Quantum Computing: Resolving Myths, From Physics to Metaphysics

A Senior Project

presented to

the Faculty of the Physics Department

California Polytechnic State University, San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Arts

Physics

by

Jacob R. Mandel

Advisor: Dr. Katharina Gillen

March 19, 2021

Table of Contents

List of Figures and Tables

Abstract

As the field of quantum computing becomes popularized, *myths* or misconceptions will inevitably come along with it. From the sci-fi genre to the casual usage of the term *quantum*, idealism begins to take over our projections of the technological future. But *what are quantum computers?* And what does *quantum* mean? How are they any different than the computers we use on an everyday basis? Will there be quantum computing smartphones? Are quantum computers just a faster version of conventional computing or a wholly new way of computing altogether? The objective of this paper is to resolve common *myths* or misconceptions about the concept of quantum computers, as well as the outlook and potential of this technology. In the attempt to construct a sound narrative involving a wide range of disciplines, we will draw concepts from classical computing, quantum physics, computational complexity, as well as philosophy to decipher the mystery within this unique field.

Introduction

"The beginning of knowledge is the discovery of something we do not understand."

- Frank Herbert

Misconceptions about quantum computing are becoming more and more common as mainstream appeal to the field becomes widespread. The general familiarity of the term, *quantum computing,* is increasing among scientists and non-scientists alike with hopeful promises of bringing a paradigm shift to technology. However, the reality of these promises can quickly become inflated as we get carried away by hope and idealism and the sight of our scientific ground becomes lost. There is nothing wrong with having hope for an ideal and aiming high, but it is imperative to take a step back and reclaim our footing now and again.

One may be acquainted with the idea that quantum computers can solve certain problems more efficiently than any computer that we have now. But *why*? How are quantum computers any different than the computers we use on an everyday basis? Will there be quantum computing smartphones? Are quantum computers just a faster version of conventional computing? And what is the *quantum* in quantum computing? Questions of this sort will be of primary importance as we examine the field of quantum computing.

Quantum computing is a unique and compelling field for many areas of knowledge-seeking. It inhabits an intersection between the fields of theoretical and experimental physics, computer science, and philosophy. This paper aims to resolve common *myths* or misconceptions of the concept of quantum computers, as well as the outlook and potential of this technology. To do so, we will draw ideas from the previously mentioned fields, as necessary.

We will begin this journey by exploring the difference between the concepts, methods, and approaches to our everyday conception of computing, and quantum computing. Next, we will delve into a field of theoretical computer science, *computational complexity.* This field involves diverse mathematical approaches to classify different types of computable problems based on the required resources (such as time and space). Finally, our expedition will end with contemplation, and part speculation, about future philosophical considerations that quantum computing may bring forth. As the paper reaches its completion, we will enter an amorphous ocean that converges empirical science with the excogitation of the elusive metaphysical nature of quantum mechanics itself, as well as consciousness. By addressing our own scientific ignorance and taking a step back to consider the holistic motivation of our current work at hand, we allow ourselves the opportunity for more meaningful pursuits.

I Quantum Computing

"Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical."

- Richard Feynman

1.1 What is Quantum Computing?

Quantum computing is an intriguing field for many disciplines including computer science, physics, mathematics, information theory, and even philosophy. In my attempt to define, describe, and clarify the inner workings and myths of quantum computing, I will shift back and forth between abstract technical concepts and a more grounded qualitative narrative. To apprehend the notion of quantum computers, it is necessary to understand how computers, in general, compute and solve problems. So, this section will develop an understanding of the framework of computing, starting with Alan Turing's machine. Once we have covered this basis, we will address and resolve common myths or popular misconceptions about quantum computers.

1.1.1 Classical Computing

First, to understand the framework of computing and how computers generally solve problems, we must understand *classical* computing. Here, the term *classical* simply refers to Newtonian or classical physics; it is related to computing through the application to circuit behavior. Classical circuit behavior, also known as digital, typically involves transistors that exist in the off or on (0 or 1) state which provided an experimental foundation for Boolean algebra and logic (or the controlled manipulation of the states for a specified reason). The control and manipulation of these states, otherwise known as bits, through the utilization of concepts such as set theory, sequences, and logic gates lead to the emergence of algorithms to carry out more and more complex functions.

