[\text{measure } |z\rangle\right] = \frac{1}{2^{n+1}} \left|(-1)^{y\cdot z} + (-1)^{(y\oplus s)\cdot z}\right|^2.$$

But it is easy to convince yourself that

$$\begin{aligned}(-1)^{(y\oplus s)\cdot z} &= (-1)^{(y\cdot z)\oplus(s\cdot z)} \\ &= (-1)^{y\cdot z}(-1)^{s\cdot z}.\end{aligned}$$

Therefore,

$$\begin{aligned}\Pr\left[\text{measure } |z\rangle\right] &= \frac{1}{2^{n+1}} \left|(-1)^{y\cdot z} + (-1)^{(y\oplus s)\cdot z}\right|^2 \\ &= \frac{1}{2^{n+1}} \left|(-1)^{y\cdot z} + (-1)^{y\cdot z}(-1)^{s\cdot z}\right|^2 \\ &= \frac{1}{2^{n+1}} \left|1 + (-1)^{s\cdot z}\right|^2 \\ &= \begin{cases} 0 & \text{if } s\cdot z = 1 \\ \frac{1}{2^{n-1}} & \text{if } s\cdot z = 0. \end{cases}\end{aligned}$$

Therefore, when we run Simon's algorithm and we measure the first n qubits, we necessarily obtain $z \in \{0,1\}^n$ such that $s \cdot z = 0$, i.e., $z \in S$. Moreover, we obtain any particular $z \in \{0,1\}^n$ such that $s \cdot z = 0$ with equal probability $1/2^{n-1}$. ■

Claim 14.5. *The probability that uniformly random $z_1, z_2, \ldots, z_m \sim S$ satisfy $z_i \cdot z_j = 0$ for all $i, j \in \{1, 2, \ldots, m\}$ is at least $0.288 = O(1)$.*

Proof. Think of each z_i as a vector $(z_{i1}, z_{i2}, \ldots, z_{in})^T \in \mathbb{F}_2^n$, $z_{ij} \in \{0,1\}$, and suppose $k < m$ vectors $z_1, z_2, \ldots, z_k$ are linearly independent (i.e., that $z_i \cdot z_j = 0$ for all distinct $i, j \in \{1, 2, \ldots, k\}$). Then, these vectors span the subspace $\mathbb{F}_2^k \subseteq \mathbb{F}_2^n$, which consists of all vectors of the form

$$b_1 z_1 \oplus b_2 z_2 \oplus \cdots \oplus b_k z_k, \qquad b_i \in \{0,1\}.$$

Now draw $z_{k+1} \sim S$ uniformly, and let us calculate the probability that it is linearly independent from $z_1, z_2, \ldots, z_k$. Evidently, z_{k+1} is linearly independent of

$z_1, z_2, \ldots, z_k$ if and only if it lies *outside* of $\mathbb{F}_2^k$. This has probability

$$\begin{aligned}
\Pr\left[z_{k+1} \notin \mathbb{F}_2^k\right] &= 1 - \Pr\left[z_{k+1} \in \mathbb{F}_2^k\right] \\
&= 1 - \frac{|\mathbb{F}_2^k|}{|\mathbb{F}_2^n|} \\
&= 1 - \frac{2^k}{2^n} \\
&= 1 - \frac{1}{2^{n-k}}.
\end{aligned}$$

Therefore, the probability of sequentially drawing m different $z_i \sim S$ such that $z_i \cdot z_j = 0$ for all distinct $i, j \in \{1, 2, \ldots, m\}$ is[3]

$$\begin{aligned}
\prod_{k=0}^{m-1} \Pr\left[z_{k+1} \notin \mathbb{F}_2^k\right] &= \prod_{k=0}^{m-1} \left(1 - \frac{1}{2^{n-k}}\right) \\
&= \left(1 - \frac{1}{2^n}\right)\left(1 - \frac{1}{2^{n-1}}\right) \cdots \left(1 - \frac{1}{2^{n-m+1}}\right) \\
&\geq \prod_{\ell=1}^{\infty} \left(1 - \frac{1}{2^\ell}\right) \\
&\approx 0.288 \\
&= O(1).
\end{aligned}$$

■

Claim 14.6. *If Simon's algorithm is run $O(n)$ times, then s can be found with probability at least $1 - 2^{-O(n)}$ by using a deterministic Turing machine that halts in time $O(n^3)$.*[4]

Proof. Running Simon's algorithm $m = n$ times (so that f is queried n times) yields n n-bit strings $z_1, z_2, \ldots, z_m \in \{0,1\}^n$ such that

$$\begin{aligned}
z_1 \cdot s &= 0 \\
z_2 \cdot s &= 0 \\
&\vdots \\
z_n \cdot s &= 0.
\end{aligned}$$

[3]The value for the infinite product follows from evaluating the rth partial product and then taking the limit $r \to \infty$.

[4]In complexity lingo, we say that we can solve Simon's problem using "polynomial time classical post-processing."

In other words, writing $z_i = (z_{i1}, z_{i2}, \ldots, z_{in})$ and $s = (s_1, s_2, \ldots, s_n)$, this system of equations becomes the matrix equation

$$\begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1n} \\ z_{21} & z_{22} & \cdots & z_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n1} & z_{n2} & \cdots & z_{nn} \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix} = 0,$$

where implicitly all operations are done mod 2. By the previous claim, the probability that the z_i vectors are mutually linearly independent (i.e., that $z_i \cdot z_j = 0$ for all distinct $i, j \in \{1, 2, \ldots, n\}$) is $O(1)$. Therefore, with constant probability, this system of equations is determined, and so we can use Gaussian elimination to solve it. By the following fact, we can solve this system *efficiently* on a classical computer.

Fact 14.7. *Using Gaussian elimination, one can find s in the previous matrix equation in $O(n^3)$ time on a deterministic Turing machine.*

Note that if after running Simon's algorithm n times, we do not obtain n linearly independent strings $z_1, \ldots, z_n$ (which happens with probability at most $1 - 0.288 \approx 0.712$), then simply run Simon's algorithm n times *again* to obtain n new n-bit strings $z'_1, z'_2, \ldots, z'_n$. As before, the probability that these are all mutually linearly independent is at least 0.288, and the probability that they are not is again at most 0.712. However, the probability that in the first and the second run we do not obtain n mutually linearly independent strings is at most $0.712^2 \approx 0.507$. And after k runs, the probability that we do *not* obtain n mutually linearly strings is at most 0.712^k. Thus, after $O(n)$ runs, the probability that Simon's algorithm will return a set of n mutually linearly independent strings is at least

$$1 - 0.712^{O(n)} = 1 - (1 - 0.288)^{O(n)} \geq 1 - e^{-0.288O(n)} = 1 - 2^{-O(n)},$$ [5]

as desired. ■

Putting all of this together, we have proven Theorem 14.3. We underscore that the essential ingredient in Simon's algorithm (and hence in proving Theorem 14.3) is the Hadamard transformation. In later lectures, we will see that the essential reason for other quantum speedups is also due to a Hadamard-like transformation, which is known as the *quantum Fourier transform* (QFT). In the next lecture, we will formally define the QFT.

[5] Here we are using the fact that for all $x \in \mathbb{R}$, $1 - x \leq e^{-x}$.

Lecture 15
The Quantum Fourier Transform

Discussion 15.1. Discuss with your group what you took away from last time.

Last time, we studied Simon's algorithm, which afforded an exponential speedup over the best possible classical algorithm (in the query complexity paradigm). Recall that Simon's algorithm used the Hadamard transformation. In this lecture, we will introduce what is a unifying ingredient of almost every fast quantum algorithm we know: the quantum Fourier transform. Here, we will not introduce any new algorithms, but we will instead study some basic properties of this transformation. In later lectures, we will use this operation again and again.

15.1. The Quantum Fourier Transform over $\mathbb{Z}_N$

Definition 15.1.

- For all $N \geq 1$, $\mathbb{Z}_N$ (sometimes denoted $\mathbb{Z}/N\mathbb{Z}$) is the *additive group of integers modulo* N, i.e., $\mathbb{Z}_N \coloneqq (\{0, 1, 2, 3, \ldots, N-1\}, + \bmod N)$.
- For all n, $\mathbb{Z}_N^n$ is the n-fold direct product group

$$\mathbb{Z}_N^n \coloneqq \underbrace{\mathbb{Z}_N \times \mathbb{Z}_N \times \cdots \times \mathbb{Z}_N}_{n \text{ times}},$$

  where addition is done element-wise mod N.

Example 15.1.

- $\mathbb{Z}_2 = (\{0, 1\}, \oplus)$, where $\oplus$ is addition mod 2 (i.e., XOR).
- $\mathbb{Z}_2^n = (\{0, 1\}^n, \oplus)$, where, in this context, $\oplus$ is *bit-wise* addition mod 2 (i.e., bit-wise XOR).[1]

[1] Note that $\mathbb{Z}_2^n$ is different from $\mathbb{F}_2^n$ because $\mathbb{Z}_2^n$ is not a vector space.

We now define the quantum Fourier transform (QFT) over $\mathbb{Z}_N$ and over $\mathbb{Z}_N^n$.[2]

Definition 15.2. Let $\{|0\rangle, |1\rangle, |2\rangle, \ldots, |N-1\rangle\}$ denote the computational basis of $\mathbb{C}^N$, where for all $\ell \in \mathbb{Z}_N$,

$$|\ell\rangle \coloneqq \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow \ell\text{th index.}$$

Moreover, let $\omega_N \coloneqq e^{2\pi i/N}$ be the Nth root of unity.

- The *quantum Fourier transform (QFT) over* $\mathbb{Z}_N$ (or just the QFT if the group is contextually clear) is the matrix[3]

$$\begin{aligned} \mathcal{F}_{\mathbb{Z}_N} &\coloneqq \frac{1}{\sqrt{N}} \sum_{k,\ell \in \mathbb{Z}_N} \omega_N^{k\ell} |k\rangle\langle\ell| \\ &= \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_N & \omega_N^2 & \omega_N^3 & \cdots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \omega_N^6 & \cdots & \omega_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \omega_N^{3(N-1)} & \cdots & \omega_N^{(N-1)(N-1)} \end{pmatrix}. \end{aligned}$$

- The *QFT over* $\mathbb{Z}_N^n$ is the n-fold tensor product of $\mathcal{F}_{\mathbb{Z}_N}$:

$$\mathcal{F}_{\mathbb{Z}_N^n} \coloneqq \underbrace{\mathcal{F}_{\mathbb{Z}_N} \otimes \mathcal{F}_{\mathbb{Z}_N} \otimes \cdots \otimes \mathcal{F}_{\mathbb{Z}_N}}_{n \text{ times}}.$$

If you are familiar with the discrete Fourier transform (DFT), then the QFT might look familiar. Indeed, they are closely related.

[2] Interestingly, the QFT exists over *any* finite group G. In the case that G is abelian, the QFT exhibits particularly nice properties that enable it to solve the so-called *abelian hidden subgroup problem*, which generalizes Simon's algorithm and Shor's algorithm. Exploring this would make a great final project if you are mathematically inclined.

[3] Incidentally, $\mathcal{F}_{\mathbb{Z}_N}$ is an example of a *Vandermonde matrix*, which is the general type of matrix that exhibits the structure you see in $\mathcal{F}_{\mathbb{Z}_N}$.

Fact 15.1. *Let*

$$|\psi\rangle = \begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{N-1} \end{pmatrix} \in \mathbb{C}^N,$$

where we are expressing the vector in the computational basis. Then,

$$\mathcal{F}_{\mathbb{Z}_N}|\psi\rangle = |\tilde{\psi}\rangle \coloneqq \begin{pmatrix} \tilde{\psi}_0 \\ \tilde{\psi}_1 \\ \vdots \\ \tilde{\psi}_{N-1} \end{pmatrix} \in \mathbb{C}^N,$$

where for all $k \in \{0, 1, \ldots, N-1\}$,

$$\tilde{\psi}_k \coloneqq \frac{1}{\sqrt{N}} \sum_{\ell=0}^{N-1} \omega_N^{k\ell} \psi_\ell.$$

Therefore, up to normalization, $\mathcal{F}_{\mathbb{Z}_N}$ *implements the* inverse *DFT of the sequence* $\psi_0, \psi_1, \ldots, \psi_{N-1}$.

Note, however, that the QFT encodes the inverse DFT numbers $\tilde{\psi}_k$ in the *amplitudes* of the quantum state. Thus, the QFT does not actually compute the DFT, because we cannot directly access quantum amplitudes.

To build some intuition for $\mathcal{F}_{\mathbb{Z}_N}$ and $\mathcal{F}_{\mathbb{Z}_N^n}$, consider the case when $N = 2$.

Exercise 15.1.

- *Prove that for all* $x \in \{0, 1\} = \mathbb{Z}_2$,

$$\mathcal{F}_{\mathbb{Z}_2}|x\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + (-1)^x|1\rangle\big).$$

- *What transformation is* $\mathcal{F}_{\mathbb{Z}_2}$*?*

By a result on HW 4, you've just proved the following incredibly useful fact:

Fact 15.2 (HW4)**.** *For all* $x \in \{0, 1\}^n$,

$$\mathcal{F}_{\mathbb{Z}_2^n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y}|y\rangle,$$

where

$$x \cdot y \coloneqq \sum_{i=0}^{n-1} x_i y_i \pmod 2.$$

In consequence, we learn that the Deutsch-Jozsa algorithm, Grover's algorithm, and Simon's algorithm all used the QFT over $\mathbb{Z}_2^n$ as a key subroutine. Interestingly, we will see in a few lectures that many other important quantum algorithms employ the QFT over the more general $\mathbb{Z}_N$ as a key subroutine.

15.2. Properties of $\mathcal{F}_{\mathbb{Z}_N}$

Where we're going, then, is to use the QFT as a gate in some larger quantum circuit. Of course, to do this—to implement $\mathcal{F}_{\mathbb{Z}_N}$ on a quantum computer—presupposes that $\mathcal{F}_{\mathbb{Z}_N}$ is a unitary transformation. Thankfully, this turns out to be true.

Fact 15.3 (HW5)**.** *For all $N \geq 1$, $\mathcal{F}_{\mathbb{Z}_N} \in \mathrm{U}(N)$.*

In consequence, $\mathcal{F}_{\mathbb{Z}_N}$ is a valid quantum operation, so we can implement it on a quantum computer. In the next section, we will see that we can do this quite efficiently. Before this, however, let's build a little intuition for why the QFT is so useful.

Definition 15.3.

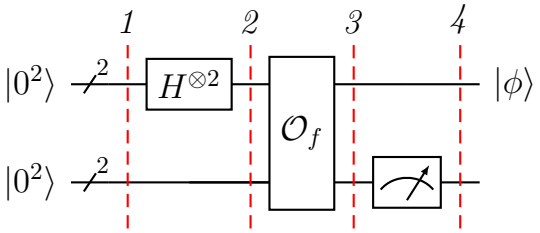
- A function $f : \mathbb{Z} \to \mathbb{Z}$ is *r-periodic* iff for all $k \in \mathbb{Z}$,

$$f(k) = f(k + r).$$

- A state $|\phi\rangle \in \mathbb{C}^N$ is *r-periodic with shift s* iff in the computational basis

$$|\phi\rangle = \sqrt{\frac{r}{N}} \sum_{k=0}^{N/r-1} |s + kr\rangle.$$

Example 15.2. Let $f : \mathbb{Z} \to \mathbb{Z}$ be 2-periodic and such that $f(0) \neq f(1)$. Then, the state $|\phi\rangle$ in the output of the circuit below is 2-periodic with a random shift $s \in \mathbb{Z}_2$.



To see this, let $|\psi_i\rangle$ denote the state at step i in the circuit. Then,

$$\begin{aligned}
|\psi_1\rangle &= |0^2\rangle|0^2\rangle \\
|\psi_2\rangle &= \left(\frac{1}{\sqrt{2^2}}\sum_{k\in\mathbb{Z}_4}|k\rangle\right)\otimes|0^2\rangle \\
|\psi_3\rangle &= \frac{1}{2}\sum_{k\in\mathbb{Z}_4}|k\rangle|f(k)\rangle \\
&= \frac{1}{2}\Big(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle + |2\rangle|f(2)\rangle + |3\rangle|f(3)\rangle\Big) \\
&= \frac{1}{2}\Big(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle + |2\rangle|f(0)\rangle + |3\rangle|f(1)\rangle\Big) \\
&= \frac{1}{2}\Big[\Big(|0\rangle + |2\rangle\Big)|f(0)\rangle + \Big(|1\rangle + |3\rangle\Big)|f(1)\rangle\Big],
\end{aligned}$$

where the last two lines follow from the 2-periodicity of f. Measuring the second register of $|\psi_3\rangle$ collapses the register to $|f(s)\rangle$ for some random shift $s \in \mathbb{Z}_2$, and the first register collapses to all states $|\ell\rangle$ that are consistent with the second register collapsing to $|f(s)\rangle$. The possibilities are:

$$|\psi_4\rangle = \begin{cases} \frac{1}{\sqrt{2}}\left(|0\rangle + |2\rangle\right)|f(0)\rangle & \text{if } s = 0 \\ \frac{1}{\sqrt{2}}\left(|1\rangle + |3\rangle\right)|f(1)\rangle & \text{if } s = 1, \end{cases}$$

where we've updated the normalization factor. Therefore,

$$|\phi\rangle = \begin{cases} \frac{1}{\sqrt{2}}\left(|0\rangle + |2\rangle\right) & \text{if } s = 0 \\ \frac{1}{\sqrt{2}}\left(|1\rangle + |3\rangle\right) & \text{if } s = 1. \end{cases}$$

Therefore, $|\phi\rangle$ is a 2-periodic state with some shift $s \in \mathbb{Z}_2$.

This example is important as it hints at how we will use the QFT in future algorithms (e.g., Shor's algorithm). In particular, what the example illustrates is that if we have an oracle for some function that is promised to be periodic, then after applying the circuit above, we get out a state that is periodic *with the same period as the function.* As the following fact shows, we can then use the QFT to learn some information about the period.

Fact 15.4 (HW5)**.** *If $|\phi\rangle \in \mathbb{C}^N$ is r-periodic with shift s, then*

$$\begin{aligned}
|\tilde{\phi}\rangle &= \mathcal{F}_{\mathbb{Z}_N}|\phi\rangle \\
&= \sum_{\ell=0}^{r-1}\alpha_\ell|\ell N/r\rangle,
\end{aligned}$$

where for all $\ell \in \{0, 1, 2, \ldots, r-1\}$, $|\alpha_\ell| = \frac{1}{\sqrt{r}}$.[4]

Therefore, by measuring $|\tilde{\phi}\rangle$, we learn information about the period r of $|\phi\rangle$. Hence, if we have a function that is r-periodic, and then we apply the circuit from the example above, then applying the QFT to the state will recover information about the period. This is the key fact of the QFT that we will exploit in later lectures to factor integers, solve the discrete logarithm problem, and so forth. We will explore this idea in detail when we talk about the quantum period finding algorithm in a few lectures.

15.3. Implementing $\mathcal{F}_{\mathbb{Z}_N}$ on a Quantum Computer

Above, we saw that $\mathcal{F}_{\mathbb{Z}_N}$ is a unitary operator. Therefore, we can in principle implement it on a quantum computer. Here, we will see how to do this, and we will see that we can do this quite efficiently, i.e., with just a small number of quantum gates. To prove this, we require some new notation for talking about the binary representation of fractions.

Definition 15.4. Given n bits $x_0, x_1, \ldots, x_{n-1} \in \{0, 1\}$, we write $[0.x_0 x_1 \ldots x_{n-1}]$ for the following rational number (which is an example of a *dyadic fraction*),

$$\begin{aligned}[0.x_0 x_1 \ldots x_{n-1}] &\coloneqq \sum_{r=0}^{n-1} \frac{x_r}{2^{r+1}} \\ &= \frac{x_0}{2^1} + \frac{x_1}{2^2} + \cdots + \frac{x_{n-1}}{2^n}.\end{aligned}$$

More generally, if $j \in \{0, 1, 2, \ldots, n-1\}$, we write

$$\begin{aligned}[0.x_j x_{j+1} \ldots x_{n-1}] &\coloneqq \sum_{r=j}^{n-1} \frac{x_r}{2^{r+1-j}} \\ &= \frac{x_j}{2^1} + \frac{x_{j+1}}{2^2} + \cdots + \frac{x_{n-1}}{2^{n-j}}.\end{aligned}$$

Using this notation, we will now specialize to $N = 2^n$ and state a simple but important fact about how $\mathcal{F}_{\mathbb{Z}_{2^n}}$ acts on the computational basis.

[4] In fact, it turns out that $\alpha_\ell = e^{2\pi i \ell s / r}/\sqrt{r}$.

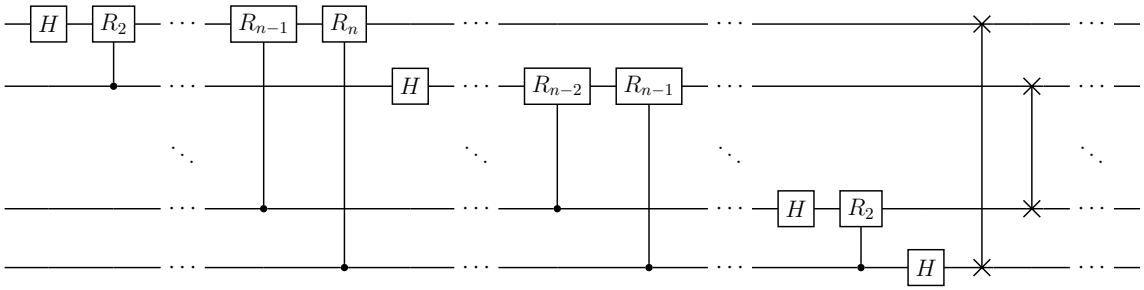
Fact 15.5 (HW5). *For all* $x = x_0 x_1 \dots x_{n-1} \in \{0,1\}^n$,

$$\begin{aligned}\mathcal{F}_{\mathbb{Z}_{2^n}}|x\rangle &= \bigotimes_{j=n-1}^{0} \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_j x_{j+1}\dots x_{n-1}]}|1\rangle\right) \\ &= \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_{n-1}]}|1\rangle\right) \otimes \cdots \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_0 x_1 \dots x_{n-1}]}|1\rangle\right).\end{aligned}$$

Consequently, $\mathcal{F}_{\mathbb{Z}_N}$ does not entangle computational basis states. Moreover, as you will explore on the homework, this fact allows one to contrive a circuit that implements $\mathcal{F}_{\mathbb{Z}_{2^n}}$. For this, we need the following T-like gate, which is easily seen to be unitary:

$$R_k \coloneqq \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}.$$

Fact 15.6 (HW5). *The following circuit exactly implements* $\mathcal{F}_{\mathbb{Z}_{2^n}}$*:*



Importantly, this is also an *efficient* implementation of $\mathcal{F}_{\mathbb{Z}_{2^n}}$.[5]

Exercise 15.2. *Prove that the number of gates in this circuit is*

$$\frac{n(n+1)}{2} + \left\lfloor \frac{n}{2} \right\rfloor = O(n^2).$$

Next lecture, we will use this fact to find the eigenvalues of a given unitary operator rather efficiently (at least in certain cases). After that, we will demonstrate how to efficiently find the period of any periodic function $f : \mathbb{Z} \to \mathbb{Z}$, which, in turn, will allow us to efficiently factor integers (after we recast the factoring problem using some cool number theory).

[5] Note that many of the R_k rotations in the above circuit have exponentially small entries (of order 2^{-k}). Thus, for large k, the gate is approximately the identity gate. One can exploit this fact to get an even more efficient implementation of the QFT. Indeed, this is precisely the idea behind the *approximate QFT*: ignore all R_k gates for k greater than some threshold. Interestingly, the AQFT reduces the size of the QFT to $O(n \log n)$.

Lecture 16
The Quantum Phase Estimation Algorithm

Discussion 16.1. Discuss with your group what you took away from last time.

Last time, we discussed the quantum Fourier transform (QFT) over the group $\mathbb{Z}_N$. We saw that this transformation, denoted $\mathcal{F}_{\mathbb{Z}_N}$, is a unitary, $N \times N$ matrix that allows us to deduce period information from periodic states. We also saw that the QFT over $\mathbb{Z}_2$ is the Hadamard transformation, which is a critical element in both Grover's algorithm and Simon's algorithm. In this lecture, we will use the QFT to solve a problem known as the *phase estimation problem*. The quantum algorithm that does this is called the *quantum phase estimation (QPE) algorithm*, and, like the QFT, it is an important subroutine in many quantum algorithms.

16.1. The Phase Estimation Problem

The phase estimation problem turns out to be a key problem to be able to solve efficiently on a quantum computer. It is as follows.

The Phase Estimation Problem

Input: $n \in \mathbb{N}$, an oracle that implements

$$\Lambda^n(U) \coloneqq \sum_{k \in \mathbb{Z}_{2^n}} |k\rangle\langle k| \otimes U^k$$

for some $U \in \mathrm{U}(2^m)$, and $|\sigma\rangle \in \mathbb{C}^{2^m}$ such that $U|\sigma\rangle = e^{2\pi i\theta}|\sigma\rangle$.[1]

Output: The first n bits in the binary expansion of $\theta \in [0, 1)$.

On the surface, this problem is a bit artificial (after all, the oracle $\Lambda^n(U)$ is rather specific!). Nevertheless, as we will see next lecture, this problem is related to period finding on a quantum computer, which itself is related to the problems of finding primes factors, finding discrete logarithms, and finding hidden subgroups.

[1] Recall, the eigenvalue of any unitary is $e^{i\phi}$ for some $\phi \in [0, 2\pi)$, i.e., it is $e^{2\pi i\theta}$ for some $\theta \in [0, 1)$.

