

Lecture 1, Tues Jan 17: Course Intro,

Church-Turing Thesis

- Quantum Information Science is an inherently interdisciplinary field (Physics, CS, Math, Engineering, Philosophy)
- It's about clarifying the workings of quantum mechanics.
 - We use it to ask questions about what you can and can't do with quantum mechanics
 - It can help us better understand the nature of quantum mechanics itself.
- Professor Aaronson is very much on the theoretical end of research.
 - Theorists inform what experimentalists make, which in turn informs theorists' queries

Today we'll articulate several "self-evident" statements about the physical world. We'll then see that quantum mechanics leaves some of these statements in place, but overturns others--with the distinctions between the statements it upholds and the ones it overturns often extremely subtle! To start with...

Probability ($P \in [0,1]$) is the standard way of representing uncertainty in the world.

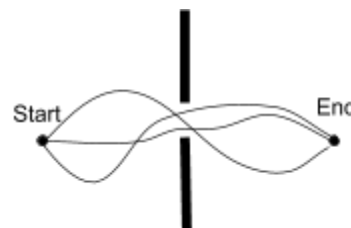
Probabilities have to follow certain obvious axioms like:

$$P_1 + \dots + P_n = 1 \quad \text{mutually exclusive exhaustive possibilities sum to 1}$$
$$P_i \geq 0$$

As an aside:

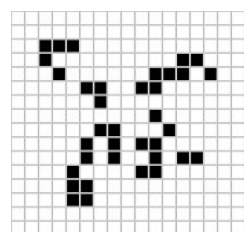
There's a view that "probabilities are all in our heads." Which is to say that if we knew everything about the universe (let's say position/velocity of all atoms in the solar system) that we could just crunch the equations and see that things either happen or they don't.

Let's say we have two points separated by a barrier with an open slit, and we want to measure the probability that a particle goes from one point to the other. It seems obviously true that increasing the number of paths (say, by opening another slit) should increase the likelihood that it will reach the other end.



We refer to this property by saying that probabilities are *monotone*.

Locality is the idea that things can only propagate through the structure of the universe at a certain speed.



When we update the state of a little patch of space, it should only require knowledge of a little neighborhood around it. Conway's Game Of Life (left) is an apt comparison: things you do to the system *can* affect it, but they propagate only at a certain speed.

Einstein's Theory of Relativity explains that a bunch of known physics things are a direct result of light's finite speed. Anything traveling past the speed of light would be tantamount to travelling back in time.

Local Realism says that any instantaneous update in knowledge about far away events can be explained by correlation of random variables.

For example, if you read your newspaper in Austin, you can instantly collapse the probability of your friend-in-San-Francisco's newspaper's headline to whatever *your* headline is.

Some popular science articles might talk about how you measure the spin of one particle, and then instantaneously you know the spin of another particle on the other side of the galaxy. But *unless and until* something more is said about it, that's no different from the case of the newspapers, and seems 100% compatible with local realism!

The **Church-Turing Thesis** says that every physical process can be simulated by a Turing machine to any desired precision.

The way that Church and Turing understood this was as a definition of computation, but we can think of it instead as a falsifiable claim about the physical world. You can think about this as the idea that the entire universe is a video game: you've got all sorts of complicated things like quarks and whatnot, but at the end of the day, you've got to be able to simulate it on a computer.

Theoretical computer science courses can be seen as basically math courses.
So what *does* connect them to reality? The Church-Turing Thesis.

The **Extended Church-Turing Thesis** says moreover that, when we simulate reality on a digital computer, there's at most a polynomial (e.g., linear or quadratic) blowup in time, space, and other computational resources.

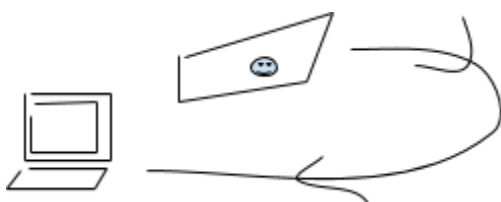
So, what does quantum mechanics have to say about each of these principles?

To give you a teaser for much of the rest of the course:

We'll still use probabilities. But the way we'll *calculate* probabilities will be totally different, and will violate the axiom of monotonicity. That is, *increasing* the number of ways for an event to happen, can *decrease* the probability that it happens.

Locality will be upheld. But *Local Realism* will be overthrown. And if those two principles sounded like restatements of each other---well, quantum mechanics will dramatically illustrate the difference between them!

As we'll see, the Church-Turing Thesis still seems to be in good shape, even in light of quantum mechanics.



Using time dilation, you could travel billions of years in the future and get results to hard problems. Fun! But you'd

need a *LOT* of energy, and if you have that much energy in one place you basically become a black hole. Not so fun!

But the *Extended* Church-Turing Thesis seems to be false, with quantum computing standing as a glaring counterexample to it--possibly the *one* counterexample that our laws of physics allow.

With that said, however, one can formulate a quantum version of the Extended Church-Turing Thesis, which remains true as far as anyone knows today.

Lecture 2, Thurs Jan 19: Probability Theory and QM

Feynman said that everything about quantum mechanics could be encapsulated in the **Double Slit Experiment**.

In the double-slit experiment, you shoot photons one at a time toward a wall with two narrow slits. Where each photon lands on a second wall is probabilistic. If we plot where photons appear on the back wall, some places are very likely, some not.

Note that this itself isn't the weird part: we could totally justify this happening, by some theory where each photon just had some extra degree of freedom (an "RFID tag") that we didn't know about, and that determined which way it went. What's weird is as follows. For some interval on the second wall:

Let P be the probability that the photon lands in the interval with both slits open.

Let P_1 be the probability that the photon lands in the interval if only slit 1 is open.

Let P_2 be the probability that the photon lands in the interval if only slit 2 is open.

You'd think that $P = P_1 + P_2$. But experiment finds that that's not the case! Even places that are *never* hit when both slits are open, can sometimes be hit if only one slit is open.

The weirdness isn't that "God plays dice," but rather that "these aren't normal dice"!

You may think to measure which slit the photon went through, but doing so *changes* the measurements into something that makes more sense. Note that it isn't important whether there's a conscious observer: if the information about which slit the photon went through leaks out in any way, the results go back to looking like they obey classical probability.

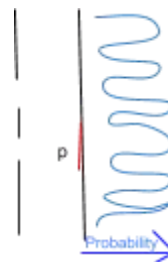
As if Nature says "What? Me? I didn't do anything!"

This reversion to classical probabilities, when systems are coupled to their environments, is called **decoherence**.

Decoherence is why the usual laws of probability look like they work in everyday life. A cat isn't found in a superposition of alive and dead states, because it interacts constantly with its environment. These interactions essentially leak information about the 'cat system' out.

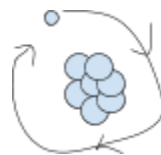
Quantum superposition is something that happens to particles, or groups of particles, when they're isolated from their environments. Needing the particles to be isolated is why it's so hard to build a quantum computer. (And what if the particles aren't *perfectly* isolated, but merely *mostly* isolated? Great question! We'll come back to it later in the course.)

The story of atomic physics between roughly 1900 and 1926 is that scientists kept finding things that didn't fit with the usual laws of mechanics or probability. They usually came up with hacky solutions that explained a phenomenon without connecting it to much else. That is, until Heisenberg, Schrödinger, etc. came up with the general rules of quantum mechanics.



A traditional quantum physics class would go through the whole series of experiments by which physicists arrived at quantum mechanics, but we're just going to accept the rules as given and see what we can do from there.

Briefly, though, take the usual high school model of the electron, rotating around a nucleus in a fixed orbit. Scientists realized that this model would mean that the electron, as an accelerating electric charge, would be constantly losing energy and spiraling inwards until it hit the nucleus. To explain this, along with the double-slit experiment and countless other phenomena, physicists eventually had to change the way probabilities are calculated.



Instead of using probabilities $p \in [0, 1]$ they started using **amplitudes** $\alpha \in \mathbb{C}$. Amplitudes can be positive or negative, or more generally complex numbers (with real and imaginary part).

The central claim of quantum mechanics is that to fully describe the state of an isolated system, you need to give one amplitude for each possible configuration that you could find the system in on measuring it.

The **Born Rule** says that the probability you see a particular outcome is the squared absolute value of the amplitude:

$$P = |\alpha|^2$$

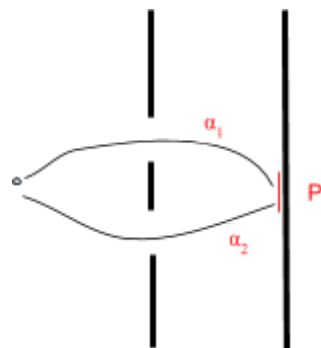
$$= (\text{the real part of } \alpha)^2 + (\text{The imaginary part of } \alpha)^2$$

So let's see how amplitudes being complex leads them to act differently from probabilities. Let's revisit the Double Slit Experiment considering **interference**. We'll say that:

the total amplitude of a photon landing in a spot,	α
is the amplitude of it getting there through the first slit,	α_1
plus the amplitude of it getting there through the second slit,	α_2

$$P = |\alpha|^2 = |\alpha_1 + \alpha_2|^2$$

$$= |\alpha_1|^2 + |\alpha_2|^2 + \alpha_1^* \alpha_2 + \alpha_1 \alpha_2^*$$

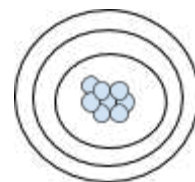


If $\alpha_1 = \frac{1}{2}$ and $\alpha_2 = -\frac{1}{2}$, then interference means that if both slits are open $P = 0$, but if only one of them is open, $P = \frac{1}{4}$.

So then to justify the electron not spiraling into the nucleus:

We say that, yes, there are many paths where the electron does do that, but some have positive amplitudes and others have negative amplitudes and they end up canceling each other out.

With some physics we won't cover in this class, you'd discover that the possibilities where amplitudes don't cancel each other out lead to discrete energy levels, where are the places where the electrons can sit—this phenomenon being what leads to chemistry.



We use **Linear Algebra** to model states of systems as vectors and the evolution of systems in isolation as transformations of vectors.

$$M \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} a_1' \\ a_2' \end{bmatrix}$$

For now, we'll consider classical probability. Let's look at flipping a coin:

$$\begin{bmatrix} p \\ q \end{bmatrix} \begin{matrix} \text{tails} \\ \text{heads} \end{matrix} \quad \begin{matrix} p, q \geq 0 \\ p + q = 1 \end{matrix}$$

We model this with a vector listing both possibilities and assigning a probability to each.

We can apply a transformation, like turning the coin over.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} q \\ p \end{bmatrix}$$

Turning the coin over means the prob that the coin *was* heads is now the probability that the coin *is* tails. If it helps, you can think of the transformation matrix as:

$$\begin{bmatrix} P(\text{tails}|p) & P(\text{tails}|q) \\ P(\text{heads}|p) & P(\text{heads}|q) \end{bmatrix}$$

We could also flip the coin fairly.

$$\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$$

Which means regardless of previous position, both possibilities are now equally likely.

Let's say we flip the coin, and if we get heads we flip again, but if we get tails we turn it to heads.

$$\begin{bmatrix} 0 & \frac{1}{2} \\ 1 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} \frac{q}{2} \\ p + \frac{q}{2} \end{bmatrix}$$

Does that make sense?

If we say that p, q are $P(\text{tails})$ and $P(\text{heads})$ after the first flip:

Then the probability the coin will land on tails in the end is:

0 if (it lands on tails on the first flip) and

$\frac{1}{2}$ if (it lands on heads and we flip again).

So we sum those values.

The probability that the coin will land on heads in the end is:

1 if (it lands on tails on the first flip) and

$\frac{1}{2}$ if (it lands on heads and we flip again).

So we sum those values.

So which matrices *can* be used as transformations?

Firstly, we know that all entries have to be non-negative (because classical probabilities can't be negative).

We also know that all columns must sum to 1, since we need the sum of initial probabilities to equal the sum of the transformed probabilities (namely, both should equal 1).

$$A \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = [\text{i}^{\text{th}} \text{ column of } A]$$

We can see this clearly by using basis vectors.

A matrix that satisfies these conditions is called a **Stochastic Matrix**.

Now let's say we want to flip two coins, or rather, two bits. For the first coin $P(a) = P(\text{getting } 0)$, $P(b) = P(\text{getting } 1)$. For the second coin we'll use $P(c)$ and $P(d)$.

$$\begin{bmatrix} a \\ b \end{bmatrix}^0 \quad \begin{bmatrix} c \\ d \end{bmatrix}^0$$

To combine the two vectors, we need a new operation, called **Tensor Product**.

$$\begin{bmatrix} a \\ b \end{bmatrix} \otimes \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} P_{00} \\ P_{01} \\ P_{10} \\ P_{11} \end{bmatrix} = \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix}$$

It's worth noting that not all possible 4-element vectors can arise by tensoring two 2-element vectors.

For example:

$$\begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ 0 \\ 0 \\ \frac{1}{2} \end{bmatrix}$$

would mean that

$$\left(\frac{1}{2}\right) \left(\frac{1}{2}\right) = (ac)(bd) = (ad)(bc) = (0)(0)$$

Therefore the right-hand side can't be a tensor product.

Let's say that if the first bit is 1, we want to flip the second bit:

$$\begin{matrix} 00 \\ 01 \\ 10 \\ 11 \end{matrix} \begin{bmatrix} \\ \\ \\ \end{bmatrix} \quad \begin{matrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \\ 0 \end{matrix} \begin{matrix} 00 \\ 01 \\ 10 \\ 11 \end{matrix}$$

We'd do:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ 0 \\ 0 \\ \frac{1}{2} \end{bmatrix}$$

This 4×4 matrix is called the **Controlled NOT** or **CNOT** matrix; it will also come up often in quantum computing.

Note that we've reached an output distribution that we previously proved can't arise as a tensor product! Such a distribution is called **correlated**: learning one bit tells you something about the other bit. (In this case, the two bits are always equal; with 50% probability they're both 0 and with 50% probability they're both 1.) So, we've learned that the CNOT matrix can *create correlations*: it can transform an uncorrelated distribution into a correlated one.

Quantum mechanics basically follows the same process to model states in quantum systems except that it uses amplitudes instead of probabilities.

$$\begin{bmatrix} & U & \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Where we preserve $\sum_{i=1}^n |a_i|^2 = 1 = \sum_{i=1}^n |b_i|^2$

and $|a_i|^2$ or $|b_i|^2$ represent the probability of measuring outcome i .

Lecture 3, Tues Jan 24: Basic Rules of QM

Tensor products are a way of building bigger vectors out of smaller ones.

Let's apply a NOT operation to the first bit, and do nothing to the second bit. That's really the same as defining the function f as $f(00) = 10, f(01) = 11, f(10) = 00, f(11) = 01$. So we can write the operation as follows:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{matrix} & \begin{matrix} 00 & 01 & 10 & 11 \end{matrix} \\ \begin{matrix} 00 \\ 01 \\ 10 \\ 11 \end{matrix} & \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{matrix}$$

A **quantum state** (technically, a “pure state”) is a unit vector in \mathbb{C}^N describing the state of a quantum system.

Formally a quantum state could exist in any dimension. Physics courses cover infinite-dimensional vectors, but we'll stick to discrete systems (which is to say that when we make a measurement, there are only finitely many possible outcomes).

What does quantum mechanics say about the universe being discrete or continuous at the base level?

It suggests a strange, hybrid picture. There's a continuum of possible quantum states, but every measurement has a discrete outcome. A system with two amplitudes, $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$, has uncountably infinitely many possible states (given the only restriction is that $|\alpha|^2 + |\beta|^2 = 1$), though note that the same would be true even if we described states using classical probabilities. In both cases, classical and quantum, the continuum is never directly observed, but is only used to calculate the probabilities of discrete outcomes.

The **qubit** is the simplest interesting quantum system.

It's a two-level system (we label the levels '0' and '1'), with an amplitude for 0 and an amplitude for 1.

A one-state quantum system would just be $\begin{bmatrix} 1 \end{bmatrix}$. Not very interesting!

In physics, following Dirac, we like to write quantum state vectors using the so-called **Ket Notation**.

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \alpha|0\rangle + \beta|1\rangle$$

$$\text{Note that } |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

and that $|\psi\rangle$ is a symbol we'll often use for quantum states.

Why do we use ket notation?

One main advantage is that practically speaking, we usually deal with really sparse vectors (where most amplitudes are 0). Ket notation makes it easier to represent only the values we're talking about.

It's really just a formalism to make life easier, we can put anything in ket notation. Look: this is Schrödinger's Cat in ket notation: $|\text{cat}\rangle + |\text{cat}\rangle$.

Often you'll need to take the transpose of a vector $\begin{bmatrix} \alpha \\ \beta \end{bmatrix} \rightarrow [\alpha \ \beta]$ or for complex values $\begin{bmatrix} \alpha \\ \beta \end{bmatrix} \rightarrow [\alpha^* \ \beta^*]$

Using the complex conjugate allows you to define a norm

$$||v||^2 = v^T v$$

$$\begin{aligned} \text{Then we get } v^T v &= [\alpha^* \ \beta^*] \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \\ &= \alpha^* \alpha + \beta^* \beta \\ &= |\alpha|^2 + |\beta|^2 \end{aligned}$$

What does this look like in ket notation?

Just like we have the **ket** $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ for $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$

We define the **bra** $\langle\psi| = \alpha^*\langle 0| + \beta^*\langle 1|$ for $[\alpha^* \ \beta^*]$

And we define $\langle x|y\rangle$ as the inner product of $|x\rangle$ with $|y\rangle$ (which automatically involves taking the conjugate transpose of $|x\rangle$).

Therefore $\langle\psi|\psi\rangle = 1$.

So $\langle v|w\rangle = \langle w|v\rangle^*$.

Remember: the way we change quantum states is by applying linear transformations:

$$U \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha' \\ \beta' \end{pmatrix}$$

A linear transformation is **unitary** if $|\alpha|^2 + |\beta|^2 = |\alpha'|^2 + |\beta'|^2$.

Unitary matrices correspond to unitary transformations.

We've got the identity $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and permutation $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ matrices, which are the only unitaries that are also stochastic.

Other unitaries include

$$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \text{ which maps } \begin{array}{l} |0\rangle \rightarrow |0\rangle \\ |1\rangle \rightarrow i|1\rangle \end{array}, \quad \begin{bmatrix} 0 & -i \\ -1 & 0 \end{bmatrix}$$

Note: Euler's Equation says $e^{i\theta} = \cos \theta + i \sin \theta$, and we could also replace any of the 1's, -1's, i 's, or $-i$'s by that

All real possible states of a qubit define a circle and all complex possible states define a hypersphere. That's because these states are all the vectors of length 1.

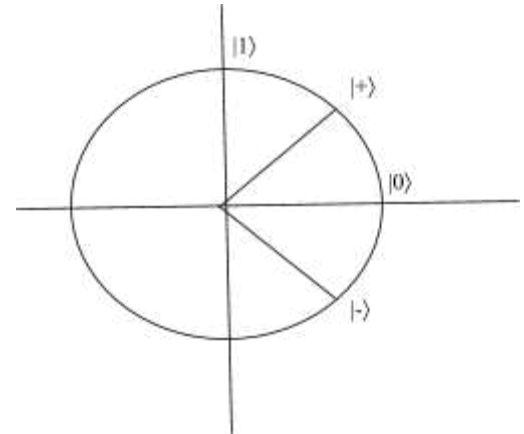
We define:

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

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

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

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



Unitary transformations are norm-preserving linear transformations.

For any angle θ you could have $R_\theta = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$ which grabs a vector and rotates it θ radians.

What does it mean that a unitary matrix preserves the 2-norm?

It means applying a unitary transformation retains the inner product, $\langle \psi | \psi \rangle$.

$$\langle \psi | \psi \rangle = (U|\psi\rangle)^\dagger U|\psi\rangle = \langle \psi | U^\dagger U | \psi \rangle$$

For this to hold for any ψ , $U^\dagger U$ must equal I . Which means $U^{-1} = U^\dagger$.

That in turn implies that the rows of U must be an orthogonal unit basis.

So you can tell if a matrix is unitary by checking if the rows (or, equivalently, the columns!) form an orthogonal unit basis.

This is not the “operational definition” of unitary matrices, but is a logical consequence of unitary transformations preserving inner products.

An **orthogonal matrix** is both unitary and real.

Any orthogonal matrix is a product of rotations and reflections.

Some examples:

$$R_{\pi/4}|0\rangle = |+\rangle$$

$$R_{\pi/4}|+\rangle = |1\rangle$$

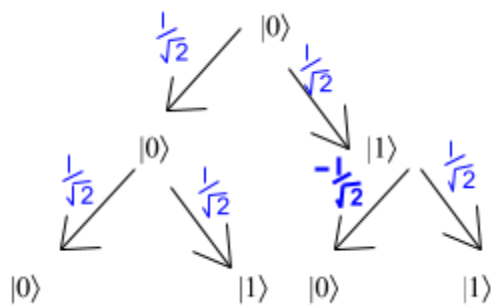
$$R_{\pi/4}|1\rangle = -|-\rangle$$

You'll get a full revolution after applying $R_{\pi/4}$ eight times.

In the classical world, if an event could happen multiple ways, but will be “random” no matter which way it happens, then it’s simply “random” overall.

But in the quantum world, you can sometimes apply a unitary transformation to a superposition state and then get a determinate answer

Anything interesting in quantum mechanics can be explained in terms of **interference**.



The $|0\rangle$ basis state can go to states $|0\rangle$ and $|1\rangle$ equally.

Above, there are two different paths that lead to the $|0\rangle$ outcome, but they cancel each other out, with one having positive amplitude and the other having negative amplitude.

The $|0\rangle$ states interfere destructively.

The $|1\rangle$ states interfere constructively.

No matter what unitary transformation you apply: If $|0\rangle$ goes to $U|0\rangle$, then $-|0\rangle$ goes to $-U|0\rangle$.

The zero state and the minus zero state are indistinguishable mathematically, which is to say:

Global phase is unobservable.

Multiplying your entire quantum state by a scalar is like if last night someone moved the entire universe twenty feet to the left. We can only really measure things relative to other things! Which leads to a second maxim:

Relative phase is observable.

To distinguish between the states $|+\rangle$ and $|-\rangle$ we can rotate by 45 degrees and then measure to see whether we got $|0\rangle$ or $|1\rangle$.

There are no second chances. Once you measure, the outcome is set.

Lecture 4: Thurs Jan 26: Quantum Gates and Circuits, Zeno Effect, Elitzur-Vaidman Bomb

We call the matrix $R_{\pi/4} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$ the \sqrt{NOT} gate, as $R_{\pi/4}^2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, aka the NOT Gate.

The **Hadamard Gate** is $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

Hadamard is ubiquitous in quantum information because it maps the $|0\rangle, |1\rangle$ basis to the $|+\rangle, |-\rangle$ basis.

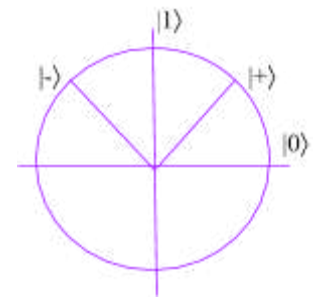
$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = |+\rangle$$

Similarly $H|+\rangle = |0\rangle$, $H|1\rangle = |-\rangle$, and $H|-\rangle = |1\rangle$

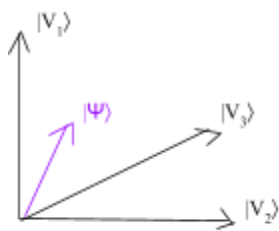
Note that we've got two orthogonal (complementary) bases: being maximally certain in the $|+\rangle, |-\rangle$ basis means that you're maximally uncertain in the $|0\rangle, |1\rangle$ basis and vice versa.

Why would we want to use 2 different bases?

We like to think of vectors existing abstractly in vector space, but to use one meaningfully, we often need to pick a basis. When we see some actual quantum algorithms and protocols, we'll see the power that comes from switching between bases.



Side note, when talking about the Born Rule, we've been using a special case of one particular basis for simplicity.



We can think about measurement more generally. Measuring the state $|\psi\rangle$ in the orthonormal basis $\{|V_1\rangle, \dots, |V_N\rangle\}$, you'll get the outcome $|V_i\rangle$ with probability $|\langle V_i|\psi\rangle|^2$.

So the probability of the outcome $|V_3\rangle$ is the projection onto the basis vector.
 $|\langle V_i|\psi\rangle|^2 = |\alpha_i|^2$

We pick bases like $\{|0\rangle, |1\rangle\}$ arbitrarily as a nice convention.

To do operations in a different basis, we can use unitary transformations to convert between bases.

So for $\{|V_1\rangle, \dots, |V_N\rangle\}$ use $|V_1\rangle = |1\rangle, \dots, U|V_N\rangle = |N\rangle$ if you want the $\{|1\rangle, \dots, |N\rangle\}$ basis

There's an extreme point of view in quantum mechanics that unitary transformations are the only thing that really exist, and measurements don't really exist. And the converse also exists: the view that

measurements are the only thing that really exist, and that unitary transformations don't. More when we talk about interpretations!

Unitary Transformations are:

- Invertible. This should be clear, since preserving the two norm means that $U^\dagger U = 1$ which means $U^\dagger = U^{-1}$.
 - Reversible. The transformation $|\psi\rangle \rightarrow U|\psi\rangle$ can be reversed with $U^{-1}U|\psi\rangle = |\psi\rangle$.
Interestingly this implies that unitary evolution can never destroy information, which should imply that the universe is reversible. Physics has treated the microscopic laws as reversible since Galileo's time (i.e. a time-reversed video of a swinging pendulum still shows it obeying the laws of physics). So for example burning a book shouldn't destroy the information within, as physics says that in principle you can get all the information from the smoke and ash left over.
- Deterministic
- Continuous
i.e. you can apply them in a time-continuous way.

That's why it's important that unitary matrices are complex.

If the transformation $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ took place in 1 second, then over the first half of the second, perhaps $\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$ took place—or some other square root of the transformation.

By the way, there is a 3x3 matrix that “squares” to $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}^2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

But to take a square root of this transformation, *either* you need complex numbers, or else you need to add a third dimension. The latter is analogous to reflecting your three-dimensional self by rotating yourself in a fourth dimension—as in some science fiction stories!

Important: If you come back reflected after a trip into the fourth dimension, don't eat anything without first consulting medical professionals. Normal food will have molecules of the wrong chirality for you to digest them.

Measurements break all three rules of unitary transformations! Measurements are:

- Irreversible
 - Whatever information about the system you didn't capture is now lost.
- Probabilistic
 - Everything in quantum mechanics is deterministic *until* measurement (or information leaves the system), but measurement outcomes are in general random.
- Discontinuous
 - The “collapse of the amplitude vector” is treated as instantaneous in textbooks.

So how can we reconcile these two sets of rules?

That's the **Measurement Problem**. We'll talk about various points of view on it later.

Despite the philosophical conflict, unitary transformations and measurement sync up well because:

- Unitary transformations preserve the 2-norm and
- Measurement gives probabilities governed by the 2-norm

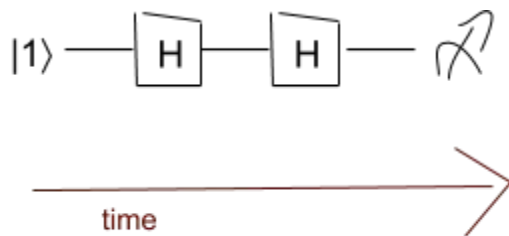
Classical probability is based on the 1-norm, while quantum mechanics is based on the 2-norm. So it's natural to wonder: what about theories based on the 3-norm, 4-norm, etc.?

Actually, there don't seem to be any interesting theories there (the extra credit problem on the homework on norm preserving linear transformations sheds light on why), making quantum mechanics a bit of "an island in theory space". If you try to adjust anything about it in any way, you typically get gunk! You could alternatively say that there seems to be "nothing near quantum mechanics, that's nearly as nice as quantum mechanics itself." As another example of this, there are many technical reasons why complex numbers work better than the reals or quaternions as amplitudes.

One more example of a linear transformation.

The matrix $\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ i & -i \end{bmatrix}$ maps $|0\rangle \rightarrow \frac{|0\rangle + i|1\rangle}{\sqrt{2}}$
and $|1\rangle \rightarrow \frac{|0\rangle - i|1\rangle}{\sqrt{2}}$

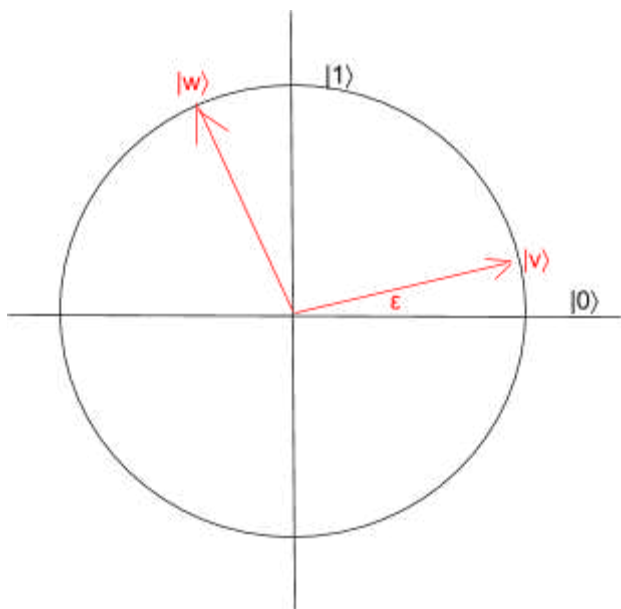
Quantum Circuit Notation helps us keep track of what qubits we have and what operations we apply to them.



So to the left we start with $|1\rangle$, apply a Hadamard Gate, apply another Hadamard Gate, then measure (implied to be in the $|0\rangle, |1\rangle$ basis)

We'll never branch a qubit line into multiple qubit lines, since that doesn't correspond to a unitary transformation. To enlarge a system we can use a new $|0\rangle$ qubit, an **ancilla** qubit.

There are several interesting phenomena that already happen in the quantum mechanics of one qubit.



Suppose you have a qubit in the state $|0\rangle$. We can know this because it's staying 0 over and over in measurements. Let's say we want to put it in the $|1\rangle$ state without using any unitary transformations.

For some small ϵ , we can measure the qubit in a basis that's rotated from $\{|0\rangle, |1\rangle\}$ by an angle ϵ . The probability of getting the qubit to move by ϵ increases as ϵ decreases.

$$\text{Prob}(|v\rangle) = \cos^2 \epsilon$$

$$|v\rangle = \cos \epsilon |0\rangle + \sin \epsilon |1\rangle$$

where

$$\text{Prob}(|w\rangle) = \sin^2 \varepsilon \approx \varepsilon^2$$

So over $\approx \frac{1}{\varepsilon}$ such measurements, we could slowly drag the qubit from $|0\rangle$ to $|1\rangle$.

What's the likelihood that we'd ever get a measurement outcome that *wasn't* the one we wanted?

By union bound, it's of order $\frac{1}{\varepsilon} \times \varepsilon^2 = \varepsilon$, so can be made arbitrarily small.

This is called **The Quantum Zeno Effect**

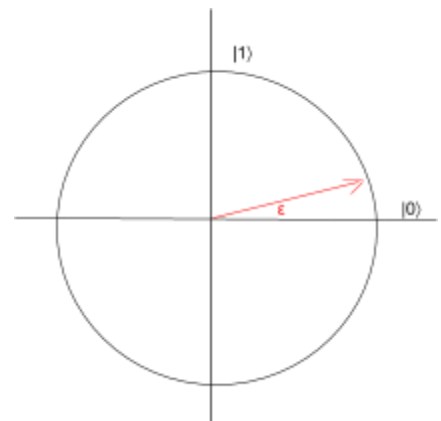
One of its discoverers was Alan Turing.

Perhaps an everyday-life analog would be asking a stranger to have coffee with you, then to go dancing, etc.—there's a higher probability of success than if you just immediately ask them to marry you!

Another interesting variant of the same kind of effect is as follows:

Say we want to keep a qubit at $|0\rangle$, but it keeps rotating towards $|1\rangle$ (it's *drifting*).

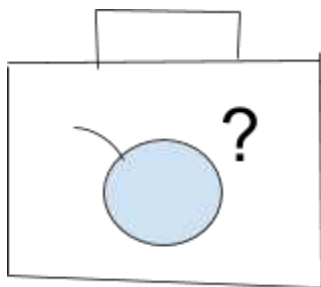
If we keep measuring it on the $|0\rangle, |1\rangle$ basis the odds of it jumping to $|1\rangle$ at any given measurement is only ε^2 . So if we repeat $\approx \frac{1}{\varepsilon}$ times, then the probability it ending up at $|1\rangle$ is only $\approx \varepsilon$, even though it would have drifted to $|1\rangle$ with certainty had we not measured.



This is called **The Watched Pot Effect**.

Another interesting phenomenon is the **Elitzur-Vaidman Bomb**.

A quantum effect discovered in the early 1990's.



Say we're at a quantum airport and there's a piece of unattended luggage which could be a bomb, but opening the suitcase would trigger it.

How do we check if there's a bomb there without triggering it?

We could make a query with a classical bit:

$$b \in \{0 \text{ (don't make query)}, 1 \text{ (make query)}\}$$

But then we either learn find nothing, *or* we set off the bomb if there's indeed a bomb there. Not good!

Instead, we can upgrade to a qubit:

$$|b\rangle = \alpha|0\rangle + \beta|1\rangle$$

Now assume: If there's no bomb the state $|b\rangle$ gets returned to you.

If there is a bomb, the bomb measures in the $\{|0\rangle, |1\rangle\}$ basis. If the outcome is $|0\rangle$, then $|0\rangle$ is returned to you, while if the outcome is $|1\rangle$, the bomb explodes.

What we can do is apply the rotation $R_\varepsilon = \begin{bmatrix} \cos \varepsilon & -\sin \varepsilon \\ \sin \varepsilon & \cos \varepsilon \end{bmatrix}$. Giving us:

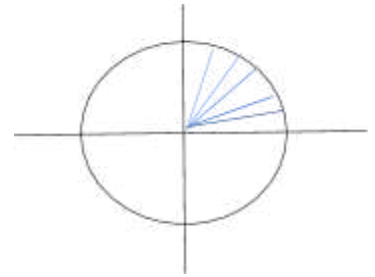
$$\cos \varepsilon |0\rangle + \sin \varepsilon |1\rangle$$

If there's a bomb, the probability it explodes is $\sin^2 \varepsilon \approx \varepsilon^2$, otherwise we get back $|0\rangle$.

If there's no bomb, we get back $\cos \varepsilon |0\rangle + \sin \varepsilon |1\rangle$

So repeating about $\pi/(2\varepsilon)$ times makes the probability of setting off the bomb only $\approx \frac{1}{\varepsilon} \times \varepsilon^2 = \varepsilon$. Yet by measuring our $|b\rangle$ qubit to see whether it's $|0\rangle$ or $|1\rangle$, we still learn whether or not a bomb was there.

Of course, the catch is that this requires not merely a qubit on our end, but also a bomb that can be “quantumly interrogated”!

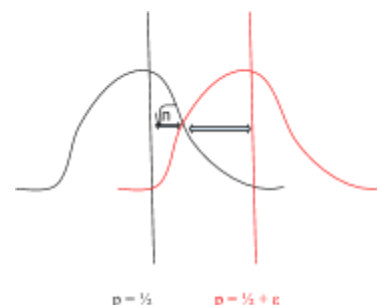


Lecture 5, Tues Jan 30: Coin Problem, Inner Products, Multi-Qubit States, Entanglement

Say you have a coin, and you want to figure out if it's fair ($p = \frac{1}{2}$) or if it's biased $p = \frac{1}{2} + \varepsilon$. How would you go about doing so?

The classical approach to solving this problem would be to flip the coin a lot (about $\frac{1}{\varepsilon^2}$ times), keeping track of heads and tails until you have a strong degree of certainty that randomness isn't affecting your results. Standard probability stuff.

This requires about $\log \frac{1}{\varepsilon}$ bits of memory to store the running totals. In fact, there's a theorem by Hellman and Cover from the 70's that says that any protocol to solve this problem requires that much storage.



What if instead we used quantum states?

We can start with a qubit in the $|0\rangle$ state, and consider the two rotations R_ε and $R_{-\varepsilon}$, which rotate by ε and $-\varepsilon$ radians respectively. We can repeatedly flip the coin, and if it lands tails apply R_ε (rotating clockwise) and if it lands heads apply $R_{-\varepsilon}$ (rotating counterclockwise). After many flips (order $\approx \frac{1}{\varepsilon^2}$) we can then measure the qubit and statistically infer that if it's in the $|0\rangle$ state, the coin was fair, while if it's in the $|1\rangle$ state, the coin is biased.

- Won't counting out the right number of steps again require a lot of storage?
 - No. We can give a protocol with a half-life (some independent probability of halting at each step) causing it to repeat approximately the number of times we want it to.
- What about if the qubit drifts by a multiple of π , won't that make a biased coin look fair?
 - That's possible, but we can make it so that a biased coin is more likely to land on $|1\rangle$ than a fair coin.

Quantum information protocols are like baking souffles.
Opening the oven too early will collapse the souffle.

This is our first example of a quantum protocol getting a resource advantage:

the quantum solution takes **1 qubit of storage** as opposed to the classical solution's $\log \frac{1}{\varepsilon^2}$ bits.

This result was shown by Professor Aaronson and his former student Andy Drucker.

It wasn't a particularly hard problem, but no one had asked the question before.

There's still "low hanging fruit," even in the mechanics of a single qubit!

Distinguishability of Quantum States

Given two orthogonal quantum states $|v\rangle$ and $|w\rangle$ there's a basis that distinguishes them.



These on the other hand are indistinguishable.

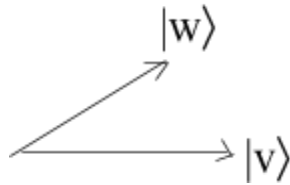


$|\langle v|w\rangle|$ gives a good measure of the distinguishability of arbitrary states.

$$|\langle v|w\rangle| = 1$$

$$|\langle v|w\rangle| = 0$$

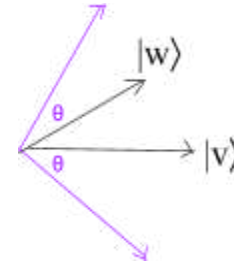
What about these?



More specifically: What measurement would minimize the chance of making a mistake in differentiating $|v\rangle$ from $|w\rangle$?

You may want to measure in $|v\rangle$ (something else) the basis, as it would eliminate one kind of error completely (not $|v\rangle$ getting ensures $|w\rangle$ the state was). But if you just want to maximize the probability of getting the right answer, $|v\rangle$ and $|w\rangle$ if and are equally likely, then there's a better way:

Take the bisector of $|v\rangle$ and $|w\rangle$, and get the angles 45° to either side, ensuring each original vector is the same distance to its closest basis vector.



A general state of **2 Qubits** is:

$$\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

The probability of getting $|00\rangle = |\alpha|^2$
 $|01\rangle = |\beta|^2$
 $|10\rangle = |\gamma|^2$
 $|11\rangle = |\delta|^2$

Note that $|00\rangle$ is the same as $|0\rangle|0\rangle$ or $|0,0\rangle$ or $|0\rangle \otimes |0\rangle$

In principle there's no distance limitation between qubits. One qubit could be on Earth, and the other could be with your friend on the moon.

In such a case, though, you'd only be able to measure the first qubit:

The probability of getting $|0\rangle$ is $|\alpha|^2 + |\beta|^2$ because those are the amplitudes compatible with 0 in the first qubit.

The probability of getting $|1\rangle$ is $|\gamma|^2 + |\delta|^2$

Suppose I measure the first qubit and get the outcome $|0\rangle$. What can I say about the second qubit?

Well we've narrowed down the possibilities to $\alpha|00\rangle$ and $\beta|01\rangle$. The state of the system is thus now in the superposition: $|0\rangle \otimes \frac{\alpha|0\rangle + \beta|1\rangle}{\sqrt{|\alpha|^2 + |\beta|^2}}$ ← Don't forget to normalize!

This is called the **Partial Measurement Rule**

Systems collapse however is needed to fit the measurement you made and the outcome you saw.

This is actually the last "basic rule" of quantum mechanics that we'll see in the course. Everything else is just logical consequences of rules we've already covered.

This $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ is the **Controlled NOT**.

Remember: it flips the 2nd bit iff the 1st bit is 1.

What if we wanted to always do NOT on the 2nd bit:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

This is $\mathbf{I} \otimes \mathbf{NOT}$

/ |

(nothing on 1st bit) with (NOT on 2st bit)

It can be decomposed as: $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ which makes it a tensor product unitary.

What if we want $\mathbf{NOT} \otimes \mathbf{I}$?

$$\begin{matrix} 00 & 01 & 10 & 11 \\ \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{matrix}$$

Remember that rows and columns represent the transformation $f(\text{row}) = \text{col}$
so the amplitude on 00 in the input
is the amplitude on 10 in the output

Very often in quantum information we'll want to take a group of qubits and perform an operation on one of them: say, "Hadamard the third qubit."

What that really means is applying the unitary matrix $I \otimes I \otimes H \otimes \dots \otimes I$:

The tensor product of the desired operation on the relevant qubit(s), with the identity operation on all the other qubits.

What's $H \otimes H$?

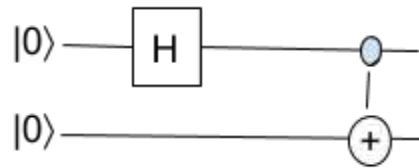
$$\frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Why should it look like this?

Let's look at the first row: $H|00\rangle = |++\rangle$. Which means for each qubit there's an equal amplitude on $|0\rangle$ and $|1\rangle$.

All of these are examples of using tensor products to build bigger unitary matrices, except for the Controlled NOT, where the first qubit affects the second. We'll need operations like Controlled NOT in order to have one qubit affect another.

2 Qubits In Quantum Circuit Notation



Start with 2 qubits in $ 0\rangle$		Apply Hadamard to 1st qubit		Apply a Controlled NOT with the 1st qubit as the control and the 2nd as the target .	
$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	$ 00\rangle$	$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}$	$\frac{ 00\rangle + 10\rangle}{\sqrt{2}} = +\rangle \otimes 0\rangle$	$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$	$\frac{ 00\rangle + 11\rangle}{\sqrt{2}}$

The action of the Controlled NOT can also be written as $|x, y\rangle \rightarrow |x, y \otimes x\rangle$

The state that this circuit ends on, $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$ is called the **Singlet** or the **Bell Pair** or the **EPR Pair**

This state is particularly interesting because measuring the first qubit collapses the state of the second qubit. The state can't be factored into a tensor product of the first qubit's state and the second's. Such a state is called **entangled**, which for pure states simply means: not decomposable into a tensor product.

A state that's not entangled is called **unentangled** or **separable** or a **product state** (for pure states, which are the only kind being discussed at this point in the course, all three of these mean the same thing).

The basic rules of quantum mechanics, which we saw earlier, force entanglement to exist. It was noticed quite early in the history of the field. It turns out that *most* states are entangled.

As we mentioned earlier, entanglement was arguably what troubled Einstein most about quantum mechanics. He thought that it meant that quantum mechanics must entail "spooky action at a distance."

That's because particles need to be close to become entangled, but once they're entangled you can separate them to an arbitrary distance and they'll stay entangled. This has actually been demonstrated experimentally for distances of up to 150 miles (improved to a couple thousand miles by Chinese satellite experiments, while the course was being taught!).



Let's say that Alice and Bob entangle a pair of particles by setting their state to $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$, then Alice brings her particle to the moon while Bob stays on Earth. If Alice measures her particle, she can *instantaneously* know whether Bob will observe a $|0\rangle$ or a $|1\rangle$ when he measures his.

This bothered Einstein, but others thought that it wasn't that big a deal. After all, Alice doesn't get to control the outcome of her measurement! She sees $|0\rangle$ and $|1\rangle$ with equal probability, which

means that in this case, the “spooky action” can be explained as just a correlation between two random variables, as we could already see in the classical world.

However, a famous 1935 paper of Einstein, Podolsky, and Rosen brought up a further problem: namely, there are other things Alice could do instead of measuring in the $|0\rangle, |1\rangle$ basis.

What happens if Alice measures in the $|+\rangle, |-\rangle$ basis?

She’ll get either $|+\rangle$ or $|-\rangle$, as you might expect.

Indeed, we can model the situation by Alice Hadamarding her qubit and then measuring in the $\{|0\rangle, |1\rangle\}$ basis.

That gives us the state:

$$\frac{|00\rangle + |10\rangle + |01\rangle - |11\rangle}{2}$$

Remember $H|0\rangle = |+\rangle$, etc.

So now, applying the *Partial Measurement Rule* what is Bob’s state?

If Alice sees $|0\rangle$, then Bob’s qubit collapses to the possibilities where Alice sees $|0\rangle$:

$$\frac{|00\rangle + |01\rangle}{2} = |+\rangle$$

Conversely, if Alice sees $|1\rangle$:

$$\frac{|10\rangle - |11\rangle}{2} = |-\rangle$$

The paper goes on to talk about how this is more troubling than before. If Alice measures in the $\{|0\rangle, |1\rangle\}$ basis, then Bob’s state collapses to $|0\rangle$ or $|1\rangle$, but if she measures in the $\{|+\rangle, |-\rangle\}$ basis, then his state collapses to $|+\rangle$ or $|-\rangle$. And *that* looks a lot like faster-than-light communication!

How can we explain this?

One thing we can do is ask “what happens if Bob makes a measurement?”

- In the case where Alice measured her qubit in the $\{|0\rangle, |1\rangle\}$ basis, Bob will see $|0\rangle$ or $|1\rangle$ with equal probability if he measures his qubit in the same basis.
- In the case where Alice measured her qubit in the $\{|+\rangle, |-\rangle\}$ basis...
 - Bob will still see $|0\rangle$ or $|1\rangle$ with equal probability (measuring in the $\{|0\rangle, |1\rangle\}$ basis)

So, at least in this case, the probability that Bob sees $|0\rangle$ or $|1\rangle$ is the same regardless of what Alice chooses to do. As an exercise, check that this remains the case even if Bob measures his qubit in the $\{|+\rangle, |-\rangle\}$ basis.

So, it looks like there might be something more general going on here! In particular, a different description should exist of Bob’s part of the state that’s unaffected by Alice’s measurements. Which brings us to the next lecture...

Lecture 6, Thurs Feb 2: Mixed States

So far we've only talked about **pure states** (i.e., isolated quantum systems), but you can also have quantum superposition layered together with regular, old probabilistic uncertainty. This becomes extremely important when we talk about states where we're only measuring one part.

Last time we discussed the **Bell Pair**, and how if Alice measures her qubit in any basis, the state of Bob's qubit collapses to whichever state she got for her qubit. Even so, there's a formalism that helps us see why Bob can't do anything to learn which basis Alice makes her measurement in, and more generally, why Alice can't transmit *any* information instantaneously--in keeping with special relativity. This is the formalism of...

Mixed States

Which in some sense, are just probability distributions over quantum superpositions. We can define a mixed state as a distribution over quantum states, so $\{p_i, |\Psi_i\rangle\} = p_1, |\Psi_1\rangle, \dots, p_n, |\Psi_n\rangle$ meaning that with probability p_i , the superposition is $|\Psi_i\rangle$.

Note that these don't have to be orthogonal

Thus, we can think of a pure state as a degenerate case of a mixed state where all the probabilities are 0 or 1.

The tricky thing about mixed states is that *different probability distributions over pure states, can give rise to exactly the same mixed state.* (We'll see an example shortly.) But to make manifest why information doesn't travel faster than light, we need a representation for mixed states that's unique. That's why we use:

Density Matrices

represented as $\rho = \sum_i p_i |\Psi_i\rangle\langle\Psi_i|$

$|\Psi_i\rangle\langle\Psi_i|$ is the **outer product** of Ψ with itself.

It's the matrix you get by multiplying $\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix} \begin{bmatrix} \alpha_1^* & \dots & \alpha_N^* \end{bmatrix} = \begin{bmatrix} |\alpha_1|^2 & & \\ & \ddots & \alpha_i \alpha_j^* \\ & \alpha_i^* \alpha_j & \ddots \\ & & & |\alpha_N|^2 \end{bmatrix}$

Note that $\alpha_i \alpha_j^* = (\alpha_i^* \alpha_j)^*$, which means that the matrix is its own conjugate transpose

$\rho = \rho^\dagger$ That makes ρ a **Hermitian Matrix**.

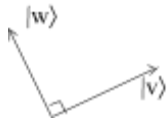
Some examples: $|0\rangle\langle 0| = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ $|1\rangle\langle 1| = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$

Therefore an even mixture of them would be $\frac{|0\rangle\langle 0| + |1\rangle\langle 1|}{2} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} = \frac{I}{2}$

Similarly: $|+\rangle\langle+| = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$ $|-\rangle\langle-| = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix}$

And $\frac{|+\rangle\langle+| + |-\rangle\langle-|}{2} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} = \frac{I}{2}$

Note that an equal mixture of $|0\rangle$ and $|1\rangle$ is *different* from an equal superposition of $|0\rangle$ and $|1\rangle$ (a.k.a. $|+\rangle$), and so they have different density matrices. However, the mixture of $|0\rangle$ and $|1\rangle$ and the mixture of $|+\rangle$ and $|-\rangle$ have the same density matrix, which makes sense because Alice converting between the two bases in our Bell pair example should maintain Bob's density matrix representation of the state.



In fact, this is true of whichever basis Alice chooses: for any two orthogonal vectors $|v\rangle$ and $|w\rangle$, we have that $\frac{|v\rangle\langle v| + |w\rangle\langle w|}{2} = \frac{I}{2}$ (check this!)

Measuring ρ in the basis $|1\rangle, \dots, |N\rangle$ gives us outcome $|i\rangle$ with probability:

$$\Pr(|i\rangle) = \rho_{ii} = \langle i|\rho|i\rangle$$

So, the diagonal entries of the density matrix directly represent probabilities.

You don't need to square them or anything because the Born Rule is already encoded in the density matrix (i.e. $(\alpha_1)(\alpha_1^*) = |\alpha_1|^2$)

$\begin{bmatrix} p_1 & & 0 \\ & \ddots & \\ 0 & & p_N \end{bmatrix}$ In particular, a density matrix that's diagonal is just a fancy way of writing a classical probability distribution.

While a pure state would look like $|\Psi\rangle\langle\Psi| = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$: that is, a matrix of rank one.

What if we want to measure a density matrix in a different basis?

Measuring ρ in the basis $\{|v\rangle, |w\rangle\}$ will give $\Pr(|v\rangle) = \langle v|\rho|v\rangle$ and similarly for $|w\rangle$

You can think of a density matrix as encoding not just one but infinitely many probability distributions, because you can measure it in any basis.

The matrix $\frac{I}{2}$ that we've encountered above, as the even mixture of $|0\rangle$ and $|1\rangle$ (and also of $|+\rangle$ and $|-\rangle$) is called the **Maximally Mixed State**. This state is basically just the outcome of a classical coin flip, which gives it a special property:

Regardless of the basis we measure it in, both outcomes will be equally likely.

