Lecture 29: Thurs May 4

For a given quantum error correction code, applying a gate usually entails:

decoding the qubit => applying the gate => re-encoding the qubit

That's why in practice people prefer quantum error correction codes with **transversality**.

We say that the Hadamard gate is **transversal** for a qubit if you can Hadamard the logical qubit by applying the Hadamard gate to each physical qubit separately.

You can work out that Hadamard is transversal for Shor's 9-qubit code.

There are quantum error correction codes where cNOT, H, and P are all transversal.

Unfortunately, there's a theorem that says that arbitrary non-stabilizer gates *can't* be transversal. That means gates like Toffoli or $R_{\pi/8}$ must be implemented through sequences of gates that are much more expensive.

So in practical quantum computing, stabilizer applications cost almost nothing.

A 2004 paper by Aaronson and Gottesman says:

To simulate a circuit with mostly stabilizer gates (n gates total, T non-stabilizer gates) requires a runtime that's polynomial in n, and exponential in T.

This means that an exponential speedup in quantum computing requires an exponential number of stabilizer gates.

There are various tricks to produce this, like Magic State Distillation.
The basic idea is that applying stabilizer gates to some nonuniformly-distributed states called 'magic states', such as $\cos(\pi/8)|0\rangle + \sin(\pi/8)|1\rangle$ lets you escape Gottesman-Knill and reach a universal quantum computer.

We've seen a high-level overview of quantum computing, so it behooves us to take a lecture to discuss practical implementations.

Professor Aaronson has visited quantum computing labs all over the world.
They all have one strict rule for theorists: "don't touch anything!"

So to recap: Why would someone want to build a scalable quantum computer?

Proving it's possible is reason enough for many—it would show whether nature defies the Extended Church-Turing Thesis, not to mention disproving the people who say it's not possible.

But there are also important computational speedups to keep in mind, the top five (in order) are:

1. Quantum Simulation

Take a Hamiltonian of a real system, trotterize it, then run.

This would let you compute the effect of any chemical reaction without physically running it, which would be amazing for chemists. We tend not to talk much about this, since it's pretty straightforward, but it's easily the best application of quantum computing.

Even imperfect implementations of quantum computing are enough to see advantages for this.

2. Code Breaking

The sexiest application of quantum computing.

This would be very important for intelligence agencies, nefarious actors, intelligence agencies *who are themselves* nefarious actors.

It would completely change how e-commerce is run, requiring everybody to move to private-key crypto, lattice cryptography, etc. However, advantages here would require a fully fault-tolerant quantum computer.

3. Grover

As we've seen, Grover's Algorithm can only provide a polynomial speedup. However, it would be over a broad range of applications.

It would essentially just "give a little more juice to Moore's Law."

4. Adiabatic Optimization

Might produce speedups better than Grover, but we'll only know once we try.

5. Machine Learning

Very hot in recent years.

It's a good match for quantum computing because many problems in the field (classifying data, creating recommendation systems, etc) boil down to performing linear algebra on large sets of data. Even better, you typically only need an approximate answer.

Over the past ten years, many papers have been published claiming that quantum computing can give up to exponential speedups on such problems. This started in 2007 with…

The HHL Algorithm (Harrow, Hassidim, Lloyd)

which is billed as a quantum algorithm to solve linear systems exponentially faster than a classical computer can.

There is a catch (which Professor Aaronson wrote an article in *Nature* about).

In the fine print of the algorithm, it's assumed that the input and the output are in a quantum format. Usually, we only assume that we have $A\overline{x} = \overline{b}$ with A and b stored in memory.

But this algorithm says:

"Suppose I have qubits encoding $|b\rangle$ and I'm able to apply a Hamiltonian $e^{-iH}|b\rangle = |x\rangle$ which uses matrix A to encode the solution vector. Then (assuming e^{-iH} follows a few conditions), you can do this with an n by n matrix"

Converting into and out of a quantum format may be hard enough that the entire process would have no speedup relative to a classical computer.

It's like the algorithm gets you halfway across the world, but leaves you stranded at the airport.

For example, if you had $|x\rangle = \sum_{i=1}^{n} \alpha_i |i\rangle$ and you want to get all the $|i\rangle$'s out, it may be necessary to run and measure n times, at which point you're not getting an exponential speedup.