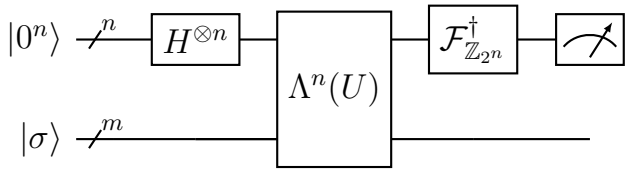
16.2. The Quantum Phase Estimation Algorithm

We will now see that we can solve the phase estimation problem on a quantum computer with high probability using the quantum phase estimation algorithm. In certain cases, this algorithm is efficient.

The Quantum Phase Estimation (QPE) Algorithm

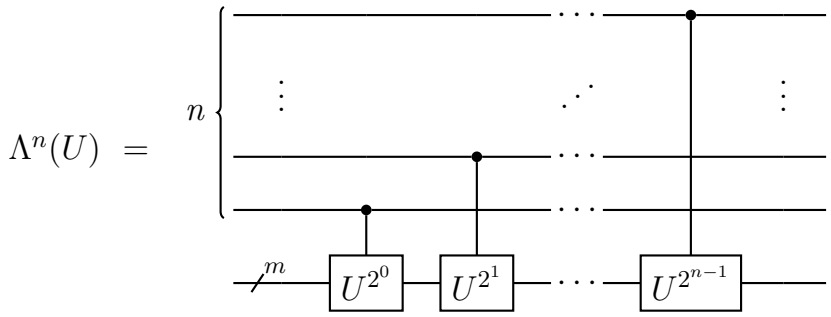
1. Prepare n qubits in the state $|0^n\rangle$. Incorporate this with the inputted, m-qubit state $|\sigma\rangle$, so that the initial state of the computer is $|0^n\rangle|\sigma\rangle$.
2. Apply $H^{\otimes n}$ to the first n qubits.
3. Apply the oracle $\Lambda^n(U)$.
4. Apply the inverse QFT, $\mathcal{F}_{\mathbb{Z}_{2^n}}^{-1} = \mathcal{F}_{\mathbb{Z}_{2^n}}^{\dagger}$, to the first n qubits.
5. Measure the first n qubits.

As a circuit, the QPE algorithm is simply



In fact, there is a more workable version of this circuit, which relies on an alternate representation of the $\Lambda^n(U)$ operator.

Claim 16.1. *For all n and all $U \in \mathrm{U}(2^m)$,*

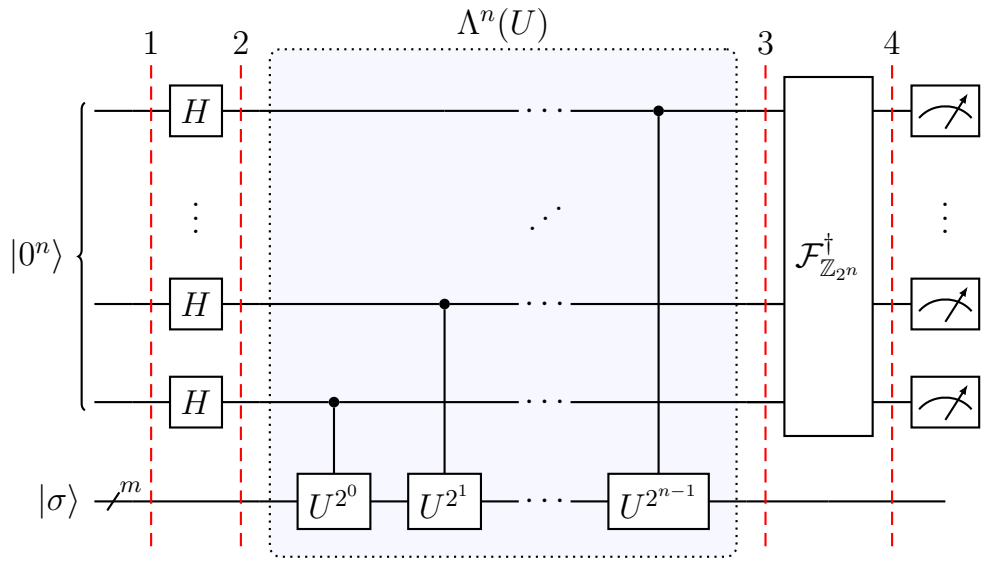


Proof. For all $|\sigma\rangle \in \mathbb{C}^{2^m}$ and all $\ell \in \mathbb{Z}_{2^n}$ with binary expansion $\ell = x_0 2^{n-1} + x_1 2^{n-2} + \cdots + x_{n-1} 2^0$, it holds that

$$\begin{aligned}\Lambda^n(U)|\ell\rangle|\sigma\rangle &= |\ell\rangle \otimes U^\ell |\sigma\rangle \\ &= |x_0 x_1 \ldots x_{n-1}\rangle \otimes U^{x_0 2^{n-1} + x_1 2^{n-2} + \cdots + x_{n-1} 2^0} |\sigma\rangle \\ &= |x_0 x_1 \ldots x_{n-1}\rangle \otimes U^{x_0 2^{n-1}} U^{x_1 2^{n-2}} \cdots U^{x_{n-1} 2^0} |\sigma\rangle,\end{aligned}$$

from which the above circuit identity follows. ■

As a circuit, then, the phase estimation algorithm is simply:



In this lecture, we will prove the following theorem.

Theorem 16.2. *Let $T_n(U)$ be the minimum number of 1- and 2-qubit gates needed to exactly implement $\Lambda^n(U)$. Then, there exists a quantum computer (namely, that specified by the QPE algorithm) of size $O(T_n(U) + n + n^2)$ that outputs an n-bit approximation of θ (and hence solves the phase estimation problem) with probability at least $1 - \frac{\pi^2}{12} \approx 0.178$.[2] Therefore, if $T_n(U)$ is a polynomial, then by the probability amplification theorem, there exists an efficient quantum computer that computes an n-bit approximation of θ with probability at least $1 - 2^{-O(\mathrm{poly}(n))}$.*

To prove this theorem, we will prove several intermediate claims. Next lecture, we will see that, in certain cases, $\Lambda^n(U)$ relates to the canonical oracle function $\mathcal{O}_f$

[2]Here, the n in the big oh is from the n Hadamards, and the n^2 is from the implementation of $\mathcal{F}^\dagger_{\mathbb{Z}_{2^n}}$.

for some particular choice of function f. This will imply that if we can implement $\mathcal{O}_f$ efficiently, then we can simulate the effect of $\Lambda^n(U)$ efficiently as well (which means that $T_n(U)$ is a polynomial in n). For now, however, let us focus on why the QPE algorithm works.

Claim 16.3. *Let $|\psi_i\rangle$ be the state of the QPE algorithm after step i. Then,*

$$|\psi_4\rangle = \left(\frac{1}{2^n}\sum_{k,\ell\in\mathbb{Z}_{2^n}} e^{2\pi i k(\theta-\ell 2^{-n})}|\ell\rangle\right)\otimes|\sigma\rangle.$$

Proof. We find:

$$\begin{aligned}|\psi_1\rangle &= |0^n\rangle|\sigma\rangle\\ |\psi_2\rangle &= \left(\frac{1}{\sqrt{2^n}}\sum_{j\in\mathbb{Z}_{2^n}}|j\rangle\right)|\sigma\rangle.\end{aligned}$$

By definition, the controlled operations satisfy

$$\begin{aligned}|\psi_3\rangle &= \left(\sum_{k\in\mathbb{Z}_{2^n}}|k\rangle\langle k|\otimes U^k\right)|\psi_2\rangle\\ &= \frac{1}{\sqrt{2^n}}\sum_{k\in\mathbb{Z}_{2^n}}|k\rangle\otimes(U^k|\sigma\rangle)\\ &= \frac{1}{\sqrt{2^n}}\sum_{k\in\mathbb{Z}_{2^n}}e^{2\pi i k\theta}|k\rangle\otimes|\sigma\rangle.\end{aligned}$$

We now apply $\mathcal{F}^\dagger_{\mathbb{Z}_{2^n}}$ to the first n qubits to get

$$\begin{aligned}|\psi_4\rangle &= \left(\mathcal{F}^\dagger_{\mathbb{Z}_{2^n}}\otimes I_{2^m}\right)|\psi_3\rangle\\ &= \left(\frac{1}{\sqrt{2^n}}\sum_{k\in\mathbb{Z}_{2^n}}e^{2\pi i k\theta}\mathcal{F}^\dagger_{\mathbb{Z}_{2^n}}|k\rangle\right)\otimes|\sigma\rangle.\end{aligned}$$

By the definition of the inverse QFT,

$$\begin{aligned}\mathcal{F}^\dagger_{\mathbb{Z}_{2^n}}|k\rangle &= \frac{1}{\sqrt{2^n}}\sum_{\ell\in\mathbb{Z}_{2^n}}(\omega^{\ell k}_{2^n})^*|\ell\rangle\\ &= \frac{1}{\sqrt{2^n}}\sum_{\ell\in\mathbb{Z}_{2^n}}e^{-2\pi i\ell k/2^n}|\ell\rangle,\end{aligned}$$

so it holds that

$$|\psi_4\rangle = \left(\frac{1}{2^n}\sum_{k,\ell\in\mathbb{Z}_{2^n}} e^{2\pi ik\theta - 2\pi ik\ell/2^n}|\ell\rangle\right)\otimes|\sigma\rangle$$
$$= \left(\frac{1}{2^n}\sum_{k,\ell\in\mathbb{Z}_{2^n}} e^{2\pi ik(\theta-\ell 2^{-n})}|\ell\rangle\right)\otimes|\sigma\rangle.$$

■

We will now study what happens when we measure the first n qubits of this state by considering two different cases: the so-called "exact" and "non-exact" cases. In both of these cases, the key to the correctness proof is the *geometric sum formula*.

Exercise 16.1. *Prove that for all complex numbers a, r and all positive integers N,*

$$a + ar + ar^2 + \cdots + ar^{N-1} = \begin{cases} aN & \text{if } r = 1 \\ a\cdot\frac{1-r^N}{1-r} & \text{otherwise.} \end{cases}$$

Here, r is called the common ratio.

16.3. Correctness of QPE: The Exact Case

In the exact case, we assume that θ has a binary expansion with at most n bits. This implies that $2^n\theta \in \mathbb{Z}_{2^n}$. It will turn out that in this case, QPE solves the phase estimation problem *exactly*.

Claim 16.4. *If $2^n\theta \in \mathbb{Z}_{2^n}$, then*

$$|\psi_4\rangle = |2^n\theta\rangle|\sigma\rangle.$$

Thus, in this case, if we measure the first n qubits of $|\psi_4\rangle$, then with probability one we obtain θ exactly, and hence we solve the phase estimation problem exactly.

Proof. Since $2^n\theta \in \mathbb{Z}_{2^n}$, the representation of $|\psi_4\rangle$ in the previous claim can be split

into a sum over all $\ell \neq 2^n\theta$ plus $\ell = 2^n\theta$:

$$\begin{aligned}
|\psi_4\rangle &= \left(\frac{1}{2^n}\sum_{k,\ell\in\mathbb{Z}_{2^n}} e^{2\pi ik(\theta-\ell 2^{-n})}|\ell\rangle\right)\otimes|\sigma\rangle \\
&= \left(\frac{1}{2^n}\sum_{\substack{k,\ell\in\mathbb{Z}_{2^n}\\ \ell\neq 2^n\theta}} e^{2\pi ik(\theta-\ell 2^{-n})}|\ell\rangle\right)\otimes|\sigma\rangle + \frac{1}{2^n}\sum_{k\in\mathbb{Z}_{2^n}}|2^n\theta\rangle\otimes|\sigma\rangle \\
&= \left(\frac{1}{2^n}\sum_{\substack{k,\ell\in\mathbb{Z}_{2^n}\\ \ell\neq 2^n\theta}} e^{2\pi ik(2^n\theta-\ell)2^{-n}}|\ell\rangle\right)\otimes|\sigma\rangle + \frac{1}{2^n}\sum_{k\in\mathbb{Z}_{2^n}}|2^n\theta\rangle\otimes|\sigma\rangle.
\end{aligned}$$

The first sum involves a geometric sum over k with common ratio $e^{2\pi i(2^n\theta-\ell)2^{-n}}$. Since $2^n\theta \in \mathbb{Z}_{2^n}$, if $\ell \in \mathbb{Z}_{2^n}\backslash\{2^n\theta\}$, then this ratio is never one. This implies that for all $\ell \in \mathbb{Z}_{2^n}\backslash\{2^n\theta\}$,

$$\sum_{k\in\mathbb{Z}_{2^n}} e^{2\pi ki(2^n\theta-\ell)2^{-n}} = \frac{1-e^{2\pi i(2^n\theta-\ell)}}{1-e^{2\pi i(\hat{\ell}-\ell)2^{-n}}} = 0,$$

so

$$\begin{aligned}
|\psi_4\rangle &= \frac{1}{2^n}\sum_{k\in\mathbb{Z}_{2^n}}|2^n\theta\rangle\otimes|\sigma\rangle \\
&= |2^n\theta\rangle|\sigma\rangle,
\end{aligned}$$

as desired. ■

16.4. Correctness of QPE: The Non-Exact Case*

In the non-exact case, we assume that θ has a binary expansion with more than n bits. This includes the case when θ is irrational. The analysis here is more complicated than the exact case, but the result is basically the same: QPE outputs an n-bit estimate of θ with high probability. To prove this, we begin by studying what the amplitudes of $|\psi_4\rangle$ look like in the non-exact case.

Claim 16.5. *Suppose $2^n\theta \notin \mathbb{Z}_{2^n}$, and let α_ℓ be the amplitude of $|\ell\rangle|\sigma\rangle$ in $|\psi_4\rangle$, i.e.,*

$$\alpha_\ell := \frac{1}{2^n}\sum_{k\in\mathbb{Z}_{2^n}} e^{2\pi ik(\theta-\ell 2^{-n})}.$$

Then,

$$\alpha_\ell = \frac{1}{2^n} \cdot \frac{e^{\pi i(2^n\theta - \ell)}}{e^{\pi i(\theta - \ell 2^{-n})}} \cdot \frac{\sin[\pi(2^n\theta - \ell)]}{\sin[\pi(\theta - \ell 2^{-n})]}.$$

Proof. For all $\ell \in \mathbb{Z}_{2^n}$, α_ℓ is 2^{-n} times a geometric sum over k with common ratio $e^{2\pi i(\theta - \ell 2^{-n})}$. This ratio is never one because $2^n\theta \notin \mathbb{Z}_{2^n}$. Therefore,

$$\begin{aligned}
\alpha_\ell &= \frac{1}{2^n} \cdot \frac{1 - e^{2\pi i(\theta 2^n - \ell)}}{1 - e^{2\pi i(\theta - \ell 2^{-n})}} \\
&= \frac{1}{2^n} \cdot \frac{e^{\pi i(\theta 2^n - \ell)}}{e^{\pi i(\theta - \ell 2^{-n})}} \cdot \frac{e^{-\pi i(\theta 2^n - \ell)} - e^{\pi i(\theta 2^n - \ell)}}{e^{-\pi i(\theta - \ell 2^{-n})} - e^{\pi i(\theta - \ell 2^{-n})}} \\
&= \frac{1}{2^n} \cdot \frac{e^{\pi i(\theta 2^n - \ell)}}{e^{\pi i(\theta - \ell 2^{-n})}} \cdot \frac{e^{\pi i(\theta 2^n - \ell)} - e^{-\pi i(\theta 2^n - \ell)}}{e^{\pi i(\theta - \ell 2^{-n})} - e^{-\pi i(\theta - \ell 2^{-n})}}.
\end{aligned}$$

By

$$\sin x = \frac{e^{ix} - e^{-ix}}{2i},$$

it follows that

$$\alpha_\ell = \frac{1}{2^n} \cdot \frac{e^{\pi i(2^n\theta - \ell)}}{e^{\pi i(\theta - \ell 2^{-n})}} \cdot \frac{\sin[\pi(2^n\theta - \ell)]}{\sin[\pi(\theta - \ell 2^{-n})]},$$

as desired. ∎

Claim 16.6. *Let $b = \lfloor 2^n\theta \rceil \in \mathbb{Z}_{2^n}$ be the nearest integer to $2^n\theta$. Then,*

$$\Pr[\text{measure } |b\rangle] \geq 1 - \frac{\pi^2}{12} \approx 0.178.$$

Proof. By Claim 16.5,

$$\begin{aligned}
\Pr[\text{measure } |b\rangle] &= |\alpha_b|^2 \\
&= \frac{1}{2^{2n}} \cdot \frac{\sin^2[\pi(2^n\theta - b)]}{\sin^2[\pi(\theta - b2^{-n})]}.
\end{aligned}$$

Using the Taylor expansion of $\sin x$, it is easy to reason that for all $x \in \mathbb{R}$,

$$x^2 - \frac{1}{3}x^4 \leq \sin^2 x \leq x^2.$$

Therefore,

$$\begin{aligned}\Pr\left[\text{measure } |b\rangle\right] &\geq \frac{1}{2^{2n}} \cdot \frac{\pi^2(2^n\theta - b)^2 - \frac{1}{3}\pi^4(2^n\theta - b)^4}{\pi^2(\theta - b2^{-n})^2} \\ &= \frac{\pi^2(2^n\theta - b)^2 - \frac{1}{3}\pi^4(2^n\theta - b)^4}{\pi^2(2^n\theta - b)^2} \\ &= 1 - \frac{\pi^2(2^n\theta - b)^2}{3}.\end{aligned}$$

Since b is the nearest integer to $2^n\theta$, there exists $0 \leq \epsilon \leq \frac{1}{2}$ such that $|b - 2^n\theta| \leq \epsilon$. Therefore, $|b - 2^n\theta|^2 = (b - 2^n\theta)^2 \leq \epsilon^2 \leq \frac{1}{4}$. Consequently,

$$\begin{aligned}\Pr\left[\text{measure } |b\rangle\right] &\geq 1 - \frac{\pi^2(2^n\theta - b)^2}{3} \\ &\geq 1 - \frac{\pi^2 \frac{1}{4}}{3} \\ &= 1 - \frac{\pi^2}{12},\end{aligned}$$

as desired. ■

Altogether, we have proven Theorem 16.2. Next time, we will see that QPE allows us find the period of a general type of periodic function.

Lecture 17
The Quantum Period Finding Algorithm

Discussion 17.1. Discuss with your group what you took away from last time.

Last time, we discussed the quantum phase estimation (QPE) algorithm, which solves the phase estimation problem. In this lecture, we will study the quantum period finding (QPF) algorithm, which allows us to find the period of a general periodic function defined on the integers. Interestingly, the quantum part of QPF is really just QPE in disguise, which, if time permits, we will discuss at the end of the lecture. Next lecture, we will use period finding (which again is really just QPE, which in turn heavily relies on the quantum Fourier transform) to efficiently factor integers.

17.1. The Period Finding Problem

The period finding problem is a natural problem to due with univariate periodic functions. It is defined as follows.

The Period Finding Problem

Input: $M \in \mathbb{N}$ and $f : \mathbb{Z} \to \mathbb{Z}$ as an oracle such that:

- f is r-periodic for some $r > 0$,
- $r \leq M$ (i.e., M upper-bounds the period r),
- $f(x)$ exists for all $x \in \mathbb{Z}$ (i.e., f is total),
- $f(0), f(1), \ldots, f(r-1)$ are all distinct.

Output: the period r by querying f.

Given the constraints in the period finding problem, the domain of f is $\mathbb{Z}_r$ and the codomain of f is a subset of $\mathbb{Z}$ that is isomorphic to $\mathbb{Z}_r$. Technically, then, we may regard f as a map from $\mathbb{Z}_r$ to $\mathbb{Z}_r$. However, because we only know M and we

do not know r, in our ignorance we can only regard f as a map from $\mathbb{Z}_M$ to $\mathbb{Z}_M$, where within $\mathbb{Z}_M$ we are guaranteed to "see" the period r, because we know $M > r$. In fact, as we shall shortly see, our approximation of f will be even coarser, as we will instead regard f as a map from $\mathbb{Z}_N$ to $\mathbb{Z}_N$, where N is a power of two such that $N \geq M^2$. This coarser approximation will be necessary to recover the period r in the classical post-processing step of the quantum period finding algorithm.

Question 17.1. *Classically, how many queries to f suffice to determine r?*

In fact, in general, one requires $\Omega(M) = \Omega(r)$ queries classically.

17.2. The Quantum Period Finding Algorithm

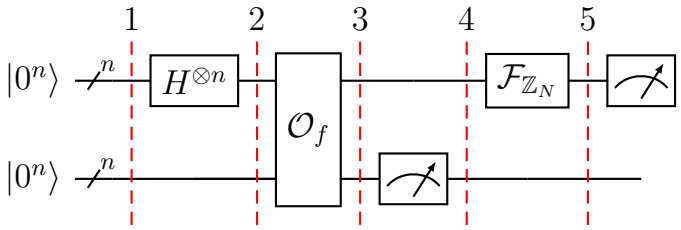
We will now see that we can solve the period finding problem on a quantum computer with high probability using the QPF algorithm. In a later section, we will see that this algorithm is just QPE in disguise for a particular choice of unitary U. It follows that if we can do QPF efficiently, then we can do QPE efficiently for that choice of U, and vice versa.

The Quantum Period Finding (QPF) Algorithm

1. Prepare $2n$ qubits in the state $|0^n\rangle|0^n\rangle$, where $N = 2^n \geq M^2$.[1]
2. Apply $H^{\otimes n}$ to the first n qubits.
3. Apply $\mathcal{O}_f$.
4. Measure the last n qubits.
5. Apply $\mathcal{F}_{\mathbb{Z}_N}$ to the first n qubits.
6. Measure the first n qubits.
7. Apply the *continued fractions algorithm* to the measurement result.

As a circuit, then, the quantum part of the period finding algorithm is simply:

[1] In fact, only the first register requires this many qubits. The second register merely needs $\lfloor \log(M-1) \rfloor + 1$ qubits, which allows you to save space. The reason is because f can only ever be as large as r, which we know is strictly less than M. We will not proceed with this assumption here, but it is a minimal modification that you can make and the proofs we give in this lecture are ultimately unaffected by that change.



Notice, this circuit is essentially the same as the circuit in Simon's algorithm, except here we are applying the QFT over $\mathbb{Z}_N$, whereas in Simon's algorithm we applied the QFT over $\mathbb{Z}_2^n$ (namely, $H^{\otimes n}$).

In this lecture, we will establish the following theorem.

Theorem 17.1. *There exists an efficient quantum computer (namely, that specified by the QPF algorithm) that outputs r (and hence solves the period finding problem) with probability*

$$\Omega\left(\frac{1}{\log\log r}\right) = \Omega\left(\frac{1}{\log\log 2^n}\right) = \Omega\left(\frac{1}{\log n}\right)$$

using just one query to f. Thus, there exists an efficient quantum computer that outputs r with probability at least $1-2^{-O(\mathrm{poly}(n))}$ using $O(\mathrm{poly}(n)\log n) = O(\mathrm{poly}(n))$ queries to f.

To prove this theorem, we will break the proof up into an "exact" case and a "non-exact" case, similar to what we did for QPE.

17.3. Correctness of QPF: The Exact Case

In the exact case, we assume that $r \mid N$, so that there exists an integer t such that $t = N/r$. It will turn out that in this case, QPF solves the period finding problem *exactly*.

Fact 17.2 (HW5)**.** *Let $|\psi_i\rangle$ be the state of the QPF algorithm after step i. If $r \mid N$, then*

$$|\psi_5\rangle = \left(\frac{1}{\sqrt{r}} \sum_{\ell \in \mathbb{Z}_r} e^{2\pi i \ell s/N} |N\ell/r\rangle\right) \otimes |f(s)\rangle,$$

where $s \in \mathbb{Z}_N$. Thus, in this case, if we measure the first n qubits of $|\psi_5\rangle$, then with probability one we obtain an integer multiple of $t = N/r$.

As indicated, you will prove this on HW5. The proof involves carefully tracking the state as it evolves through the QPF circuit above and then (crucially) applying

the assumption that $r \mid N$. Shortly, we will see how to recover the period r given N, M, and an integer multiple of N/r. First, however, let us discuss what $|\psi_5\rangle$ looks like in the non-exact case.

17.4. Correctness of QPF: The Non-Exact Case

In the non-exact case, we assume that $r \nmid N$, so that r does *not* divide N. It will turn out that in this case, QPF solves the period finding problem with a probability that is bounded below by a constant. The analysis here is more complicated than the exact case, but the essential idea is the same. Also, you may note the similarity between this analysis and the analysis we gave for the non-exact case of QPE. This is not a coincidence (see the final section in these notes).

