Introducing Quantum Computing

Christopher Granade

October 9, 2007

1 Qubits.

For the purposes of quantum computing, it's very often easiest to only work with potentials which allow only have two "good states" for the Hamiltonian (a fancy way of saying the energy of a system). We'll call these $|0\rangle$ and $|1\rangle$. Fancy names, huh? A particle in such a potential is said to encode a single *qubit*.

1.1 Superposition and measurements.

One of the axioms of QM is that if $|0\rangle$ and $|1\rangle$ are states for some particle, then for a pair of complex numbers a and b, $a|0\rangle + b|1\rangle$ is also a state as long as $a^2 + b^2 = 1$. This is what's commonly known as *superposition*, and is one of the most confusing things in all of QM. Part of the confusion comes from statements like "it's in both places at once!" Well, yes, if you're careful about what "is" means here. As soon as you look at the particle, you *measure* it, which means to project it onto a basis.

If you've had linear algebra, then you'll recognize the words "project" and "basis." For any measurement (every way of looking at a particle), there's a set of "good states" that give discrete values. For instance, if $|\psi\rangle = a|\phi_1\rangle + b|\phi_2\rangle$, and if $|\phi_1\rangle$ and $|\phi_2\rangle$ are the good states that give P_1 and P_2 when you perform some measurement Φ on it, then you'll get P_1 with probability $|a|^2$ and you'll get P_2 the other $|b|^2$ of the time. So, yes, the particle is "in both places at once," but you only ever see it in one at a time. After you see it, the state is fixed to whatever state gave you your measurement. If you saw P_1, then $|\psi\rangle$ becomes $|\phi_1\rangle$. This is what's known as the waveform collapse, and also is the subject of a great many horrible sci-fi plots.

1.2 Vector notation.

A convenient way to write the state of a qubit is to use a 2 by 1 vector: $|\psi\rangle = \begin{bmatrix} a \\ b \end{bmatrix} = a|0\rangle + b|1\rangle$. This allows us to write down gates (we'll come to this later) as matrix transformations of qubit vectors, much like how matrices can be used in computer graphics to scale, rotate and translate geometric objects.

1.3 Multiple qubits.

When we have two or more qubits, we write the state of the entire *system* like $|\phi\rangle |\psi\rangle$. It looks like we multiplied the states, but it's really something called the tensor product. We don't need to worry about that aside from knowing that when we put the states next to each other like that, it means a system of multiple qubits with each in their own states. Sometimes, though, we get even lazier and write $|\phi\psi\rangle$. That's really the same thing as $|\phi\rangle |\psi\rangle$.

2 Quantum gates.

We can do things to a quantum state without actually looking at the state. To use a classical analogy (which is always dangerous – don't do it unsupervised), if we have a NOT gate in the middle of a huge circuit, we don't actually have to look at the voltage on the wire coming out. We only care about the final answer coming out of our circuit. in the same way, subject to a lot of caveats, we can change things about a qubit or system of qubits without performing a measurement.

The processes which change qubits this way are called *quantum gates*. As mentioned before, these gates can all be written as matrix transformations of qubit vectors, much as rotation and translation matrices transform points in computer graphics. In fact, there exist gates to "rotate" the phase of a state, and to shift a state between $|0\rangle$ and $|1\rangle$ using a rotation matrix.

2.1 Single qubit gates.

Just as with classical circuits, there are gates that modify single qubits. As opposed to classical systems, in which only two interesting single-bit gates are defined (NOT and IDENTITY), in quantum circuits, the extra degrees of freedom offered by superposition allows for more interesting gates to be defined. Among these are the Pauli spin gates:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

X is often called the quantum NOT gate, for reasons that become obvious when applying it to $|0\rangle$ and $|1\rangle$:

$$\begin{aligned} X|0\rangle &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle \\ X|1\rangle &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle \end{aligned}$$

The Z gate is often called the phase flip gate, as:

$$Z\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Y does not get as much attention for its own part. Other single-qubit gates include the Hadamard gate $H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$, the phase gate $S = \begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}$ and the $\pi/8$ gate $T = \begin{bmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{bmatrix}$.

2.2 Multiple qubit gates.

When we're working with multiple qubits, we can write them down as a single vector:

$$|0\rangle|0\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \qquad |0\rangle|1\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$$
$$|1\rangle|0\rangle = \begin{bmatrix}0\\0\\1\\0\end{bmatrix} \qquad |1\rangle|1\rangle = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

This allows us to write down gates of multiple qubits as matrix transformations of these vectors. Of all the multiple qubit gates, however, we will only look at one: the CNOT, or Controlled-NOT gate:

$$U_{CN} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}$$

The CNOT gate acts as follows on our basic qubits:

$$|00\rangle \to |00\rangle \qquad |01\rangle \to |01\rangle$$
$$|10\rangle \to |11\rangle \qquad |11\rangle \to |10\rangle$$

It can be shown that all other multiple qubit gates, for any arbitrary number of qubits, can be written as the composition of single qubit gates and of CNOT. Another way of saying this is that CNOT, along with single qubit gates, are universal for quantum computing. There's still a lot of other gates that get used, however, in the same way that we discuss more classical gates than just NAND.

2.3 Gates we wish we could build.

In classical computing, we so often use a very simple gate that we hardly even recognize we're using it at all: the FANOUT gate. FANOUT copies an input signal

to several output wires, and can be used to split a bit into multiple copies. Sadly, no such gate can ever exist for a qubit via a useful if depressing result called the Quantum No-Cloning Theorem. In general, all quantum gates must be *reversible*. The mathematical formalism for this requires that the matrices representing gates be *unitary*. What exactly this means is beyond the scope of this lecture, but it is important to realize that there are quite a few ways in which quantum computers are harder to work with. Operations that we take for granted in classical computing don't always exist for qubits.

