1 Introduction to Quantum Computing

In the following chapter, we will delve into the basics of quantum computing. Before proceeding, it is important to realize that quantum computing is a fundamentally different *model of computation* than all computer systems currently on the market. Much like the first programmable computer was realized by Alan Turing far before he ever saw any practical realizations of his machines, we too study quantum computing at a time when only the most rudimentary quantum computers are available in practice.

The quantum mechanical properties upon which quantum computers are based have been studied extensively by physicists; we will avoid discussing such details and instead take these properties for granted in order to focus on the information-theoretic behavior of this new model of computation.

Due to the complex nature of quantum mechanics, there are a plethora of misconceptions about quantum computing. One of the most widespread is that quantum computers gain computational speedups by *trying all possible solutions at once.* This is simply not true. Perhaps my favorite that I have heard upon starting my thesis is that *quantum computers are like classical computers but in trinary.* False, we cannot efficiently simulate a quantum computer on a classical one. Lastly, probably most confusing is that *quantum computers are faster at all tasks compared to classical computers.* Quantum computers are faster than classical computers at some tasks, such as searching an unstructured list, but they have the same asymptotic runtime[1] as classical computers for other tasks. Now that I have outlined some common misconceptions, hopefully this introduction will help codify the true properties quantum computers.

1.1 Information Representation

Classical computers, i.e. the computers that we all know and love, run on *bits* or 1's and 0's. This is the fundamental unit of information in classical computers. In quantum computers, information is built upon an analagous concept, the *quantum bit* or *qubit.* We will discuss the defining properties of qubits, some of which may seem very different from those of bits.

In classical computing, each bit can be in one of two states–1 or 0. We denote *quantum states* by $|0\rangle$ and $|1\rangle$, pronounced "*ket zero*" and "*ket one*," respectively. This follows traditional *Dirac notation*[2]. Unlike classical bits, qubits can be in a *superposition* between these two states. We describe a quantum state as follows: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers and $|\alpha|^2 + |\beta|^2 = 1$. This describes

[1]We will discuss what this means exactly later

[2]See reference on notation

a probability distribution over the quantum states $|0\rangle$ and $|1\rangle$ with *amplitudes* α and β. We can also think of $|\psi\rangle$ as a vector in a two-dimensional complex vector space, say $\mathbb{C}^2$. The states $|0\rangle$ and $|1\rangle$ form an orthonomal basis in this complex vector space and are called the *computational basis states*.

So how can we know which state a bit or qubit is in? In classical computing, we can simply read the bit to determine whether it's a 0 or a 1. Qubits are a little trickier. We cannot read $|\psi\rangle$ to determine its exact amplitudes, α and β. As soon as we measure $|\psi\rangle$, the superposition will collapse to either $|0\rangle$ with probability $|\alpha|^2$ or $|1\rangle$ with probability $|\beta|^2$. In other words, $|\psi\rangle$ is like a weighted coin that we can flip once to get either heads or tails, but we have no way to directly measure the bias in the coin. Unlike a coin, as soon as $|\psi\rangle$ has been measured to reveal some quantum state, $|\psi\rangle$ will permanently collapse to that state, i.e. $|\psi\rangle = 1|0\rangle + 0|1\rangle$ or $|\psi\rangle = 0|0\rangle + 1|1\rangle$. Every time we measure it thereafter, we will observe the same state that we measured the first time. In the coin analogy, it's like we have a weighted coin that we can flip once to reveal heads or tails, and every subsequent flip always yields the initial state that we flipped to.

You may be wondering why this is useful. It seems impossible to build a model of computation on a fundamental unit of information that is unknowable, immeasurable. The beauty of quantum computation lies in the *manipulation* of these immeasurable qubits such that by the time we measure them at the end, the result will inform us of the state they started in. Simply measuring the qubits before doing any transformations is fundamentally the same as running a random number generator on a classical computer and trying a random possible solution. Obviously, this is not very useful, so we must *transform* the qubits to say anything intelligent about the end result that we measure.