Claim 17.3. *If $r \nmid N$, then*

$$|\psi_5\rangle = \left(\frac{1}{\sqrt{tN}} \sum_{\ell \in \mathbb{Z}_N} \frac{e^{\pi i \ell (2s+rt)/N}}{e^{\pi i r \ell / N}} \cdot \frac{\sin[\pi r t \ell / N]}{\sin[\pi r \ell / N]} |\ell\rangle \right) \otimes |f(s)\rangle,$$

where $s \in \mathbb{Z}_N$ and

$$t = \begin{cases} \lfloor \frac{N}{r} \rfloor + 1 & \text{if } s < N - r\lfloor \frac{N}{r} \rfloor \\ \lfloor \frac{N}{r} \rfloor & \text{otherwise.} \end{cases}$$

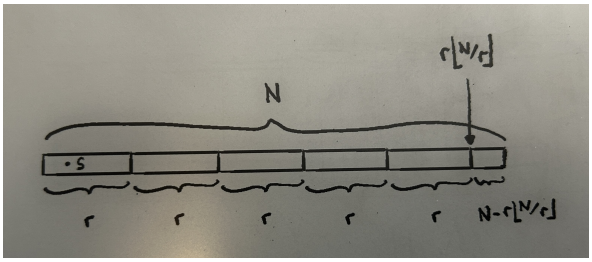
Proof.* We find:

$$\begin{aligned} |\psi_1\rangle &= |0^n\rangle |0^n\rangle \\ |\psi_2\rangle &= \frac{1}{\sqrt{N}} \sum_{k \in \mathbb{Z}_N} |k\rangle |0^n\rangle \\ |\psi_3\rangle &= \frac{1}{\sqrt{N}} \sum_{k \in \mathbb{Z}_N} |k\rangle |f(k)\rangle. \end{aligned}$$

Measuring the second register collapses it to some state $|f(s)\rangle$, which in turn collapses the first register to an equal superposition of all states consistent with the collapse of the second register, i.e., to all states $|\ell\rangle$ with $f(\ell) = f(s)$. By the r-periodicity of f, each $|\ell\rangle$ is a shift by s from some integer multiple of r, i.e., for all ℓ, $\ell = s + kr$, where $k \in \mathbb{Z}_t$ for some positive integer t that we now describe.

- If $s < N - r\lfloor N/r \rfloor$, then the largest choice of k (i.e., t) is $\lfloor N/r \rfloor + 1$.
- If, however, $s \geq N - r\lfloor N/r \rfloor$, then the largest choice of k is $\lfloor N/r \rfloor$.

Both of these claims are evident from the following figure.



In summary, then,

$$t = \begin{cases} \lfloor \frac{N}{r} \rfloor + 1 & \text{if } s < N - r\lfloor \frac{N}{r} \rfloor \\ \lfloor \frac{N}{r} \rfloor & \text{otherwise.} \end{cases}$$

Consequently,

$$|\psi_4\rangle = \frac{1}{\sqrt{t}} \sum_{k \in \mathbb{Z}_t} |s + kr\rangle |f(s)\rangle.$$

We recognize the first register as an r-periodic state with shift s. Therefore, applying the QFT over $\mathbb{Z}_N$ gives

$$\begin{aligned} |\psi_5\rangle &= \left(\frac{1}{\sqrt{tN}} \sum_{k \in \mathbb{Z}_t} \sum_{\ell \in \mathbb{Z}_N} e^{2\pi i (s+kr)\ell/N} |\ell\rangle \right) \otimes |f(s)\rangle \\ &= \left(\frac{1}{\sqrt{tN}} \sum_{\ell \in \mathbb{Z}_N} e^{2\pi i \ell s/N} \sum_{k \in \mathbb{Z}_t} e^{2\pi i k r \ell/N} |\ell\rangle \right) \otimes |f(s)\rangle. \end{aligned}$$

Here, the sum over k is a geometric sum with common ratio $e^{2\pi i r\ell/N}$. For all $\ell \in \mathbb{Z}_N$, this ratio is never one because $r \nmid N$ and $\ell < N$. Therefore, it holds that

$$\begin{aligned} \sum_{k \in \mathbb{Z}_t} e^{2\pi i k r \ell/N} &= \frac{1 - e^{2\pi i r t \ell/N}}{1 - e^{2\pi i r \ell/N}} \\ &= \frac{e^{\pi i r t \ell/N}}{e^{\pi i r \ell/N}} \cdot \frac{e^{-\pi i r t \ell/N} - e^{\pi i r t \ell/N}}{e^{-\pi i r t \ell/N} - e^{\pi i r t \ell/N}} \\ &= \frac{e^{\pi i r t \ell/N}}{e^{\pi i r \ell/N}} \cdot \frac{e^{\pi i r t \ell/N} - e^{-\pi i r t \ell/N}}{e^{\pi i r t \ell/N} - e^{-\pi i r t \ell/N}}. \end{aligned}$$

By

$$\sin x = \frac{e^{ix} - e^{-ix}}{2i},$$

it follows that

$$\sum_{k\in\mathbb{Z}_t} e^{2\pi ikr\ell/N} = \frac{e^{\pi irt\ell/N}}{e^{\pi ir\ell/N}} \cdot \frac{\sin[\pi rt\ell/N]}{\sin[\pi r\ell/N]}.$$

Hence,

$$|\psi_5\rangle = \left(\frac{1}{\sqrt{tN}} \sum_{\ell\in\mathbb{Z}_N} \frac{e^{\pi i\ell(2s+rt)/N}}{e^{\pi ir\ell/N}} \cdot \frac{\sin[\pi rt\ell/N]}{\sin[\pi r\ell/N]} |\ell\rangle\right) \otimes |f(s)\rangle,$$

as desired. ■

Given this result, it is plain that if we measure $|\psi_5\rangle$ in the computational basis, then we expect the distribution of measurement outcomes to be strongly peaked around those $\ell \in \mathbb{Z}_N$ that are close to integer multiples of N/r. Indeed, this is true.

Claim 17.4. *For all $j \in \mathbb{Z}_r$, let $b_j := \lfloor jN/r \rceil$. Then, for all $j \in \mathbb{Z}_r$,*

$$\Pr[\text{measure } |b_j\rangle] \geq \frac{1}{r}\left(1 - \frac{\pi^2}{12} - \frac{1}{420}\right) \approx \frac{0.175}{r}.$$

Proof.* By Claim 17.3,

$$\Pr[\text{measure } |b_j\rangle] = \frac{1}{tN} \cdot \frac{\sin^2[\pi rtb_j/N]}{\sin^2[\pi rb_j/N]}.$$

Writing $b_j = \lfloor jN/r \rceil = jN/r + \epsilon$, where $|\epsilon| \leq 1/2$, gives

$$\begin{aligned}
\Pr[\text{measure } |b_j\rangle] &= \frac{1}{tN} \cdot \frac{\sin^2[\pi rt(jN/r+\epsilon)/N]}{\sin^2[\pi r(jN/r+\epsilon)/N]} \\
&= \frac{1}{tN} \cdot \frac{\sin^2[\pi tj + \pi rt\epsilon/N]}{\sin^2[\pi j + \pi r\epsilon/N]} \\
&= \frac{1}{tN} \cdot \frac{\sin^2[\pi rt\epsilon/N]}{\sin^2[\pi r\epsilon/N]},
\end{aligned}$$

where in the last line we've used the fact that $\sin(\pi k + x) = \sin(x)$ for all $x \in \mathbb{R}$ and all $k \in \mathbb{Z}$. Now, using the Taylor expansion of $\sin x$, it is easy to reason that for all $x \in \mathbb{R}$,

$$x^2 - \frac{1}{3}x^4 \leq \sin^2 x \leq x^2.$$

Therefore,

$$\begin{aligned}
\Pr[\text{measure } |b_j\rangle] &\geq \frac{1}{tN} \cdot \frac{(\pi rt\epsilon/N)^2 - \frac{1}{3}(\pi rt\epsilon/N)^4}{(\pi r\epsilon/N)^2} \\
&= \frac{t^2 - \frac{1}{3}(\pi r\epsilon/N)^2 t^4}{tN}.
\end{aligned}$$

Since $|\epsilon| \leq 1/2$, $\epsilon^2 \leq 1/4$, so

$$\Pr\left[\text{measure } |b_j\rangle\right] \geq \frac{t^2 - \frac{1}{12}(\pi r/N)^2 t^4}{tN}.$$

Finally, since $t = N/r + \delta$ for some δ such that $|\delta| < 1$, it holds that

$$\begin{aligned}
\Pr\left[\text{measure } |b_j\rangle\right] &\geq \frac{(N/r+\delta)^2 - \frac{1}{12}(\pi r/N)^2 (N/r+\delta)^4}{(N/r+\delta)N} \\
&\geq \frac{(N/r-1)^2 - \frac{1}{12}(\pi r/N)^2 (N/r-1)^4}{(N/r+1)N} \\
&= \frac{(N/r-1)^2 - \frac{1}{12}(\pi r/N)^2 (N/r-1)^4}{N^2/r[1+r/N]} \\
&= \frac{(N/r-1)^2 - \frac{1}{12}(\pi r/N)^2 (N/r-1)^4}{N^2/r} \left[1 - O\left(\frac{r}{N}\right)\right],
\end{aligned}$$

where we've used the Taylor expansion of $\frac{1}{1+x}$ to get the last line. It remains to simplify this expression. Doing so, we find that

$$\Pr\left[\text{measure } |b_j\rangle\right] \geq \frac{1}{r}\left(1 - \frac{\pi^2}{12}\right) - O\left(\frac{1}{N}\right).$$

Since, for sufficiently large N, $O(1/N) < 1/420r$, it holds that

$$\Pr\left[\text{measure } |b_j\rangle\right] \geq \frac{1}{r}\left(1 - \frac{\pi^2}{12} - \frac{1}{420}\right),$$

as desired. ■

Corollary 17.5. *Let b_j be as before. Then,*

$$\Pr\left[\text{measure } |b_0\rangle \text{ or } |b_1\rangle \text{ or } \ldots \text{ or } |b_{r-1}\rangle\right] \geq 1 - \frac{\pi^2}{12} - \frac{1}{420}.$$

Proof. Since for all $i \neq j$, measuring $|b_i\rangle$ and measuring $|b_j\rangle$ are disjoint events (meaning $\Pr[\text{measure } |b_i\rangle \text{ and } |b_j\rangle] = 0$), it holds that

$$\begin{aligned}
\Pr\left[\text{measure } |b_0\rangle \text{ or } |b_1\rangle \text{ or } \ldots \text{ or } |b_{r-1}\rangle\right] &= \sum_{j \in \mathbb{Z}_r} \Pr\left[\text{measure } |b_j\rangle\right] \\
&\geq r \cdot \frac{1}{r}\left(1 - \frac{\pi^2}{12} - \frac{1}{420}\right) \\
&= 1 - \frac{\pi^2}{12} - \frac{1}{420},
\end{aligned}$$

where the second line follows from Claim 17.4. ■

17.5. Extracting the Period: The Continued Fractions Algorithm

We have now seen that in both the exact and non-exact cases, for N sufficiently large, upon measuring $|\psi_5\rangle$ in the computational basis, we obtain an integer $b_j = \lfloor jN/r \rceil$ for some positive integer $j \leq \lfloor N/r \rfloor + 1$ with a probability that is bounded below by a constant. Thus, by repeating the quantum part of the QPF algorithm polynomially many times, we are exponentially more likely than not to obtain such a b_j. The question, then, is given b_j and N, how can we recover r? The answer is to exploit certain properties of *continued fractions*.

Definition 17.1.

- Let $a_0, \ldots, a_m \in \mathbb{Z}$ with $a_1, \ldots, a_m \geq 1$. The expression

$$\phi = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots + \cfrac{1}{a_m}}}}$$

  is called the *continued fraction expansion* (abbreviated *CFE*) of the rational number ϕ.[2] We denote this more conveniently by $\phi = [a_0; a_1, \ldots, a_m]$.

- For all $k \leq m$, we call $c_k \coloneqq [a_0; a_1, \ldots, a_k]$ the *kth convergent of* ϕ.

The following exercise underscores a basic fact about continued fractions that we will not prove formally in this course.

Exercise 17.1. *Argue that if the CFE of a number ϕ terminates, then ϕ is rational.*

In fact, the converse is also true (albeit it is more difficult to prove): if ϕ is rational, then it has a finite CFE. Thus, ϕ is rational if and only if it has a finite CFE. Given this, there is a relatively simple and efficient algorithm that determines the CFE of any rational number.

Fact 17.6 (The Continued Fractions Algorithm)**.** *There exists a deterministic classical algorithm that, given a rational number $\phi = \frac{p}{q}$ as input, where p and q are at most n-bit numbers, outputs the CFE of ϕ in $O(n^3)$ time.*

[2] In fact, there are exactly two CFEs of any rational number ϕ (namely, $[a_0; a_1, \ldots, a_k]$ and $[a_0; a_1, \ldots, a_k - 1, 1]$), so calling one *the* CFE of ϕ is not exactly right, but it is close enough to right that we need not worry about it.

Rather than prove this theorem (and describe the continued fractions algorithm in generality), we will provide an example of the algorithm that demonstrates its essence.

Example 17.1. Suppose we want to find the CFE of $\phi = \frac{31}{13}$. The first step in the continued fractions algorithm is to split 31/13 into its integer and fractional part:

$$\frac{31}{13} = 2 + \frac{5}{13}.$$

Next, simply invert the fractional part:

$$\frac{31}{13} = 2 + \frac{1}{\frac{13}{5}}.$$

These two steps—split then invert—are then recursively applied to $\frac{13}{5}$, giving:

$$\begin{aligned}\frac{31}{13} &= 2 + \frac{1}{2 + \frac{1}{\frac{5}{3}}} \\ &= 2 + \frac{1}{2 + \frac{1}{1 + \frac{1}{\frac{3}{2}}}} \\ &= 2 + \frac{1}{2 + \frac{1}{1 + \frac{1}{1 + \frac{1}{2}}}}\end{aligned}$$

At this point, the algorithm halts, because the final expression is a valid continued fraction, namely, $[2; 2, 1, 1, 2]$. Note that for more general rational ϕ, the algorithm is guaranteed to halt because, by Exercise 17.1, every rational number has a finite CFE. Also note that here we needed $O(n)$ "split-then-invert" steps and $O(n^2)$ elementary arithmetic operations, for a total deterministic time complexity of $O(n^3)$.

That is the continued fractions algorithm in a nutshell. But how does it help in extracting the period r from a measurement of the state $|\psi_5\rangle$? To understand this requires the following technical fact, which is relevant in the study of rational approximations of irrational numbers.[3]

Fact 17.7. *Let $\phi > 0$ be rational, let $M > 1$ be an integer, and let $\frac{p_k}{q_k}$ be the kth convergent of ϕ. If $\frac{j}{r}$ is a positive, irreducible rational such that $r \leq M$ and*

$$\left|\phi - \frac{j}{r}\right| \leq \frac{1}{2M^2},$$

[3]For a proof of this fact, there is a great walkthrough here.

then $\frac{j}{r}$ is a convergent in the CFE of ϕ. In particular, $\frac{j}{r} = \frac{p_i}{q_i}$, where i is the largest index such that $q_k \geq M$ for all $k > i$ (i.e., q_i is the largest denominator in the CFE of ϕ whose value is less than M).

In the context of QPF, we get the following corollary.

Corollary 17.8. *Let $b_j = \lfloor jN/r \rceil$ be as before, but with the additional assumption that $\frac{j}{r}$ is irreducible. Moreover, let*

$$\phi = \frac{b_j}{N} = \frac{\lfloor jN/r \rceil}{N},$$

and let $\frac{p_k}{q_k}$ be the kth convergent of ϕ. If $r \leq M$ and $N \geq M^2$, then j/r is a convergent in the CFE of ϕ. In particular, $\frac{j}{r} = \frac{p_i}{q_i}$, where i is the largest index such that $q_k \geq M$ for all $k > i$. In this case, by using the continued fractions algorithm, we can recover r in deterministic time $O(\log^3 N)$.

Proof. We can write $\lfloor jN/r \rceil = jN/r + \epsilon$, where $|\epsilon| \leq 1/2$. Consequently,

$$\begin{aligned}\left|\phi - \frac{j}{r}\right| &= \left|\frac{j}{r} + \frac{\epsilon}{N} - \frac{j}{r}\right| \\ &= \left|\frac{\epsilon}{N}\right| \\ &\leq \frac{1}{2N} \\ &\leq \frac{1}{2M^2}.\end{aligned}$$

Thus, the result follows from Facts 17.6 and 17.7. ■

Looking closely at this corollary's statement, there is still one final caveat that needs to be addressed: how can we guarantee that j/r is irreducible? Recall that this means we require j and r to share no common factors, which is to say that we require them to be *coprime* (a.k.a. *relatively prime*). A necessary and sufficient condition for coprimality is that the greatest common divisor of j and r is one, i.e., that $\gcd(j, r) = 1$. To better understand this requires the following definition.

Definition 17.2. For all $n \in \mathbb{N}$, let

$$\varphi(n) \coloneqq \#\text{ of positive integers } k < n \text{ such that } \gcd(k, n) = 1.$$

This function is called *Euler's totient function*.

Exercise 17.2. *Prove that for all primes p, $\varphi(p) = p - 1$.*

Next lecture, we will see several important properties of the totient function. For this lecture, the key fact we need is the following inequality.

Fact 17.9. *As $m \to \infty$,*[4]

$$\min_{n \le m} \frac{\varphi(n) \log \log n}{n} > 0.$$

Therefore,

$$\frac{\varphi(n)}{n} = \Omega\left(\frac{1}{\log \log n}\right).$$

Combining this fact with the results from before, we can now conclude the proof of Theorem 17.1.

Claim 17.10. *Let b_j be as before. The probability P that we measure $|\psi_5\rangle$ to be in state $|b_j\rangle$ for any $j \in \mathbb{Z}_r$ such that $\gcd(j, r) = 1$, and hence the probability that on a single run of QPF we recover the period r, is $\Omega\left(\frac{1}{\log \log r}\right)$.*

Proof. By the definition of conditional probability and the previous results, we have

$$\begin{aligned}
P &= \sum_{j \in \mathbb{Z}_r} \Pr\left[\gcd(j, r) = 1 \mid \text{measure } |b_j\rangle\right] \cdot \Pr\left[\text{measure } |b_j\rangle\right] \\
&\ge \sum_{j \in \mathbb{Z}_r} \frac{\varphi(r)}{r} \cdot \frac{1}{r}\left(1 - \frac{\pi^2}{12} - \frac{1}{420}\right) \\
&= r \cdot \frac{\varphi(r)}{r} \cdot \frac{1}{r}\left(1 - \frac{\pi^2}{12} - \frac{1}{420}\right) \\
&= \frac{\varphi(r)}{r}\left(1 - \frac{\pi^2}{12} - \frac{1}{420}\right) \\
&= \Omega\left(\frac{1}{\log \log r}\right),
\end{aligned}$$

as desired. ∎

[4] Technically, this minimum should be an infimum (i.e., the greatest lower bound), so that the limit is the *limit inferior*, but we need not concern ourselves with that mathematical subtlety. Interestingly, if you use the natural logarithm, then this limit inferior approaches e^γ, where

$$\gamma := \lim_{n \to \infty}\left(-\log n + \sum_{k=1}^{n} \frac{1}{n}\right) \approx 0.577$$

is the *Euler–Mascheroni constant*.

We have therefore proven the main theorem, Theorem 17.1, which establishes the correctness of the QPF algorithm in both the exact and non-exact cases. In the next section, we discuss how QPF relates to QPE.

17.6. QPF is QPE in Disguise*

We have said many times that QPE is the essential algorithm that underlies many important algorithms, such as Shor's algorithm. In the same breath, we have also said that QPF is the essential algorithm that underlies many important algorithms, again like Shor's. Here, we will show that QPE is really the deeper algorithm, as QPF, while in many respects "cleaner", is just QPE in disguise.

Definition 17.3. Let $f : \mathbb{Z} \to \mathbb{Z}$ be a total, r-periodic function for which $f(0), f(1), \ldots, f(r-1)$ are all distinct. The *shift operator of* f, denoted by $\mathcal{S}_f$, is the unitary matrix $\mathcal{S}_f$ such that for all $x \in \mathbb{Z}_r$,

$$\mathcal{S}_f|f(x)\rangle = |f(x+1)\rangle,$$

where addition is implicitly modulo r.

The unitarity of $\mathcal{S}_f$ is not difficult to establish, and it relies on the fact that $f(0), f(1), \ldots, f(r-1)$ are all distinct values (so that $|f(0)\rangle, \ldots, |f(r-1)\rangle$ is a basis of $\mathbb{C}^r$). Indeed, one can write $\mathcal{S}_f = \sum_{x\in\mathbb{Z}_r} |f(x+1)\rangle\langle f(x)|$, which makes it easy to see that $\mathcal{S}_f^{-1} = \mathcal{S}_f^\dagger$.

We now state an important fact about the shift operator.

Claim 17.11. *For all $\ell \in \mathbb{Z}_r$, define*

$$|\tilde{f}(\ell)\rangle \coloneqq \frac{1}{\sqrt{r}} \sum_{x\in\mathbb{Z}_r} e^{-2\pi i\ell x/r}|f(x)\rangle.$$

Then, $|\tilde{f}(\ell)\rangle$ is an eigenvector of $\mathcal{S}_f$ with eigenvalue $e^{2\pi i\ell/r}$.

Proof. By the definition of $\mathcal{S}_f$,

$$\begin{aligned}
\mathcal{S}_f|\tilde{f}(\ell)\rangle &= \frac{1}{\sqrt{r}} \sum_{x\in\mathbb{Z}_r} e^{-2\pi i\ell x/r}|f(x+1)\rangle \\
&= \frac{1}{\sqrt{r}} \sum_{x\in\mathbb{Z}_r} e^{2\pi i\ell/r - 2\pi i\ell(x+1)/r}|f(x+1)\rangle \\
&= e^{2\pi i\ell/r}\frac{1}{\sqrt{r}} \sum_{x\in\mathbb{Z}_r} e^{-2\pi i\ell(x+1)/r}|f(x+1)\rangle \\
&= e^{2\pi i\ell/r}|\tilde{f}(\ell)\rangle,
\end{aligned}$$

as desired. ■

Importantly, using the geometric sum formula, it is also straightforward to establish the following fact, which we will leave as an exercise to the reader.

Fact 17.12. *For all $x \in \mathbb{Z}_r$,*

$$|f(x)\rangle = \frac{1}{\sqrt{r}} \sum_{\ell \in \mathbb{Z}_r} e^{2\pi i \ell x / r} |\tilde{f}(\ell)\rangle.$$

Now recall the state $|\psi_3\rangle$ in the QPF algorithm,

$$|\psi_3\rangle = \frac{1}{\sqrt{N}} \sum_{k \in \mathbb{Z}_N} |k\rangle |f(k)\rangle.$$

By the previous fact, we can write this as follows

$$|\psi_3\rangle = \frac{1}{\sqrt{rN}} \sum_{k \in \mathbb{Z}_N} \sum_{\ell \in \mathbb{Z}_r} e^{2\pi i \ell k / r} |k\rangle |\tilde{f}(\ell)\rangle.$$

Note, however, that this is exactly the state you get in QPE after applying $\Lambda^n(\mathcal{S}_f)$ to the state $\frac{1}{\sqrt{r}} \sum_{\ell \in \mathbb{Z}_r} H^{\otimes n} |0^n\rangle \otimes |\tilde{f}(\ell)\rangle$, where the eigenvalue phase is $\theta = \ell / r$. Thus, continuing with the QPE algorithm, one will recover a good approximation to $N\theta$, which in turn gives a good approximation to an integer multiple of N/r. Using continued fractions, one can then find r. Consequently, we see that QPE with $\mathcal{S}_f$ is period finding, and in this way underlies the QPF algorithm.

Incidentally, we note that the effect of the oracle $\mathcal{O}_f$ in QPF can be simulated by $\Lambda^n(\mathcal{S}_f)$. To see this, fix $a \in \mathbb{Z}_N$ and let s be such that $f(s) = a$. Then, for all $k \in \mathbb{Z}_N$ with binary expansion $k = \sum_{i=0}^{\lceil \log(N-1) \rceil} x_i 2^{n-i-1}$,

$$\begin{aligned}
\Lambda^n(\mathcal{S}_f)|k\rangle|a\rangle &= \Lambda^n(\mathcal{S}_f)|k\rangle|f(s)\rangle \\
&= |k\rangle \mathcal{S}_f^{x_0 2^{n-1}} \mathcal{S}_f^{x_1 2^{n-2}} \cdots \mathcal{S}_f^{x_{n-1} 2^0} |f(s)\rangle \\
&= |k\rangle \mathcal{S}_f^{x_0 2^{n-1}} \mathcal{S}_f^{x_1 2^{n-2}} \cdots \mathcal{S}_f^{x_{n-2} 2^1} |f(x_{n-1} 2^0 + s)\rangle \\
&= |k\rangle \mathcal{S}_f^{x_0 2^{n-1}} \mathcal{S}_f^{x_1 2^{n-2}} \cdots \mathcal{S}_f^{x_{n-3} 2^2} |f(x_{n-2} 2^1 + x_{n-1} 2^0 + s)\rangle \\
&\;\;\vdots \\
&= |k\rangle |f(x_0 2^{n-1} + x_1 2^{n-2} + \cdots + x_{n-1} 2^0 + s)\rangle \\
&= |k\rangle |f(k+s)\rangle.
\end{aligned}$$

Therefore, when we measure the second register, we will obtain some state $|f(s')\rangle$, and the first register will collapse into a superposition of states that are consistent with the collapse of the second register. From here, the QPF algorithm proceeds in the same way, in both the exact and non-exact cases. What this shows is that the controlled operation $\Lambda^n(\mathcal{S}_f)$ is just as good in QPF as the canonical quantum oracle $\mathcal{O}_f$.