So for every basis $|v\rangle, |w\rangle$ we get the probabilities $\langle v|\frac{I}{2}|v\rangle = \frac{1}{2}\langle v|v\rangle = \frac{1}{2}$

$$\langle w|\frac{I}{2}|w\rangle = \frac{1}{2}\langle w|w\rangle = \frac{1}{2}$$

This explains why Alice is unsuccessful in sending a message to Bob, by measuring her half of a Bell pair. Namely, because the maximally mixed state in any other basis is *still the maximally mixed state*.

So how do we handle unitary transformations with density matrices?

Since $\rho = \sum_i p_i |\Psi_i\rangle\langle\Psi_i|$, applying U to ρ means:

$$\sum_i p_i (U|\Psi_i\rangle)(U|\Psi_i\rangle)^\dagger = \sum_i p_i U|\Psi_i\rangle\langle\Psi_i|U^\dagger = U\rho U^\dagger$$

You can pull out the U 's since it's the same one applied to each mixture.

It's worth noting that getting n^2 numbers in the density matrix isn't some formal artifact; we really do need all those extra parameters. What do the off-diagonal entries represent?

$$|+\rangle\langle+| = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

These are where all the 'quantumness' resides.
It's where the interference between $|0\rangle$ and $|1\rangle$ is represented.

The off-diagonal entries can vary depending on relative phase:

$|+\rangle$ has positive off-diagonal entries

$|-\rangle$ has negative off-diagonal entries

$$|i\rangle\langle i| = \begin{bmatrix} \frac{1}{2} & -\frac{i}{2} \\ \frac{i}{2} & \frac{1}{2} \end{bmatrix}$$

Later we'll see that as a quantum system interacts with the environment, the off-diagonal entries tend to get pushed down toward 0.

$$\begin{bmatrix} \frac{1}{2} & \varepsilon \\ \varepsilon & \frac{1}{2} \end{bmatrix}$$

The density matrices in experimental quantum papers typically look like

The bigger the off-diagonal values, the better the experiment, because it represents them seeing more of the quantum effect!

Which matrices can arise as density matrices?

We're effectively asking: What constraints does the form $\sum_i p_i |\Psi_i\rangle\langle\Psi_i|$ put on the matrix ρ ?
Well, such ρ must be:

- Square
- Hermitian

- $\sum_i p_{ii} = 1$ (which is to say: the **trace**, $Tr(\rho) = 1$)

Could $M = \begin{bmatrix} \frac{1}{2} & -10 \\ -10 & \frac{1}{2} \end{bmatrix}$ be a density matrix?

No. Measuring this in the $|+\rangle, |-\rangle$ basis would give
Bad! $\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} M \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{19}{2}$

Remember that you can always transform ρ to $U\rho U^\dagger$, whose diagonal then has to be a probability distribution for all U . If we want that condition to hold, then in linear algebra terms, we need to add the restriction:

- All eigenvalues are non-negative (aka being **PSD: Positive Semidefinite**)

As a refresher: For the matrix ρ , the eigenvectors $|\Psi\rangle$ are the vectors that satisfy the equation:
 $\rho|\Psi\rangle = \lambda|\Psi\rangle$ for some eigenvalue λ

If we had a negative eigenvector $|\Psi\rangle$,

the probability $\langle\Psi|\rho|\Psi\rangle = \lambda$ would be negative, which is nonsense.

Could we have missed a condition? Let's check.

We claim: any Hermitian PSD matrix with trace 1 can arise as a density matrix of a quantum state.

For such a ρ , we can represent it in the form $\sum_i \lambda_i |\Psi_i\rangle\langle\Psi_i|$ where the $|\Psi_i\rangle$ are the (unit) eigenvectors.

Then $\langle\Psi_i|\rho|\Psi_i\rangle = \lambda_i$, so the λ_i 's sum to $\text{Tr}(\rho)=1$.

This process of obtaining eigenvalues and eigenvectors is called **eigendecomposition**.

We know the eigenvalues will be real because the matrix is Hermitian,
They're non-negative because the matrix is PSD.

One quantity you can always compute for density matrices is:

Rank

$\text{rank}(\rho)$ = the number of non-zero eigenvalues λ_i (counting with multiplicity)

A density matrix of rank n might look like $\begin{bmatrix} p_1 & & 0 \\ & \ddots & \\ 0 & & p_n \end{bmatrix}$

While a density matrix of rank 1 represents a pure state.

We know from linear algebra that the rank of an $n \times n$ matrix is always at most n . Physically, this means that every n -dimensional mixed state can be written as a mixture of at most n pure states.

In general, rank tells you the number of pure states that you have to mix to reach a given mixed state.

Now, consider the 2-qubit pure state $\frac{|00\rangle + |01\rangle + |10\rangle}{\sqrt{3}}$.

We'll give the first qubit to Alice and the second to Bob.
How does Bob calculate his density matrix?

By picking some orthogonal basis for Alice's side.

You can rewrite the state as $\sqrt{\frac{2}{3}}|0\rangle|+\rangle + \sqrt{\frac{1}{3}}|1\rangle|0\rangle$, which lets you calculate Bob's density matrix as:

$$\begin{aligned} & \frac{2}{3}|+\rangle\langle+| + \frac{1}{3}|0\rangle\langle 0| \\ &= \frac{2}{3} \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} + \frac{1}{3} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} \end{bmatrix} \end{aligned}$$

In general, if you have a bipartite pure state, it'll look like $\sum_{i,j=1}^N \alpha_{ij} |i\rangle|j\rangle = |\Psi\rangle$

And you can get Bob's local density matrix

$$(\rho_{\text{Bob}})_{j,j'} = \sum_i \alpha_{ij} \alpha_{ij'}^*$$

The process of going from a pure state of a composite system, to the mixed state of part of the system, is called **tracing out**.

The Key Points:

- 1) A density matrix encodes all and only what is physically observable
 - Two quantum states will lead to different probabilities *iff* they have different density matrices
- 2) No-Communication Theorem
 - If Alice and Bob share an entangled state, nothing Alice chooses to do will have any effect on Bob's density matrix.

In other words, there's no observable effect on Bob's end. Which is the fundamental reason that quantum mechanics *is* compatible with the limitations of relativity.

We've already seen particular examples of both statements. But both of them hold in full generality, and you'll prove that in your homework!

OK, just to get you started a bit: recall that the No Communication Theorem says that, if Alice and Bob share an entangled state

$$|\Psi\rangle = \sum_{i,j=1}^N \alpha_{ij} |i\rangle_{\text{Alice}} |j\rangle_{\text{Bob}}$$

there's nothing that Alice can do to her subsystem that affects Bob's density matrix.

You already have the tools to prove this: just calculate Bob's density matrix, then apply a unitary transformation to Alice's side, then see if Bob's density matrix changes. Or have Alice measure her qubit, and see if *that* changes Bob's density matrix changes.

Note that if we condition on the *outcome* of Alice's measurement, then we do need to update Bob's density matrix to reflect the new knowledge: if Alice sees i then Bob sees j , etc. But that's not terribly surprising, since the same would also be true even with classical correlation! In particular, this doesn't provide a mechanism for faster-than-light communication.

To review, we've seen three different types of states in play, each more general than the last:

- Basis States, or Classical States
 - exist in a computational basis $|i\rangle$
- Pure States
 - superpositions of basis states $|\Psi\rangle = \sum \alpha_i |i\rangle$
- Mixed States
 - classical probability distributions over pure states $\rho = \sum \rho_i |\Psi_i\rangle \langle \Psi_i|$

Which represents the actual physical reality: pure or mixed states?

It's complicated. Sometimes we use density matrices to represent our probabilistic ignorance of a pure state. But when we look at part of an entangled state, a mixed state is the most complete representation possible that only talks about the part that we're looking at.

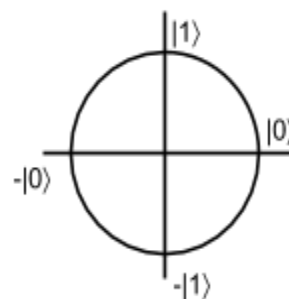
We'll generally just focus on what these representations are useful for.

Lecture 7, Tues Feb 7: Bloch Sphere,

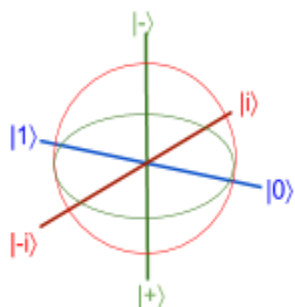
No-Cloning, Wiesner's Quantum Money

The Bloch Sphere

is a geometric representation of all possible states of a qubit.
We've often drawn the state of qubits as a circle, which is already a little awkward: half of the circle is going to waste since $|0\rangle = -|0\rangle$ (both represent the same density matrix).



Instead, what if vectors that pointed in *opposite* directions were orthogonal?
We get the Bloch Sphere...



We can see that $|+\rangle$ and $|-\rangle$ should be between $|0\rangle$ and $|1\rangle$. Then we can add $|i\rangle$ and $| - i\rangle$ as a new dimension.

In this representation, points on the surface of the sphere are pure states, such that
if they're 180° apart, they're orthogonal,
and if they're 90° apart, they're conjugate.

What about mixed states?

Well we know that the maximally mixed state, $\frac{I}{2}$, can be defined as $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$, $\frac{|+\rangle+|-\rangle}{\sqrt{2}}$, or $\frac{|i\rangle+|-i\rangle}{\sqrt{2}}$.
The sum of any two of these vectors on the sphere is the origin.

We can in this way represent any mixed state as a point inside of the sphere.

The mixture of any states $|v\rangle$ and $|w\rangle$, represented as points on the surface of the sphere, will be a point on the line segment connecting the two.

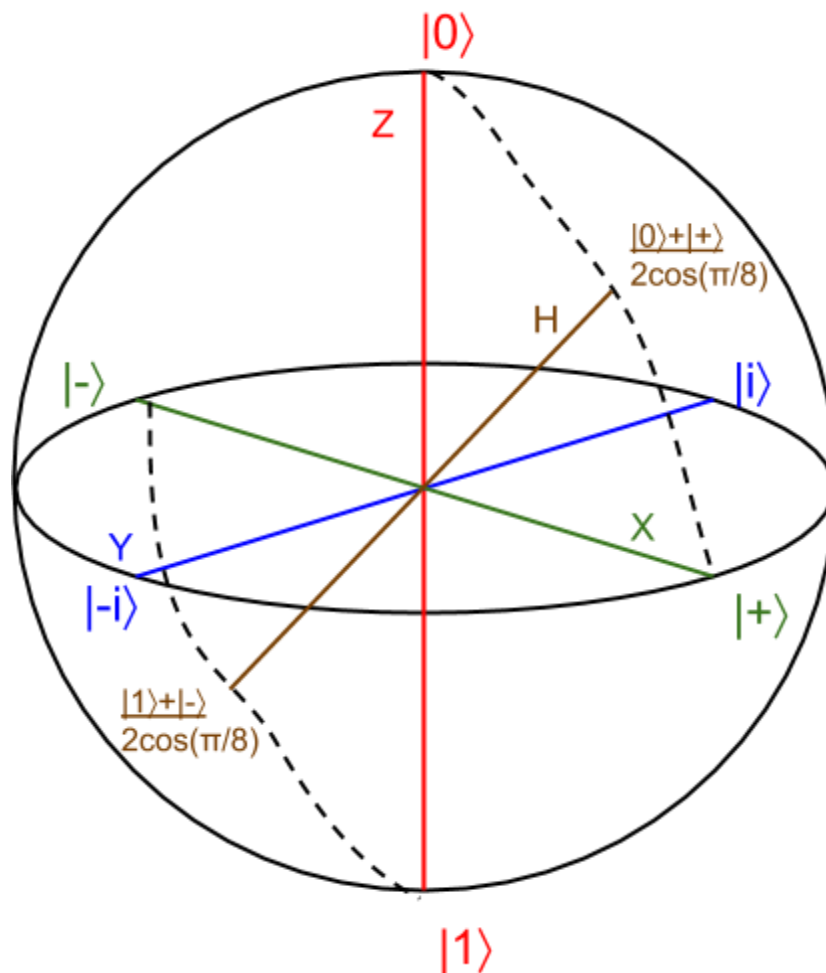
We can show geometrically that every mixed state can be written as a mixture of only two pure states. Why? Because you can always draw a line that connects any pure state you want to some point in the sphere representing a mixed state, and then see which other pure state that the line intersects on its way out. By some vector math, the point can be described as some linear combination of the vectors representing the pure states.

Experimentalists love the Bloch sphere, because it works almost identically to how spin works with electrons and other spin- $\frac{1}{2}$ particles.

With these things, you can measure the particle's "spin"—a qubit attached to the particle, basically—relative to any axis of the sphere. You see if the electron is spinning clockwise or

counterclockwise relative to the axis. And it behaves just like a qubit, in that the measurement collapses a more complex behavior into a binary result.

The weird part about spin- $\frac{1}{2}$ particles is that you *could have* asked the direction of the spin relative to any other axis. So what's really going on: what's the real spin direction? Well, the actual state is just some point on the Bloch sphere. So if the state of the electron is that it's spinning clockwise around the $(1, 0, 0)$ axis, we can say that it's in the $|0\rangle$ state, and if it's spinning clockwise around the $(0, 1, 0)$ axis, we can say that it's in the $|+\rangle$ state, and so forth. The crazy part here is how the three-dimensionality of the Bloch sphere “perfectly syncs up” with the three-dimensionality of actual physical space.



Visualizing the actions of gates on the Bloch sphere:

Applying gates X, Y, Z or H is the same as doing a half turn on their respective axis.

S corresponds to a quarter turn around Z . [in the $|+\rangle$ to $|1\rangle$ direction]

$T^2 = S$, so T corresponds to an eighth turn around Z .

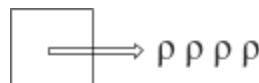
$R_{\frac{\pi}{4}}$ corresponds to a quarter turn (i.e. $\frac{\pi}{4}$) on Y .

The No-Cloning Theorem

We've seen how entanglement seems to lead to "non-local effects," like for the state $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$, where if Alice measures her qubit then she learns the state of Bob's. The reason that Alice isn't communicating faster than light boils down to Bob not being able to tell if his qubit's state is in the $|0\rangle$, $|1\rangle$ basis or the $|+\rangle$, $|-\rangle$ basis.

But what if Bob could make unlimited copies of his qubit? He could figure it out through repeated measurements, and so he'd be able to tell what basis Alice measured in. Faster than light communication!

Learning a classical description of a quantum state, given lots of copies of the state, is called **Quantum State Tomography**,

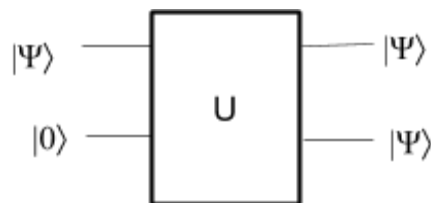


It turns out that we can prove that a procedure to reliably copy an unknown quantum state cannot exist. It's fairly easy to prove, but it's a fundamental fact about quantum mechanics.

In effect, we already saw one proof: namely, cloning would imply superluminal communication, which would violate the No-Communication Theorem that you proved in the homework! But let's see more directly why cloning is impossible.

Let's try to clone a single qubit, $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$

In our quantum circuit we want to apply some unitary transformation that takes $|\Psi\rangle$ and a $|0\rangle$ ancilla as input, and produces two copies of $|\Psi\rangle$ as output.



Algebraically, a cloner would need to do:

$$(\alpha|0\rangle + \beta|1\rangle) \otimes |0\rangle \rightarrow (\alpha|0\rangle + \beta|1\rangle) \otimes (\alpha|0\rangle + \beta|1\rangle) \\ = \alpha^2|00\rangle + \alpha\beta|01\rangle + \alpha\beta|10\rangle + \beta^2|11\rangle$$

The cloner would need to look like:

$$\begin{bmatrix} \alpha^2 \\ \alpha\beta \\ \alpha\beta \\ \beta^2 \end{bmatrix} = \begin{bmatrix} & & & \\ & U & & \\ & & & \\ & & & \end{bmatrix} \begin{bmatrix} \alpha \\ \alpha \\ \alpha \\ \alpha \end{bmatrix}$$

The problem: this transformation **isn't linear** so it can't be unitary!

To clarify, a procedure that outputs some $|\Psi\rangle$ can be rerun to get $|\Psi\rangle$ repeatedly. What the No Cloning Theorem says is that if $|\Psi\rangle$ is given to you but is otherwise unknown, then you can't make a copy of it.

Another clarification:

cNOT seems like a copying gate [as it maps $|00\rangle \rightarrow |00\rangle, |10\rangle \rightarrow |11\rangle$]

So why doesn't it violate the No Cloning Theorem?

Because it only copies if the input state is $|0\rangle$ or $|1\rangle$.

Classical information CAN be copied. Just ask Richard Stallman!



Doing cNOT on $|+\rangle$ produces the Bell Pair: $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$. Which sort of copies the first

qubit in an entangled way, but that's different making a copy of $|+\rangle$.

Having two qubits be $\frac{I}{2}, \frac{I}{2}$ is not the same as $|+\rangle, |+\rangle$.

In general, for any orthonormal basis you can clone the basis vectors, if you know that your input state is one of them.

Since the No Cloning Theorem is so important, we'll present another proof of it:

A unitary transformation can be defined as a linear transformation that preserves inner product. Which is to say that the angle between $U|v\rangle$ and $U|w\rangle$ is the same as the one between $|v\rangle$ and $|w\rangle$.



$$\text{Thus } \langle w|U^\dagger U|v\rangle = \langle w|v\rangle.$$

What would a cloning map do to this inner product?

$$\text{Let } |\langle v|w\rangle|^2 = c$$

$$\text{Then } |(\langle v| \otimes \langle v|)(|w\rangle \otimes |w\rangle)|^2 = c^2$$

c only ever equals c^2 if the inner product is 0 or 1: so the transformation can only copy if v and w belong to the same orthonormal basis.

There's a fact in classical probability that provides a nice analog to the No-Cloning Theorem.

If we're given the outcome of a coin flip—from a coin that lands heads with some unknown probability —can we simulate a second, independent flip of the same coin, without having access to the coin?

$$\text{You'd need } \begin{bmatrix} p^2 \\ p(1-p) \\ p(1-p) \\ (1-p)^2 \end{bmatrix} = \begin{bmatrix} & \\ S & \end{bmatrix} \begin{bmatrix} p \\ 1-p \\ 0 \\ 0 \end{bmatrix} \text{ to be true for some stochastic matrix } S.$$

But once again, this transformation isn't stochastic, because it's not linear.

The No Cloning Theorem has all sorts of applications to science fiction, because you can't make arbitrary copies of a physical system (say for teleporting yourself) if any of the relevant information (say, in your brain) were encoded in quantum states that didn't belong to a known orthogonal basis.

Quantum Money

is an application of the No Cloning Theorem. In some sense it was the first idea in quantum information, and was involved in the birth of the field. The original quantum money scheme was proposed by Wiesner in 1969, though it was only published in the 80s.

Wiesner had left research by then, and had become a manual laborer.

Wiesner realized that the quantum No-Cloning Theorem--though it wasn't yet called that--could be useful to prevent counterfeiting of money. In practice, mints use special ink, watermarks, etc., but all such devices basically just lead to an arms race with the counterfeiters. So Wiesner proposed using qubits to make money that would be physically impossible to counterfeit.

The immediate problem is that a money scheme needs not only *unclonability* but also *verifiability*. How did Wiesner solve this problem?

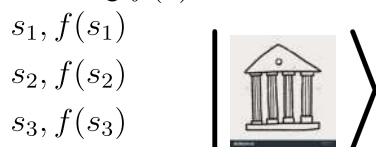
Wiesner's Scheme

The bank prints quantum bills (we'll assume for simplicity that they're all same denomination). Each bill has:

- A classical serial number $s = \{0, 1\}^n$
- A quantum state $|\Psi_{f(s)}\rangle$ (of n qubits)
 - The qubits in this state are unentangled, and each will always be in one of four states:

$$|\Psi_{00}\rangle = |0\rangle \quad |\Psi_{01}\rangle = |1\rangle \quad |\Psi_{10}\rangle = |+\rangle \quad |\Psi_{11}\rangle = |-\rangle$$

The bank maintains a giant database that stores, for each bill in circulation, the classical serial number s , as well as a string $f(s)$ that encodes what the quantum state attached to bill s is supposed to be.



Wiesner's scheme has an important practical problem though: you need to ensure that the qubits in a bill don't lose their state (coherence). With current technology, qubits in a lab decohere in like an hour, tops.

Qubits stored in a wallet would decohere much faster!

To verify a bill, you bring it back to the bank. The bank verifies the bill by looking at the serial number, and then measuring each qubit in the bill in the basis in which it was supposed to be prepared. E.g., if the qubit was supposed to be $|0\rangle$ or $|1\rangle$, then measure in the $\{|0\rangle, |1\rangle\}$ basis. For each measurement, check that you get the expected outcome.

Consider a counterfeiter who doesn't know which basis each qubit is supposed to be in, so they guess the bases uniformly at random. They only have a $(\frac{1}{2})^n$ chance of making all n guesses correctly.

Of course one could imagine a more sophisticated counterfeiter---but it's possible to prove that, *regardless* of what the counterfeiter does, if they map a single input bill to two output bills, then the output bills will both pass verification with probability at most $(\frac{3}{4})^n$.

Wiesner didn't actually prove the security of this scheme at the time he proposed it.

Professor Aaronson asked about it on Stack Exchange a few years ago which prompted Molina, Vidick, and Watrous to write a paper that formally proved the scheme's security.

Lecture 8, Thurs Feb 9: More on Quantum Money, BB84 QKD

Guest Lecture by Supartha Podder

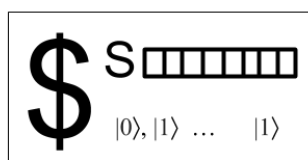
Continuation of Quantum Money

Last time we discussed how classical money is copyable and described a scheme for making money uncopyable through an application of the No-Cloning Theorem.

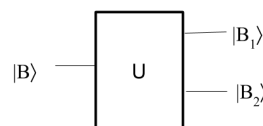
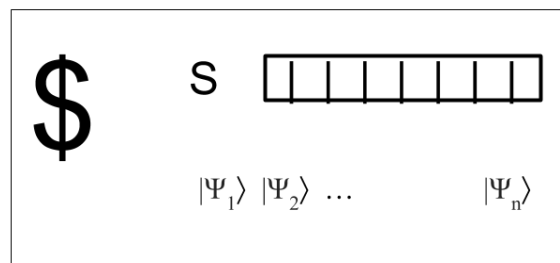
Let's consider a counterfeiter who wants to take a copy of a legitimate bill B and submit it for verification.

Say the counterfeiter decides to measure all qubits in the $|0\rangle, |1\rangle$ basis.

Their new bill becomes:



- s gets copied
 - (classical information)
- Puts $|0\rangle$ or $|1\rangle$ as each qubit.



So the bank will measure each qubit. The ones that should be in the $\{|0\rangle, |1\rangle\}$ basis are correct all of the time. But the ones that should be in the $\{|+\rangle, |-\rangle\}$ basis are correct in both bills only $\frac{1}{4}$ of the time.

Thus the probability that the counterfeiter succeeds (i.e., that both bills pass verification) is $(\frac{5}{8})^n$.

As we mentioned last time, it was recently shown that any such attack succeeds with probability at most $(\frac{3}{4})^n$.

Interactive Attack

There's a clever attack on Wiesner's scheme based around the assumption that verification involves giving the bank a bill, and then the bank returns the bill whether or not it passed verification.

We can start with a legitimate bill, then repeatedly go to the bank and ask them to verify it—but manipulating the qubits of the bill one at a time.

For example, if we set the first qubit to $|0\rangle$ and the bill still passes verification each and every time, then we've learned that the first qubit *should* be $|0\rangle$. Otherwise, we can successively try setting the first qubit to $|1\rangle$, $|+\rangle$, $|-\rangle$, and see which choice makes the bank consistently happy. Then, once we know, we move on to toggling the second qubit, and so on.

OK, but surely the bank wouldn't be so naïve as to return the bill even if it fails verification! We should assume instead that if verification fails (or fails often enough), then the bank alerts the police or something.

Can we come up with an attack that works even then? A recent paper by Nagaj and Sattath points out that we can!

Recall the **Elitzur Vaidman Bomb**. The general idea is that by making a succession of measurements, none of which reveals that much by itself, we can with a high probability of success learn whether a system is a certain state, without triggering a “bad event” that would happen if the system were actually measured to be in that state (such as a bomb going off). Applying a similar idea to quantum money gives us an...

Attack Based on the Elitzur Vaidman Bomb

Set $|c\rangle$ to $|0\rangle$

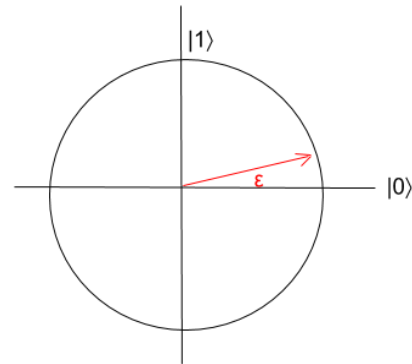
Let $|\Psi\rangle$ be the qubit of the banknote we’re trying to learn

Repeat $\frac{\pi}{2\varepsilon}$ times:

Apply the rotation R_ε to $|c\rangle$

Apply a cNOT gate to $|c\rangle|\Psi\rangle$

Send the bill to the bank for verification.



Suppose $|\Psi\rangle = |0\rangle$. Then each time we apply cNOT, we get

$$(\cos \varepsilon |0\rangle + \sin \varepsilon |1\rangle)|0\rangle = \cos \varepsilon |00\rangle + \sin \varepsilon |11\rangle$$

Most of the time $|c\rangle$ will stay at $|0\rangle$.

At each step, the probability of getting caught (i.e. failing verification) is $\sin^2 \varepsilon$.

Thus Prob[getting caught at all] is upper-bounded by $\leq \frac{\pi}{2\varepsilon} \sin^2 \varepsilon = O(\varepsilon)$

A similar analysis can be done if $|\Psi\rangle$ is $|1\rangle$ or $|-\rangle$: we’re unlikely to get caught, *and* the $|c\rangle$ qubit keeps “snapping back” to $|0\rangle$.

But if $|\Psi\rangle = |+\rangle$, then something different happens: the $|c\rangle$ qubit gradually rotates from $|0\rangle$ to $|1\rangle$.

So when we measure at the end, we can distinguish $|+\rangle$ from the other states, because it’s the only one that causes the $|c\rangle$ qubit to rotate to $|1\rangle$.

By symmetry, we can give analogous procedures to recognize the other three possible states for $|\Psi\rangle$.

So then we just iterate over all n qubits in the bill, learning them one by one, just like in the previous attack on Wiesner’s scheme.

Can Wiesner’s scheme be fixed to patch this vulnerability?

Yes! The bank can just give the customer a *new* bill (of the same value) after each verification, instead of the bill that was verified.

There’s an additional problem with Wiesner’s scheme, as we’ve seen it. Namely, it requires the bank to hold a huge amount of information: one secret for every bill in circulation. However, the paper (Bennett Brassard Breidbart Wiesner 82) points out how to circumvent this, by basically saying: let f be a pseudorandom function with a secret key k , so that for any serial number s , the bank can compute $f_k(s)$ for itself, rather than needing to look it up.

Of course the bank had better keep k itself secret: if it leaks out, the entire money system collapses! But assuming that k remains a secret, why is this secure?

We use a reduction argument. Suppose that the counterfeiter can copy money by some means. What does that say about f_k ? If it were truly random, then the counterfeiter wouldn't have succeeded. So by checking whether the counterfeiter succeeds, we can distinguish f_k from a random function. So f_k wasn't very good at being pseudorandom!

Note that with this change, we give up on provable security, of the sort we had with Wiesner's original scheme. Now we "only" have security assuming that f_k is computationally intractable to distinguish from random. (And a recent result by Prof. Aaronson shows that some computational assumption is necessary, if we don't want the bank to have to store a giant database.)

However, even after we make the improvements above, Wiesner's scheme still has a fundamental problem, which is that to verify a bill, you need to go to the bank. And if you have to go to the bank, then arguably you might as well have used a credit card or something instead! The point of cash is supposed to be that we don't need a bank to complete a transaction. Which brings us to...

Public-Key Quantum Money

This is quantum money that *anyone* can verify using a "public key," but that can only be produced or copied using a "private key" known only to the bank.

For formal definitions see (Aaronson 2009), (Aaronson, Christiano 2012).

With this sort of scheme, you'll *always* need computational assumptions on the counterfeiter, in addition to quantum mechanics. Why? Because a counterfeiter with infinite computational power could always just try *every* possible quantum state (or an approximation thereof) on the appropriate number of qubits, until it found one that made the public verification procedure accept.

Quantum Key Distribution

Now we'll discuss something closely related to quantum money, but that doesn't require storing quantum states for long times—and that, for that reason, is actually practical today (though so far there's only a small market for it).

Key distribution is a fundamental task in cryptography. It just means causing two agents, Alice and Bob, to share a secret key (without loss of generality, a uniformly random string), when they didn't have one before.

Once Alice and Bob share a long enough key, they can then exchange secret messages, using the central technique in cryptography called the **One-Time Pad**.

Given a shared key $k \in \{0, 1\}^n$

Alice has a secret message $m \in \{0, 1\}^n$

Alice sends the ciphertext $c = m \oplus k$, where \oplus denotes bitwise XOR

Bob decodes the message m as $m = c \oplus k$

As its name implies, the One-Time Pad can only be used once securely with a given key k , so it requires a large amount of sharing of keys. In fact, in the classical world, it's been proven that if they want to communicate securely, Alice and Bob either need initial secret information in common, or else they must make computational assumptions on the eavesdropper Eve.

The great discovery of Quantum Key Distribution was that quantum mechanics lets us get encryption with no computational assumptions! (But we do need communication channels capable of sending quantum states.)

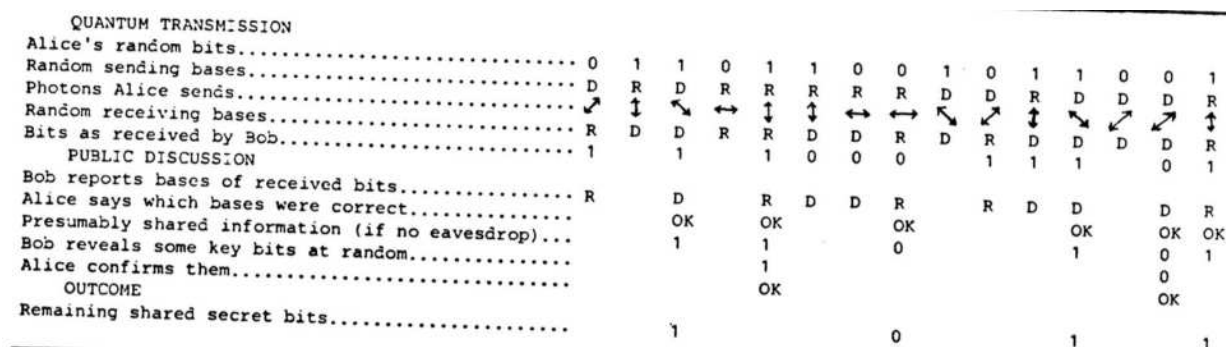
In cryptography, besides secrecy, an equally important goal is authentication. However, we're only going to deal with secrecy.

BB84

We'll describe the BB84 scheme, the first full quantum key distribution scheme. This scheme was proposed by Bennett and Brassard in 1984, though it was partly anticipated in Wiesner's paper (the same one that introduced quantum money!). It circumvents the issues we've seen in maintaining a qubit, because it only requires coherence for the time it takes for communication between Alice and Bob.

There are companies that are already doing quantum key distribution through fiber optic cables over up to about 10 miles. In addition, just this year a team from China demonstrated QKD over distances of thousands of miles, by sending photons to and from a satellite that was launched into space for that express purpose.

Here's a diagram from the original paper that shows how BB84 works.



The basic idea is that you're trying to establish some shared secret knowledge and you want to know for certain that no eavesdroppers on the channel can uncover it. You've got a channel in which to transmit quantum information, and a channel in which to transmit classical information. In both, eavesdroppers may be able to listen in (no secrecy). But in the classical channel, we'll assume you at least have *authenticity*: Bob knows that any messages really come from Alice and vice versa.

- So Alice chooses a string x of random bits $\in \{0, 1\}^n$
- And another string y of random bits $\in \{0, 1\}^n$, which she uses to decide which basis to encode each bit from x in.
- She then encodes each bit of x in the $\{|0\rangle, |1\rangle\}$ basis (in the diagram it's R), if the corresponding bit of y is 0, or the $\{|+\rangle, |-\rangle\}$ basis (D), if the corresponding bit of y is 1
- Then she sends over the qubits to Bob.
- Bob picks his own random string $y' \in \{0, 1\}^n$ and uses y'_i to decide in which basis

to decode the i^{th} qubit sent over (picking again between D and R)

Now Alice and Bob share which bases they picked to encode and measure the bits of x (the strings y and y'). They discard any bits of x for which they didn't pick the same basis (which will be about half the bits).

At this point we consider an eavesdropper Eve who was watching the qubits as they were sent over. The whole magic of using qubits is that if Eve tries to measure the qubits, then she inherently changes what Bob receives! Sure, if she measures a $\{|0\rangle, |1\rangle\}$ qubit in the $\{|0\rangle, |1\rangle\}$ basis, then the qubit doesn't change. But what if she's unlucky, and measures a $\{|+\rangle, |-\rangle\}$ qubit in the $\{|0\rangle, |1\rangle\}$ basis? And eventually, she almost certainly *will* be unlucky.

In more detail: suppose Alice sent $|+\rangle$, then Eve measured $|0\rangle$ and passed that along to Bob. Then even if Bob measures in the $\{|+\rangle, |-\rangle\}$ basis (i.e., the "right" basis), he has a 50% chance of measuring $|+\rangle$ and a 50% chance of measuring $|-\rangle$. In the latter case, Alice and Bob will be able to see that the channel was tampered with.

So Alice and Bob can verify that no one listened in to their qubit transmission by making sure that some portion of their qubits that *should* match, do match. Of course, after Alice and Bob discuss those qubits over the channel, they aren't going to be secret anymore! But they've still got all the others.

If any of the qubits didn't match, then Alice and Bob deduce that Eve eavesdropped. So then they can just keep trying again and again until they can get a batch where no one listened in. At worst, Eve can prevent Alice and Bob from ever communicating by listening in constantly. But we can prevent a situation where Alice and Bob *think* their shared key is secure even though it isn't.

Again, once Alice and Bob share a secret key, they can then use some classical encryption scheme, like the One-Time Pad.

Lecture 9: Tues Feb 14: Superdense Coding

OK, now on to some new stuff!

Superdense Coding

is the first protocol we'll see that requires entanglement. Basic information theory (Shannon) tells us that "by sending n bits, you can't communicate more than n bits of information."

Now, by contrast, we'll see how Alice can send Bob *two* classical bits by sending him only *one* qubit, though there is a catch: Alice and Bob must share some entanglement ahead of time.

In the scenario with no prior entanglement, Alice can't send more than one bit per qubit—a fundamental result known as *Holevo's Theorem*.

We're not going to prove Holevo's Theorem here, but the intuition is pretty simple: if Alice sends $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$ to Bob, he can only measure it once in some basis and then the rest of the information in $|\Psi\rangle$ is lost.

Instead, let's suppose that Alice and Bob share a Bell pair in advance: $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$

We claim that Alice can manipulate her half, then send her half to Bob, and then Bob can measure both qubits and get two bits of information from Alice.

The key is to realize that Alice can get three different states, all of them orthogonal to the original Bell pair and to each other, by applying the following gates to her qubit:

- NOT $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ which gives us $\frac{|01\rangle + |10\rangle}{\sqrt{2}}$
- A phase change $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ which gives us $\frac{|00\rangle - |11\rangle}{\sqrt{2}}$
- And applying both NOT and a phase change, which gives us $\frac{|01\rangle - |10\rangle}{\sqrt{2}}$

These four states form an orthogonal basis.

So suppose Alice wants to transmit two bits X , and Y :

If $X = 1$, she applies the NOT gate.

If $Y = 1$, she applies a phase gate.

Then she sends her qubit to Bob.

We can derive her encoding matrix as: $\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & -1 & 0 & 0 \end{bmatrix}$

Which makes sense, because each column corresponds to one of the four states we listed above.

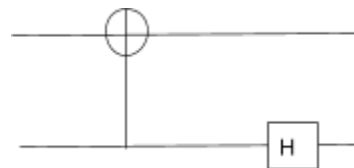
(e.g. the second column corresponds to $\frac{|00\rangle - |11\rangle}{\sqrt{2}}$)

For Bob to decode this transformation, he'll want to use the inverse transformation:

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

Which corresponds to the circuit:

cNOT (2nd controls 1st)
then Hadamard on the 2nd qubit



So, Alice transforms the Bell pair into one of the four orthogonal states above, then Bob decodes that two-qubit state into one of the four possible combinations of $|0\rangle$ and $|1\rangle$, corresponding to the original bits X and Y .

For example:

if Bob receives $\frac{|01\rangle - |10\rangle}{\sqrt{2}}$, applying cNOT gets him $|1\rangle \otimes |-\rangle$, and Hadamard gets him $|1\rangle \otimes |1\rangle$.

if Bob receives $\frac{|00\rangle - |11\rangle}{\sqrt{2}}$, applying cNOT gets him $|0\rangle \otimes |+\rangle$, and Hadamard gets him $|0\rangle \otimes |1\rangle$.

Naturally, we could ask: if Alice and Bob had even more pre-shared entanglement, could Alice send an arbitrarily large amount of information by transmitting only one qubit?

There's a theorem that says: No.

It turns out that for every qubit, and any amount of entangled qubits (ebits), you can send two bits of classical information, but no more. I.e., we can write the inequality:

$$1 \text{ qubit} + \text{ebits} \geq 2 \text{ bits}$$

but **not**

$$1 \text{ qubit} + \text{ebits} \geq 3 \text{ bits}$$

As far as quantum speed-ups go, a factor of two isn't particularly impressive, but it is pretty cool that it challenges the most basic rules of information theory established by Shannon himself.

Lecture 10, Thurs Feb 16: Teleportation, Entanglement Swapping, GHZ, Monogamy

Next let's see...

Quantum Teleportation

which is a result from 1991 that came as a great surprise. Science journalists still love it given its irresistible name. In this lecture we'll see what it can and can't do.

Firstly, what does teleportation mean?

You might think it implies sending qubits instantaneously over vast distances, but that can't be done, as it violates the causal structure of the universe. So we're only going to send qubits at the speed of light, no faster. Of course, there are other ways to move qubits at the speed of light or slower, like just picking them up and moving them, or putting them on a bus! (It doesn't sound as sexy that way.)

OK, but what if you only had a phone line, or a standard Internet connection? That would let you send classical bits, but not qubits. With teleportation, though, we'll achieve something surprising. We'll show:

It is possible for Alice and Bob to use pre-shared entanglement plus classical communication to perfectly transmit a qubit.

The inequality here is almost the converse of the one for superdense coding:

$$1 \text{ ebit} + 2 \text{ bits} \geq 1 \text{ qubit}$$

Which is to say, you need one pair of entangled qubits plus two classical bits in order to transmit one qubit. (This can also be shown to be optimal.)

We'll give a more in-depth explanation in the next lecture, but the gist of it is:

Alice has, say, a single qubit, $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$. She also shares a Bell pair with Bob.

Alice applies some transformation to $|\Psi\rangle$ that entangles it with her half of the Bell pair. She then measures her qubits.

Alice tells Bob the measurement outcomes over the phone.

Bob applies some transformations (to his qubit of the entangled pair), based on what he hears from Alice. "Magically," Bob now has $|\Psi\rangle$

At the end, will Alice also have $|\Psi\rangle$?

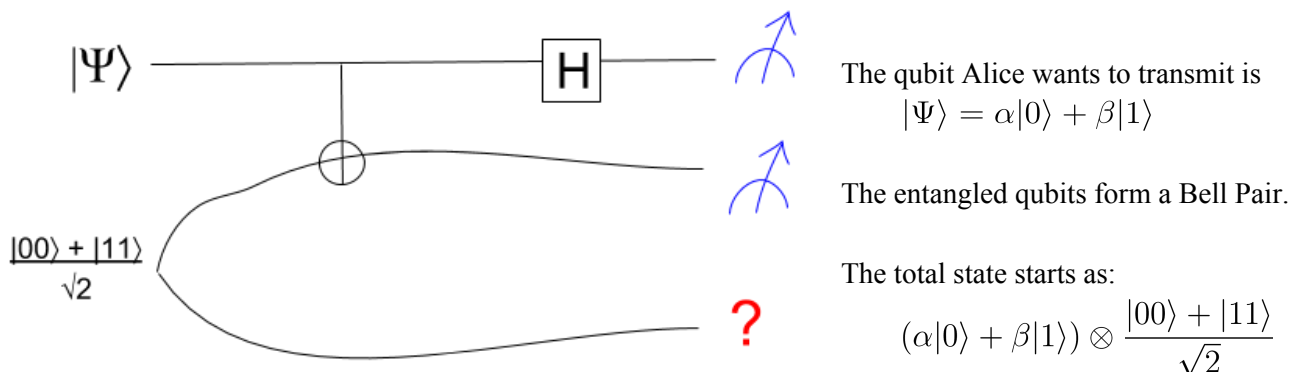
No. A logical consequence of the No Cloning Theorem is that there can only be one copy of the qubit.

Could we hope for a similar protocol *without* sending classical information?

No, because of the No-Communication Theorem.

So let's say Alice wants to get a qubit over to Bob, *without* using a quantum communication channel, but *with* a classical channel together with preshared entanglement. How should Alice go about this?

Once the question is posed, you can play around with different combinations of operations, and you'd eventually discover that what works is this:



Then Alice applies a cNOT gate (with $|\Psi\rangle$ as the control, and her half of the Bell pair as the target):

$$\frac{1}{\sqrt{2}}[\alpha|000\rangle + \alpha|011\rangle + \beta|110\rangle + \beta|101\rangle]$$

Alice then Hadamards her $|\Psi\rangle$ qubit:

$$\frac{1}{2}[\alpha|000\rangle + \alpha|100\rangle + \alpha|011\rangle + \alpha|111\rangle + \beta|010\rangle - \beta|110\rangle + \beta|001\rangle - \beta|101\rangle]$$

Finally, Alice measures both of her qubits in the $\{|0\rangle, |1\rangle\}$ basis.

This leads to four possible outcomes:

If Alice Sees	00	01	10	11
Then Bob's qubit is	$\alpha 0\rangle + \beta 1\rangle$	$\alpha 1\rangle + \beta 0\rangle$	$\alpha 0\rangle - \beta 1\rangle$	$\alpha 1\rangle - \beta 0\rangle$

We're deducing information about by Bob's state by using the partial measurement rule. E.g., if Alice sees 00, then we narrow down the state of the entire system to the possibilities that fit, namely $|000\rangle$ and $|001\rangle$.

What is Bob's state, if he knows that Alice measured, but doesn't know the measurement outcome?

It's an equal mixture of all four possibilities, which is just the Maximally Mixed State.

This makes sense given the No-Communication Theorem! Until Alice sends information over, Bob's qubit can't possibly depend on $|\Psi\rangle$.

Next, Alice tells Bob her measurement results via a classical channel. And Bob uses the information to "correct" his qubit to $|\Psi\rangle$.

If the first bit sent by Alice is 1, then Bob applies $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

If the second bit sent by Alice is 1, then Bob applies $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

These transformations will bring Bob's qubit to the state $\alpha|0\rangle + \beta|1\rangle = |\Psi\rangle$.

That means they've successfully transmitted a qubit without a quantum channel!

This protocol never assumed that Alice knew what was $|\Psi\rangle$.

For this protocol to work, Alice had to measure her **syndrome** bits. These measurements were destructive (since we can't ensure that they'll be made in a basis orthonormal to $|\Psi\rangle$, and thus Alice doesn't have $|\Psi\rangle$ at the end. Alice and Bob also "use up" their Bell pair in the process of teleporting $|\Psi\rangle$.

Something to think about: Where is $|\Psi\rangle$ after Alice's measurement, but before Bob does his operations?

How do people come up with this stuff? I can't picture how anyone trying to solve this problem would even begin their search...

Well it's worth pointing out that quantum mechanics was discovered in 1926 and that quantum teleportation was only discovered in the 90's. These sorts of protocols *can* be hard to find. Sometimes someone tries to prove that something is impossible, and in doing so eventually figures out a way to get it done...

Aren't we fundamentally sending infinitely more information than two classical bits if we've sent over enough information to perfectly describe an arbitrary qubit, since the qubit's amplitudes could be arbitrary complex numbers?

In some sense, but at the end of the day, Bob only really obtains the information that he can measure, which is significantly less. Amplitudes may "exist" physically, but they're different from other physical quantities like length, in that they seem to act a lot more like probabilities.

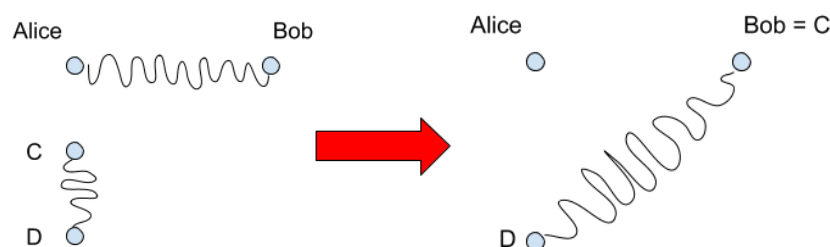
Like, there's a state $\alpha|0\rangle + \beta|1\rangle$, of a single qubit, such that β is a binary encoding of the complete works of Shakespeare—the rules of quantum mechanics don't put a limit on the amount of information that it takes to specify an amplitude. With that said, we could also encode the complete works of Shakespeare into the probability that a classical coin lands heads! In both cases, the works of Shakespeare wouldn't actually be retrievable by measuring the system.

If we can teleport one qubit, the next question we may want to ask is:

Can we go further? What would it take to teleport an arbitrary quantum state, say of n qubits?

To answer this question, let's notice that nothing said that a qubit that's teleported has to be unentangled with the rest of the world.

You could run the protocol and have $|\Psi\rangle$ be half of another Bell pair. That would entangle the fourth qubit to Bob's qubit (you can check this via calculation). That's not a particularly interesting operation, since it lands you where you started, with one qubit of entanglement between Alice and Bob, but it does have an interesting implication.

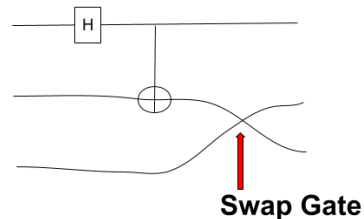


It suggests that it should be possible to teleport an arbitrary n -qubit entangled state, by simply teleporting the qubits one at a time, thus using n ebits of preshared entanglement. And indeed it's not hard to check that that works.

One further consequence of this is that two qubits don't need to interact directly to become entangled.

In some sense, we already knew that:

Consider for example the following circuit.



Here the first and third end up entangled, even though there's never "direct" contact between them: the second qubit serves as an intermediary.

What does it take for Alice and Bob to get entangled?

The obvious way is for Alice to create a Bell pair and then send one of the qubits to Bob.

In most practical experiments, the entangled qubits are created somewhere between Alice and Bob, then one qubit is sent to each.

However, teleportation leads to something much more surprising than this, called...

Entanglement Swapping



If Alice has two entangled qubits, and also two Bell pairs shared with Bob, she can teleport both of her qubits to Bob, whereupon they'll be entangled on Bob's end ... even though the two qubits on Bob's end, which are now entangled, were never in causal contact with one another!

This process has been used in real experiments, such as the recent "loophole-free Bell tests," about which we'll learn more later in the course.

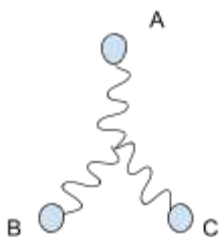
By the way, quantum teleportation itself has been demonstrated experimentally many times.

A few more comments on the nature of entanglement:

We've seen the Bell pair, and what it's good for. There's a 3-qubit analogue of it called the **GHZ state**: $\frac{|000\rangle + |111\rangle}{\sqrt{2}}$. We'll see applications of the GHZ state later in the course, but for now we'll use it to illustrate an interesting conceptual point.

Let's say that Alice, Bob, and Charlie hold random bits, which are either all 0 or all 1 (so, they're classically correlated). If all three of them get together, they can see that their bits are correlated, and *the same is true if only two of them are together*.

But now suppose the three players share a GHZ state. With all three of them, they can see that the state is entangled, but what if Charlie is gone? Can Alice and Bob see that they're entangled with each other?



No. To see this, observe that by the No-Communication Theorem, Charlie could've measured without Alice and Bob knowing. But if he did, then Alice and Bob would clearly have classical correlation only: either both 0's (if Charlie got the measurement outcome 0) or both 1 (if Charlie got 1). From this it follows that Alice and Bob have only classical correlation *regardless of whether Charlie measured or not*.

A different way to see this is to look at the density matrix of the state shared by Alice and Bob:

$$\rho_{AB} = \begin{bmatrix} \frac{1}{2} & & \\ & & \\ & & \frac{1}{2} \end{bmatrix} \quad (\text{all blank entries are } 0)$$

And notice that this is different than the density matrix of a Bell pair shared by Alice and Bob

$$\rho_{AB} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} = |\Psi\rangle\langle\Psi|$$

Where $|\Psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$

This is one illustration of a general principle called...

The Monogamy of Entanglement

Simply put, this means that if Alice has a qubit that is maximally entangled with Bob, then that qubit can't also be maximally entangled with Charlie.

With GHZ, you can only see the entanglement if you have all three qubits together. This is sometimes analogized to the Borromean Rings (right), an arrangement of three rings with the property that all three are linked together, without any two of them being linked together.

There are other 3-qubit states which aren't like that...

In the W state, $\frac{1}{\sqrt{3}}(|100\rangle + |010\rangle + |001\rangle)$, there's some entanglement between Alice and Bob, and there's some entanglement between Alice and Charlie, but neither pair is maximally entangled.

As for how you quantify entanglement ... well, that will be the subject of the next lecture!



Lecture 11, Tues Feb 21: Quantifying Entanglement, Mixed State Entanglement

How do you quantify how much entanglement there is between two quantum systems?

It's worth noting that we sort of get to decide what we think a measure of entanglement *ought* to mean. We've seen how it can be useful to think of entanglement as a resource, so we can phrase the question as "how many 'Bell pairs of entanglement' does a given state correspond to?"

A priori, there could be different, incomparable kinds of entanglement that are good for different things. And that's actually the case for entangled mixed states, or entangled pure states shared by three or more parties. But for the special case of an entangled pure state shared by two parties, Alice and Bob, it turns out that there's a single measure of entanglement, which counts "the number of Bell pairs needed to form this state, and equivalently the number that can be extracted from it."

So, given $\sum_{ij} \alpha_{ij} |i\rangle_A |j\rangle_B$, how do we calculate many Bell pairs it's worth?

