Introduction to Quantum Computing

Lecture 1

History of computer science

- 1936 - Alan Turing developed an abstract model of computation, now called the "Turing Machine". Turing showed there is a "Universal Turing Machine" capable of simulating any other Turing Machine. He claimed that any algorithmic process can be performed on a Turing machine.
- Alonzo Church, building on the work of Turing, made the *Church-Turing Thesis*: Any algorithm performed by any machine can be performed with at most polynomial "overhead" on a Turing Machine. Although this is very vague, and unprovable, it has withstood almost 70 years of attempts to violate it. One key point is that a Turing Machine is *deterministic*.

The Church-Turing Thesis

Whatever can be calculated by machine (with finite data and a finite program) is Turing-machine-computable.

- John von Neumann developed a simple model of how to put together electronic circuits to implement a Turing machine.
- 1947 - John Bardeen, Walter Brittain, and Will Shockley invented the transistor, for which they received the Nobel Prize.

- 1965 - Gordon Moore states Moore's Law: computing power will double for constant cost every 18 months. This has roughly held since then, but should fail in the next few decades due to quantum mechanical effects.
- 1970's - Robert Solovay and Volker Strassen found a randomized primality testing algorithm, which uses random numbers in a fundamental way, challenging the Church-Turing thesis. It seemed that no efficient primality test was possible. [a] Thus we make a new thesis:

The Church Turing Thesis - Take II

Whatever can be calculated by machine (with finite data and a finite program) can be efficiently simulated using a probabilistic Turing machine.

[a] This was changed only recently by Prof. Manindra Agarwal and two of his students, Nitin Saxena and Neeraj Kayal, who in 2002 constructed a polynomial time algorithm to prove primality.

History of quantum mechanics

- Was formalized in the 1920's and 1930's.
- Needed to avoid certain infinities when computing with classical physics, like the "ultraviolet catastrophe", black body emissions, and the photoelectric effect.
- Quantum mechanics is a mathematical framework explaining physical phenomena, and has been extremely successful in the explanation of many experiments. Few physicists doubt its validity in the domain where it applies.
- The rules are simple, but the outcomes are very non-intuitive, causing much initial resistance to the adoption of quantum mechanics. Albert Einstein went to his grave doubting its validity, and had the famous quote "God does not play dice with the universe."[a]
- The rules of quantum mechanics seem to permit "quantum teleportation" of information, spooky "action at a distance", unbreakable encryption, and more.
- Much of quantum computing deals with trying to deepen our intuition about quantum processes, and understand the limits imposed by Nature on "information" and computation.
- On the experimental side, much progress has been made since the 1970's in controlling single quantum states, by trapping single items (photons, electrons, etc.) and manipulating them, in order to probe quantum effects. These techniques have been applied to make simple quantum computers, but these computers are still very much in their infancy.

[a] As we will see, John Bell showed conclusively that God does indeed play dice with the universe. Neils Nohr told Einstein "Don't tell God what to do." And Gregory Chaitin showed "God not only plays dice in physics but also in pure mathematics."

Other computing devices

In recent years, many other types of computing device have been tried, each with strengths and weaknesses. For example

- Analog computers - instead of using discrete digital devices, researchers have used analog devices to do computations. However all analog models of computation, once you allow error, do not offer any computing power over classical machines in all known instances.
- DNA computers - here the replication of DNA in a petri dish offers tremendous parallel processing. It is not known how strong these may be, and as a model are still a Turing machine, although highly parallel.
- Optical computers - some devices, most notable a device to factor integers based on light modulation and spinning colored disks, have some promise, but again are very far from being widespread, and appear to have limited uses.

