A 20 Minute Introduction to Quantum Computing

Emma Strubell, University of Massachusetts Amherst, February 2018

1 Introduction

Companies including IBM, Google, Microsoft, Huawei, and Intel all have research groups working on quantum computing. This is because researchers in computational complexity theory have shown that quantum computing is a fundamentally more powerful computational paradigm than "classical" computing, i.e. the type of computing that we're familiar with, developed by Alan Turing and George Boole. If we had a working quantum computer, we would all of a sudden be able to solve previously computationally infeasible problems with applications across many fields including physics, chemistry, machine learning, and computer security.

The main problem that quantum computers solve is efficiently considering all, exponentially many possible configurations of N binary variables. A natural application of quantum computing is simulating quantum mechanical systems, a common desire in physics and chemistry research, with models that are also used in machine learning. Consider a small system consisting of 300 particles, each of which may have either positive or negative charge. The number of possible configurations of that system (each particle having positive or negative charge, or in the case of machine learning each variable having value 0 or 1) is $2^{300} \approx 10^{90}$, which is about 10 orders of magnitude more than the estimated number of particles in the known universe.[1] Using our current computing hardware, it is infeasible to enumerate all of those possibilities, yet in order to perform a simple query, such as computing the most likely configuration, we would have to enumerate all these possible configurations. Using classical computing hardware, this problem requires an amount of computation that is *exponential* in the size of the system (number of particles). Using a quantum computer, we could compute the answer to this query with an amount of computation that scales *linearly* with the size of the system, making the computation feasible.

Unfortunately in 20 minutes we do not have enough time to cover the full pre-requisites for understanding how quantum algorithms work[2], but my hope is that you come away from this short lesson with a basic understanding of (1) *qubits* and *quantum gates*, the building blocks of quantum computing, and (2) the fundamental ways in which quantum computation differs from classical computation, namely: *quantum superposition* and *entanglement*.

2 Classical computing: Bits, registers and logic gates

Before I describe the paradigm of quantum computing, I want to review a little bit about how classical computing works.

We're used to thinking of computer algorithms in terms of really high-level constructs such as maps, arrays, trees, and floating-point math. But to date, quantum algorithms operate at a much more primitive level: Quantum bits, registers and gates. So before discussing the quantum versions of these things, I want to establish a basis for how these work in classical computing.

At the base of all the complex algorithms we write are some registers filled with binary values in the CPU. So, each bit can be either 0 or 1, and an 8-bit register can represent $2^8 = 256$ different

[1] https://en.wikipedia.org/wiki/Elementary_particle

[2] Check out my full tutorial, aimed at undergraduates in computer science, here: https://bit.ly/2DsijHQ

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

Table 1: Truth table for the AND gate.

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

Table 2: Truth table for the OR gate.

values. We can then perform computations using those values by applying logic gates, which are essentially functions taking in and producing as output binary values. We can define their behavior using a truth table. For example, the AND gate takes in two bits and returns 1 if both bits are 1. The truth table is depicted in Table 1. The OR gate returns 1 if either of its inputs are 1. Its truth table is given in Table 2. We can apply the gates element-wise to two registers of n bits to perform larger computations, e.g.:

$$\boxed{100} \vee \boxed{010} = \boxed{110}$$

and this forms the basis for all the complex code and programs that run the world today.

3 Probabilistic bits

In the above example, we had two exact settings of bits in the registers, with exactly one possible output when applying the OR gate. Now, consider a computing paradigm where, rather than the contents of a register being fixed, it is instead represented by a probability distribution over all possible binary configurations of the register. A single bit would have two probabilities, non-negative real numbers a_0 and a_1 corresponding to the values 0 and 1, with $a_0 + a_1 = 1$:

$$a_0\boxed{0}, \quad a_1\boxed{1}$$

Taking the 3-bit example from before, we would now have a probability a_i assigned to each of the $2^3 = 8$ possible configurations:

a_0	000
a_1	001
a_2	010
a_3	011
a_4	100
a_5	101
a_6	110
a_7	111

With $\sum_{i=0}^{7} a_i = 1$.