Our first observation here is that given any bipartite state, you can always find a change of basis on Alice's side, and another change of basis on Bob's side, that puts the state into the simpler form

$$\sum \lambda_i |v_i\rangle |w_i\rangle,$$

where all $|v_i\rangle$'s are orthonormal, and all $|w_i\rangle$'s are also orthonormal. To put the state into this form, we use a tool from linear algebra called...

Schmidt Decomposition

Given a the matrix $A = \begin{bmatrix} \alpha_{11} & & \alpha_{1n} \\ & \ddots & \\ \alpha_{n1} & & \alpha_{nn} \end{bmatrix}$ representing the entire quantum state.

We can multiply A by two unitary matrices, one on each side, to get a diagonal matrix:

$$UAV = \Lambda$$

U and V can be found efficiently using linear algebra

U and V represent the changes of basis that Alice and Bob respectively would need to apply, in order to get their state into the Schmidt form

$$\sum \lambda_i |v_i\rangle |w_i\rangle.$$

Measuring in the $\{|v_i\rangle, |w_i\rangle\}$ basis would then yield the probability distribution

$$\begin{bmatrix} |\lambda_1|^2 \\ \vdots \\ |\lambda_n|^2 \end{bmatrix}$$

Now, recall that, for a classical probability distribution $D = (p_1, \dots, p_n)$, its **Shannon entropy** is

$$H(D) = \sum_{i=1}^n P_i \log_2 \frac{1}{P_i}$$

So now we just need to calculate the ordinary Shannon entropy of our probability vector,

$$\sum_i |\lambda_i|^2 \log \frac{1}{|\lambda_i|^2},$$

in order to figure out how many Bell pairs our state is equivalent to.

To come at it a bit differently: there's a measure called **von Neumann entropy**, which generalizes Shannon entropy from classical probability distributions to quantum mixed states. We say that the von Neumann entropy of a mixed state ρ is

$$S(\rho) = \sum_{i=1}^n \lambda_i \log_2 \frac{1}{\lambda_i}$$

You could also say that von Neumann entropy *is* the Shannon entropy of the vector of eigenvalues of the density matrix of ρ . If you diagonalize the density matrix, the diagonal represents a probability distribution over n possible outcomes, and taking the Shannon entropy of *that* distribution gives you the von Neumann entropy of your quantum state.

Yet another way to think about it:

Say you looked at all the possible probability distributions, that could arise by measuring the mixed state ρ in all possible orthogonal bases. Then the von Neumann entropy of ρ is the *minimum* of the Shannon entropies of all those distributions.

$$S(\rho) = \min_U \left\{ H(\text{diag}(U\rho U^\dagger)) \right\}$$

where $\text{diag}(A)$ means the length- n vector obtained from the diagonal of the $n \times n$ matrix A .

So the von Neumann entropy of any pure state $|\Psi\rangle\langle\Psi|$ is 0, because there's always some measurement basis (namely, a basis containing $|\Psi\rangle$) that returns a definite outcome.

You could choose to measure $|+\rangle$ in the $\{|0\rangle, |1\rangle\}$ basis and you'll have complete uncertainty, and an entropy of 1. But if you measure $|+\rangle$ in the $\{|+\rangle, |-\rangle\}$ basis, you have an entropy of 0, because you'll always get the outcome at $|+\rangle$.

So $S(|\Psi\rangle\langle\Psi|) = 0$.

By contrast, the von Neumann entropy of the maximally mixed state, $\frac{I}{2}$, is 1.

Similarly, the von Neumann Entropy of the n -qubit maximally mixed state is n .

We can now talk about how much *entanglement entropy* is in a bipartite pure state.

Entanglement Entropy

Suppose Alice and Bob share a bipartite pure state $|\Psi\rangle = \sum_{i,j} \alpha_{ij} |i\rangle_A |j\rangle_B$

To quantify the entanglement entropy, we'll trace out Bob's part, and look at the von Neumann entropy of Alice's side, $S(\rho_A)$, in effect asking: if Alice made an optimal measurement, how much could she learn about Bob's state?

$$S(\rho_A) = S(\rho_B) = H\{\lambda_i\}$$

↑ This is the Shannon entropy of the vector of eigenvalues, which you can get by diagonalizing Alice's (or Bob's) density matrix, *or* by putting $|\Psi\rangle$ in Schmidt form, as we did in the previous lecture.

The entanglement entropy of any product state, $|\Psi\rangle \otimes |\Phi\rangle$, is 0.

The entanglement entropy of a Bell pair, $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$, is 1.

You can think of entanglement entropy as either:

- The number of Bell pairs it would take to create the state
- The number of Bell pairs that you can extract from the state

It's not immediately obvious that these two values are the same, but for pure states, they are.
(For mixed states, they need not be!)

A sample calculation...

Let $|\Psi\rangle = \frac{3}{5}|0\rangle_A|+\rangle_B + \frac{4}{5}|1\rangle_A|-\rangle_B$

This state is already in Schmidt form
(otherwise, we'd have to put it in that form)

Then the entanglement entropy is

$$E = \left(\frac{3}{5}\right)^2 \log_2\left(\frac{5}{3}\right)^2 + \left(\frac{4}{5}\right)^2 \log_2\left(\frac{5}{4}\right)^2$$

$$= \sim .942$$

This means that if Alice and Bob shared 1000 copies of $|\Psi\rangle$, they'd be able to teleport about 942 qubits.

For bipartite *mixed* states, by contrast, there are two values to consider:

The Entanglement of Formation

$$E_F(\rho_{AB})$$

is the number of ebits that Alice and Bob need to create one copy of the state ρ_{AB} , in the limit where they're creating many copies of it, and assuming they're allowed unlimited local operations and classical communication (called "LOCC" in the lingo) for free

The Distillable Entanglement

$$E_D(\rho_{AB})$$

is the number of ebits that Alice and Bob can extract per copy of ρ_{AB} , again in the limit where they're given many copies of it, and assuming local operations and classical communication are free

Clearly $E_F \geq E_D$, since if you could ever get out more entanglement than you put in, it would give you a way to increase entanglement arbitrarily using LOCC, which is easily seen to be impossible. But what about the other direction?

It turns out that there exist bipartite pure states for which $E_F \gg E_D$, which is to say that those states take a lot of entanglement to make, but then you can only extract a small fraction of the entanglement that you put in. We won't have time to explain this in more detail.

We call a bipartite mixed state ρ_{AB} *separable* if there's any way to write it as a mixture of product states:

$$\rho_{AB} = \sum_i p_i |v_i\rangle\langle v_i| \otimes |w_i\rangle\langle w_i|$$

A mixed state is called entangled if and only if it's not separable.

This is subtle: it sometimes happens that a density matrix looks entangled, but there's some weird decomposition that shows that no, actually it's separable.

And indeed, in 2003 Leonid Gurvits proved a pretty crazy fact:

If you're given as input a density matrix ρ_{AB} for a bipartite state, then deciding whether ρ_{AB} represents a separable or entangled state is an NP-hard problem!

As a result, unless $P = NP$, there can be no “nice characterization” for telling apart entangled and unentangled bipartite mixed states—in contrast to the situation with bipartite pure states.

This helps to explain why there are endless paper writing opportunities in trying to classify different types of entanglement...

Lecture 12, Thurs Feb 23: Interpretation of QM (Copenhagen, Dynamical Collapse, MWI, Decoherence)

At this point in the course, we're finally in a position to step back and ask, "What is quantum mechanics telling us about reality?" It should be no surprise that there isn't a consensus on this question (to put it mildly)! But regardless of your own views, it's important to know something about the various positions people have defended over the years, as the development of these positions has sometimes gone hand in hand with breakthroughs in quantum mechanics. (We'll see an example of this later, with the Bell inequality, and arguably quantum computing itself is another example.)

Most discussions about the implications of quantum mechanics for our understanding of reality center around the so-called Measurement Problem.

In most physics texts (and in this class, for that matter...), measurement is introduced as just a primitive operation that we don't try to understand more deeply. However, there's a fundamental weirdness about measurement in QM, which stems from the fact that the theory seems to demand both:

1. Unitary Evolution

when no one is watching

$$|\Psi\rangle \rightarrow U|\Psi\rangle$$

2. Measurement

which collapses a state to a single possibility

$$|\Psi\rangle \rightarrow i \text{ with probability}$$

$$|\langle\Psi|i\rangle|^2 = |\alpha_i|^2$$

In other words, quantum mechanics seems to work in a way that's deterministic, reversible, and continuous most of the time (1), *except* during measurement (2), which is the only time we see it work in a way that's probabilistic, irreversible, and sudden. So we can phrase the question as:

"How does the universe know when to apply unitary evolution and when to apply measurement?"

People have argued about this for about 100 years. The discussion is sometimes compared to the discussion about the nature of consciousness (which has gone on for millennia) in that they both tend to devolve into people talking in circles around each other.

But it's worth understanding the main schools of thought, starting with...

The Copenhagen Interpretation

This was the preferred interpretation of most of the founders of quantum mechanics. It's closely associated with Niels Bohr and his institute in Copenhagen (hence the name) and with Werner Heisenberg. Note that the different founders said different and sometimes contradictory things, sometimes in very abstruse language, so it's notoriously hard to pin down what the "Copenhagen interpretation" actually is!

Basically, though, the Copenhagen viewpoint is that there are two different worlds (or parts of reality): the quantum world and the classical world. We live in the classical world, where objects have definite locations, the objects can be measured without disturbing them, etc. But in doing experiments we've discovered that there also exists the quantum world "beneath" ours, which obeys very different rules.

Measurement, in this view, is the operation that bridges the two worlds.

It lets us "peek under the hood" into the quantum world and see what's going on.

Bohr wrote long tracts saying that even to make statements about the quantum world and the classical world is to presuppose that *is* a classical world in which those statements can be made. So there's some "boundary" or "cut" between the quantum and classical worlds. The exact location of this boundary might be fuzzy, and might vary depending on what sort of question we're asking. But in any case, we should never make the error of insisting that our commonsense, classical concepts remain valid on the quantum side of the boundary.

Believers in the Copenhagen interpretation love to say things like: "if this doesn't make sense to you, then you're just stuck in the old way of thinking, and you need to change. The problem is not with quantum mechanics, it's with you."

The next interpretation, which is closely related is...

S.U.A.C. : "Shut Up And Calculate!"

This is probably the preferred "interpretation" of most physicists, chemists, and others who work with quantum mechanics.

It says that at the end of the day, quantum mechanics works: it correctly predicts the results of experiments. And that's all we can reasonably ask of a scientific theory, or all that it's fruitful to ask.

Prof. Aaronson likes to say that the Copenhagen interpretation is basically just S.U.A.C. without the S.U. part! Copenhagen starts from the intuition that "it's pointless to philosophize about what this means," but then elevates *that* to a philosophy, which of course is a little ironic.

In any case, while the S.U.A.C. view has some obvious practical advantages, it seems clear that it can't satisfy people's curiosity forever. This is not only because science has always aspired to *understand what the world is like*, with experiments and predictions just a means to that end. A second reason is that, as experimenters become able to create ever larger and more complicated quantum superpositions—in effect, "breaching" the Bohr/Heisenberg boundary between the quantum and classical worlds—it becomes less and less viable to "quarantine" quantum mechanics as just a weird mathematical formalism that happens to work for predicting the behavior of electrons and photons. The more QM impinges on the world of everyday experience, the more it seems necessary to come to terms with whatever it says about that world.

This seems like a good time for a digression about two celebrated thought experiments, which were invented to probe exactly this "breaching"...

Schrödinger's Cat

There were physicists in the 20s and 30s who never accepted the Copenhagen interpretation, of whom the most famous were Einstein and Schrödinger. They came up with many examples to try to show just how untenable it is to have a rigid boundary between the quantum and classical worlds, if you think hard about it.

By far the most famous example is Schrödinger's Cat, which first appears with Einstein remarking in a letter that if you think of a pile of gunpowder as being inherently unstable, you could model it as a quantum state which looks like $|\text{pile}\rangle + |\text{explosion}\rangle$.

Then Schrödinger adds some flair by asking, “What happens if we create a quantum state that corresponds to a superposition of a state in which a cat is alive and one where the cat is dead?” He isolates the state from its external environment by putting it in a box $|\text{cat}\rangle + |\text{dead cat}\rangle$.

The point of the thought experiment is that the formal rules of quantum mechanics apply *whenever* you have distinguishable states, regardless of their size. In particular, they say that you can create arbitrary linear combinations of such states. But by the time we're talking about something as big as a cat, it seems patently obvious that we should have to say something about the nature of what's going on before measurement. Otherwise we'd devolve into extreme solipsism—saying, for example, that the cat only exists once we've opened the box to observe it.

Wigner's Friend

is a similar thought experiment. It says that Eugene Wigner—the physicist who proposed the experiment—could be put into an equal superposition of thinking one thought and thinking another one, which we model as

$$\frac{1}{\sqrt{2}}(|\text{Wigner}_0\rangle + |\text{Wigner}_1\rangle)$$

Now consider the joint state of Wigner and a friend who hasn't yet measured his state:

$$|\text{Friend}\rangle \otimes \frac{1}{\sqrt{2}}(|\text{Wigner}_0\rangle + |\text{Wigner}_1\rangle)$$

From Wigner's point of view, he's thinking one thought or the other one. But from his friend's point of view, he isn't thinking *either* of them until a measurement gets made. At that point we'll have an entangled state like

$$\frac{1}{\sqrt{2}}(|\text{Friend}_0\rangle|\text{Wigner}_0\rangle + |\text{Friend}_1\rangle|\text{Wigner}_1\rangle)$$

But then what happens if another friend comes along, and then another?

The point is to highlight an apparent incompatibility between the perspectives of different observers. It seems like *either* we need to retreat into a sort of solipsism—holding that an event that happened for Wigner might not have happened for his friend—or else we need some way of regarding measurement as fictitious.

OK, now let's discuss a few more interpretations of quantum mechanics. Our next one isn't really an “interpretation,” but rather a demand for a new physical theory.

Dynamical Collapse

If quantum mechanics doesn't make sense to us, it's worth at least considering the possibility that it's not a complete theory. I.e., that maybe it does a good job of describing microscopic systems, but we're not looking at all of the rules that govern reality.

In more detail: maybe there exist some physics rules that we haven't discovered which say that qubits *normally* evolve via unitary transformations, but that the bigger the system is (or something), the more likely it is to collapse. In that case, we could view collapse as a straightforward *physical* process that turns pure states into mixed states.

$$\sum_i \alpha_i |i\rangle \rightarrow |i\rangle \text{ with probability } |\langle \Psi | i \rangle|^2 = |\alpha_i|^2$$

So in the Schrödinger's cat example, dynamical collapse would say that it doesn't matter how isolated the box is. There's some yet-unknown physical law that says that a system that big would quickly evolve into a mixed state.

$$\frac{1}{\sqrt{2}}(|\text{cat alive}\rangle + |\text{cat dead}\rangle) \rightarrow \frac{1}{2}(|\text{cat alive}\rangle\langle \text{cat alive}| + |\text{cat dead}\rangle\langle \text{cat dead}|)$$

Note that in principle, there's a measurement that can distinguish the two states above.

Such a measurement would, admittedly, be absurdly hard to implement. In fact, a recent result by Prof. Aaronson says, informally, that if you have the technological capability to distinguish the two states above, then you also have the technological capability to rotate between the cat's "alive" and "dead" states. For this reason, the Schrödinger's cat experiment involves *far* less animal cruelty than most people say! If you can do the experiment at all, and prove that you did it, then you can also bring a dead cat back to life.

But setting aside technological difficulties, for us the relevant point is this: in saying that the first state evolves to the second, we're proposing new physics, *different* from standard quantum mechanics, that in principle has testable implications.

In other words, this isn't *really* interpreting quantum mechanics, it's just proposing new laws of physics! Physicists have a high bar for such proposals; the burden of proof is on the person proposing the new law to explain in quantitative detail how it works. In this case, that would mean giving a criterion for *exactly* which systems are "big" enough, or whatever, to trigger a collapse like the above—and ideally, deriving that criterion from more fundamental laws. Some suggestions include:

- Collapse happens when some number of atoms get involved
- Collapse happens after a certain mass is reached
- Collapse happens when a system reaches a certain level of "complexity" (defined how?)
- etc.
 - On their face, all these views seem contradictory to our understanding of physics, which relies on *reductionism*: each atom just keeps obeying the same simple equations, regardless of how big or complicated a system the atom might be part of.

To escape that problem, one famous proposal is the...

Ghirardi-Rimini-Weber (GRW) Theory

which says that each atom has some tiny probability of collapsing (or if you like, “being measured by God”) at each point in time. And if even one atom of Schrödinger’s cat was “measured by God,” that would cause the entire cat to collapse to the Alive or Dead states: this part is just the usual partial measurement rule of quantum mechanics. By analogy, measuring just one qubit of $\frac{1}{\sqrt{2}}(|00\dots 0\rangle + |11\dots 1\rangle)$ will resolve all of the qubits to 0 or 1. So in the GRW theory, Schrödinger cats are inherently unstable—and the bigger the system, the shorter the time it can be maintained in a Schrödinger-cat-like state.

another option is the...

Penrose Theory

which says that superpositions spontaneously collapse when enough mass gets involved, and the mass is separated by a big enough distance across different branches of the superposition.

Why mass and distance? mass here ▼ or ▼ mass there

Say we have the superposition of $|\text{mass here}\rangle + |\text{mass there}\rangle$. General relativity tells us that mass curves the nearby space-time: indeed, bends it like a mattress. That means that a mass in one location would make spacetime curve differently than the same mass somewhere else.

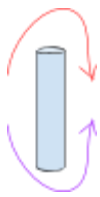
The thing is, no one really knows how to combine general relativity and quantum mechanics; it’s one of the biggest unsolved problems in physics. In particular, ordinary quantum mechanics presupposes space and time, or at least a *causal structure*: in terms of quantum circuits, if you like, a definite collection of qubits and gates acting on those qubits in some order. No one quite knows what it means to have a quantum superposition of different causal structures—yet, that *seems* to be what we’d be talking about in the situation with the widely-separated masses. So, Penrose’s proposal is basically that this could be the place where quantum mechanics breaks down, to be superseded by a more complete theory that includes “spontaneous collapses.” And then he has further ideas about how all of this might be related to consciousness, which we won’t go into.

One difficulty with this sort of theory, in general, is that the experimenters keep producing examples of bigger and bigger states in superposition. And as long as that continues, it seems like the believers in these theories will always need to be on the defensive—adjusting their answers to questions like “so, how much mass *is* enough to collapse a state?” to avoid contradicting the latest experiments.

Early on, we discussed the significance of the double-slit experiment as performed with photons. Later on, though, people managed to do the same experiment with protons, then molecules, and in 1999 the Zeilinger group in Vienna performed it with buckyballs: molecules with 60 atoms and hundreds of electrons. Since then the double-slit experiment has been done with larger molecules still.

To go even further...

Superconducting Qubits



If you take a coil, like maybe a micrometer across, and cool it to almost absolute zero, you can get a current that's in a superposition of the electrons flowing clockwise or counterclockwise around the coil.

This constitutes a quantum superposition involving billions of particles!

We'll come back to these superconducting coils at the end of the course, as they're an important technology for quantum computers.

Penrose has a specific prediction for the scale at which collapse happens, which might be testable in our lifetime. But with GRW, the prediction is basically just made to avoid contradicting any existing experiments.

One position, popular among people who want nature to be efficiently simulatable by a classical computer (and thus don't want quantum computers to work) says that:

A frog can be in a superposition of two states. However, a complex quantum computer wouldn't work, because quantum systems spontaneously collapse after they achieve "sufficient complexity" (whatever that means).

This position is interesting because it could be falsified by building a scalable quantum computer, and reaching falsifiable theories is what moves these discussions from philosophy to science.

What happens if we keep doing experiments and quantum mechanics keeps perfectly describing everything we see?

In particular, suppose we *don't* want to add any new physical laws, but we also insist on being *scientific realists*—holding that there exists a real state of the universe, and that the job of physics is to describe that state, not just to predict the results of measurements made by apes like ourselves.

Well, that combination of choices basically gets you to...

Everett's Many Worlds Interpretation (1957)

This famous view holds that the entire universe has a single quantum state $|\Psi\rangle$, and the entire history of the universe is just the vector $|\Psi\rangle$ going through unitary evolution.

On the Everett view, what we call "measurement," or "collapse," is just a special case of quantum systems becoming entangled with each other when they interact. In particular, *your brain*—not to mention, your measuring apparatus, the air molecules in the room, etc. etc.—all become entangled with the quantum system that you're measuring. You can think of it as a giant Controlled-NOT gate, with the system you're observing as the control qubit and you as the target qubit.

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} |\text{You}\rangle \rightarrow \frac{|0\rangle |\text{You}_0\rangle + |1\rangle |\text{You}_1\rangle}{\sqrt{2}}$$

The thing is, if you take this view seriously, it implies that *you yourself* have now “branched” into two possibilities, one where you observed the qubit in the $|0\rangle$ state, and another where you observed the qubit in the $|1\rangle$ state. Because unitary evolution is linear, these two branches are unlikely ever to interfere with each other again, or at least not for many quadrillions of years (more about that later). So your experience is *as if* only one of the branches was realized—but in truth, neither branch is more real than the other.



More generally, according to MWI, the universe “branches” each and every time a microscopic quantum state gets amplified to have macroscopic effect—even if there’s no one around to observe the amplification (e.g., if it’s in the interior of the Sun or something). So there’s a staggering amount of branching happening! And because of the well-known phenomenon of *chaos*, we can expect the branching to influence pretty much everything we care about. So for example, there are some branches of the quantum state of the universe where Austin is sunny at a specific moment one month from now, others where it’s rainy, others where it’s been destroyed in a nuclear war, etc., and no one of these branches is more “real” than the rest.

Some variants of the Many Worlds interpretation choose words carefully to avoid sounding like there’s *literal branching into different, equally-real worlds of experience*, but that’s basically what they all imply. When Everett came up with MWI as a grad student at Princeton, his advisor—the famous John Wheeler—told him to remove from his paper all references to the physical reality of parallel worlds, because it wouldn’t chime with the physics establishment at the time. So Everett did so, and partly as a result, it took almost 20 years for the rest of the physics community to rediscover Everett’s proposal and understand what it meant.

Soon after publishing his thesis about MWI—which he called the “relative state interpretation”—Everett left theoretical physics to become a nuclear war strategist for the Pentagon. The only public lecture Everett ever gave on MWI was here at UT Austin, decades later, when some people were finally coming around to the idea.

David Deutsch, the biggest current advocate of the Many Worlds Interpretation and one of the founders of quantum computing, was there.

One issue that we should return to is interference between branches.

If the different branches could interfere with each other, it would be as if not just the future but *the past* was constantly shifting, with no definite sequence of things that happened and were recorded. To avoid that, we need the “ $|0\rangle |\text{You}_0\rangle$ branch” to *not* affect the “ $|1\rangle |\text{You}_1\rangle$ branch,” and vice versa. Both branches might be equally real, but once you’re *in* one of the branches, you ought to be able to continue doing physics *as if* your branch was the only real one.

Fortunately, the usual rules of quantum mechanics give us this property: we don’t need to add anything extra to them. Recall: to calculate the amplitude of a given basis state $|x\rangle$ in a larger $|x\rangle$

superposition, you add up a contribution from every possible “path” that ends at . Interference would only happen if two “macroscopically different” paths led to *exactly* the same outcome—meaning, every single atom in the universe in the same position. But, while that’s not impossible, it’s massively “thermodynamically disfavored,” which basically means that it’s less likely to happen than seeing an egg unscramble itself. In fact, the constant proliferation of branches—the way the universe’s state, $|\Psi\rangle$, constantly sprouts new branches, but they almost never recombine—can be seen as literally an *instance* of the Second Law of Thermodynamics in action, as a process that constantly increases the universe’s entropy from the low value it had at the Big Bang.

And *why* did the universe have such a low entropy at the Big Bang? Well, to this day no one knows the answer to *that*, except that if it didn’t, we probably wouldn’t be here to wonder about it...

Interestingly, if we *also* believed that the universe was only finitely large—and in particular, that it could be fully described by the unitary evolution of a finite number of qubits (say 10^{122} of them)—then *eventually* we’d run out of room, and the branches would necessarily start colliding with each other. But even under that assumption, there doesn’t seem to be any reason for this happen in (say) the next 10^{100} years.

We said before that measurement is the one random and irreversible part of quantum mechanics. But Many Worlds denies that even that part is random or irreversible. After applying a unitary transformation U that describes a measurement process, in principle we could always apply U^{-1} to make the measurement “unhappen.” But just like with unscrambling an egg, thermodynamics isn’t going to make it easy.

Let’s now discuss some of the most common other questions people have about Many Worlds.

(1) Even if we accept that “measurements” have no fundamental physical status—still, where do the *apparent* probabilities come from?

That is, why does measuring a qubit $\alpha|0\rangle + \beta|1\rangle$, in the $\{|0\rangle, |1\rangle\}$ basis, yield the outcomes $|0\rangle$ and $|1\rangle$ with probabilities $|\alpha|^2$ and $|\beta|^2$ respectively?

It’s not enough to say that sometimes we see 0 and sometimes we see 1. Quantum mechanics gives very specific probabilities that each will occur. But if the world is just branching once for each observation, then how can we justify these probabilities as corresponding to anything meaningful? Does an “ $|\alpha|^2$ fraction of souls” go down one branch while a “ $|\beta|^2$ fraction of souls” goes down the other? Or: does the “splitting of the worlds” happen in such a way that amplitudes of $\frac{3}{5}$ and $\frac{4}{5}$ would correspond to $\frac{9}{25}$ “volume of worldness” going one way, and $\frac{16}{25}$ going the other?

Some philosophers don’t like this because if all the worlds are equally real, then why wouldn’t they just occur with equal probabilities? Why bother with amplitudes at all?

Everett’s response was to argue that if the universe branched many times in succession, then in “almost all branches” (where “almost all” is measured by amplitude), it would *look like* the Born probability rule was obeyed. But many people in the past half-century have been unsatisfied with that

argument, seeing it as circular—indeed, as smuggling the Born rule into the definition of “almost all branches”! So they’ve continued to look for something better.

There are many arguments, which we won’t go into here, that try to formalize the intuition that the Born probabilities are just “baked into” how quantum mechanics works. After all, unitary evolution already singles out the 2-norm as special by preserving it, so then why shouldn’t the probabilities *also* be governed by the 2-norm? More pointedly, one can argue that, if the probabilities were governed by something other than the 2-norm, then we’d get bizarre effects like faster-than-light communication. But, while these arguments help explain why the Born rule is perhaps the only choice of probability rule that makes internal mathematical sense, they still leave slightly mysterious how probability enters *at all* into Everett’s vision of a deterministically evolving wavefunction.

In Everett’s defense, one could ask the same questions—“where do these probabilities come from? why should they follow the Born rule, rather than some other rule?”—in *any* interpretation, not just in MWI.

(2) “If there’s no experiment that could differentiate the Copenhagen Interpretation from Many Worlds, why bother arguing about it?”

Many Worlders say that the opponents of Galileo and Copernicus could also claim the same about the Copernican versus Ptolemaic theories, since Copernican heliocentrism made no difference to the predictions of celestial movement.

Today, we might say that the Copernican view is better because you could fly outside of the solar system and see all the planets (including Earth) rotating around the far more massive sun; it’s only our parochial situation of living on Earth that ever motivated geocentrism in the first place.

But if we push this analogy further, it might be harder to think of anything similar for the Many Worlds interpretation, since quantum mechanics itself explains why we can’t really get outside of the universe to see the branching—or even get outside our own branch to interact in any way with the other decoherent branches.

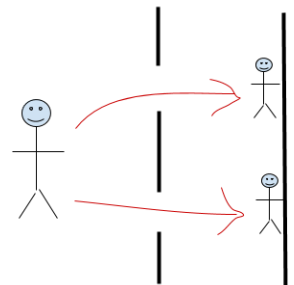
There is one neat way you could imagine differentiating the two, though...

Before we talked about doing the double-slit experiment with larger and larger systems. Bringing that thread to its logical conclusion, what if we could run the double-slit experiment with a person going through the slits?

It seems like it would then be necessary to say that “observers” can, indeed, exist in superpositions of having one experience and having a different one. This is what Many Worlds said all along, but seems to put a lot of rhetorical strain on the Copenhagen interpretation.

If you talk to modern Copenhagenists about this they’ll take a quasi-solipsistic view, saying that if this experiment were run, “the person behaving quantumly doesn’t count as an observer---only I, the experimenter, do.”

Of course, the Wigner’s Friend experiment was trying to get at this same difficulty.

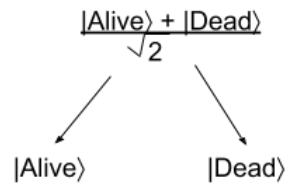


The third question we want to tackle is the **Preferred Basis Problem**. It says:

“Let’s say I buy into the argument that the universe keeps branching, well then...”

(3) In what basis is this branching occurring?

We talked about Schrödinger's cat as branching into the $|\text{Alive}\rangle$ state and the $|\text{Dead}\rangle$ state.



But mathematically, we could equally well have decomposed the cat's state in a basis like

$$\frac{(|\text{Alive}\rangle + |\text{Dead}\rangle)}{\sqrt{2}}, \frac{(|\text{Alive}\rangle - |\text{Dead}\rangle)}{\sqrt{2}}$$

So is there anything besides our intuition to “prefer” the first decomposition over the second one?

There's a whole field of physics that tries to answer questions like these, called...

Decoherence Theory

which says that there are certain bases whose states tend to be robust to interactions with the environment, but most bases aren't like that.

So in the example above, decoherence theory would explain that an alive cat doesn't easily decohere if you poke it, but that a cat in the $\frac{(|\text{Alive}\rangle + |\text{Dead}\rangle)}{\sqrt{2}}$ state does, because the $|\text{Alive}\rangle$ and $|\text{Dead}\rangle$ branches interact differently with the environment. This, according to decoherence theory, is more-or-less how the laws of physics pick out certain bases as being special.

From the standpoint of decoherence theory, we can say that an event has “definitely happened” if and only if there exist many records of the event scattered all over the place, so that it's no longer feasible to erase them all.

This is perhaps best compared to putting an embarrassing picture on Facebook. If only a few friends share it, you can still take it down. On the other hand, if the picture goes viral, then the cat is out of the bag, and deleting all the copies becomes next to impossible.

Lecture 13, Tues Feb 28: Hidden Variables,

Bell's Inequality

In the last lecture, we discussed four different attitudes people take toward quantum mechanics: Copenhagen, “shut up and calculate,” dynamical collapse, and Everett’s Many-Worlds Interpretation. You might think that *all* the options we’ve seen so far are bizarre and incomprehensible (Einstein certainly did), and wonder if we could come up with a theory that avoids all of the craziness. This leads us to the old dream of...

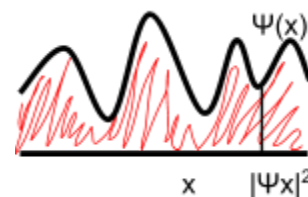
Hidden Variable Theories

which try to supplement quantum state vectors with some sort of hidden ingredients. The idea is to have a state, like $\alpha|0\rangle + \beta|1\rangle$, represent “merely” a way of making a prediction about what the universe *has already* set the result of measuring the qubit to be: either $|0\rangle$ or $|1\rangle$.

The most famous hidden-variable theory is...

Bohmian Mechanics

which was developed by David Bohm in the 1950s. It’s also called the “deBroglie-Bohm theory,” because it turns out that Louis de Broglie had the exact same idea in the 1920s—although deBroglie quickly disavowed it, after the idea faced a poor reception from other quantum mechanics pioneers.



Normal quantum mechanics says that a particle is in a superposition of locations, which we can use to calculate the probability that the particle will be found in one place or another when measured—and moreover, that this superposition exhausts what can be said about the particle’s location. But, while keeping that superposition as part of physics, we now want to say that there’s *also* a “real place” where the particle is, even before anyone measures it. To make that work, we need to give a rule for how the superposition “guides” the real particle. And this rule should have the property that, if anyone *does* measure the particle, they’ll find exactly the result that quantum mechanics predicted for it---since we certainly don’t want to give up on quantum mechanics’ empirical success!

At first, you might think it would be tricky to find such a rule; indeed, you might wonder whether such a rule is possible at all. However, the real problem turns out to be more like an embarrassment of riches! There are infinitely many possible rules that could satisfy the above property—but by design, they all yield exactly the same predictions as standard quantum mechanics. So there’s no experimental way to know which one is correct.

To explain this in a bit more detail, let’s switch from particle positions back to the discrete quantum mechanics that we’re more comfortable with in this course. Suppose we have a quantum pure state, represented as an amplitude vector in some fixed basis. Then when we multiply by a unitary transformation, suppose we want to be able to say: “this is the basis state we were *really in* before the

unitary was applied, and this is the one we're *really in* afterwards.” In other words, we want to take the equation

$$\begin{bmatrix} \beta_1 \\ \vdots \\ \beta_n \end{bmatrix} = \begin{bmatrix} U_{11} & & U_{1n} \\ & \ddots & \\ U_{n1} & & U_{nn} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}$$

and map it to an equation

$$\begin{bmatrix} |\beta_1|^2 \\ \vdots \\ |\beta_n|^2 \end{bmatrix} = \begin{bmatrix} & & \\ & S & \\ & & \end{bmatrix} \begin{bmatrix} |\alpha_1|^2 \\ \vdots \\ |\alpha_n|^2 \end{bmatrix}$$

for some choice of stochastic matrix S (possibly depending on the input and output vectors).

There are many, many such matrices S . For example you could put $\begin{bmatrix} |\beta_1|^2 \\ \vdots \\ |\beta_n|^2 \end{bmatrix}$ in every column, which would say that you're always jumping randomly over time, but in a way that preserves the Born Rule. You could have been in a different galaxy a Planck time ago; now you're here (with fictitious memories planted in your brain); who knows where you'll be a Planck time from now?

But Bohm thought, not about this discrete setting, but mostly about the example of a particle moving around in continuous Euclidean space. And in the latter case, it turns out that one can do something nice that isn't possible with finite-dimensional unitary matrices. Namely, one can give a *deterministic* rule for how the particle moves around—a differential equation—that still reproduces the Born rule at every instant in time, provided only that it reproduces the Born rule at any *one* time. More poetically, “God needs to use a random-number generator to initialize the hidden variables at the beginning of time”—say, at the Big Bang—but afterwards, they just follow the differential equation. And furthermore, while the choice of differential equation isn't quite unique, in simple scenarios (like a particle moving around in space) there's one choice that seems *better*, simpler, more motivated than the rest.

However, in thinking through the implications of Bohmian mechanics, Bohm and others noticed lots of weird things. It looks very elegant with just one particle, but new issues arise when there are two entangled particles. Bohmian mechanics says that you need to give a definite position for both particles, but people noticed that acting on Alice's particle would *instantaneously* change the Bohmian position of Bob's particle, however far away the particles were—even while Bob's *density matrix* remained unchanged because of the No-Communication Theorem.

While unsettling, this still wouldn't be useful for faster-than-light communication, since the Bohmian hidden variables are explicitly designed to have no measurable effects, over and above the effects we'd predict using the quantum state itself.

When Bohm proposed his interpretation, he was super eager for Einstein to accept it, but Einstein didn't really go for it, probably because of this non-locality problem.

What Einstein really wanted (in modern terms), is a...

Local Hidden Variable Theory

where hidden variables not only exist, but can be localized to specific points in space, and are only influenced by things happening close to them.

For example, imagine that when an entangled pair $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$ is created, the qubits secretly flip a coin and decide, "if anyone measures us in the $\{|0\rangle, |1\rangle\}$ basis, let's both be 0." More broadly, imagine that they agree in advance on such answers for all questions that could be asked (i.e., all bases in which they could possibly be measured), and that each qubit carries around its own local copy of the answers.

This is not Bohmian mechanics. In fact, around 1963 John Bell wrote a paper that drew attention to the non-local character of Bohmian mechanics. Bell remarked that it would be interesting to prove that *all* hidden variable theories must be non-local: that this isn't just a defect of Bohm's proposal, but inherent to what Bohm was trying to do. The paper has a footnote saying that as the paper was going to press, such a proof was found. This was the first announcement of one of the most famous discoveries ever made about quantum mechanics, what we now call

Bell's Inequality / Bell's Theorem

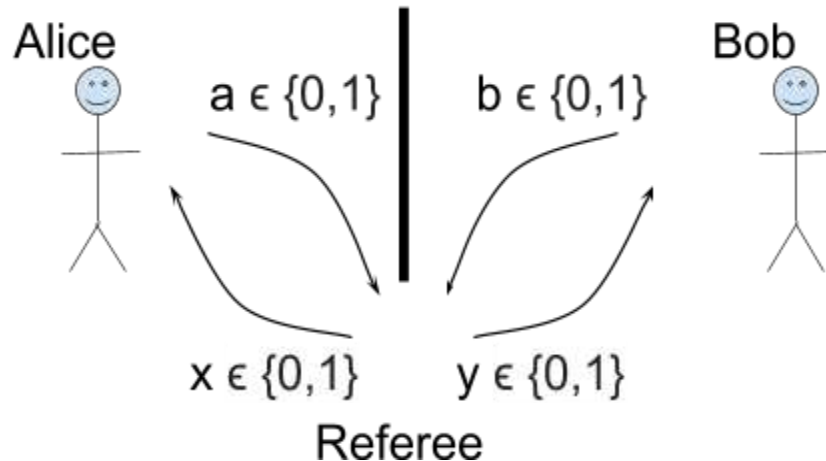
Einstein and others had already touched on the idea of local hidden variables in their philosophical debates in the 1930s. But Bell was the first to ask: do local hidden variables have any *empirical consequences* that disagree with the predictions of quantum mechanics? Is there an actual experiment that could rule out the possibility of local hidden variables?

Bell came up with such an experiment. We'll describe it differently from how Bell did—more computer science-y—as a game with two cooperating players named (what else?) Alice and Bob, where the win probability can be improved through shared entanglement. It's called...

The CHSH Game

named after four people (Clauser, Horne, Shimony, and Holt) who in 1969 wrote a paper saying "*this* is how to think about what Bell did." The game itself doesn't involve quantum mechanics, but quantum mechanics can help us win it.

The CHSH game could be seen as a precursor to quantum computing, in that it's one of the first cases where people looked to see which information processing tasks quantum mechanics helps us solve better—and where they enforced a conceptual separation between the task itself (which is classical), and the strategy to solve it (which can be quantum).



The idea is that Alice and Bob are placed in separate rooms, and are each given a challenge bit (x and y , respectively) by a referee. The challenge bits are chosen uniformly at random, and independently of each other. Then Alice sends an answer bit a back to the referee, and Bob sends back an answer bit b .

Alice and Bob “win” the game iff $a + b = xy \pmod{2}$

So if either x or y is 0: a and b should be equal

But if $x = y = 1$: a and b should be different

Alice and Bob are allowed to agree on a strategy in advance, and to share random bits.

The classical strategy to maximize winning probability is simply that Alice and Bob always send the referee $a = b = 0$, regardless of what x and y are. In this case, Alice and Bob win 75% of the time, losing only if x and y are both 1.

To prove that this is optimal, the first step is to notice that, without loss of generality, Alice and Bob’s strategy can be assumed to be deterministic (i.e., to involve no random bits besides x and y themselves). For any probabilistic strategy is just a mixture of deterministic ones—but then the win probability is just the average over all the strategies, so there must be *some* deterministic strategy in the mixture that does at least as well as the average. (This is called a *convexity argument*.)

So we can treat Alice’s output bit, a , as a function of her input bit x , and Bob’s output bit b as a function of his input bit y . And then we need the equation

$$a(x) + b(y) = xy \pmod{2}$$

OK, but you can easily check by enumerating cases that this equation can’t possibly hold for all 4 values of x and y ! At best it can hold for 3 of the 4 values, which is exactly what the trivial strategy above gets.

The Bell Inequality, in this framework, is just the slightly-boring statement that we proved above: namely, that the maximum classical win probability in the CHSH game is 75%.

Bell noticed an additional fact though. Namely, if Alice and Bob had a pre-shared Bell pair, $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$, then there’s a better strategy. In that case, in fact, their maximum win probability is

$$\cos^2\left(\frac{\pi}{8}\right) = \frac{1 + \sqrt{\frac{1}{2}}}{2} \sim 85\%$$

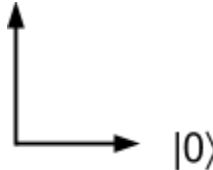

Why? Tune in next time to find out!

Lecture 14, Thurs March 2: Nonlocal Games

Last time we talked about the CHSH Game, and how no classical strategy lets Alice and Bob win it more than 75% of the time. Today we'll see how, by using entanglement, they can win 85% of the time—and then we'll delve deeper to try to understand what's going on.

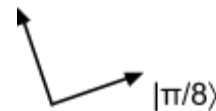
The strategy involves Alice and Bob measuring their respective qubits in different bases, depending on whether their input bits x and y are 0 or 1, and then outputting bits a and b respectively based on the outcomes of those measurements.

Let $|\frac{\pi}{8}\rangle = \cos(\frac{\pi}{8})|0\rangle + \sin(\frac{\pi}{8})|1\rangle$, and let $|\frac{-\pi}{8}\rangle = \cos(-\frac{\pi}{8})|0\rangle + \sin(-\frac{\pi}{8})|1\rangle$.

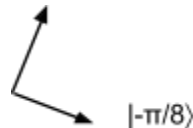
If $x = 0$, Alice measures in  and if $x = 1$, Alice measures in  $|+\rangle$

She sets a to 0 if she measures $|0\rangle$ or $|+\rangle$
and 1 if she measures $|1\rangle$ or $|-\rangle$

If $y = 0$, Bob measures in the X basis rotated by $\frac{\pi}{8}$ clockwise.



and if $y = 1$ rotated by $-\frac{\pi}{8}$.



he sets b to 0 if he measures $|\frac{\pi}{8}\rangle$ or $|\frac{-\pi}{8}\rangle$
1 if otherwise

This strategy has the amazing property of making Alice and Bob win with probability $\cos^2(\frac{\pi}{8})$ for all possible values of x and y .

So why does this strategy work 85% of the time?

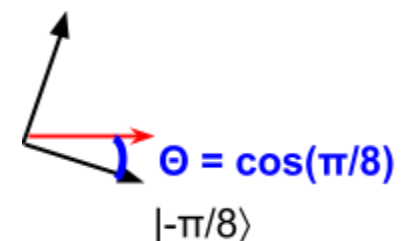
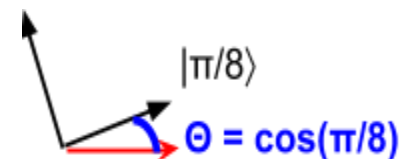
Let's consider the case where Alice gets $x = 0$ and measures $|0\rangle$.

She'll output $a = 0$, and she and Bob will win iff Bob outputs $b = 0$.

So what are the odds that Bob outputs 0?

Given that Alice measured her qubit already, Bob's qubit collapsed to the $|0\rangle$ state.

First suppose $y = 0$. Then Bob measures the $|0\rangle$ state in a basis rotated by $\frac{\pi}{8}$ clockwise. He outputs 0 if he measures $|\frac{\pi}{8}\rangle$. Thus, the probability that Bob outputs 0 in this case is $\cos^2(\frac{\pi}{8}) \approx 85\%$.



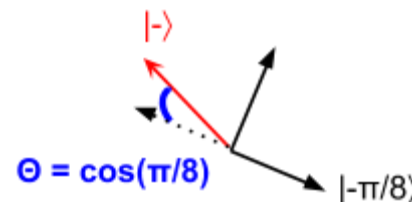
We can do the same calculation for the case $y = 1$. The angle between vectors is still $\frac{\pi}{8}$. In fact, we can generalize this result to *all* the cases where either x or y is 0.

Note that we can assume without loss of generality that Alice measures first, because of the No Communication Theorem.

The interesting case is where a and b are both 1.

Here Alice measures in the $\{|+\rangle, |-\rangle\}$ basis. Assume without loss of generality that Alice gets the outcome $|-\rangle$. Then what's Bob's probability of getting the outcome $|-\frac{\pi}{8}\rangle$?

It's still $\cos^2(\frac{\pi}{8})$, because the angle between $|-\rangle$ and $|-\frac{\pi}{8}\rangle$ is $\frac{\pi}{8}$, and global phase doesn't matter.



So, Alice and Bob win the game with probability $\cos^2(\frac{\pi}{8})$ in all four cases.

How does this game relate to hidden variable theories? Well, if all correlations between the qubits could be explained by stories like “if anyone asks, we’re both 0,” then we’d make a firm prediction: that Alice and Bob can win the CHSH game at most 75% of the time (because that’s how well they can do by pre-sharing arbitrary amounts of classical information).

So if they play the game repeatedly, and demonstrate that they can win more than 75% of the time, then local realism is false. Notice that nowhere in this argument did we ever need to presuppose that quantum mechanics is true.

Does Alice and Bob’s ability to succeed more than 75% of the time mean that they are communicating?

Well, we know it’s not possible for either to send a signal to the other, by the No-Communication Theorem. But how can we reconcile that with their success in the CHSH game?

One way to understand what’s going on, is to work out Alice and Bob’s density matrices explicitly.

Bob’s initial density matrix is $\begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$ and after Alice measures it’s still $\begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$.

So in that sense, no signal has been communicated from Alice to Bob. Nevertheless, *if* you knew both Alice’s measurement and its outcome, then you could update Bob’s density matrix to that of a pure state. That shouldn’t worry us though, since even classically, if you condition on what Alice sees then you can change your predictions for Bob.

Imagine a hierarchy of possibilities for what the universe allows. Classical Local Realism is at the bottom: that’s where you only ever need to use classical probability theory when you have incomplete information about physical systems, *and also* signals propagate only at the speed of light.

At the top of the hierarchy is the Faster-Than-Light Science-Fiction Utopia, where Alice and Bob can communicate instantaneously, you can travel faster than light, and so forth.

A priori, people tend to believe that reality must be one or the other. So when they read pop-science articles about how classical local realism is false, they think, “OK, then we must live in the FTL sci-fi utopia.”

Instead, the truth—according to quantum mechanics—is in the middle, and is so subtle that perhaps no science-fiction writer would ever have had the imagination to invent it. We live in a world where there’s no classical local realism, but no faster-than-light communication either. Or to put it another way: a *classical simulation* of our universe would involve FTL communication, but our universe itself does not.

Maybe no science fiction writer ever came up with this possibility, simply because it’s hard to think of a plot that requires Alice and Bob to win the CHSH game 85% of the time instead of 75%!

Hierarchy of Possibilities



Indeed, this is a key piece of evidence that our world really *is* quantum, and is not secretly classical behind the scenes: because we know from Bell that the latter possibility would require FTL communication.

So where is that $\cos^2(\frac{\pi}{8})$ coming from anyways? It seems so arbitrary...

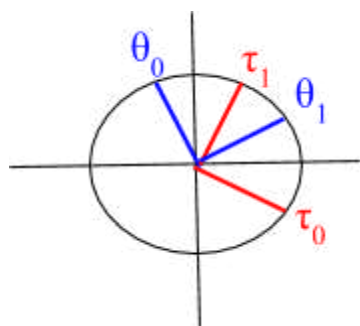
It may seem like the $\cos^2(\frac{\pi}{8})$ is simply coming from our particular approach to the problem. Maybe if we came at it another way, we could use entanglement to win *even more* than 85% of the time: why not 100%?

Surprisingly, $\cos^2(\frac{\pi}{8})$ turns out to be optimal, even if Alice and Bob share unlimited amounts of entanglement. This is the upshot of

Tsirelson’s Inequality

...a cousin of the Bell inequality, proved in the 1980s.

It requires a *bit* too much machinery to give a complete proof of Tsirelson’s Inequality here. However, we’ll convey the intuition, by showing that, among strategies “similar to the one we used,” ours was the optimal one.



Let’s say that Alice has two angles:

θ_0 , the angle she measures in if she receives input $x = 0$, and

θ_1 , the angle she measures in if she receives input $x = 1$.

Similarly, Bob has τ_0 and τ_1 , corresponding to $y = 0$ and $y = 1$ respectively.

The same rules apply from the solution we constructed earlier for the CHSH game. All we’re doing here is changing the chosen vectors into variables to try and show that there’s no better vectors to choose than the ones we did.

The key formula is this: Alice and Bob’s total success probability is

$$P[\text{success}] = \frac{1}{4}[\cos^2(\theta_0 - \tau_0) + \cos^2(\theta_0 - \tau_1) + \cos^2(\theta_1 - \tau_0) + \sin^2(\theta_1 - \tau_1)]$$

Why?

1. Each of the four input pairs has an equal chance of occurring.
2. In the first three cases, Alice and Bob win iff they output the same bit, so we take the squared cosine between their measurement angles.
3. But in the fourth case, $x = y = 1$, Alice and Bob win iff they output *different* bits. So in this case, we take the squared sine between their measurement angles.

Now we use some high-school trigonometry to get the above equals

$$\frac{1}{2} + \frac{1}{8} [\cos(2(\theta_0 - \tau_0)) + \cos(2(\theta_0 - \tau_1)) + \cos(2(\theta_1 - \tau_0)) - \cos(2(\theta_1 - \tau_1))]$$

We can get rid of the 2's inside the cosines, by simply realizing that we could adjust our original angles to account for them.

It will be helpful to think of the cosines as the inner products between unit vectors. In that case, we can rewrite the above as

$$\begin{aligned} & \frac{1}{2} + \frac{1}{8} [u_0 \cdot v_0 + u_0 \cdot v_1 + u_1 \cdot v_0 - u_1 \cdot v_1] \\ &= \frac{1}{2} + \frac{1}{8} [u_0 \cdot (v_0 + v_1) + u_1 \cdot (v_0 - v_1)] \end{aligned}$$

Since u_0 , u_1 , v_0 , and v_1 are all unit vectors, the above is upper-bounded by

$$\leq \frac{1}{2} + \frac{1}{8} [\|v_0 + v_1\| + \|v_0 - v_1\|]$$

From here, we can use the parallelogram inequality to bound it further:

$$\leq \frac{1}{2} + \frac{1}{8} \sqrt{2(\|v_0 + v_1\|^2 + \|v_0 - v_1\|^2)}$$

Which equals

$$\frac{1}{2} + \left(\frac{\sqrt{2}}{8}\right) \sqrt{4} = \frac{1}{4} (2 + \sqrt{2})$$

Which wouldn't you know it, is equal to

$$\cos^2\left(\frac{\pi}{8}\right) \approx 85\%$$

So that really is the maximum winning probability for the CHSH game.

There's been a trend in the last 15 years to study theories that would go past quantum mechanics by letting you violate Tsirelson's Inequality, but that would still prohibit faster-than-light travel.

In such a world, it's been proven (among other things) that if Alice and Bob wanted to schedule something on a calendar, they could decide if there's a date where they're both free by exchanging only one bit of communication. That's a lot better than can be done under the rules of quantum mechanics!

Testing the Bell Inequality

When Bell proved his inequality, he was just trying to make a conceptual point, about the necessity for nonlocal influences in any hidden-variable theory underlying quantum mechanics. But by the 1980s, technology had advanced to the point where playing the CHSH game was actually a feasible physics experiment! Alain Aspect (and others) ran the experiment, and the results were fully consistent with quantum mechanics, and extremely problematic for local hidden variable theories.