Quantum computation

The field of quantum computation has come about as a result of many influences. Some are:

- Richard Feynman in 1982 pointed out that classical computers cannot simulate quantum systems, since in general to n quantum particles require 2^n complex numbers to describe. So he suggested building quantum systems to run these simulations.
- In 1985, David Deutsch, trying to derive a Church-Turing Thesis from physical law, was led to study quantum systems. He defined Deutsch's notion of a Universal Quantum Computer, a device capable of efficiently simulating any Turing machine. He then found a problem that his machine could solve efficiently, but a Turing machine could not. It is not known if any physical system can be efficiently simulated on his machine. Some people think Topological Quantum Field Theory (TQFT) computers will be stronger again.
- The shrinking component size in recent chip design is running against quantum effects, so researchers were wondering how to avoid them. It turns out that they might actually help computation, if used properly.
- The most remarkable step so far was the integer factoring algorithm in 1994 by Peter Shor, reducing the best known integer factoring algorithm complexity for an N digit number from $O(e^{cN^{\frac{1}{3}}(\log N)^{\frac{2}{3}}})$ to $O(N^2 \log N \log\log N)$, an exponential speedup. For example, it would require 1000 workstations 10^{29} years to factor a 4096 bit number using GNFS, but take a 100 MHZ quantum computer only 4.8 hours.
- Then Lov Grover found an algorithm that can find a given item in an unsorted table of N items, using $\Theta(\sqrt{N})$ quantum operations, while classically this takes $O(N)$.

- In 1995 Alexi Kitaev noticed these algorithms and others (Deutsch, Simon, Shor, Grover, DLP, order finding, hidden linear function, Abelian stabilizer, ...) fit in a framework called the Hidden Subgroup Problem (HSP).
- It is currently an important problem to determine which groups can have the HSP solved *efficiently* by a quantum algorithm. For example, if the symmetric groups S_n have efficient quantum HSP algorithms, then it would give a polynomial time algorithm for the graph isomorphism problem.
- For quantum computers to work, a lot of error correction is needed. Shor, Gottesman, and others found sufficient methods to do quantum error correction, and along with the study of quantum information theory and quantum noise, we will leave this topic for next semester.
- There was initial interest to show **P**=**NP** using quantum computing (Freedman, others), but it now is the feeling that this is not true, although it is still open how powerful quantum computers are. TQFT computers have a better chance of computing **NP** hard problems in polynomial time, since they perform some **NP** hard computations in polynomial time already (certain Jones polynomials evaluated at n^{th} roots of unity).
- Current research deals with many topics, from implementation detail, to understanding quantum entanglement, to circuit construction, to error correction, to quantum cryptography, to much much more.

An example - Deutsch's algorithm

In 1985 Deutsch conceived the following algorithm (here it is modified slightly and simplified). He was trying to derive a Church-Turing thesis like statement from physical law, and was ultimately led to consider quantum manipulations.

The Deutsch-Josza algorithm

Suppose you have a function on the two elements $f : \{0, 1\} \to \{0, 1\}$, and someone has built a machine that evaluated the function. The question is: how many times do you need to run the machine to know if the function is constant or one-to-one? That is, we want to which type of function f is : $f(0) = f(1)$ or $f(0) \neq f(1)$. If the machine falls under the Church Turing thesis, it requires being run two times, once for $f(0)$ and once for $f(1)$.

David Deutsch found a clever way to answer this question using only *one* execution of the machine, if it is built using quantum mechanics principles. Although this is a toy example, it has been generalized to make a problem that provably requires exponential time on a Turing machine, but only polynomial time on a quantum computer. Here is the calculation.