1.2 Quantum Logic Gates

As we already glossed over, $|0\rangle = 1|0\rangle + 0|1\rangle$ and $|1\rangle = 0|0\rangle + 1|1\rangle$. We sometimes represent these states as column vectors of the amplitudes:

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

Thus, we can represent a transformation on qubits by a 2-by-2 matrix where the first column is the image of $|0\rangle$ and the second column is the image of $|1\rangle$ under the transformation. For example, the quantum NOT gate can be realized as a matrix:

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

This takes a state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and transforms it into $X|\psi\rangle = \beta|0\rangle + \alpha|1\rangle$. Alternatively, by matrix multiplication we see that

$$X|\psi\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}.$$

So, any transformation on a single qubit can be represented by a 2-by-2 matrix. What about the transformation that, when given $|0\rangle$ or $|1\rangle$, outputs an *equal superposition* of $|0\rangle$ and $|1\rangle$? This would essentially give us a fair coin. We might represent that as follows:

$$\widehat{H} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

There are a few problems with this transformation. First, applying this transformation $\widehat{H}|0\rangle = 1|0\rangle + 1|1\rangle$, which means that $|\alpha|^2 + |\beta|^2 = 1^2 + 1^2 \neq 1$. So, we must *normalize* the matrix, to get

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

Now, applying the transformation to our basis vectors, we get $\widehat{H}|0\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and $\widehat{H}|1\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. There is still a problem with this transformation. All quantum transformations must be *reversible*, but we have mapped the two basis vectors to the same state, so the transformation is not reversible. To fix this, we will simply change the image of this transformation under $|1\rangle$:

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

This is called the *Hadamard transform* or *Hadamard gate*. It acts on the basis vectors as follows: $H|0\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and $H|1\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$. Both of these states are in an equal superposition between $|0\rangle$ and $|1\rangle$, but they are distinct states. These states come up rather often, so they have been given the special shorthand notations $|+\rangle$ and $|-\rangle$, respectively.

Definition.

$$|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2} \text{ and}$$
$$|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}.$$

The requirements that quantum transformations be reversible and normalized are encapsulated by the property that the matrix representation for any transformation must be *unitary*. That is, $U^\dagger U = I$ where $U^\dagger$ is the transpose of the complex conjugate of U and I is the 2-by-2 identity matrix. Within those requirements, we can construct any transformation we like. What about transforming multiple qubits?

1.3 Multiple Qubits

To represent multiple qubits, we expand our two quantum states to many using tensor products [3]. For example, in a two qubit system, we have the following basis states:

$$|0\rangle \otimes |0\rangle,\ |0\rangle \otimes |1\rangle,\ |1\rangle \otimes |0\rangle,\ \text{and } |1\rangle \otimes |1\rangle.$$

These four states can equivalently be written as:

$$|00\rangle,\ |01\rangle,\ |10\rangle,\ \text{and } |11\rangle.$$

Thus, any two-qubit state can be represented as a linear combination of these basis states, namely $|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$. In general, an n-qubit state can be represented as

$$|\psi\rangle = \sum_{i \in \mathbb{Z}_2^n} \alpha_i |i\rangle,$$

where i is the binary representation of the numbers $0, \ldots, n-1$. The probability of observing a state $|i\rangle$ upon measuring $|\psi\rangle$ is $|\alpha_i|^2$.

We may measure qubits individually as well. For example, measuring just the first qubit gives us 0 with probability

$$\sum_{i \in \mathbb{Z}_2^{n-1}} |\alpha_{0i}|^2,$$

leaving the post-measurement state

$$|\psi'\rangle = \frac{\sum_{i \in \mathbb{Z}_2^{n-1}} \alpha_{0i} |\alpha_{0i}\rangle}{\sqrt{\sum_{i \in \mathbb{Z}_2^{n-1}} |\alpha_{0i}|^2}}.$$

1.4 Entangled States

Something that follows from the above equation but which is not self-evident is the concept of *entangled states*. Consider the following state:

$$|\beta_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

Suppose we measured the first qubit. We will observe $|0\rangle$ with probability $1/2$, and the resulting post-measurement state is: $|\beta'_{00}\rangle = |00\rangle$ (by the equation above). How can this be? We only measured the first qubit, and yet, we now have information

[3] See appendix on tensor products

about both qubits. This is precisely because the state is entangled. Another example of an entangled state is:

$$|\beta_{01}\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}$$

These two states, $|\beta_{00}\rangle$ and $|\beta_{01}\rangle$, are the first two *Bell states* or *EPR pairs*. Quantum entanglement is a powerful computational tool and will be used in many quantum algorithms in the rest of the text.

Now that we have the means to represent an n-qubit state, the state transformations can be represented by 2^n-by-2^n matrices. In particular, the n-qubit Hadamard transform is $H^{\otimes n} = H \otimes H \otimes \ldots \otimes H$, i.e. n tensor products.

1.5 Universal Quantum Gate

In classical computation, there are three basic logic gates: NOT, AND, and OR. We can combine these gates to represent any possible logical expression. Furthermore, the NAND gate and the NOR gate, which we get from taking the negation (NOT) of the output of AND and OR respectively, are said to be *universal gates*. This means that we can construct the three basic logic gates just from NAND gates or just from OR gates. So, NAND and NOR are universal in that any logical expression can be represented with just NAND or NOR circuits. This is very useful in practice because NAND gates are very cheap to construct, so using only NANDs can keep the cost of a computer chip down.

