Lecture 2

Why study quantum computation?

For this course, our main answer will be:

> **There are *certain* problems which quantum computers, using *some* quantum algorithm, can solve faster than classical computers, using *any* classical algorithm.**

Faster here is in terms of the complexity of the algorithm as a function of the input size n. We don't usually mean $n \to 0.5n$ (though these also exist) but more like $n^2 \to n$ (quadratic, or polynomial, speedup) or $2^n \to n^{10}$ (superpolynomial speedup: 2^n cannot be written as a polynomial function of n^{10}). We care much about latter types essentially because quantum is much more expensive on the hardware end: a small speedup is not enough to justify the expenditure. Naturally, we also care more about superpolynomial speedups vs polynomial speedups for the same reason.

To be totally transparent, this is a *belief* that is widely held by those in my field for various theoretical reasons, some of which we will see. It is not a *proven* statement in the mathematical sense, and unlikely to be provable. But this is just like how many similar statements in computer science (e.g., P $\neq$ NP) are unlikely to be provable (though most computer scientists believe that P $\neq$ NP.). Why is it hard to prove? Note the statement has the words "any classical algorithm": proving it requires bounding the amount of ingenuity that classical algorithms can involve – if you've ever been surprised by a clever classical algorithm in algorithms class, you'll appreciate this is hard!

There are other strong answers such as quantum cryptography, quantum science (e.g., physics, chemistry, materials), quantum sensing, philosophy. We will cover some of these too in this course.

Quantum algorithms

Example problems. ["Classical" below refers to "randomized classical", the most general class of classical]

1. SATISFIABILITY: e.g., $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_3 \vee \neg x_2)$ (the fact that's it's a large AND of small ORs is important; a large OR of small ANDs is easy!) Best-known classical $O(2^n)$, best-known quantum $O(2^{n/2})$.

   SATISFIABILITY: very general problem that models any problem that you want to solve brute force by trying all possible solutions.

2. factoring ($15 = 5 \times 3$, $221 = 13 \times 17$, $30743126349163 = 4210601 \times 7301363$ make sure you appreciate this is hard!) Best-known classical $2^{O(n^{1/3})}$, best-known quantum $O(n^3)$, where n is the *number of digits* in the number to be factored. Comment: if N is the number, n is $\Theta(\log(N))$; e.g., $N = 1,000,000$ has $n = 7$ digits; it's important to appreciate this distinction. In particular, the complexities are stated in terms of n and not N.

   Factoring: online security assumes this problem is hard to solve. More discussion: opposite of multiplying which is easy, which is why it's suitable for cryptography: encryption uses multiplication, decryption (without the secret key) uses factoring. This pushed NIST to release new online encryption standards: see press announcement from August 2024. (Optional material: RSA public-key encryption: merchant: choose p, q large distinct prime numbers, x coprime to $(p-1)(q-1)$ and compute y such that $xy = 1 \mod (p-1)(q-1)$; sk $= (N, x)$, pk $= (N, y)$. customer: encrypt $m \in \mathbb{Z}_N$ into ciphertext $c := m^y$. merchant: decrypt by c^x. Claim $c^x = m \mod N$. Proof by Fermat's little theorem.)

3. simulating quantum systems. Problem size is n = number of quantum particles, e.g., electrons. Want to compute some properties of the system after time t, e.g., electron population in a certain state: Best-known classical $O(t2^n)$, best-known quantum $O(t\,\mathrm{poly}(n))$.

   Then talked about hydrogen H_2 molecule. In general this problem can help with discovering new drugs, batteries, and catalysts. A catalyst that has received a lot of attention starting with this paper: FeMoCo, could help with converting nitrogen + hydrogen into ammonia (aka nitrogen fixation) at normal pressures/temperatures. Currently done using Haber-Bosch process which is very high-pressure and high-temperature.

Timeline. Notion of a 2-qubit gate, quantum analogue of a classical 2-bit gate: last time's XORing of the second bit onto the third bit is such a gate (two bits interacted).

- Today: 100s physical qubits, 1000s of two-qubit gates. A single gate acts between two qubits (think of it as the generalization of a Boolean logic gate like AND). Number of gates limited by qubit interacting with environment and losing its quantum state – "decoherance". [Surface code: 1000 physical qubits for 1 logical qubit. qLDPC code: 100 physical qubits for 1 logical qubit. Other tradeoff though, e.g., qLDPC need more classical electronics and wiring.]
- IBM targets 100M gates on 200 logical qubits by 2029 (see here). Google had similar target but seems to have withdrawn their timeframe (the milestones are still up here); last year, Google's Willow chip reported: 105 physical qubits, with two-qubit gate error 0.33% (data source: here) – this means, expect to be able to do $1/0.0033 \approx 300$ gates.

  Not as crazy as it sounds if a "quantum Moore's law" holds (maybe holds? see chart but you can be the judge.)

- Context:
    1. Breaking online security (factoring RSA-2048): 20 million physical qubits, 3 billion two-qubit gates. Gidney and Ekerå, Quantum 2021
    2. Simulating FeMoco: 4 million physical qubits, 5 billion two-qubit gates. Lee,...,Babbush, PRX Quantum 2021