17.7. A Number Theoretic Digression*

While this section is not needed for anything we will discuss in this class, there is an interesting, somewhat relevant number theoretic fact that pertains to our studying the probability that some $j \in \mathbb{Z}_r$ is coprime to r. Instead of this question, you might ask, what is the probability that two numbers, k_1 and k_2, that are drawn uniformly from $\mathbb{Z}_N$ are coprime? Interestingly, this probability is bounded below by a (very cool) constant. This derives from a beautiful fact, an informal[5] proof of which employs Euler's product formula for the Riemann zeta function:

$$\zeta(s) \coloneqq \sum_{n \geq 1} \frac{1}{n^s} = \prod_{p \text{ prime}} \frac{1}{1 - p^{-s}}.$$

Claim 17.13. *If $k_1, k_2 \sim \mathbb{Z}_N$ uniformly, then the probability that $\gcd(k_1, k_2) = 1$ is at least $1/\zeta(2) = 6/\pi^2$.*

Proof (Informal). Let p be prime. Then, every pth integer is divisible by p, so, heuristically speaking, the probability that $k_1 \sim \mathbb{Z}_N$ is divisible by p is $1/p$. Likewise for $k_2 \sim \mathbb{Z}_N$. Therefore, the probability that both k_1 and k_2 are divisible by p is $1/p^2$, so the probability that both are *not* divisible by p is

$$1 - \frac{1}{p^2} = 1 - p^{-2}.$$

[5] Using some tools from analytic number theory (in particular the so-called *Möbius function*) one can give an elegant, formal proof of this fact by deriving the following asymptotic formula:

$$\sum_{\substack{k_1, k_2 \in \mathbb{Z}_N \\ \gcd(k_1, k_2) = 1}} 1 = N^2 \frac{6}{\pi^2} + O(N \log N).$$

Given this, it's plain that

$$\Pr_{k_1, k_2 \sim \mathbb{Z}_N} [\gcd(k_1, k_2) = 1] = \frac{N^2 \frac{6}{\pi^2} + O(N \log N)}{N^2} = \frac{6}{\pi^2} + O\left(\frac{\log N}{N}\right),$$

as desired.

Consequently, the probability that k_1 and k_2 are coprime (meaning that they share no common prime factors) is then

$$\Pr_{k_1,k_2\sim\mathbb{Z}_N}[\gcd(k_1,k_2)=1] = \prod_{\substack{p \text{ prime}\\ p<N}} 1-p^{-2} = \left(\prod_{\substack{p \text{ prime}\\ p<N}} \frac{1}{1-p^{-2}}\right)^{-1}.$$

Therefore,

$$\Pr_{k_1,k_2\sim\mathbb{Z}_N}[\gcd(k_1,k_2)=1] \geq \frac{1}{\zeta(2)},$$

where $\zeta(2) = \sum_{n\geq 1} n^{-2}$ is the famous *Basel sum*, which equals $\pi^2/6$. ■

This generalizes in the obvious way to many pairs of integers.

Fact 17.14. *If $k_1, k_2, \ldots, k_n \sim \mathbb{Z}_N$ uniformly, then the probability that $\gcd(k_i, k_j) = 1$ for all $i \neq j$ is at least $1/\zeta(n)$.*

Again, we will not need these results in this class, but they are good to know.

Lecture 18
Shor's Algorithm for Factoring Integers

Discussion 18.1. Discuss with your group what you took away from last time.

Last time, we discussed the quantum period finding (QPF) algorithm, which allows us to find the period of a general r-periodic function $f : \mathbb{Z} \to \mathbb{Z}$, provided we have an upper bound on r. The key quantum gate to this algorithm was of course the quantum Fourier transform. Additionally, this algorithm required classical post-processing in the form of the continued fractions algorithm to efficiently recover the period r from our measurements of the quantum circuit involved in QPF.

In this lecture, we will introduce Shor's algorithm for factoring integers, which uses the QPF algorithm as a subroutine to factor large integers efficiently. To do this, we will prove that factoring reduces to finding the period of a particular function. As we will explore in a few lectures from now, Shor's algorithm for factoring proves that many cryptosystems (e.g., RSA) are not secure against quantum adversaries.

18.1. The Integer Factorization Problem

Hopefully, the integer factorization problem is familiar.

The Integer Factorization Problem

Input: a positive integer N.

Output: a non-trivial divisor of N.

In this problem, we require the output to be a *non-trivial* divisor d of N, meaning $1 < d < N$ and $d \mid N$, for otherwise we could always output 1 or N and be done.

Recall from the lecture on computational complexity theory that factoring can also be phrased as a language in $\{0,1\}^*$, namely,

$$L_{\text{factoring}} = \{x.y \in \{0,1\}^* : x \text{ has a non-trivial divisor that is at most } y\}.$$

Classically, it is expected that $L_{\text{factoring}} \notin \mathsf{BPP}$, i.e., that there is no efficient, probabilistic classical algorithm for solving the integer factorization problem (and this is why so much of cryptography is based on factoring). That said, it is not hard to reason that $L_{\text{factoring}} \in \mathsf{NP}$. In this lecture, we will see that $L_{\text{factoring}} \in \mathsf{BQP}$, which gives strong evidence that $\mathsf{BPP} \neq \mathsf{BQP}$, i.e., that efficient quantum computers can compute a function that no efficient classical computer can. An unfortunate side-effect of this, however, is that the whole RSA cryptosystem that underlies much of the internet is now vulnerable to quantum attacks.

18.2. When is Factoring Easy?

Here, we will briefly spell out two cases when factoring an integer is easy.

Question 18.1. *If in binary the least significant digit of N is 0, what is a non-trivial divisor of N?*

Therefore, we can always assume that the input N is odd, for otherwise the integer factorization problem is trivial.

Fact 18.1 (AKS Test)**.** *$L_{primes} \in \mathsf{P}$, i.e., there is an efficient, deterministic classical algorithm for deciding if N is prime or not.*

Consequently, we can know with just polynomial time pre-processing whether or not our input N *has* non-trivial divisors at all (i.e., if the integer-factorization problem is even worth solving). Thus, we can always assume that the input N is not prime (so that the integer-factorization problem has a non-trivial solution).

Claim 18.2. *Suppose $N = M^k$ for integers $M \geq 1$ and $k \geq 2$. Then, there is an efficient, deterministic classical algorithm for finding M.*

Proof Idea.* Simply compute $\sqrt[k]{N}$ for $k = 2, 3, \ldots, \lceil \log_2(N) \rceil$. If $N = M^k$, then $\log_M(N) = k \leq \log_2(N) = k \log_2(M)$, so you will eventually recover M by doing this. Note that it is possible to compute $\sqrt[k]{N}$ in deterministic time $O(k \log N)$ using, for example, Newton's method for finding the roots of a polynomial. ■

Therefore, if N is a prime power, then we can efficiently recover the prime factor, and hence efficiently solve the integer factorization problem. More generally, this fact implies that we can always suppose that N is not a power of some smaller integer M. Altogether, the above facts will appear in the pre-processing steps of Shor's algorithm, and the key point is that these checks can all be completed in

polynomial time on a deterministic classical computer *before* running the quantum part of Shor's algorithm.

We will now turn to the remaining cases where factoring appears harder. To do this requires some number theory.

18.3. Some Requisite Number Theory

We begin with a fact that is hopefully familiar from a course on classical algorithms.

Fact 18.3. *There exists an efficient deterministic classical algorithm (e.g., the Euclidean algorithm) that computes* $\gcd(x, y)$ *for all inputs* $x, y \in \mathbb{Z}$.

Given this, we will now introduce the key mathematical object that underlies Shor's factoring algorithm.

Definition 18.1. The *multiplicative group of integers modulo* N is[1]

$$\mathbb{Z}_N^\times \coloneqq \Big(\{k \in \{1, 2, \ldots, N-1\} : \gcd(k, N) = 1\}, \times \bmod N \Big).$$

We call N the *modulus*.

Note that thanks to the Euclidean algorithm, given any integer $k < N$, it is easy to check whether or not $k \in \mathbb{Z}_N^\times$, because it is easy to check whether or not $\gcd(k, N) = 1$.

Exercise 18.1.

- *What is* $\mathbb{Z}_p^\times$ *for any prime* p*?*
- *What is* $|\mathbb{Z}_p^\times|$ *for any prime* p*?*

Definition 18.2. The cardinality of $\mathbb{Z}_N^\times$ defines *Euler's totient function*,

$$\varphi(N) \coloneqq |\mathbb{Z}_N^\times|,$$

which we saw last lecture. In words, $\varphi(N)$ equals the number of positive integers strictly less than N that are coprime to N.

By the previous exercise, you have proven that for all primes p, $\varphi(p) = p - 1$. This says that all integers less than a prime p are coprime to p, which of course makes sense by the definition of primality. The following are two more facts about φ that are incredibly important.

[1] This group is also sometimes denoted by $(\mathbb{Z}/N\mathbb{Z})^\times$.

Fact 18.4.

- *$\varphi(N)$ contains information about the prime factors of N:*

$$\varphi(N) = N \prod_{\substack{p|N \\ p \text{ prime}}} \left(1 - \frac{1}{p}\right).$$

  This is called Euler's product formula for φ.

- *For all $a \in \mathbb{Z}_N^\times$ (i.e., for all a coprime to N),*

$$a^{\varphi(N)} = 1 \pmod{N}.$$

  This is known as Euler's theorem*, which generalizes* Fermat's little theorem.[2]

Altogether, we learn that $\varphi(N)$, and hence the group $\mathbb{Z}_N^\times$, contains information about the prime factors of N. It is ultimately this connection that Shor's algorithm exploits to factor N. Note that by Euler's theorem and the well-ordering principle, there must be a *smallest* positive integer r such that $a^r = 1 \pmod{N}$.

Definition 18.3. For $a \in \mathbb{Z}_N^\times$, the smallest integer $r > 0$ such that $a^r = 1 \pmod{N}$ is called *the order of a in $\mathbb{Z}_N^\times$*, *the order of a modulo N*, or, if N is contextually clear, just *the order of a*.

Exercise 18.2. *Prove that for all $a \in \mathbb{Z}_N^\times$, there exists $b \in \mathbb{Z}_N^\times$ such that $ab = 1 \pmod{N}$. We call b the* modular multiplicative inverse of a *and write it as $b = a^{-1}$.*[3]

We will now see how to frame the integer factorization problem as finding the order of some $a \in \mathbb{Z}_N^\times$. This, in turn, is easily framed as finding the period of a certain function, which we will discuss in the next section.

Fact 18.5 (HW5)**.** *Let $a \in \mathbb{Z}_N^\times$ be such that the following two conditions hold:*

(i) the order r of a is even,

(ii) $a^{r/2} \neq -1 \pmod{N}$.[4]

[2] Indeed, taking N to be prime in Euler's theorem gives Fermat's little theorem. Interestingly, Euler's theorem also goes the other way, so that if $a^{\varphi(N)} = 1 \pmod{N}$, then $\gcd(a, N) = 1$. Incidentally, the correctness of the RSA public-key cryptosystem relies on this theorem.

[3] Of course, this is needed to justify our unsubstantiated claim that $\mathbb{Z}_N^\times$ is a group.

[4] In general, if $x^2 = 1 \pmod{N}$, then we call x a *non-trivial square-root of 1 modulo N*.

Then, $\gcd(a^{r/2}+1, N)$ *is a non-trivial divisor of* N*.*[5]

At first, these two conditions might seem a bit ad hoc. However, thanks to the following incredible fact, it is actually quite likely that a uniformly drawn $a \sim \mathbb{Z}_N$ satisfies the above two conditions, and so this fact is of general use.[6]

Fact 18.6. *For odd* N *that is not a prime power, over half of all* $a \in \mathbb{Z}_N$ *satisfy one of the following two conditions:*

(i) $\gcd(a, N) > 1$,

(ii) $\gcd(a, N) = 1$ *(i.e.,* $a \in \mathbb{Z}_N^\times$*), the order* r *of* a *is even, and* $a^{r/2} \neq -1 \pmod N$.

Corollary 18.7 (Reduction of Factoring to Order Finding)**.** *If* $a \sim \mathbb{Z}_N$ *uniformly with order* r*, then with probability at least* $1/2$*, either* $\gcd(a, N)$ *or* $\gcd(a^{r/2}+1, N)$ *is a non-trivial divisor of* N*. Therefore, an efficient (classical or quantum) algorithm for computing the order* r *of any* $a \in \mathbb{Z}_N^\times$ *implies an efficient (classical or quantum) algorithm for factoring* N*.*

Shor's algorithm is based on this reduction of factoring to order finding, as we will now see.

18.4. Shor's Algorithm for Factoring Integers

In this section, we will present Shor's algorithm. To do so, however, we first recast the order finding problem discussed in the previous corollary to the problem of finding the period of a particular function.

Definition 18.4. For all $N \in \mathbb{N}$ and all $a \in \mathbb{Z}_N^\times$, let

$$\begin{aligned} f_a : \mathbb{Z} &\to \mathbb{Z}_N^\times \\ x &\mapsto a^x \bmod N. \end{aligned}$$

It is not difficult to see that f_a is periodic with a very specific period, which you will now prove.

[5] In fact, a more general statement is true that is related to something called *Euclid's lemma*, which you will also prove on HW5.

[6] I recommend this writeup by Keith Conrad for a proof of the following fact, which is relevant in the context of the *Miller–Rabin primality test*.

Exercise 18.3. *Prove that f_a is r-periodic, where r is the order of a.*

We now state an important fact about the complexity of computing f_a.

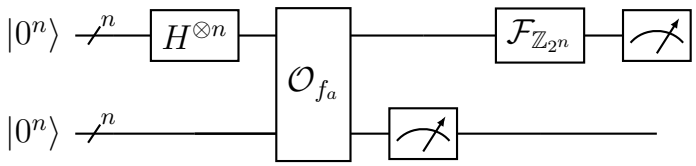
Fact 18.8. *For all positive integers N and all $a \in \mathbb{Z}_N^\times$, f_a is computable by an efficient deterministic classical computer (by implementing, for example, the modular exponentiation by repeated squaring algorithm). Hence, the quantum oracle $\mathcal{O}_{f_a} : |x\rangle|0\rangle \mapsto |x\rangle|f_a(x)\rangle$ can be implemented efficiently on a quantum computer.*

We are now ready to present Shor's algorithm for the integer factorization problem.

Shor's Algorithm for Factoring Integers

1. Check if N is even. If so, return 2. Else, continue.
2. Check if $N = M^k$ for integers $M \geq 1$ and $k \geq 2$ (using, for example, Fact 18.2). If so, replace $N \leftarrow M$ and continue. Else, continue with the original N.
3. Check if N is prime (using, for example, the AKS test, i.e., Fact 18.1). If so, return N. Else, continue.
4. Generate $a \sim \mathbb{Z}_N$ uniformly (by generating $O(\log N)$ random bits).
5. Check if $\gcd(a, N) > 1$ (using, for example, the Euclidean algorithm). If so, return $\gcd(a, N)$. Else, continue.
6. Run the quantum period finding (QPF) algorithm on $f_a(x) = a^x \bmod N$ using n qubits, where $2^n \geq N^2$, to recover the order r of a.
7. Check if r is even and if $a^{r/2} \neq -1 \pmod N$. If so, return $\gcd(a^{r/2} + 1, N)$. Else, go back to step 4.

As a circuit, then, the quantum part of Shor's factoring algorithm is simply QPF for the particular function f_a:



Using the results we have already stated in this lecture, we will now prove the following theorem.

Theorem 18.9. *There exists an efficient quantum computer (namely, that specified by Shor's factoring algorithm) that finds a non-trivial divisor of N (and hence solves the integer factorization problem) with probability $\Omega(\frac{1}{\log\log N})$. Therefore, there exists an efficient quantum computer that finds a non-trivial divisor of N with probability at least $1 - 2^{-O(\mathrm{poly}(\log N))}$. Consequently, $L_{factoring} \in \mathsf{BQP}$.*

Proof. Every step in Shor's algorithm can be done efficiently on a classical or quantum computer, therefore all of Shor's algorithm can be implemented on an efficient quantum computer.

If N is even or a prime power, then Shor's algorithm outputs a non-trivial divisor of N with probability 1 (thanks to steps 1–3).

If, however, N is odd and not a prime power, then, after generating $a \sim \mathbb{Z}_N$ uniformly, it follows from Corollary 18.7 that with probability at least $1/2$, either $\gcd(a, N)$ or $\gcd(a^{r/2} + 1, N)$ is a non-trivial divisor of N, where r is the order of a. Since $r \leq \varphi(N) < N$, we will find r on a single run of the QPF algorithm with probability

$$\Omega\left(\frac{1}{\log\log r}\right) = \Omega\left(\frac{1}{\log\log N}\right).$$

Therefore, if r' is the output of the QPF subroutine (step 6), then with probability at least

$$\frac{1}{2} \cdot \Omega\left(\frac{1}{\log\log N}\right) = \Omega\left(\frac{1}{\log\log N}\right),$$

either $\gcd(a, N)$ or $\gcd(a^{r'/2} + 1, N)$ is a non-trivial divisor of N. In consequence, Shor's algorithm outputs a non-trivial divisor of N with probability at least $\Omega\left(\frac{1}{\log\log N}\right)$, as desired. ■

In a later lecture, we will discuss the implications of this result for contemporary cryptography, and how we are currently in need of new "post-quantum" cryptosystems that can withstand quantum attacks. We also note that there are other known efficient quantum factoring algorithms, such as one by Oded Regev that is based on lattices, and which, at least in some respects, performs better than Shor's algorithm as it reduces the depth of the quantum circuit (but requires more qubits).[7]

[7] His paper is here if you are interested.

Lecture 19
Shor's Algorithm for Discrete Logarithms

Discussion 19.1. Discuss with your group what you took away from last time.

Last time, we discussed Shor's algorithm for factoring integers, which proves that $L_{\text{factoring}} \in \mathsf{BQP}$. Since we expect $L_{\text{factoring}} \notin \mathsf{BPP}$, Shor's algorithm gives strong evidence that efficient quantum computers can perform a computational task that no efficient classical computer can, i.e., that $\mathsf{BPP} \neq \mathsf{BQP}$. Recall that the key to Shor's algorithm was to reduce factoring a number N to finding the period of the univariate function $f_a(x) = a^x \bmod N$, where $a \in \mathbb{Z}_N^\times$.

In this lecture, we will see another example of a computational task that an efficient quantum computer can do, but that (we suspect) no efficient classical computer can do. This is the *discrete logarithm problem*, which, like factoring, underlies many important cryptographic primitives and which, also like factoring, reduces to finding the period of a particular function. That said, the discrete log period is "higher-dimensional", so we will need some new techniques.

19.1. Diffie–Hellman Key Exchange

To motivate the importance of the discrete logarithm problem, we will first consider the following seemingly impossible task, which underscores the magic of public-key cryptography:

1. You and I are in a room full of people. We have never talked before.
2. I am going to shout something, so that everyone hears me.
3. You are going to shout something, so that everyone hears you.
4. Despite our shouting and the many eavesdroppers in the room, you and I now know something that no one else in the room knows.

Believe it or not, this is, in fact, possible, thanks to a protocol discovered by cryptographers Whitfield Diffie and Martin Hellman. It goes as follows (where now you and I are replaced by Alice and Bob).

Diffie–Hellman Key Exchange

1. Alice and Bob are in a room full of people. They have never talked before.
2. Alice announces the following numbers to everyone in the room:
   - an integer N,
   - an integer $g \in \mathbb{Z}_N^\times$,
   - an integer $A = g^a \bmod N$, where a is an integer that Alice keeps to herself.
3. Bob announces $B = g^b \bmod N$ to everyone in the room, where b is an integer that Bob keeps to himself.
4. Alice and Bob share the secret $s = g^{ab} \bmod N$, because Alice can efficiently compute

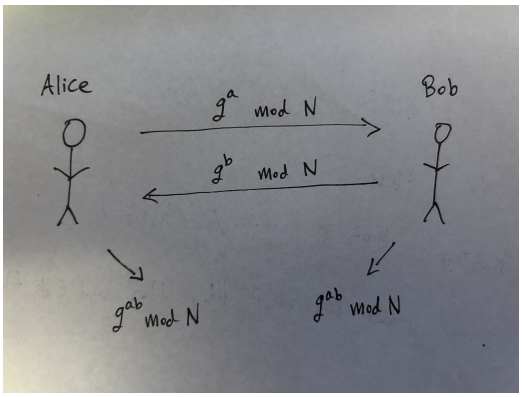
   $$B^a \bmod N = g^{ab} \bmod N = s$$

   and Bob can efficiently compute

   $$A^b \bmod N = g^{ab} \bmod N = s,$$

   however no one else in the room knows s.

In a picture, Diffie–Hellman key exchange is as follows:



Of course, this final conclusion—that on one else in the room knows s (or, more precisely, that no one else in the room can efficiently compute s given N, g, A, and B)—is ultimately a *belief*, in the same way that "factoring large numbers is classically hard" is a belief. Here, though, the belief is not based on the difficultly of factoring, but rather based on the difficulty of computing *discrete logarithms*.

Importantly, discrete logarithms appear in all sorts of cryptographic primitives, such as in key exchange à la Diffie–Hellman, which itself does not provide a means to send private messages. Nevertheless, the discrete log problem underlies many important cryptosystems, such as the ElGamal Public-Key Cryptosystem, which does allow you to send private messages and which is used in GNU Privacy Guard software and recent versions of PGP ("Pretty Good Privacy").

In this lecture, we will see the discrete log problem, and we will see how quantum computers can solve it, even in the hardest case. Consequently, the Diffie–Hellman key exchange protocol (and in fact all cryptosystems that get their security guarantees from the discrete log problem) are not secure against quantum adversaries.

19.2. The Discrete Logarithm Problem

We begin with a few definitions.

Definition 19.1.

- Let $N \in \mathbb{N}$ be a modulus (soon to be prime, but not yet), let $g \in \mathbb{Z}_N^\times$, and let r_g be its order.[1] The *group generated by* g is the set

$$\langle g \rangle \coloneqq \left\{ g^0, g^1, g^2, \ldots, g^{r_g - 1} \right\} \subseteq \mathbb{Z}_N^\times,$$

  together with multiplication modulo N.

- We call g *a generator of* $\mathbb{Z}_N^\times$ (a.k.a. *a primitive root modulo* N) iff $\langle g \rangle = \mathbb{Z}_N^\times$. In this case, $\mathbb{Z}_N^\times$ is called a *cyclic group*.

Importantly, due to a beautiful result of Gauss, only for certain N is $\mathbb{Z}_N^\times$ cyclic.

Theorem 19.1 (Gauss)**.** *The group $\mathbb{Z}_N^\times$ is cyclic if and only if N is $1, 2, 4, p^k$, or $2p^k$, where p is an odd prime and $k > 0$.*

This result will be important later. For now, let us define what a "discrete log" is, as well as the associated "discrete log problem".

[1] Recall that the *order r_g of g* is the smallest positive integer such that $g^{r_g} = 1 \pmod{N}$.

Definition 19.2. Given $h \in \langle g \rangle$, the unique integer $\alpha \in \mathbb{Z}_{r_g}$ such that $h = g^\alpha$ is called *the discrete logarithm of h with respect to g* (or just the *discrete log of h* for short, especially when g is contextually clear). The discrete log of h with respect to g is often denoted by $\alpha = \log_g h$.

The discrete log problem over $\mathbb{Z}_N^\times$ is then as follows.[2]

The Discrete Logarithm Problem

Input: a modulus $N \in \mathbb{N}$, $g \in \mathbb{Z}_N^\times$, and $h \in \langle g \rangle$.

Output: $\log_g h$.

Like factoring, there are certain cases where the discrete log problem is easy classically. We will briefly discuss these to gain some intuition for when we expect the discrete log problem to be hard, at least classically.

19.3. When is Discrete Log Easy?

In the discrete log problem, we get to choose N and g. Here, we will see that a poor choice of these makes the problem easy.

Exercise 19.1. *If a modulus N satisfies*

$$\varphi(N) = |\mathbb{Z}_N^\times| = O(\log N),$$

argue that there is a classical algorithm that finds $\log_g h$ in $O(\log N)$ time.

Therefore, if we want a classically hard instance of discrete log, then we better choose N such that $\varphi(N) \gg \log N$.

Question 19.1. *For what (large) integers N can we be certain that $\varphi(N) \gg \log N$?*

In addition to the above, there is another way that a poor choice of modulus can make the discrete log problem easy classically.

Theorem 19.2 (Pohlig–Hellman)**.** *If $N = \prod_i p_i^{e_i}$ is the prime factorization of N, then there exists a deterministic classical algorithm (called the* Pohlig–Hellman algorithm*) that finds $\log_g h$ in time*

$$O\left(\sum_i e_i\big(\log N + \sqrt{p_i}\big)\right).$$

[2]Incidentally, the discrete log problem can be defined over other groups.

Therefore, if N is a product of "small" primes (e.g., $p_i = O(\log N)$ for all i), then there is an efficient classical algorithm for the discrete log problem.

Given these results, we see that a universal fix to make the discrete log problem hard (at least classically) is to choose N to be a large prime p. Indeed, in this case:

- $\varphi(N) = \varphi(p) = p - 1 \gg \log p$, so we resolve the issue raised in Exercise 19.1,
- the runtime of the Pohlig–Hellman algorithm is $O(N) = O(p)$, so it is not guaranteed to be efficient.

Indeed, taking N to be a large prime is what is usually done in discrete log-based cryptographic schemes. In addition, if N is prime, then it follows from Gauss's[3] Theorem 19.1 that $\mathbb{Z}_N^\times$ is cyclic. In this case, it is typical to take g to be a generator of $\mathbb{Z}_N^\times$, so that h can be any one of a wide range of values (namely, $\varphi(N) = N - 1$ many values).

The Discrete Logarithm Problem (Hard Version)

Input: a (large) prime p, a generator g of $\mathbb{Z}_p^\times$, and $h \in \langle g \rangle = \mathbb{Z}_p^\times$.

Output: $\log_g h$.

That said, in this lecture we will proceed in generality, and see that there is an efficient quantum computer that solves the discrete log problem in every case.

19.4. The Discrete Log Problem as Lattice Period Finding

Like factoring, the discrete log problem can be cast as a period finding problem for a particular choice of function. Given the QPF algorithm, perhaps, then, it is not surprising that a quantum computer can find discrete logs efficiently. That said, the exact notion of periodicity here is more complicated than the univariate periodicity to which QPF applies. Indeed, for discrete log, the periodicity is "higher dimensional". For this reason, we will now embark on a brief aside into the theory of *lattices*, which are mathematical objects that capture high dimensional periodicity.