The experiments don't quite get to 85% success probability, given the usual difficulties that afflict quantum experiments. But you can reach a high statistical confidence that you're winning more than, say, 80% of the time.

This was evidence, not only that local realism was false, but also that entanglement had been created.

Most physicists shrugged, already sold on quantum mechanics (and on the existence of entanglement). But a few, still committed to a classical view of the world, continued to look for loopholes in the experiment.

Skeptics pointed out two loopholes in the existing experiments, essentially saying “if you squint enough, classical local realism might still be possible”:

1. Detector Inefficiency

Sometimes detectors fail to detect a photon, or they detect non-existent photons. Enough noise in the experiments could skew the data.

2. The Locality Loophole

Performing the measurements and storing their results in a computer memory takes some time: maybe nanoseconds or microseconds. Now, unless Alice and Bob and the referee are *very* far away from each other, this opens the possibility of a sort of “local hidden variable conspiracy,” where as soon as Alice measures, some particle (unknown to present-day physics) flies over to Bob and says “hey, Alice got the measurement outcome 0; you should return the measurement outcome 0 too.” The particle would travel “only” at the speed of light, yet could still reach Bob before his computer registered the measurement outcome.

By the 2000s, physicists were able to close loophole 2, but only in experiments still subject to loophole 1. And conversely, they could close loophole 1, but only in experiments still subject to 2. Finally, in 2016, several teams managed to do experiments that closed both loopholes simultaneously.

There are still people who deny the reality of quantum entanglement, but through increasingly solipsistic arguments. For example...

Superdeterminism

is a strange way to maintain that classical local realism is still the law of the land. Superdeterminism explains the results of CHSH experiments by saying “We only *think* Alice and Bob can choose measurement bases randomly. Actually, there's a grand cosmic conspiracy involving all of our brains, our computers, and our random number generators, with the purpose of rigging the measurement bases to ensure that Alice and Bob can win the CHSH game ~85% of the time. But that's all this cosmic conspiracy does: it doesn't allow FTL communication or anything like that, even though it easily could.”

Nobel Laureate Gerard 't Hooft advocates superdeterminism, so it's not like the idea lacks distinguished supporters, but Professor Aaronson is on board with entanglement.

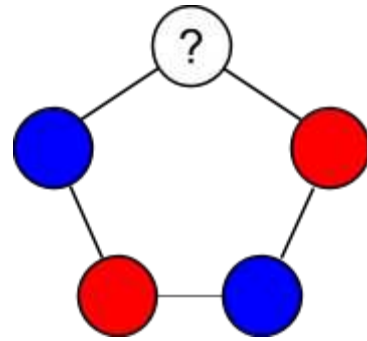
Now we'll look at some other non-local games, to see what else entanglement can help with. First we have...

The Odd Cycle Game

There's a cycle with an odd number of vertices n .

Alice and Bob claim that they have a two-coloring of the cycle, but basic graph theory tells us that this isn't possible.

Nevertheless, Alice and Bob will try to convince a referee that they've found a two-coloring anyway. They'll do that by using entanglement to coordinate their responses to challenges from a referee.



The referee performs two obvious consistency checks:

- He can ask them both the color of a random vertex v , in the two-coloring they found.
 - They pass the test iff their answers are the same
- He can ask Alice the color of a random vertex v , and Bob the color of an adjacent vertex w .
 - They pass the test iff their answers are different

The referee chooses between these tests with equal probability—and crucially, he doesn't tell Alice or Bob which test he's performing.

In a single run of the game, the referee performs one such test, and gets answers from Alice and Bob. Without loss of generality, assume their answers are always RED or BLUE.

What strategy provides the best probability that Alice and Bob will pass the referee's test and win the game?

Classically, we know that regardless of what Alice and Bob do, $Pr[\text{win}] < 1$.

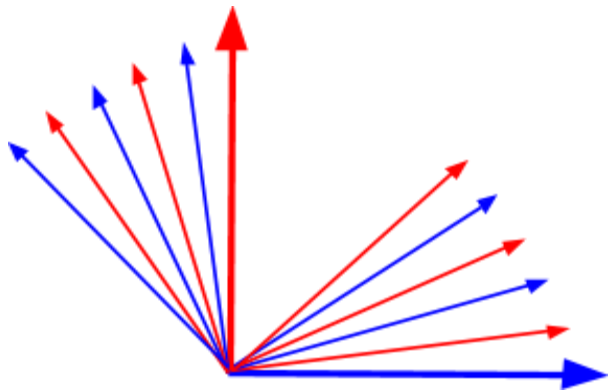
Why? Because for Alice and Bob to answer all possible challenges correctly, they'd need an actual two-coloring, which is impossible. The best they can do is agree on a coloring for all but one of the vertices, which gets them $Pr[\text{win}] = 1 - \frac{1}{2n}$.

Nevertheless, we claim that quantumly Alice and Bob can do better, and can achieve

$$Pr[\text{win}] \approx 1 - \frac{1}{n^2}.$$

if they pre-share a single Bell pair, $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$.

The strategy is as follows: Alice and Bob both measure their qubits in a basis depending on the vertex they're asked about.



The measurement bases for adjacent vertices differ by $\frac{2\pi}{n}$, so the bases rotate all the way around the unit circle.

The first basis has outcome $|0\rangle$ map to answering BLUE, and outcome $|1\rangle$ map to answering RED. The second basis has outcome $|\frac{2\pi}{n}\rangle$ map to answering RED, and outcome $|\frac{\pi}{2} + \frac{2\pi}{n}\rangle$ map to answering BLUE. They continue alternating in this way.

The upshot is that, when Alice and Bob are asked about the same vertex, they both measure in the same basis, and thus answer with the same color.

On the other hand, when Alice and Bob are asked about adjacent vertices, we get a similar situation to the CHSH game, where the probability of Bob producing the same output as Alice, is the squared sine of the angle between their vectors:

$$\sin^2 \theta = \sin^2 \left(\frac{1}{n} \right) \approx \frac{1}{n^2}.$$

Another crazy game is...

The Magic Square Game

Here Alice and Bob claim that they can fill a 3×3 grid with 0's and 1's so that:

1. Every row has an even sum
2. Every column has an odd sum

The referee asks Alice to provide a randomly-chosen row of the grid, and asks Bob to provide a randomly-chosen column. Alice and Bob “win” the game if and only if:

- The row has an even sum
- The column has an odd sum
- The row and column agree on the square where they intersect

		0
0	1	1
		0

You can see that constraints 1 and 2 *can't* actually both be satisfied, by examining the total number of 1's in the grid. Constraint 1 requires this total number to be even, while constraint 2 requires it to be odd. This implies that there's no classical strategy that lets Alice and Bob win the game with probability 1.

Nevertheless, David Mermin (the author of our textbook) discovered a quantum strategy where Alice and Bob win with probability 1. This strategy requires them to share 2 ebits of entanglement.

We won't describe the strategy in class, since it's very complicated to state without the “stabilizer formalism,” which we haven't yet seen (but will by the end of the course).

Lecture 15, Thurs March 9: Einstein-Certified Randomness

Until recently, the Bell inequality was taught because it was historically and conceptually important, not because it had any practical applications. Sure, it establishes that you can't get away with a local hidden variable theory, but in real life, no one *actually* wants to play the CHSH game, do they? In the last 10 years, however, it's found applications in...

Generating Guaranteed Random Numbers

This is one of the most important tasks in computing (and certainly in cryptography). Once we have quantum mechanics, you might think that the solution is trivial. After all, you can get a random bit by measuring the $|+\rangle$ state in the $\{|0\rangle, |1\rangle\}$ basis. Easy, right? But this solution isn't good enough for cryptography. Cryptographers are paranoid people, and they want the ability to maintain security, even if the hardware they're using was designed by their worst enemy.



These sorts of assumptions aren't just academic speculation, especially given the Snowden revelations.

For example, NIST (the National Institute of Standards and Technology) put out a standard for pseudo-random number generation based on elliptic curves to be used for encryption a while back. This standard was later discovered to have a backdoor created by the NSA that would allow them to predict the output numbers, thus being able to break systems encrypted under this standard.

Thus cryptographers want to base their random number generation on the smallest set of assumptions possible. They want bits that are guaranteed to be random, and to be sure that no one added predictability through any sort of backdoor.

You might think that, logically, one can never prove that numbers are truly random: that the best one can ever say is "I can't find any pattern here." After all, you can't prove a negative, and if not the NSA, who's to say that God himself didn't insert a pseudorandom pattern the workings of quantum mechanics?

Though presumably, if God wanted to read our emails he could also do it some other way...

That's what makes it so interesting, and non-obvious, that the Bell inequality *lets us certify numbers as being truly random* under very weak assumptions, which basically boil down to "no faster-than-light travel is possible." Let's now explain how.

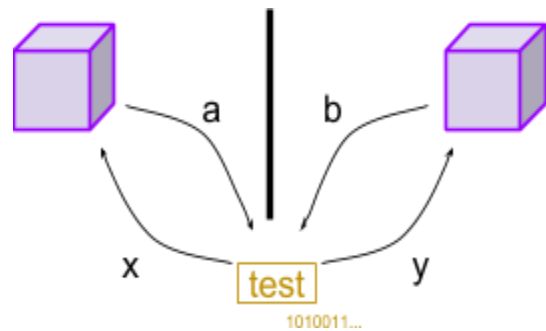
Suppose you have two boxes that share quantum entanglement. We'll imagine the boxes were designed by your worst enemy, so you trust nothing about them. All we'll assume is that the boxes can't send signals back and forth (say, because you put them in Faraday cages, or separate them by so large a distance that light has no time to travel between them).

A referee sends the boxes challenge numbers, x and y respectively.

The boxes return numbers a and b respectively.

If the returned numbers pass a test, we'll declare them to be truly random.

So what's the trick? Well, we already saw the trick; it's just the CHSH game!



The usual way to present the CHSH game is as a way for Alice and Bob to prove that they share entanglement—and thus, that the universe is quantum-mechanical, and that local hidden-variable theories are false.

However, winning the CHSH game more than 75% of the time *also* establishes that a and b must have some randomness, that there was some amount of entropy generated.

Why? Because suppose instead that a and b were deterministic functions—i.e., suppose they could be written as $a(x, r)$ and $b(y, r)$ respectively, in terms of Alice and Bob's inputs as well as shared random bits. In that case, whatever these functions were, they'd define a local hidden-variable theory, which is precisely what Bell rules out!

So the conclusion is that, if

- (1) x and y are random and
- (2) there's no communication between Alice and Bob,

then there must exist at least some randomness in the outputs a and b .

Around 2012, Umesh Vazirani coined the term Einstein-Certified Randomness for this sort of thing. The basic idea goes back earlier—for example, to Roger Colbeck's 2006 PhD thesis, and (in cruder form) to Prof. Aaronson's 2002 review of Stephen Wolfram's *A New Kind of Science*, which used the idea to refute Wolfram's proposal for a deterministic hidden-variable theory underlying quantum mechanics.

OK, so how do we actually extract random bits from the results of the CHSH game?

You could just take the stream of all a 's and b 's that are output, after many plays of the CHSH game. Admittedly, this need not give us a *uniform* random string. In other words, if the output string x has length n , then its Shannon entropy,

$$\sum p_x \log_2 \frac{1}{p_x}$$

where p_x is the probability of string x , will in general be less than n . However, we can then convert x , if we like, into an (almost) uniformly random string on a smaller number of bits, say $\frac{n}{10}$ or something, by using a well-known tool from classical theoretical computer science called a randomness extractor. A randomness extractor—something we already met in the context of quantum key distribution—is a function that crunches down many sort-of-random bits (and, typically, a tiny number of truly random bits, called the *seed*) to fewer very random bits.

David Zuckerman (here at UT) is an expert on randomness extractors.

OK, but there's an obvious problem with this whole scheme.

Namely: we needed the input bits to be uniformly random, in order to play the CHSH game. But that means we put in two perfect random bits, x and y , in order to get out two bits a and b that are *not* perfectly random! In other words, the entropy we put in is greater than the entropy we get out, and the whole thing is a net loss.

A paper by Pironio et al. addressed this by pointing out that you don't have to give Alice and Bob perfectly random bits every time the CHSH game is played. Instead, you can just input $x = y = 0$ most of the time, and occasionally stick in some random x 's and y 's to prevent Alice and Bob from using hidden variables. Crucially, if Alice or Bob gets a 0 input in a given round, then they have no way of knowing whether that round is for testing or for randomness generation. So, if they want to pass the randomly-inserted tests, then they'll need to play CHSH correctly in *all* the rounds (or almost all of them), which means generating a lot of randomness.

At this point it all comes down to a quantitative question:

So how much entropy can we get out, per bit of entropy that we put in?

There was a race to answer this, by designing better and better protocols that got more and more randomness out per bit of randomness invested. First Colbeck showed how to get cn bits out per n bits in, for some constant $c > 1$. Then Pironio et al. showed how to get $\sim n^2$ bits out per n bits in. Then Vazirani and Vidick showed how to get $\sim \exp(\sqrt{n})$ bits out per n bits in, which is the first time we had exponential randomness expansion. But all this time, an obvious question remained in the background: “why not just use a constant amount of randomness to jumpstart the randomness generation, and then feed the randomness outputted by Alice and Bob back in as input, and so on forever, thereby getting *unlimited* randomness out?”

It turns out that a naïve way of doing this doesn't work: if you just feed Alice and Bob the same random bits that they themselves generated, then they'll *recognize* those bits, so they won't be random *to them*—and that will let Alice and Bob cheat, making their further outputs non-random.

Remember: We're working under the assumption that
“Alice” and “Bob” are machines designed by our worst enemy!

If you don't have a limit on the number of devices used, then a simple fix for this problem is to feed Alice and Bob's outputs to two other machines, Charlie and David. Then you can feed Charlie and David's outputs to two more machines, Edith and Fay, and so on forever, getting exponentially more randomness each time.

OK, but what if we have only a *fixed* number of devices (like 4, or 6), and we still want unlimited randomness expansion? In that case, a few years ago Coudron and Yuen, and independently Chung, Miller, Shi, and Wu, figured out how to use the additional devices as “randomness laundering machines”—basically, converting random bits that Alice and Bob can predict into random bits that they *can't* predict, so that then the output bits can be fed back to Alice and Bob for further expansion, and so on as often as desired.

One question that these breakthrough works *didn't* address, was exactly how many random “seed” bits are needed to jump-start this entire process. Like, are we talking a billion bits or 2 bits? In a student project supervised by Prof. Aaronson, Renan Gross calculated the first explicit upper bound, showing that a few tens of thousands of random bits suffice. That's likely still far from the truth: it might be possible with as few as 10 or 20.

It's a pretty amazing conceptual fact that playing the CHSH game can lead to certified random bits (and worth mentioning that this sort of protocol has already been experimentally demonstrated at NIST). So you might wonder...

What else can you certify about two separated boxes, by seeing them win at the CHSH game?

It turns out that the answer is: an *enormous* number of things.

In a tour-de-force in 2012, Reichardt, Unger, and Vazirani showed how to use a sequence of CHSH-like challenges to certify that Alice and Bob performed a specific sequence of unitary transformations on their qubits (up to local changes of basis). This means that, just by making Alice and Bob repeatedly win the CHSH game, you can force them to do *any quantum computation of your choice*. Reichardt et al. describe this as a “classical leash for a quantum system.”

This sort of thing constitutes one of the main current ideas for how a classical skeptic could verify the operation of a quantum computer. For (say) factoring a huge integer into primes, we can easily verify the output of a quantum algorithm, by simply multiplying the claimed factors and seeing if they work! But this isn't believed to be the case for all problems that a quantum computer can efficiently solve. Sometimes, the only way to efficiently verify a quantum computer is working correctly might involve using quantum resources yourself. What Reichardt et al. show is that, as long as we have *two* quantum computers, and as long as those quantum computers are entangled with each other but unable to exchange messages, we can use the CHSH game to verify that the computers are behaving as expected.

This brings us nicely to **quantum computation**, which is probably the subject that most of you took the course to learn about! We'll begin discussing quantum computation in earnest in the next lecture.

Lecture 16, Tues March 21: Quantum Computing, Universal Gate Sets

Guest Lecture by Tom Wong

Having seen lots of quantum protocols, we're finally ready to tackle the holy grail of the field: a *programmable quantum computer*, a single machine that could do *any* short series of quantum-mechanical operations.

Quantum computation has two intellectual origins.

One comes from David Deutsch, who was thinking about experimentally testing the Many-Worlds Interpretation (of which he was and remains a firm believer), during his time as a postdoc here at UT Austin. Much like with Wigner's friend, Deutsch imagined creating an equal superposition of a human brain having measured a qubit as $|0\rangle$, and the same brain having measured the qubit as $|1\rangle$. In some sense, if you ever had to ascribe such a superposition state to *yourself*, then we'd have to discard the Copenhagen Interpretation.

But how could we ever test this? Given the practical impossibility of isolating all the degrees of freedom in a human brain, *Step 1* would presumably have to be: take a complete description of a human brain, and upload it to a computer. Then *Step 2* would be to put the computer into a superposition of states corresponding to different thoughts.

But then, this leads to more general questions: *could* anything as complicated as a computer be maintained in a superposition of “semantically different” states? How would you arrange that, in practice? And would such a computer be able to use its superposition power to do anything that it couldn't do otherwise?

The other, more “respectable,” path to the same questions came from Richard Feynman, who gave a famous lecture in 1982 concerned with the question, “how do you simulate quantum mechanics on a classical computer?”

Chemists and physicists had known for decades that this is hard, basically because the number of amplitudes that you need to keep track of increases exponentially with the number of particles. This is the case because, as we know, an n -qubit state can be highly entangled.

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

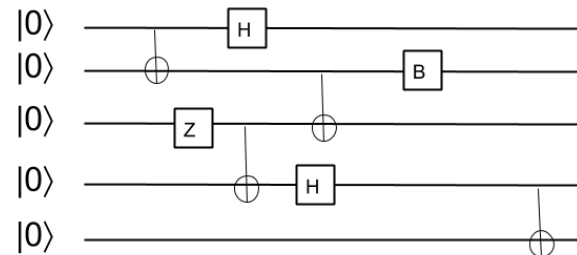
The state must be described by the vector $\begin{bmatrix} \alpha_{00\dots 0} \\ \alpha_{00\dots 1} \\ \vdots \end{bmatrix}$ of length 2^n

Chemists and physicists know many approximation methods for such problems—going by names like Density Functional Theory and Quantum Monte Carlo—that often work in practice. In the worst case, though, even to solve for the energy of the system, or for the state of some particular qubit, there's no known shortcut to dealing with the whole vector of 2^n amplitudes. So Feynman raised the question, “Why don't we build computers *out of qubits*, which themselves could take advantage of superposition

and interference and entanglement?” Of course he then faced the question: supposing we built such a computer, what would it be useful for? At that time, he was really only able to suggest one answer to that question: namely, it would be good for simulating quantum mechanics itself! If and when quantum computers really become practical, that’s probably still the most important application we know for them. In any case, no one knew at the time whether quantum computers would be useful for “classical” tasks as well: that’s a different, harder question, which will occupy us for much of the rest of the course.

We already have all the tools we need to discuss quantum computers.

The basic picture of a quantum computer is that it’s just a quantum circuit, but we’re jumping from working with 1, 2, or 3 qubits at a time to n qubits—where n could be, say, a million or more. You apply a sequence of gates on these qubits, each gate acting on just a few qubits (say, 1 or 2 of them), then measure some or all of the qubits.



Let’s address a few conceptual questions before proceeding further:

1. Will we be able to solve any problem on a quantum computer that literally *can’t* be solved on a classical computer, regardless of resources?

It’s easy to see that the answer is no. Using a classical computer, one can always simulate a quantum computer with n qubits, by just explicitly storing the vector of 2^n amplitudes (to some suitable precision, enough to approximate the final probabilities), and updating the vector whenever a unitary transformation gets applied. For this reason, we see that a quantum computer could “only, at most” achieve an exponential speedup over a classical computer: it could “only” falsify the Extended Church-Turing Thesis, rather than the original Church-Turing Thesis. Quantum computers might change what’s *computable in polynomial time*—how drastically, we don’t yet know—but they’re not going to change what’s computable at all, by letting us solve the halting problem or anything of that kind.

2. Why does each gate act on only a few qubits? Where is this assumption coming from?

It’s similar to how classical computers don’t have gates act on arbitrarily large numbers of bits, and instead use small gates like AND, OR, and NOT to build up complex circuits. The laws of physics provide us with *local* interactions—one particle colliding with another one, two currents flowing into a transistor, etc.—and it’s up to us to string together those local interactions into something more complicated.

In the quantum case, you could imagine a giant unitary matrix U , which takes n qubits, interprets them as encoding an instance of the 3SAT problem (or some other NP-complete problem), and then cNOTs the answer (1 for yes, 0 for no) into an $(n + 1)^{\text{st}}$ qubit. That U formally exists, is formally allowed by the rules of quantum mechanics. OK, but how would you go about actually implementing it? Well, you’d need to build it up out of smaller components—say, components that act on only a few qubits at a time.

It turns out that it’s possible to implement any unitary U you want, using exponentially many simple components (we’ll say more about that later in the lecture). The question that will interest us, in

quantum computing theory, is which U 's can be implemented using only polynomially many simple components. Those are the U 's that we'll consider "feasible" or "efficient."

3. What is the role of interference in quantum computing?

Quantum amplitudes can cancel each other out; that's the most important way in which they differ from classical probabilities. The goal, in quantum computing, is always to choreograph a pattern of interference such that, for each wrong answer, some of the contributions to its amplitude are positive and others are negative (or, they point every which way in the complex plane), so on the whole they *interfere destructively* and cancel each other out. Meanwhile, the contributions to the right answer's amplitude should *interfere constructively* (say, by being all positive or negative). If we can arrange that, then when we measure, the right answer will be observed with high probability.

Note that, if it weren't for interference, then we might as well have just used a classical computer with a random-number generator, and saved the effort of building a quantum computer. In that sense, all quantum-computational advantage relies on interference.

4. What is the role of entanglement in quantum computing?

In some sense, we can develop the entire theory of quantum computing without ever talking about entanglement. However, pretty much any time we're doing an interesting quantum computation, entanglement *is* something that will be there, "along for the ride." The reason for this is simply that an *unentangled* pure state of n qubits can always be written as

$$(\alpha_1|0\rangle + \beta_1|1\rangle) \otimes (\alpha_2|0\rangle + \beta_2|1\rangle) \otimes \cdots \otimes (\alpha_n|0\rangle + \beta_n|1\rangle).$$

Specifying such a state requires only $2n$ amplitudes, so we can store it efficiently. Thus, if for some reason the (pure) state of our quantum computer never had any entanglement in it, then the computer would be easy to simulate classically, and would be unable to achieve any speedup.

By contrast, a general entangled state of n qubits, $\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$, requires 2^n amplitudes to specify.

It very quickly becomes hopelessly intractable to store and manipulate all these amplitudes on a classical computer.

With 300 qubits, we already have more amplitudes to deal with than there are atoms in the observable universe.

Just to recap: it's a theorem, which we won't prove in this class, that *any* unitary transformation on *any* number of qubits n can be decomposed as a product of 1- and 2-qubit gates. However, if you just run the decomposition blindly, it will produce a quantum circuit with something like 4^n gates—just like, if you use the method of truth tables to build a circuit to compute some arbitrary Boolean function $f: \{0,1\}^n \rightarrow \{0,1\}$, you'll get something with about 2^n AND, OR, and NOT gates. Just like in the classical world, our real interest is in which Boolean functions can be *efficiently* computed—say, using a polynomial number of AND, OR, and NOT gates—so too in the quantum world, our real interest is in which n -qubit unitary transformations can be realized using only polynomially many 1- and 2-qubit gates.

But that being so, it behooves us to ask: is it possible that *all* n -qubit unitaries U can be realized by polynomial-size circuits?

The answer turns out to be no: in fact, just like “almost all” Boolean functions require exponentially large circuits to compute them, so too “almost all” unitaries require exponentially large quantum circuits. As in the classical case, the way to prove this is using a...

Counting Argument...

something that goes back to Claude Shannon in 1949. Shannon observed that almost every n -bit Boolean function requires a circuit of at least $\sim \frac{2^n}{n}$ AND, OR, and NOT gates to compute it. The reason for this, in one sentence, is that there are too many Boolean functions, and not enough small circuits! In more detail, there are 2^{2^n} different Boolean functions $f: \{0, 1\}^n \rightarrow \{0, 1\}$, but only $(n + T)^{O(T)}$ different circuits that take n inputs and that have at most T gates. Since each circuit can only compute one function, we can simply set the two expressions equal and solve to find that almost every function must require a circuit of nearly 2^n size.

Strikingly, and famously, this argument doesn't give us a single *example* of a hard-to-compute Boolean function. It merely tells us that such functions must exist, and be ubiquitous!

We can use a similar sort of argument in the quantum case—although, since $2^n \times 2^n$ unitary matrices form a continuous manifold, it's easier to talk in terms of dimensionality rather than cardinality. For simplicity, let's even restrict attention to those $2^n \times 2^n$ unitary matrices that are *diagonal*. Even then, specifying such a matrix clearly requires us to specify 2^n independent complex numbers of norm 1 (or $2^n - 1$, if we ignore global phase). By contrast, a quantum circuit with T gates, each acting on at most 2 qubits, can be specified using only $O(T)$ continuous parameters (plus some discrete choices about where to apply the gates, which won't affect the argument).

So we have a 2^n -dimensional manifold in the one case, and a union of $O(T)$ -dimensional manifolds in the other. Clearly, for the one to cover the other, we need T to grow at least like $\sim 2^n$. Hence there exist n -qubit unitary transformations U that require exponentially many gates to implement. In fact, “almost all” U 's (now in the technical, measure-theory sense of 100% of them) have this property.

You might complain that we've only showed that exponentially many gates are needed to implement most n -qubit unitary transformations *exactly*. What about approximately implementing them—that is, implementing them up to some small error ε (say, in each entry)? In fact, though, a little algebraic geometry (which we won't go into here) is enough to show that exponentially many gates are needed to *approximate* most n -qubit unitaries as well, or even most n -qubit diagonal unitary transformations.

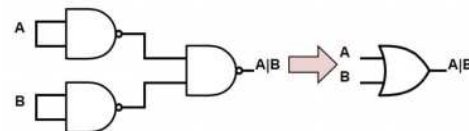
Once again, these arguments don't give us a single *example* of a hard-to-implement unitary transformation. They just tell us that that's the norm, that the easy-to-implement unitaries are the rare exceptions! Yet, rare though they might be, the subset of unitaries that are easy (i.e., that can be implemented by polynomial-size quantum circuits) are the main ones that will interest us in quantum computing.

Up till now, we've assumed that *any* 1- and 2-qubit gates are available, but is that assumption realistic? Is it necessary? This brings us to our next topic...

Universal Gate Sets

In classical computing, you're probably familiar with logic gates like AND, OR, NOT, NAND, etc. A (classical) gate set is called *universal* if, by stringing together enough gates from the set, you can express any Boolean function on any number of bits.

For example, the NAND gate by itself is universal. The diagram on the right shows how you'd construct an OR gate out of NANDs. You can also work out how to construct an AND gate and a NOT gate, and from there you can get anything else.



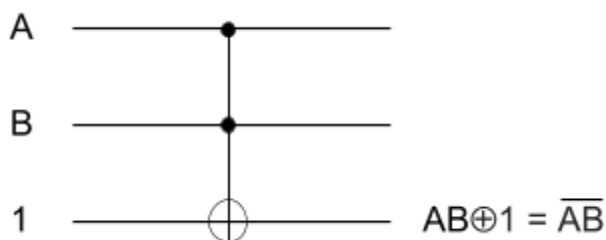
By contrast, the set {AND,OR} is *not* universal, because it can only express monotone Boolean functions: that is, changing an input bit from 0 to 1 can never change an output bit from 1 to 0. Likewise, the set {NOT,XOR} is not universal, because it can only express Boolean functions that are linear or affine mod 2. So, while “most” gate sets are universal, being universal isn't completely automatic.

Next let's discuss *classical reversible gates* (that is, reversible mappings from k -bit strings to k -bit strings), where we'll find that something similar happens. We call a set of reversible gates universal if, by composing enough of them, you can express any reversible transformation on any number of bits n .

Here we allow the use of “ancilla bits” (that is, bits other than the n being acted on), as long as the ancilla bits are returned to their initial states by the end of the computation. The use of ancilla bits is provably necessary for universality in this setting.

The most famous example of a universal reversible gate—the reversible analogue of the NAND gate, if you like—is called the **Toffoli gate**. The Toffoli gate, also known as controlled-controlled-NOT, is a 3-bit gate that flips the third bit *if and only if* the first two bits are both set to 1.

To show that Toffoli is at least capable of universal computation, we construct a NAND gate out of one (in the diagram on the right). Because Toffoli can simulate NAND, it can also simulate any ordinary (non-reversible) Boolean circuit.



The third bit ends up as 0 if only if both A and B are both 1, meaning that the third bit gets flipped. Thus, we've got a NAND gate.

Note that this argument does *not* yet establish that Toffoli is a universal reversible gate, in the sense we defined above—because for that we need to implement all possible reversible transformations, not just compute all Boolean functions. However, a more careful argument, which we'll give later in the course, shows that Toffoli really is universal in that stronger sense as well.

A good example of a gate that separates the two kinds of universality is the so-called **Fredkin gate**, which is also called **Controlled-SWAP** or **CSWAP**. This is a 3-bit gate that swaps the second and third bits, if and only if the first bit is set to 1. Just like the Toffoli gate, the Fredkin gate can simulate a NAND gate (exercise to see how), and is therefore capable of universal computation. However, Fredkin is *not* universal in the stronger sense, because it can never change the total number of 1's in the input

string (the so-called *Hamming weight*). For example, there's no way, by composing any number of Fredkin gates, to map the string 000 to 111. A gate with this property, of always preserving the Hamming weight, is called *conservative*.

There are also reversible gates that are not even capable, on their own, of universal computation. An important example is the Controlled-NOT or CNOT gate, which we saw earlier. By composing CNOT gates, we can only express Boolean functions that are affine mod 2: for example, we can never express AND or OR. More generally, and in contrast to the irreversible case, it turns out that *no* 2-bit classical reversible gate is capable of universal computation. For universality, we need reversible gates (such as Toffoli or Fredkin) that act on at least 3 bits.

It's worth noting that any classical reversible gate can *also* be used as a quantum gate (i.e., it's unitary). So in particular, Toffoli can be used as a quantum gate. So we see that, if nothing else, a quantum circuit can compute any function that a classical circuit of similar size can compute: we simply need to transform the classical circuit into one made of Toffoli gates.

We're now ready to talk about universal quantum gate sets.

We'll call a set S of quantum gates *universal* if, by composing gates from S , you can approximate any unitary transformation on any number of qubits to any desired precision. Note that, if S is finite, then approximation is all we can hope for, because there are uncountably many unitary transformations, but only a countable infinity of quantum circuits built out of S -gates.

Just like with classical reversible gates, there are weaker kinds of universality for quantum gate sets that are often enough in practice. That is, even if gates *can't* be used to approximate an arbitrary unitary to any desired precision, they'll often anyway suffice for universal quantum computation. We'll see examples of this soon.

First, though, let's list some of the ways a quantum gate set could be limited, in such a way that it *fails* to be universal. We'll discuss such four ways: three of them obvious and one of them not.

1. Your gate set doesn't create interference/superposition

Example: The set {CNOT} can only map computational basis states, like $|10\rangle$, to other computational basis states, like $|11\rangle$. It can maintain existing superpositions, but it can't *create* a superposition of basis states where there wasn't one already.

2. Your gate set can create superpositions, but not entanglement

Example: The set {Hadamard} can map $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$, thereby creating a superposition. But it should be obvious that the Hadamard gate can't map a product state to an entangled state, since it acts on only one qubit. In fact, any quantum circuit made of Hadamard gates—or any 1-qubit gates, for that matter—will just act independently on each of the n qubits, so it will be trivial to simulate classically.

3. Your gate set only has real gates

Example: The set {CNOT, Hadamard} is getting closer to universality, as it's capable of creating entangled superposition states. But it's still not there, as can be seen from the fact that the CNOT and

Hadamard matrices have real entries only. So composing them could never give us a unitary transformation like

$$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

4. Your gate set is “contained in the stabilizer set”

This is the non-obvious case. Later in the course, we’ll explain stabilizer gates in more detail, as well as their central importance for quantum error correction. For now, though, the *stabilizer gates* are the following three quantum gates: cNOT, Hadamard, and

$$\text{Phase} = P = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

These gates pass every test for universality that we saw before: they can generate superposition, *and* entanglement, *and* amplitudes that aren’t real. Furthermore, they’re enough to demonstrate many (though not all) of the quantum phenomena that we’ve seen in this course, such as teleportation and superdense coding.

Nevertheless, it turns out that the particular set

{CNOT, Hadamard, P}

is not universal. Indeed, a famous result called the **Gottesman-Knill Theorem** shows that these gates generate only a discrete subset of unitary transformations—and that furthermore, any circuit made of stabilizer gates, and applied to the initial state $|0 \dots 0\rangle$, can be simulated in polynomial time on a classical computer. Thus, despite containing an interesting subset of quantum mechanics, these gates still aren’t enough to realize exponential quantum speedups.

Are there any other ways for a set of quantum gates, acting on qubits, to fail to be universal?

That’s currently an open question! It’s one of Prof. Aaronson’s personal favorites. Not many people in the field care about this particular question, since “we have lots of universal gate sets that work, so just roll with it,” but it would be nice to know, and you should go solve it.

So then which quantum gate sets *are* universal?

It turns out that, in the stabilizer set {CNOT, Hadamard, P}, if you swap out the Hadamard gate for nearly anything else, then the set becomes universal.

So for example, {cNOT, $R_{\frac{\pi}{8}} = \begin{bmatrix} \cos \frac{\pi}{8} & -\sin \frac{\pi}{8} \\ \sin \frac{\pi}{8} & \cos \frac{\pi}{8} \end{bmatrix}$, P } is universal.

Also, {Toffoli, Hadamard, P} is universal, by a 2002 result of Yaoyun Shi.

Also, if you just pick a 2-qubit gate uniformly at random, then it’s known to have a 100% chance of being universal. With universality, the whole difficulty comes from the remaining 0% of gates!

Above, we listed bad cases—ways of failing to be universal—but there are also general criteria such that if your gate set meets them then it *is* universal. We won’t cover this, but see (for example) the paper of Shi, or earlier literature on the subject, for examples of such criteria.

So far, in discussing universal gate sets, we’ve swept an important question under the rug. Namely, a universal gate set lets us approximate any unitary to any desired accuracy ε —but how does the

number of gates needed with ε ? If, for example, the number scaled like $2^{1/\varepsilon}$, then our gate set would be “universal” in only the most theoretical sense. Fortunately, there’s a central result in quantum computing theory, called the...

Solovay-Kitaev Theorem

which assures us that this never happens! In more detail, call a gate set S *closed under inverses* if, whenever a gate G is in S , the inverse gate G^{-1} is also in S . Then the Solovay-Kitaev Theorem says that, using any universal gate set G that’s closed under inverses, we can approximate any unitary on n qubits to within precision ε (say, entrywise precision) using only $O(4^n \text{ polylog}(\frac{1}{\varepsilon}))$ gates. In other words, if we treat n as fixed—say, if we’re just trying to emulate a gate from one universal set using gates from a different universal set—then the complexity scales only like some power of $\log(\frac{1}{\varepsilon})$. This means that *all* universal gate sets, or at least the ones closed under inverses, “fill in the space of all unitary transformations reasonably quickly.” Furthermore, the gate sequences that achieve the Solovay-Kitaev bound can actually be found by a reasonably fast algorithm.

Whether the closed-under-inverses condition can be removed remains an unsolved problem to this day.

With the original proofs of Solovay-Kitaev, from the late 1990s, the number of gates needed grew like $\log^{3.97}(\frac{1}{\varepsilon})$. However, more recently it’s been shown that, at least if we use special universal gate sets arising from algebra and number theory, we can get the number of gates to grow only like $\log(\frac{1}{\varepsilon})$ —which is not only much more practical, but also the best one could possibly hope for on information-theoretic grounds.

The proof of Solovay-Kitaev is beyond the scope of this course, but it’s contained in many quantum computing textbooks (including Nielsen & Chuang), and is something you should know is out there.

Lecture 17, Thurs March 23: Quantum Query Complexity, Deutsch-Jozsa

People often want to know where the true power of quantum computing comes from.

- Is it the ability of amplitudes to interfere with one another?
- Is it the huge size of Hilbert space (the space of possible quantum states)?
- Is it that entanglement gives us 2^n amplitudes to work with?

But that's sort of like dropping your keys and asking "what made them fall?"

- Is it their proximity to the Earth?
- Is it the curvature of spacetime?
- Is it the fact that you dropped them?

There can be many complementary explanations for the same fact, all of them valid. And that's the case here. If there weren't a huge number of amplitudes, quantum mechanics would be easy to simulate classically. If the "amplitudes" were probabilities, rather than complex numbers that could interfere with each other, QM would also be easy to simulate classically. If no entanglement were allowed, then at least all pure states would be product states, once again easy to represent and simulate classically. If we were restricted to stabilizer operations, QM would be easy to simulate classically. But as far as we know, full QM---involving interference among exponentially many amplitudes in complicated, non-stabilizer entangled states---is hard to simulate classically, and that's what opens up the possibility of getting exponential speedups using a quantum computer.

Quantum Complexity

There are two major ways we look at the complexity of quantum algorithms

The circuit complexity of a unitary transformation U is the size (i.e., number of gates) of the smallest circuit that implements U . We like unitaries with polynomial circuit complexity. Alas, typically it's *extremely* hard to determine the circuit complexity of a unitary; the best we can do is to prove upper bounds, and conjecture lower bounds on the basis of hardness assumptions and reduction arguments. Note that the reasons why this sort of problem is insanely hard have nothing to do with quantum mechanics.

"What's the smallest circuit that solves Boolean satisfiability?"
is a similarly hard problem, indeed closely related to P vs NP.

So if we want a more precise picture of what's going on, albeit in a more limited model, we instead often use...

Query complexity, which is the number of calls the algorithm makes to an oracle (or black box function). The idea is that your oracle takes an input x and produces an output $f(x)$, where say

$$f : \{0, 1\}^n \rightarrow \{0, 1\}$$

is a Boolean function. In quantum mechanics, our first guess for what this would mean might be: we map the input state $|x\rangle$ to the output state $|f(x)\rangle$, or maybe the output state $|x\rangle|f(x)\rangle$.

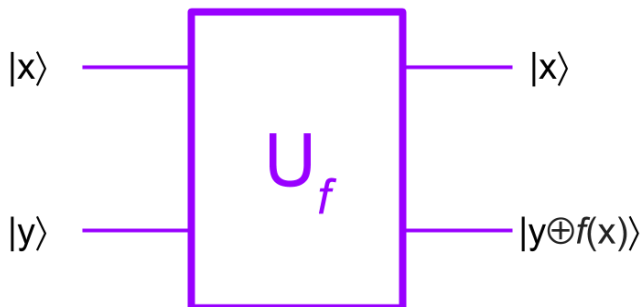
Here, though, we run into a bit of trouble because such a transformation is not unitary. To make the transformation unitary, we need a so-called *answer* or *target* register.

So we give the black box two inputs: x , which is unchanged, and y , which has the answer $f(x)$ written into it in a reversible way, typically exclusive-ORing:

$$|x, y\rangle \rightarrow |x, y \oplus f(x)\rangle$$

If we took care to ensure that $y = 0$ initially, then this would map $|x, 0\rangle$ to $|x, f(x)\rangle$, exactly like in the classical case.

A lot of times we ignore the answer qubit by moving the phases around. So let's say we prepare the answer qubit as $|-\rangle$.



One important note for later: if f happens to be a Boolean function, then often it's most convenient to consider quantum queries that map each basis state $|x\rangle$ to $(-1)^{f(x)} |x\rangle$: in other words, that *write the function value into the phase of the amplitude*, rather than storing it explicitly in memory. Or more precisely, we consider queries that map each basis state $|x, b\rangle$ to $(-1)^{f(x) \cdot b} |x, b\rangle$, where b is a bit that controls whether the query should even take place or not. This sort of “phase oracle” doesn't really have any classical counterpart, but is extremely useful for setting up desired interference patterns.

So it behooves us to ask: how do the “phase oracle” and the “XOR oracle” compare? Could one be more powerful than the other? Happily, it turns out that they're *equivalent*, in the sense that either can be used to simulate the other, with no cost in the number of queries. This is a result that we'll need later in this lecture.

To see the equivalence, all we need to do is consider what happens when the second register—the one containing y or b —is placed in the $|-\rangle$ state before a query. You can check that this converts a XOR oracle into a phase oracle: we get

$$\frac{1}{\sqrt{2}} (|x, 0\rangle - |x, 1\rangle) \rightarrow \frac{1}{\sqrt{2}} (|x, 0 \oplus f(x)\rangle - |x, 1 \oplus f(x)\rangle)$$

which equals $\frac{1}{\sqrt{2}} (|x, 0\rangle - |x, 1\rangle)$ if $f(x) = 0$

$$\frac{1}{\sqrt{2}} (|x, 0\rangle + |x, 1\rangle) \quad f(x) = 1$$

which we can rewrite as just $(-1)^{f(x)} |x, -\rangle$. Meanwhile, if the second register is placed in the $|+\rangle$ state, then nothing happens, so we really do get the $|x, b\rangle \rightarrow (-1)^{f(x) \cdot b} |x, b\rangle$ behavior.

Conveniently, the converse is also true! That is, if we know that a phase oracle will be acting, then by placing the b register in one of the states $|+\rangle$ or $|-\rangle$, we can simulate the effect of a XOR oracle, with the phase oracle causing $|+\rangle$ and $|-\rangle$ to be swapped if and only if $f(x) = 1$.

Taking a step back, though, what are we really doing when we design a quantum algorithm in the query complexity model? We're abstracting away part of the problem, by saying:

“Our problem involves some function, say, $f : \{0, 1\}^n \rightarrow \{0, 1\}$, and we’re trying to learn some property of f , in a way that only requires evaluating f on various inputs, not looking at the internals of how f is computed.”

So for example, you might want to learn:

Is there some input x such that $f(x) = 1$?

Does $f(x) = 1$ for the majority of x ’s?

Does f satisfy some global symmetry, such as periodic, or is it far from any function satisfying the symmetry?

Etc.

We then want to know how many queries to f are needed to learn this property.

In this model, we abstract out the cost of any quantum gates that are needed before or after the queries: those are treated as “free.”

Before we delve deeper, it’s worth asking:

“Why do we care about the black-box model? You’re debating how you’d phrase your wishes if you found a magical genie. Who cares?”

The truth is more prosaic, though. You can think of a black box as basically a huge input string. From that standpoint, querying $f(i)$ just means looking up the i^{th} element in the string.

$f(0)$	$f(1)$	\dots	$f(n)$
--------	--------	---------	--------

Another way to think about it:

Imagine I’m writing code, and I have a subroutine that computes $f(x)$. How many times do I need to call the subroutine to find some information about f ? Assuming, in the quantum case, that I even get to call the subroutine on a superposition of different x values, and get back a superposition of answers?

But also assuming that we’re not going to “violate abstraction boundaries” by examining the code of the subroutine.

To justify the quantum black-box model, there’s one technical question we need to answer...

Suppose we did have a small circuit to compute a function f . Could we then implement the quantum black-box behavior that we described above—without loss of generality, the XOR behavior.

$|x, a\rangle \rightarrow |x, a \oplus f(x)\rangle$?

The reason this isn’t entirely obvious, is that if you’ll recall, quantum circuits have to be *reversible*. So just because there’s a small circuit to compute $f(x)$, doesn’t immediately imply that there’s a small circuit that maps the basis state $|x, a\rangle$ to the basis state $|x, a \oplus f(x)\rangle$ *with nothing else lying around*.

Indeed, let’s step back, and think about the constraints on computation that are imposed by reversibility. To start with the obvious: if we had a reversible circuit that maps $|x\rangle$ to $|g(x)\rangle$, then g must be an injective function. But now for the subtle part: even if g is both injective *and* efficiently

computable, that still doesn't imply (at least, as far as we know) that the map $|x\rangle \rightarrow |g(x)\rangle$ is efficiently computable.

Why not? Well, imagine that g were an injective one-way function: that is, a function that's easy to compute but hard to invert. Such functions are the basic building blocks of cryptography, and are strongly conjectured to exist, even in a world with quantum computers.

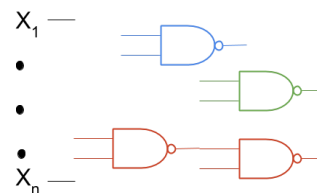
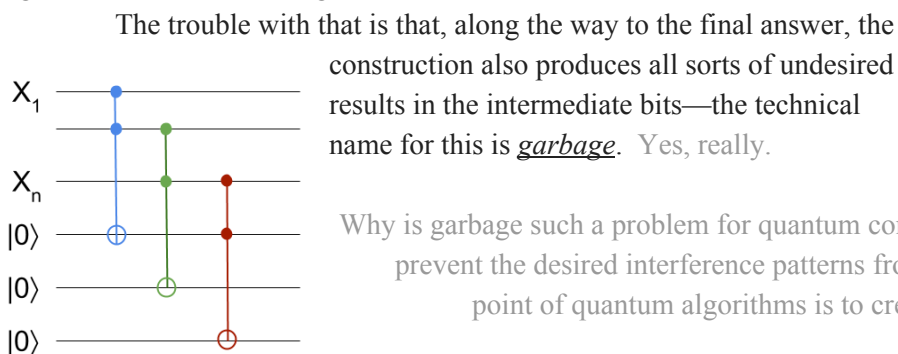
Note that even though quantum computers can break a few supposedly one-way functions, like those based on factoring and discrete log, there are many, many more "generic," less "structured" one-way functions that don't seem threatened by quantum computers. We'll have more to say about such issues later.

Anyway, now suppose we had a small circuit C such that $C|x\rangle = |g(x)\rangle$. Then simply by running that circuit backwards—that is, inverting all the gates and reversing their order—we could get $C^{-1}|g(x)\rangle = |x\rangle$, thereby inverting the supposed one-way function!

But why doesn't this contradict our starting assumption that g was easy to compute? Here's why: because a reversible circuit for g would at best give us a $|x\rangle|0\rangle \rightarrow |x\rangle|g(x)\rangle$ mapping like , a mapping that leaves x around afterward. And inverting *that* mapping will only $|g(x)\rangle$ take us from to x if we know x already, so it's no help in breaking the one-way function.

OK, but this still leaves the question: how do we even efficiently implement the mapping $|x\rangle|0\rangle \rightarrow |x\rangle|g(x)\rangle$, if given a small *non-reversible* circuit for g ?

In the last lecture, we saw how it's possible to take any non-reversible circuit and simulate it by a reversible one, by using Toffoli gates to simulate NAND gates.



Why is garbage such a problem for quantum computing? Because garbage can prevent the desired interference patterns from showing up—and the whole point of quantum algorithms is to create those interference patterns.

For example, what's the difference between having $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and having $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$, where we treat the second qubit as unwanted garbage?

The garbage is entangled with the first qubit, the qubit we *do* want. So, in the second case, when we look at the first qubit only, we see the maximally mixed state rather than the $|+\rangle$ state that we wanted.

So to return to the question: suppose you have a circuit to compute f . How you we get a circuit that maps $\sum_x \alpha_x |x, 0\rangle \rightarrow \sum_x \alpha_x |x, f(x)\rangle$ without all the garbage? Back in the 1970s, Charles Bennett invented a trick for this called...

Uncomputing

It's simple, though also strange when you first see it. The trick to getting rid of garbage is to run computations *forward and then in reverse*.

Let's say I have some circuit C such that

$$C|x, 0, \dots, 0\rangle = |x, \text{gar}(x), f(x)\rangle,$$

where $\text{gar}(x)$ is a generic term for all the garbage: that is, byproducts of the computation that I don't want. Then I do the following:

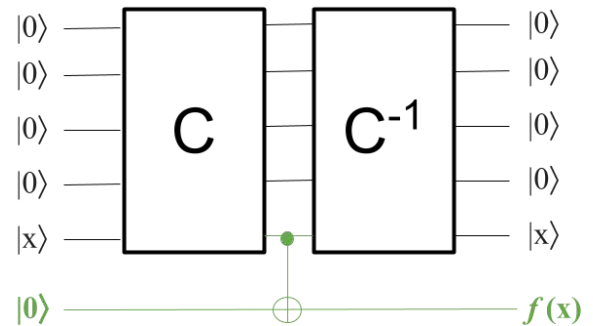
First run the circuit C , to get $|x, \text{gar}(x), f(x)\rangle$.

Then cNOT the answer $f(x)$ into a register initialized to 0, to get $|x, \text{gar}(x), f(x), f(x)\rangle$
(in other words, make a copy of $f(x)$ in a safe place)

Finally, run the inverse circuit, C^{-1} , to get $|x, 0, \dots, 0, f(x)\rangle$ or just $|x, f(x)\rangle$ if we ignore the 0 qubits.

The reason why we can copy x , in spite of the No-Cloning Theorem, is that we're assuming that x is a classical answer. This won't work if the output is a general quantum state.

This justifies the quantum query model because if we can compute f at all, then we do have the ability to map $|x, a\rangle$ to $|x, a \oplus f(x)\rangle$. (Note that in the uncomputing process, if the "safe" register was initialized to some arbitrary a rather than to 0, then it would end up in the $a \oplus f(x)$ state.)



With that out of the way, we're finally ready to talk about some quantum algorithms.

Deutsch's Algorithm

was, by some definition, the first quantum algorithm proposed, in the mid-1980s. It achieves something unimpressive, except for the fact of being possible at all: namely, it computes the parity of two bits using only one (superposed) query to the bits.

In more detail, we're given two unknown bits, b_0 and b_1 .

Given an index $x \in \{0, 1\}$, our oracle returns the bit. i.e. $f(x) = b_x$

What we want to know is, "What is the parity of these bits?"

Parity is whether the bits have different values, so $b_0 + b_1 \pmod{2}$ or $b_0 \oplus b_1$

Classically, this would clearly take two queries since we need to know both bits. So using a quantum algorithm, how do we do it in one?

Simple: we start with a qubit at $|0\rangle$, Hadamard it to get $|+\rangle$, then apply a phase query, which applies a phase change to each branch of the superposition depending on the value of the function. (Here

we're using the result from earlier in this lecture, that we can use a single "ordinary" query to simulate the effect of a single phase query.) This yields:

$$|0\rangle \rightarrow \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \rightarrow \frac{1}{\sqrt{2}} ((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle)$$

Let's factor out $f(0)$ to get

$$= \frac{1}{\sqrt{2}} (-1)^{f(0)} (|0\rangle + (-1)^{f(1)-f(0)}|1\rangle)$$

So now if $f(0) = f(1)$ we get $(-1)^{f(0)} \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)$

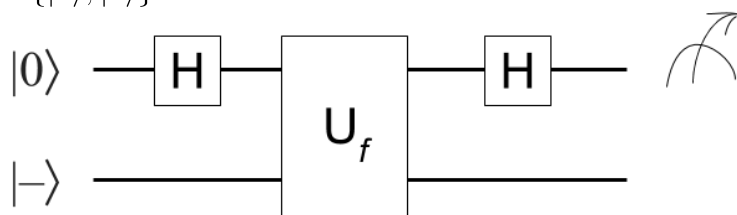
while if $f(0) \neq f(1)$ we get $(-1)^{f(0)} \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$

We can ignore the phase out front since global phase doesn't affect measurement, and then Hadamard again to get our quantum states back in the $\{|0\rangle, |1\rangle\}$ basis.

