Lecture 29, Thurs May 4: Experimental Realizations of QC

In this course, we've had 28 lectures about the theory of quantum computing and information. So it behooves us to take one lecture to survey the current state of experimental implementations.

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

First, though, let's recap: Why would someone *want* to build a scalable quantum computer?

Proving that it's possible is reason enough for many of us—we could test whether Nature defies the Extended Church-Turing Thesis, not to mention the people who claim QC is impossible.

But there are also some practical computational speedups we could hope to achieve. Among those known today, perhaps the top five are:

1. Quantum Simulation

Take a Hamiltonian of a real quantum system, Trotterize it, then simulate on a gate-based quantum computer.

In principle, this could let you quickly see the effect of any chemical reaction without physically performing it, which could be transformative for some parts of chemistry and materials science. Until recently, we've tended not to talk much about this in quantum computing theory, since it seemed too "obvious." But it was Feynman's "original" application of QC, and it's still easily the most commercially important application that we're confident about.

Notably, even imperfect, non-fully-error-corrected quantum computers with a few hundred qubits, of the sort we hope to build in the near future, might be enough to start seeing advantages for quantum simulation.

2. Codebreaking

This is the "sexiest" application of quantum computing, the one that brought the field to worldwide attention in 1994.

This application could be very important for intelligence agencies, nefarious actors, and intelligence agencies who are *themselves* nefarious actors.

It would cause a drastic change to the infrastructure behind electronic commerce--forcing everybody to migrate to lattice-based or other quantum-resistant cryptosystems, or conceivably even quantum key distribution.

However, this is not exactly a *commercial* application of QC! And in any case, breaking public-key crypto would require a fully fault-tolerant, scalable quantum computer, which we seem unlikely to get in the next decade.

3. Grover

As we saw, Grover's algorithm can only provide a square-root speedup, not an exponential one. However, Grover and its variants could give this sort of speedup for an enormous range of applications.

Grover could, in effect, "give a little more juice to Moore's Law."

4. Adiabatic Optimization

As we discussed, this approach *might* produce speedups better than Grover's algorithm, at least for special classes of optimization problems, but we'll only know once we try.

5. Machine Learning

The possibility of quantum speedups for machine learning problems has become an extremely hot topic in recent years. Since we didn't really discuss this earlier in the course, let's have an extended digression now.

Intuitively, machine learning seems like a good match for quantum computing because many machine-learning tasks (support vector machines, recommendation systems, etc.) boil down to performing linear algebra on extremely high-dimensional real vectors. Even better, you typically only need an approximate answer.

And indeed, over the past decade, many papers have been published claiming that quantum computing can give up to exponential speedups for various machine learning problems. The trend started in 2007 with…

The HHL Algorithm (Harrow, Hassidim, Lloyd)

which is sometimes described as a quantum algorithm to solve linear systems of equations exponentially faster than a classical computer can.

However, there are several catches in the "fine print" of the algorithm.

Most importantly, it's assumed that the input and the output are in a quantum format. If we're trying to solve

$$A\overline{x} = \overline{b},$$

where $\overline{b}$ is a vector of length n and A is an $n{\times}n$ matrix, normally we'd assume only that A and $\overline{b}$ are stored somewhere in memory. But the HHL algorithm says:

"Suppose I have a ($\log n$)-qubit quantum state $|b\rangle$ whose amplitudes encode (a normalized version of) the vector $\overline{b}$. Suppose also that I'm able to efficiently apply a Hamiltonian H that corresponds to A. Then assuming one further condition (about the matrix A being far from singular), in logarithmic time, I can prepare a ($\log n$)-qubit quantum state $|x\rangle$ whose amplitudes encode a normalized version of the solution vector $\overline{x}$."

The issue is that, for many applications, converting into and out of a quantum format might be hard enough that the entire process would have no speedup relative to a classical computer.

It's as if 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 needed to get all the α_i's out, even approximately, you'd in general need to run the algorithm and measure O(n) times. But at that point you'd no longer be getting an exponential speedup.

Whether we can use HHL, or similar quantum algorithms, to get exponential quantum speedups for "end-to-end" machine learning applications--that is, problems with (1) a classical input, (2) a classical output, and (3) plausibly no efficient classical algorithm for mapping the input to the output--is an extremely active research question right now.