Whenever classical computing is mentioned, we are referring to Alan Turing's theoretical method of computing, namely the Turing machine. The Turing machine is an idealized theoretical model, in that (i) it has unlimited memory, (ii) it can perform arbitrarily many computational steps with perfect reliability, and (iii) it is describable within the kinematics of classical physics [1]. Our personal computers are an actuality of what is represented by this model. The Turing machine is a purely theoretical concept whereas the personal computer is a practical Turing machine [2]. "The Turing machine merely serves as a precise model for the definition of algorithm" [3]. An algorithm is a collection of simple instructions for carrying out some task. Commonplace analogies of algorithms range from procedures to recipes, to simple instruction manuals [3]. Correspondingly, how does a robot *do* anything? That's simple, they are *programmed* to do it. In other words, they are programmed to accept an input, follow the rules and guidelines set by the algorithm, and produce a definitive output. For example, asking Alexa to tell you a joke is the input, and the mediocre pun is the output. Alexa accepts an audio input and determines an appropriate response ruled by a programmed algorithm or set of rules.

This illustrates the definition of an algorithm given by the Church-Turing thesis in 1936 by Alonzo Church and Alan Turing. “It asserts the equivalence between the physical concept of what class of algorithms can be performed on some physical device with the rigorous mathematical concept of a Universal Turing Machine” [4]. The thesis provides a connection to the intuitive and customary notion of an algorithm with the notion of algorithms conceptualized by a Turing machine [3]. The Universal Turing Machine completely captures what it means to perform a task by algorithmic means. That is, if an algorithm can be performed on any piece of hardware (say, a modern personal computer), then there is an equivalent algorithm for a Universal Turing Machine which performs exactly the same task as the algorithm running on the personal computer [4]. The Church-Turing thesis proved to be a major development in computer science as it showed that we can connect our abstract theory of computing (Turing machine) to the real-world of computability.

Nevertheless, classical computing does have its limitations. The main limitation is its ability to solve problems *efficiently*. In the computing world, problems can either be solved efficiently, not efficiently or not at all. There are problems that classical computers would not be able to find a solution for, even if they were given millions of years. The concept of efficiency will be explored more thoroughly in the computational complexity section of this paper. A final limitation of classical computers is the ability to simulate the universe. Since classical computers are established, and operate according to the principles of classical physics, they are unable to efficiently integrate quantum physics in the operation of their computing.

1.1.2 *Quantum* Computing

Despite the progress and achievements of classical computers, certain tasks will remain out of reach, as noted by its limitations. The field of quantum computing aims to alleviate some of these limitations. The *quantum* in quantum computing evidently suggests the implementation of quantum physics in the field of computing. Richard Feynman had pointed out that there seemed to be essential difficulties in simulating quantum mechanical systems on classical computers and suggested that building computers based on the principles of quantum mechanics would allow us to avoid those difficulties [4]. That is to say, it might not only be possible to construct computers based on the laws of quantum physics but necessary. However, this paradigm-shifting scheme would require a whole new area of expertise and discovery. Quantum computing is not simply a faster way of computing or solving problems than classical computing. It is a *wholly new way* of computing altogether. A clear exemplification of the difference in computing is depicted by the foundational state-holding mathematical and physical object, the classical bit and the quantum bit. In classical computing, the bit follows strict rules of classical physics whereas the quantum bit adheres to the rules of probabilistic quantum physics. We will go into further detail discriminating and characterizing the bit for both methods of computing in a subsequent section. The distinction between the rules that govern the primordial object of computing prompts the possibility of the discovery of algorithms and mathematical approaches to problems unforeseen by classical computers. “Quantum computers would exploit the strange rules of quantum mechanics to process information in ways that are impossible on a standard computer” [5]. Most notable is the ‘speed up’ quantum computers offer to certain computable problems. Peter Shor in 1994, demonstrated that two enormously important problems – the problem of finding the prime factors of an integer, and the so-called ‘discrete logarithm’ problem – can be efficiently solved using quantum computing algorithms. This was of groundbreaking importance since these problems were not believed to have an efficient solution on a classical computer. This showed the potency and

promise of quantum computers as it demonstrated, in theory, to be more powerful than the existing Turing machine [4].