Now consider how this effects the application of the gate. Suppose we have two probabilistic bits, p and q, and we wish to compute a probabilistic OR between them. We can compute the probability for each possible output by multiplying the probabilities of the possible inputs. We can visualize this by adding probabilities to our truth table (Table 3). Concretely, let's assign some

p		q		p ∨ q	
a_0	0	b_0	0	a_0b_0	0
a_0	0	b_1	1	a_0b_1	1
a_1	1	b_0	0	a_1b_0	1
a_1	1	b_1	1	a_1b_1	1

Table 3: Probabilistic truth table for the OR gate.

probabilities. Let $a_0 = \frac{3}{4}$, $a_1 = \frac{1}{4}$, $b_0 = \frac{3}{4}$, $b_1 = \frac{1}{4}$. The results are listed in Table 4. Given these

p		q		p ∨ q	
$\frac{3}{4}$	0	$\frac{3}{4}$	0	$\frac{9}{16}$	0
$\frac{3}{4}$	0	$\frac{1}{4}$	1	$\frac{3}{16}$	1
$\frac{1}{4}$	1	$\frac{3}{4}$	0	$\frac{3}{16}$	1
$\frac{1}{4}$	1	$\frac{1}{4}$	1	$\frac{1}{16}$	1

Table 4: Probabilistic truth table for the OR gate with $a_0 = \frac{3}{4}$, $a_1 = \frac{1}{4}$, $b_0 = \frac{3}{4}$, $b_1 = \frac{1}{4}$.

probabilistic bits, though there is no one final output of this gate, we can compute the probability of the two possible outputs: The probability of the output being 1 is equal to $\frac{3}{16} + \frac{3}{16} + \frac{1}{16} = \frac{7}{16}$, and the probability of the output being 0 is equal to: $\frac{9}{16}$. So the output is more likely to be 1. Now, you can imagine a computational paradigm where, rather than designing an algorithm which, given concrete inputs is guaranteed to produce a concrete output, instead you design an algorithm which, given probabilistic inputs, produces probabilistic outputs, but you can guarantee that it will produce the correct answer with probability above a certain bound.

4 The qubit

The concept of a qubit is a fairly straightforward extension of the probabilistic bits just described. The single difference, with many implications, is that rather than *real number probabilities*, qubits have associated with them *complex* numbers called *probability amplitudes*. Rather than summing to 1 like probabilities, amplitudes are restricted such that their *absolute values squared* must sum to 1.

We typically use vectors to represent qubits. The most common parameterization, called the *computational basis*, represents each qubit in a register of size n as a 2^n-dimensional vector. I will denote a qubit with the value 0 as $|0\rangle$ and that with value 1 as $|1\rangle$, following the typical bra-ket notation[3] :

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

5 Negative probabilities and the superposition principle

The first distinguishing trait of a quantum system is known as *superposition*, or more formally the *superposition principle of quantum mechanics*. Rather than existing in one distinct state at a time, a quantum system is actually in all of its possible states at the same time. Denoting the amplitudes for each of the possible values as α_i, we write the configuration like this:

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle = \alpha_0 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \alpha_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

A common configuration is for all the states to have equal probability, which would look like this:

$$|\psi\rangle = \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Once the system is observed (measured), it collapses into an observable, definite classical state. The actual value observed may be any state with non-zero amplitude, with the probability of observing a given state given by the absolute value of its squared amplitude.

6 A quantum gate: The Hadamard transform

There are a small number of frequently used quantum logic gates analogous to classical logic gates such as AND or OR in the role that they play in the field of quantum computing. One of the most fundamental gates is the *Hadamard* operator:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Often called the "fair coin flip," the Hadamard operator applied to a qubit with the value $|0\rangle$ or $|1\rangle$ will induce an equal superposition of the states $|0\rangle$ and $|1\rangle$, an equal probability of the qubit being in the state $|0\rangle$ or $|1\rangle$ when observed. Many quantum algorithms begin by applying the Hadamard operator to each qubit in a register, which gives each of the 2^n possible bitwise configurations of the n qubits an equal probability of 2^{-n} of being observed when the system is measured:

$$H |0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1+0 \\ 1+0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