Now in the $f(0)=f(1)$ case we get $|0\rangle$

and in the $f(0)\neq f(1)$ case we get $|1\rangle$.

The complete quantum circuit is shown on the right.



Note that, if we wanted the parity of an n -bit input string x , Deutsch's algorithm would let us get that with $n/2$ queries. We simply need to break x up into $n/2$ blocks of 2 bits each, use Deutsch's algorithm to learn the parity of each block B (using $n/2$ queries in total), then calculate the parity of all the 's. This last step doesn't increase the query complexity, because it doesn't involve making any additional queries to x .

OK, if we understand Deutsch's algorithm, then let's next see a generalization, called...

The Deutsch-Jozsa Algorithm

Suppose we have a black box that computes a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, and suppose we're promised that f is either:

- a constant function All outputs are 0 or all outputs are 1
- a balanced function There are the same number of 0 outputs as 1 outputs

The problem is to decide which.

Classically, deterministically, you could solve this problem by examining any $2^{n-1} + 1$ values of the function. If all the values match, then the function is constant; otherwise the function is balanced. If you want no possibility of error, then it's not hard to see that this is the best you can do.

Of course, you can do much better by using random sampling. On average, you'd need maybe 5 or 6 queries—or at any rate, a constant number—to get an answer with small probability of error. If all of your samples match, you can guess that the function is constant; if they don't, you know that it's balanced.

What the Deutsch-Jozsa algorithm does, is to solve the problem *perfectly* (that is, with zero probability of error), with only one quantum query. That's something that isn't possible in the classical world.

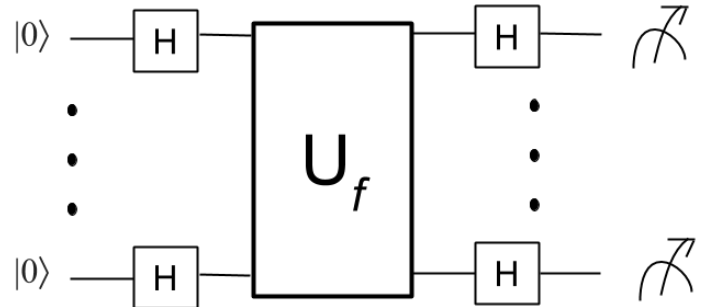
Truth is, this speedup still isn't all that impressive, because the classical probabilistic algorithm

is nearly as fast, and would be perfectly fine in practice. This helps to explain why, until 1994 or so, most people didn't care about quantum computing. To whatever extent they looked into it at all, they figured all quantum speedups would be in the same vein.

Anyway, here's the quantum circuit for Deutsch-Jozsa:

You'll begin to notice that some patterns appear a lot in quantum algorithms.

- You start by putting everything in superposition
- You then query a function f —in this case, mapping each basis state $|x\rangle$ to $(-1)^{f(x)}|x\rangle$
- You then apply a change a basis (in this case, another round of Hadamards)
- Finally, you measure to get the information you want to know



If you can't figure out what to do next in a quantum algorithm, a round of Hadamards is always a good guess!

So given this circuit (call it C), let's ask: what's the probability of getting back the state $|00 \dots 0\rangle$?

In other words, what is $|\langle 00 \dots 0 | C | 00 \dots 0 \rangle|^2$?

Well, the Hadamard gate maps $|0\rangle \rightarrow |+\rangle$ and $|1\rangle \rightarrow |-\rangle$.

We can summarize this by saying that, for a bit $x \in \{0, 1\}$, it maps $|x\rangle \rightarrow \frac{|0\rangle + (-1)^x |1\rangle}{\sqrt{2}}$

So given a string $|x(1), \dots, x(n)\rangle$, Hadamarding all n of the qubits produces the state

$$\frac{|0\rangle + (-1)^{x(1)}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + (-1)^{x(2)}|1\rangle}{\sqrt{2}} \otimes \dots \otimes \frac{|0\rangle + (-1)^{x(n)}|1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle$$

This is a fact that we'll also have several occasions to use in the next lecture.

Note: Here $x \cdot y$ denotes the inner product. The formula is saying that we pick up a -1 phase for every i such that $x_i = y_i = 1$.

Now, coming back to the Deutsch-Jozsa algorithm, after we Hadamard all n of the qubits and then query the oracle, we get the state

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle$$

So by our previous result, after the second round of Hadamards we get

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle$$

Rather than simplifying this entire sum, let's take a shortcut by just asking: what is the amplitude for the basis state $y = |00 \dots 0\rangle$?

$$\text{Well, it's } \frac{1}{2^n} \sum_{x \in \{0,1\}} (-1)^{f(x)}.$$

Now, what does this amplitude have to do with whether f is constant or balanced?

Well, if f is constant, then the above amplitude is either 1 (if f is identically 0) or -1 (if f is identically 1).

On the other hand, if f is balanced, then the amplitude is 0.

So when we measure, if we see the outcome $|00 \dots 0\rangle$ then we know that f is constant, and if we see any other outcome then we know that f is balanced! That was the Deutsch-Jozsa algorithm.

The first problem we'll see in the next lecture is the so-called *Bernstein-Vazirani problem*, for which there's a quantum algorithm that achieves a more impressive speedup than Deutsch-Jozsa does. And the speedups will continue to get more impressive from there.

Lecture 18, Tues March 28: Bernstein-Vazirani, Simon

We ended last time with the Deutsch-Jozsa problem. Today we'll start with another black-box problem for which quantum algorithms provide an advantage:

The Bernstein-Vazirani Problem

We're given access to a black box function $f : \{0, 1\}^n \rightarrow \{0, 1\}$

We're promised that $f(x) = s \cdot x \pmod{2}$ for some secret string $s \in \{0, 1\}^n$

The problem is to find s .

Classically, you could get an answer one bit at a time by querying all the strings of Hamming weight one: for example, $f(1000) = s_1$

$$f(0100) = s_2$$

$$f(0010) = s_3$$

$$f(0001) = s_4$$

But no classical algorithm can do better than this, since each query can only provide one bit of information about s .

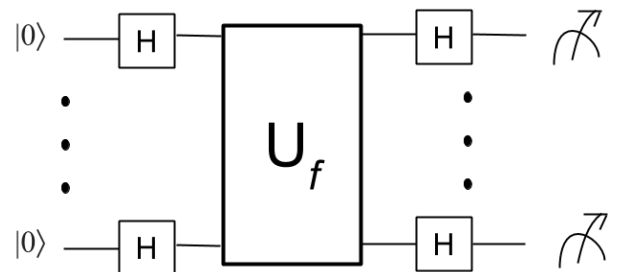
The Bernstein-Vazirani algorithm, however, solves the problem quantumly using only one query (!).

The Bernstein-Vazirani Algorithm

We start with n qubits all in the $|0\rangle$ state.

We then Hadamard them all (what else?).

Next we query f (using the 'phase' type of query).



The question is:

How do we measure the resulting state in a way that gives us information about the secret string s ?

We can reuse work from the last lecture...

We know that Hadamard gate maps $|0\rangle$ to $|+\rangle$ and $|1\rangle$ to $|-\rangle$ (and vice versa)

$$\text{i.e. it does } |b\rangle \leftrightarrow \frac{|0\rangle + (-1)^b |1\rangle}{\sqrt{2}}$$

and so Hadamarding a string of bits gets us

$$|s_1, \dots, s_n\rangle \leftrightarrow \frac{|0\rangle + (-1)^{s_1} |1\rangle}{\sqrt{2}} \otimes \dots \otimes \frac{|0\rangle + (-1)^{s_n} |1\rangle}{\sqrt{2}}$$

We pick up a -1 factor only when s_i and x_i are both 1.

Now, note that we can write the current state of the Bernstein-Vazirani algorithm as

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{s \cdot x} |x\rangle = \frac{|0\rangle + (-1)^{s_1} |1\rangle}{\sqrt{2}} \otimes \dots \otimes \frac{|0\rangle + (-1)^{s_n} |1\rangle}{\sqrt{2}}$$

But this means that if we Hadamard all the qubits again, we'll change:

- The qubits that picked up a phase (i.e., for which $s_i = 1$) from $|-\rangle$ to $|1\rangle$.
- The qubits that *didn't* pick up a global phase (i.e., for which $s_i = 0$) from $|+\rangle$ to $|0\rangle$.

So from here we can simply measure the qubits in the $\{|0\rangle, |1\rangle\}$ basis to retrieve s .

You can see that Bernstein and Vazirani designed their problem around what a quantum computer would be able to do!

Don't tell anyone, but: this is actually pretty common in this field

So, only for $|s\rangle$ are all 2^n contributions to the amplitude of a measurement outcome pointing the same way. For all the other outcomes, the contributions interfere destructively, with equally many positive and negative terms (all of the same magnitude), so the total amplitude for each of those outcomes is 0.

With pretty much *every* quantum algorithm, a similar story can be told, about the contributions to the amplitude reinforcing each other only for the outcomes that we want.

On that note, let's next see...

Simon's Problem (1994)

The story goes that Simon looked at the quantum algorithms coming out, and he didn't believe that any of them would give a *real* speedup. Even the Bernstein-Vazirani problem is easy classically: a classical computer can find the n bits of s with n queries. Sure, the quantum algorithm needs only one query, but it also requires $O(n)$ gates, so maybe it's not that impressive.

Simon believed there was a limit that would prevent you from getting a “true” exponential speedup from a quantum computer, and he set out to prove it. What he ended up finding, instead, was that there *is* a true exponential speedup, at least for an artificial black-box problem. As we'll see, this then played a central role in subsequent progress in quantum algorithms: particularly Shor's algorithm, which came shortly afterward.

In Simon's problem, we're once again given an oracle function, this time mapping n bits to n bits:

$$f : \{0, 1\}^n \rightarrow \{0, 1\}^n$$

We're promised that there's a secret string $s \neq 0^n$, such that

$$f(x) = f(y) \Leftrightarrow y = x \oplus s$$

for all inputs x, y , where the symbol \oplus denotes bitwise XOR. The problem is to find the secret string s , by querying f as few times as possible.

Compared to Bernstein-Vazirani, here there's more freedom in the choice of function f . In Simon's problem, all we require is that f has a “hidden XOR mask”: that is, a subset of bits such that when you flip the bits in that subset (but *only* when you do so), the output is unaffected.

What does this mean?

Let's do an example with 3-bit inputs, and with secret string $s = 110$.

Let's say we query f a few times and get...

$$f(000) = 5$$

$$f(110) = 0$$

$$f(001) = 6$$

$$f(111) = 6$$

We're given no information about how to interpret the outputs themselves, so it doesn't really matter whether we think of them as strings, integers, or whatever. The only thing that *does* matter is whether two inputs map to the same output.

Since $f(001) = f(111) = 6$, we know that $s = 001 \oplus 111 = 110$.

This is simple enough with 3 bits, but we're more interested in f 's with, say, 1000-bit inputs. In that case, we claim that finding the secret string s is prohibitively expensive classically. How expensive? Well, it's not hard to show that any deterministic algorithm needs to query f at least $2^{n-1} + 1$ times, by an argument similar to the one that we used for Deutsch-Jozsa. But once again, the more relevant question is how many queries are needed for a *randomized* algorithm.

We claim that we can do a little bit better in that case, getting down to $O(2^{n/2})$ queries. This is related to the famous **Birthday Paradox**, which isn't really a paradox, it's more of a "birthday fact."

It says that, if you gather merely 23 people in a room, that's enough to have a ~50% probability of getting at least one pair of people who share a birthday. More generally, if there were n days in the year, then you'd need \sqrt{n} about people in the room for a likely birthday collision. (At least, assuming birthdays are uniformly distributed, which they're not exactly: e.g., there are clusters of them about 9 months after major holidays.)

The takeaway here is that the number of pairs of people is what's important, and that scales quadratically with the number of people. Similarly, with Simon's problem, the number of pairs of inputs that could collide is what's important, and that grows quadratically with the number of inputs we check.

The Birthday Paradox is useful in cryptanalysis.

For example, cryptographic hash functions need to make it intractable to find any two inputs x, y with the same hash value, $f(x) = f(y)$. But by using a "birthday attack"—i.e., repeatedly choosing a random input x , then comparing $f(x)$ against $f(y)$ for *every* previously queried input y , looking for a match—we can find a collision pair using a number of queries to f that scales only like the square root of the size of f 's range. This is quadratically faster than one might have expected naïvely.

Whatever other structure it has, Simon's problem involves a two-to-one function in which we're looking for a collision pair—so it also admits a birthday attack. Roughly speaking, given two randomly-chosen inputs x and y , we'll observe $f(x) = f(y)$ with probability $p = \frac{1}{2^{n-1}}$, and while these events aren't quite independent between the various (x, y) pairs, they're nearly so. Doing the calculation carefully, we find that we'll observe a collision with high probability after querying $\frac{1}{\sqrt{p}} \sim 2^{n/2}$ values of f .

Is there a better classical algorithm?

Let's prove that the answer is no. We'll use an **Adversary Argument**: basically,

"If my worst enemy got to choose f , what would he do?"

Presumably, he'd choose a secret string $s \neq 0^n$ uniformly at random among all possible s 's, to make it as hard as possible to find an underlying structure in f . And then, perhaps, choose a random f among all those consistent with that s .

Now, once we fix such a strategy of the adversary, we can assume without loss of generality that the algorithm is deterministic. This is because a randomized algorithm is just a probabilistic mixture of deterministic algorithms, and there must be at least one algorithm in the mixture that does at least as well as the average! (This observation—together with the complementary observation that *all* randomized lower bounds can be proved in this way—is sometimes referred to as *Yao's minimax principle*.)

So the upshot is that we can view an optimal strategy as just a deterministic sequence of queries. Let the queried inputs be x_1, x_2 , etc. Then the question is:

What information can we derive about s after the first t queries?

If we've found a collision pair, $f(x_i) = f(x_j)$ for some $i \neq j$, then we're done; we now know that $s = x_i \oplus x_j$. So let's assume that hasn't happened yet. Then all we can conclude about s is that $s \neq x_i \oplus x_j$ for every $i \neq j$. At most this rules out t^2 possible values of s , with all the other possibilities remaining equally consistent with what we've seen (i.e., having equal posterior probabilities). It follows that, *unless* we observe a collision, narrowing the possibilities down to a single s requires $\Omega(2^{n/2})$ queries. And the probability that we *do* observe a collision in the t^{th} query is only $\frac{t}{2^n - 1}$ —so by the union bound, with high probability we won't observe any collisions at all in the first $\sim 2^{n/2}$ queries.

And that's the adversary argument: examining a "worst-case" distribution over f 's gives us a lower bound of $\Omega(2^{n/2})$ queries for any randomized algorithm solving Simon's problem.

i.e. the birthday attack yields a quadratic speedup, but the problem still takes exponential time classically.

Quantumly, by contrast, we can solve Simon's problem with only $O(n)$ queries to f , by using...

Simon's Algorithm!

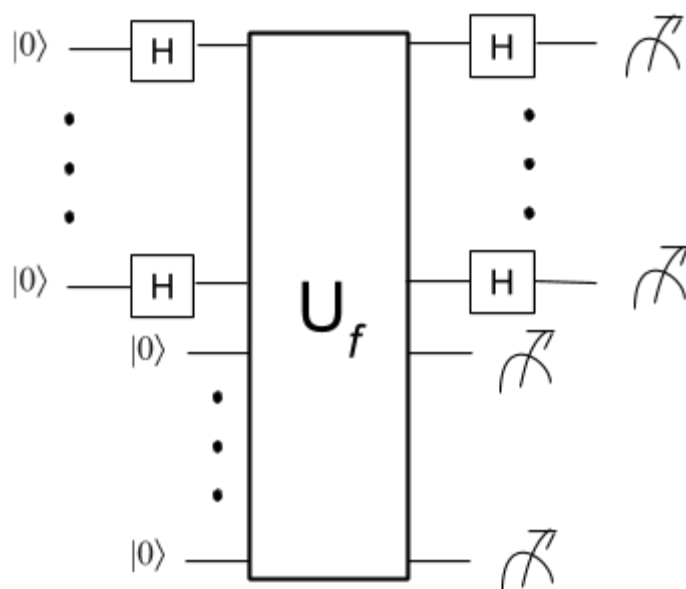
The algorithm follows a now-familiar pattern:

1. Start with n qubits all in the $|0\rangle$ state.
2. Hadamard them all.
3. Query f .

This yields the state

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

This time the function f has a large output, so we need to write out its values in a separate n -qubit



answer register, rather than just encoding it into the phase.

But even though we're giving more space to write out the answers $f(x)$, it's important to note that the answers themselves aren't what we care about!

Instead, we're only writing them out because by doing so, we create a desired interference pattern in the x register. Indeed at this point in the algorithm we could simply discard the $f(x)$ qubits, or do anything else we liked with them.

So for pedagogical simplicity, let's assume we now *measure* the $f(x)$ qubits. And let's assume that the result of the measurement is $|w\rangle$.

Keeping track of all of the w 's we could have seen would've resulted in a mixed state.

Instead, we're just conditioning on a particular w .

Now how many different values are superposed in the input register?

By the partial measurement rule, we're left with an equal superposition over all the different x 's that are consistent with the $f(x)$ value that we observed, namely w . In Simon's problem, there are necessarily two such x 's. In other words, we're left with a superposition $\frac{|x\rangle+|y\rangle}{\sqrt{2}}$ such that $f(x) = f(y) = w$.

By the Simon promise, this means in particular that $y = x \oplus s$.

So what is this state good for?

First, observe that if we could just measure $\frac{|x\rangle+|y\rangle}{\sqrt{2}}$ twice, then with high probability we'd first get x and then y —so then bitwise-XORing the two strings would give us the secret string s , and we'd be done! Alas, in quantum mechanics we only get one chance to measure a state, so we'll see either $|x\rangle$ or

$|y\rangle$, which tells us nothing about s . We could of course repeat the whole algorithm from the beginning. But if we did, then with overwhelming probability we'd get a different w , corresponding to a new pair.

So we'll need to be more clever—although not *that* much more! In particular, let's see what happens if we measure the qubits of $\frac{|x\rangle+|y\rangle}{\sqrt{2}}$ in the Hadamard basis.

What's the effect of Hadamarding all the qubits on $\frac{|x\rangle+|y\rangle}{\sqrt{2}}$?

Well for starters, we saw in the last lecture that Hadamarding all qubits maps the basis state $|x\rangle$ to

$$\frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{x \cdot z} |z\rangle$$

and maps $|y\rangle$ to

$$\frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{y \cdot z} |z\rangle$$

By linearity, this means that doing the same thing to an equal superposition of $|x\rangle$ and $|y\rangle$ must give

$$H^{\otimes n} \frac{|x\rangle + |y\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2^{n+1}}} \sum_{z \in \{0,1\}^n} [(-1)^{x \cdot z} + (-1)^{y \cdot z}] |z\rangle$$

Now let's measure the above state in the standard basis.

Which z 's could we get when we do so?

For a given z to be observed, it must have a nonzero amplitude. This means that $(-1)^{x \cdot z}$ and $(-1)^{y \cdot z}$ must be equal, which occurs if and only if

$$x \cdot z = y \cdot z \pmod{2}.$$

Or rewriting this equation a bit:

$$x \cdot z + y \cdot z = 0 \pmod{2}$$

$$\Rightarrow (x \oplus y) \cdot z = 0 \pmod{2}$$

$$\Rightarrow s \cdot z = 0 \pmod{2}$$

Note that every z satisfying the above equation will appear with the same probability as every other, namely $1/2^{n-1}$. So, what we get when we measure is an n -bit string z , chosen uniformly at random among all the 2^{n-1} strings whose inner product with s is 0. In other words: we haven't yet learned s itself, but we've learned a bit of information about s , which I hope you'll grant is something!

But what if we really want s itself? In that case, we can just repeat Simon's algorithm over and over, starting from the beginning! This will give us a collection of strings , which are uniformly random and independent of each other, subject to satisfying the equations

$$s \cdot z_1 = 0 \pmod{2}$$

$$s \cdot z_2 = 0 \pmod{2}$$

⋮

$$s \cdot z_k = 0 \pmod{2}$$

After we've repeated enough times, what could we tell a classical computer to do with this information?

Well, suppose $k = n + c$, where c is some constant. Then we now have a collection of linear equations in n unknowns, over a finite field with two elements. We can solve this system efficiently using a classical computer.

Through an algorithm called "Run Matlab."

Or Gaussian elimination, taking $O(n^3)$ time.

Or if you want to get all theoretical about it, the fastest known algorithm—whose constant-factor overheads make it useless in practice—takes $O(n^{2.373})$ time.

It's not hard to do a probabilistic analysis showing that, after we've seen slightly more than n equations, with overwhelming probability we'll be left with a system that has exactly two solutions: namely, 0^n and s itself. We can throw away 0^n , because we assumed $s \neq 0^n$. So that leaves us with s .

That's Simon's algorithm, which solved Simon's problem using only $O(n)$ queries to f plus a polynomial amount of side computation, as compared to the $\Omega(2^{n/2})$ queries that are provably needed classically.

Does Simon's algorithm have a deterministic counterpart?

Yes, one can modify the algorithm so that it succeeds with certainty rather than “merely” overwhelming probability. We won't go into the details.

So why doesn't this just prove that quantum algorithms are better?

It's sort of tricky to translate Simon's algorithm into “real-world” consequences. To get a speedup over classical computing in terms of the sheer number of gates, or computational steps, we'd need some small circuit to compute a function f that was actually like our magical Simon function f has (i.e., that satisfied the same promise). For example, $f(x) = Ax$ for some rank- $(n-1)$ Boolean matrix A would do the trick.

But the difficulty in claiming that we're getting a quantum speedup this way is that, once we pin down the details of how we're computing f —so for example, the actual matrix A —we then need to compare against classical algorithms that know those details as well. And as soon as we reveal the innards of the black box, the odds of an efficient classical solution become much higher! So for example, if we knew the matrix A , then we could solve Simon's problem in classical polynomial time just by calculating A 's nullspace. More generally, it's not clear whether anyone to this day has found a straightforward “application” of Simon's algorithm, in the sense of a class of efficiently computable functions f that satisfy the Simon promise, *and* for which any classical algorithm plausibly needs exponential time to solve Simon's problem, even if the algorithm is given the code for f .

So the story goes that Simon wrote a paper about this theoretical black-box problem with an exponential quantum speedup, and the paper got rejected. But there was one guy who was like, “Hey, this is interesting.” He figured that if you changed a few aspects of what Simon was doing, you could get a quantum algorithm to find the periods of periodic functions, which would in turn let you do all sorts of fun stuff.

That guy was Peter Shor.

Lecture 19, Thurs March 30: RSA and Shor's Algorithm

Today we'll see Shor's algorithm. Given a positive integer N , which we'll assume for simplicity is a product of two primes p and q , this algorithm lets you find p and q using only about $O(\log^2 N)$ steps. This application captured the world's attention because **RSA**, one of the most widely-used public-key encryption methods, relies on the assumption that factoring is hard.

So before we start in on Shor's algorithm, which will take a few lectures, let's briefly review RSA. The basic idea is:

Some website, like Amazon, wants you to be able to send them messages that only they can decrypt (say your credit card number). But they've never met with you in private to agree on a secret encryption key. So what they do instead is that they find two large primes p and q (say a thousand digits each), which they keep secret. They Amazon multiplies them to get $N = pq$, which they publish to the world. Note that the fact that Amazon *can* efficiently pick two huge random prime numbers p and q , and know for sure that they're prime, is already not quite obvious, but it follows from some classical number theory that we won't go into.

Now you can encrypt a message using the public key, N : if your plaintext message is x , then in the simplest version of RSA, your encrypted message would just be $x^3 \bmod N$. And given that encrypted message, and *also* knowing p and q , there's a way for Amazon to efficiently recover x —it's again some basic number theory that we won't go into right now, although we'll see aspects of it later. The key idea is that, if you know p and q , then you also know the order of the *multiplicative group* modulo N , and that knowledge lets you do things like efficiently take cube roots modulo N .

By contrast, an eavesdropper who doesn't know p and q , but only knows N , *seems* to face an exponentially hard problem in recovering the plaintext—though it's been proven neither that factoring is hard, nor even that breaking RSA is necessarily as hard as factoring.

Some number theorists conjecture that factoring is in P, or profess agnosticism.

The problem has only been seriously worked on for, like, 40 years.

In any case, if you can factor, then you can break RSA, and that certainly provides more than enough reason to be interested in the complexity of factoring.

The naive algorithm to factor N is trial division—in the worst case, requiring you to test all possible divisors up to \sqrt{N} (note that every divisor greater than \sqrt{N} must have an accompanying divisor that's smaller). We call this an “exponential-time” algorithm, since the running time is exponential in $n = \log(N)$, which is the number of digits needed to specify N .

Number theorists have discovered several faster algorithms.

The Quadratic Field Sieve, from 1981, takes time roughly $2^{O(\sqrt{n})}$, a milder exponential.

The Number Field Sieve brings that down to $2^{n^{1/3}}$, though its correctness depends on the proof of a yet-unproven conjecture.

This is why 512-bit, 768-bit, and maybe even 1024-bit encryption aren't quite secure anymore.

They can be cracked using known algorithms and sufficient money for hardware.

In the Snowden documents there's evidence of money allocated for this sort of thing. In a way that's almost reassuring, to people who worry that the NSA can break anything...

Another public-key cryptosystem in widespread use is **Diffie-Hellman**, which is based on a different problem than factoring, called *discrete logarithms*. While we won't cover this in lecture, it turns out that Shor's algorithm *also* solves the discrete log problem in polynomial time, thereby breaking Diffie-Hellman as well.

No one has shown that factoring and discrete log are necessarily related, e.g. by giving a reduction between them. In practice, though, advances in solving one problem almost always seem to lead in short order to advances in solving the other.

We now know that both RSA and Diffie-Hellman were first discovered in secret, by a mathematician named Clifford Cocks at the GCHQ (the British NSA), before they were rediscovered in public.

What Shor's algorithm really does, under the hood, is something called *period-finding*. Shor observed that, for classical number theory reasons having nothing to do with quantum mechanics, a fast algorithm for period-finding leads to a fast algorithm for problems like factoring and discrete logarithm.

Period Finding

is yet another black box problem—one that should remind you of Simon's problem from the last lecture.

You're given oracle access to a function $f : \mathbb{N} \rightarrow \mathbb{N}$

You're also promised that there's a secret integer $s > 0$ (the “period”) such that

$$f(x) = f(y) \Leftrightarrow s \mid y - x$$

for all x, y .

The problem is to find s .

f has *period* s , which is to say: it returns to the same value after every s steps.

How many queries to f do we need to solve period-finding classically?

Let's assume that $s \sim 2^n$ (i.e., that s is an n -bit integer), and give our answer in terms of n . Observe that once you find a pair x, y such that $f(x) = f(y)$, you're very close to solving the problem. Indeed, if you found a few such collisions, you could just take their greatest common divisor, like so:

$$\gcd(x_1 - y_1, x_2 - y_2, x_3 - y_3)$$

We won't give the analysis, but after not too many collisions, the odds are high that this will yield the period.

Incidentally, how do we get the gcd of two integers in polynomial time?

We use **Euclid's GCD Algorithm**—possibly the oldest interesting polynomial-time algorithm in history!

To find the gcd of x and y (with $x > y$), we find q and r that satisfy:

$$x = qy + r$$

such that qy is the greatest multiple of y less than x , which means $y > r$.

Then we find the gcd of y and r , and we keep recursing in this way until $r = 0$.

Each time you run this, the size of the numbers goes down by about a constant factor, which means the whole algorithm runs in time linear in n (i.e., the bit-length of x and y).

OK, but how do we find collisions? This is the birthday paradox all over again. Recall from the last lecture that something like $\sqrt{2^n}$ queries to f is both necessary and sufficient.

This is *still* a huge number of queries, if (say) $n = 2000$.

Now we're ready to talk about...

Shor's Algorithm

which has two very different ingredients:

- A reduction from factoring to period-finding.

This part is purely classical.

- A quantum algorithm for period-finding that uses only a *constant* ($O(1)$) number of queries to f , as well as a polynomial ($n^{O(1)}$) number of computational steps.

Why is factoring \leq period finding?

Here \leq means “reducible to”

The main connection between the two is the multiplicative group modulo n . This is a finite abelian group—that is, a finite set with a commutative, associative, invertible multiplication operation—that's one of the most basic examples of a group in all of math.

For a prime p , the multiplicative group consists of the set $\{1, 2, \dots, p-1\}$, with the group operation being multiplication mod p .

To be in the multiplicative group mod N where N is composite, you again need to be less than N , and now also relatively prime with N , since otherwise you won't have a multiplicative inverse mod N .

For example, when $n = 15$, the multiplicative group mod N consists of the 8-element set

$$\{1, 2, 4, 7, 8, 11, 13, 14\},$$

with the group operation being multiplication mod 15.

To check that this is a group, you can fill out the multiplication table and see for yourself!

In general, the size of this group, given $N = pq$, is going to be $|\mathbb{Z}_N^\times| = (p-1)(q-1)$, since that's how many positive integers less than N are relatively prime to N . For example, when $N = 15 = 5 \times 3$, we saw that the size of the group is $(5-1)(3-1) = 8$.

An extremely useful fact about finite groups G is that, for any element $x \in G$, we have $x^{|G|} = 1$.

This has a few corollaries, such as

Fermat's Little Theorem

$$x^{p-1} \equiv 1 \pmod{p}, \text{ for all primes } p \text{ and integers } x \in \{1, \dots, p-1\}$$

and its generalization, **Euler's Theorem**

$$x^{(p-1)(q-1)} \equiv 1 \pmod{pq}, \text{ for all primes } p, q \text{ and integers } x \text{ relatively prime to them}$$

Euler's Theorem is super important in RSA encryption as well.

More generally, **Euler's Totient Function** $\varphi(N)$ returns the order of the multiplicative group mod N , which is the number of integers from 1 to N that are relatively prime to N . For example, if p and q are prime, then $\varphi(p) = p - 1$ and $\varphi(pq) = (p - 1)(q - 1)$. And we can write:

$$x^{\varphi(N)} \equiv 1 \pmod{N}$$

Why is this important?

Well, let's say we want to factor some number $N = pq$, a product of distinct primes.

Then here's an approach: pick an x such that $\gcd(x, N) = 1$

We can assume such x 's are easy to find. Indeed, if $\gcd(x, N) > 1$, then we can run Euclid's algorithm on x and N to factor N right then and there.

Shor's algorithm is based on a careful study of the **modular exponentiation function**:

$$f(r) := x^r \pmod{N}$$

What can we say about this function? First of all, how hard is it to compute? A naïve approach would use $r-1$ multiplications, which is exponential in $n = \log(N)$. But there's a much, much faster approach, called **Repeated Squaring**. It's best illustrated with an example.

Say we want to calculate $13^{21} \pmod{15}$.

We could calculate $13 \times 13 \times \cdots \times 13 \pmod{15}$, by alternating multiplication with reducing mod 15, but that's still 20 multiplications.

Instead, let's rewrite it as a product of 13 raised to various powers of 2:

$$13^{16} \times 13^4 \times 13 \pmod{15}.$$

Then we can further rewrite that as $\left(\left((13^2)^2\right)^2\right)^2 \times (13^2)^2 \times 13 \pmod{15}$.

This may be ugly, but it requires considerably fewer multiplications, an advantage that grows rapidly as the exponent increases. Indeed, the total time needed is polynomial in $\log(n)$.

Once again, repeated squaring also plays a central role in RSA decryption. Ironically, many of the same number theory facts that led to RSA also allow lead to Shor's algorithm, which breaks RSA.

OK, so $f(r) = x^r \pmod{N}$ is efficiently computable. It's also, clearly, a periodic function. What could we learn by figuring out its period? Here's the key point: we claim that **finding the period of f will help us factor N** .

Why?

First an intuition. Suppose we were able to learn $\varphi(N)$, the order of the multiplicative group mod N .

Then we'd surely be able to factor N into pq . Indeed,

$$\varphi(N) = (p - 1)(q - 1) = N - p - q + 1.$$

So now we know both pq and $p+q$, and we can use the quadratic formula to solve for p and q themselves.

Unfortunately, this doesn't quite work with Shor's algorithm, because the period of f might not be the same as $\varphi(N)$: the most we can say is that the period *divides* $\varphi(N)$.

So here's what we'll do instead. We pick a random x relatively prime to N , and find the period s of $f(r) = x^r \pmod{N}$. We then have that $x^s \equiv 1 \pmod{N}$.

Now let's imagine we're lucky, and s is even (which intuitively should happen maybe half the time?). In that case we can write:

$$\begin{aligned} x^s - 1 &\equiv 0 \pmod{N} \\ \Rightarrow (x^{s/2} - 1)(x^{s/2} + 1) &\equiv 0 \pmod{N} \end{aligned}$$

Now let's imagine we get lucky a second time, and neither $x^{s/2} - 1$ nor $x^{s/2} + 1$ are multiples of N .

Then that means we've learned a factor of N .

For we just compute $\gcd(x^{s/2} - 1, N)$, which will give us either or .

Because if neither $x^{s/2} - 1$ nor $x^{s/2} + 1$ contains a full N , then one must contain a multiple of and the other a multiple of .

Furthermore, both of these “imagine we get lucky” steps can be shown to happen with a constant probability over the choice of x , by using a little number theory that we won't go into here. The precise statement is as follows:

For any N , if x is randomly chosen relatively prime to N , then with probability at least $\frac{3}{8}$:

- s is even
- Neither $x^{s/2} - 1$ nor $x^{s/2} + 1$ is a multiple of N

This gives us a plan of attack for...

Shor's Algorithm (The Quantum Part)

Basically, we'll give a quantum algorithm that solves the black-box problem of period-finding, in time polynomial in $\log(N)$. We'll then apply that algorithm to find the period of the function

$$f(r) = x^r \pmod{N},$$

for a randomly chosen base x . (I.e., we'll use the above f to “instantiate” the black box.) By the previous discussion, this f can be computed in polynomial time—and better yet, the ability to find its period implies the ability to factor N .

When we write f , what we really mean is f_x for a specific base x that we choose.

As we saw before, to factor a given N , we might need to try several different values of x until we find one that works.

The first step is to make an equal superposition over all positive integers r less than some upper bound Q ,

with each integer written in binary notation: $\frac{1}{\sqrt{Q}} \sum_{r=0}^{Q-1} |r\rangle |f(r)\rangle$

For technical reasons, we set Q to be the smallest power of 2 larger than N^2 .

We can prepare this state by Hadamarding the qubits in the first register, then querying f .

Unlike with Simon's algorithm and so forth, in Shor's algorithm we don't need to treat U_f as an abstract black box. We can find an actual quantum circuit to implement U_f , because f is not arbitrary:

$$f(r) = x^r \pmod{N}$$

By using the repeated squaring trick, we can create actual circuit U_f that maps $|r\rangle|0 \dots 0\rangle$ to $|r\rangle|f(r)\rangle$, out of a network of $\text{polylog}(N)$ Toffoli gates.

We'll use ancilla qubits to store the output $|f(r)\rangle$. Then, just like with Simon's algorithm, we won't care at all about the actual value of $f(r)$ —only about the effect that computing $f(r)$ had on the $|r\rangle$ register. So for pedagogical purposes, we'll immediately measure the $|f(r)\rangle$ register and then discard the result.

What's left in the $|r\rangle$ register?

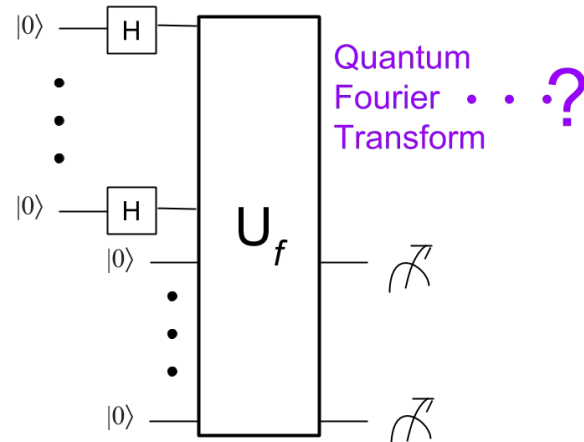
By the partial measurement rule, what's left is an equal superposition over all the possible r 's that could've led to the observed value $f(r)$. Since f is periodic with a secret period s , these values will differ from each other by multiples of s . In other words, we now have:

$$\frac{1}{\sqrt{\text{The quantity of these}}} [|r\rangle + |r + s\rangle + |r + 2s\rangle + \dots]$$

The central challenge of Shor's algorithm is to measure the above state in a way that reveals useful information about the period s . Just like with Simon's algorithm, *if* we could measure the state multiple times, then we could compare the results to each other and take some gcd's to find s . But we can't do that! We can repeat the whole algorithm from the beginning, but if we do, then we'll almost certainly end up with a different offset r , preventing useful comparisons.

This simply means that, just like with everything else in quantum computing, to see a speedup at some point we'll have to exploit the magic of minus signs: interference, cancellations, change of basis, whatever term you want to use. But how do we change the basis to one that reveals the period s —and moreover, do so efficiently, using a number of gates that's only polynomial in $\log(N)$?

That brings us to the topic of the next lecture: the **Quantum Fourier Transform!**



Lecture 20, Tues April 4: Shor, Quantum Fourier Transform

Last time we started in on Shor's algorithm, a quantum algorithm that can factor N into $p \times q$ in polynomial time by reducing the problem to period-finding. Today we'll start to see how to solve period-finding in polynomial time. Before we do, though, a conceptual clarification:

Is Shor's algorithm *provably* faster than any classical algorithm for the same task?

If by "Shor's algorithm," we mean the period-finding core of the algorithm, then the answer is yes. As we saw in the last lecture, there's a provable speedup from $\sim \sqrt{s}$ queries to only $O(1)$.

If, on the other hand, we think of Shor's algorithm as a way to do integer factorization, then the speedup remains conjectural. Indeed it has to, because no one has even proven that $P \neq NP$ ---and if $P=NP$, then of course factoring is classically easy.

The way to reconcile these two statements is simply to observe that there are many ways to factor a number besides by reducing factoring to period-finding. In fact, the best known classical factoring algorithms---the Quadratic Field Sieve and Number Field Sieve mentioned in the last lecture---do much better than \sqrt{s} by exploiting additional structure in the factoring problem. The most we can currently prove is that Shor's algorithm achieves an exponential speedup over any classical factoring algorithm *that works via the last lecture's reduction to black-box period-finding*.

We left off last time with our quantum state in the form

$$|r\rangle + |r+s\rangle + |r+2s\rangle + \dots$$

Now we'll see how to measure this state to extract useful information about the period s .

In science and engineering, any time you have a periodic signal and you're trying to extract its period, there's a single tool that gets called upon...

The Fourier Transform!

There are many types of Fourier transforms: continuous, Boolean, etc. For us, though, the Q -dimensional Fourier transform will be the $Q \times Q$ matrix F_Q defined as follows:

$$\langle i | F_Q | j \rangle = \omega^{ij} / \sqrt{Q}, \quad \text{where } \omega = e^{2\pi i / Q} \text{ is a } Q^{\text{th}} \text{ root of unity.}$$

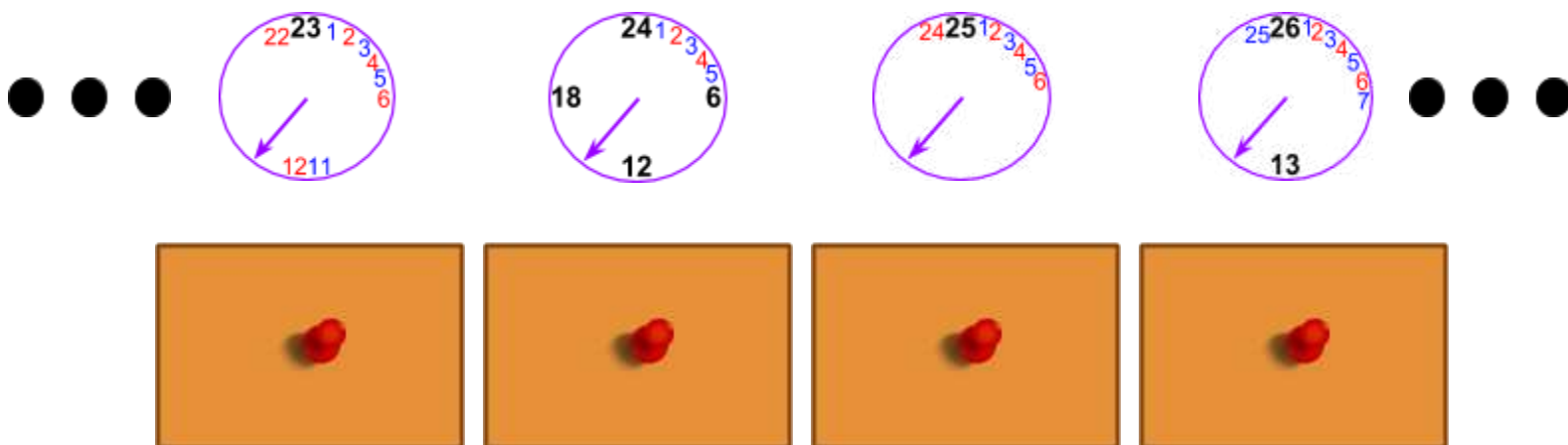
So, the ij^{th} entry of the matrix involves ω raised to the ij^{th} power.

Here's some useful intuition for how it works:

In grad school, you can easily fall into a 26-hour-per-day cycle. So one day you wake up at 8am, the next day you wake up at 10am, then 12pm, and so forth.

Suppose you've fallen into such a cycle, and you want to figure out how long the cycle is, without doing any complicated calculations like subtraction.

What you can do is install a series of clocks in your room, each tracking “days” of different lengths. So you’d have a 23-hour clock, a 24-hour clock, a 25-hour clock, etc. In addition, install a bulletin board below each clock and place a single thumbtack in the center.



Now: every time you wake up, for each clock, move the thumbtack one inch in the direction that the hour hand points.

What will happen if you keep doing this, week after week?



The thumbtack corresponding to the 26-hour clock will always move in the same direction.

This is constructive interference!

And the same can be said for the 13-hour clock (as well as the 2- and 1-hour clocks).

All the others, the 23-hour clock, the 24-hour clock, etc, will have the thumbtack move around, but sometimes one way, sometimes another way, so that it eventually returns to the origin.

The Quantum Fourier Transform is essentially this, but with quantum-mechanical amplitudes instead of thumbtacks.

There are two questions we need to answer here:

1. How do we implement the Quantum Fourier Transform using a small quantum circuit?

Since it's a $Q \times Q$ matrix, it's not obvious whether we can do it using a circuit with $\text{polylog}(Q)$ gates.

2. Once we've applied the QFT and measured, how do we make sense of the outcome?

Complications can arise because the period s doesn't divide Q (in all likelihood).

To answer the first question, let's look at some examples.

$$F_2 = H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \text{because it's } \begin{pmatrix} \omega^{0*0} & \omega^{0*1} \\ \omega^{1*0} & \omega^{1*1} \end{pmatrix}$$

$$\begin{pmatrix} 1 & -1 \end{pmatrix} \qquad \begin{pmatrix} \omega^{1*0} & \omega^{1*1} \end{pmatrix}$$

$$F_4 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \qquad \text{i.e.} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 \\ 1 & \omega^2 & 1 & \omega^2 \\ 1 & \omega^3 & \omega^2 & \omega^1 \end{pmatrix}$$

$$\text{For } F_8 \text{ you'd have } \frac{1}{2\sqrt{2}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega^1 \end{pmatrix}$$

keep in mind that you can simplify

$$\omega^2 = i$$

$$\omega^4 = -1$$

We could design an algorithm to apply these matrices by brute force, but there's a better way.

This method is related to one of the most widely used classical algorithms...

The FFT – Fast Fourier Transform

Suppose we have a vector of length Q , and we want to apply a $Q \times Q$ matrix A to it. In general this takes $\sim Q^2$ operations. However, *if* we know that A is the Fourier transform, then the FFT lets us apply it in only $O(Q \log Q)$ steps, by exploiting regularities in the Fourier matrix.

So what regularity *is* there in the Fourier matrix?

$$\text{Look at } F_4 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

If we swap the second and third columns, we get

$$\frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & i & -i \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -i & i \end{pmatrix}$$

This quadrant looks like H... $\begin{pmatrix} 1 & -1 & i & -i \end{pmatrix}$

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

This one too... $\begin{pmatrix} 1 & -1 & -i & i \end{pmatrix}$

In fact, we can rewrite the whole matrix as

$$\begin{pmatrix} \textcolor{red}{H} & \textcolor{blue}{AH} \end{pmatrix}$$

$$\begin{pmatrix} \textcolor{violet}{H} & \textcolor{brown}{-AH} \end{pmatrix}$$

for $A = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$ [the phase shift]

You can do the same procedure for F_8

$$\frac{1}{2\sqrt{2}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega^1 \end{pmatrix}$$

Moving all the odd columns to the left, and the even columns to the right, and simplifying the ω 's, we get

$$\frac{1}{2\sqrt{2}} \begin{pmatrix} \textcolor{red}{1} & \textcolor{red}{1} & \textcolor{red}{1} & \textcolor{red}{1} & \textcolor{red}{1} & \textcolor{blue}{1} & \textcolor{blue}{1} & \textcolor{blue}{1} & \textcolor{blue}{1} \\ \textcolor{red}{1} & \textcolor{red}{i} & \textcolor{red}{-1} & \textcolor{red}{-i} & \textcolor{red}{\omega^1} & \textcolor{blue}{\omega^3} & \textcolor{blue}{\omega^5} & \textcolor{blue}{\omega^7} \\ \textcolor{red}{1} & \textcolor{red}{-1} & \textcolor{red}{1} & \textcolor{red}{-1} & \textcolor{red}{i} & \textcolor{blue}{-i} & \textcolor{blue}{i} & \textcolor{blue}{-i} \\ \textcolor{red}{1} & \textcolor{red}{-i} & \textcolor{red}{-1} & \textcolor{red}{i} & \textcolor{red}{\omega^3} & \textcolor{blue}{\omega} & \textcolor{blue}{\omega^7} & \textcolor{blue}{\omega^5} \\ \textcolor{red}{1} & \textcolor{red}{1} & \textcolor{red}{1} & \textcolor{red}{1} & \textcolor{red}{-1} & \textcolor{blue}{-1} & \textcolor{blue}{-1} & \textcolor{blue}{-1} \\ \textcolor{red}{1} & \textcolor{red}{i} & \textcolor{red}{-1} & \textcolor{red}{-i} & \textcolor{red}{\omega^5} & \textcolor{blue}{\omega^7} & \textcolor{blue}{\omega} & \textcolor{blue}{\omega^3} \\ \textcolor{red}{1} & \textcolor{red}{-1} & \textcolor{red}{1} & \textcolor{red}{-1} & \textcolor{red}{-i} & \textcolor{blue}{i} & \textcolor{blue}{-i} & \textcolor{blue}{i} \\ \textcolor{red}{1} & \textcolor{red}{-i} & \textcolor{red}{-1} & \textcolor{red}{i} & \textcolor{red}{\omega^7} & \textcolor{blue}{\omega^5} & \textcolor{blue}{\omega^3} & \textcolor{blue}{\omega^1} \end{pmatrix}$$

Which, again, we can now rewrite as

$$\begin{pmatrix} \textcolor{red}{F}_4 & \textcolor{blue}{AF}_4 \end{pmatrix}$$

$$\begin{pmatrix} \textcolor{violet}{F}_4 & \textcolor{brown}{-AF}_4 \end{pmatrix}$$

This time $A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \omega & 0 & 0 \\ 0 & 0 & \omega^2 & 0 \\ 0 & 0 & 0 & \omega^3 \end{pmatrix}$

You can work out why it happens on your own, but it turns out that we can define F_Q in terms of this nesting recurrence:

$$F_Q = \frac{1}{\sqrt{2}} \begin{pmatrix} \textcolor{red}{F}_{Q/2} & \textcolor{blue}{AF}_{Q/2} \\ \textcolor{violet}{F}_{Q/2} & \textcolor{brown}{-AF}_{Q/2} \end{pmatrix}$$

Notice that applying F_Q takes linear time plus twice however much time it takes to apply $F_{Q/2}$, so solving the recurrence, the overall running time of the FFT algorithm is $O(Q \log Q)$.

In the quantum case, we're actually interested in the unitary transformation

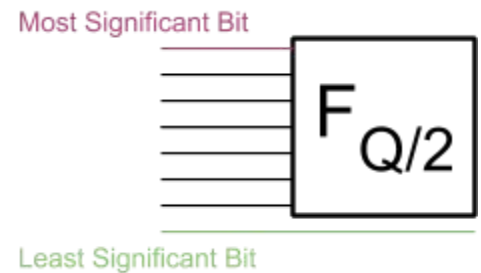
$$|\Psi\rangle \rightarrow F_Q |\Psi\rangle \quad \text{where } |\Psi\rangle \text{ is a quantum state of } \log_2 Q \text{ qubits.}$$

In one sense, our new goal is less ambitious: we don't need the result vector written explicitly in memory anywhere; it only needs to be encoded implicitly in a vector of amplitudes. In another sense, though, our new goal is more ambitious, since we now want to apply a Fourier transform in time polynomial in only $\log Q$.

So how do we do it?

We can use the same recursion that we used for the FFT, *plus* the additional observation that that recursion behaves "linearly" with respect to quantum states.

Let's think of our $\log(Q)$ qubits as representing an integer from 0 to $Q-1$ in binary notation. And let's order the bits of that integer from most to least significant. Then we can try applying the circuit $F_{Q/2}$ to the "first half" of the number: in other words, to all but the least significant bit.



This corresponds to applying the matrix

$$\begin{pmatrix} F_{Q/2} & \\ & \end{pmatrix}$$

To get an A on the bottom right quadrant, as in the FFT algorithm, we can then apply a control- A :

$$\begin{pmatrix} F_{Q/2} & \\ & AF_{Q/2} \end{pmatrix}$$

We can implement A using a linear number of gates, because it simply amounts to the following:

If the most significant bit is 1, then do a rotation
 Else if the second bit is 1, then do a rotation that's half as big
 Else if the third bit is 1, ... etc...

By the time you reach the last couple of bits, the rotation is exponentially small.