Suppose we build a quantum machine capable of performing the function f. Since quantum mechanics must be reversible, f needs some extra workspace, so we have the machine which works on two qubits:

$$f : |x\rangle|y\rangle \to |x\rangle|f(x) \oplus y\rangle$$

The algorithm - details

1. Start with a state $|\psi\rangle = |01\rangle$
2. Apply the unitary Hadamard matrix $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ to get the mixed state $\left[\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$.
3. Apply f to this mixed quantum state. Note that f applied to the state $|x\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$ gives the state $(-1)^{f(x)}|x\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$. So we one of the states:
$$\pm\left[\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right] \quad \text{if } f(0) = f(1)$$
$$\pm\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right] \quad \text{if } f(0) \neq f(1)$$
Note that we only ran the machine f one time!
4. Applying H to the first qubit, and only looking at that qubit we obtain the quantum state:
$$\begin{aligned} &\pm|0\rangle \quad \text{if } f(0) = f(1) \\ &\pm|1\rangle \quad \text{if } f(0) \neq f(1) \end{aligned}$$
5. Perform a measurement of the first qubit. We obtain the measurement 0 if and only if $f(0) = f(1)$, and hence the measurement 1 if and only if $f(0) \neq f(1)$.

Thus, with certainty, we can determine whether f is constant or one-to-one, using half as many queries as a classical computer. A neat trick, right? So what? Well, it has been tested experimentally, and does work. And more importantly, Dan Simon of Microsoft Research extended this idea in 1994 to a more interesting case:

Let $f : \{0,1\}^n \to \{0,1\}^m$ for $m \geq n$. Compute s where $f(x \oplus s) = f(x)$ for all $x \in \{0,1\}^n$, or return 0^n otherwise. This has exponential complexity in n on any classical device, yet has polynomial complexity on a quantum computer.

Thus, there exist problems a quantum computer can solve efficiently that no classical computer will ever solve!

Course Layout

1. Cover the necessary linear algebra and notation used in the literature.
2. Cover the postulates of quantum mechanics.
3. Cover some Theory of Computation.
   - Definition of Turing machine.
   - Definition of some complexity classes: **P**, **NP**, **PSPACE**, **BPP**, others.
   - $O(f(n))$, $\Theta(f(n))$, $\Omega(f(n))$ notation.
   - Ccomplexity theory results: comparison sorting, **3-SAT**, graph isomorphism, factoring.
4. Cover quantum computer models.
   - Deutsch's Quantum Turing Machine (QTM).
   - Yao's quantum circuit model.
   - Equivalence of different models, and some open questions.
   - Simple circuits.
   - Quantum circuits can emulate Turing Machines.
   - open problems and newer models : topological quantum field theories?
5. The big ones: Shor's algorithm and Grover's algorithm.
   - RSA encryption, relevance of factoring, discrete log problem (DLP).
   - The quantum Fourier transform in $O(\log^2 N)$ steps.
   - Simple extensions like min and max.

6. The Hidden Subgroup Problem (HSP)
   - Almost all known algorithms that are better than are in this formalism.
   - Certain groups known, many open problems.
   - I plan to spend some time on this if possible, since it has significant theoretical importance for the future of computing.
7. Future directions - TQFT computing and the Jones polynomial connection.
8. Next semester (if possible)
   - Review of quantum computing principles, including some more algorithms.
   - Introduction to classical information theory, including
     - Entropy and information measurement.
     - Error correcting codes.
   - Quantum information theory, including
     - Quantum noise models.
     - Quantum data compression.
     - Entropy and information measurement.
     - Quantum error correcting codes.

For next time

- I plan to put these notes on the web, and if all goes well, they will be ready before the next meeting. I will post them at `www.math.purdue.edu/~clomont/QC/QC.html`.
- I will post references to papers and books online also.
- Next time we will cover some linear algebra and the notation used by quantum computing researchers (that is, the Dirac bra and ket notation).
- If we have time, we will start an axiomatic introduction to the quantum mechanics necessary to understand quantum computation.

References

- The bible of quantum computing is *Quantum Computation and Quantum Information*, Michael Nielsen and Isaac Chuang, Cambridge University Press, 2000, ISBN 0-521-63503-9. We will follow a lot of their organization, with some topics skipped or reordered. Almost everything covered in this seminar is in their book.
- Searching the `quant-ph` archive at vb+xxx.lanl.gov+. This has most of the papers on quantum computing archived.
- A Google search will reveal a lot more.