A natural question, then, is whether there exists a quantum analogue, a universal quantum gate. The short answer is that there is no one universal quantum gate, however the following three gates are enough to make an *arbitrary approximation* of any quantum gate: Hadamard, CNOT, and phase gate.

The phase gate is defined

$$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \tag{1}$$

And the CNOT, or controlled-NOT gate acts on two qubits by flipping the second qubit if and only if the first qubit is a 1.

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2}$$

As you can see, by multiplying the CNOT matrix by the column vector $(a \quad b \quad c \quad d)$ this takes $a|00\rangle+b|01\rangle+c|10\rangle+d|11\rangle$ and transforms it into $a|00\rangle+b|01\rangle+c|11\rangle+d|10\rangle$, thereby swapping the amplitudes on the states $|10\rangle$ and $|11\rangle$, or equivalently "flipping" the second bit of the state if the first bit is a 1.

The *gate complexity* of a unitary transformation is the minimum number of gates needed to implement the circuit.

1.6 No-Cloning Theorem

Suppose my friend has a secret state $|\psi\rangle$ that she wants me to have a copy of. Classically, we could easily build a circuit to copy her state $x = 0$ or $x = 1$ without measuring it. We would simply initialize a temporary register to 0, and our circuit would write $x \oplus 0$, the XOR of x and 0, to the destination register.

The quantum case is a bit trickier. Suppose $|\psi\rangle$ is only known to our friend. We wish to copy this exact state into a target slot, which starts out in some standard known state $|s\rangle$. Thus, the initial state of our copying machine is $|\psi\rangle \otimes |s\rangle$. Now, we apply some unitary operation U to these registers to obtain

$$|\psi\rangle \otimes |s\rangle \xrightarrow{U} U(|\psi\rangle \otimes |s\rangle) = |\psi\rangle \otimes |\psi\rangle \tag{3}$$

Now, suppose this quantum copying circuit works for two arbitrary states $|\psi\rangle$ and $|\phi\rangle$. Then, we have

$$U(|\psi\rangle \otimes |s\rangle) = |\psi\rangle \otimes |\psi\rangle \tag{4}$$

and

$$U(|\phi\rangle \otimes |s\rangle) = |\phi\rangle \otimes |\phi\rangle \tag{5}$$

Now, taking the inner product of these two equations gives[4]

$$\begin{aligned}
(\langle\psi| \otimes \langle s|)U^\dagger U|\phi\rangle \otimes |s\rangle &= \langle\psi \mid \phi\rangle \otimes \langle s \mid s\rangle \otimes I & (6)\\
&= \langle\psi \mid \phi\rangle & (7)\\
&= ((\langle\psi| \otimes \langle\psi|)(|\phi\rangle \otimes |\phi\rangle) & (8)\\
&= \langle\psi \mid \phi\rangle \otimes \langle\psi \mid \phi\rangle & (9)\\
&= (\langle\psi \mid \phi\rangle)^2 & (10)
\end{aligned}$$

Line (5) holds because s is normalized. Critically, lines (5) and (8) give

$$\langle\psi \mid \phi\rangle = (\langle\psi \mid \phi\rangle)^2.$$

However, this equation is only true if $|\psi\rangle = |\phi\rangle$ or if $|\psi\rangle$ is orthogonal to $|\phi\rangle$. Thus, a general cloning device can only clone states that are orthogonal, which means that we cannot clone states which we know nothing about.

This is an important fundamental difference between the classical and quantum models of computation. We take for granted in classical computing that we can copy any unknown states, and much of classical computation relies upon this fact.

[4]Check this math???

1.7 Phase-kickback Trick

Suppose we have "black-box access" to some function f. That is, we can query f on some input x and it will compute $f(x)$. We record the output in a *response register*, the second qubit in the example below; the first qubit is called the *query register*. This can be modeled as the following unitary transformation:

$$|x, z\rangle \xrightarrow{f} |x, z \oplus f(x)\rangle$$

Given z, we can deduce what $f(x)$ is by taking $z \oplus f(x) \oplus z = f(x)$, addition mod 2. This is essential to keeping the transformation unitary.