Journalists often ask Professor Aaronson, "When will we all have personal quantum computers and qPhones in our pocket?" It's hard to imagine that'll ever happen though, because most things we do on our PCs can be done quickly on classical computers. At most we'll likely see cloud quantum computing, like the IBM Quantum Experience, where a central location deals with the issues of maintaining quantum states while we reap the benefits.

Maybe this'll seem myopic in a hundred years, like a guy from the 70s saying, "I only see a market for five computers in the world, tops." But you could argue that such people were simply ahead of their time. We *are* moving to a world where most computation is done on the cloud in a few centralized locations. Though this might also be shortsighted because our current list of applications of quantum computing may be woefully incomplete.

<u>So what *do* you have to do to build a quantum computer?</u>

There's a famous list of requirements it takes for a system to be able to to quantum operations called the **DiVincenzo Criteria**. There are several, but the four most important are…

- *Long-Lived Qubits*

It's self-evident that you need some system that can maintain quantum states over long periods of time. As we've said before, "the first requirement of quantum computing is the ability to perform I."

- *Universal Gates*

You must be able to apply *some* universal set of gates.

Implicit here is the requirement that qubits can interact with one another.

- *Initialization*

You must be able to get qubits to $|00\ldots0\rangle$.

- *Measurement*

You're familiar with measurement.

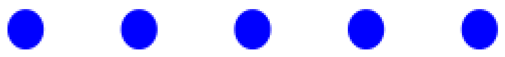
Different architectures have achieved different combinations of these. There are architectures where initialization is hard or measurement is hard.

<u>So what are the major architectures that have been built?</u>

To a theorist, a qubit is a qubit is a qubit.
But experimentalists talk about four main architectures that may work.

The oldest approach, dating back to 1985 is **Trapped Ions**.

The basic idea is that you have a bunch of ions (let's say they're atomic nuclei) with electric charges so that they respond to a magnetic field. You can then manipulate the magnetic field to get them trapped in a line.



Such a lab will have ions, a magnet, and a classical computer that lets them see images of the ions. This method isn't totally reliable, and it takes work to keep the ions penned in.

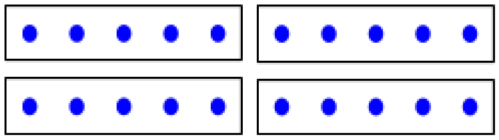
It's a bit like herding cattle.

Atomic nuclei have a spin state that can be clockwise, counterclockwise, or a superposition of both. So we treat the nuclei's spin as their quantum state. If you bring two ions close to each other, the *Coolum Effect* can be used to create something resembling a CNOT gate.

You can manipulate qubits by using a laser to pick them up and move them around. This may sound like it would be a tough balancing act, moving nuclei via laser *while* keeping them all floating magnetically…

Yes. Yes it is.

But it has been demonstrated with up to 10-20 qubits. After that it becomes hard to interact with a single qubit at a time.



Proposals to scale up this method often involve many small ion traps, with some 2-qubit gates that interact between traps. You could use quantum teleportation to communicate between traps.

Many groups are pursuing this at NIST, UMaryland, and Innsbruck (Austria).

In the last few years, several such ventures that were originally academic became start-ups—which tend to give out much less information about how they're doing.

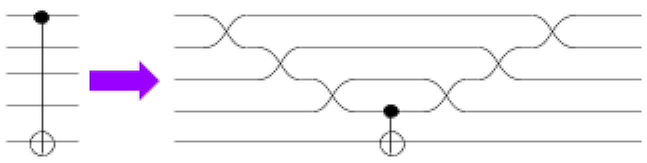
Another approach is **Superconducting Qubits**.

We talked about this earlier with DWave, but there are many others trying this.

In this approach, everything happens on a chip in a refrigerator cooled down enough for the coils to superconduct.

Electrons can flow around the qubit clockwise, counterclockwise, or in a superposition of both. If two of these qubits come close, *some* 2-qubit Hamiltonian happens between them.

Advantages of this setup:	– You can make lots of these coils
	– Gate operations are very fast
Disadvantages:	– Coherence times are much shorter
The big one is that	– These qubits can't move around and can only talk to their neighbors.



In designing quantum circuits, we've been implicitly assuming that any two qubits can interact. Of course you could always simulate it with a whole cascade of swaps, but you pay a price for that.