Journalists often ask Professor Aaronson, "When will we all have personal quantum computers, and qPhones in our pocket?" The trouble with answering that question is that most things we do on our PCs or phones can already be done perfectly well by classical computers, with at most limited prospects for a quantum speedup. We do think that quantum speedups could be transformative, but mostly for quantum simulation and a few other specialized applications that would only indirectly affect the end user. Combined with the extreme difficulty of maintaining a coherent quantum state, this seems to make it seem more natural to imagine quantum computing as a *cloud* resource, as IBM and Rigetti and others are already demonstrating on a small scale. In this model, a central location would deal with all the issues of cooling to near absolute zero, battling decoherence, etc. while everyone else could reap the benefits.

Maybe this will seem hilariously myopic in a hundred years, like the guy in the 1940s who foresaw a world market for at most five computers, or Ken Olsen (the former Digital Equipment CEO) declaring in the 1970s that there was absolutely no reason why anyone needed a computer in their home. But you could also argue that those people were simply ahead of their time! We *are* moving to a world where a great deal of computation is done on the cloud in centralized locations, and end users just access the results remotely.

Of course, the picture might also be shortsighted simply because our current list of applications of quantum computing is woefully incomplete.

So what *do* you have to do to build a quantum computer?

There's a famous list of engineering requirements for a scalable universal quantum computer, called the **DiVincenzo Criteria** (after the physicist David DiVincenzo). The actual list sometimes varies, but we'll give perhaps the four most important entries, the ones everyone agrees about:

- *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 for quantum computing is the ability to perform the identity gate reliably."

- *Universal Gates*

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

Implicit here is the requirement that the qubits can interact with one another in a controlled way.

- *Initialization*

You must be able to initialize the qubits to a convenient state like |00…0⟩.

- *Measurement*

You must be able to measure a given qubit, say in the {|0⟩,|1⟩} basis.

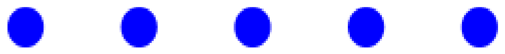
Currently, some quantum computing architectures have made more progress on some of these requirements, some on others. E.g., there are architectures where preparing the initial state is hard or where measurement is hard, others where those tasks are near-trivial. There are architectures where the qubits have extremely long lifetimes but are difficult to apply gates to, others where the qubits have short lifetimes but are much easier to operate on before they decohere.

So what are the major architectures that have been explored so far?
To a theorist, a qubit is a qubit is a qubit.

But experimentalists talk about many different *kinds* of qubits. And while a few implementation proposals have enjoyed the lion's share of success and interest, it's fair to say that no one proposal is yet overwhelmingly dominant. In the rest of this lecture, we'll say a bit about four proposals: Trapped Ions, Superconducting Qubits, Photonics, and Nonabelian Anyons.

Perhaps the oldest approach, dating back to 1995 or so, is Trapped Ions.

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



A trapped ion lab will have a (perhaps fist-sized) magnetic trap, and a classical computer that displays a noisy, pixelated image of where the ions currently are. (Yes, you can now routinely "see" individual atomic nuclei!)

The technology isn't totally reliable, and it takes work to keep the ions penned in.

It's a bit like herding cattle.

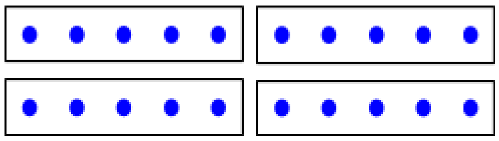
Once we fix a direction, each atomic nucleus has a spin state that can be up, down, or a superposition of both. So we can use these nuclear spin states as our qubits. Also, if you bring two ions close together, the natural *Coulomb interaction* between them can be used to engineer something resembling a CNOT gate.

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

Yes. Yes it is.

But it's now been convincingly demonstrated with up to a 20 or so qubits in a single trap. As the number of qubits in the trap increases much beyond that, it becomes harder and harder to use the laser to target a single qubit at a time.

So, proposals to scale up trapped-ion quantum computing often involve *many* traps, with the challenge then being how to implement 2-qubit gates between one trap and another one. One idea for how to do this is to use quantum teleportation--which, of course, reduces the problem to distributing Bell pairs between the traps, plus doing classical communication between them.



Currently, some of the leading experimental ion-trap groups are based at NIST, the University of Maryland, and Innsbruck, Austria.

Over the last few years, some such ventures that were originally academic became start-ups—which tend to give out less information about what they're doing! (We've also seen the same trend with the other implementation proposals discussed below.)