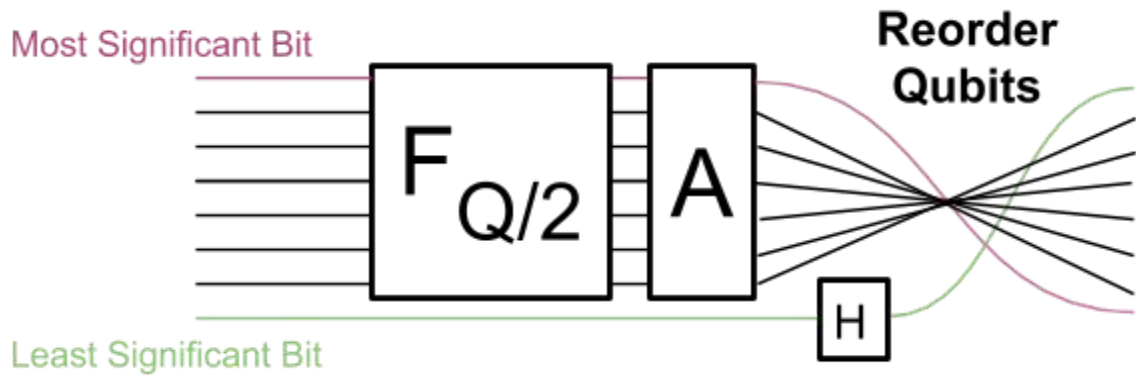
When they first learn about Shor's algorithm, some people object that it's "unphysical," since there's no practical way to apply such tiny rotations. But it turns out that the exponentially small rotations *don't matter* for the algorithm---indeed, there's a theorem that says that you can just *omit* these tiny rotations.

Doing so even improves the size of the quantum circuit that implements F_Q , from $O(q^2)$ to $O(q \log q)$.

The final step to get the matrix we want is a Hadamard gate.

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} F_{Q/2} & \\ & AF_{Q/2} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} F_{Q/2} & AF_{Q/2} \\ F_{Q/2} & -AF_{Q/2} \end{pmatrix}$$

So here's our finished quantum circuit:



Note that the reordering of the qubits isn't part of the recursion: it's performed only once, at the very end of the circuit. We'll leave it as an exercise to see why the qubits emerge from the circuit in *reverse* order, with the least significant at the top and the most significant at the bottom. (Hint: it has to do with the least significant, Hadamarded qubit "jumping up" to become the most significant one, at each level of the recursion.)

Okay. So now that we've seen how to implement the Quantum Fourier Transform as a quantum circuit, it's time to answer our second question:

What comes out when we measure, and how can we use it to learn s ?

We have a state like $\frac{1}{\sqrt{L}} (|r\rangle + |r+s\rangle + |r+2s\rangle + \dots + |r+(L-1)s\rangle)$

And we know that the QFT maps this state to

$$\frac{1}{\sqrt{QL}} \sum_{j=0}^{Q-1} \sum_{l=0}^{L-1} \omega^{(r+ls)j} |j\rangle.$$

What's going on with the above state? Let's start with an easy special case, and only later handle the general case.

The easy case is that s divides Q .

Assuming that, let's answer the question: which j 's can be observed, when we measure the above state in the computational basis? This in turn boils down to: for a given j , do the various contributions to j 's amplitude interfere constructively or destructively?

To answer this question, we can ignore the global phase ω^r and just look at the sum $\sum_{l=0}^{L-1} \omega^{jls}$.

The key is to identify whether js is a multiple of Q .

- If js is **not** a multiple of Q

Then we have destructive interference because the terms ω^{js} , $(\omega^{js})^2$, $(\omega^{js})^3$, etc. are all pointing in different directions in the complex plane, and they cancel each other out.

Just like the thumbtack's movement coming back to the origin.

- If js is a multiple of Q --or equivalently, if $js = kQ$ and $j = kQ/s$ for some integer k .

Then, since ω was a Q^{th} root of unity, the terms ω^{js} , $(\omega^{js})^2$, $(\omega^{js})^3$, ... all point in the same direction in the complex plane, producing constructive interference.

If we repeat this procedure several times, we'll generate a list of such j 's, each of which is an integer multiple of Q/s :

$$j_1 = k_1 Q/s, \quad j_2 = k_2 Q/s, \dots$$

So then we just need to take their GCD to get Q/s itself (with overwhelming probability), from which we can compute s , given our knowledge of Q .

Remember: This is only possible because we assumed that s divides Q .

The harder, general case is that s doesn't divide Q .

In this case, if we calculate the final amplitude for a specific basis state j , then ignoring the global phase

ω^r , we'll still get a sum of the form $\sum_{l=0}^{L-1} \omega^{j sl}$. So, how likely we are to observe j will still depend on

whether this sum involves constructive or destructive interference.

What changes is that now Q/s isn't an integer--and as a result, neither the constructive nor the destructive interference will be perfect. But we'll see that they're still good enough for the period s to be efficiently recovered.

Let's ask whether j has the form $\lfloor k \frac{Q}{s} \rfloor$

for some integer k : in other words, whether it's the nearest integer to some multiple of $\frac{Q}{s}$.

- If $j = \lfloor k \frac{Q}{s} \rfloor$

then we'll claim that we'll see *mostly* constructive interference.

- If $j \neq \lfloor k \frac{Q}{s} \rfloor$

then we'll claim that we'll see *mostly* destructive interference.

Let's look at the constructive case first.

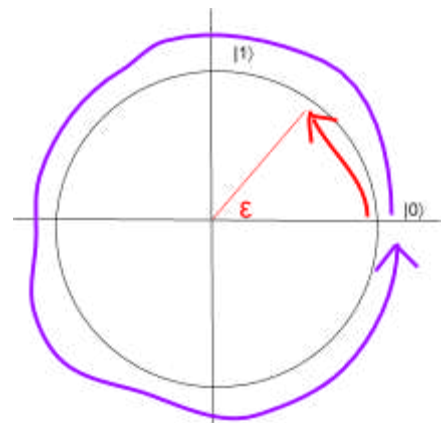
Assume we get a bit lucky, and we have (say) $j = k \frac{Q}{s} + \epsilon$ where $|\epsilon| \sim \frac{1}{10}$.

That means that ignoring normalization, the final amplitude of basis state j has the form

$$\sum_{l=0}^{L-1} \omega^{(k \frac{Q}{s} + \epsilon)sl} = \sum_{l=0}^{L-1} \omega^{kQl} \omega^{\epsilon sl}.$$

We can go further, and say that the ω^{kQl} term doesn't matter, because

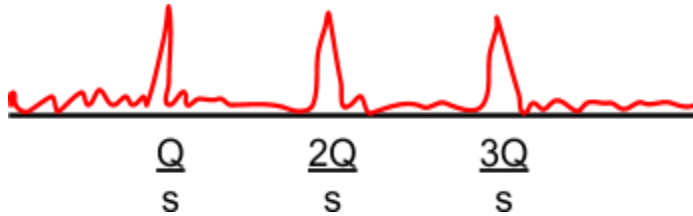
$$\omega^{kQl} = e^{(2\pi i/Q)(kQl)} = e^{2\pi ikl} = 1.$$



So then we're left with the $\omega^{\varepsilon l}$ part, which amounts to a rotation around an ε fraction of the unit circle.

i.e. mostly constructive interference.

Next suppose j isn't the nearest integer to a multiple of Q/s . In that case, as we vary l , the term ω^{jl} will loop all the way around the unit circle one or more times. So we'll get mostly destructive interference, except for a small amount of constructive interference from the final rotation.



If you plot the final amplitude as a function of j , you get something like the graph on the left.

Lecture 21, Thurs April 6: Continued Fractions,

Shor Wrap-Up

Today we'll finish Shor's algorithm and then discuss some of its implications.

Last we saw our protagonists, they were in a superposition of the form

$$|r\rangle + |r+s\rangle + |r+2s\rangle + \dots,$$

and we were trying to use the Quantum Fourier Transform (QFT) to extract the period s . Our first order of business was to give a polynomial-size quantum circuit to implement the QFT. Our second order of business was to understand what we observe after we apply the QFT and then measure in the computational basis.

Recall that there were two cases:

If s divides Q , then each possible measurement outcomes has either perfectly constructive or perfectly destructive interference. And the outcomes with constructive interference---i.e., the ones that we could see---are all integer multiples of Q/s . From a few random such outcomes, it's easy to recover s itself, given our knowledge of Q .

If s doesn't divide Q , then the pattern of constructive and destructive interference is no longer perfect, but has some noise to it.

That's because $k \frac{Q}{s}$ is no longer generally an integer, but the algorithm's output still needs to be an integer. So we effectively get a rounding effect, where the nearest integers to $k \frac{Q}{s}$ have the strongest constructive interference.

So we run the algorithm once and get an integer, let's say $l_1 = \lfloor k \frac{Q}{s} \rfloor$, then run it again to get l_2, l_3 , etc.

And the question is then: given these integers, almost all of which are close to integer multiples of Q/s , how do we use them to deduce s itself?

This brings us to the final step of Shor's algorithm, which is another piece of classical number theory called the...

The Continued Fraction Algorithm

Whenever an outcome l is observed, we'd like to determine whether it's close to an integer multiple of Q/s , and if so what the multiple is. This is where we use continued fractions.

It's easiest to illustrate with an example. Let's look at the continued fraction expansion of an approximation of π , 3.14.

$$3.14 = 3 + \frac{14}{100} = 3 + \frac{1}{\frac{100}{14}} = 3 + \frac{1}{7 + \frac{2}{14}}$$

The idea is that we keep pulling out the largest integer we can and rewriting, until we have an approximation of Q/s to within an accuracy of about $1/Q^2$.

The reason why the method works is that s is a relatively small integer, so $\frac{Q}{s}$ is not only rational but has a relatively small denominator.

In more detail, let's write $l = k \frac{Q}{s} \pm \epsilon$, where ϵ is some small value.

Then we divide the above equation through by Q , to get $\frac{l}{Q} = \frac{k}{s} \pm \frac{\epsilon}{Q}$.

And so... $\left| \frac{l}{Q} - \frac{k}{s} \right| \leq \frac{\epsilon}{Q}$.

We'll exploit the key inequality above, along with:

- the fact that we know l
- the fact that we know Q (because we picked it: it's at most $\sim N^2$)
- the fact that we know that s isn't too large

$s \leq N$ because the order of the multiplicative group is less than N , and the order of any element in the group is at most the number of elements.

So $\frac{l}{Q}$ isn't just close to *any* rational number $\frac{k}{s}$, it's close to a rational number with a pretty small denominator, and that doesn't happen by random chance.

There's math that backs this up.

This is the reason why we set Q to be $\sim N^2$ in the first place. Doing so ensures that the rational approximation $\frac{k}{s}$ to $\frac{l}{Q}$ is more-or-less unique, and moreover that there's an efficient algorithm to find it.

Suppose I give you a rational number, say 0.25001, and I tell you that it's close to a rational number with an unusually small denominator. How could you figure out which such rational number it's close to, without having to try *all possible* small denominators, of which there might still be too many?

In this particular example, you just stare at the thing, and immediately see $\frac{1}{4}$ is the answer! OK, but what would be a more systematic way of doing it?

The more systematic way is to expand the input number as a continued fraction, until the leftover part is so small that we can safely discard it. To illustrate:

$$\frac{25001}{100000} = \frac{1}{\frac{100000}{25001}} = \frac{1}{3 + \frac{24997}{25001}} = \frac{1}{3 + \frac{1}{\frac{25001}{24997}}} = \frac{1}{3 + \frac{1}{1 + \frac{4}{24997}}}$$

Now we've reached $\frac{4}{24997}$, a number small enough for us to discard, which leaves us with

$$\sim \frac{1}{3 + \frac{1}{1}} = \frac{1}{4}$$

So now we have a way to find $\frac{k}{s}$. Are we done?

Well, we still have the same difficulty that we encountered in the s divides Q case, namely that k and s might share a nontrivial divisor. If, for example, k and s were even, then we'd have no possible way to tell $\frac{k}{s}$ apart from $\frac{k/2}{s/2}$.

We solve this using exactly the same approach as before: we repeat the algorithm several times to generate $\frac{k_1}{s_1}, \frac{k_2}{s_2}, \frac{k_3}{s_3}, \dots$, etc.

One can then show that the least common multiple of the s_i 's will be s itself, with a sufficiently high probability.

Today, half of pop-science articles *still* say that “quantum computers would factor numbers by trying all the possible divisors in parallel.”
If you’ve taken anything away from our discussion of how Shor’s algorithm works, I hope you now agree that it’s more subtle than that!

In the next lecture, we’ll see how a quantum speedup for “pure, brute-force search” *does* exist, but it’s not exponential, but “merely” quadratic.

The ink wasn’t dry on Shor’s paper before people started asking...

What else might Shor’s algorithm be good for, besides factoring?

For starters, as we mentioned a couple lectures ago, and as Shor showed in his original paper, it also gives exponential speedup for **Discrete Log**:

Given a prime p , and an integer g such that $g^x = a \pmod{p}$. Find x .

This is how Shor’s algorithm breaks the Diffie-Hellman cryptosystem.

And it was noted shortly afterward that Shor’s algorithm can also be modified to break **Elliptic Curve Cryptosystems**. Indeed, people quickly figured out that Shor’s algorithm can be modified to solve pretty much *any* problem related to finding hidden structures in abelian groups.

Almost all the public-key cryptosystems that we currently use involve finding such hidden structures.

In the years after Shor’s algorithm, a lot of research in quantum algorithms was directed towards answering the question:

“To what extent can we generalize Shor’s algorithm to solve problems about *non*-abelian groups?”

By now, though, many people have given up on this direction. It’s very, very hard.

Why did people care about non-abelian groups? Well, if Shor’s algorithm could be generalized to handle them, there are two famous problems that would help us solve.

1. Graph Isomorphism

This is where you’re given two graphs, and need to decide whether they’re isomorphic. It’s a problem that no one yet knows how to solve in polynomial time, but that famously seems to have “too much structure” to be NP-complete.

In the early 1970s, when Leonid Levin co-discovered the theory of NP-completeness, legend has it that he sat on his discovery for more than a year because he was trying to show that Graph Isomorphism is NP-complete---something that we now believe is impossible.

People quickly realized that if you could generalize Simon's and Shor's algorithms to a situation where the underlying group is the symmetric group S_n , instead of an abelian group like \mathbb{Z}_2^n or \mathbb{Z}_N^\times , then it would solve Graph Isomorphism in quantum polynomial time.

In 2016, though, Babai (who's been studying Graph Isomorphism for forty years) found a classical algorithm to solve Graph Isomorphism in *quasipolynomial* time, meaning $n^{\text{polylog}(n)}$.

Many people suspect that Graph Isomorphism is in P, for one thing because the problem is easy in practice almost all of the time. In any case, since we now know that Graph Isomorphism is at worst quasipolynomial classically, there's no longer any possibility of getting an *exponential* quantum speedup for the problem.

2. Lattice-Based Cryptography

There's a set of public-key cryptosystems based on lattices, which are becoming increasingly important theoretically and even practically, and we don't know how to break (yet) even with a quantum computer.

Given a collection $\{z_1, \dots, z_n\}$ of vectors in \mathbb{R}^n , the lattice spanned by the collection is the set of all integer linear combinations of the vectors:

$$L = \{a_1 z_1 + \dots + a_n z_n : a_1, \dots, a_n \in \mathbb{Z}\}$$

A typical problem would be: "given $\{z_1, \dots, z_n\}$, find the shortest nonzero vector in L ---or at least, a vector that's within a \sqrt{n} factor of being the shortest."

It turns out that you can create entire public-key cryptosystems around these sorts of problems.

There was an important result by **Regev (2005)**, which says that we could break lattice-based cryptosystems if we could generalize Shor's algorithm to work for a nonabelian group called the dihedral group.

Needless to say---because otherwise I would've told you!--no one has yet succeeded in doing so.

So, lattice-based crypto is an attractive alternative to RSA and Diffie-Hellman for those who are paranoid about quantum computers. But it's also attractive for other reasons, including the prospect of **Fully Homomorphic Encryption**: the ability to do arbitrary computations on encrypted data without ever decrypting it. This would let people submit their data to cloud computing servers, and then get back the results, without the cloud server ever learning what computation it did. In 2009 Craig Gentry proposed the first Fully Homomorphic Encryption scheme using lattice-based crypto; since then other schemes have been proposed. These are not schemes that we know how to break even using a quantum computer.

There's still a practical problem with these schemes: the key sizes, message sizes, and computation times tend to be huge. But the schemes have been steadily improving, and many of them are now either practical or nearing practicality.

Lecture 22, Tues April 11: Grover

The next quantum algorithm we'll cover is...

Grover's Algorithm

which was discovered in 1995, shortly after Shor's algorithm.

Both Grover and Shor were working at Bell Labs at the time.

Grover's algorithm gives a smaller speedup than Shor's (quadratic rather than exponential), but for a much wider range of problems. Just like with all the other quantum algorithms we've seen, it's easiest to think of Grover's algorithm in terms of a black box:

Given an oracle function $f: \{1, \dots, N\} \rightarrow \{0, 1\}$.

We'd like to answer two questions:

- Is there an x such that $f(x) = 1$?
- If there is, what is such an x ?

The basic problem that Grover's Algorithm addresses is unordered search.

a.k.a looking through a list of bits for a 1 bit.

Classically, we'd need a linear number of queries, $\Omega(N)$, to solve this problem deterministically.

Why? Simply because, if we want to know for certain whether there's a treasure hidden in one of n boxes, then even after opening $N-1$ boxes and finding them empty, we still need to open the N^{th} box!

If the treasure is guaranteed to be in *some* box, then finding it takes $\sim \frac{N}{2}$ queries on average, which is still linear in N .

Grover's algorithm solves both problems using only $O(\sqrt{N})$ quantum queries to the function f .

This might seem unreasonable---but as we'll see, it's quite similar to how the Elitzur-Vaidman Bomb worked.

The number of qubits needed to run Grover's algorithm is very low, $O(\log N)$, and the number of gates required is also reasonable, $O(\sqrt{N} \log N)$.

However, for Grover's algorithm to work, we do need to assume that we have quantum access to the function f , in such a way that we can apply the unitary transformation $|x, a\rangle \rightarrow |x, a \oplus f(x)\rangle$. This wasn't important in Shor's Algorithm because we only made one query and then discarded the result.

There are two main example applications to keep in mind with Grover's algorithm.

The first application is solving combinatorial search and optimization problems, such as NP-complete problems. Here, we think of $N = 2^n$ as being exponentially large, and we think of each candidate solution $x \in \{0, 1\}^n$ as an n -bit string. We then set, for example,

$$f(x) = \varphi(x),$$

where φ is an instance of Satisfiability or some other NP-complete problem.

Then Grover's algorithm can solve the problem in $O(2^{n/2} \text{poly}(n))$ time: namely, $\sqrt{N} = 2^{n/2}$ queries to f , and $\text{poly}(n)$ time to implement each query (say, by checking whether a given x satisfies φ).

This is a speedup for **NP**-complete problems---but at most a quadratic one, and also only *conjectural*, because of course we can't even rule out the possibility of **P=NP**, which would annihilate this sort of speedup.

For an **NP**-complete problem like CircuitSAT, we can be pretty confident that the Grover speedup is real, because no one has found any classical algorithm that's even slightly better than brute force. On the other hand, for more "structured" **NP**-complete problems, we *do* know exponential-time algorithms that are faster than brute force: for example, 3SAT is solvable in about $O(1.3^n)$ time. So then the question becomes a subtle one, of whether Grover's algorithm can be *combined* with the best classical tricks that we know, to achieve a polynomial speedup even compared to a classical computer that uses the same tricks. For many **NP**-complete problems, the answer seems to be yes, but it need not be yes for all of them.

The second example application of Grover's algorithm to keep in mind---Grover's original application---is searching an actual physical database.

$f(0)$	$f(1)$	\dots	$f(n)$
--------	--------	---------	--------

Say you have a database of personnel records, and you want to find a person who matches various conditions (hair color, hometown, etc).

You can set $f(x) = 1$ if person x meets the criteria
0 otherwise

Then Grover's algorithm can search for an x such that $f(x) = 1$ in $O(\sqrt{N})$ steps.

Some people have questioned the practicality of using Grover's algorithm to search a physical database, because the database needs to support "superposed queries": that is, you need to be able to query many records in superposition and get back a superposition of answers.

A memory that would support these kinds of queries is called a "quantum RAM." Building one is a whole additional technological problem, beyond building a quantum computer itself. And it remains unclear whether people will be able to build quantum RAMs without n active, parallel computing elements---which, if you had them, would remove the need to run Grover's algorithm. In this lecture, though, we'll treat such things as "mere engineering difficulties"!

Anyway, one big advantage of Grover's algorithm as applied to actual physical databases is that there the quantum speedup is provable: it doesn't rely on any unproved computational hardness assumptions.

OK, so without further ado, how does Grover's algorithm work?

For simplicity, let's assume that a solution exists and is unique. We call this the marked item: it's the unique x^* such that $f(x^*) = 1$

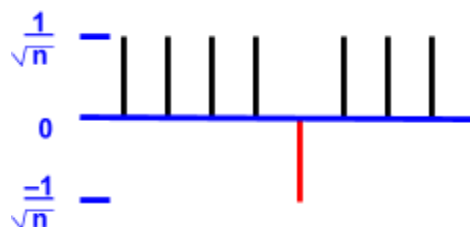
We'll also assume for simplicity that $N = 2^n$ is a power of 2, which will allow us to do our favorite trick: start by Hadamarding n qubits.



Doing so brings the initially all-0 state to a uniform superposition:

$$|00\dots 0\rangle \rightarrow (1/\sqrt{N}) \sum_{x=1}^N |x\rangle$$

As shown, all possible x values have the same amplitudes.



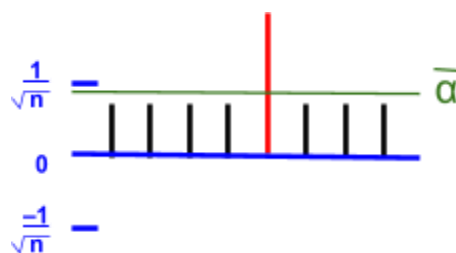
Then we query with U_f , a unitary transformation that flips the amplitude of the marked item: $U_f|x\rangle = (-1)^{f(x)}|x\rangle$.

As we've already seen in this course, if we can apply $|x, a\rangle \rightarrow |x, a \oplus f(x)\rangle$, then we can also apply the phase oracle $|x\rangle \rightarrow (-1)^{f(x)}|x\rangle$. Phase oracles are more convenient for the purposes of Grover's algorithm.

Next we apply a unitary matrix D (shown below), the so-called "Grover diffusion operator," which has the effect of flipping all N amplitudes about the mean amplitude.

$$\alpha_x \rightarrow 2\bar{\alpha} - \alpha_x \text{ where } \bar{\alpha} = \frac{1}{N} \sum_{x=1}^N \alpha_x$$

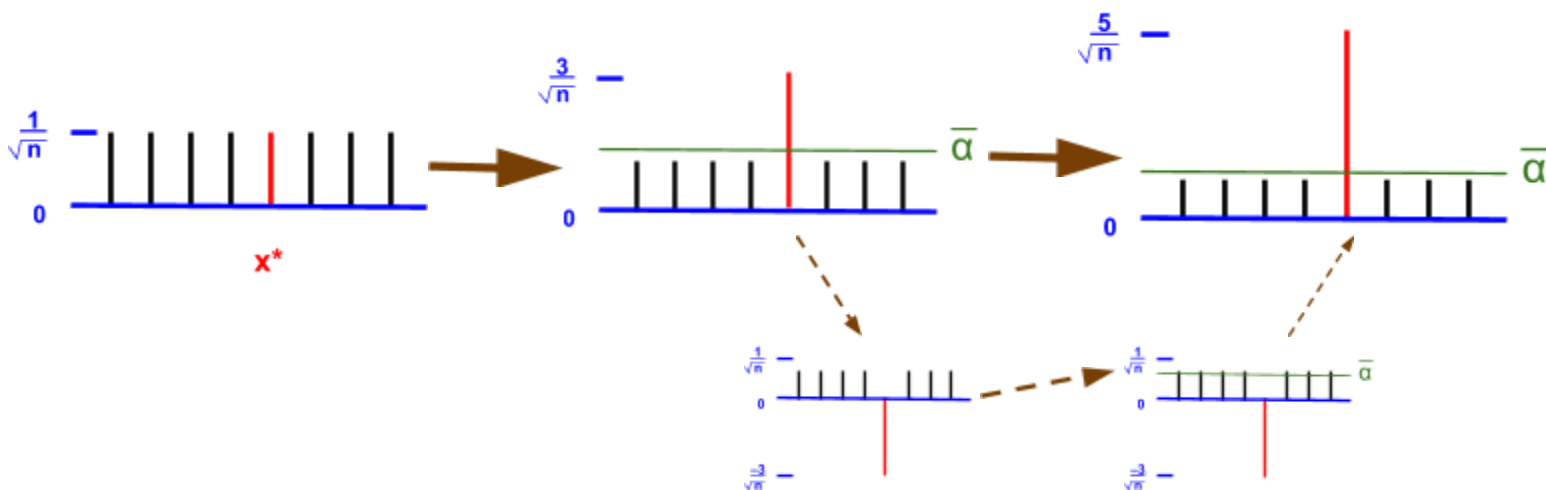
$$D = \begin{bmatrix} \frac{2}{N} - 1 & \frac{2}{N} & \dots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N} - 1 & \dots & \frac{2}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{N} & \frac{2}{N} & \dots & \frac{2}{N} - 1 \end{bmatrix}$$



So why does applying D help us?

Well, let's look at what's happened after a single Grover iteration. We've managed to increase the amplitude of the marked item, to roughly $3/\sqrt{N}$, and decrease the amplitudes of all the other items.

Now we keep repeating: another U_f , then another D , then another U_f , then another D , and so on. If we run those steps repeatedly, we can increase the amplitude of the marked item further as pictured below.



As an approximation, we can say that when the number of queries is small, the repetition increases the marked item's amplitude as

$$\frac{1}{\sqrt{N}}, \frac{3}{\sqrt{N}}, \frac{5}{\sqrt{N}}, \frac{7}{\sqrt{N}}, \dots$$

Notice that it would take $O(\sqrt{N})$ steps for this series to reach $\frac{\sqrt{N}}{\sqrt{N}} = 1$.

This is a quadratic speedup. Classically we'd need about N queries to find the answer, because after t queries we'd have a probability of $\frac{t}{N}$ of having found the marked item. Meanwhile, if we measure after t queries in Grover's algorithm, we find the marked item with probability of order $\left(\frac{2t}{\sqrt{N}}\right)^2 \sim \frac{4t^2}{N}$.

This picture isn't exactly right, though, because we ignored some details. Over time the mean gets smaller, so the increase in the marked item's amplitude slows down.

Which makes sense, because otherwise the amplitude would continue increasing past 1! We'll see exactly what happens shortly.

First, though, a natural question to ask about Grover's algorithm is "Why should it take \sqrt{N} steps? Why not $\sqrt[3]{N}$ or $\log N$?"

We see here that, in some sense, the ultimate source of the \sqrt{N} is the fact that amplitudes are the square roots of probabilities. Instead of adding $1/N$ probability to the marked item with each query, quantum mechanics lets us add $1/\sqrt{N}$ amplitude, resulting in quadratically faster convergence.

Of course, if we want to use Grover's algorithm in practice, then in addition to bounding the number of queries by $O(\sqrt{N})$, we'll *also* need to find a small quantum circuit to implement the Grover diffusion operator D .

Say we want to implement D on an n -qubit state ($N = 2^n$). It's easiest if we look at what D does in the Hadamard basis. What *does* it do in that basis?

$$\text{Well, the } \beta^{\text{th}} \text{ amplitude would be } \beta_s = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} (-1)^{s \cdot x} \alpha_x$$

We've seen previously that this is the result of switching to the Hadamard basis.

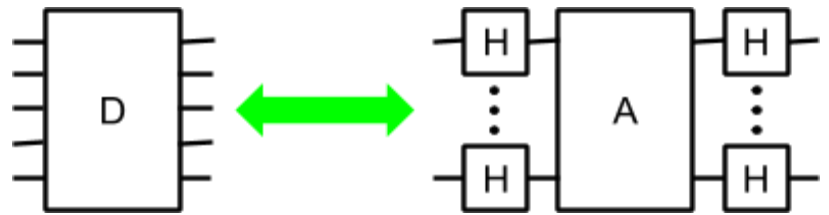
So what happens to these β_s 's?

The first one plays a special role: if $s = 0$, we have something proportional to the average, which is good because our goal was to invert about the average. The other s values play no particular role in Grover's algorithm.

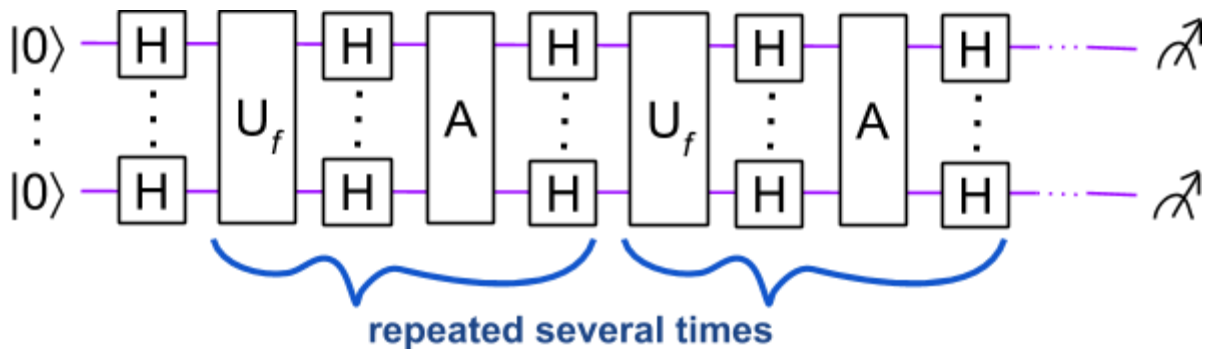
So in the Hadamard basis, if you think about it, what we want is to perform the diagonal matrix A on the right. This A is easy to implement as a quantum circuit, by using some ancilla qubits to check whether the input is all 0's, inverting the phase if not, and finally uncomputing garbage (details left as an exercise).

$$A = \begin{bmatrix} 1 & & & \\ & -1 & & \\ & & -1 & \\ & & & -1 \end{bmatrix}$$

Again, we claim that the A transformation, conjugated by Hadamard gates on either side, just implements the Grover diffusion transform D . If you don't believe this, you can verify it by an explicit calculation.



Our final circuit for Grover's Algorithm (with $\sim \sqrt{N} \log N$ gates and $\sim \sqrt{N}$ queries to f) is drawn below.



Now let's analyze Grover's algorithm more carefully, and actually prove that it works.

e.g. we haven't yet shown that $O(\sqrt{N})$ queries is enough to take us to $\text{Pr}(\text{success}) \approx 1$.

Let's call the initial state $|\Psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$

Somewhere in N -dimensional space is the basis state corresponding to the marked item we're looking for: $|x^*\rangle$.

What is $\langle \Psi | x^* \rangle$?

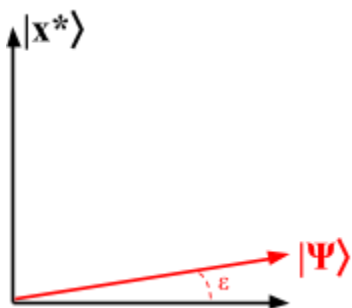
It's $\frac{1}{\sqrt{N}}$. So $|\Psi\rangle$ and $|x^*\rangle$ are not quite orthogonal, but *almost*.

Now, even though $|\Psi\rangle$ and $|x^*\rangle$ are not orthogonal, these two vectors span a two-dimensional subspace. A crucial insight about Grover's algorithm is that it operates entirely within this subspace. Why? Simply because we start in the subspace, and then neither the queries nor the Grover diffusion operations can ever cause us to leave it!

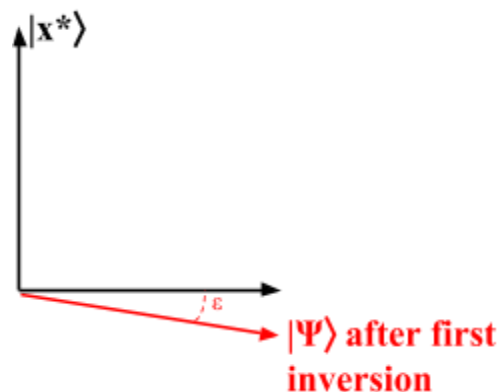
This means that we can visualize everything Grover's algorithm is doing by just drawing a picture in the plane.

We saw how the algorithm alternates between two types of operations:

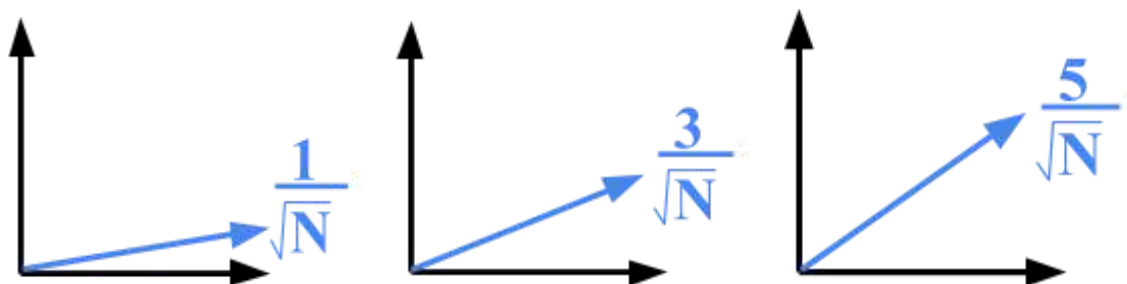
- Reflecting our state about $|\Psi\rangle$ (the diffusion transform D)



- Inverting the component of our state that points in the $|x^*\rangle$ direction (a query U_f)



So the initial angle with the horizontal is $\theta = \arcsin\left(\frac{1}{\sqrt{N}}\right) \approx \frac{1}{\sqrt{N}}$



And then we rotate by $\frac{2}{\sqrt{N}}$ at each iteration. This means that we'll get super close to $|x^*\rangle$, and have a high probability of observing x^* when we measure, after about $\frac{\pi}{4}\sqrt{N}$ iterations--something that you can directly see from the geometric picture. We might not get *exactly* to 1, if the step size of the rotations causes us to overshoot the vertical direction slightly, but at any rate we'll get close.

There are several interesting things about this picture...

What happens if we run Grover's algorithm for too long?

Well it has a property that almost no classical algorithm has: it starts getting worse.

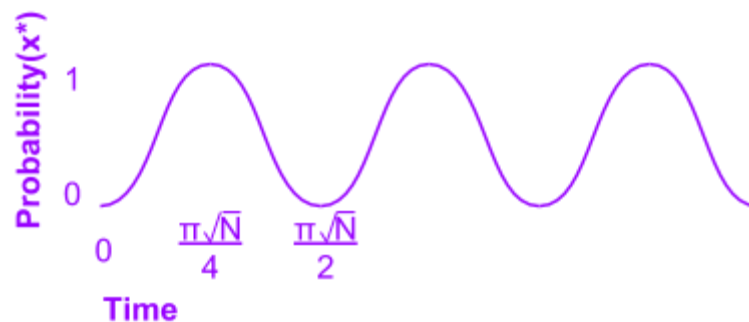
Grover's algorithm has been compared to baking a soufflé:

Once risen, it must be taken out of oven or it'll get short again.

On the other hand, Grover's algorithm has at least one property not shared by soufflés: namely, if you "leave it in the oven" for even longer, it rises a second time, then goes down a second time, and so on forever!



Graphing the success probability over time produces a sinusoidal curve, since the probability is just the squared projection of the current quantum state onto the y-axis (i.e., $\sin^2\Theta$):



Grover's algorithm can also put you into an interesting dilemma:

Suppose you've run the algorithm for a given number of iterations, fewer than $\frac{\pi}{4}\sqrt{N}$. Then you could measure right now, and take your chances with whether you'll observe the solution, or you could let it run longer to boost your chances. If you measure right now and *don't* see the solution, then you need to start over from the very beginning.

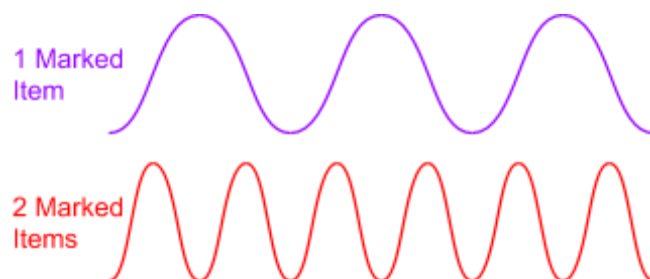
This could make for an interesting science-fiction story: the heroes need to break a cryptographic code to beat villains, so they run Grover's algorithm over all the possible decryption keys. They've run it for a day and gotten up to 45% probability of observing the solution, but now the villains have entered their compound and are closing in on them. So, do they measure now, or do they let the algorithm run a bit more?

Could you get a solution faster by doing many runs, each with measurement after a shorter amount of time?

It depends on the desired level of confidence in getting the right answer after a set amount of time. If your goal is just to minimize the *average* number of queries until you learn the answer, then it turns out to be optimal to stop some amount of time before you're able to get a guaranteed answer (details left as an exercise).

For simplicity, everything we said above assumed exactly one marked item. What happens if there are more? Simple: the entire cycle happens faster. The more items, the faster the cycle.

In particular, if k items out of N are marked, then Grover's Algorithm peaks at $\frac{\pi}{4}\sqrt{\frac{N}{k}}$ queries.



One of the first questions people asked about Grover's algorithm was, "but what if the number of marked items isn't known?"

You can sort of see the danger. It's possible to run Grover's algorithm the right number of times to hit the peak when there's a single marked item, but that might result in a trough if the number of marked items is larger.

The most basic way to solve this problem is simply to run the algorithm for a *random* number of iterations, say between 0 and $2\sqrt{N}$. If we do this, then most of the time, we expect to end up somewhere around the middle of a sinusoid, neither at a trough nor a peak, where we have a constant probability (say, 40% or 60%) of observing a solution if we measure. This is perfectly sufficient from an algorithmic standpoint, since it means that we only need to repeat the algorithm $O(1)$ times on average.

This gives us an upper bound of $O(\sqrt{N})$ queries to find a marked item with high probability, regardless of how many there are (assuming there's at least one).

While we weren't explicit about this point before, given a candidate solution x output by Grover's algorithm, we can simply evaluate $f(x)$ classically to check whether x is really a marked item.

What happens if we run Grover's algorithm, but the database turns out to have no marked items?

When we query f : nothing happens

When we do the diffusion transform: nothing happens

So, the state just remains a uniform superposition over all N items for the entire duration of the algorithm. This means that when we measure, we just get a random item. Then we can check that item and see that it isn't marked.

How can we be certain that there no marked items?

This is the question that arises in the decision version of Grover's algorithm. In fact, no matter how many times we run Grover's algorithm, we never become *100% sure* that there are no marked items, since we could've just gotten unlucky and failed to find the items every time. However, because after $O(1)$ repetitions, the algorithm has as high a probability as we like (say, >99.99%) of finding a marked item assuming that there's at least one of them, if after that time it *hasn't* found a marked item, then we can deduce that there almost certainly weren't any. This again requires only $O(\sqrt{N})$ queries.

We've said that if there are k marked items, then we find one of them in $O(\sqrt{N})$ queries without knowing k . But in fact we can do even better than that, and find a marked item in only $O(\sqrt{N/k})$ queries, again without knowing k : the same performance as if we *did* know k . How?

Assume for simplicity that N is a power of 2.

Then first, we guess that almost all items are marked: we do a single query, then measure. If we find a marked item, great.

If not, next we guess that $\frac{N}{2}$ items are marked, and run Grover's algorithm with $k=N/2$. If we find a marked item, great.

Next we run Grover's algorithm with $k=N/4$, then $k=N/8$, and so on, repeating halving our guess for the number of marked items, until either we've found a marked item or we've searched unsuccessfully with $k=1$.

This method wastes some queries on “wrong” values of k . But crucially, because the number of queries is increasing exponentially, the number of wasted queries is only a constant factor greater than the number of queries used in the final iteration: the one that guesses an approximately correct value of k . Details of the analysis are left as an exercise.

Let’s end by mentioning a different way to handle the case of multiple marked items. This way achieves essentially the same performance using a purely classical trick.

Again suppose we have N items, k of which are marked. We want to reduce to the case of just a single marked item.

How do we do that? Simple: we pick $\frac{N}{k}$ items uniformly at random, and then run Grover’s algorithm on that subset only.

The number of marked items that we’ll catch has a Poisson distribution. And one can calculate the probability of catching exactly one marked item in our sample as $\sim \frac{1}{e}$. So, we search that subset of $\frac{N}{k}$ items, using Grover’s algorithm for the single marked item case (which uses $O(\sqrt{N/k})$ queries). If we don’t find a marked item, we can try again with a new random subset.

Exercise for the reader: Show that, if there are k marked items and we want to find *all* of them, we can do that using $O(\sqrt{Nk})$ queries.

Lecture 23, Thurs April 13: BBBV,

Applications of Grover

It's great that we can get a quadratic speedup with Grover's algorithm, but we were able to get an *exponential* speedup with Shor's algorithm...

So why can't we get a bigger speedup for unordered search?

By now, you should have some intuition for the differences between Shor's algorithm and Grover's algorithm.

Shor's algorithm provided an exponential speedup by orchestrating a very "global" phenomenon: an interference effect that revealed the period of a black-box periodic function.

Grover's algorithm let us turn a little amplitude into a bigger amplitude by adding $1/\sqrt{N}$ with each query. It's faster than classical brute-force search, but still laborious ($\sim\sqrt{N}$ time).

We're *still* hand-waving the issue though. We haven't ruled out the possibility of a quantum algorithm that beats Grover, solving unordered search in $\sqrt[3]{N}$ or $\log(N)$ queries or whatever. For that we need...

The BBBV Theorem (Bennett, Bernstein, Brassard, Vazirani 1994)

which proved that Grover's algorithm is indeed asymptotically optimal for the black-box unordered search problem.

Note that the BBBV Theorem was published in 1994, so it actually predates Grover.

Grover's algorithm thus has the rare distinction of being an algorithm that was proved to be optimal *before* it was discovered.

Amusingly, BBBV were trying to prove that there's no magic way to search faster using a quantum computer. They were able to get a lower bound of $\sim\sqrt{N}$, and figured that tightening the bound to $\sim N$ was a technical issue that they could leave for the future—until Grover came along and showed why such a tightening is impossible.

While by now we know many proofs of the BBBV Theorem, the original (and still most self-contained) proof uses what's called a Hybrid Argument.

Imagine we're using an arbitrary quantum algorithm to search for a single marked item in a list of size N . Without loss of generality, we can say that the algorithm makes T queries and looks like this:

$$U_0 \rightarrow Q \rightarrow U_1 \rightarrow Q \rightarrow U_2 \rightarrow Q \rightarrow \dots$$

where each Q is a query, and each U_i is a unitary transformation that doesn't depend on the list. Now we do a test run of the algorithm on the all-zero list (without a marked item), and see what happens.

We'll argue that at least one item in the list was queried with a small amplitude, because there's only so much amplitude to go around. But now, if we change the value of *that item* from 0 to 1, then we can show that the algorithm

0	0	0	0	0 1	0	0
---	---	---	---	----------------	---	---

wouldn't much notice the change, by exploiting the fact that unitary transformations are linear and norm-preserving.

More concretely, we'll show that the final state of the algorithm is at most $O(\frac{t}{\sqrt{N}})$ away from what it would have been, had we kept the list all-zero.

Note: while we'll only consider algorithms that apply unitary transformations, exactly the same proof carries over to algorithms that have intermediate measurements---because we can always model a measurement by a unitary transformation on a larger set of qubits, just like in the Many-Worlds Interpretation.

The argument is called "hybrid" because we'll create hybrid oracles, which answer the first queries as if they're the all-zero oracle, but then switch partway through the algorithm to having a single "1" entry.

What is the state of the algorithm immediately before the t^{th} query?

$$|\Psi_t\rangle = \sum_{x, w} \alpha_{x,w,t} |x, w\rangle \quad \text{Assuming the all-zero input.}$$

x is whichever list item that we're querying next.

w is what's known in quantum algorithms as "the workspace," any qubits that the algorithm might use for internal purposes but that don't participate in the query.

In Grover's algorithm there's hardly any "workspace" to speak of: we do use auxiliary qubits to implement the diffusion operator, but those qubits are reset to their original state by the time the diffusion operator is finished. But the BBBV Theorem, to be fully general, will allow for unlimited workspace.

We'll show that regardless of the workspace, the algorithm would require $\sim \sqrt{N}$ queries.

$\alpha_{x,w,t}$ is the amplitude of basis state $|x, w\rangle$ at time t .

Now we define the "query magnitude" of an element $x \in \{1, \dots, N\}$ to be

$$M_x = \sum_{t=1}^T \sum_w |\alpha_{x,w,t}|^2$$

I.e., the query magnitude is the sum, over all time steps t , of the probability that we would find the algorithm querying item x if we measured at time t .

Observe that by doing some rearranging, we find that the sum of all the query magnitudes is

$$\sum_{x=1}^N M_x = \sum_{t=1}^T \left(\sum_{x=1}^N \sum_w |\alpha_{x,w,t}|^2 \right) = \sum_{t=1}^T 1 = T$$

Since the sum is T , the average query magnitude is $\frac{T}{N}$. Now, any list of numbers has at least one number that's at most the average.

This is sometimes referred to as the "Lake Wobegon Principle," after the fictional town where everyone was above average.

So let x^* be a list element with query magnitude $M_{x^*} \leq \frac{T}{N}$.

The idea here is that the algorithm only has so much amplitude to spread around, and thus most database items must not get “monitored” too closely. So if we pick one such item, x^* , and make it the marked item, then the algorithm will mostly fail to notice the change.

More formally, we have

$$\sum_{t=1}^T \sum_w |\alpha_{x^*,w,t}|^2 \leq \frac{T}{N}.$$

But it would be more useful to us to have an upper bound on

$$\sum_{t=1}^T \sqrt{\sum_w |\alpha_{x^*,w,t}|^2}.$$

To get from one to the other we use the Cauchy-Schwarz Inequality, which is super useful in quantum information (and many other fields).

Given a unit vector $\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix}$, what is the maximal sum of the absolute values of its entries, $\sum_{n=1}^N |\alpha_n|$?

The Cauchy-Schwarz Inequality says that we can maximize the sum by making all entries equal, with the vector $\begin{bmatrix} \frac{1}{\sqrt{N}} \\ \vdots \\ \frac{1}{\sqrt{N}} \end{bmatrix}$, so that the sum is $\frac{N}{\sqrt{N}} = \sqrt{N}$.

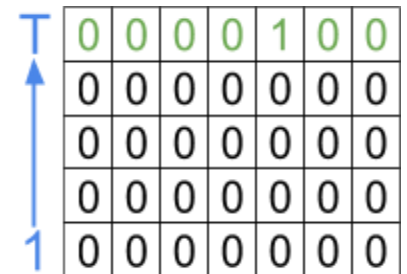
So what does the Cauchy-Schwarz Inequality say about our case?

Well, if each of the T terms of the form $\sum_w |\alpha_{x^*,w,t}|^2$ were set equal to $\frac{1}{N}$, then the sum of their square roots would be $\frac{T}{\sqrt{N}}$. So this is the maximum.

Why is this relevant?

Now comes the hybrid part of the argument.

Picture a table where each row is our database at a particular point in time, with time increasing upwards. Initially the table is filled with zeros, meaning that the oracle answers all queries with zero.



0	0	0	0	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Now we’re going to change the table so that the oracle answers $f(x^*) = 1$ for the last query (and *only* for the last query). This means that the state of the algorithm after the final query $|\Psi_T\rangle$, is going to change from what it was before---but we know it can change only in branches that were lucky enough to query x^* .

So how much can things change (in Euclidean distance)? Well, since the query flips the amplitude, the change is equal to the total norm with which we made the query, times 2:

$$|\Psi_T\rangle - |\Psi_T'\rangle = 2 \sqrt{\sum_w |\alpha_{x,w,T}|^2}$$

Here we're using the fact that, by assumption, the states immediately before the T^{th} query are identical in the two situations.



So what happens if the oracle treats x^* as marked only for the last two queries?

The total amplitude devoted to querying x^* before the final amplitude is $2 \sqrt{\sum_w |\alpha_{x,w,T-1}|^2}$,

so we get

$$|\Psi_{T-1}'\rangle - |\Psi_{T-1}''\rangle \leq 2 \sqrt{\sum_w |\alpha_{x,w,T-1}|^2}.$$

But couldn't the last query push these further apart?

Here we come to a crucial point: we claim that it can't. For in both cases, the last query is applying the same unitary transformation, which means that the inner product between the states can't change.

So we get $|\Psi_T'\rangle - |\Psi_T''\rangle \leq 2 \sqrt{\sum_w |\alpha_{x,w,T-1}|^2}$ and hence

$$|\Psi_T'\rangle - |\Psi_T''\rangle \leq 2 \sqrt{\sum_w |\alpha_{x,w,T-1}|^2} + 2 \sqrt{\sum_w |\alpha_{x,w,T}|^2} \text{ by the triangle inequality.}$$

Continuing in the same way for all T of the queries, we find that $|\Psi_T'\rangle - |\Psi_T''\rangle$ is upper-bounded by $2 \frac{T}{\sqrt{N}}$. This means, in particular, that after we measure at the end of the algorithm, we can have found the marked item x^* with probability at most $O(\frac{T^2}{N})$, precisely the success probability that Grover's algorithm achieves.

This "BBBV Theorem" has since been enormously generalized, to a whole theory about lower bounds on quantum query complexity, which unfortunately we won't really enter into in this course--but see Prof. Aaronson's graduate course for more!

Now that we understand Grover's algorithm, we can apply it to solve many, many problems that are not quite as simple as unordered search. Our first example of this is...

The OR's of AND's

N input bits are arranged into a square table of size \sqrt{N} by \sqrt{N} .

The problem is to determine: are there any rows with all 1's?

\sqrt{N}

0	0	0	0	1	0	0
1	0	1	1	0	1	0
0	0	1	0	0	0	0
1	1	1	1	1	1	1
0	1	0	1	0	0	1
0	0	0	0	1	0	0

\sqrt{N}

This problem lets us encode many other problems that involve multiple quantifiers. For example, in chess, we may want to know “Is there a move I can make such that my opponent has no possible response that checkmates me?”

Classically, it’s clear that you have to look through almost the entire table, searching each row until you’ve either found a 0 or found that the row is all 1’s.

Quantumly, we could speed this up by searching each row for 0’s using Grover’s algorithm. The running time for each row would be $\sqrt{\sqrt{N}} = N^{1/4}$, or technically $N^{1/4} \log N$, if we repeat the Grover search on each row enough times to have (say) a $1/N$ probability of error. This means that searching the whole table would take time

$$\sqrt{N} N^{1/4} \log N = N^{3/4} \log N.$$

Alternatively we could do Grover’s algorithm over all the rows, such that each row is counted as a “marked item” if and only if a classical algorithm (which we run as an inner loop) finds a zero in that row. This also has an $\sim N^{3/4}$ runtime.

Naturally, the next idea is to run Grover’s algorithm recursively, *inside of itself*, where the outer Grover (over the rows) will count a given row as being marked, if and only if the inner Grover failed to find a zero in that row. Again, because Grover’s algorithm has some probability of error, at least naïvely we have to repeat the inner runs about $\log N$ times to push the error probability per row down to about $\frac{1}{N}$.