It's actually not too hard to see that the classical FANIN gate should also be beyond the reach of quantum computers for the same or similar rationales; once we use a classical FANIN gate to throw away bits, the bits are gone. Such a gate clearly isn't reversible, and so we can't make one for qubits. A technical argument can be made as well, but would be beyond the scope of this lecture.

3 Entanglement.

Entanglement is where a lot of the real power of quantum computers comes from. To talk about entanglement, it's easiest to write down a set of states that are entangled and then talk about what it means. These states are a famous set called the *Bell states*:

$$\begin{aligned} |\beta_{00}\rangle &= \frac{|00\rangle + |11\rangle}{\sqrt{2}} \\ |\beta_{01}\rangle &= \frac{|00\rangle - |11\rangle}{\sqrt{2}} \\ |\beta_{10}\rangle &= \frac{|10\rangle + |01\rangle}{\sqrt{2}} \\ |\beta_{11}\rangle &= \frac{|01\rangle - |10\rangle}{\sqrt{2}} \end{aligned}$$

These states have the property that measuring one of the qubits changes the state of the other. For example, if you measure the first of two qubits in the $|\beta_{00}\rangle$ state and obtain $|0\rangle$, then you know immediately that the second qubit is also a $|0\rangle$.

4 Why Do We Care?

In this section, I'll discuss a few of the reasons why we care about quantum computing. Mostly, I'll just skim the surface, and mention what an algorithm can do without any details as to how. Of course, I won't be mentioning anywhere near all of the awesome things that quantum computers can do. For more of these, read *Quantum Computation and Quantum Information*, by Michael A. Nielsen and Issac L. Chuang (amazon, b&n).

4.1 Superdense Coding.

Perhaps the easiest application of quantum computing to understand is the application of superdense coding. Using superdense coding, Alice and Bob can exchange two classical bits by sending a single qubit. If Alice and Bob each have one qubit from an entangled pair in the $|\beta_{00}\rangle$ state (which can be done without direct communication if they both receive one from a third party), then Alice can apply one of four different single-qubit gates to her half of the pair, depending on which pair of classical bits she wants to send. She then sends her qubit to Bob, who measures the pair, who can then find out which of the four Bell states the pair was in. This works because, depending on which gate Alice chooses, changing her qubit has a measurable effect on the pair:

$$\begin{aligned} I\,|\beta_{00}\rangle &= |\beta_{00}\rangle \\ Z\,|\beta_{00}\rangle &= |\beta_{01}\rangle \\ X\,|\beta_{00}\rangle &= |\beta_{10}\rangle \\ iY\,|\beta_{00}\rangle &= |\beta_{11}\rangle \end{aligned}$$

4.2 Shor's Algorithm.

Of all the algorithms invented for quantum computers, few are as plain scary as Peter Shor's algorithm for factoring integers (published as arxiv:quant-ph/9508027). As opposed to classical algorithms for factorization, of which the best takes $O\left(\exp\left[\sqrt[3]{\frac{64}{9}\lg n\cdot(\lg\lg n)^2}\right]\right)$ time to factor an integer n, Shor's algorithm can factor numbers in $O\left([\lg n]^3\right)$ time and $O(\lg n)$ space[1].

Shor's Algorithm works by reducing the problem of factorization to the problem of period finding (this step may be done classically), and then applying the Quantum Fourier Transformation (QFT) to find the period of the reduced problem instance. The details of this reduction are better suited to a course on number theory, but what is essential to understand is that Shor's Algorithm is a specific example of a general class of quantum algorithms based on the QFT.

Since factorization is essential to most modern cryptosystems, the power of Shor's Algorithm to factor numbers in polynomial time (as a function of the number of bits $\lg n$) is why I said that the algorithm scares quite a number of people. Of course, no one has yet factored a number larger than 15 using a quantum computer, so we do have a bit of time in which to upgrade to elliptical curve based cryptosystems.

4.3 Grover's Algorithm.

The last algorithm to be mentioned is a general search algorithm called Grover's Algorithm after its inventor. By general, what we mean is an algorithm that

[1] For more details on the time and space complexity of classical factorization algorithms, please refer to Wikipedia's treatment of the subject.

searches a list for an item specified by some other circuit (that we call an *oracle*) that tells the algorithm when the item has been found. Thus, by changing the oracle, we can make the search find whatever kind of item we want. What's truly amazing, however about the algorithm, is that it takes $O(\sqrt{n})$ queries to the oracle to find the special item, where n is the number of items in the list. Classically, the idea of performing a search on unsorted data in less operations than the a linear function of the list size would be patently insane, as one would expect that the computer must at least look at each item. Yet, this is precisely what Grover's Algorithm lets you do for lists encoded with qubits. Superposition allows the quantum computer to, in some limited fashion, to look at multiple items together.

5 A Parting Note: Complexity Classes.

The algorithms that I've laid out here don't quite make the case that quantum computers are ridiculously useful; even with Shor's Algorithm, if the constant multiple of the order is large enough, then the sub-exponential classical algorithms are still more attractive. Thus, we turn to the study of complexity theory, which is largely dedicated to answering the question of what a given computational model can do at all, and what it can do quickly. I won't talk about it any more here, other than to make two points: we still don't know $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$, and the class of things that can be done quickly with quantum computers (BQP) is not known to be different from the class of things that can be done quickly with classical computers (P). For those interested, Complexity Zoo or its spin-off, Petting Zoo, are excellent resources hosted by Caltech's Qwiki. For a list of problems and their complexity classes, Qwiki's Complexity Garden is a good place to start.