The second proposal we'll discuss is Superconducting Qubits.

We already saw superconducting qubits in the context of D-Wave Systems. But many others are now trying to scale superconducting QC, using qubits with much higher coherence times than D-Wave's, and also the ability to do quantum computations other than quantum annealing.

In the superconducting approach, all the action happens on coils on a chip, which has the size and appearance of an ordinary computer chip, but which is placed in a dilution refrigerator and cooled down enough for the coils to superconduct. (In current setups, the temperature is around 10 milliKelvin.) So, the entire device is the size of a small room, but almost all of it is for cooling and for control electronics.

Superconductivity--where current flows with zero resistance--is a famous quantum effect, which is notorious for being present even at macroscopic scales. Related to that, superconducting qubits are absolutely enormous by the standards of qubits: in some cases, the coils are nearly large enough to see with the naked eye!

But what *are* the qubits? Well, given a superconducting current flowing around a coil, there are various degrees of freedom that one could use as a qubit: one example is whether the current is flowing clockwise, counterclockwise, or in a superposition of the two.

Between two of the coils, you can place a thin layer called a Josephson junction, which lets electrons tunnel from one coil to the other, and can result in a 2-qubit gate being implemented.

Advantages of this setup:

– You can make lots of these coils
– The gate operations are extremely fast (nanoseconds)
– Much of the needed chip fabrication technology already exists (because of classical computers!)

Disadvantages (compared to trapped ions):

– The coherence times are much shorter (currently tens of microseconds, compared to as long as seconds for trapped ions)
– The superconducting qubits can't be moved around, and can only talk to their near neighbors.

Throughout this course, when designing quantum circuits, we implicitly assumed that you could apply a two-qubit gate between *any* pair of qubits. But with implementation proposals like superconducting qubits, that assumption doesn't hold: two-qubit gates are only between neighbors. Naturally, you can always *simulate* a two-qubit interaction between faraway qubits using a whole cascade of swaps: swap qubit A until it's close to B, then apply your two-qubit gate between A and B, and finally swap A back into position. But you pay a price for that in efficiency, which grows like the diameter of your lattice of qubits (so, linearly if the qubits are in a 1D chain, like $\sqrt{n}$ if they're in a square grid, etc).

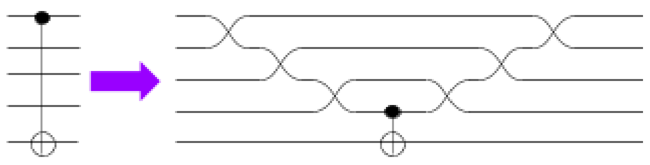
Despite the drawbacks, superconducting qubits are the currently the most popular approach to quantum computing, with major efforts underway right now at Google, IBM, Intel, Rigetti (a startup company), and elsewhere.

The superconducting qubits group of John Martinis is considered possibly the most advanced in the world right now. This group has successfully demonstrated a 22-qubit superconducting chip, and is right now trying to scale up to 72-qubit chip (called "Bristlecone"), with which they hope to give the first demonstration of "quantum supremacy" (that is, a clear quantum computing speedup, possibly for a contrived task). The Martinis group had been based in UC Santa Barbara, but a few years ago, Google bought up the entire group. The lab is still in Santa Barbara, but off-campus now.

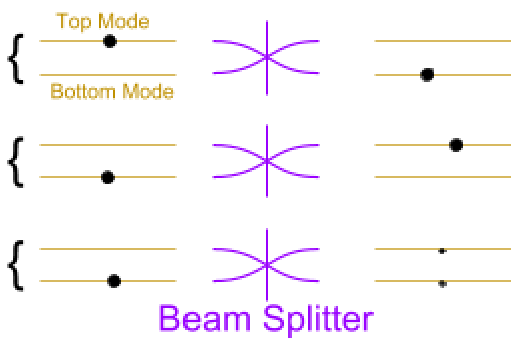
IBM also has a strong superconducting group based in Yorktown Heights, NY, which is aiming for 50 or more integrated and controllable superconducting qubits. Meanwhile, the startup Rigetti, based in Berkeley, was formed by superconducting qubit people who left IBM.

The third approach we'll discuss is Photonics.