So our final runtime is $O(\underbrace{\sqrt{\sqrt{N}}}_{\text{outer G.A.}} \underbrace{\sqrt{\sqrt{N}}}_{\text{inner G.A.}} \underbrace{\log N}_{\text{Error Avoidance}}) = O(\sqrt{N} \log N)$ **the Grover speedup!**

With some cleverness, people have since been able to remove the $\log N$ factor.

Why couldn’t we just do Grover’s algorithm once, over the whole table?

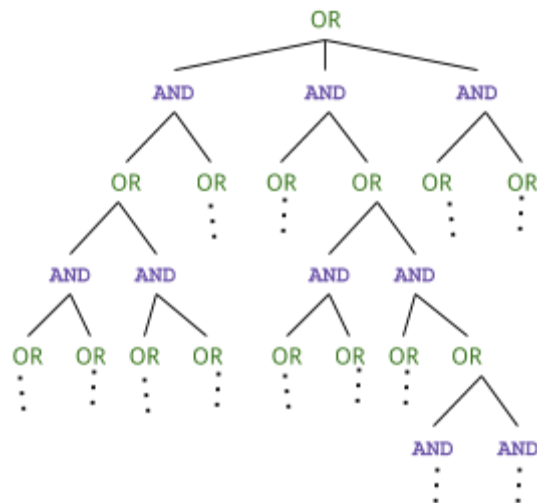
Well, just because there’s a 0 *somewhere* in the table, doesn’t mean that there couldn’t be a row of all 1’s somewhere else.

The first problem in quantum computing that Professor Aaronson worked on was trying to generalize the BBBV Theorem to show that “recursive Grover” is optimal: in other words, that any possible quantum algorithm for the OR-of-ANDs requires at least $\sim \sqrt{N}$ queries. (The obvious lower bound is only $\sim N^{1/4}$.) He spent a whole summer trying to solve this problem using the “polynomial method,” but couldn’t crack it. Later, Andris Ambainis, then a PhD student at Berkeley, invented a totally new method for proving lower bounds on quantum query complexity, and applied it to solve this problem.

Which was why Prof. Aaronson decided to go to Berkeley for grad school.

We could easily generalize this to evaluate (e.g.) an OR of ANDs of ORs by doing *three* recursive layers of Grover search, and so forth.

If we allow an arbitrary number of layers, then we get the A.I. concept of game trees, for two-player games of alternation such as chess and Go. Here the goal is to find a move you can make (represented by an OR over various options), which given any move that your opponent makes (represented by ANDs), allows for a move that you can make, that for any move your opponent makes, etc. ... eventually wins you the game.



The problem is that, as the game tree gets deeper and deeper, the advantage of Grover's algorithm over classical search *seems* to get weaker and weaker, for two reasons: first, the amplification that's needed at each layer to prevent error buildup, and second, the constant factors, which multiply across the layers. Note that each layer actually needs to run Grover's algorithm on the layer below it *twice*:

- Once to do $|x\rangle \rightarrow |x\rangle/f(x)\rangle$
- And once to uncompute garbage.

For this reason, the constant factor $\pi/4$, in the running time of Grover's algorithm, actually becomes $\pi/2$, and $\pi/2 > 1$.

In short, none of this answers the natural question: "*Can a quantum computer help you play chess?*" For game-tree search with a deep enough tree, Prof. Aaronson and some others conjectured that the diminishing returns from Grover's algorithm would end up negating any asymptotic advantage over a classical computer.

In 2007, however, Farhi, Goldstone, and Gutmann, and others who built on their work, dramatically refuted that conjecture. The upshot of their work is that we now know how to evaluate *any* game tree with N leaves, no matter how deep, in $O(\sqrt{N})$ time on a quantum computer. (This is also known to be asymptotically optimal.)

So, yes, quantum computers probably *would* help you play chess!

To put some numbers on this: Claude Shannon famously estimated the number of possible board positions in chess as $\sim 10^{43}$, which is certainly out of range for any existing computer on earth. But if quantum computers brought that down to $\sim 10^{21.5}$, solving chess might *just* be doable.

Though it raises a philosophical question: Have you actually "solved" chess if you don't have a solution table that anyone can examine, but only a quantum computer that always wins?

In the next lecture, we'll see some additional applications of Grover's algorithm, to the so-called *collision* and *element distinctness* problems.

Lecture 24, Tues April 18: Collision and Other Applications of Grover

We've seen the application of Grover's algorithm to searching game trees. Now let's see another important application, to...

The Collision Problem

In the simplest version of this problem, we're given quantum black-box access to a function

$$f: \{1, \dots, N\} \rightarrow \{1, \dots, N\}$$

where N is even, and we're promised that f is two-to-one. The problem is to find x and y such that $x \neq y$, and $f(x) = f(y)$. So, there are lots of "collisions" to be found, and the challenge is just to find one of them.

In another version of the problem, we're promised that *either* f is one-to-one or it's two-to-one, and the problem is to decide which. Clearly, if we could solve the "search" version, then we could also solve the "decision" version, by simply outputting that f is two-to-one if a collision is found, or that f is probably one-to-one if not. Conversely, any lower bound for the decision version implies the same lower bound for the search version.

The collision problem often arises in cryptography, when we're trying to break collision resistant hash functions. You can think of the collision problem as being a lot like Simon's Problem but with less structure---or alternatively, as being like the Grover search problem but with *more* structure.

With a classical randomized algorithm, if we have black-box access to f , then $\Theta(\sqrt{N})$ queries are necessary and sufficient to solve the collision problem. Why? The upper bound follows from the famous "birthday attack": if there are N days in the year, then you only need to ask about \sqrt{N} people before there's an excellent chance that you'll find two with the same birthday, because what matters is the number of *pairs* of people. The lower bound can be proven using the union bound: with a random two-to-one function, each pair has only a $\sim 1/N$ chance of being a collision, so to find a collision with constant probability you need to look at $\sim \sqrt{N}$ pairs or more.

What about quantumly? Well, we could of course simulate the above randomized algorithm to get $\sim \sqrt{N}$. But there's also a completely different way to get $\sim \sqrt{N}$: namely, we could first query $f(1)$, and then do a Grover search for an $x \neq 1$ such that $f(x) = f(1)$. So a question arises: can we combine the two approaches to do even better than $\sim \sqrt{N}$?

(Brassard, Hoyer, Tapp 1997) showed how to do exactly that. Here's their algorithm:

First, pick $\sqrt[3]{N}$ random inputs to f , query them classically, and sort the results for fast lookup.

Next, run Grover's algorithm on $N^{2/3}$ *more* random inputs to f (inputs that weren't queried in the first step). In this Grover search, count each input x as "marked" if and only if $f(x) = f(y)$ for one of the $\sqrt[3]{N}$ inputs y that was already queried in the first step. (This requires lookups to our sorted list, but no additional queries to f .)

How many pairwise comparisons do we make this way?

$$N^{2/3} \times N^{1/3} = N$$

What's the runtime?

$$N^{1/3} + \sqrt{N^{2/3}} = O(N^{1/3})$$

The centerpiece of Professor Aaronson's PhD thesis was showing that you can't improve on this by much.
It was later shown by Yaoyun Shi that you can't improve on it at all.

The BHT algorithm gives a good illustration of how quantum algorithms can end up with weird running times. You have two or more phases of the algorithm that you try to balance against each other, make about equally time-consuming, in order to minimize the total time,.

At a high level, you can see why the BBBV proof that we used to prove the optimality of Grover's algorithm doesn't work for the collision problem. In the BBBV proof, we changed a single element from 0 to 1, then showed that it would take many iterations for the algorithm to notice. With the collision problem, by contrast, the key issue is that turning a one-to-one function into a two-to-one function requires changing half the elements.

Instead, Aaronson and Shi used polynomial approximation theory (a branch of math) to rule out super-fast quantum algorithms for the collision problem.

In some sense, proving a quantum lower bound for the collision problem *should* be harder than proving one for the Grover problem, because if the lower bound for collision did too much, then it would rule out things like Simon's algorithm or Shor's algorithm. What the proof does is take advantage of the symmetry of the collision problem.

Symmetry in the sense that you can arbitrarily permute the function in the collision problem, and it's still a valid input, which isn't the case for something like Simon's problem.

More broadly, after Grover published his algorithm, there was a ten-year flood of people realizing you can use it for speedups in all sorts of problems.

For example, in addition to the collision problem, there's also the closely related problem of ...

Element Distinctness

Given black-box access to the function $f: \{1, \dots, N\} \rightarrow \{1, \dots, N\}$, with no promises about f .
Determine if f is one-to-one.

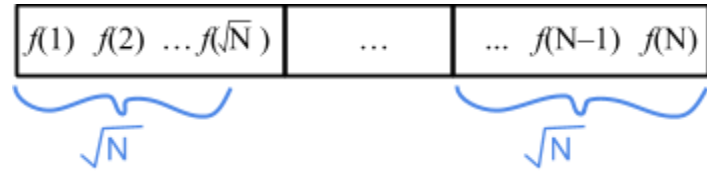
In other words: Are there any duplicates/collisions?

Classically, you'd hash the elements (or sort them, or use a binary search tree, etc.), which would take N queries plus the amount of computation required for sorting (say, $N \log N$ steps).

Quantumly, there's an algorithm that takes only $O(N^{3/4})$ queries, as shown in a paper of Buhrman et al. from 2000.

This is another cute application of Grover search.

Given a list of the N values of a function, we split them into \sqrt{N} blocks of \sqrt{N} values each. Then, by using Grover's algorithm over these blocks, we produce a quantum algorithm that makes $O(\sqrt{N})$ queries to f and finds a collision (if there is one) with probability $\frac{1}{\sqrt{N}}$.



So, how can we do that?

Pick a block at random and query all elements in it.

If you find a collision in the block, you're already done! If you don't, then sort the elements in the block for fast lookup. Next, do a Grover search on the other items in the list, counting an item as "marked" if and only if it equals an element from the collision block.

As long we were lucky enough to pick a block that contains at least one element of a collision pair, this algorithm will find such a pair with constant probability. Hence it succeeds with $\frac{1}{\sqrt{N}}$ probability.

We can improve on this (after all, we *do* have access to a quantum computer), by doing an outer layer of Grover search that searches through the \sqrt{N} blocks, counting a block as "marked" if and only if the "inner" algorithm above finds a collision involving that block.

Our final runtime is $\sqrt{\sqrt{N}} * \sqrt{N} = O(N^{3/4})$

Outer Grover Inner collision search Element Distinctness in $N^{3/4}$

What's the lower bound on Element Distinctness?

As a baseline, we know the query complexity has to be at least that of searching a list for a given i , which is $\sim \sqrt{N}$. Pinning down the complexity of Element Distinctness between $\sim \sqrt{N}$ and $\sim N^{3/4}$ was an open problem for several years.

As it turns out, the answer is $\sim N^{2/3}$.

Yaoyun Shi noticed that an $\sim N^{2/3}$ lower bound follows from the $\sim N^{1/3}$ lower bound for the collision problem.

That is, suppose for a contradiction that we could solve Element Distinctness in $t \ll N^{2/3}$ queries.

This would let us solve the collision problem in $\sqrt{t} \ll N^{1/3}$ queries.

How?

Given a 2-to-1 function f , pick $\sim \sqrt{N}$ inputs uniformly at random. Since, by the birthday paradox, we can expect to find a collision within that set of inputs (with constant probability), we now simply run our hypothesized Element Distinctness algorithm on that subset.

Matching this lower bound, in 2003 Andris Ambainis found a quantum algorithm that solves Element Distinctness with $O(N^{2/3})$ queries. His algorithm used “quantum walks,” which are vaguely like Grover’s algorithm but more sophisticated. It also required a huge amount of workspace qubits, on the order of $N^{2/3}$. Whether this large number of workspace qubits is necessary remains open to this day.

OK, how about one more vignette on the quantum query complexity of fundamental problems from computer science, which is now something we know a lot about.

Parity of an n -bit String

Given $x \in \{0,1\}^n$, suppose we just want to determine $x_1 \oplus x_2 \oplus \dots \oplus x_n$ (i.e., whether the total number of 1 bits is odd or even).

Classically, of course, this requires n queries. Quantumly, we’ve seen that we can do it in $\frac{n}{2}$ queries, by splitting x into $\frac{n}{2}$ pairs and then applying the Deutsch-Jozsa algorithm separately to each pair.

A beautiful result shows that this is optimal: $\frac{n}{2}$ queries are needed by any quantum algorithm for Parity. This can be shown using the polynomial method (Beals et al., 1998). Just to give you a brief taste:

Suppose that we have quantum algorithm A , which makes t queries to an input string x . We can study the probability that A accepts x , call it $p(x)$.

For simplicity we’ll assume that A is trying to compute a Boolean function, so it either accepts or rejects.

A critical fact proven by Beals et al. is that $p : \{0,1\}^n \rightarrow \mathbb{R}$ turns out to be a multivariate polynomial in the n bits of x . Furthermore, the degree of that polynomial is at most $2t$.

So, suppose we can show that any polynomial p that can approximate a given Boolean function f must have $\deg(p) \geq D$. Then we can deduce that the quantum algorithm must have made at least $\frac{D}{2}$ queries.

This reduces questions about quantum query complexity to purely mathematical questions about the degrees of real polynomials, with no further CS or quantum computing needed!

In the case of Parity, it turns out one can show that any polynomial approximating the Parity function needs degree n . This implies that any quantum algorithm for Parity must make at least $\frac{n}{2}$ queries.

As a complement to Parity, it’s also worth briefly discussing the n -bit Majority function, which outputs 0 or 1 depending on whether the input string has more 0’s or 1’s respectively. The quantum query complexity of Majority turns out to be order n --i.e., there is no asymptotic quantum speedup for this problem--and that can also be proved using the polynomial method.

However, there *is* a quantum speedup for a problem closely related to Majority.

For a poll to be accurate within x percent, how many people do you need to classically query?

Suppose you want to approximate the Hamming weight (i.e., number of 1's) in your n -bit input string, to within an additive error $\pm \epsilon n$.

Classically, you can do this by sampling $\sim \frac{1}{\epsilon^2}$ uniformly random bits and taking their average, and this is also tight. (This fact is extremely useful to know when, e.g., choosing the sample size for a political poll, to achieve a desired margin of error like $\pm 3\%$.)

Quantumly, by contrast, via a clever application of Grover's algorithm, it turns out that we can solve this problem using only $\sim \frac{1}{\epsilon}$ queries: a quadratic speedup.

To conclude this lecture, let's talk a bit about **Quantum Complexity Theory**, the generalization of computational complexity theory to the quantum realm, so we can understand the broader context of the quantum algorithms we've seen. Classically, we define complexity classes such as:

P (Polynomial-Time), the class of decision problems solvable by a standard, deterministic digital computer in polynomial time.

(examples: linear programming, connectivity of graphs)

NP (Nondeterministic Polynomial-Time), the class of decision problems for which there's a deterministic polynomial-time algorithm to *verify* yes-answers.

(example: factoring, when suitably phrased as a yes-or-no decision problem)

NP-hard problems are, roughly speaking, problems to which every **NP** problem can be reduced in polynomial time. So in particular, if you could solve any **NP-hard** problem in polynomial time, then you'd be able to solve everything in **NP** in polynomial time.

NP-complete problems are those that are both in **NP** and **NP-hard**. Informally, they're "the hardest problems in **NP**."

(examples: Traveling Salesman, 3SAT, Max Clique, Bin Packing, VLSI layout, Sudoku, Super Mario, and many other problems of practical and not-so-practical importance)

This picture already involves an enormous mathematical unknown: famously, no one has ruled out the possibility that **P** = **NP**, in which case all **NP** problems (so in particular, all **NP-complete** problems) would be solvable in polynomial time.

Where does quantum computing fit in?

In 1993, Bernstein and Vazirani defined the complexity class **BQP** (Bounded-Error Quantum Polynomial-Time) as a quantum generalization of **P**. Loosely speaking, **BQP** contains all decision problems that can be solved in polynomial time with a quantum computer.

How does **BQP** relate to classical complexity classes?

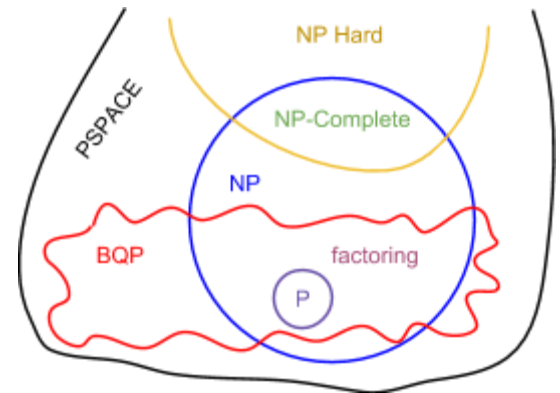
We know that **P** \subseteq **BQP**, basically because Toffoli gates can simulate AND, OR, and NOT gates, and hence universal classical digital computation--and hence quantum computers can simulate classical ones. We also know, from Shor's algorithm, that Factoring (when suitably phrased as a decision problem) is in **BQP**, though it's not known (to put it mildly) whether Factoring is in **P**.

We don't know whether **NP** \subseteq **BQP**--that is, whether quantum computers can solve all **NP** problems (including **NP-complete** problems) in polynomial time. The BBBV Theorem does tell us that there isn't an easy proof of **NP** \subseteq **BQP** to be had that just treats the **NP** problem as a black box.

“Can quantum computers solve **NP**-complete problems in polynomial time?”
is one of the big open problems of Quantum Complexity Theory.

We could also ask the converse, “Is **BQP** \subseteq **NP**?” In other words: for every problem that a quantum computer can solve, is there a short proof of the answer that’s easy to verify classically?

It’s possible that there are problems that a quantum computer could solve easily which can’t be classically solvable, but we don’t have any present examples.



The last important question to ask here is, “If **BQP** doesn’t seem to be contained in **P**, and maybe not even in **NP**, then what *is* it contained in?” In other words:

What classical class gives an upper bound on what a quantum computer can do?

Well, Bernstein and Vazirani showed that it’s possible to simulate a quantum computer classically with exponential time and polynomial memory, basically by writing an amplitude of interest as a sum of exponentially many contributions, and then evaluating the contributions one by one, reusing the same memory each time, and adding them to a running total.

This gives us an upper bound: **BQP** \subseteq **PSPACE**, where **PSPACE** (Polynomial Space) is the class of problems solvable on a digital computer using a polynomial amount of memory, but possibly exponential time.

It’s possible to get a better upper bound on **BQP**, but it involves other complexity classes that we won’t define here.

So, what would it take to prove that **P** is different from **BQP**? Of course this would follow if Factoring wasn’t in **P**, but proving the latter would require showing **P** \neq **NP**! So, is there better hope for proving **P** \neq **BQP** in the near future than there is for proving **P** \neq **NP**?

Unfortunately, not so much. The reason is that **BQP** is sandwiched between **P** and **PSPACE**:

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

For this reason, any proof of **P** \neq **BQP** would also need to show that **P** \neq **PSPACE**, which is a big unsolved problem in itself.

Lecture 25, Thurs April 20: Hamiltonians

Now we'll move on to our second-to-last unit...

Hamiltonians and the Adiabatic Algorithm

We've seen how it's an open question whether quantum computers can solve **NP**-complete problems in polynomial time. If this turned out to be possible, it would be world-changing.

Like, it would be time for a Manhattan Project to build scalable quantum computers...

But if it turns out that quantum computers can't solve **NP**-complete problems in polynomial time, the question still remains, "How close can they get to solving?"

We know from the BBBV Theorem that any approach that ignores the structure of **NP**-complete problems will only yield the Grover speedup.

There have been many papers on arXiv.org that claim to solve **NP**-complete problems in polynomial time with a quantum computer, but do so in ways that violate this theorem.

Virtually all quantum computing papers can be found on arXiv.org, but the site has no peer review.

So to do better than Grover, we'd need to exploit problem structure in some way. For example, with Boolean satisfiability, we could imagine devising some quantum algorithm that dealt with certain parts of the formula first, and worried about other parts later.

If we managed to show that *any* **NP**-complete problem was in **BQP** (i.e., solvable in polynomial time by a quantum computer), then by definition, *all* of **NP** would be in **BQP**.

However, if we're talking about small speedups, then the choice of **NP**-complete problem might actually matter, because the process of reduction from one **NP**-complete problem to another might cancel out a speed advantage.

The Adiabatic Algorithm (Farhi, Goldstone, Gutmann, Sipser 2000)

is a famous attempt to do exactly the above--i.e., get a quantum speedup for **NP**-complete problems (conceivably, even an exponential speedup) by actually exploiting their structure.

It's an extremely important quantum algorithm, but unlike (say) Shor's or Grover's algorithms, it doesn't come with any rigorous analysis guaranteeing it will run fast in all cases--and indeed, we now know that it doesn't. To this day, no one really knows how useful this algorithm will be in practice.

It's something that people will eagerly experiment with, as soon as they have reliable large-scale quantum computers to test it on!

For some instances of optimization problems, the adiabatic algorithm might give a huge speed advantage, but for other instances it gives little or no advantage, or is even outperformed by classical algorithms. People are still trying to figure out for which types of instances the algorithm is most useful.

To understand the adiabatic algorithm, we first need to back up, and familiarize ourselves with a central concept in quantum mechanics called **Hamiltonians**.

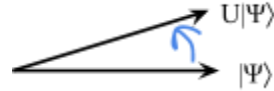
In a physics course on quantum mechanics, Hamiltonians would be day-one material,

while we'd only get to quantum computing and information at the very end (if at all). In this course, it's exactly the opposite!

Recall that unitaries are discrete linear transformations of quantum states:

$$|\Psi\rangle \rightarrow U|\Psi\rangle$$

But a physicist would treat time as *continuous*, and say that the state $|\Psi\rangle$ rotates continuously to $U|\Psi\rangle$ over some interval of time.



Hamiltonians are just the instantaneous time generators of unitary transformations. I.e., they're things that give rise to unitary transformations when you "leave them running" for some period of time. Like density matrices, Hamiltonians are described by *Hermitian matrices*. (But unlike density matrices, Hamiltonians don't need to be positive semidefinite or to have trace 1.)

Remember: for H to be Hermitian means that $H = H^\dagger$

From a physics perspective, the central equation of quantum mechanics is **Schrödinger's Equation**:

$$i \frac{d|\Psi\rangle}{dt} = H|\Psi\rangle \quad \text{with } H \text{ being some Hamiltonian.}$$

This equation describes the evolution of an isolated quantum pure state in continuous time.

Well, the full version of Schrödinger's equation also includes the so-called *Planck's constant* \hbar , which is needed to convert between units of time and units of energy. But unless we're dealing with actual experimental data, involving meters, seconds, joules, and so forth, it's more convenient just to set $\hbar=1$ -- the convention we'll adopt throughout! And for future reference: if it ever comes up, the speed of light c is also 1.

We can solve Schrödinger's equation, to find that the state after time t is

$$|\Psi(t)\rangle = e^{-iHt} |\Psi(0)\rangle.$$

Basically, we have here a whole *system* of linear differential equations--one for each coordinate of the vector $|\Psi\rangle$ --but we can formally solve it by pretending that the matrix H is a scalar.

Here, though, we need to back up to address a mathematical point:

What does it mean to raise e to the power of a matrix?

The "right" definition turns out to be: you take the standard Taylor series for the exponential function,

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!},$$

and then just plug in a matrix instead of a number, to get a matrix-valued result.

To give some examples:

$$e \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad e \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} e & 0 \\ 0 & 1 \end{bmatrix}$$

and

More generally, we can say that

$$e^{\begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}} = \begin{bmatrix} e^{\lambda_1} & & \\ & \ddots & \\ & & e^{\lambda_n} \end{bmatrix}$$

I.e., you can exponentiate a diagonal matrix by exponentiating each diagonal term individually.

That might look like a very special case, but for Hermitian matrices, in some sense it's really all we need. Suppose we're given a matrix A , which can be written as $A = UDU^{-1}$ where D is diagonal.

Then to compute e^A we can write:

$$e^A = \sum_{k=0}^{\infty} \frac{(UDU^{-1})^k}{k!} = \sum_{k=0}^{\infty} \frac{UD^kU^{-1}}{k!} = Ue^DU^{-1}$$

Thus we have a simple algorithm to exponentiate any diagonalizable matrix.

What leverage do we get from H being Hermitian?

For what we've said to make sense--and in particular, for it to be consistent with the discrete-time version of QM we've used in the rest of the course--it better be the case that the matrix e^{-iHt} is unitary. Let's now prove this to be true, by using the fact that H is Hermitian.

First claim: If H is Hermitian, then all its eigenvalues are real.

This is a special property of Hermitian matrices, although it's not "if and only if."

Proof: Suppose λ is an eigenvalue. Then by definition, there's some eigenvector $|v\rangle$ such that

$$H|v\rangle = \lambda|v\rangle,$$

$$\langle v|H|v\rangle = \lambda.$$

What do we get by conjugate-transposing the whole thing?

$$\langle v|H^\dagger|v\rangle = \bar{\lambda}$$

But since $H = H^\dagger$, this is equivalent to

$$\langle v|H|v\rangle = \bar{\lambda},$$

$$\langle v|H^\dagger|v\rangle = \lambda.$$

So $\lambda = \bar{\lambda}$, which means $\lambda \in \mathbb{R}$.

Something stronger is also true: every Hermitian matrix is diagonalizable.

One can prove that by induction on the dimension of the matrix. We won't go through the details here. Generalizing these facts, the Spectral Theorem says that any Hermitian matrix can be written as

$$H = UDU^\dagger \quad \text{with } U \text{ being unitary and} \\ \quad \quad \quad D \text{ being diagonal and real}$$

Now, to show that e^{-iHt} is unitary, just diagonalize H:

$$\begin{aligned} e^{-iHt} &= e^{-itUDU^\dagger} \\ &= U e^{-itD} U^\dagger \end{aligned}$$

But $e^{-itD} = \begin{bmatrix} e^{-it\lambda_1} & & \\ & \ddots & \\ & & e^{-it\lambda_n} \end{bmatrix}$

which is a diagonal unitary matrix (since $\lambda_1, \dots, \lambda_n$ are real). Hence e^{-iHt} is a unitary matrix.

Note that, if $|\nu\rangle$ is an eigenvector of H associated with the eigenvalue λ ,

then $|\nu\rangle$ is also an eigenvector of e^{-iHt} , associated with the eigenvalue $e^{-i\lambda t}$.

So, eigenvectors of H give rise to eigenvectors of the unitary.

Now, what about going backwards:

Given a unitary U, can we find a Hermitian matrix H such that $U = e^{-iHt}$?

Yes, this is not hard.

First diagonalize U, to get $U = VDV^\dagger$.

We then just need to take a logarithm of each diagonal entry of D:

That is, for each D_{jj} , find a λ such that $D_{jj} = e^{-i\lambda t}$.

Will the λ that we get by solving this be unique?

No, because by Euler's formula, we can always add $2\pi i$ to the exponent and the equation will still hold.

We saw $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ that is a logarithm of the identity matrix. What else is?

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = e^{i \begin{bmatrix} 2\pi & 0 \\ 0 & 2\pi \end{bmatrix}}$$

and so on.

Thus, any given unitary can arise from infinitely many different Hamiltonians.

Physicists have a special name for the eigenvalues that you get by diagonalizing a Hamiltonian. They call them **energies**. Note that they're all real, and can therefore be ordered from least to greatest:

$$\begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}$$

$$H = U U^\dagger, \quad \text{with } \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$$

To each energy λ_j , there corresponds an **energy eigenstate** $|v_j\rangle$ such that $H|v_j\rangle = \lambda_j|v_j\rangle$.

Why are they called energies?

Because they *are* energies. These values are amounts of energy that the system can have.

Quantum mechanics gives us one explanation for why the concept of “energy” arises in physics: because unitary matrices arise by exponentiating Hamiltonians, and Hamiltonians can be diagonalized and have real eigenvalues.

If we apply the unitary transformation e^{-iHt} to the energy eigenstate $|v_j\rangle$, we get

$$e^{-iHt} |v_j\rangle = e^{-i\lambda_j t} |v_j\rangle.$$

Meaning that nothing happened, apart from the state picking up a global phase (unobservable by itself) dependent on the energy.

We can write an arbitrary state as a superposition over the energy eigenstates:

$$|\Psi\rangle = \alpha_1 |v_1\rangle + \dots + \alpha_n |v_n\rangle$$

From this perspective, applying the Hamiltonian H is equivalent to doing:

$$e^{-iHt} |\Psi\rangle = \alpha_1 e^{-i\lambda_1 t} |v_1\rangle + \dots + \alpha_n e^{-i\lambda_n t} |v_n\rangle$$

This presents a terrifyingly boring picture of the history of universe! It suggests that all that has ever happened, and all that ever *will* happen, is that the various energy eigenstates of the universe pick up phases, each rotating around the unit circle at a speed proportional to its energy.

From the utter lack of interesting activity when we view the world in the energy eigenbasis, we conclude that life is a basis-dependent phenomenon.

But the above picture is extremely useful. For one thing, it suggests that we simply *define* energy as the speed at which a quantum state picks up a phase.

It's not obvious that this corresponds to the usual conceptions of energy in physics, but it turns out that it does. You'll have to go to the physics department for details though!

One thing that *is* clear, from our definition, is that “energy is conserved.” More formally: the expectation value of the energy in the state $|\Psi\rangle$, namely $\sum_j |\alpha_j|^2 \lambda_j$, stays the same over time.

Let's give a few more definitions.

The energy eigenstate $|v_1\rangle$ corresponding to the lowest energy is the **ground state**, which as we'll see plays an extremely special role. The corresponding energy λ_1 is the **ground state energy**.

The energy eigenstate $|v_2\rangle$ corresponding to the second-lowest energy is the **first excited state**. The energy eigenstate $|v_3\rangle$ corresponding to the third-lowest energy is the **second excited state**, and so forth.

One detail: if $\lambda_1 = \lambda_2$, then we get what's called a **ground state degeneracy**: there's no longer a unique ground state, but a subspace of two or more dimensions, in which every state equally minimizes the energy. We can similarly get degenerate excited states, if $\lambda_j = \lambda_{j+1}$ for some larger j . For the most part, though, we'll be able to ignore this.

The standard game plan for much of modern physics goes like this:

1. Start with the Hamiltonian H of your system.
2. Diagonalize H .
3. Get out the energy eigenstates.
4. Then, as a first guess, see if your system is just sitting in its ground state $|v_1\rangle$ doing nothing.

Why are quantum systems often found sitting in their ground states doing nothing?

Intuitively, because physical systems “like” to minimize their energy, so they tend to get into lower energy states. And the ground state, by definition, is the lowest they can go.

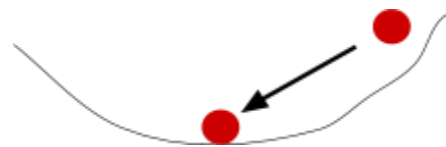
But since quantum mechanics is time-reversible, how is it even possible for a system to be “attracted” to a certain state?

Excellent question! You can thank the Second Law of Thermodynamics and the conservation of energy for this.

Note that the same question arises in classical physics, which is time-reversible too. If you leave a ball rolling around in a basin, then return a while later, you probably won't find it in an “excited state”—i.e., continuing to roll around. Whatever energy it had in its excited state, it could reach a lower energy by rolling downhill, slowing down, and giving off *heat* via *friction*.

When this happens, the kinetic energy that used to be in the ball dissipates away in the heat.

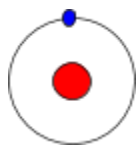
In principle, it's *possible* that the reverse could happen, and that the heat could coalesce back into the ball and make it spontaneously move. But we essentially never observe that, and the reason comes down to entropy. For all the heat to coalesce back into motion would require an absurdly finely-tuned “conspiracy,” whose probability of occurring by chance falls off exponentially with the number of particles in the ball. But the reverse process, motion dissipating into heat, requires no similar conspiracy: it only requires that our universe does contain low-entropy objects like balls. (This, in turn, can ultimately be traced back to the low entropy of the universe at the Big Bang—something that no one has satisfactorily explained in terms of anything deeper, but *we'll* be content to leave it there!)



Pretty much exactly the same story works in the quantum case, and explains why, when we find quantum systems in Nature, they're often sitting in their ground states. (Namely because, if they weren't, then their interactions with surrounding systems would tend to carry away excess energy until they were.)

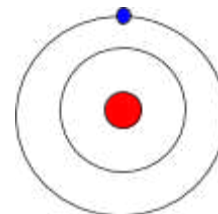
By contrast, all the quantum algorithms and protocols that we've seen in this course are examples of quantum systems that *don't* just sit in their ground states. Stuff happens; the system evolves!

We as people don't just sit in ground states either.



To give an example of these concepts that the physicists *really* love: the ground state of a hydrogen atom has the electron sitting in the lowest shell (the one closest to the nucleus).

The first excited state has the electron in the next shell up.



If the atom is in its first excited state, it can drop back down to its ground state via the electron emitting a photon. The photon carries away an amount of energy that's exactly equal to the difference between the ground and the first excited energies.

Conversely, a hydrogen atom in its ground state can jump up to its first excited state via the electron *absorbing* a photon. For this to happen, though, requires a photon to happen to hit the electron, which makes this process less likely than its reverse, for a hydrogen atom that's just sitting by itself in space somewhere. So, that's why hydrogen atoms in Nature are often found in their ground states.

Our final topic for this lecture is an extremely important operation that we can do with Hamiltonians, even though we could never do it with unitaries. Namely, we can add them!

Addition of Hamiltonians

$$H = H_0 + H_1$$

What does this mean physically?

Intuitively, it just means we've got two things going on at the same time. For example, H_0 and H_1 could correspond to two different forces acting on our system. To illustrate, at an *extremely* high level, we could write the Hamiltonian for the Standard Model of elementary particle physics as

$$H_{\text{SM}} = H_{\text{Kinetic}} + H_{\text{EM}} + H_{\text{Strong}} + H_{\text{Weak}}$$

Here H_{Kinetic} is the Hamiltonian that would act even if there were no forces (corresponding to Newton's First Law of Motion), and H_{EM} , H_{Strong} , and H_{Weak} are the Hamiltonians corresponding to electromagnetism and to the strong and weak nuclear forces respectively.

Recall that the Standard Model excludes gravity.

This isn't exactly right for all sorts of reasons, but is good enough to get across the point.

Once we're adding Hamiltonians, we immediately face a mathematical question:

If A and B are matrices, is it generally the case that $e^{A+B} = e^A e^B$?

Alas, the answer is no. Indeed, it's not hard to find a 2×2 counterexample (exercise).

On the other hand, you can check using the Taylor series definition that *if* A and B commute (that is, $AB=BA$), then $e^{A+B} = e^A e^B$ *does* hold.

We'll care about unitary transformations like $e^{-it(H_1+H_2)}$, or their counterparts with many more than two H_i 's. In particular, we'll need a way to apply these sorts of unitaries efficiently, given only the ability to apply H_1 and H_2 by themselves---even if H_1 and H_2 don't happen to commute.

Fortunately, there's a trick for this, known as **Trotterization**. The trick is to use the following approximation:

$$e^{A+B} \approx \underbrace{e^{\epsilon A} e^{\epsilon B} e^{\epsilon A} e^{\epsilon B} \dots e^{\epsilon A} e^{\epsilon B}}_{\frac{1}{\epsilon} \text{ times}}$$

This basically means that we can achieve the same effect as A and B occurring simultaneously, by repeatedly switching between doing a tiny bit of A and a tiny bit of B.

We won't do it here, but it's possible to prove that the approximation improves as ϵ decreases, becoming an exact equality in the limit $\epsilon \rightarrow 0$.

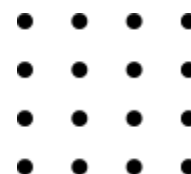
This is important for the question of how to simulate a real-world quantum system using a quantum computer. Indeed, the straightforward approach is just:

1. Discretize all the degrees of freedom (positions, momenta, etc.) that aren't already discrete.
2. Write the total Hamiltonian H acting on the system as a sum of "simple" terms (say, terms that act on only 1 or 2 particles at a time).
3. Trotterize H, in order to simulate it by a product of "simple" unitary transformations.

To flesh this out a bit more, we ought to say *something* about what the Hamiltonians of real physical systems tend to look like, at the level of abstraction relevant for this class.

Let's model the universe as a gigantic lattice of qubits, say in 2 or 3 dimensions (hey, it *is* a quantum computing class!). In that case, the total Hamiltonian that acts on the qubits can typically be written

$$H = \sum_j H_j + \sum_{j \sim k} H_{jk}$$



Here H_j is a Hamiltonian that acts only on the qubit j , and trivially on all the others, for example:

$$H_1 = \begin{bmatrix} h_1 & h_2 \\ h_3 & h_4 \end{bmatrix} \otimes \begin{bmatrix} 1 & \\ & 1 \end{bmatrix} \otimes \dots \otimes \begin{bmatrix} 1 & \\ & 1 \end{bmatrix}$$

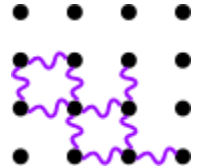
We can achieve this by, for each qubit, tensoring a 2×2 Hamiltonian on that qubit with the identity on all the other qubits--similar to what's done with unitary gates.

Meanwhile, H_{jk} is a Hamiltonian that acts on the neighboring qubits j and k , for example:

$$\begin{bmatrix} 0 & & & \\ & 0 & & \\ & & 0 & \\ & & & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & \\ & 1 \end{bmatrix}^{\otimes n-2}$$

This means that each qubit “talks” only to its immediate neighbors in the lattice, but evolving the Hamiltonian over time gives us effects that can propagate arbitrarily far.

$$e^{-iHt} = \sum_{k=0}^{\infty} \frac{(-iHt)^k}{k!}$$



As soon as we've written this, though, we face a puzzle:

Won't this lead to faster-than-light travel?

Indeed, even when t is arbitrarily small, one can check that the unitary matrix e^{-iHt} will generically contain effects from *every* qubit in the lattice to every other one. Granted, the magnitude of these effects will fall off exponentially with distance, but causality demands that there should be literally *zero* effects propagating across the lattice faster than light.

So what's the resolution? Basically, it's just that the picture we're using comes from non-relativistic quantum mechanics, so it yields a good approximation only if the relevant speeds are small compared to the speed of light. When the speeds are larger, we need the framework of quantum field theory, which *does* entail that faster-than-light influences are exactly zero.

OK, now we're ready to set things up for the next lecture. Suppose that H , a Hamiltonian acting on n qubits, is the sum of many “simple” Hamiltonians acting on a few qubits each:

$$H = H_1 + \dots + H_m.$$

Because H is a $2^n \times 2^n$ matrix, figuring out its ground state (or ground states) by brute force could be extremely time-consuming. Which leads to a question:

If I know the ground state of the H_j 's individually, can I combine them in some simple way to get the ground state of H itself?

Alas, the answer is almost certainly “no.” More precisely, we claim that finding the ground state of a Hamiltonian of this form is an **NP-hard** problem. To prove this, we'll show how to take any instance of the famous 3SAT problem, and encode it into the ground state problem. Thus, suppose we have a Boolean formula in n variables,

$$\varphi(x_1, \dots, x_n) = c_1 \wedge \dots \wedge c_m$$

where each clause c_i acts on at most 3 variables.

We can now define a Hamiltonian H as follows: H will act on n qubits, and will contain one term H_i for each clause c_i . The term H_i will act on at most 3 qubits, corresponding to the bits acted on by c_i , and will impose an “energy penalty,” say of 1, if and only if the qubits are set in such a way that c_i is violated. For example, you could encode the clause $(v_1 \vee v_2 \vee v_3)$ as the Hamiltonian

$$h = \begin{bmatrix} 1 & & & & & & \\ & 0 & & & & & \\ & & 0 & & & & \\ & & & 0 & & & \\ & & & & 0 & & \\ & & & & & 0 & \\ & & & & & & 0 \\ & & & & & & & 0 \end{bmatrix}$$

This Hamiltonian will yield 0 (no energy penalty) unless all three qubits are set to 0, which corresponds to the case where the clause is not satisfied.

It’s not hard to see that, to minimize the energy for such a Hamiltonian, without loss of generality you just set all the variables classically, and see which setting leads to the smallest energy.

In other words: you solve 3SAT!

In particular, H ’s ground-state energy will be 0 if φ is satisfiable, or positive if φ is unsatisfiable.

So not only is this problem **NP**-hard, it’s **NP**-hard for reasons that have nothing to do with quantum mechanics.

Could this observation somehow let Nature solve **NP**-hard problems in polynomial time? For more, tune in next time!

Lecture 26, Tues April 25: Adiabatic Algorithm

At the end of the last lecture, we saw that the following problem is **NP**-hard:

Given as input an n -qubit Hamiltonian H of the special form $H = H_1 + \dots + H_m$, with each H_i acting on at most 3 of the n qubits, estimate H 's ground state energy.

This is a quantum generalization of the 3SAT problem, one that arises very naturally in condensed matter physics. Building a complicated Hamiltonian by summing local terms could also be seen as somewhat analogous to building a complicated quantum circuit by composing 1- and 2-qubit gates.

But how do you “give” someone a Hamiltonian like that?

Providing the full $2^n \times 2^n$ Hermitian matrix would be wasteful. Instead, you can simply list the local terms H_1, \dots, H_m (to some suitable precision), together with the qubits to which they're applied.

Is this problem **NP**-complete?

Since we know it's **NP**-hard, what we're asking here is whether it has polynomial-time verification. In other words, when we claim that the ground-state energy of some Hamiltonian is at most (say) 5, can we prove it by giving a short witness?

It turns out that we can---but as far as anyone knows today, only by giving a *quantum* witness! A quantum witness that works is simply the n -qubit ground state itself.

Thus, the Local Hamiltonians problem is not known to be in **NP**: it's only known to be in the quantum analogue of **NP**, which for reasons we won't go into is called **QMA** (Quantum Merlin-Arthur)

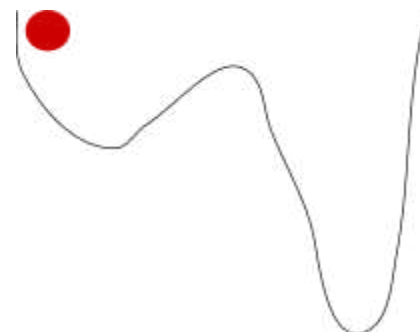
An important theorem from around 1999 says that Local Hamiltonians is actually *complete* for **QMA**, just like 3SAT is complete for **NP**.

For the specific Hamiltonian H we constructed in the last lecture---the one that encoded 3SAT---there *would* be a short classical witness, because that H was a diagonal matrix, so its ground state is always just a classical basis state. But what if H is non-diagonal, and its ground state is some complicated entangled state?

So if natural quantum systems like to settle into their ground states, and if finding the ground state is **NP**-hard, does this mean that we could use quantum systems to solve **NP**-hard problems in polynomial time “naturally”?

People talked about such questions even before the concept of quantum computing was in place.

But there's a serious problem: it's not *always* true that natural quantum systems quickly settle into their ground states. And starting from hard instances of 3SAT might produce complicated and exotic Hamiltonians, far from physicists' usual experience. Those complicated Hamiltonians might often be ones for which it's hard to reach the ground state.



In the hillside (above), will the ball get to the point that minimizes its gravitational potential energy?

Probably not anytime soon!

If we wait a million years, maybe a thunderstorm will push the ball up over the hill in the middle. But for the foreseeable future, it's much more likely for the ball to rest at the **local optimum** on the left.

In hard real-world optimization problems, you'd have a very bumpy 1000-dimensional landscape or whatever with plenty of local optima to get trapped in. You might wonder if quantum computing could help us wade through these local optima—and that certainly seems plausible. In fact, the hope that this would be true was a central starting point for today's topic:

The Adiabatic Algorithm

This is the last quantum algorithm we'll cover in this course. To this day, it remains a source of significant research and discussion.

There's a physics theorem from the 1920s called the Adiabatic Theorem:

"Adiabatic" is a term from thermodynamics; we won't really need to understand what it means in order to discuss the adiabatic algorithm.

Anyway, here's what the theorem says: suppose you prepare the ground state of some initial Hamiltonian H_i , with H_i being applied to that ground state. You then slowly and continuously change H_i into some final Hamiltonian H_f , until at the end you're applying exactly H_f . Then provided that the transition was slow enough, you'll end up at (or extremely close to) the ground state of H_f .

Assuming a few fine-print conditions that we won't go into.

So, gradual change from one ground state brings us to another ground state.

In the minds of Farhi, Goldstone, and other physicists, this suggested the following plan to solve NP-hard problems using Hamiltonians.

We know that $H = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$ is a one-qubit Hamiltonian with $|+\rangle$ as its unique ground state.

Its eigenstates are $|+\rangle$ and $|-\rangle$.

The energy of $|+\rangle$ is 0 and the energy of $|-\rangle$ is 2, so $|+\rangle$ is the ground state.

So we can create an initial Hamiltonian H_i (where here the i means "initial") by applying H to each qubit individually:

$$H_i = \sum_{i=1} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}_i$$

The state $|+\rangle^{\otimes n}$:

- is easy to prepare in a quantum computer

- would stay put forever if we continued to apply H_i forever.

But then, we gradually change H_i to another Hamiltonian H_f , which encodes some n -bit 3SAT instance that we'd like to solve:

Parameterizing time as $t \in [0, 1]$

$$H_i = \sum_{i=1} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}_i \quad \xrightarrow{\hspace{2cm}} \quad H_f = \sum_{i=1}^m h_i$$

In the simplest version, at any point in time t , we apply the Hamiltonian

$$H_t = (1-t)H_i + tH_f$$

Since each H_t is a sum of terms acting on a few qubits only, we can efficiently apply these H_t 's using Trotterization (discussed in the last lecture).

If everything goes according to plan, we'll end up in the ground state of H_f ---in other words, the optimal solution to our 3SAT instance (!!).

So what's the catch? Well, the crux of the matter is that the transition from H_i to H_f must be sufficiently slow.

How slow?

To answer that question, let's plot the eigenvalues of H_t as a function of the time parameter t (example shown on the right).

H_t could have as many as 2^n eigenvalues (all real numbers). But we're especially interested in the **smallest** eigenvalue (the ground energy) and the **one right above that** (the first excited energy).

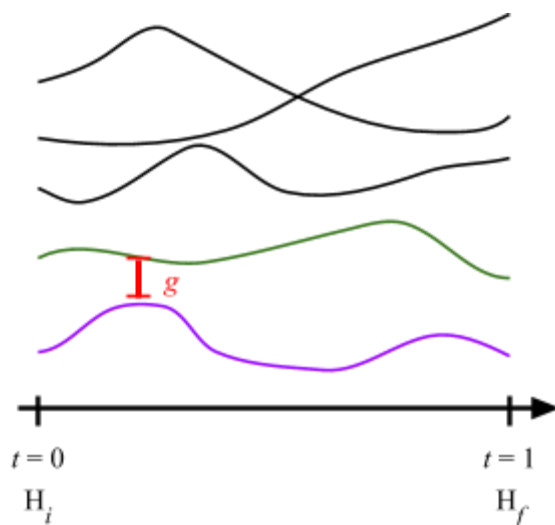
Our goal is to stay in the ground state throughout.

The eigenvalues, also called **energy levels**, can change continuously over time.

Sometimes two energy levels can even cross each other. Physicists call that a **level crossing**.

When they *almost* cross, it's an **avoided level crossing**.

If the two lowest energies cross each other, then we leave the ground state. Even if the two lowest energies *nearly* cross each other, there's a significant risk that we'll leave the ground state. The closer together the two lowest energies get, the slower we have to run the algorithm to avoid confusing them.



We define the **Minimum Eigenvalue Gap**, g , as the smallest the gap ever gets between the first excited energy and the ground energy, as we vary the time t .

$$g = \min_{t \in [0, 1]} (\underbrace{\lambda_2^{(t)}}_{\text{First excited energy}} - \underbrace{\lambda_1^{(t)}}_{\text{Ground energy}})$$

In the statement of the adiabatic theorem, one of the “fine print” conditions that we didn’t mention earlier is $g > 0$.

g turns out to be a crucial quantity for determining how long the adiabatic algorithm will take to solve a given problem instance. Roughly speaking: in order to ensure that we remain in the ground state throughout, as we pass the value of t where the gap is g , we need to vary t at a speed of only $\sim g^2$, which takes $\sim \frac{1}{g^2}$ computation steps.

So in particular: *if* we could show that g was always lower-bounded by $1/n^{O(1)}$ for 3SAT (or some other **NP**-complete problem), then we’d get **NP** \subseteq **BQP**: quantum computers would be able to solve all **NP** problems in polynomial time.

An early paper about this, which was published in *Science* in 2001, basically said, “Well... this looks like it could work.”

Note: In reality we’re approximating the Hamiltonians with discrete-time unitary transformations, so we’d end up with something *close* to the ground state of H_f , not the ground state itself. But that would still be good enough to solve our **NP**-hard problem.

So the question boils down to:

What is the behavior of g , the minimum spectral gap, for the problem instances that we want to solve?

Farhi once went to an expert in condensed matter physics and said: this is the system we’re looking for, do you think that g will decrease polynomially or exponentially as a function of n ?

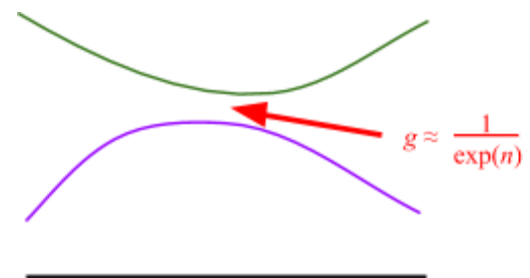
The expert says: I think it will decrease exponentially.

That’s not the answer Farhi wants to hear. So Farhi asks: why? What’s the physical reason?

After thinking about it some more, the expert responds: because if it only decreased polynomially, then your algorithm would work.

What emerged after a couple of years is that, for hard instances of 3SAT (or even 2SAT, for that matter), the ground energy and the first excited energy often *do* get exponentially close to touching. At the point where that happens--i.e., the avoided level crossing--you’d need to run the algorithm for an exponential amount of time in order to remain in the ground state, and thereby solve your SAT instance.

But the story doesn’t end here. The physicists regrouped and said, “Okay, so maybe this technique doesn’t *always* work, but it might still give big advantages for *some* types of optimization problems!”



By now there's been lots of research on classifying the types of solution landscapes where the adiabatic algorithm performs well and those where it performs poorly. Some encouraging results came from:

(Farhi, Goldstone, Gutmann 2002)



These authors constructed fitness landscapes that had a global minimum at the bottom of a wide basin, but also a tall thin spike blocking the way to that minimum. Starting from the far left, a classical algorithm based on Steepest Descent would get stuck forever at the base of the spike (i.e., at a local minimum), and would never reach the global minimum.

OK, but before we examine the performance of the adiabatic algorithm on this sort of landscape, shouldn't we first look at better classical algorithms?

Indeed: Simulated Annealing is a better classical optimization algorithm--one that, much like the adiabatic algorithm, will always *eventually* reach the global minimum, if you run it for long enough.

In fact, simulated annealing can be thought of as a classical counterpart to the adiabatic algorithm.