Definition 19.3.

- Let $\vec{b}_1, \ldots, \vec{b}_n$ be n linearly independent vectors in $\mathbb{R}^m$, where $m \geq n$. A *lattice* $\mathcal{L}$ is all integer linear combinations of these vectors,

$$\mathcal{L} = \left\{ z_1 \vec{b}_1 + \cdots + z_n \vec{b}_n : z_1, \ldots, z_n \in \mathbb{Z} \right\}.$$

[3] That's not a typo, the possessive form of Gauss is Gauss's not Gauss'.

- Each $\vec{v} \in \mathcal{L}$ is called a *lattice vector*.
- A function $f : \mathbb{Z}^n \to \mathbb{Z}$ is *$\mathcal{L}$-periodic* iff for all $\vec{x} \in \mathbb{Z}^n$ and all $\vec{v} \in \mathcal{L}$,

$$f(\vec{x} + \vec{v}) = f(\vec{x}).$$

Exercise 19.2. *Argue that a univariate function $f : \mathbb{Z} \to \mathbb{Z}$ is r-periodic iff it is $\mathcal{L}$-periodic, where $\mathcal{L} = \{zr : z \in \mathbb{Z}\}$.*

Thus, the lattice-based periodicity introduced above is a natural, multivariate generalization of the univariate function periodicity from before. Given this, we will now see how to cast the discrete logarithm problem as finding a vector in some lattice $\mathcal{L}$ that characterizes a particular $\mathcal{L}$-periodic function.

Claim 19.3. *Let N be a modulus, $g \in \mathbb{Z}_N^\times$, and $h \in \langle g \rangle$. Moreover, let $\alpha = \log_g h$ be the discrete log of h with respect to g and let*

$$\mathcal{L}_\alpha \coloneqq \left\{ z\vec{b}_\alpha : z \in \mathbb{Z} \right\} \subset \mathbb{R}^2$$

be a lattice, where $\vec{b}_\alpha = (\alpha, 1)$. Then, the bivariate function

$$\begin{aligned} f_{g,h} &: \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z} \\ &: (x, y) \mapsto g^x h^{-y} \bmod N \end{aligned}$$

is $\mathcal{L}_\alpha$-periodic.

Proof. For all $\vec{x} = (x, y) \in \mathbb{Z}^2$ and all lattice vectors $\vec{v} = (\alpha z, z) \in \mathcal{L}_\alpha$, it holds that

$$\begin{aligned} f_{g,h}(\vec{x} + \vec{v}) &= f_{g,h}(x + \alpha z, y + z) \\ &= g^{x+\alpha z} h^{-y-z} \bmod N \\ &= g^{x+\alpha z} g^{-\alpha y - \alpha z} \bmod N \\ &= g^x g^{-\alpha y} \underbrace{g^{\alpha z - \alpha z}}_{=1} \bmod N \\ &= g^x h^{-y} \bmod N \\ &= f_{g,h}(x, y) \\ &= f_{g,h}(\vec{x}). \end{aligned}$$

Therefore, $f_{g,h}$ is $\mathcal{L}_\alpha$-periodic, as claimed. ∎

Crucially, there is an alternate characterization of the periodicity of fg, h.

Fact 19.4 (HW6)**.** *For all $\vec{x}, \vec{y} \in \mathbb{Z}^2$, $f(\vec{x}) = f(\vec{y})$ iff there exists $\vec{v} = (v_1, v_2) \in \mathbb{Z}^2$ such that $v_2\alpha = v_1 \pmod{r_g}$, where r_g is the order of g modulo N.*

To finish casting the discrete log problem as a period finding problem, we will need the following fact, which can be found in any standard text on number theory.[4]

Fact 19.5. *Let a, b, and n be integers such that $\gcd(a, n) = 1$. Then, there exists a unique integer $x \in \mathbb{Z}_n$ such that $ax = b \pmod{n}$. Moreover, there exists an efficient, deterministic, classical algorithm to find x.*[5]

In univariate period finding in which we want to find the period r of an r-periodic function $f : \mathbb{Z}_N \to \mathbb{Z}_N$, recall that it sufficed to find an integer z such that $zr = kN$ (for some integer k) and $\gcd(z, r) = 1$. In other words, it sufficed to find a *lattice vector* in the lattice $\mathcal{L} = \{zr : z \in \mathbb{Z}\}$ that is an integer multiple of N and that satisfies $\gcd(z, r) = 1$. A similar thing is true here for the bivariate case, in which to solve for α (and hence understand the periodicity of $f_{g,h}$), it suffices to find a lattice vector $\vec{v} = (v_1, v_2) \in \mathcal{L}_\alpha$ such that $\gcd(v_2, r_g) = 1$. Ultimately, this affords a reduction from the discrete log problem to what one might call "lattice period finding".

Corollary 19.6 (Reduction from Discrete Log to Lattice Period Finding, HW6)**.** *Given a modulus N, $g \in \mathbb{Z}_N^\times$, its order r_g, and $h \in \langle g \rangle$, an efficient (classical or quantum) algorithm for finding a lattice vector $\vec{v} = (v_1, v_2) \in \mathcal{L}_\alpha$ such that $\gcd(v_2, r_g) = 1$ implies an efficient (classical or quantum) algorithm for finding the discrete log $\alpha = \log_g h$.*

19.5. Shor's Algorithm for the Discrete Log Problem

We now state an important fact about the complexity of computing $f_{g,h}$, which is related to the complexity of computing f_a from the previous lecture.

Fact 19.7. *For all moduli N and all $g, h \in \mathbb{Z}_N^\times$, $f_{g,h}$ is computable by an efficient, deterministic, classical computer (by implementing, for example, the modular exponentiation by repeated squaring algorithm). Hence, the quantum oracle*

$$\mathcal{O}_{f_{g,h}} : |x\rangle|y\rangle|0\rangle \mapsto |x\rangle|y\rangle|f_{g,h}(x, y)\rangle$$

can be implemented efficiently on a quantum computer.

[4] See, for example, Stark's *An Introduction to Number Theory*.

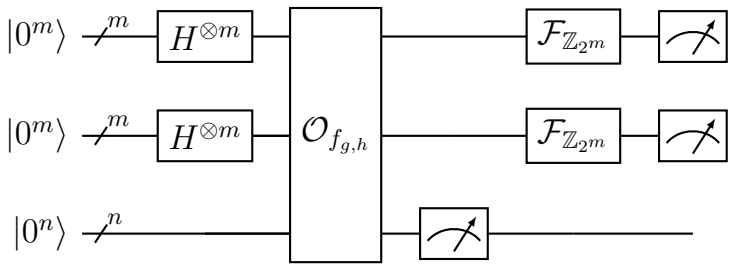
[5] For example, one could use the *extended Euclidean algorithm* to find a^{-1} (the modular multiplicative inverse of a, which we introduced last lecture) and set $x = a^{-1}b \bmod n$.

We are now ready to present Shor's algorithm for the discrete log problem. In the following, N is a modulus, $g \in \mathbb{Z}_N^\times$, r_g is the order of g (and obviously satisfies $r_g \leq N$), and $h \in \langle g \rangle$. Note, if you did not know r_g a priori, then you could use Shor's order finding algorithm from last lecture to first find r_g, and then proceed with the algorithm below. That said, in most cases of discrete log, $r_g = N - 1$, because discrete log is really only interesting when N is prime and g is a generator of $\mathbb{Z}_N^\times$. Thus, it is natural to assume that we already know r_g.

Shor's Algorithm for Discrete Log

1. Prepare $2m + n$ qubits in the state $|0^m\rangle|0^m\rangle|0^n\rangle$, where $2^m \geq r_g$ and $2^n \geq N$.[6]
2. Run the *bivariate QPF circuit* below on $f_{g,h}(x, y) = g^x h^{-y} \bmod N$ to recover a lattice vector $(v_1, v_2) \in \mathcal{L}_\alpha$.[7]
3. Check if $\gcd(v_2, r_g) = 1$. If so, continue. If not, go back to step 1.
4. Solve for α in $v_2\alpha = v_1 \pmod{r_g}$ (using, for example, the extended Euclidean algorithm) and return α.

The *bivariate QPF circuit* mentioned above is just the QPF circuit but with another register, and with a QFT over $\mathbb{Z}_{2^m} \times \mathbb{Z}_{2^m}$ as opposed to $\mathbb{Z}_{2^m}$. In particular, it is the quantum circuit:



Note, the reason we need n qubits in the bottom register and only m qubits in each of the upper registers is because the image of $f_{g,h}$ is $\mathbb{Z}_N$ but the domain is just $\mathbb{Z}_{r_g}$.

We will now state the main theorem, a special case of which you will prove on HW6 (namely, you will prove the "exact case" in which it is assumed that $r_g = 2^m$).

[6] Since $N \geq r_g$ for all discrete log instances, $n \geq m$.

[7] In fact, this step is only exact in the "exact case" in which $r_g = 2^m$. In the "non-exact case" in which r_g is not a power of two (so $r_g \nmid 2^m$), this step succeeds with a sufficiently high probability. We will not prove this in this class (because it is quite complicated), but the point is basically the same as the non-exact cases we have studied for QPE and QPF. For the details, see Shor's original paper, which is quite accessible, especially at this point in the course.

Theorem 19.8 (HW6)**.** *There exists an efficient quantum computer (namely, that specified by Shor's discrete log algorithm) that outputs* $\log_g h$ *(and hence solves the discrete log problem) with probability*

$$\Omega\left(\frac{1}{\log\log r_g}\right) = \Omega\left(\frac{1}{\log\log N}\right) = \Omega\left(\frac{1}{\log n}\right).$$

Thus, there exists an efficient quantum computer that solves the discrete log problem with probability at least $1 - 2^{-O(\mathrm{poly}(n))}$*.*

In consequence, quantum computers can break any cryposystem whose security guarantee comes from the assumption that the discrete logarithm problem is hard. We will explore this cryptographic implication in two lectures from now. Before this, though, we will see a problem that quantum computers can solve efficiently, and that, in some sense, subsumes all the other interesting quantum algorithms that we have seen so far. In this sense, the problem we will discuss next lecture is *the* problem that quantum computers can solve efficiently.

Lecture 20
The Hidden Subgroup Problem

Discussion 20.1. Discuss with your group what you took away from last time.

Last lecture, we discussed Shor's algorithm for the discrete logarithm problem. Like Shor's factoring algorithm we saw that the discrete log problem reduces to a period finding problem, albeit in higher dimensions. Nevertheless, Shor's discrete log algorithm affords an exponential speed-up relative to all currently known classical algorithms for discrete log.

In this lecture, we will introduce the problem that unifies all the interesting quantum exponential speed-ups we have seen so far. In particular, we will define the so-called *hidden subgroup problem* (abbreviated *HSP*), and we will see that Simon's problem, the factoring problem, and the discrete log problem can all be framed as an instance of this problem. Moreover, we will see that there is actually an efficient quantum algorithm for the *abelian* HSP (abbreviated *AHSP*), which implies an efficient quantum algorithm for Simon's problem, factoring, and discrete log. Thus, in a very real sense, the quantum algorithm for AHSP is *the* quantum algorithm that subsumes most of the important quantum algorithms that we know.[1]

20.1. Groups, Hiding Functions, and the Hidden Subgroup Problem

To define the hidden subgroup problem requires several new mathematical notions. (That said, the notion of a *group* should be familiar, as we have seen many examples of groups in this course.)

Definition 20.1.

- Let G be a non-empty set and $\cdot : G \times G \to G$ a binary operation on G. We

[1] For a great lecture on the HSP (and quantum algorithms for other algebraic problems), I recommend Andrew Childs' tutorial from QIP 2021, which is available on YouTube here.

call the pair $(G, \cdot)$ a *group* iff the following three conditions hold:[2]

(i) the binary operation $\cdot$ is *associative*, i.e., for all $a, b, c \in G$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$,

(ii) G has an *identity*, i.e., there exists $e \in G$ such that for all $a \in G$, $a \cdot e = e \cdot a = a$,

(iii) G contains *inverses*, i.e., for all $a \in G$, there exists $a^{-1} \in G$ such that $a \cdot a^{-1} = a^{-1} \cdot a = e$.

- The *order of* $(G, \cdot)$ is the cardinality of G, namely $|G|$.
- A group $(G, \cdot)$ is *finite* iff its order $|G|$ is finite.
- We say $(G, \cdot)$ is *abelian* iff the binary operation $\cdot$ is commutative, i.e., for all $a, b \in G$, $a \cdot b = b \cdot a$. Otherwise, $(G, \cdot)$ is *non-abelian.*
- We call H a *subgroup of* $(G, \cdot)$ (or just a *subgroup of* G if the binary operation is contextually clear) iff H is a subset of G and $(H, \cdot)$ is a group. We write $H \leq G$ iff H is a subgroup of G.

Exercise 20.1.

1. *Is* $\mathbb{Z}_N$ *a finite non-abelian group?*
2. *Is* $\mathbb{Z}_N^\times$ *a finite abelian group?*
3. *What is the identity element in* $\mathbb{Z}_2^n = (\{0,1\}^n, \oplus)$*, where* $\oplus$ *is bitwise XOR?*
4. *Is*
$$1 + \mathbb{Z}_N = \big(\{1, 2, 3, \ldots, 1 + (N-1)\}, + \bmod N\big)$$
a subgroup of $\mathbb{Z}_N$*?*

Definition 20.2. Let G be a group, H a subgroup of G, and X a set. We say a function $f : G \to X$ *hides the subgroup* H iff for all $x, y \in G$,

$$f(x) = f(y) \iff y = x \cdot h \text{ for some } h \in H.[3]$$

In this case, we call H *the subgroup hidden by* f, or, if f is contextually clear, *the hidden subgroup.*

[2] Often the binary operation in a group is contextually clear. In this case, we will sometimes label a group $(G, \cdot)$ not as a tuple, but just by the set G.

[3] For the mathematically inclined, this is equivalent to the statement that f is constant on the *left cosets of* H and is different on the different left cosets of H.

Already, you may see how hidden subgroups relate to Simon's problem.

Example 20.1. Let

- $G = \mathbb{Z}_2^n = (\{0,1\}^n, \oplus)$, where $\oplus$ is bitwise addition modulo 2,
- $H = \{0^n, s\}$ for some $s \in \{0,1\}^n \backslash \{0^n\}$ (indeed, $H \leq G$, as you will prove on HW6),
- $X = \{0,1\}^n$,
- and $f : G \to X$ such that

$$f(x) = f(y) \iff y = x \text{ or } y = x \oplus s.$$

Since

$$y = x \text{ or } y = x \oplus s \iff y = x \oplus h \text{ for some } h \in H = \{0^n, s\},$$

it is plain that f hides the subgroup $H \leq G$.

In order to define the hidden subgroup *problem* requires just one more mathematical notion, which should also be somewhat familiar given our past discussions of universal gate sets and generators of $\mathbb{Z}_N^\times$.

Definition 20.3. Let $(G, \cdot)$ be a group. We call $S \subseteq G$ a *generating set of* $(G, \cdot)$ iff for all $a \in G$, there exist $s_1, s_2, \cdots, s_k \in S$ such that

$$a = s_1 \cdot s_2 \cdot \cdots \cdot s_k.$$

In other words, S is a *generating set of* $(G, \cdot)$ iff every element in G can be written as elements in S that are composed using the group operation in $(G, \cdot)$.

Question 20.1. *Is $S = \{0\}$ a generating set of $\mathbb{Z}_N$? What about $S = \{1\}$?*

We are now in a position to define the hidden subgroup problem.

The Hidden Subgroup Problem (HSP)

Input: A group $(G, \cdot)$ and a function $f : G \to X$ as an oracle that hides a subgroup $H \leq G$.[4]

Output: A generating set of H by querying f.

[4] In fact, one needs to restrict to *discrete* groups that can be computationally interpreted, but these are subtleties that we will not address.

Note, the *abelian HSP* (abbreviated *AHSP*) is the HSP when the underlying group $(G, \cdot)$ is abelian. Similarly, the *non-abelian HSP* is the HSP when the underlying group is non-abelian.

Exercise 20.2. *In terms of $|G|$, classically, how many queries to f suffice to solve the HSP?*

Indeed, classically one can show the following theorem, whose proof is similar in spirit to the proof that Simon's problem is classically hard.[5]

Theorem 20.1. *Let G be a finite group with N distinct subgroups H. Then, a classical computer must make $\Omega(\sqrt{N})$ queries to solve the HSP. In many cases, $N \gg \log |G|$, so this algorithm is not efficient.*

Given this, we will now see why the AHSP is particularly interesting, and why having an efficient algorithm for it implies an efficient algorithm for a myriad of other interesting problems.

20.2. The Many Reductions to AHSP

Here, we will show that several problems we have seen in this course reduce to particular instances of the AHSP. Therefore, an efficient quantum algorithm for the general AHSP implies an efficient quantum algorithm for all of these problems.[6]

Let's start with Simon's problem. Example 20.1 proves that Simon's problem involves a hiding function. On HW6, you will prove the following reduction.

Fact 20.2 (Reduction from Simon's Problem to AHSP, HW6)**.** *An efficient (classical or quantum) algorithm for the AHSP implies an efficient (classical or quantum) algorithm for Simon's problem.*

Beyond Simon's problem, period finding is also an instance of the AHSP, as the following claim shows.

Claim 20.3. *Let $f : \mathbb{Z} \to \mathbb{Z}$ be an r-periodic function such that $f(0), f(1), \ldots, f(r-1)$ are all distinct values. Moreover, let $G = \mathbb{Z}$ and $X = \mathbb{Z}$. Then, f hides $H = r\mathbb{Z} = \{\ldots, -2r, -r, 0, r, 2r, \ldots\}$.*

[5] For a proof, see, for example, Andrew Childs' notes here.

[6] Here we will not discuss the Deutsch–Jozsa problem, even though it does (at least in certain cases), reduce to an instance of the AHSP.

Proof. It is easy to see that $H \leq \mathbb{Z}$. Since f is r-periodic, for all $h = kr \in r\mathbb{Z}$,

$$y = x + h \implies f(x) = f(y).$$

Finally, since $f(0), f(1), \ldots, f(r-1)$ are all distinct values, it holds that if there does *not* exist $k \in \mathbb{Z}$ such that $y = x + rk$, then $f(x) \neq f(y)$. Thus, taking the contrapositive of this last statement, we get that

$$\begin{aligned} f(x) = f(y) &\implies y = x + kr \text{ for some } k \in \mathbb{Z} \\ &\implies y = x + h \text{ for some } h \in H = r\mathbb{Z}. \end{aligned}$$

Putting the above two implications together then gives

$$f(x) = f(y) \iff y = x + h \text{ for some } h \in H.$$

Therefore, f hides H, as claimed. ■

Consequently, we obtain a reduction from period finding to the AHSP. Using the fact that factoring is just a particular instance of period finding (see Lecture 17), it holds that factoring also reduces to the AHSP.

Corollary 20.4 (Reduction from Period Finding/Factoring to AHSP)**.** *An efficient (classical or quantum) algorithm for the AHSP implies an efficient (classical or quantum) quantum algorithm for period finding (and hence an efficient (classical or quantum) algorithm for factoring integers).*

Finally, we will prove the more general result that *lattice* period finding is also an instance of the AHSP.

Claim 20.5. *Let $f : \mathbb{Z}^n \to \mathbb{Z}$ be an $\mathcal{L}$-periodic function for some lattice $\mathcal{L}$ such that if $\vec{y} - \vec{x} \notin \mathcal{L}$, then $f(\vec{x}) \neq f(\vec{y})$.*[7] *Moreover, let $G = \mathbb{Z}^n$ and $X = \mathbb{Z}$. Then, f hides $H = \mathcal{L} \leq \mathbb{Z}^n$.*

Proof. It is easy to see that $H = \mathcal{L} \leq \mathbb{Z}^n$. Moreover, since f is $\mathcal{L}$-periodic, if $\vec{v} \in H$, then

$$\vec{y} = \vec{x} + \vec{v} \implies f(\vec{x}) = f(\vec{y}).$$

Finally, since $\vec{y} - \vec{x} \notin H$ implies $f(\vec{x}) \neq f(\vec{y})$, it holds that

$$\begin{aligned} f(\vec{x}) = f(\vec{y}) &\implies \vec{y} - \vec{x} \in \mathcal{L} \\ &\implies \vec{y} = \vec{x} + \vec{v} \text{ for some } \vec{v} \in H. \end{aligned}$$

[7] In the case of a univariate, r-periodic function $f : \mathbb{Z} \to \mathbb{Z}$, this condition is equivalent to the condition that $f(0), f(1), \ldots, f(r-1)$ are all distinct.

Putting the above two implications together then gives

$$f(\vec{x}) = f(\vec{y}) \iff \vec{y} = \vec{x} + \vec{v} \text{ for some } \vec{v} \in H.$$

Therefore, f hides H, as claimed. ■

Therefore, since the discrete log problem is an instance of lattice period finding, we obtain the following reduction.

Corollary 20.6 (Reduction from Discrete Log to AHSP)**.** *An efficient (classical or quantum) algorithm for the AHSP implies an efficient (classical or quantum) algorithm for the discrete logarithm problem.*

All together, we learn that almost all of the interesting quantum algorithms we have studied in this course (save, for example, Grover's algorithm) are just particular instances of the AHSP. In the next section, we will briefly cover how the quantum part of the quantum algorithm for the general AHSP works.

20.3. An Efficient Quantum Algorithm for AHSP

We will begin with the following fact whose proof is, unfortunately, beyond the scope of this course, as it requires a considerable amount of group theory and representation theory.

Fact 20.7. *Let G be a finite group. There exists an efficient quantum computer that solves the AHSP with probability $\Omega(1)$ using $O(\log |G|)$ many queries to the hiding function f. Therefore, if f is efficiently computable, then there exists an efficient quantum computer that solves the AHSP with probability $\Omega(1)$.*

Note, in the case of lattice period finding (where the group G is infinite), this algorithm can still work by imposing a "cutoff" on the size of the group. Indeed, this is exactly what we did for univariate period finding, where we were given some upper-bound on the period, and so we were able to solve the period finding problem on a finite part of the underlying group.

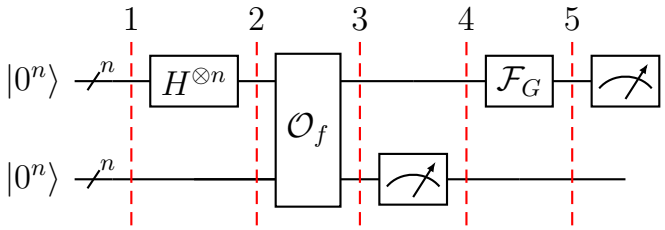
To gain some intuition for how the quantum part of this algorithm works, we will briefly go through the quantum circuit so to reveal that it is structurally similar to many of the other quantum circuits we have seen throughout this course.

The Quantum Part of the Quantum Algorithm for the AHSP

1. Prepare $2n$ qubits in the state $|0^n\rangle|0^n\rangle$, where $2^n \geq |G|$.

2. Apply $H^{\otimes n}$ to the first n qubits.
3. Apply $\mathcal{O}_f$.
4. Measure the last n qubits.
5. Apply the *QFT over* G, denoted $\mathcal{F}_G$, to the first n qubits.
6. Measure the first n qubits.

As a quantum circuit, the quantum part of the quantum algorithm for the AHSP is as follows.



Of course, this circuit is structurally identical to the QPF circuit (both the univariate and bivariate instances) and the circuit in Simon's problem. As the last section proves, this is not a coincidence, because all of these problems are in some sense the same problem—namely, just the AHSP for a different choice of group and function. Because of this, the analyses of the quantum algorithms for all of these problems were nearly identical, and that is because, structurally speaking, they use the same quantum circuit. Indeed, using the circuit above, it is a relatively easy exercise to show the following fact.

Fact 20.8. *Let* $|\psi_i\rangle$ *be the state at step* i*. Then,*

$$|\psi_4\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |g_0 \cdot h\rangle |f(g_0)\rangle.$$

Thus, what you get in the general, AHSP quantum algorithm is a uniform superposition of all elements in the first register that are consistent with the measurement result in the second register, but shifted (according to the group operation) by whatever the argument in the measurement outcome is. This follows immediately from the structure of the AHSP. In the next step of the quantum algorithm, namely, applying the quantum Fourier transform (QFT) over the group G, denoted $\mathcal{F}_G$, is more difficult. Suffice it to say, the basic idea is the same: $\mathcal{F}_G$ weeds out the "period" in the state $|\psi_4\rangle$, so that after you measure $|\psi_5\rangle$, all it takes is polynomial time classical post-processing (e.g., continued fractions) to learn the subgroup H. For more on the quantum algorithm for the AHSP, I recommend this survey article by Andrew Childs and Wim van Dam.

20.4. The Non-Abelian HSP*

Given that the past discussions have all been about the *abelian* HSP, perhaps the most obvious question to ask is, what about the *non-abelian* HSP? Indeed, this is a great question with a rich history, but the immediate reply is that, unlike AHSP, there is no known, general quantum algorithm for the non-abelian HSP. That said, there are many partial results known,[8] which are interesting given the computational problems that reduce to instances of the non-abelian HSP.

For example, there is a reduction from the *graph isomorphism problem* to the HSP over the *symmetric group*, which is the (non-abelian) group of bijections from a set back to itself. As you may know, the graph isomorphism problem is one of the most important problems in computational complexity theory, in part because it is neither known to be NP-complete nor is it known to be classically or quantumly tractable. Thus, what this reduction shows is that if quantum computers can efficiently solve the general, non-abelian HSP, then quantum computers can efficiently solve the graph isomorphism problem as well. Many consider that unlikely.