Here, not surprisingly, qubits are implemented using light. A photon has several degrees of freedom that can be used as a qubit. For example, a given photon could be either horizontally polarized (representing the state $|0\rangle$), vertically polarized (representing the state $|1\rangle$), or diagonally polarized (representing a superposition of $|0\rangle$ and $|1\rangle$).



But even more simply, imagine a photon that can be traveling down channel A (representing $|0\rangle$), channel B (representing $|1\rangle$), or a superposition of the two channels. This sort of encoding of qubits is called the **Dual-Rail Representation**, since the two channels, when drawn side-by-side, look like railway tracks. Each channel where a photon could be is called a **spatial mode**. So, a photon could be in the top mode, the bottom mode, or a superposition of both.



The basic idea is that you generate photons and send them through channels (which could be fiber-optic cables, but could also just be grooves cut into an optical chip). To cause two spatial modes to interact with each other, you can use a **beamsplitter**, which corresponds to a 2×2 unitary matrix acting on those modes, with the identity matrix on all the other modes.

In contrast to the situation with qubits, the total Hilbert space is *not* a tensor product of Hilbert spaces for the individual modes, and the unitary transformations are also not formed by tensor products. A discussion of the multi-photon case would take us too far afield, but if there's just a single photon that can be in any of m modes, then we're simply talking about an m-dimensional Hilbert space, acted on by $m \times m$ unitary transformations. A beamsplitter is then such a unitary that happens to be the identity except on a 2×2 block.

Given qubits that are encoded in the dual-rail representation, it's easy to apply any 1-qubit gate of our choice (like Hadamard), by just applying the appropriate beamsplitter to the $|0\rangle$ and $|1\rangle$ rails.

Two-qubit gates are harder to implement. Indeed, it's not even obvious a-priori whether beamsplitters *suffice* to implement a two-qubit gate, and hence universal quantum computation.

There was a breakthrough in this area in 2001 called...

The KLM Theorem (after its discoverers Knill, Laflamme, and Milburn)

This theorem says the following: if you can generate identical single photons and send them through a network of beamsplitters, and *additionally*, for any given channel, you can measure to see whether there's a photon there or not, and then feed the answer forward (using classical computation) so that it influences which beamsplitters get applied in the future, *then* you can implement universal quantum computation.

We could summarize this semi-facetiously in an equation:

LO + **FFM** = **BQP** (linear optics + feedforward measurement = **BQP**)

Besides its theoretical interest, the KLM Theorem opened up the possibility of a new way to build a quantum computer, where the qubits would be encoded by photons traveling at the speed of light, and gates would be encoded by beamsplitters and feedforward measurements. This approach is now being pursued by various groups around the world, perhaps most notably the startup Psi Quantum (based in Palo Alto).

Perhaps the central advantage of optical QC is that photons can maintain their quantum coherence for *extremely* long times---as long as they travel in straight lines! Indeed, the photons in the cosmic microwave background (CMB) radiation may have maintained their quantum coherence for billions of years. The trouble, of course, is that these photons are *also* flying at the speed of light, which makes it very hard to control them! One can stall photons (e.g., by making them travel around and around a loop of fiber-optic cable), but the stalling can introduce decoherence.

The above discussion left one key question unanswered: namely, what about a photonic quantum computer that *didn't* include feedforward measurements? What would be the power of *that*?

In 2011, Professor Aaronson and Alex Arkhipov addressed this question in their study of the so-called **BosonSampling** model. BosonSampling basically just means optical quantum computing where you can:

- Generate identical single photons, one per mode
- Send the photons through an arbitrary network of beamsplitters
- Measure each mode to see whether a photon is present

but you don't have feedforward measurements.

Aaronson and Arkhipov concluded that, while these resources probably don't suffice to build a universal quantum computer, capable of *all* **BQP** problems, they *do* suffice to solve a special "BosonSampling" problem in polynomial time, which very plausibly requires exponential time for a classical computer.

Mind you, BosonSampling might not be a problem that anyone would ever care about in practice! Indeed, the problem involves sampling a random matrix of i.i.d. Gaussian complex numbers, but in a way that's biased toward matrices with larger values of their so-called *permanent* function---not a problem that companies have exactly been clamoring to solve. Nevertheless, if one could build single-photon sources, beamsplitter networks, and so on that worked reliably enough, then this could be a good candidate to show that *any* quantum speedups are truly possible in our world.

The fourth and final implementation proposal that we'll discuss today is also the most esoteric one. It's called Topological Quantum Computing with Non-Abelian Anyons.