One common query method is using what is called the *phase kickback trick*. The basic idea is that we initialize the response register to $|-\rangle$ so that both the query and response registers stay the same after the query, and the value of $f(x)$ is encapsulated by the phase of the state. More concretely, we have:

$$\begin{aligned}
|x\rangle \otimes |-\rangle &= |x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\
&= \frac{1}{\sqrt{2}}(|x, 0\rangle - |x, 1\rangle) \\
&\xrightarrow{f} \frac{1}{\sqrt{2}}(|x, f(x)\rangle - |x, 1 \oplus f(x)\rangle \\
&= |x\rangle \otimes \frac{1}{\sqrt{2}}\Big(|f(x)\rangle - |\overline{f(x)}\rangle\Big) \\
&= (-1)^{f(x)}|x\rangle \otimes |-\rangle.
\end{aligned}$$

1.8 Quantum Oracles

move to section on query complexity

1.9 Deustch-Josza Algorithm

1.10 Asymptotic Runtime Complexity

Computer scientists are very interested in how long their algorithm or function will take to run on given inputs. The crudest, most theoretical, runtime analysis looks at the *asymptotic runtime* of an algorithm. In practice, the actual runtime can be affected significantly by any number of factors like the operating system and memory latency, but for our purposes the *asymptotic* runtime is the only metric we will care about (particularly because current existent quantum computers are rather rudimentary).

The basic idea behind asymptotic runtime analysis is that we consider how long our algorithm takes to compute as a function of the *length* (number of bits) of the input, where the time interval is calculated as the number of basic operations required. Classically, the basic operations are generally considered to be the ones that the Arithmetic Logic Unit (ALU) in the CPU can do. These include: addition, subtraction, multiplication, division, boolean comparisons, and variable assignment. Quantumly, the basic operations are the same as above in addition to arbitrary unitary transformations and oracle queries (which take one time step).

Now that we have the basic idea, time to get technical. We define notions of relative asymptotic growth rates of functions: O, Ω, o, ω, and Θ. We will define these notions rigorously below, but for all practical purposes, the following distinctions are sufficient. Suppose we have two functions $f(n)$ and $g(n)$. Then,

- We say $f(n) = O(g(n))$, *"f is big-oh of g,"* if $f(n) \leq g(n)$ as n gets large;
- $f(n) = o(g(n))$, *"f is little-oh of g,"* if $f(n) < g(n)$ (strictly) as n gets large;
- $f(n) = \Omega(g(n))$, *"f is big-omega of g,"* if $f(n) \geq g(n)$ as $n \to \infty$;
- $f(n) = \omega(g(n))$, *"f is little-omega of g,"* if $f(n) > g(n)$ (strictly) as n gets large.
- Lastly, $f(n) = \Theta(g(n))$, *"f is theta of g,"* if $f(n) = g(n)$ (up to constant factors) as n gets large.

The above imprecise definitions are rigorous enough for our discussions; however, I present the formal definitions below for the overly pedantic or curious reader.

Theorem 1. *A function f(n) is O(g(n)), denoted f(n) = O(g(n)), if there exist* $c > 0, n_0 > 0$ *such that*

$$f(n) \leq cg(n) \quad \forall n \geq n_0.$$

Theorem 2. *A function f(n) is o(g(n)), denoted f(n) = o(g(n)), if there exists* $n_0 > 0$ *such that for all* $c > 0$,

$$f(n) < cg(n) \quad \forall n \geq n_0.$$

Theorem 3. *A function f(n) is* $\Omega(g(n))$, *denoted* $f(n) = \Omega(g(n))$, *if there exist* $c > 0, n_0 > 0$ *such that*

$$f(n) \geq cg(n) \quad \forall n \geq n_0.$$

Theorem 4. *A function f(n) is* $\omega(g(n))$, *denoted* $f(n) = \omega(g(n))$, *if there exists* $n_0 > 0$ *such that for all* $c > 0$,

$$f(n) > cg(n) \quad \forall n \geq n_0.$$

Theorem 5. *A function f(n) is* $\Theta(g(n))$*, denoted* $f(n) = \Theta(g(n))$*, if f(n) = O(g(n)) and g(n) = O(f(n)).*

Determining the exact values of n_0 and c is not necessary. Instead, we will present a few heuristics:

1. Constant factors don't matter.
2. Only the largest complexity class matters if we are adding terms.

For example, $f(n) = 25n^3 + 170000n$ is $O(n^{17})$ because $n^3 \leq n^{17}$ as n grows, and $f(n) = \Theta(n^3)$ because the $170000n$ doesn't matter as n grows and we don't care about the constant factor of 25 either, and $f(n) = \omega(n^2)$, because $n^2 < n^3$ as n goes to infinity.

1.11 Summary

In summary, below are the main constraints that our quantum computations must abide by:

- After measuring a state, it collapses irreversibly to one of the basis states.
- Aside from measurement, all transformations on the qubits must be reversible and normalized, i.e. the matrix representation must be unitary.