Moreover, there is also an important problem to do with lattices that reduces to the HSP over the *dihedral group*, which is the (non-abelian) group of symmetries of a regular polygon. The problem is called *the unique shortest vector problem*, and next lecture we will briefly describe what (variants of) this problem are all about. The upshot is that these lattice problems may underlie future cryptosystems, and the inability of quantum computers to solve the non-abelian HSP increases our credences that these cryptosystems are quantum-secure.

[8] See, for example, Greg Kuperberg's sieve for the HSP over the dihedral group here, as well as the Ettinger–Høyer–Knill Theorem, available here.

Lecture 21
Quantum and Post-Quantum Cryptography

Discussion 21.1. Discuss with your group what you took away from last time.

Last lecture, we discussed the hidden subgroup problem (HSP), and we learned that there is a quantum algorithm that can solve this problem efficiently. Moreover, we saw that Shor's algorithms for factoring and the discrete logarithm problem are both instances of the *abelian* HSP (for different choices of the underlying group), as is Simon's problem.

Taken together, Shor's algorithms for factoring and the discrete log problem are a huge concern when it comes to cryptography, as many contemporary cryptosystems get their security guarantees from the factoring and discrete log problems. Thus, in a quantum world, quantum adversaries (i.e., bad actors with access to a universal, fault-tolerant quantum computer) could easily read internet traffic, break into private bank accounts, decrypt intercepted communications (such as in *harvest now, decrypt later* surveillance), etc. In this lecture, we will discuss some details of *public-key cryptography*, and we will also discuss two different ways to remedy the fact that quantum computers break many current public-key cryptosystems.

21.1. Public-Key Cryptography

While there are many different types of cryptosystems, among the most practical and important are those that are *public-key cryptosystems* (or *PKC*s for short). These cryptosystems allow the following seemingly-impossible task, part of which we discussed when we talked about the Diffie–Hellman protocol a few lectures ago.

1. You and I are in a room full of people, and we have never talked before.
2. You and I shout across the room, and we make sure everyone hears.

3. After a short time, you and I shout across the room, and we make sure that everyone hears. This time, however, only you and I understand what we are saying to each other. Because of this, we can publicly communicate our private messages, and no one (at least on a reasonable time scale) will be able to figure out what we are saying.

That this is possible is evidently important in our present-day world, because, as unfortunate as it is, there are many types of bad actors who are interested in eavesdropping on our private conversations. Now, there are many computational solutions to this, but there is overwhelming agreement that any genuine solution should be one in which the eavesdroppers *know and understand* exactly how it is we are protecting our messages. This is called *Kerckhoffs's principle*.

In the context of PKCs, Kerckhoffs's principle says that the security guarantee of the PKC cannot come from an assumption that is based on an eavesdropper not knowing some part of how the system works. Rather, it must stem from a sort of computationally difficult problem that is intrinsic to the system. To see how to put this formally, we will now briefly explain what a PKC even is, along with some of the formal notions that go in to defining their security. To do this right, however, requires the following definition.

Definition 21.1. A function $\epsilon : \mathbb{N} \to [0, \infty)$ is *negligible* iff for all $c \in \mathbb{N} \cup \{0\}$, there exists $N_c \in \mathbb{N}$ such that for all $n > N_c$,

$$\epsilon(n) < \frac{1}{n^c}.$$

Exercise 21.1. *Determine which of the following functions are negligible:*

- $\frac{1}{3n^4+n^3}$
- $\frac{1}{2^n}$
- $\frac{1}{(1.0000000001)^n}$
- $\frac{1}{2^{c \log n}}$, $c \geq 0$.

We will now formally define what a PKC is.

Definition 21.2.

- A *public-key cryptosystem* (abbreviated *PKC*) Π is a triple of classical, probabilistic, polynomial time algorithms $(\texttt{Gen}, \texttt{Enc}, \texttt{Dec})$ such that:

(i) `Gen` is a *key generation* algorithm.

Input: A unary string 1^n, where n is a *security parameter*.[1]

Output: (PK, SK), where PK is a *public key* (a.k.a. an *encryption key*) and SK is a *secret key* (a.k.a. a *decryption* or *private key*).

We write this as $(PK, SK) \leftarrow \texttt{Gen}(1^n)$.

(ii) `Enc` is an *encryption algorithm*.

Input: A public key PK and a *plaintext message* m.

Output: A *ciphertext* c.

We write this as $c \leftarrow \texttt{Enc}(PK, m)$.

(iii) `Dec` is a *decryption algorithm*.

Input: A secret key SK and a ciphertext c.

Output: A plaintext message m'.

We write this as $m' \leftarrow \texttt{Dec}(SK, c)$.

- We say Π is *complete* (a.k.a. *correct*) iff there exists a negligible function ϵ such that for all security parameters n and all plaintext messages m,

$$\Pr_{\substack{(PK,SK)\leftarrow\texttt{Gen}(1^n)\\ c\leftarrow\texttt{Enc}(PK,m)\\ m'\leftarrow\texttt{Dec}(SK,c)}}[m' = m] = 1 - \epsilon(n).$$

  In other words, with probability $1 - \epsilon(n)$ over the internal randomness of `Gen`, `Enc`, and `Dec`, decrypting any encrypted message m will return the original message m with overwhelming probability.

Example 21.1.

- The *Rivest–Shamir–Adleman (RSA) cryptosystem*. It is complete.
- The *ElGamal cryptosystem*. It is also complete.

Given a PKC, one of the many things we would like it to be is *secure*.[2] Note

[1] You can think of the security parameter as specifying the length of the public key, i.e., $|PK| \approx n$.

[2] You also want it to admit *authentication*, so that you can be confident who you are talking to is who they say they are. This is usually done using a Message Authentication Code (MAC), which itself is a triple of classical algorithms that satisfy certain constraints. We will not discuss MACs here, but any good book on cryptography will, e.g., Stinson and Paterson's *Cryptography: Theory and Practice*.

that it more or less suffices to define security with respect to single bit messages, because an overall message is just a bunch of single bits.[3]

Definition 21.3.

- We say a PKC $\Pi = (\texttt{Gen}, \texttt{Enc}, \texttt{Dec})$ is *classically secure* iff for all classical, probabilistic, polynomial time adversaries $\mathcal{A}$, there exists a negligible function ϵ such that for all security parameters n and all single-bit messages $m \in \{0,1\}$,

$$\Pr_{\substack{(PK,SK)\leftarrow\texttt{Gen}(1^n)\\ c\leftarrow\texttt{Enc}(PK,m)\\ m'\leftarrow\mathcal{A}(PK,c)}}[m' = m] \leq \frac{1}{2} + \epsilon(n).$$

  In other words, no efficient, classical adversary $\mathcal{A}$ can recover m with just the ciphertext c and public key PK better than they could by just randomly guessing m.

- We say a PKC Π is *quantum-secure* iff the above definition holds for all efficient *quantum* adversaries $\mathcal{A}$.

Exercise 21.2. *Let Π be a PKC. Argue that if a classical or quantum adversary $\mathcal{A}$ knows SK, then Π is not classically or quantum-secure, respectively.*

Fact 21.1 (Shor)**.**

- *The RSA cryptosystem is not quantum-secure.*
- *More generally, if Π is a PKC with primitives that rely on the security of instances of the abelian hidden subgroup problem (e.g., factoring, discrete log, or elliptic curves), then Π is not quantum-secure.*[4]

Therefore, many contemporary cryptosystems are not quantum-secure. This raises a very important question: what should we do about that?

[3] For example, to encrypt a k bit message $m \in \{0,1\}^k$, you can do it as follows:

$$\texttt{Enc}(PK, m) = \texttt{Enc}(PK, m_0).\texttt{Enc}(PK, m_1).\ldots.\texttt{Enc}(PK, m_{k-1}).$$

Of course, there are caveats to this, but we will not discuss those details here. See, e.g., Stinson and Paterson's *Cryptography: Theory and Practice* for details.

[4] For example, Diffie–Hellman key exchange and the whole ElGamal cryptosystem are not quantum-secure, because both rely on the security guarantee of the discrete log problem.

To physicists, the most theoretically interesting solution here is to "upgrade" the algorithms used in the definition of PKC to *quantum* algorithms and also to upgrade the communication channels used to *quantum* channels, so that they transmit quantum states and not just bits. At least on the surface, this is interesting from a quantum information point of view, because quantum mechanics forbids any eavesdropper from copying the quantum information thanks to the no-cloning theorem. In this respect, any eavesdropper will necessarily disturb the quantum state being sent. Thus, again on the surface, it sure seems like quantum-based PKCs could be as secure as we are confident that the laws of quantum mechanics are a true representation of how the world works. Of course, it doesn't turn out to be quite that good, but it's an interesting prospect nonetheless.

To illustrate one example of this "quantum alternative", we will present a famous quantum protocol for key distribution called *BB84*. Here, "BB" encodes the last initials of its two inventors, Charles Bennett and Gilles Brassard, and the number "84" is the year in which they proposed it. Interestingly, BB84 is *provably secure* in an information-theoretic sense, provided certain assumptions hold (one of which is rather idealistic, as we shall discuss below).

The BB84 Quantum Key Distribution (QKD) Protocol

1. Given a security parameter n, Alice generates $x, y \sim \{0,1\}^n$ uniformly. Alice then prepares the n-qubit state

$$|\psi\rangle = \bigotimes_{k=0}^{n-1} |\psi_{x_k y_k}\rangle,$$

where

$$|\psi_{x_k y_k}\rangle = \begin{cases} |0\rangle & \text{if } x_k y_k = 00 \\ |1\rangle & \text{if } x_k y_k = 10 \\ |+\rangle & \text{if } x_k y_k = 01 \\ |-\rangle & \text{if } x_k y_k = 11. \end{cases}$$

In other words, Alice encodes x quantumly as follows:

- If $x_k = 0$, then she randomly assigns one of $|0\rangle, |+\rangle$ to it.
- If $x_k = 1$, then she randomly assigns one of $|1\rangle, |-\rangle$ to it.

2. Alice sends $|\psi\rangle$ to Bob.

3. In transit, $|\psi\rangle$ will change due to imperfections in the quantum channel (e.g., imperfections in the fiber optic cable used to send photons in a certain polarization state) or due to an eavesdropper, Eve, interfering with the state $|\psi\rangle$, perhaps trying to gain information about the state. Here, to illustrate how the protocol works, we will make the unrealistic, physical assumption that the quantum channel is perfect (i.e., *noiseless*) and that the state Bob receives is *pure*.

4. Bob receives a state $|\phi\rangle$, which was potentially tampered with by Eve. Bob generates $y' \sim \{0,1\}^n$ uniformly, and then measures $|\phi\rangle$ using the Hermitian operator

$$H = \bigotimes_{k=0}^{n-1} W_{y'_k},$$

where

$$W_{y'_k} = \begin{cases} Z & \text{if } y'_k = 0 \\ X & \text{if } y'_k = 1. \end{cases}$$

Bob then constructs an n-bit string x', where

$$x'_k = \begin{cases} 0 & \text{if } W_{y'_k} = Z \text{ and Bob measured } |0\rangle, \\ 0 & \text{if } W_{y'_k} = X \text{ and Bob measured } |+\rangle, \\ 1 & \text{if } W_{y'_k} = Z \text{ and Bob measured } |1\rangle, \\ 1 & \text{if } W_{y'_k} = X \text{ and Bob measured } |-\rangle. \end{cases}$$

At this point in the protocol, it is easy to reason that if $y_k = y'_k$ for some k, and if there was no noise and no eavesdropping on the channel (so that $|\phi\rangle = |\psi\rangle$, then $x_k = x'_k$ with certainty.

5. Over a (not necessarily secure) classical communication channel, Alice announces y and Bob announces y'.

6. Alice and Bob discard any bits x_k and x'_k for which $y_k \neq y'_k$. This corresponds to the bits where Bob measured a different basis than Alice prepared. Supposing they exist, the remaining bits of x_k and x'_k represent a semi-private (and possibly noisy) key.

7. At this point, Alice and Bob need to understand how much Eve might know about x and x'. To do this, they will sacrifice about half of the remaining bits of x and x' to "test" for Eve. Note, the sacrificed bits are chosen randomly. For Alice, the sacrificed bits form a string w

and the remaining, unsacrificed bits form a string z, while for Bob, the sacrificed bits form a string w' and the remaining, unsacrificed bits form a string z'.

8. Alice and Bob compare w and w' over the classical channel. If the error rate between w and w' is above some pre-chosen threshold, then they abort the protocol. Otherwise, they continue.
9. Alice and Bob now have the strings z and z' whose bits agree with high probability. Using classical techniques known as *information reconciliation* and *privacy amplification*, Alice and Bob can "boost" the strings z and z' to become a usable secret key SK.[5]

In the above, *information reconciliation* is a sort of "distributed error correction" in which Alice and Bob can boost the probability that z and z' agree in all positions with high probability (though this will invariably leak a small amount of information to Eve). On the other hand, *privacy amplification* is a classical technique involving a suitable *hash function* that takes z and z' (which after information reconciliation are more or less the same string) and compresses it to a shared string that Eve will know almost nothing about (despite knowing a little bit about z and z').

Now, as presented, the key thing to note about BB84 is that if done via a perfect, noiseless quantum channel, then Alice and Bob can detect Eve's presence with high probability. This is due to the following, informal fact.

Fact 21.2 (Informal)**.** *The more information Eve gains from $|\psi\rangle$, the more x and x' will differ in the BB84 protocol.*

The main difficulty in proving the above fact is that Eve can use *any* strategy that is allowed by quantum mechanics. In particular, it is not sufficient to consider what happens if Eve interacts with the state using some unitary transformation like CNOT, nor is it sufficient to consider what happens if Eve measures the state in some basis. Nevertheless, there is a simple, no-cloning-like argument for why Eve cannot extract information from the state, which you will show on HW6.

Fact 21.3 (HW6)**.** *Fix $n, m \geq 1$ and $|u\rangle \in \mathbb{C}^{2^m}$, and let $U \in \mathrm{U}(2^{n+m})$ be such that for all $|\psi\rangle \in \mathbb{C}^{2^n}$,*

$$U|\psi\rangle|u\rangle = |\psi\rangle|v_\psi\rangle$$

for some $|v_\psi\rangle \in \mathbb{C}^{2^m}$ that (in general) depends on $|\psi\rangle$. If $|\psi\rangle, |\phi\rangle \in \mathbb{C}^{2^n}$ are such that $\langle\psi|\phi\rangle \neq 0$, then

$$|v_\psi\rangle \sim |v_\phi\rangle,$$

[5]For example, they could use SK with a *one-time pad* (OTP) that, with the assumptions made in our presentation of BB84, admits *provable, information-theoretic security.*

where $\sim$ denotes the operational equivalence between states.

In the context of BB84, the above fact implies the following corollary.

Corollary 21.4. *If Eve tries to glean information from the intercepted BB84 state $|\psi\rangle$ by applying a unitary transformation like that in the above fact (which notably does not disturb the BB84 state), then she will not be able to learn anything from the state by measuring $|v_\psi\rangle$, because the state $|v_\psi\rangle$ is operationally the same for all possible intercepted BB84 states $|\psi\rangle$.*

We will now discuss some of the downsides to BB84. In our presentation of it, we assumed the existence of a *noiseless* quantum channel, which is physically untenable.[6] In reality, there will always be noise in the quantum signal, which implies that in BB84, Bob will never receive the exact state $|\psi\rangle$ that Alice prepared, even if there is no eavesdropper. Of course, we can imagine error-correcting the channel, but still Alice and Bob will need a way of determining whether the error in Bob's state is ultimately due to imperfections in the channel or due to an eavesdropper. You can imagine that this is complicated.

Another big downside of BB84 (and in fact all quantum cryptography schemes) is that they assume a world in which scalable, fault-tolerant quantum devices not only exist, but one in which they are somewhat mainstream. This, of course, is far away, because, at least at the moment, such quantum systems are really only achievable in advanced scientific laboratories that have the resources to properly shield quantum states from their environment. Given this (and also many other reasons[7]), it is natural to consider what many regard as the more practical alternative for the future of cryptography.

21.3. The Classical Alternative: Post-Quantum Cryptography

Besides the quantum alternative, there is also the classical alternative, which most call the "post-quantum" or "quantum-resistant" alternative. This alternative is to build future cryptographic schemes not based on quantum devices, but rather to just do the "same" classical cryptography as before, but upgrade the cryptosystems so that they get their security guarantees from problems that we suspect are *not* in BQP. From a practical and infrastructural perspective, this is arguably the most sensible approach, because, if possible, it implies that we do not need to

[6] One also has to assume that Alice and Bob possess a classical *authenticated* channel, but we will not discuss that subtlety here.

[7] See this note from the National Security Agency (NSA) for others.

completely revamp hardware to be quantum-compatible. Rather, all it takes is a software update to run whatever new classical algorithm that underlies the new cryptosystem. Any cryptographic proposal like this falls under the heading of *post-quantum cryptography*, because it is classical cryptography in a quantum world that is resistant to quantum adversaries.

Of course, it is not easy to come up with a new classical cryptosystem. You might think, "why not just base it on some NP-complete problem like 3-SAT?" Indeed, this is worth thinking about for a second.

Exercise 21.3. *We suspect* BQP $\neq$ NP*, so why not devise a purely classical PKC whose security guarantee stems from an* NP*-complete problem like* 3*-SAT?*

One reason this doesn't work is because you want the security guarantee of a cryptosystem to come from a problem that is hard *on average*, not from a problem that is hard in *the worst case*, the latter being how NP problems are defined.[8] This is why problems like factoring are used in today's cryptosystems, because it seems hard in almost every case, provided the numbers involved are large enough. Interestingly, the problems we mention below possess this feature in a very formal sense, since they exhibit what is called a *worst-to-average case reduction*. This means that an efficient algorithm for the (ostensibly easier) average case problem implies an efficient algorithm for the (ostensibly harder) worst case problem. Thus, if there is no efficient algorithm for the worst case problem, then there is no efficient algorithm for the average case problem. That is a superb security guarantee!

Before discussing some of these ideas, however, we note that post-quantum cryptography is a huge field, and there is no way to do it justice in just a single lecture. Because of this, below we will focus in on a small (but very important) corner of post-quantum cryptography, namely, *lattice-based cryptography*. For this reason, we will need to recall some things about lattices.

Definition 21.4.

- Let $\vec{b}_1, \ldots, \vec{b}_n$ be n linearly independent vectors in $\mathbb{R}^m$, where $m \geq n$. Recall that a *lattice* $\mathcal{L}$ is all integer linear combinations of these vectors,

$$\mathcal{L} = \left\{ z_1\vec{b}_1 + \cdots + z_n\vec{b}_n : z_1, \ldots, z_n \in \mathbb{Z} \right\}.$$

- $B = \left\{\vec{b}_1, \ldots, \vec{b}_n\right\}$ is called a *basis* of the lattice $\mathcal{L}$.

[8] There are other reasons, too. See, for example, this paper by Goldreich and Goldwasser *On the Possibility of Basing Cryptography on the Assumption that* P $\neq$ NP, and also Impagliazzo's famous paper, *A Personal View of Average-Case Complexity*, available here.

- Each $\vec{v} \in \mathcal{L}$ is called a *lattice vector*.

Given a lattice, it is guaranteed that there is a *shortest* vector in the lattice, with respect to the usual Euclidean ℓ^2-norm.

Question 21.1. *Given a lattice $\mathcal{L}$, what lattice vector $\vec{v} = (v_1, v_2, \ldots, v_n) \in \mathcal{L}$ minimizes the ℓ^2-norm,*

$$\|\vec{v}\|_2 = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2} \ ?$$

This isn't all that interesting. What is interesting, however, is the shortest *non-zero* lattice vector, where here "shortest" is meant with respect to the ℓ^2-norm.[9]

Definition 21.5.

- Let $\mathcal{L}$ be a lattice. We write $\lambda_1(\mathcal{L})$ for the length (in ℓ^2-norm) of the shortest non-zero lattice vector in $\mathcal{L}$. Symbolically,

$$\lambda_1(\mathcal{L}) \coloneqq \min_{\vec{w} \in \mathcal{L} \setminus \{\vec{0}\}} \|\vec{w}\|_2.$$

- We call a vector $\vec{v} \in \mathcal{L}$ *a shortest vector in* $\mathcal{L}$ iff $\vec{v} \neq \vec{0}$ and $\|\vec{v}\|_2 = \lambda_1(\mathcal{L})$. In other words, $\vec{v}$ is a shortest vector iff it is an *argument* that satisfies the above minimization,

$$\vec{v} = \arg\min_{\vec{w} \in \mathcal{L} \setminus \{\vec{0}\}} \|\vec{w}\|_2.$$

These definitions naturally gives rise to the following computational problem.

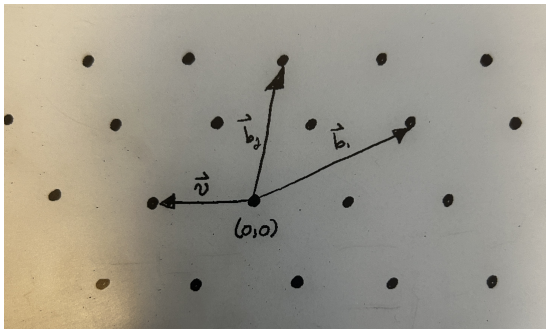
The Shortest Vector Problem (SVP)

Input: a basis B of a lattice $\mathcal{L}$.

Output: a shortest vector $\vec{v} \in \mathcal{L}$.

Pictorially, the two dimensional SVP is to find the vector label $\vec{v}$ in the image below, where $\vec{b}_1$ and $\vec{b}_2$ are the inputted basis vectors.

[9] It is of course interesting to study short non-zero vectors with respect to other norms. See, for example, Huck Bennett's survey here, where this is discussed.



Question 21.2. *In general, are the coefficients in the linearly combination of* $\vec{b}_1$ *and* $\vec{b}_2$ *that equal* $\vec{v}$ *guaranteed to be small?*

Thus, even a brute-force search of the lattice (via some naïve enumeration algorithm, say), is, at least intuitively, inefficient.

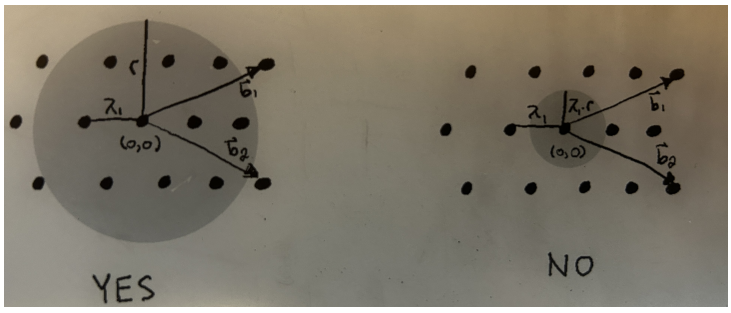
Now, given SVP, a natural thing to do is to contrive a *decision* version of SVP, so that it corresponds to a language. At the same time, it is also possible to define an "approximate" version of SVP, where instead of finding the shortest vector, all we require is to find a "short enough" vector. This version of SVP is as follows, where γ is a computable function like n^4 or 2^n, and n is the dimension of the lattice.

The γ-Approximate Decisional SVP (γ-GapSVP)

Input: a basis B of a lattice $\mathcal{L}$ and $r > 0$.

Output: $\begin{cases} \text{YES} & \text{if } \lambda_1(\mathcal{L}) \leq r \\ \text{NO} & \text{if } \lambda_1(\mathcal{L}) > \gamma r. \end{cases}$

Pictorially, two-dimensional γ-GapSVP looks as follows.



Exercise 21.4.

- *If* $\gamma' > \gamma$*, is* γ'-GapSVP *easier or harder than* γ-GapSVP*?*
- *Is* γ-GapSVP $\in$ NP*? Why or why not?*

In fact, γ-GapSVP $\in$ NP for all choices of $\gamma \geq 1$, but it is unknown if it is ever NP-complete.[10]

Now, the reason we've introduced these two problems is so that we can state the following two facts, one of which relates to the power of quantum computers, and the other of which relates to a classical public-key cryptosystem.

Fact 21.5. *An efficient (classical or quantum) algorithm for the non-abelian hidden subgroup problem implies an efficient (classical or quantum) algorithm for (a version of)* poly-GapSVP*.*[11]

In particular, the instance of the hidden subgroup problem (HSP) is over the *dihedral group*, which is the group of symmetries of regular polygons. Crucially, this group is not abelian, which, as we discussed last lecture, is interesting from a quantum computing point of view because, as far as we know, quantum computers cannot efficiently solve the non-abelian HSP. Thus, the above result suggests that quantum computers are equally bad at solving poly-GapSVP and other, related lattice problems. Given this, it is natural to devise cryptosystems who security guarantees stem from lattice problems, as this will plausibly make them quantum-secure. Along these lines, Oded Regev (along with many others) have proposed cryptosystems that do just this, and along the way establish theorems like the following.

Fact 21.6. *Breaking many lattice-based cryptosystems (e.g.,* Regev encryption*) is as hard as the worst case instance of* poly-GapSVP*. Therefore, it is plausible that these cryptosystems are quantum-secure.*

Hence, unlike the RSA and ElGamel cryptosystems, lattice-based cryptosystems do not get their security guarantees from *abelian* HSP-like problems, but, at least roughly speaking, from *non-abelian* HSP-like problems. This gives some evidence that in a post-quantum world, classical, lattice-based cryptosystems will be quantum-secure. Currently, this seems to be the most likely future, as the National Institute of Standards and Technology (NIST) has started to standardize some of these lattice-based protocols.

[10] That said, it is known to be NP-hard under so-called "randomized reductions".

[11] In particular, the version is called the *unique* SVP, which we will not discuss here as we have already ventured deep into a lattice rabbit hole. Nevertheless, if you are interested, this paper by Curtis Bright proves this reduction in full.

Lecture 22
Hamiltonian Simulation

Discussion 22.1. Discuss with your group what you took away from last time.