While we unfortunately didn't have time to cover it in this course, there are two basic types of fundamental particle in our universe: **Bosons** and **Fermions**. They differ in what happens to the amplitude of a given state when you swap two identical particles. If the swapped particles are bosons, nothing happens. If they're fermions, then the amplitude gets multiplied by -1.

Photons are bosons, as are other force-carrying particles like the gluons and W and Z and the famous Higgs boson. Meanwhile, "matter" particles, like electrons, quarks, and neutrinos, are all fermions. The central difference in behavior is that fermions "take up space"---the famous *Pauli exclusion principle* prevents two fermions from occupying the exact same state---whereas bosons can "pile up on top of each other," as actually happens with (e.g.) lasers and Bose-Einstein condensates.

Anyway, in our 3-dimensional space, there's a famous theorem showing that every particle must be either a fermion or a boson. But that theorem breaks down in 2 dimensions. And indeed, it turns out that in 2 dimensions, you can have particles that are neither bosons *nor* fermions, but exhibit some more complicated behavior when you swap two identical ones. These more general particles are called *anyons*. Physically, they could arise as "quasiparticles"--that is, particle-like excitations in a 2-dimensional medium.

Now, here's what emerged from the work of Freedman, Kitaev, Wang and others twenty years ago: if you could make such quasiparticles in a two-dimensional surface, then just *moving the particles around each other* in a suitable sequence would be enough to do a universal quantum computation. And it gets better: in order to affect the quantum computation's output, it wouldn't be enough to move some quasiparticle a little to the left or the right. Instead, you'd need to change the *topology* of the braiding pattern formed by the particles' worldlines. Because of this "topological" property, this setup might be naturally much more robust to decoherence than more traditional approaches to building a QC, and might require much less error-correction. It might even someday constitute the Holy Grail that we talked about earlier, the "quantum computing analogue of the transistor."

So what's the caveat? Well, we're only now *beginning* to understand how to create even the simplest quasiparticles in the lab. In contrast to (e.g.) the trapped-ion or superconducting approaches, which have by now demonstrated quantum computation with several dozen qubits, it seems fair to say that not even *one* topological qubit has clearly been made yet.

Microsoft is the current world leader in the topological approach to QC, and has made an enormous gamble on that approach, despite the unclear prospects for payoffs in the near future.

So, final question before we end this course: what should we expect from experimental QC in the coming years and decades?

We're not going to do anything as foolish as trying to predict the number of years until various milestones are achieved. But we'll do something else that's hopefully useful as well: articulate three milestones to watch out for, which are often conflated but ought to be kept separate, and which probably need to be achieved one at a time.

<u>Milestone 1: "Quantum Supremacy"---using a programmable QC to solve *some* well-defined problem unequivocally faster than the fastest known methods running on a classical computer.</u>

For obvious reasons, many people dislike the term "quantum supremacy" (which was coined by the physicist John Preskill in 2012), but it seems to have stuck for now. ("Quantum inimitability," "quantum ascendancy," and "quantum advantage" just don't have the same ring!)

Crucially, ~50 qubits, which we hope to have soon, might already be enough to achieve this milestone---and if the qubits are good enough, they won't even need to be error-corrected (which would add a huge overhead). As mentioned above, BosonSampling is one possible route to achieving quantum supremacy, but today it looks likely that quantum supremacy might instead be first achieved using superconducting qubits---with sampling tasks that are conceptually similar to BosonSampling, but different in detail.

<u>Milestone 2: Useful simulations of quantum physics and chemistry</u>

A 2016 Microsoft paper argues that 100-200 logical qubits (or sufficiently well-behaved physical qubits...) would already be enough to let us simulate nitrogen fixation in the Haber process, the chemical reaction that the world uses to make fertilizer. This problem has proven to be intractable for classical computers. And if, using knowledge gained from the quantum simulation, one could design a similar reaction that worked at lower temperatures, that would be worth billions of dollars.

But a big unknown right now is whether doing any useful simulations of this kind would require fault-tolerance, which would of course introduce an enormous overhead.

<u>Milestone 3: A full, scalable, universal, fault-tolerant quantum computer</u>

It's only at this final stage that we could, e.g., implement Shor's algorithm to factor numbers with thousands of digits, and thereby break today's public-key cryptography.

Milestone 1 might be coming in the next few years, but no guesses about milestones 2 and 3!