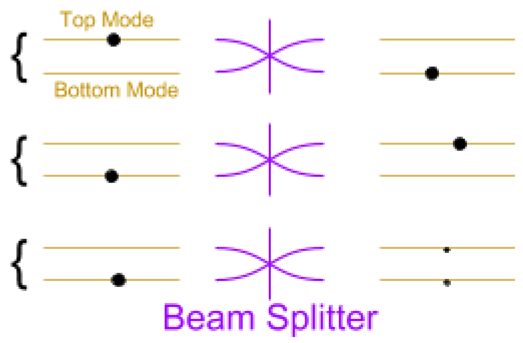
That being said, this is the currently the most popular approach.

Google bought out almost the whole Santa Barbara lab (Martinez's group), and have publicly announced that they expect to have a 50-qubit system in a year. IBM also has a superconducting group with similar claims. In addition there are several startups working with superconducting qubits, like Rigetti—made up of several people who left IBM.

A third approach is **Photonics**

which treats photons as qubits. For example, you could say that the photon is either horizontally polarized, vertically polarized, or in a superposition of being polarized both ways.

There's many ways to generate photons in such a system. One way is to use **Dual Rail**. Each photon has two modes—two places it can be in. A photon can be in the top mode, the bottom mode, or in a superposition of both.



The basic idea is that you generate photons and send them through fiber optic cables. When the two come together, you use a **Beam Splitter**, which corresponds as a 2x2 unitary acting on the qubit, take taking the state to or from superposition.

2-qubit gates are harder to do. There's a set of operations that are easy to implement in such a system. What's not obvious is whether those operations are sufficient to produce a universal quantum computer.

There was a breakthrough in this area in 2001 called

The KLM Theorem (Knill, Laflamme, Milburn)

which says that if I can generate photons and send them through beam splitters, phase shifters, *plus* at any time, for all channels, I can tell if there's a qubit, or not, or a piece of qubit in the channel and feed the answer forward to further operations:

Then LO + FFM = BQP (linear optics + feed-forward measurement = BQP)

Furthermore, the KLM Theorem opens the possibility of a new way to build a quantum computer, where qubits are photons travel at the speed of light.

Photons can maintain superposition indefinitely by flying in a vacuum. The trouble is that they're flying at the speed of light, which makes it hard for them to interact with one another. This may require stalling photons, but that may introduce decoherence.

In 2011 Professor Aaronson and Alex Arkhipov proposed **Boson Sampling**,

The idea was to investigate what can be done with linear optics if you <u>don't</u> have feed-forward measurement. They concluded that this probably can't produce a universal quantum computer, *but* it could do some problems faster than a classical computer.

Not any problems that people actually care about, mind you.
Nevertheless, it would be a good candidate to prove that *any* quantum speedups truly exist.

<u>Could you represent a beam splitter as a 2x2 matrix?</u>

We're sweeping several lectures about photonics under the rug here. The jist of it is that instead of building tensor products, composite systems are created out of smaller systems in a different way.

The last approach we'll cover is fairly esoteric, it's called **Non-abelian Anyons**.

There are two types of fundamental particles: Bosons and Fermions. But in a two-dimensional field, you can have particles that behave as neither Bosons nor Fermions.

Lots of physicists won Nobels in the 80s for this stuff.

If you can make such "quasiparticles" in a two-dimensional surface, just moving them around would be sufficient to create a universal quantum computer. This setup may be naturally resistant to decoherence. The caveat is that we're only now starting to understand how to create the simplest quasiparticles.

Microsoft is the current leader in this approach, and has hired several experts in this field recently.

As a path forward…
Professor Aaronson's thinks about what to expect from **Quantum Supremacy** in terms of three steps.

Lots of people dislike this term ^ for obvious reasons, but it has stuck for now.

<u>Step 1: Doing *something* faster than a classical computer can.</u>

50 qubits may be enough, and Boson Sampling may be used to achieve this.

<u>Step 2: Doing useful quantum simulations</u>

A Microsoft paper claims that 100 qubits would be enough to simulate one quantum system using another for several useful applications.

<u>Step 3: Creating a full universal quantum computer</u>

Which would let us finally run scalable implementations of Shor's Algorithm.

Step 1 may likely be coming soon, but no promises on steps 2 and 3!