Last lecture, we discussed quantum and post-quantum cryptography. This discussion was motivated by the fact that Shor's algorithms for factoring and discrete log threaten current cryptosystems, such as the RSA cryptosystem. Therefore, it is both natural and important to consider new cryptosystems that are secure against quantum adversaries.

In this lecture, we will consider problems beyond factoring and discrete log (and other hidden subgroup problems) that quantum computers can solve efficiently. In particular, we will consider the problem of *Hamiltonian simulation*, which is just a fancy way of saying the problem of simulating quantum systems. Indeed, as you may recall from Lecture 1, this was Richard Feynman's original idea for what a quantum computer would be good for.[1]

22.1. Matrix Exponentials

Like the other lectures in this course, we will require some new mathematics to formally introduce the notion of a "Hamiltonian", and also to see how Hamiltonians relate to the unitary evolution of quantum systems that we are accustom to.[2] As we will see, the main mathematical workhorse of this lecture is the *matrix exponential*.

Definition 22.1. Let A be an $N \times N$ complex-valued matrix. The *matrix exponential of* A is the $N \times N$ complex-valued matrix

$$e^A := \sum_{k \geq 0} \frac{A^k}{k!},$$

[1] I recommend reading his seminal paper *Simulating Physics with Computers*, available here.

[2] Note, the term "Hamiltonian" is named after the 19th century physicist and mathematician William Hamilton, not the United States founding father Alexander Hamilton.

where $A^0 \coloneqq I_N$.

Here, the definition stems from the Taylor series for e^x,

$$e^x = \sum_{k \geq 0} \frac{x^k}{k!},$$

which is defined for all real x. Of course, in the case of the matrix exponential, it is not obvious that the series converges, but, in fact, it does for any square matrix A.[3]

Example 22.1. Let

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

Evidently,

$$C^2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I_2,$$

which implies that for all $k \geq 0$, $C^{2k} = I_2$ and $C^{2k+1} = C$. Consequently,[4]

$$\begin{aligned} e^C &= \sum_{k \geq 0} \frac{1}{k!} C^k \\ &= \sum_{k \geq 0} \frac{1}{(2k)!} C^{2k} + \sum_{k \geq 0} \frac{1}{(2k+1)!} C^{2k+1} \\ &= I_2 \sum_{k \geq 0} \frac{1}{(2k)!} + C \sum_{k \geq 0} \frac{1}{(2k+1)!} \\ &= I_2 \left(\frac{e + e^{-1}}{2} \right) + C \left(\frac{e - e^{-1}}{2} \right) \\ &= \frac{1}{2} \begin{pmatrix} e + e^{-1} & e - e^{-1} \\ e - e^{-1} & e + e^{-1} \end{pmatrix}. \end{aligned}$$

Exercise 22.1. *Let*

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \quad \textit{and} \quad B = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}.$$

[3] This follows from the observation that $\|A^k\|_{\text{op}} \leq \|A\|_{\text{op}}^k$ for all $k \geq 0$, from which it follows that e^A is a bounded operator, as the previous inequality implies $\|e^A\|_{\text{op}} \leq e^{\|A\|_{\text{op}}}$.

[4] Using the hyperbolic trigonometric functions $\cosh(x) = (e^x + e^{-x})/2$ and $\sinh(x) = (e^x - e^{-x})/2$, we can equivalently write e^C as

$$e^C = \begin{pmatrix} \cosh(1) & \sinh(1) \\ \sinh(1) & \cosh(1) \end{pmatrix}.$$

Prove that

$$e^A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad \textit{and} \quad e^B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

Now consider the following question.

Question 22.1. *Given $N \times N$ complex-valued matrices A and B, is $e^A e^B = e^{A+B}$?*

Nevertheless, there are several facts that are useful for approximating the matrix C such that $e^A e^B = e^C$.[5]

Fact 22.1. *If A and B are $N \times N$ complex-valued matrices that commute (meaning their* commutator *$[A, B] \coloneqq AB - BA = 0$), then $e^A e^B = e^{A+B}$.*

More generally, regardless of the value of $[A, B]$, we have the following fact, which can be found in any standard text on Lie theory.

Fact 22.2 (Lie Product Formula)**.** *If A and B are $N \times N$ complex-valued matrices, then*

$$e^{A+B} = \lim_{m \to \infty} \left(e^{A/m} e^{B/m} \right)^m.$$

More generally, if $A_1, \ldots, A_n$ are $N \times N$ complex-valued matrices, then

$$e^{A_1 + \cdots + A_n} = \lim_{m \to \infty} \left(e^{A_1/m} \cdots e^{A_n/m} \right)^m.$$

These identities will help in simulating certain compositions of quantum systems, as we will shortly discuss. First, however, we need to understand why matrix exponentials are even relevant in the context of quantum mechanics.

22.2. The Differential Form of Schrödinger's Equation

In physics, we are generally interested in how the state of a physical system (a block on a frictionless table, an electron in a magnetic field, a spaceship near a Schwarzschild black hole, etc.) evolves in time. In physics, then, we seek a means to describe how the state of a system is *changing* at any given time, so that, given some initial condition, we can use calculus to figure out the state at any other time. Indeed, this is precisely the content of Newton's second law, $F = ma$, which is a

[5] Although we will not explicitly use it here, among the most important results along these lines is the *Baker–Cambell–Hausdorff (BCH) Formula*, which gives an explicit formula for C in terms of the commutator of A and B, at least for "small" A and B. I recommend Brian Hall's book *Lie Groups, Lie Algebras, and Representations* if you are interested in this stuff.

second-order differential equation that, for any time t, can be integrated to solve for the velocity and position of a particle experiencing the force F.

Given the way we introduced quantum mechanics, however, we do not know how to do this. In particular, all we know is that, absent any measurements, if the state of a quantum system is $|\psi(t_1)\rangle$ at time t_1, and if the state at some other time $t_2 \neq t_1$ is $|\psi(t_2)\rangle$, then there must exist a unitary transformation U that relates them:

$$|\psi(t_2)\rangle = U|\psi(t_1)\rangle.$$

This, however, tells us nothing about the *rate* of change of $|\psi(t)\rangle$ with respect to t, nor does it tell us anything about what the unitary U is (and how, for example, it depends on t_1 and t_2, which it certainly must). Of course, at least up to now, this has not been a problem. In the context of quantum mechanics, and in particular of *simulating* quantum mechanics, however, these extra pieces of information are crucial. To mend this, we will now introduce the *differential* form of Schrödinger's equation, which is really *the* Schrödinger equation that Erwin Schrödinger originally postulated.

Postulate 22.1. Let S be a quantum system with Hilbert space $\mathbb{C}^N$, and let $|\psi(t)\rangle$ be the state of S at time t. Absent any measurements of the system, there exists a (possibly time-dependent) $N \times N$ hermitian operator $H(t)$ called a *Hamiltonian operator* (or just *Hamiltonian* for short) such that[6]

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

This is called the *time-dependent Schrödinger equation*.

Intuitively, this equation says that the Hamiltonian generates time evolution. In other words, changes in the state in time (the left hand side) are generated by the action of the Hamiltonian on the state (the right hand side). Below are two examples of Hamiltonians that appear in nature.

Example 22.2.

- Consider a particle of mass m moving in three dimensional space in a potential $V(\vec{x}, t)$. Then,
$$H = H(\vec{x}, t) = -\frac{1}{2m}\nabla^2 + V(\vec{x}, t)$$

[6]The physicist reading this might be concerned that I forgot $\hbar$, the reduced Plank constant, on the left side. Not at all. WLOG, we can assume we are working with units in which $\hbar = 1$.

Note that in this case, the Hamiltonian depends on the position $\vec{x}$ of the particle (in fact, the position *operator*), so Schrödinger's equation becomes a first-order *partial* differential equation. Note also that in this case, H is an infinite dimensional operator (as ∇ is an infinite dimensional operator, as is the position operator), which complicates the operator analysis (though physics students are undoubtedly familiar with this). That said, an appropriate discretization of space and time makes this Hamiltonian amenable to computational analysis.

- Suppose there is magnetic field along the z-axis that is incident on n qubits in a circle that are exhibiting nearest-neighbor interactions. This physical situation underlies the so-called *quantum Heisenberg model*, which is related to the famous *Ising model*. The Hamiltonian for this system is

$$H_{\text{QHM}} = -\frac{1}{2} \sum_{j \in \mathbb{Z}_n} \left(J_x X_j X_{j+1} + J_y Y_j Y_{j+1} + J_z Z_j Z_{j+1} + h Z_j \right).$$

  Here, h, J_x, J_y, and J_z are constants that specify the field strength and the nearest-neighbor interactions, and X_j is shorthand for the n-fold tensor product

$$X_j = \underbrace{I_2 \otimes \cdots \otimes I_2}_{j-1 \text{ times}} \otimes X \otimes \underbrace{I_2 \otimes \cdots \otimes I_2}_{n-j \text{ times}}$$

  (and similarly for Y_j and Z_j). Also note that here the sums in the indices of the terms in H_{QHM} are all modulo n. This is because we are supposing that the n qubits lie in a circle. (In physics, we would say that the quantum Heisenberg model has *periodic boundary conditions*.) Note that in this case, the Hamiltonian is independent of time.

To relate all of this to the unitary evolution of quantum states, note that given an initial condition, i.e., the quantum state $|\psi(t_1)\rangle$ at some time t_1, we can simply integrate Schrödinger's equation to find the quantum state $|\psi(t_2)\rangle$ at a different (though not necessarily later) time $t_2 \neq t_1$. The upshot of this is the following fact.

Fact 22.3. *Let S be a quantum system with Hilbert space $\mathbb{C}^N$, let $|\psi(t)\rangle$ be the state of S at time t, and suppose $|\psi(t_1)\rangle$ is known for some initial time t_1. Additionally, for all time t, define*

$$U(t_1, t) \coloneqq \exp\left(-i \int_{t_1}^{t} H(s)\ ds \right).$$

Then,

(i) $U(t_1, t) \in \mathrm{U}(N)$,

(ii) and $|\psi(t)\rangle = U(t_1, t)|\psi(t_1)\rangle$.

Therefore, by integrating the differential form of Schrödinger's equation, we recover the unitary evolution postulate of quantum mechanics that we are so accustom to. In particular, setting $t = t_2$, we find that

$$|\psi(t_2)\rangle = U(t_1, t_2)|\psi(t_1)\rangle.$$

What's more, however, is that this form tells us the exact unitary that relates two quantum states at two different times, and we see how that unitary depends on the time parameters.

Philosophically, this is quite interesting, because the preceding fact shows that the Schrödinger equation is *deterministic*, meaning that as long as we have some initial data (such as the state of the system at some time and the Hamiltonian of the system), then we can deduce, on pain of irrationality, the state at all later times. Of course, this assumes no measurements are occurring (whatever a "measurement" is), but nonetheless if you think that measurements are a figment of our misunderstanding of quantum mechanics (such as in the Many-Worlds interpretation), then it follows that quantum mechanics is a completely deterministic theory, despite being probabilistic in nature!

Now, getting back to the matter at hand, suppose that the Hamiltonian is independent of time. In this case, it is a straightforward consequence of Fact 22.3 that for all times t_1 and t_2,

$$U(t_1, t_2) = e^{-iH(t_2 - t_1)}.$$

This is the form of the unitary that we will assume in the next section, as it underlies many interesting physical systems (e.g., electrons in a constant magnetic field). Of course, in the context of quantum computing, it is very interesting to consider time-dependent Hamiltonians, as this underlies a model of quantum computation called *adiabatic quantum computing*, which I encourage you to read about.

Note, given the above expression for the unitary evolution induced by a time-independent Hamiltonian H, it evidently suffices to approximate the unitary $U(t_1, t_2)$ in the operator norm for all times t_1 and t_2 to simulate the evolution induced by H. Indeed, this is exactly how one simulates a quantum system on a quantum computer, as we will now discuss.

Given the above discussion, we see that to simulate a quantum system, it suffices to simulate the unitary generated by its Hamiltonian à la Fact 22.3. Formally, the definition of simulating a quantum system on a quantum computer is as follows.

Definition 22.2. Let S be an n-qubit quantum system with Hamiltonian H. We say S can be *efficiently simulated on a quantum computer* iff there exists a P-uniform family of n-qubit quantum circuits $Q = \{Q_\ell : \ell \in \mathbb{N}\}$ over a universal gate set $\mathcal{G}$ and a function $f : \mathbb{R} \times \mathbb{R} \to \mathbb{N}$ such that for all $t > 0$ and all $\epsilon > 0$:

(i) $f(t, \epsilon)$ is computable in polynomial time on a deterministic Turing machine,

(ii) the size of $Q_{f(t,\epsilon)}$ is $O(\text{poly}(n, t, \epsilon^{-1}))$,

(iii) and $\|Q_{f(t,\epsilon)} - e^{-iHt}\|_{\text{op}} < \epsilon$.

Here, we require the family to consist solely of n-qubit circuits because e^{-iHt} is an n-qubit unitary for all time t. Additionally, condition (i) is an additional uniformity constraint that enforces the generation of a given circuit is efficient with respect to the time and precision parameters, and (ii) is a natural requirement for the quantum computer to be efficient, and for the quantum computer to "know" the precision ϵ we seek in the approximation. (You should compare and contrast this definition with the definition of an efficient quantum computer, which we saw many lectures ago.) Finally, condition (iii) is the sense in which the circuit is actually simulating the Hamiltonian H, as it emulates the unitary operation (at time t) that H implements to within error ϵ in operator norm.

For the rest of this section, we will explore a handful of cases when quantum systems can be efficiently simulated, contingent on some other quantum system being efficiently simulable.

Exercise 22.2. *Let S be an n-qubit quantum system with Hamiltonian H, and suppose S can be efficiently simulated on a quantum computer. Argue that for all polynomials p, the n-qubit quantum system S' with Hamiltonian $p(n)H$ can also be efficiently simulated on a quantum computer.*

In words, this exercise shows that by re-parameterizing time by some polynomial factor, the efficiency of the simulation does not change. We will now consider a slightly more interesting type of simulation.

Claim 22.4. *Let S be an n-qubit quantum system with Hamiltonian H, and let $U \in \mathrm{U}(2^n)$. If S can be efficiently simulated on a quantum computer, then so can the n-qubit system S' with Hamiltonian $UHU^\dagger$.*

Proof. This follows from the fact that for any unitary $U \in \mathrm{U}(N)$ and any $N \times N$ complex-valued matrix A,[7]

$$e^{UAU^\dagger} = Ue^A U^\dagger.$$

Thus, to simulate $e^{-iUHU^\dagger t}$, it suffices to simulate U, $U^\dagger$, and e^{-iHt}. By assumption we can simulate e^{-iHt}, and by the fact that we have access to a universal gate set $\mathcal{G}$, we can simulate U to within error ϵ' using $O(\mathrm{poly}(\log 1/\epsilon'))$ gates by the Solovay-Kitaev theorem. Therefore, we can efficiently simulate the quantum system S' with Hamiltonian $e^{-iUHU^\dagger t}$ on a quantum computer. ■

Among the many physical configurations one might consider, perhaps the most natural is to take two quantum systems and put them together. Intuitively, one expects if both systems are independently simulable, then the combined system should be as well, assuming, of course, that they don't interact in a significant way. Indeed, this turns out to be true.

Claim 22.5. *Let S_1 and S_2 be n-qubit quantum systems with Hamiltonians H_1 and H_2, respectively. If both S_1 and S_2 can be efficiently simulated on a quantum computer, then so can the composite system $S_1 + S_2$ with Hamiltonian $H_1 + H_2$.*

Proof Idea. The key to this proof is the Lie product formula from Fact 22.2, which, in the case of Hamiltonian simulation, implies that

$$e^{-i(H_1+H_2)t} = \lim_{m\to\infty} \left(e^{-iH_1 t/m} e^{-iH_2 t/m}\right)^m.$$

Taking $m = O(\mathrm{poly}(n, t, \epsilon^{-1}))$ gives the polynomially-sized product approximation (called a *Trotterization*)

$$e^{-i(H_1+H_2)t} \approx \underbrace{\left(e^{-H_1 t/m} e^{-H_2 t/m}\right) \cdots \left(e^{-H_1 t/m} e^{-H_2 t/m}\right)}_{m \text{ product pairs}}.$$

In this case, since each unitary $e^{-H_1 t/m}$ and $e^{-H_2 t/m}$ can be simulated efficiently on a quantum computer (see Example 22.2), the whole product can also be simulated efficiently on a quantum computer.[8] ■

[7] It is a good exercise to prove this identity.

[8] For details on the error analysis and other approximations, see these notes by Andrew Childs.

A more general statement holds for the composition of many quantum systems.

Fact 22.6. *Let $S_1, \ldots, S_k$ be n-qubit quantum systems with Hamiltonians $H_1, \ldots, H_k$, respectively. If $S_1, \ldots, S_k$ can all be efficiently simulated on a quantum computer, then so can the composite system $S_1 + \cdots + S_k$ with Hamiltonian $H_1 + \cdots + H_k$.*

The proof of this fact is the same as the two system case, with the exception that one must use the generalized Lie product formula in Fact 22.2.

Another interesting fact is the following, which relates the simulability of two systems with Hamiltonians H_1 and H_2 to the simulability of the system whose Hamiltonian is essentially the *commutator* of H_1 and H_2.

Claim 22.7. *Let S_1 and S_2 be n-qubit quantum systems with Hamiltonians H_1 and H_2, respectively. If both S_1 and S_2 can be efficiently simulated on a quantum computer, then the system S' with Hamiltonian $i[H_A, H_B]$ can be efficiently simulated on a quantum computer.*

Proof Idea. The key to this proof is to use a version of the Lie product formula that holds for the commutator. In particular, it is a fact that

$$e^{[H_1,H_2]t} = \lim_{m\to\infty} \left(e^{-iH_1\sqrt{t/m}} e^{-iH_2\sqrt{t/m}} e^{iH_1\sqrt{t/m}} e^{iH_2\sqrt{t/m}} \right)^m .$$

While we will not prove this identity (since, like the Lie product formula, it requires group theory beyond the scope of this course), the idea of simulating S' is then the same as in the previous claims: for $m = O(\text{poly}(n, t, \epsilon^{-1}))$, we get a good approximation of $e^{[H_1,H_2]t}$ in terms of $e^{-iH_1\sqrt{t/m}}$ and $e^{-iH_2\sqrt{t/m}}$, each of which can be efficiently simulated. ■

Finally, we give what is one of the more important facts when it comes to simulating quantum systems with quantum computers. As we will see in the next section, this fact is also important for quantum algorithms that solve certain linear algebraic problems, such as finding the maximum eigenvalue of a Hermitian matrix.

Claim 22.8. *Let S be an n-qubit quantum system with a Hamiltonian H that contains at most $O(\text{poly}(\log n))$ many non-zero entries (when expressed in the computational basis).*[9] *Suppose, further, that for all $x \in \mathbb{Z}_n$, there is an efficient classical computer that outputs every $y \in \mathbb{Z}_n$ for which $\langle y|H|x\rangle \neq 0$ and, moreover, outputs the value of $\langle y|H|x\rangle$ for each such y (in this case, H is called* efficiently row computable*). Then, S can be efficiently simulated on a quantum computer.*

[9] Such a Hamiltonian is called *sparse*.

Proof Idea. If H is sparse, then it turns out that H can be decomposed into a sum of just $\ell = O(\text{poly}(\log n))$-many Hamiltonians $H_1, H_2, \ldots, H_\ell$ that mutually commute. By Fact 22.1, it follows that

$$\begin{aligned} e^{-iHt} &= e^{-i(H_1+\cdots+H_\ell)t} \\ &= e^{-iH_1 t} \cdots e^{-iH_\ell t}. \end{aligned}$$

One can then show that each of these exponentials can be well-approximated on the quantum computer, which then implies that the overall product can be too.[10] ■

22.4. Matrix Encodings*

In the last section, we stated several simulation results in the context of simulating a quantum system with a particular Hamiltonian H. One can, however, give this a more computational interpretation and just recognize that, independent of the quantum system that the Hamiltonian H represents, at the end of the day H is just a Hermitian matrix, and it is natural to ask questions about that Hermitian matrix. For example, given a Hermitian matrix H, what is its smallest or largest eigenvalue? Or, given H and a vector $\vec{b}$, find $\vec{x}$ such that $H\vec{x} = \vec{b}$. Indeed, the previous discussion allows us to solve such linear algebraic questions as well, thanks to an idea called *matrix encoding*.

Definition 22.3. Let H be a Hermitian matrix. The *matrix encoding of H into a unitary operator* is the matrix exponential e^{-iHt}, where t is a parameter.

Of course, nothing has changed from the previous discussion, except our interpretation of what H represents. In particular, here we do not think of H as the Hamiltonian of some quantum system, but just as some matrix whose properties we want to investigate. What matrix encoding allows us to do is to represent the Hermitian matrix H in a quantum-interpretable way.

Given the discussion so far, you may think that requiring H to be a Hermitian matrix is rather stringent. In fact, however, it is not, as any square matrix admits a Hermitian form.

Exercise 22.3. *Let A be an $N \times N$ complex-valued matrix. Prove that the matrix*

$$H_A = \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}$$

is Hermitian. Conclude that $e^{-iH_A t}$ is unitary.

[10] See Andrew Childs' notes here for the full proof.

This is the generic trick that one must first employ in order for the following two facts to hold for general complex-valued matrices.

Fact 22.9 (Nghiem–Wei Algorithm)**.** *Let A be an $N \times N$, complex-valued, sparse matrix. Then, there exists a quantum algorithm that finds the largest eigenvalue of A (in magnitude) to within error ϵ in time $O(\sqrt{N} \log N/\epsilon^4)$. In particular cases, this can be reduced to just $O(\log N/\epsilon^4)$.*

Since the largest eigenvalue of A^{-1} is the reciprocal of the smallest eigenvalue of A, it follows that the above algorithm can also find the smallest eigenvalue of A (provided you know A^{-1}). In addition, the above algorithm is remarkable in that in certain cases, it need not read every entry of A. Thus, the above algorithm can offer an exponential speedup to finding the largest or smallest eigenvalue of a matrix A.[11] We note that this also has applications in quantum mechanics, in which often one is interested in finding the smallest eigenvalue of a Hamiltonian, which corresponds to the lowest energy state of the quantum system.

Another interesting quantum algorithm pertains to the linear algebraic problem of solving a system of linear equations. This is obviously important for a myriad of tasks, but in particular it is important for many machine learning tasks.

Fact 22.10 (Harrow–Hassidim–Lloyd (HHL) Algorithm)**.** *Let A be an $N \times N$ complex-valued matrix and let $\vec{b} \in \mathbb{R}^N$ be a unit vector for which there exists a unit vector $\vec{x} \in \mathbb{R}^N$ such that $A\vec{x} = \vec{b}$. Supposing that $|\vec{b}\rangle$ can be realized efficiently and that A is sparse and efficiently row computable, there is a quantum algorithm that runs in time $O(\log N/\epsilon)$ that computes $\langle \vec{x}|A|\vec{x}\rangle$ to within error ϵ.*

Of course, this doesn't *find* the vector $\vec{x}$, but in many applications the matrix element $\langle \vec{x}|A|\vec{x}\rangle$ is all that matters. Indeed, a myriad of other quantum algorithms utilize the HHL algorithm as a subroutine (or a modification thereof), including the Nghiem–Wei algorithm from before, but also quantum algorithms for differential equation solving, least-squares fitting, and quantum chemistry. As with other topics in this course, Hamiltonian simulation and matrix encodings constitute a vast subfield of quantum computation, which you are encouraged to read about on your own time if it interests you.

22.5. BQP-Completeness*

To return to Feynman's original idea that quantum computers ought to be good at simulating quantum dynamics, here we will sketch how simulating certain

[11] For more details, the reader is referred to Nghiem and Wei's paper here.

Hamiltonians (and thus how simulating certain quantum systems) relates to the complexity class BQP, which, you'll recall, is the set of decision problems that can be decided on an efficient quantum computer.

Interestingly, it turns out that simulating certain Hamiltonians is BQP-*complete*. This implies that such Hamiltonian simulation problems are the *hardest* problem in BQP, and thus that BPP = BQP if and only if classical computers can simulate quantum systems just as well as quantum computers can.[12] Note that this is in the same spirit of NP-complete problems, which constitute the *hardest* problems in NP (e.g., 3-SAT), and which are such that P = NP if and only if polynomial time, deterministic, classical computers can solve them. In the case of justifying that certain instances of Hamiltonian simulation are BQP-complete, one must show two things.

First, one must show that every unitary evolution is an example of Hamiltonian simulation. Feynman was the first to show this formally, and we will not go through the details. Intuitively, however, this makes sense in the context of *Stone's Theorem*, which says that for all one-parameter families of unitaries $U(t)$ that obey reasonable constraints, there exists a (unique!) Hermitian matrix H such that $U(t) = e^{-iHt}$ for all times t. Thus, since every language $L \in \mathsf{BQP}$ is decided by a sequence of unitary operators, every such unitary can be represented as a Hamiltonian simulation. Thus, at least intuitively speaking, it is plain that the generic problem of Hamiltonian simulation is BQP-*hard*, i.e., that Hamiltonian simulation is at least as hard as every problem in BQP.

Second, one must show that certain instances of Hamiltonian simulation that are BQP-hard are also in BQP. Indeed, this turns out to be true (see, for example, the problem of simulating *local* Hamiltonians, which is in BQP and which is the Hamiltonian simulation problem with just local interactions of the underlying quantum systems).

Together, these results demonstrate that certain instances of Hamiltonian simulation are BQP-complete, and thus represent the *hardest* problems in BQP. In other words, Feynman's intuition is proved right: the problem of simulating quantum systems *is* the problem that quantum computers are good at. If you want to read more about this, I recommend Nielsen and Chuang's textbook, as well as John Preskill's lecture notes, which are available online.