[3] Details on this notation, which simplifies the vector representations of qubits, are outside the scope of these notes.

$$H|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1+0 \\ 0-1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \end{bmatrix} - \frac{1}{\sqrt{2}}\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The difference between amplitudes and probabilities becomes apparent when operations are applied to quantum systems which may have the same probability distributions, but different amplitudes from which those probabilities are derived. The two above states have the same probabilities – that is, either 0 or 1 is equally likely, with probability $\frac{1}{2}$, to be observed. But, their amplitudes differ – the amplitude associated with observing 1 is $-\frac{1}{2}$. When we again apply the Hadamard transform to these seemingly equivalent values:

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1+1 \\ 1-1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$H|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1-1 \\ 1+1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

We obtain completely different results. Information encoded in the amplitudes (called the *phase*), which can't be obtained simply by observing the system in its previous state, led to very different outcomes following the application of the same operator.

7 Quantum entanglement of qubits

Another defining factor of quantum systems over classical ones is that they may exhibit *entanglement*. A state is considered entangled if it cannot be decomposed into its more fundamental parts. In other words, two distinct elements of a system are entangled if one part cannot be described without taking the other part into consideration. In a quantum computer, it is possible for the probability of observing a given qubit to depend on the probability of observing another qubit, and observing one qubit will determine the value of the other. An especially interesting quality of quantum entanglement is that elements of a quantum system may be entangled even when they are separated by considerable space. Quantum teleportation, an important concept in the field of quantum cryptography, relies on entangled quantum states to send quantum information adequately accurately and over relatively long distances.

Now that you know at a high level how qubits behave mathematically, we can look at an example of two entangled qubits to see a simple example of how entanglement works.

Consider the following quantum state[4] composed of two qubits, where A and B denote two separate qubits possessed by Alice and Bob, respectively:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \end{bmatrix}_A \otimes \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \end{bmatrix}_B + \frac{1}{\sqrt{2}}\begin{bmatrix} 0 \\ 1 \end{bmatrix}_A \otimes \frac{1}{\sqrt{2}}\begin{bmatrix} 0 \\ 1 \end{bmatrix}_B$$

Though Alice and Bob both have the same probability ($\frac{1}{2}$) of observing either $|00\rangle$ or $|11\rangle$, these two qubits are in an entangled state. Due to quantum superposition, if Alice observes her qubit,

[4] This is actually one of four special 2-qubit states exhibiting entanglement called the Bell states.

this immediately determines the state of both qubits, and Bob will observe the same value as Alice. To see how this works, suppose that Alice observes her qubit, and it has value $|00\rangle$:

$$\begin{aligned}
&1 \begin{bmatrix} 1 \\ 0 \end{bmatrix}_A \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \end{bmatrix}_B + 0 \begin{bmatrix} 0 \\ 1 \end{bmatrix}_A \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \end{bmatrix}_B \\
&= \begin{bmatrix} 1 \\ 0 \end{bmatrix}_A \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \end{bmatrix}_B + \begin{bmatrix} 0 \\ 0 \end{bmatrix}_A \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \end{bmatrix}_B \\
&= \begin{bmatrix} 1 \\ 0 \end{bmatrix}_A \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \end{bmatrix}_B + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ 0 \end{bmatrix}_A \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}_B
\end{aligned}$$

By observing her qubit, Alice collapses the entire entangled system, determining Bob's qubit, which will always be the same as Alice's.

7.1 Aside: Tensor products

Multiple qubits interact through *tensor* (or *Kroenecker*) products, a generalization of the outer product denoted with the symbol $\otimes$. This operation differs from the typical matrix multiply:

$$\mathbf{u} \otimes \mathbf{v} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \otimes \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} u_1 v_1 \\ u_1 v_2 \\ u_1 v_3 \\ u_2 v_1 \\ u_2 v_2 \\ u_2 v_3 \\ u_3 v_1 \\ u_3 v_2 \\ u_3 v_3 \\ u_4 v_1 \\ u_4 v_2 \\ u_4 v_3 \end{bmatrix}$$