1.1.3 Motivations

Some of the most prominent motivations of quantum computers have already been briefly discussed, however, it is worthy to note that one of the *key* motivations of quantum computing is to have a better understanding of the nature of quantum mechanics itself: One of the goals of quantum computation and quantum information is to develop tools which sharpen our intuition about quantum mechanics, and make its predictions more transparent to human minds [4]. A major mathematical and computational difficulty when it comes to quantum physics is the simulation of complex quantum systems. In other words, by utilizing mechanics and concepts of quantum physics in our method of computing, we can alleviate some of the difficulty in simulating a quantum system itself. For example, large macroscopic systems utilize classical or Newtonian physics that is deterministic and therefore simple (compared to quantum physics). Smaller microscopic systems, on the other hand, utilize quantum physics that is probabilistic with respect to each component of the system, therefore, making the mathematics and simulation extremely tedious and complex, thus requiring more computing power and resources. Classical computers can simulate small-scale quantum systems, i.e., a 3-atom system, however, when attempting to simulate larger and larger quantum systems, i.e., 1000 atoms or 5000 atoms, it would take exponentially longer and could not be done efficiently. Quantum computing has the potential to simulate quantum systems of this scale efficiently (the concept of efficiency used here will be discussed in further detail in the section on computational complexity).

1.2 The Qubit

For classical computing, a bit represents a mathematical object that exists in a particular state, namely 0 or 1. The bit is the essential building block for computing theory. For quantum computing, the essential building block is represented by a qubit (‘Q’-bit), as in, quantum-bit. The qubit is a physical object, but mostly seen in the context of, and utilized algorithmically as an abstract mathematical object that can be in the state of 0 and 1 *simultaneously.* Upon measurement or observation, the state collapses into *one* state. Either 0 *or* 1. This is possible through the capacity of the conceptual notions and mathematical framework of quantum physics. Some of these concepts include superposition, complex amplitudes, and measurement collapse. In my attempt to describe the qubit, I will only offer a superficial understanding of the concepts of quantum physics which I deem suitable for the scope of this paper.

1.2.1 Schrödinger’s Cat

One of the most common pragmatic illustrations of quantum physics is the thought experiment of *Schrödinger’s Cat.* In the thought experiment, a cat is placed in a box with a radioactive substance. When the substance decays (at a random time), it triggers a Geiger counter that causes a chain reaction, Rube Goldberg style, that swings a hammer shattering a vial of poison, killing the cat. The radioactive substance adheres to the probabilistic nature and rules of quantum mechanics, therefore, will decay at a random time that is impossible to precisely pin down. Accordingly, as we, the conscious observers, are outside of the box, there is no one to observe the inside contents of the box. So, the cat could very well be alive or dead. What Schrödinger and

quantum mechanics say, is that the cat is *both* alive AND dead because the whole system is in a combination of both possibilities. The cat is in a *superposition* of the states of being alive and dead. That is until someone opens the box and observes the cat as either alive or dead. This is measurement. The system is no longer in a superposition of both possible states and the cat 'collapses' to the state of either being alive or dead.

Schrödinger's Cat

$$|\psi\rangle = \alpha\,|\text{Dead}\rangle + \beta\,|\text{Alive}\rangle$$

Qubit

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

$$\alpha, \beta = \text{complex amplitudes}$$
$$|\,\alpha\,|^2 + |\beta|^2 = 1$$

As for a qubit, we replace the cat's state of being dead or alive with a qubit's state of being $|0\rangle$ or $|1\rangle$. The qubit's states are depicted in Dirac or 'bra-ket' notation, '$|\ \rangle$', which is the standard representation of quantum states [4]. A qubit exists in a superposition of the states of $|0\rangle$ and $|1\rangle$ until measured or observed. The symbols in front of the qubit's state, α and β are the *complex amplitudes*. Expressly, they represent a value that determines the probability of the qubit being in the corresponding state. "When we measure a qubit we get either the result $|0\rangle$, with probability $|\alpha|^2$, or the result $|1\rangle$, with probability $|ß|^2$. Naturally, $|\alpha|^2 + |ß|^2 = 1$, since the probabilities must sum to one" [4].

Once measured or observed, the qubit's wavefunction (the mathematical description of the quantum state of a quantum system, represented by the Greek symbol Psi, Ψ) collapses into one state or the other, no longer in superposition. This is one reason why a qubit must be largely shut out from the external world and have very minimal contact outside of the system as there is a multitude of possibilities for a qubit's wavefunction to collapse. One may ask, "What exactly causes a qubit, or a quantum system's wavefunction to collapse?" The answer is unknown, and the best explanations are speculation. This topic is widely referred to as the measurement problem. However, by observing the system's state, thus causing the 'wavefunction to collapse,' how can one know with certainty if it was in superposition in the first place? It may as well have been in that state, and that state alone, the entire time. We know the existence of superposition in quantum mechanics through theory as well as experimental research that has been conducted over the years. One of the most notable examples is Thomas Young's double-slit experiment that showed the nature of the particle-wave duality of light. "Young postulated that light is a wave and is subject to the superposition principle; his great experimental achievement was to demonstrate the constructive and destructive interference of light" [6]. The demonstration of the constructive and destructive interference of light showed the quantum mechanical nature of superposition. It was found that the wave theory was not enough to fully characterize the nature of light. Even if a single photon is sent through the slit and onto the screen one at a time, the same interference pattern culminated. Therefore, it is only the quantum mechanical wavefunction that can fully account for the nature of light.