[12] Unfortunately, this is not quite right. Since BQP is a so-called *semantic* complexity class, it is actually not known to have complete problems. The exact class we are talking about, therefore, is a BQP-like class called PromiseBQP that does have complete problems (and is an example of a *syntactic* complexity class).

Lecture 23
Post-Selection and Quantum Advantage

Discussion 23.1. Discuss with your group what you took away from last time.

Last time, we discussed how quantum computers can simulate quantum systems, which was Feynman's original motivation for introducing the notion of a quantum computer. The previous lecture goes beyond the use of quantum computers for solving hidden subgroup problems, such as factoring and discrete log, and afforded additional, formal evidence that efficient quantum computers indeed offer a computational advantage over efficient classical computers.

In this last lecture, we will discuss another type of formal evidence that efficient quantum computers can outperform all efficient classical computers. The argument is complexity-theoretic in nature, and it gives rise to a conditional adynaton of the form "if quantum computers do not afford a particular type of computational advantage, then pigs can fly".

23.1. Relativized Complexity Classes

To begin, recall the basis of the oracle complexity paradigm, which we introduced in the lecture on Grover's algorithm. In this paradigm, it is assumed that one has access to a subroutine that computes some difficult problem in just a single computational step. Given this, it is natural to consider the complexity class that corresponds to P, but when the underlying determinitic, polynomial time machine has access to a potentially powerful oracle. Such classes are called *relativized complexity classes*, and we will now define a particularly important example.

Definition 23.1.

- Let $L \subseteq \{0,1\}^*$ be a language. The complexity class P^L consists of all languages $L' \subseteq \{0,1\}^*$ for which there is a deterministic, polynomial time Turing machine

M with oracle access to the indicator function of L, namely the function

$$\begin{aligned}\chi_L : \{0,1\}^* &\to \{0,1\} \\ x &\mapsto \begin{cases} 1 & \text{if } x \in L \\ 0 & \text{if } x \notin L, \end{cases}\end{aligned}$$

such that for all $x \in \{0,1\}^*$, $M(x) = \chi_{L'}(x)$.

- Let C be a complexity class. The complexity class P^{C} is the union of the classes P^L over all $L \in \mathsf{C}$, i.e.,

$$\mathsf{P}^{\mathsf{C}} := \bigcup_{L \in \mathsf{C}} \mathsf{P}^L.$$

  Therefore, $L' \in \mathsf{P}^{\mathsf{C}}$ iff there exists $L \in \mathsf{C}$ such that $L' \in \mathsf{P}^L$.

As stated in the introduction of this section, this is how we formally talk about the problems that can be decided by computational devices with access to an oracle.

Exercise 23.1.

1. *Argue that if* $L \in \mathsf{P}$*, then* $\mathsf{P}^L = \mathsf{P}$.
2. *Argue that* $\mathsf{P}^{\mathsf{P}} = \mathsf{P}$.[1]
3. *Is* $\mathsf{NP} \subseteq \mathsf{P}^{\mathsf{NP}}$*?*
4. *Is* $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{NP}$*?*

In fact, this last question is an open question in computational complexity theory.

23.2. The Polynomial Hierarchy

We will now introduce the main complexity class that underlies the "pigs can fly" part of the quantum advantage argument we will soon give. As we will see, this class, which is called the *the polynomial hierarchy* and is denoted by PH, is defined in such a way that it naturally generalizes the famous P versus NP question.

[1] As an aside, the fact that $\mathsf{P}^{\mathsf{P}} = \mathsf{P}$ is related to Scott Aaronson and Dave Bacon's definition of a "physical" complexity class. Their idea is that any complexity class that is "physical", or rather that captures "physically realistic problems", "must have a sensible notion of subroutines and recursion, and that hooking up a machine in your class to a subroutine also in the class must *not* give you a new, more powerful class." This quote is from Aaronson's blog here.

Definition 23.2.

- For all $k \in \mathbb{N}$, the class $\Sigma_k\mathsf{P}$ consists of the languages $L \subseteq \{0,1\}^*$ for which there exists a deterministic, polynomial time Turing machine M and polynomials $p_1, p_2, \ldots, p_k$ such that for all $x \in \{0,1\}^*$, $x \in L$ iff[2]

  $$\left(\exists y_1 \in \{0,1\}^{p_1(|x|)}\right)\left(\forall y_2 \in \{0,1\}^{p_2(|x|)}\right)\ldots\left(Q_k y_k \in \{0,1\}^{p_k(|x|)}\right) M(x, y_1, \ldots, y_k) = 1,$$

  where

  $$Q_k = \begin{cases} \exists & \text{if } k \text{ is odd} \\ \forall & \text{if } k \text{ is even.} \end{cases}$$

- The *polynomial hierarchy* is then the class

  $$\mathsf{PH} \coloneqq \bigcup_{k \geq 0} \Sigma_k\mathsf{P}.$$

The polynomial hierarchy is an important complexity class that deserves several lectures dedicated to it. Of course, we will not have time to do that here, so the reader is instead referred to any standard text on computational complexity theory to learn more about it.[3] That said, one useful way of thinking about it is in the context of games. In this setting, languages in $\Sigma_k\mathsf{P}$ correspond to two-player games for which there is a winning strategy in $k/2$ rounds for the first player. To see this, we interpret the quantifiers by asking whether there exists a move y_1 for player one such that no matter what move y_2 player two makes, there exists a move y_3 for player one, and so on for $k/2$ rounds, such that player one wins.[4]

We now present a handful of important facts about the polynomial hierarchy, which demonstrate its relationship to P and NP.

Exercise 23.2. *Prove that $\Sigma_0\mathsf{P} = \mathsf{P}$, where $\Sigma_0\mathsf{P}$ is $\Sigma_k\mathsf{P}$ but with no quantifiers.*

Therefore, $\mathsf{P} \subseteq \mathsf{PH}$. Importantly, $\mathsf{NP} \subseteq \mathsf{PH}$ as well.

Claim 23.1. $\Sigma_1\mathsf{P} = \mathsf{NP}$. *Therefore,* $\mathsf{NP} \subseteq \mathsf{PH}$.

Proof.* By definition, $L \in \Sigma_1\mathsf{P}$ iff there exists a deterministic, polynomial time Turing machine M and a polynomial p such that for all $x \in \{0,1\}^*$,

$$x \in L \iff \left(\exists y \in \{0,1\}^{p(|x|)}\right) M(x, y) = 1.$$

This is precisely the definition of NP. Therefore, $\mathsf{NP} = \Sigma_1\mathsf{P} \subseteq \mathsf{PH}$, as desired. ■

[2] Some people denote $\Sigma_k\mathsf{P}$ with P in the exponent, namely, as Σ_k^{P}. However, I find this confusing, especially given our preceding discussion about relativized complexity classes.

[3] See, for example, Arora and Barak's textbook *Computational Complexity: A Modern Approach*.

[4] There is an analogous class called $\Pi_k\mathsf{P}$ in which player two wins in this interpretation.

An immediate corollary of the above two results is that the P versus NP question is equivalent to a particular statement about the $\Sigma_k\mathsf{P}$ classes that define PH.

Corollary 23.2. $\Sigma_0\mathsf{P} = \Sigma_1\mathsf{P}$ *iff* $\mathsf{P} = \mathsf{NP}$.

Of course, this only concerns $\Sigma_k\mathsf{P}$ for $k \in \{0, 1\}$. One can therefore generalize the P versus NP problem by considering larger values of k.

Open Problem 23.3 (Generalization of P versus NP)**.** *Does there exist $k \neq \ell$ for which $\Sigma_k\mathsf{P} = \Sigma_\ell\mathsf{P}$?*

Interestingly, if this is true, then the polynomial hierarchy "collapses" to a considerably smaller class.

Fact 23.4 (Polynomial Hierarchy Collapse)**.** *Let $k < \ell$. Then, $\Sigma_k\mathsf{P} = \Sigma_\ell\mathsf{P}$ iff $\mathsf{PH} = \Sigma_k\mathsf{P}$, i.e., iff* the polynomial hierarchy collapses to its kth level.[5]

Most in the computer science community regard the collapse of the polynomial hierarchy to be extremely unlikely, tantamount (though not necessarily as strong) as the statement that $\mathsf{P} = \mathsf{NP}$ (which itself collapses the polynomial hierarchy to the first level). Instead, most think that $\Sigma_k\mathsf{P} \subsetneq \mathsf{PH}$ for all $k \in \mathbb{N}$, which is to say that most think that the polynomial hierarchy is *infinite*. For this reason, if we could prove a mathematical statement of the form "if efficient classical computers can simulate efficient quantum computers, then the polynomial hierarchy collapses", we would obtain considerably strong evidence that, in fact, efficient classical computers cannot simulate efficient quantum computers, because there is considerably strong evidence that the polynomial hierarchy does not collapse. Of course, this is the goal of this lecture, but to do this properly requires a few more definitions and results.

23.3. Post-Selection and Post-Selected Complexity Classes

In this section, we will introduce two new complexity classes, which correspond to non-realistic versions of BPP and BQP, but which turn out to be quite interesting from a complexity theory point of view. In particular, both of these classes employ the non-physical ability to *post-select*, which is roughly the ability to force extremely unlikely events and to then condition on them happening. For example, the following

[5] For those who like pertinent, nerdy humor, you'll enjoy this essay by Scott Aaronson. Also, if you'd like to see a proof of this claim, then see Arora and Barak's textbook *Computational Complexity: A Modern Approach*.

map is a post-selection map on a qubit state:

$$\frac{1}{\sqrt{2^{1000}}}|0\rangle + \sqrt{1 - \frac{1}{2^{1000}}}|1\rangle \mapsto |0\rangle.$$

In words, here post-selection allows us to change the state to any state with a non-zero amplitude in just a single step, no matter how unlikely the post-selected state is. Of course, such an evolution is not unitary, which is why this is not realistic.[6] Nevertheless, as we will shortly see, this "magical power" is very interesting from a theoretical point of view. We will now define both the classical and quantum classes that correspond to "efficient computers that can post-select".

Definition 23.3.

- The class $\mathsf{PostBPP}$ (a.k.a. $\mathsf{BPP}_{\text{path}}$) consists of the languages L for which there is an efficient probabilistic classical computer $(C, \mathcal{B}, s)$ such that for all $x \in \{0,1\}^*$,
  (i) the probability that the second bit outputted by $(C, \mathcal{B}, s)$ is 0 is non-zero, i.e.,
  $$\Pr\left[C(x)_1 = 0\right] > 0,$$
  (ii) and, conditioned on the second bit outputted by $(C, \mathcal{B}, s)$ equalling 0, the probability that the first bit outputted by $(C, \mathcal{B}, s)$ is $\chi_L(x)$ is at least $1/2 + \delta$ for some fixed $\delta \in (0, 1/2]$, i.e.,
  $$\Pr\left[C(x)_0 = \chi_L(x) \mid C(x)_1 = 0\right] \geq \frac{1}{2} + \delta.$$
- Similarly, the class $\mathsf{PostBQP}$ consists of the languages L for which there is an efficient quantum computer $(Q, \mathcal{G}, a)$ over a universal gate set $\mathcal{G}$ such that for all $x \in \{0,1\}^*$,[7]
  (i) the probability that the second qubit outputted by $(Q, \mathcal{G}, a)$ is $|0\rangle$ is non-zero, i.e.,
  $$\Pr\left[Q(x)_1 = |0\rangle\right] > 0,$$
  (ii) and, conditioned on the second qubit outputted by $(Q, \mathcal{G}, a)$ equalling $|0\rangle$, the probability that the first qubit outputted by $(Q, \mathcal{G}, a)$ is $|\chi_L(x)\rangle$ is at least $1/2 + \delta$ for some fixed $\delta \in (0, 1/2]$, i.e.,
  $$\Pr\left[Q(x)_0 = |\chi_L(x)\rangle \mid Q(x)_1 = |0\rangle\right] \geq \frac{1}{2} + \delta.$$

[6] For more on this, I will plug my paper here.

[7] The fact that $\mathcal{G}$ is universal makes this class independent of the underlying gate set, in the same way that BQP over a universal gate set is actually independent of the underlying gate set.

It is easy to reason that $\mathsf{BPP} \subseteq \mathsf{PostBPP}$, that $\mathsf{BQP} \subseteq \mathsf{PostBQP}$, and that $\mathsf{PostBPP} \subseteq \mathsf{PostBQP}$. However, it is unknown how, for example, BQP and $\mathsf{PostBPP}$ relate, and how $\mathsf{PostBPP}$ and $\mathsf{PostBQP}$ relate. That said, there are several results that inform this last question, which relate to the polynomial hierarchy. The following two theorems give some idea of the power of these post-selected classes.

Theorem 23.5 (Aaronson–Toda)**.** $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{PostBQP}}$.[8]

Theorem 23.6 (Han–Hemaspaandra–Thierauf)**.** $\mathsf{P}^{\mathsf{PostBPP}} \subseteq \Sigma_3\mathsf{P}$.[9]

Altogether, it seems that $\mathsf{PostBQP}$ is considerably more powerful than $\mathsf{PostBPP}$, because the whole polynomial hierarchy is contained in the class $\mathsf{P}^{\mathsf{PostBQP}}$, but not $\mathsf{P}^{\mathsf{PostBPP}}$, unless the polynomial hierarchy collapses. The formal statement of this is as follows.

Corollary 23.7. *If* $\mathsf{PostBPP} = \mathsf{PostBQP}$*, then* $\mathsf{PH} \subseteq \Sigma_3\mathsf{P}$*, i.e., the polynomial hierarchy collapses to its third level.*

Proof. If $\mathsf{PostBPP} = \mathsf{PostBQP}$, then

$$\begin{aligned} \mathsf{PH} &\subseteq \mathsf{P}^{\mathsf{PostBQP}} && \text{(Theorem 23.5)} \\ &= \mathsf{P}^{\mathsf{PostBPP}} && \text{(assumption)} \\ &\subseteq \Sigma_3\mathsf{P} && \text{(Theorem 23.6).} \end{aligned}$$

In other words, if $\mathsf{PostBPP} = \mathsf{PostBQP}$, then the polynomial hierarchy collapses to its third level. ■

Since the collapse of the polynomial hierarchy to *any* level is unlikely, this gives strong evidence that $\mathsf{PostBPP} \neq \mathsf{PostBQP}$. In the next section, we will exploit the above result to show that efficient classical computers cannot simulate efficient quantum computers in a particular sense unless the polynomial hierarchy collapses.

23.4. Weak Multiplicative Simulations and PH Collapse

In general, one says that "efficient quantum computers afford a quantum advantage iff there exists a computational task that no efficient classical computer can do."

[8] Here we have combined two independent results. Toda proved that $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{PP}}$, where PP is a BPP-like class that you should look up, and Aaronson proved that $\mathsf{PP} = \mathsf{PostBQP}$. Together, therefore, their results imply $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{PostBQP}}$, as stated.

[9] Here we have also combined two independent results. Han, Hemaspaandra, and Thierauf proved that $\mathsf{PostBPP} \subseteq \mathsf{P}^{\Sigma_2\mathsf{P}}$, and it is a relatively straightforward exercise to show that $\mathsf{P}^{\mathsf{P}^{\Sigma_2\mathsf{P}}} \subseteq \Sigma_3\mathsf{P}$. Together, therefore, these results imply $\mathsf{P}^{\mathsf{PostBPP}} \subseteq \Sigma_3\mathsf{P}$, as stated.

In this section, we will explore one particular type of computational task (called a *weak multiplicative simulation*) that does indeed offer a computational advantage. That said, what we discuss here by no means proves, for example, that $\mathsf{BPP} \neq \mathsf{BQP}$, or anything morally equivalent. Nevertheless, what we do get is additional formal evidence that efficient quantum computers are strictly more powerful than efficient classical computers.

Definition 23.4. Let $\epsilon \geq 0$, let $(Q, \mathcal{G}, a)$ be an efficient quantum computer, and let $(C, \mathcal{B}, s)$ be an efficient classical computer. We say that $(C, \mathcal{B}, s)$ *weakly*[10] *simulates* $(Q, \mathcal{G}, a)$ *to within multiplicative error* ϵ iff for all $x \in \{0,1\}^*$ and all strings y in the output of Q on inputs of size $|x|$,

$$\frac{1}{1+\epsilon} \Pr\left[Q(x) = |y\rangle\right] \leq \Pr\left[C(x) = y\right] \leq (1+\epsilon) \Pr\left[Q(x) = |y\rangle\right].$$

Since for small $\epsilon \geq 0$, $1/(1+\epsilon) \approx 1 - \epsilon$, an efficient classical computer weakly simulates an efficient quantum computer to within multiplicative error ϵ iff the output distribution of the classical computer is within a multiplicative factor of ϵ of the output distribution of the quantum computer, i.e.,

$$\left|\Pr\left[C(x) = y\right] - \Pr\left[Q(x) = |y\rangle\right]\right| \leq \epsilon \Pr\left[Q(x) = |y\rangle\right].$$

Right off the bat, we note that this computational task is quite difficult. The reason is because if the quantum computer has two likely outputs with probability $1/2 - 1/2^n$ and two unlikely outputs with probability $1/2^n$, then insisting that the classical computer simulates the quantum computer to within small multiplicative error means that the classical computer must approximate the two unlikely events essentially just as well as it approximates the two likely events. However, it should be difficult to actually differentiate between the correct distribution underlying the quantum computer and the distribution where the two likely events have probability $1/2$ each. For this reason, in the next section we mention a different notion of simulation that is more realistic and is more experimentally viable. Nevertheless, here we will charge ahead to see that this (rather restrictive) notion of simulation affords a quantum advantage, assuming that the polynomial hierarchy is infinite.

Claim 23.8. *If for all efficient quantum computers $(Q, \mathcal{G}, a)$ there exists an efficient probabilistic classical computer $(C, \mathcal{B}, s)$ that weakly simulates $(Q, \mathcal{G}, a)$ to within multiplicative error $\epsilon < \sqrt{2} - 1 \approx 0.414$, then the polynomial hierarchy collapses to its third level.*

[10] Here the word "weak" refers to the fact that the classical computer is only required to approximately *sample* from the distribution of the quantum computer, not the much stronger notion of computing the probability that the quantum computer outputs a particular string.

Proof.* By assumption, there is an efficient, probabilistic classical computer $(C, \mathcal{B}, s)$ that simulates $(Q, \mathcal{G}, a)$ to within multiplicative error $\epsilon < \sqrt{2} - 1$. Therefore, for all $x \in \{0,1\}^*$ and all strings y in the output on inputs of size $|x|$,

$$\frac{1}{1+\epsilon} \Pr[C(x) = y] \leq \Pr[Q(x) = |y\rangle] \leq (1+\epsilon) \Pr[C(x) = y].$$

Given this, let $L \in \mathsf{PostBQP}$. Then, by the definition of conditional probability and the weak multiplicative simulation assumption from above,

$$\begin{aligned}
\Pr[Q(x)_0 = |\chi_L(x)\rangle \mid Q(x)_1 = |0\rangle] &= \frac{\Pr\left[Q(x) = |\chi_L(x)\rangle|0\rangle\right]}{\Pr\left[Q(x)_1 = |0\rangle\right]} \\
&\leq \frac{(1+\epsilon) \Pr\left[C(x) = \chi_L(x).0\right]}{\Pr\left[C(x)_1 = 0\right]/(1+\epsilon)} \\
&= (1+\epsilon)^2 \Pr[C(x)_0 = \chi_L(x) \mid C(x)_1 = 0],
\end{aligned}$$

Since for all $x \in \{0,1\}^*$ and any fixed $\delta \in (0, 1/2]$,

$$\Pr[Q(x)_0 = |\chi_L(x)\rangle \mid Q(x)_1 = |0\rangle] \geq \frac{1}{2} + \delta,$$

it holds that

$$\Pr[C(x)_0 = \chi_L(x) \mid C(x)_1 = 0] \geq \frac{1}{(1+\epsilon)^2} \left(\frac{1}{2} + \delta\right) = \frac{1}{2} \cdot \frac{1+2\delta}{(1+\epsilon)^2}.$$

Consequently, $(C, \mathcal{B}, s)$ decides L in the sense of $\mathsf{PostBPP}$ provided $(1+\epsilon)^2 < 1+2\delta$. Since δ can be any value satisfying $0 < \delta < 1/2$, it suffices for $0 \leq \epsilon < \sqrt{2} - 1$ to ensure $L \in \mathsf{PostBPP}$. In this case, it follows that $\mathsf{PostBPP} = \mathsf{PostBQP}$, and the polynomial hierarchy collapses to its third level à la Corollary 23.7. ■

23.5. Weak Additive Simulations and PH Collapse*

In the last section, we mentioned that the notion of a weak multiplicative simulation is rather restrictive, because such a simulation requires distinguishing between distributions that are exponentially close in total variation distance (e.g., $1/2 - 1/2^n$ and $1/2$). Additionally, it is known that in the presence of errors, a quantum computer cannot even weakly simulate itself to within small multiplicative error, so of course it is a strong restriction to require a *classical* computer to do something that no "realistic" quantum computer can do (i.e., a quantum computer with

non-zero errors). Because of this, at least from an experimental point of view, a better notion of simulation is that of a *weak additive simulation*. In this case, it is merely required that on the average (as opposed to on every valuation) the classical computer samples the output distribution of the quantum computer to within some error $\epsilon \geq 0$. We put this more formally as follows.

Definition 23.5. Let $\epsilon \geq 0$, let $(Q, \mathcal{G}, a)$ be an efficient quantum computer, and let $(C, \mathcal{B}, s)$ be an efficient classical computer. We say that $(C, \mathcal{B}, s)$ *weakly simulates* $(Q, \mathcal{G}, a)$ *to within additive error* ϵ iff for all $n \in \mathbb{N}$, all $x \in \{0,1\}^n$, and all strings y in the output of Q on inputs of size $|x| = n$,

$$\frac{1}{2} \sum_{x \in \{0,1\}^n} |\Pr[Q(x) = |y\rangle] - \Pr[C(x) = y]| \leq \epsilon.$$

In other words, $(C, \mathcal{B}, s)$ weakly simulates $(Q, \mathcal{G}, a)$ to within additive error ϵ iff the output distributions of $(C, \mathcal{B}, s)$ and $(Q, \mathcal{G}, a)$ are within ϵ in total variation distance.

For small $\epsilon \geq 0$, it is a simple exercise to show that if $(C, \mathcal{B}, s)$ weakly simulates $(Q, \mathcal{G}, a)$ to within multiplicative error ϵ, then $(C, \mathcal{B}, s)$ weakly simulates $(Q, \mathcal{G}, a)$ to within additive error 2ϵ.

Exercise 23.3. *Prove this claim.*

Indeed, thanks to the so-called *Threshold Theorem for Quantum Computation*,[11] quantum error correction only guarantees correctness up to *additive error* between the target and actualized output distributions of an error-corrected quantum computer. Therefore, unlike multiplicative error, with sufficient error correction, imperfect quantum computers can weakly simulate themselves to within small additive error ϵ. For this reason, a more appropriate type of simulation of quantum computers by classical computers is the weak additive type. Unfortunately, this type of simulation does not immediately afford the argument that we saw for weak multiplicative simulation, in which a "good enough" additive simulation implies polynomial hierarchy collapse.

That said, there is a general argument to get a statement of the form "if for all efficient quantum computers there is an efficient classical computer that can weakly simulate it to within small additive error, then the polynomial hierarchy collapses," however, this argument (as far as I know) is always conditional on additional, unproven statements.

[11] See, for example, Daniel Gottesman's book *Surviving as a Quantum Computer in a Classical World*, available online here.

The basic idea to get a statement like this to hold is to note that it suffices to "upgrade" a weak additive simulation to a sufficiently strong weak multiplicative simulation, so that the weak multiplicative simulation argument from before goes through. In general, there are two components to such an upgrade.

The first component is *anti-concentration*. Anti-concentration mends the problem that in any weak additive simulation, while we are guaranteed that

$$\sum_{x \in \{0,1\}^n} |\Pr[Q(x) = |y\rangle] - \Pr[C(x) = y]| \leq 2\epsilon,$$

we have no knowledge of how small any particular difference

$$|\Pr[Q(x) = |y\rangle] - \Pr[C(x) = y]|$$

is, except, of course, that it must be less than 2ϵ. In particular, if $\Pr[Q(x) = |y\rangle]$ is much smaller than ϵ, then $\Pr[C(x) = y]$ is free to be much larger than ϵ. This is unlike weak multiplicative simulations, where it is guaranteed that each difference is small. An anti-concentration bound fixes this difficulty, but almost always at the expense of an unproven conjecture.

The second component is a sort of *worst-to-average case reduction*, which is similar in spirit to the worst-to-average case reductions that we briefly mentioned in the lecture on quantum and post-quantum cryptography. Here, however, the idea is that we want our efficient quantum computer to be such that the existence of any efficient classical computer that can *strongly* simulate it (i.e., compute the probabilities that the quantum computer outputs a particular string) implies the existence of a classical computer that can solve any problem in a complexity class $\sharp\mathsf{P}$, which is related to the class PP that was mentioned in a footnote above. Again, showing this generally requires another unproven conjecture.

Nevertheless, it turns out that with these two components, it is possible to upgrade a weak additive simulation to a weak multiplicative simulation, from which the preceding argument that the polynomial hierarchy collapses goes through. For an excellent survey of the details of this technique, I recommend this article by Hangleiter et al. Interestingly, by employing some new ideas, one can probably extend this approach to address what is arguably the most important question in quantum computation, as the following exercise shows.

Exercise 23.4. *Using these and new ideas, prove unconditionally that* $\mathsf{BPP} \neq \mathsf{BQP}$.

Thank you for the great semester!