The basic idea of **Simulated Annealing** is to evaluate the fitness function around the current point and then:

- Make a move that improves fitness (Most of the time)
- Make a move that worsens fitness (Some of the time)

The probability of making a move that worsens fitness is time-dependent, and decreases as time goes on.

Trading off exploration for exploitation.

The name “simulated annealing” comes from a 7000-year old technology. **Annealing** is the process of making a metal stronger by heating it up and then slowly cooling it. This gives the atoms in the metal a chance to bounce around, escape from a local optimum that might be causing brittleness, and then slowly settle into a better optimum.

On the fitness landscape with the spike (pictured above), simulated annealing would *eventually* get over the spike despite how energetically unfavorable it is.

However, if the spike is tall enough, then it would take an exponential amount of time.

We'd be waiting for the thunderstorm, so to speak.

On the other hand, if the spike is thin enough, then Farhi et al. showed that the adiabatic algorithm can get over it in only polynomial time. It does so by exploiting a well-known quantum phenomenon called **tunneling**.

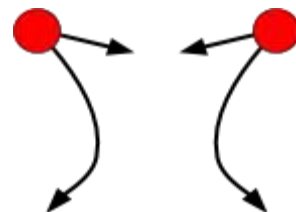
Popular articles explain tunneling by saying that a quantum particle can get through barriers that a classical particle could never get through. But it would probably be more accurate to say: “in places that a classical particle would need exponential time to get through, *sometimes* a quantum particle can get through in polynomial time.”

In terms of interference, we can say, “The paths that involve the particle not going over the spike interfere destructively and cancel each other out, leaving only paths where the particle does get over it.”

The phenomenon of tunneling is important in many places in physics. For one thing, it’s why the sun can shine.

Nuclear fusion requires hydrogen atoms to get super close for them to realize that it’s energetically favorable for them to fuse. The trouble is, when they’re not quite so close, they also have strong repulsion (because both nuclei are positive).

When quantum mechanics came along, it explained that while the energy barrier would prevent fusion classically, it still happens because the nuclei are able to tunnel past the barrier.



Anyway, the 2002 paper of Farhi et al. was good news for the prospects of using the adiabatic algorithm, but tunneling only helps if the spike is sufficiently thin.

Since then, we’ve learned more about the types of fitness landscapes for which the adiabatic algorithm is expected to help.

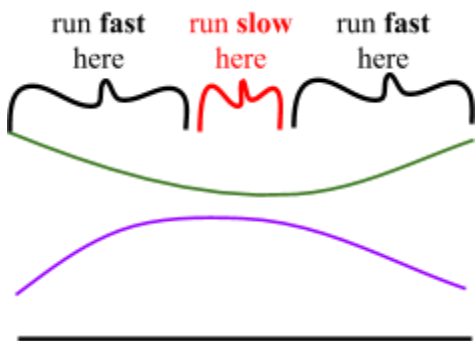


In a landscape like the one pictured on the left, simulated annealing and the adiabatic algorithm would both have trouble and would both take an exponential amount of time.

Or consider the fitness landscape on the right (we can only draw 1 dimension, but imagine that all of the 2^n solutions in an n -dimensional hypercube have equal values except for the one good solution). This would also take exponential time for both simulated annealing and the adiabatic algorithm to traverse.



We actually already know this by the BBBV Theorem---because in this case, we’re effectively just querying a black box in an attempt to find a unique marked item.



It turns out that if you’re clever about how you run the adiabatic algorithm, you can achieve the Grover speedup in the case above, but not anything faster.

Indeed, the BBBV Theorem tells us that $\Omega(2^{n/2})$ steps are needed, using the adiabatic algorithm or any other quantum algorithm.

What’s cool is that just by knowing BBBV (without any physics), you can say that this decrease has to be exponential.

Just like the expert who Farhi consulted was alluding to with his wisecrack, physicists can use knowledge from quantum algorithms to learn new things about spectral gaps.

OK, here's a subtler question:

Suppose the adiabatic algorithm had worked to solve 3SAT in polynomial time. Would that have violated the BBBV Theorem?

It turns out the answer is no.

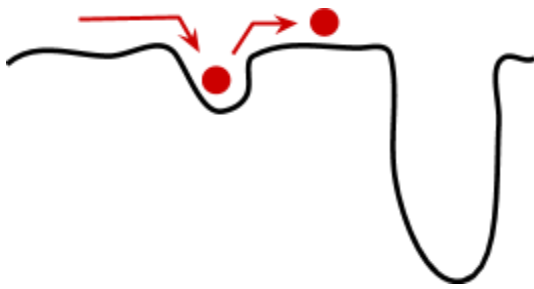
The BBBV Theorem applies only to **black-box** search--which, in the context of the adiabatic algorithm, would mean a diagonal Hamiltonian like the one on the left.

While the Hamiltonian encoding a 3SAT instance (right) is also diagonal, it contains richer information than the black box considered by BBBV. In particular, it encodes not merely whether each possible solution is satisfying or unsatisfying, but the *number* of clauses that it violates. And a 2002 paper of van Dam, Mosca, and Vazirani showed that that information, alone, is enough to reconstruct the 3SAT instance in polynomial time---and hence also to *solve* the instance in polynomial time, if we assumed (for example) that **P=NP**! This means that there's no hope of proving a black-box lower bound à la BBBV in this setting.



We also know classical algorithms that can solve 3SAT in less than $2^{n/2}$ time---indeed, about $O(1.3^n)$. This is another way of seeing that the BBBV Theorem can't encompass everything that it's possible to do on 3SAT.

OK, let's consider one more type of fitness landscape. A funny thing happens with landscapes like the



one pictured below: simulated annealing gets into the local minimum, but the algorithm then escapes it, crosses the plateau, and reaches the global minimum in polynomial time. Meanwhile, the adiabatic algorithm just keeps returning to the local minimum, and takes exponential time to reach the global minimum!

Sometimes a classical algorithm performs better.

But there's even a further problem with establishing quantum speedups for adiabatic optimization, which is that "classical computing is a moving target." A classical computer is not limited to simulated annealing or any other specific algorithm. Thus, even if we established that the adiabatic algorithm was exponentially faster than simulated annealing for some class of "real-world" fitness landscapes, we'd still need to compare against *other* classical algorithms, which might exploit detailed knowledge about the landscapes in question. Particularly relevant here is Quantum Monte Carlo (QMC)---which, despite its name, is a quantum-inspired algorithm that runs on a classical computer, and which is widely used in many-body physics. We won't explain QMC in any detail, but will simply say that, even in the artificial cases where the adiabatic algorithm exponentially outperforms simulated annealing, recent work suggests that QMC can typically match the adiabatic algorithm's asymptotic performance classically (albeit, often with a much larger constant prefactor).

So, what about the fitness landscapes that arise in real-world, industrial applications? Do there or don't there exist any that the adiabatic algorithm can traverse exponentially faster than any classical algorithm?

The truth is, we really don't know yet. Reaching a consensus on this might require building a scalable quantum computer and then testing the algorithm out!

Which brings us to our final point about the adiabatic algorithm. We've talked about implementing the adiabatic algorithm on a conventional, gate-based quantum computer, by using Trotterization to approximate Hamiltonians by discrete sequences of gates. But you might complain that that approach seems roundabout. Since the underlying physics that describes our qubits is based on Hamiltonians anyway, why not just directly map the adiabatic algorithm onto the continuous-time evolution of the qubits, and skip the Trotterization part?

Indeed, the adiabatic algorithm could be seen not only as an algorithm, but also as a proposal for the physical implementation of quantum computers. An important 2004 result of Aharonov et al. says that adiabatic quantum computers would be universal for quantum computation: that is, able to solve all **BQP** problems efficiently.

There's a venture-capital-backed startup called D-Wave that's been building special-purpose devices to implement a noisy approximation to the adiabatic algorithm (called quantum annealing), using physical Hamiltonians themselves. D-Wave's latest model has about 2000 superconducting qubits. You can encode an optimization problem of your choice onto their chip, by choosing the interaction Hamiltonian for each pair of neighboring qubits.

D-Wave was all over the press because they actually sold a few of their machines, to companies like Google and Lockheed Martin, and were notorious for claiming that quantum computing is “already useful in practice.” They were even on the cover of *Time* magazine!

Professor Aaronson has been to D-Wave’s headquarters. Funnily enough, their machine is *literally* a room-sized black box (most of the hardware inside the box is devoted to cooling; the actual qubits are on a chip no larger than an ordinary computer chip).

So what’s the verdict? Experimental data shows that the D-Wave device is indeed able to solve optimization problems encoded in its special format, at a speed that’s often competitive with the best known classical algorithms. Unfortunately, results over the past five years do not clearly show any quantum *speedup* over the best classical algorithms.

Why not?

Roughly speaking, there are three main possible causes for the lack of speedup on D-Wave’s current devices. As far as we know today, the truth might be any combination of them.

1) *Inherent limitations of the adiabatic algorithm.*

Even if we had a perfect quantum computer, running at absolute zero, the minimum spectral gaps might simply be exponentially small for almost all interesting optimization problems---in which case, the adiabatic algorithm could only provide limited speedups, even in theory.

2) *Limitations in the quality of D-Wave’s qubits.*

Even if the minimum spectral gap were inverse-polynomial, it still probably wouldn’t be *constant*: that is, it would presumably shrink to zero as a function of the input size. And this is a problem for the following reason. It turns out that, if the *temperature* of the qubits exceeds the minimum spectral gap, then we’ll typically see level crossings even if we run the adiabatic algorithm at the “right” speed.

The D-Wave qubits are cooled to about 10 milliKelvin. By normal standards, that sounds extremely cold, but it might not be cold *enough* to avoid level crossings on instances of interesting sizes! What one really wants here is absolute zero---but of course, the cost would increase enormously as one approached that, eventually becoming prohibitive.

3) *Stoquastic Hamiltonians.*

A Hamiltonian H is called stoquastic if all its off-diagonal terms are real and non-positive. One can show that, if H is stoquastic, then H has a ground state involving nonnegative real amplitudes only. It turns out that, for reasons stemming from that fact, stoquastic Hamiltonians are often much easier than arbitrary Hamiltonians to simulate on a classical computer. For example, the QMC algorithm mentioned earlier tends to work extremely well for approximating the ground states of stoquastic Hamiltonians, and less well for non-stoquastic Hamiltonians.

Unfortunately, D-Wave’s hardware is currently limited to applying stoquastic Hamiltonians only. Thus, even if quantum annealing were able to yield an exponential speedup at a fixed nonzero temperature (like 10 milliKelvin), it might be that it could only do so with non-stoquastic Hamiltonians.

This sounds pretty bad! Given, especially, the temperature issue, why is anyone optimistic that quantum computing could scale even in principle? We'll see why in the next lecture, when we explore the basics of quantum error-correction: a technique that D-Wave is not currently using, but that many researchers expect will ultimately be necessary for scalable QC.

Lecture 27, Thurs April 27: Quantum Error

Correction

At the end of the last lecture, we discussed some of the difficulties with achieving a quantum speedup using currently available quantum computing devices, like D-Wave's. We saw how D-Wave's devices are cooled to 10 milliKelvin, but even that might be too hot, and lead to too much decoherence and error! (Which, in the setting of adiabatic QC, shows up mostly as unwanted level crossings.)

You might have wondered:

If even 10 milliKelvin isn't cold enough---if nothing short of absolute zero and perfect isolation seem to suffice---then why should building a scalable quantum computer (one that achieves actual quantum speedups) be possible at all?

We need to separate two issues. First, there's the "engineering challenge" of building a scalable QC. Everyone agrees that, at a bare minimum, it will be *staggeringly hard* to achieve the required degree of isolation for thousands or millions of qubits, when those qubits also need to interact with each other in a carefully choreographed way. Maybe various practical problems will prevent human beings from doing it in the next 50 or 100 years. Maybe it will be too expensive. Theory alone can't answer such questions.

But then separately, there's the question of whether anything prevents scalable QC *even in principle*. Of course, if quantum mechanics itself were to break down, that could certainly prevent QC---but it would also represent a much more revolutionary development for physics than a "mere success" in building QC!

So, short of a breakdown of QM, on what grounds have people argued that scalable QCs aren't possible?

1) *Large entangled states are inherently fragile, so they might be impossible to maintain.*

If only one qubit in the computer decoheres, then the entire system could decohere.

If only one qubit in a "Schrödinger cat" type state leaks out into the environment, the quantum coherence between the $|\text{Alive}\rangle$ and $|\text{Dead}\rangle$ components is destroyed.

2) *Applying unitary gates may produce lots of errors, which will snowball over time.*

Maybe your Hadamard gate rotates by 46° instead of 45° . If so, then over many applications of the gate, the errors would pile up.

If you pick up a CS textbook from the 1950's, you'll see plenty of discussion surrounding "analog vs digital computers". The disappearance of analog computers from the scene had much to do with the difficulty of correcting continuous errors. Is a quantum computer simply another kind of analog computer?

3) *The No-Cloning Theorem limits us.*

Classically we deal with error by repetition (which implicitly means repeated measurements), but because of the No-Cloning Theorem, it seems that we can't do the same with quantum computers.

However, there were fundamental discoveries in the mid-1990s that addressed all three of these concerns, and that convinced most physicists and computer scientists that, short of some revolutionary change to known physics, the difficulties of building a scalable QC are “merely” difficulties of engineering, and not of principle. The first of these discoveries was the theory of...

Quantum Error Correction

Before discussing quantum error correction, let’s briefly review how error correction works classically.

You could take a whole course about this subject.

We’ll do the 5-minute version here.

The 3-bit Repetition Code

is a way to encode one logical bit using 3 physical bits.

$\bar{0}$, the logical 0 is encoded as “000”

and $\bar{1}$, the logical 1 is encoded as “111”.

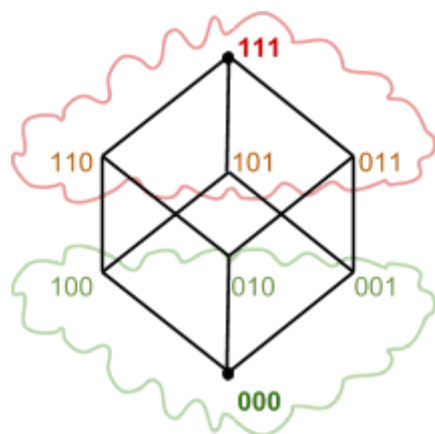
We claim that this code lets us both *detect* and *correct* an error in any one physical bit.

We do **error detection** by checking whether $x = y = z$. Assuming at most one incorrect bit, we’ll be able to identify it. If we detect an error (say in x), we can do **error correction** by setting $x := \text{MAJ}(x,y,z)$.

As an exercise, show that any code that can both detect and correct a single bit-flip error must use at least 3 bits. By contrast, if we just want to detect a single bit-flip error, and not correct it, show that 2 bits suffice.

More sophisticated codes are able to encode many logical bits at once, not just a single bit, while detecting and correcting a large number of errors (say, any 10% of the physical bits being flipped). In this lecture, though, we’ll focus on encoding just a single bit (or qubit) and protecting against just a single error, in order to get the main conceptual points across.

Here’s a useful geometric picture for why the 3-bit repetition code works.



Essentially, the code simply picks two points on the Hamming cube $\{0,1\}^3$ that are maximally far from each other, and declares one to be the encoding of 0 and the other to be the encoding of 1.

000 and 111 can each get corrupted to any point in their respective clouds, but since the two clouds don’t overlap we’re able to correct the error.

We’ll seek to replicate this behavior in the quantum case.

In the early days of classical computing there were skeptics who doubted that the technology would ever scale. “Obviously,” they said, “once the machine has enough vacuum tubes, some of them will fail, and that will cause the entire computer to fail.”

John von Neumann was annoyed by this line of reasoning, so he went off and proved...

The Classical Fault-Tolerance Theorem

Von Neumann showed that, even given logic gates (like AND, OR, and NOT) every one of which fails with some independent probability ϵ , as long as ϵ is sufficiently small, it's possible to build a reliable circuit for any Boolean function f . Moreover, the circuit only needs to be a small factor larger than a circuit for f built of perfect gates. Informally, you can build a reliable computer out of unreliable parts. Admittedly, if the output of the circuit is just a single bit, then that bit can't possibly take the correct value with probability greater than $1-\epsilon$ ---for what if the very last gate is erroneous? However, one way to deal with this issue is to let the output be a long string of bits, which is then processed by (say) a simple majority to get the value of f . This final majority computation is assumed to be error-free.

How did von Neumann prove the classical fault-tolerance theorem? We won't go through all the details, but the basic idea is to use the 3-bit repetition code recursively---pushing the probability of a catastrophic error down further and further with a majority of majorities (on 9 bits), a majority of majorities of majorities (on 27 bits), and so on.

Now, this seems to raise a conceptual puzzle:

Our error-correction circuits will themselves be subject to error. So when we compute these recursive majorities, won't we simply be introducing more errors than we correct?

Fortunately, von Neumann found that, as long as the physical error probability ϵ is small enough, each round of error-correction will be a “net win,” making things better rather than worse.

Anyway, von Neumann's whole idea ended up being mostly unnecessary when transistors replaced vacuum tubes, since transistors err with such minuscule probabilities (in your entire computer, with its billions of transistors, perhaps one transistor will output a wrong result per year when it's hit by a cosmic ray).

Could such a thing happen with quantum computing: that is, a “QC transistor,” which implements 1- and 2-qubit gates so reliably that error-correction is then superfluous?

Maybe! In the last lecture, we'll discuss topological quantum computing, which some physicists think could get us part of the way towards this dream. For now, though, we seem to be stuck in von Neumann's situation, with quantum gates that *do* have significant probabilities of error.

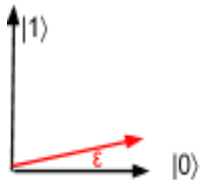
Since we discussed D-Wave in the last lecture: a potentially fateful decision that D-Wave made, early on, was that they weren't going to do any error correction. However, even D-Wave might now be coming around to the view that some degree of error correction is necessary.

Crucially, *until* you get over the hurdle of error correction, it may not look like your QC is doing much of anything useful. This is the main reason why progress in experimental quantum computing has

often seemed slow (with the world record for Shor's algorithm remaining the factorization of 21 into 3×7 , etc.). Some people believe that practically important speedups will come only after we've overcome this hurdle.

You might worry that nothing like the 3-bit repetition code could possibly work in the quantum case, because quantum errors form a continuum: it's not a yes-or-no question whether an error has happened. So for example, we might get the error

$$|0\rangle \rightarrow \sqrt{1-\epsilon^2}|0\rangle + \epsilon|1\rangle.$$



It's said, only half-jokingly, that "80% of the work of building a quantum computer is reliably implementing the identity transformation," to keep the qubits steady.

Early in this course, though, we saw that the Watched Pot Effect (also called the Quantum Zeno Effect) was a way to keep a drifting qubit in check.

The trick is just to keep measuring the qubit in the $\{|0\rangle, |1\rangle\}$ basis.

If you get $|0\rangle$: "Was there an error?" "Well, *now* there's not!"

It's like the joke where someone calls 911 to report a dead body, and the operator asks the caller to check if the person is actually dead. Gunshots are heard; then the caller says, "OK, now what?"

If you get $|1\rangle$, correct it to $|0\rangle$.

But the Watched Pot Effect only solves the problem of quantum error-correction if

- (1) the *only* thing we're worried about is continuous drift (rather than, e.g., discrete bit-flips), and
- (2) we know a basis in which our qubit is supposed to be one of the basis vectors.

So how can we go further?

If we only needed to correct bit-flip errors, we could just use the obvious quantum analogue of the 3-bit repetition code, namely:

$$|\overline{0}\rangle \rightarrow |000\rangle \text{ and } |\overline{1}\rangle \rightarrow |111\rangle.$$

But bit-flip errors aren't the only things we're worried about!

For an example of what else could go wrong, let's look at the encodings of $|+\rangle$ and $|-\rangle$ under the above 3-bit repetition code:

$$|\overline{+}\rangle \rightarrow \frac{|000\rangle + |111\rangle}{\sqrt{2}} \text{ and } |\overline{-}\rangle \rightarrow \frac{|000\rangle - |111\rangle}{\sqrt{2}}$$

Question: How many qubits of $|\overline{+}\rangle$ do we need to act on to convert it to $|\overline{-}\rangle$?

Answer: Only one! (It suffices to apply a phase gate to any qubit.)

I.e., we don't have separate clouds.

But you might think the problem is even worse. Suppose we *did* manage to design a code that could correct both bit-flip errors and phase-flip errors. Even then, aren't there infinitely many *other* ways for a qubit to err? So won't we need to add an infinite amount of additional redundancy to our code?

Here's where a little piece of magic comes in and saves us. It turns out that, if a quantum error-correcting code protects against both bit-flip and phase-flip errors, then that's automatically enough to protect against *all possible* 1-qubit errors.

Why?

Without loss of generality, let's assume that we have an error on the first qubit of the entangled state $|\Psi_0\rangle$:

$$|\Psi_0\rangle = \alpha|0\rangle|v\rangle + \beta|1\rangle|w\rangle$$

A bit-flip error on the first qubit would result in

$$|\Psi_1\rangle = \alpha|1\rangle|v\rangle + \beta|0\rangle|w\rangle$$

A phase-flip on the first qubit would result in

$$|\Psi_2\rangle = \alpha|0\rangle|v\rangle - \beta|1\rangle|w\rangle$$

And both would result in

$$|\Psi_3\rangle = \alpha|1\rangle|v\rangle - \beta|0\rangle|w\rangle$$

The key observation is now that these four states constitute an orthogonal basis for the 4-dimensional subspace of all possible states that you could reach from $|\Psi_0\rangle$ by transformations on the first qubit.

This is extremely similar to Superdense Coding: in both cases, entanglement doubles the number of independent transformations that we can apply to a given qubit, leaving a perfect record of the fact that the transformation was applied, from 1 to 2, .

So, the simplest possible goal of quantum error-correction is now as follows: *assuming* that the error was in the first qubit only, get us back to $|\Psi_0\rangle$ from wherever we might happen to be in this 4-dimensional "error subspace."

Measuring all the qubits would be a really bad idea--since even if that told us where we were in the subspace, it would be a "pyrrhic victory" that destroyed our quantum state in the process.

Instead, maybe we can do a measurement that just projects onto one of the four basis vectors we saw.

If the outcome is $|\Psi_0\rangle$, we do nothing.

If the outcome is $|\Psi_1\rangle$, we get back to $|\Psi_0\rangle$ by doing a bit-flip.

If the outcome is $|\Psi_2\rangle$, we get back to $|\Psi_0\rangle$ by doing a phase-flip.

If the outcome is $|\Psi_3\rangle$, we get back to $|\Psi_0\rangle$ by doing both.

But what's a code that can detect and correct a bit-flip or a phase-flip error on *any* qubit?

Well, for starters, if we just wanted to detect and correct phase-flip errors, we could do so using the following code, namely "the 3-qubit repetition code except in the Hadamard basis":

$$|\overline{+}\rangle \rightarrow |+\rangle|+\rangle|+\rangle$$

$$|\overline{-}\rangle \rightarrow |-\rangle|-\rangle|-\rangle$$

OK, but just like the other 3-qubit code failed to protect against phase-flip errors, this code fails to protect against bit-flip errors. For by linearity,

$$|\overline{+}\rangle + |\overline{-}\rangle = |\overline{0}\rangle$$

$$|\overline{+}\rangle - |\overline{-}\rangle = |\overline{1}\rangle,$$

and this means (if you work it out) that applying a bit-flip to any qubit can change $|\overline{0}\rangle$ to $|\overline{1}\rangle$ or vice versa.

This observation brings us to our first serious quantum error-correcting code:

The Shor 9-qubit code (1995)

yes, it's the same Shor

Shor's proposal was simply to *combine* the two codes we've seen. He encodes each logical qubit as a 3×3 square of physical qubits, in such a way that each row corresponds to a 3-qubit repetition code to protect against bit-flip errors, and each column corresponds to a 3-qubit repetition code to protect against phase-flip errors.

$$\text{Thus we encode } |\overline{0}\rangle \text{ as } \left(\frac{|000\rangle + |111\rangle}{\sqrt{2}} \right)^{\otimes 3}$$

$$\text{and } |\overline{1}\rangle \text{ as } \left(\frac{|000\rangle - |111\rangle}{\sqrt{2}} \right)^{\otimes 3}$$

We claim that this code lets us detect and correct a bit flip *or* a phase flip (and hence, any possible error) on any one of the 9 qubits.

Why does this detect and correct bit-flip errors?

We just need build a little quantum circuit that checks whether all 3 of the qubits in a given row have the same value ($|0\rangle$ or $|1\rangle$), and if they don't, sets the wayward qubit equal to the majority of all 3 qubits in the row. We then apply that circuit to each of the 3 rows separately.

More interestingly...

Why does this code *also* detect and correct phase-flip errors?

Because we can build a quantum circuit that computes the relative phase between $|000\rangle$ and $|111\rangle$ within each row (+ or -), checks whether all 3 phases have the same value, and if they don't, sets the wayward phase equal to the majority of all 3 phases.

You can check that both of the operations above work, not just for $|\overline{0}\rangle$ and $|\overline{1}\rangle$, but for arbitrary superpositions of the form $\alpha |\overline{0}\rangle + \beta |\overline{1}\rangle$.

So, the above will fix any stray unitary transformations that get applied to any one qubit. But what about errors that involve decoherence or measurement?

We claim that once we've handled all possible unitary transformations, we've automatically handled *all* possible errors---because an arbitrary error might turn pure states into mixed states, but it still

keeps us within the same 4-dimensional subspace. So a measurement of the error syndrome still projects us down to one of the same four orthogonal states $|\Psi_0\rangle, |\Psi_1\rangle, |\Psi_2\rangle, |\Psi_3\rangle$, and we can still get back to our original state $|\Psi_0\rangle$ by applying bit flips and phase flips as needed.

What if it's not just one qubit, but *all* qubits that are a little bit off?

A key observation: if all qubits are a little bit off, then that's the same thing with having a superposition over many different configurations, in almost all of which only a few qubits are off (possibly a lot off). So we just apply a standard quantum error-correcting code that's able to deal with a few qubits being a lot off, and that lets us return our state *almost* to what it was before the errors happened.

Shor's 9-qubit code was the first quantum error-correcting code. Not long afterward, Andrew Steane found a shorter code that could also detect and correct any error on 1 qubit. Steane's code encoded 1 logical qubit into only 7 physical qubits. Then Raymond Laflamme and others found codes that used only 5 qubits.

5 qubits turns out to be the least possible if you want to detect and correct an arbitrary 1-qubit error, just like 3 bits is the least possible for a classical error-correcting code. We won't prove this.

In the next lecture, we'll discuss the stabilizer formalism, which gives us an amazingly compact and efficient notation for manipulating these sorts of quantum error-correcting codes.

So we can encode a qubit, let the qubit sit around passively, and protect it against a single physical error. How about doing a full fault tolerant quantum computation?

That's the subject of the famous....

Quantum Fault-Tolerance Theorem

Also known as the **Threshold Theorem**

Proved independently by several groups (Aharonov & Ben-Or / Zurek et al. ~1996)

The theorem says the following: suppose that, in your quantum computer, each qubit fails at each time step with independent probability ϵ . (Where "fails" could mean, e.g., gets swapped out for the maximally mixed state.)

Even then, assuming we're able to:

- apply gates to many qubits in parallel
- measure and discard bad qubits
- pump in fresh $|0\rangle$ qubits
- do extremely fast and reliable classical computation

we can *still* solve any problem in **BQP**, so long as ϵ is sufficiently small

Moreover, with only $\text{polylog}(n)$ overhead of number of gates.

(The initial estimates for ϵ were $\sim 10^{-6}$. That's since been improved, depending on the assumptions.)

Since its discovery, the Fault-Tolerance Theorem has set much of the research agenda for experimental quantum computing. For it says that, once we can decrease error below a certain threshold,

we'll effectively be able to make it arbitrarily small, by applying multiple recursive layers of error-correction.

Journalists often try to gauge progress in experimental quantum computing by asking about the number of qubits, but at least as important is the *reliability* of the qubits. It's reliability that will determine when (if ever) we cross the threshold that would let us get to arbitrarily small error, and add as many additional qubits as we liked.

We're not there yet, but lots of progress is being made on two fronts:

1. Making physical qubits more reliable

Initially, the probability of failure per qubit per time step was of order 1, meaning that the quantum state would barely hold at all. But over the last twenty years, in multiple architectures (including superconducting qubits and trapped ions), there were improvements by several orders of magnitude. Today, anyone can access a 5-qubit superconducting device over the Internet (IBM's "Quantum Experience") that has decoherence rates that wouldn't have been achievable a decade ago.

Meanwhile, a few years ago the group of John Martinis at Google demonstrated decoherence rates, for one or two qubits in isolation, that are *already* below the fault-tolerance threshold (say, $\sim 0.1\%$). So is the problem solved? Unfortunately, it's a bit misleading, because integrating more qubits on a single chip pushes the decoherence rate back up. So the challenge now is to figure out how to scale up to 50 or 100 or 300 qubits while still keeping the error rate down.

2. Inventing better error-correcting codes and fault-tolerance schemes

There are many tradeoffs here: to take one example, between the distance of a code (i.e., the rate of error it can handle) and the number of physical qubits it requires per logical qubit. At least heuristically, it's now known how to handle error rates of up to 3-5%, but only via fault-tolerance schemes that use thousands of physical qubits for every logical qubit. So one big challenge is to invent schemes that can handle large error and *also* have low overheads. Topological qubits, to be discussed two lectures from now, could also help with this tradeoff.

Here's one milestone that hasn't *quite* been achieved yet, but that you should look out for in the relatively near future: the use of a quantum error-correcting code to keep a logical qubit alive for longer that the physical qubits comprising it.

Lecture 28, Tues May 2: Stabilizer Formalism

Today we'll see a beautiful formalism that was originally invented to describe quantum-error correcting codes, but now plays many different roles in quantum computation. First, some definitions:

Stabilizer Gates

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

are the gates CNOT, Hadamard, and $P = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$ (also called the “phase gate”)

Stabilizer Circuits

are quantum circuits made entirely of stabilizer gates

Stabilizer States

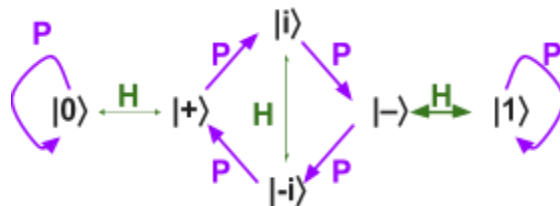
are states that a stabilizer circuit can generate, starting from $|00\dots 0\rangle$

We briefly met stabilizer gates earlier in the course, when we discussed universal quantum gate sets, and needed to include a warning that the set $S = \{\text{CNOT}, \text{Hadamard}, P\}$ is *not* universal. At first, this failure of universality might be surprising. After all, the set S seems to have everything: the Hadamard gate can create superpositions, CNOT acts on two qubits and (in concert with Hadamard) can create complicated entangled states, and P can even add complex phases.

What's more, many of the weird quantum effects and protocols that we saw in this course can be demonstrated entirely using stabilizer gates. Examples include superdense coding, quantum teleportation, BB84 quantum key distribution, Wiesner's quantum money, the Deutsch-Jozsa and Bernstein-Vazirani algorithms, and the Shor 9-qubit code.

So then what prevents S from being universal? Well, if you try playing around with the CNOT, Hadamard, and Phase gates, you'll notice that you tend to reach certain discrete states, but never anything between them. You'll also notice that, whenever you can create an n -qubit superposition that assigns nonzero amplitudes to the strings in some set $A \subseteq \{0,1\}^n$, it's always an *equal* superposition over A (possibly with $+1, -1, +i, -i$ phases), and furthermore A is always an affine subspace of \mathbb{F}_2^n (so in particular, $|A|$ is always a power of 2).

With only 1 qubit, the H and P gates can only get us to 6 states in total (ignoring global phases), via the reachability diagram shown on the right. These 6 states-- $|0\rangle, |1\rangle, |+\rangle, |-\rangle, |i\rangle, |-i\rangle$ --are the 1-qubit stabilizer states.



What about with two qubits?

Now you can reach some more interesting states, like $\frac{|00\rangle + i|11\rangle}{\sqrt{2}}$ or $\frac{|01\rangle - i|10\rangle}{\sqrt{2}}$. But these always follow certain patterns, as mentioned above. For example, they're always equal superpositions over power-of-2 numbers of strings, and a measurement of a given qubit in the $\{|0\rangle, |1\rangle\}$ basis always produces either

(1) always $|0\rangle$,

- (2) always $|1\rangle$, or
 (3) $|0\rangle$ and $|1\rangle$ with equal probabilities.

So what gives?

To answer that question, it will help to define a few concepts.

We say that a unitary U stabilizes a pure state $|\Psi\rangle$ if $U|\Psi\rangle = |\Psi\rangle$.

In other words, if $|\Psi\rangle$ is an eigenstate of U with eigenvalue $+1$. Crucially, global phase matters here! If $U|\Psi\rangle = -|\Psi\rangle$, then U does not stabilize $|\Psi\rangle$.

Notice that if U and V both stabilize $|\Psi\rangle$, then any product of them, like UV or VU , *also* stabilizes $|\Psi\rangle$, as do their inverses U^{-1} and V^{-1} . Also, the identity matrix, I , stabilizes everything.

This means that the set the unitaries that stabilize $|\Psi\rangle$ form a *group* under multiplication.

We already know that unitaries have inverses and are associative.

The next ingredient we need is the **Pauli Matrices**.

These four matrices come up a lot in quantum physics. They are:

$$\begin{array}{cccc} \mathbf{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & \mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & \mathbf{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} & \mathbf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \end{array}$$

Notice that these matrices match up with the errors we need to worry about in quantum error-correction:

<u>No error</u>	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$	<u>Bit flip</u>	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$
$\mathbf{I} 1\rangle = 1\rangle$	$\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$	$\mathbf{X} 1\rangle = 0\rangle$	$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

<u>Phase flip</u>	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$	<u>and Both</u>	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -i \\ -i \end{pmatrix}$
$\mathbf{Z} 1\rangle = - 1\rangle$	$\begin{pmatrix} 0 & -1 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$	$\mathbf{Y} 1\rangle = -i 0\rangle$	$\begin{pmatrix} i & 0 \\ i & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

That's not a coincidence!

The Pauli matrices satisfy several beautiful identities:

$$\begin{array}{lll} X^2 = Y^2 = Z^2 = I & XY = iZ & YX = -iZ \\ & YZ = iX & ZY = -iX \\ & ZX = iY & XZ = -iY \end{array}$$

If you've seen the quaternions, you might recall that they're defined using the same kinds of relations.

This is also not a coincidence! Nothing is a coincidence in math!

Also, all four Pauli matrices are both unitary and Hermitian.

So what does each Pauli matrix stabilize?

I stabilizes everything

$-I$ stabilizes nothing

Remember: global phase matters, so $-I|\Psi\rangle \neq |\Psi\rangle$.

X stabilizes $|+\rangle$
 $-X$ stabilizes $|-\rangle$
 Z stabilizes $|0\rangle$
 $-Z$ stabilizes $|1\rangle$
 Y stabilizes $|i\rangle$
 $-Y$ stabilizes $|-i\rangle$

So each of the six 1-qubit stabilizer states corresponds to a Pauli matrix that stabilizes it.

Next, given an n -qubit pure state $|\Psi\rangle$, we define $|\Psi\rangle$'s stabilizer group as:

The group of all tensor products of Pauli matrices that stabilize $|\Psi\rangle$.

We know this is a group since being a Pauli matrix is closed under multiplication, and so is stabilizing $|\Psi\rangle$. As you can check, stabilizer groups have the additional interesting property of being abelian.

For example, the stabilizer group of $|0\rangle$ is $\{I, Z\}$ closed because $Z^2 = I$
 while the stabilizer group of $|+\rangle$ is $\{I, X\}$

The stabilizer group of $|0\rangle \otimes |+\rangle$ will be the Cartesian product of those two groups:

$\{I \otimes I, I \otimes X, Z \otimes I, Z \otimes X\}$

as a convention, from now on we omit the \otimes 's, so that for example the above is just $\{II, IX, ZI, ZX\}$

For a slightly more interesting example, what's the stabilizer group of a Bell pair?

We know XX is in it because $\frac{X|0\rangle \otimes X|0\rangle + X|1\rangle \otimes X|1\rangle}{\sqrt{2}} = \frac{|11\rangle + |00\rangle}{\sqrt{2}} = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$.

A similar argument can be made for $-YY$.

We can get another element by doing component-wise multiplication: $XX \cdot -YY = -(iZ)(iZ) = ZZ$

So the stabilizer group of $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$ contains $\{II, XX, -YY, ZZ\}$. And you can check that it doesn't contain anything else.

You can similarly compute the stabilizer group of $\frac{|00\rangle - |11\rangle}{\sqrt{2}}$ to be $\{II, -XX, YY, ZZ\}$.

Now, here's an amazing fact, which we won't give a proof of:

The n -qubit stabilizer states are exactly the n -qubit states that have a stabilizer group of size 2^n .

So the 1-qubit stabilizer states are those states with a 2-element stabilizer group, the 2-qubit stabilizer states are those states with a 4-element stabilizer group, and so on.

This is a completely different characterization of stabilizer states, a structural one. It makes no mention of stabilizer circuits, but tells us something about the invariant that stabilizer circuits are preserving.

OK, so suppose we have an n -qubit stabilizer state, which (by the above) has a 2^n -element stabilizer group G . Then here's the next thing we might want to know:

How can we succinctly specify G ? Does G always have a small generating set--that is, a few elements from which we can get all the others by multiplication?

While we again won't prove it, the answer turns out to be yes. Given any n -qubit stabilizer state, its stabilizer group is always generated by only n elements (i.e., \pm tensor products of Pauli matrices). So, to specify a stabilizer group (and hence, a stabilizer state), you only need to specify n such generators.

Let's see an example. To specify the Bell pair, which has stabilizer group $\{II, XX, -YY, ZZ\}$, it's enough to give the following generating set:

(XX)

(ZZ)

Or we could also give a different generating set, like

(XX)

$-(YY)$

Now we come to a crucial point:

How many bits does it take to store such a generating set in your computer?

Well, there are n generators, and each one takes $2n+1$ bits to specify: 2 bits for each of the n Pauli matrices, plus 1 additional bit for the \pm sign. So the total number of bits is

$$n(2n+1) = 2n^2 + n = O(n^2).$$

Naïvely writing out the entire amplitude vector, or the entire stabilizer group, would have taken $\sim 2^n$ bits, so we've gotten an exponential savings. We're already starting to see the power of the stabilizer formalism.

But that power turns out to go much further. Around 1998, Daniel Gottesman and Manny Knill proved the...

Gottesman-Knill Theorem

which says that there's a polynomial-time classical algorithm to simulate any stabilizer circuit that acts on a stabilizer initial state like $|00\dots 0\rangle$.

Here, "simulate" means pretty much anything you could ask for: you can compute the probability of any possible sequence of measurement outcomes, or you can *simulate* the measurement outcomes if given access to a random bit source.

A more negative interpretation is:

stabilizer states and gates, by themselves, are useless for producing superpolynomial quantum speedups.

So, how does the classical simulation work? Just by keeping track, at each point in time, of a list of generators for the current state's stabilizer group! And updating the list whenever a CNOT, Hadamard, Phase, or measurement gate is applied.

Almost the only time that Professor Aaronson (being a theorist) ever wrote code that other people actually used, was when he did a project in grad school for a Computer Architecture course.

He wrote a fast simulator for stabilizer circuits called CHP, which could handle thousands of qubits on a normal laptop (limited only by the available RAM). He was only trying to pass the class, but the challenge of actually implementing the Gottesman-Knill algorithm in an optimized way led to the discovery of an *even faster* classical algorithm for simulating stabilizer circuits, so Aaronson ended up publishing a paper with Gottesman about this.

Truth be told, this project had very little to do with Computer Architecture. He's still not sure why the professor accepted it.

So how does the Gottesman-Knill algorithm work?

For simplicity, let's assume the initial state is $|00\dots 0\rangle$. Then the first step is to find a stabilizer representation (that is, a list of generators) for $|00\dots 0\rangle$.

We know the stabilizer group contains $II\dots I$, but we won't put that into the generating set: it's implied. Since $|0\rangle$ is a +1 eigenstate of Z , you can check that the following generating set works:

$ZIII\dots I$
 $IZII\dots I$
 $IIZI\dots I$
 \vdots
 $IIII\dots Z$

For purposes of the algorithm, it's useful to write these lists of generators in a slightly different way:

Tableau Representation

Here we'll keep track of two $n \times n$ matrices of 1's and 0's (as well as n signs). The two matrices can be combined entrywise to produce an $\{I, X, Y, Z\}$ matrix like the one above. We call them:

The X Matrix and The Z Matrix

+ (0 0 0 0 1 0 0 0)	← Each row represents one generator of the stabilizer group
+ (0 0 0 0 0 1 0 0)	
+ (0 0 0 0 0 0 1 0)	
+ (0 0 0 0 0 0 0 1)	
<div style="display: inline-block; width: 40px; text-align: center;">↑</div> <div style="display: inline-block; width: 40px; text-align: center;">↑</div>	
1 if X or Y	1 if Z or Y
0 otherwise	0 otherwise

Thus, the first row of the above tableau represents the generator $+ZIII$, the second $+IZII$, the third $+IIZI$, and the fourth $+IIIZ$. So this is just another way to represent the generating set $\{ZIII, IZII, IIZI, IIIZ\}$ for the state $|0000\rangle$.

We're now going to provide rules for updating this tableau representation whenever a CNOT, Hadamard, or phase gate is applied. We won't prove that the rules are correct, but you should examine them one by one and see if you can convince yourself.

We're also going to cheat a little. Keeping track of the +’s and -’s is tricky and not particularly illuminating, so we’ll just ignore them. What do we lose by ignoring them? Well, whenever measuring a qubit has a definite outcome (either $|0\rangle$ or $|1\rangle$), we need the +’s and -’s to figure out *which* of the two it is. On the other hand, if we only want to know whether measuring a qubit will give a definite outcome or a random outcome (and not *which* definite outcome, in the former case), then we can ignore the signs.

So what are the rules?

The gates available to us are CNOT, H, and P, so we need to figure out how to update the tableau for each.

- To apply H to the i^{th} qubit:
 - Swap the i^{th} column of the X matrix with the i^{th} column of the Z matrix.

This should be pretty intuitive: the whole point of the Hadamard gate is to “swap the X and Z bases.”
- To apply P to the i^{th} qubit:
 - Bitwise XOR the i^{th} column of the X matrix into the i^{th} column of the Z matrix.

Note that P has no effect on the tableau representation of $|00\dots0\rangle$.
Coincidence? I think not.
- To apply CNOT from the i^{th} qubit to the j^{th} qubit:
 - Bitwise XOR the i^{th} column of the X matrix into the j^{th} column of the X matrix.

That seems reasonable enough, *but ...* remember how a CNOT from i to j is equivalent, when viewed in the Hadamard basis, to a CNOT from j to i ?
That means we also have to...

 - Bitwise XOR the j^{th} column of the Z matrix into the i^{th} column of the Z matrix.

Finally, whenever the i^{th} qubit is measured in the $\{|0\rangle, |1\rangle\}$ basis: the measurement will have a determinate outcome *if and only if* the i^{th} column of the X matrix is all 0’s. (Can you figure out why?)

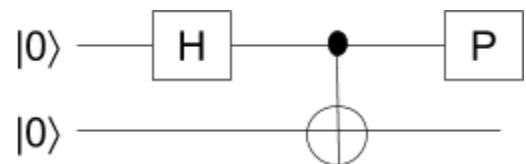
There are also rules for updating the tableau in the case that the measurement outcome is not determinate, but we won’t cover them here.

Here’s another cool fact: in our state, the number of basis states that have nonzero amplitudes is just 2^k , where k is the rank of the X matrix.

In the above example, $\text{rank}(X) = 0$, corresponding to the fact that our “superposition” only contains a single basis state, namely $|0000\rangle$.

Let’s test this out, by keeping track of the tableau for the circuit on the right. We start with

(0 0 | 1 0)
(0 0 | 0 1)



After the Hadamard we get

(1 0 | 0 0)
(0 0 | 0 1)

(swap 1st columns of X and Z)

You could convert this back into Pauli notation by saying that the current state is the one generated by +XI and +IZ.

That makes sense, since those do indeed generate the stabilizer group for $|0\rangle \otimes |+\rangle$

After the CNOT: (In X: XOR 1st column into 2nd. In Z: XOR 2nd column into 1st)
 $(1\ 1\ | 0\ 0)$ This is $(X\ X)$, the stabilizer generator for a Bell pair.
 $(0\ 0\ | 1\ 1)$ $(Y\ Y)$

After the phase gate: (XOR 1st column of X into 1st column of Z)
 $(1\ 1\ | 1\ 0)$
 $(0\ 0\ | 1\ 1)$

This corresponds to the state $\frac{|00\rangle + i|11\rangle}{\sqrt{2}}$.

A stabilizer code is a quantum error-correcting code in which the encoding and decoding (at least if there are no errors!) can be done entirely by stabilizer circuits. In particular, this means that all the code states are stabilizer states.

In quantum computing research, most of the error-correcting codes that have been seriously considered are stabilizer codes. The reason for this is similar to why linear codes play such a central role in classical error correction: namely,

- (1) it makes everything *much* easier to calculate and reason about, and
- (2) by insisting on it, we don't seem to give up any of the error-correcting properties we want.

As a result, the stabilizer formalism is the lingua franca of quantum error-correction; it's completely indispensable there.

To take an example: with Shor's 9-qubit code, we were dealing with states of the form

$$\left(\frac{|000\rangle \pm |111\rangle}{\sqrt{2}} \right)^{\otimes 3}$$

We claim that a generating set for the above state's stabilizer group is as follows:

$$\begin{aligned} \{ & \begin{array}{ccccccccc} \mathbf{Z} & \mathbf{Z} & \mathbf{I} & & \mathbf{I} & \mathbf{I} & \mathbf{I} & & \mathbf{I} \\ \mathbf{I} & & & & \mathbf{I} & & & & \mathbf{I} \\ \mathbf{I} & & & & \mathbf{I} & & & & \mathbf{I} \\ \mathbf{I} & & & & \mathbf{I} & & & & \mathbf{I} \\ \mathbf{I} & & & & \mathbf{I} & & & & \mathbf{I} \\ \mathbf{I} & & & & \mathbf{I} & & & & \mathbf{I} \\ \mathbf{X} & \mathbf{X} & \mathbf{X} & & \mathbf{X} & \mathbf{X} & \mathbf{X} & & \mathbf{I} \\ \mathbf{I} & & & & \mathbf{X} & \mathbf{X} & \mathbf{X} & & \mathbf{X} \end{array} \\ & \pm \begin{array}{ccccccccc} \mathbf{X} & \mathbf{X} & \mathbf{X} & & \mathbf{X} & \mathbf{X} & \mathbf{X} & & \mathbf{X} \end{array} \} \end{aligned}$$

The last line can have either a + or −, encoding $|\bar{0}\rangle$ or $|\bar{1}\rangle$ respectively.

Why are the above elements in the stabilizer group? Well, phase-flips applied to any pair of qubits in the same block cancel each other out. Bit-flips also take us back to where we started, except possibly with the addition of a global -1 phase.

You then just need to check that these 9 elements are linearly independent of each other, meaning that there aren't any more to be found.

Now that we know the stabilizer formalism, we're finally ready to see an "optimal" (5-qubit) code for detecting and correcting an error in any one qubit. The codeword states would be a mess if we wrote them out explicitly--superpositions over 32 different 5-bit strings! But everything is much more compact if we use the stabilizer formalism. Here's the code:

$$\begin{aligned} & \{ XZZXI, \\ & \quad IXZZX, \\ & \quad XIXZZ, \\ & \quad ZXIXZ, \\ & \pm XXXXX \} \end{aligned}$$

Once again, the sign on the last generator is + if we want the logical $|\bar{0}\rangle$ state, or - if we want the logical $|\bar{1}\rangle$ state.

One can check (we won't prove it here) that this code can indeed correct either a bit-flip or a phase-flip error on any one of the five qubits.

To conclude this lecture, let's say a tiny bit about doing actual quantum *computations* on qubits that are encoded using stabilizer codes.

Thus, suppose we have n logical qubits, each encoded with a stabilizer code, and we want to apply a gate to one or two of the logical qubits. The "obvious" way to do this would be:

1. Decode the qubits.
2. Apply the desired gate to the "bare," unencoded qubits.
3. Re-encode the result.

But doing all that is expensive, and creates lots of new opportunities for error! (E.g., while the qubits are unencoded, there's "nothing to protect them" from decoherence.)

So it would be awesome if we had a code where applying gates to encoded qubits was hardly more complicated than applying them to unencoded qubits. This motivates the following definition:

The gate G is transversal for the code C , if in order to apply G to qubits encoded using C , all you need to do is:

- Apply G to the first qubits of the codewords
- Apply G to the second qubits of the codewords
- etc.

So for example, the Hadamard gate is transversal if you can Hadamard a logical qubit by just separately Hadamarding each physical qubit in the codeword.

You should check that the Hadamard gate is transversal for Shor's 9-qubit code.

It turns out that there are quantum error-correcting codes for which the CNOT, Hadamard, and Phase gates are *all* transversal. Thus, if you use one of these codes, then applying *any* stabilizer circuit to the encoded qubits is extremely cheap and easy.

Unfortunately, we already saw that the stabilizer gates are non-universal---and there's a theorem that says that non-stabilizer gates *can't* all be transversal.

This means that, if we want universal quantum computer, we're going to need non-stabilizer gates like Toffoli or $R_{\pi/8}$ that *can't* be implemented transversally, but only via sequences of gates that are much more expensive.

So the quantum computer engineers tend to adopt a worldview wherein stabilizer gates are "free"---they're so cheap to implement that you might as well not even count them---and the "complexity" of a quantum circuit equals the number of non-stabilizer gates. The non-stabilizer gates are so much more expensive that they completely dominate the running time.

In practice, a lot of quantum computer engineering has boiled down to designing improved methods for getting non-stabilizer gates into a circuit. There are various tricks, a famous example being Magic State Distillation.

The idea there is that, if you can just produce certain non-stabilizer states like $\cos(\pi/8)|0\rangle + \sin(\pi/8)|1\rangle$ -- those are the "magic states" -- then applying stabilizer operations to those states, together with measurements (and adapting based on the outcome of the measurements), is enough to *simulate* the effect of non-stabilizer gates. In other words, with help from magic states, stabilizer operations can break out of the Gottesman-Knill prison and get all the way up to universal quantum computation. On the other hand, actually realizing this idea seems to require building a quantum computer where the *overwhelming majority* of the work would happen in "magic state factories," with the actual quantum computation on the magic states almost an afterthought.

There's a different way to understand the importance of non-stabilizer states for quantum computation. The paper by Aaronson and Gottesman from 2004, mentioned earlier, also proved the following result:

Suppose we have a quantum circuit on n qubits, which contains mostly stabilizer gates---say, $n^{O(1)}$ of them---but also a small number T of non-stabilizer gates. Then there's a classical algorithm to simulate the circuit in time that's polynomial in n and exponential in T .

This tells us that, if we want an exponential quantum speedup, then not only do we need non-stabilizer gates in our circuit, we need a polynomial *number* of such gates.