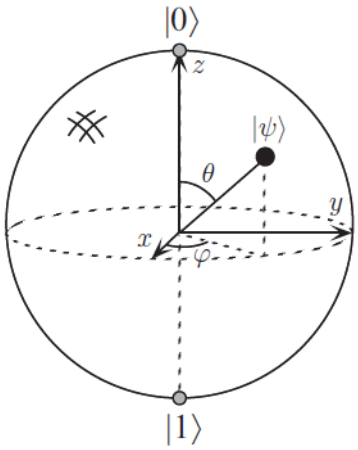

Figure 1. Bloch sphere as a representation of a qubit's state [4].

One thing to note about the difference between a classical bit and a quantum bit is the amount of potential information stored in the respective bit. The classical bit can only ever be in either the state 0 or 1, which is meager compared to the potential of the qubit. However, it is important to remark that the *potential* of the information stored in the qubit relies entirely on the quality of design of the quantum algorithm. The quantum bit, in superposition, can be represented by the *Bloch sphere*. The Bloch sphere provides a useful means of visualizing the state of a single qubit. The point with a line starting from the center of the sphere is a vector that represents all possible states for the qubit. Since there is an infinite number of points on the unit sphere, the number of possible states is extremely substantial. That is, before measurement. "Measurement changes the state of a qubit, collapsing it from its superposition of $|0\rangle$ and $|1\rangle$ to the specific state consistent with the measurement result." [4]. Again, we run into the measurement problem.

> This dichotomy between the unobservable state of a qubit and the observations we can make lies at the heart of quantum computation and quantum information. In most of our abstract models of the world, there is a direct correspondence between elements of the abstraction and the real world, just as an architect's plans for a building are in correspondence with the final building. The lack of this direct correspondence in quantum mechanics makes it difficult to intuit the behavior of quantum systems; however, there is an indirect correspondence, for qubit states can be manipulated and transformed in ways which lead to measurement outcomes which depend distinctly on the different properties of the state. Thus, these quantum states have real, experimentally verifiable consequences, which we shall see are essential to the power of quantum computation and quantum information [4].

Although the theory and mathematical representation of qubits with the utilization of quantum mechanical properties such as superposition and probability amplitudes seem abstract, and one may even be skeptical to accept it as a *real* method of computing, there are undeniably real experimentally verifiable consequences and results from this method of computing.

1.3 From Qubit to Computation

Up to this point, we have primarily covered concepts regarding a single qubit. Here, we will continue constructing an understanding of quantum computing by discussing the significance of a system of multiple qubits. You cannot do much, in terms of computing, with a single qubit. The real power of quantum computing starts to become apparent once you begin to amass more and more qubits. As we move forward with multiple qubit systems, and eventually the construction of quantum algorithms, we will be introduced to fundamental aspects of computing and quantum systems such as logic gates, entanglement, and computational circuits.

1.3.1 Classical Logic Gates

A logic gate is any physical structure or system that takes a set of binary inputs (0's and 1's, apples and oranges, or spin-up and spin-down electrons) and returns a single binary output: a 1, an apple, a spin-up electron, or one of two states of superposition. What goes on in between the input and output, is a Boolean function that determines what the output should be depending on the input and directive of the particular Boolean function. The Boolean function is nothing more than a rule or set of instructions for how to respond to Yes/No questions [7]. The only single-bit gate is the NOT gate, meaning it only requires one input. The NOT gate simply reverses the state of the input as the output (from 0 to 1, or 1 to 0). The six most basic two-bit logic gates in classical computing are the AND gate, OR gate, XOR gate, NAND gate, NOR gate, and the XNOR gate. The two-bit gates take in two bits as inputs and generate a single output, 0 or 1. The gates are then combined into circuits, and the circuits are programmed into CPUs or other computational components [7]. Each of these gates consists of a particular logic or functionality that contributes to the complexity of the resulting circuit. However, the NAND gate (NOT-AND), which is equivalent to adding an AND gate to a NOT gate, is considered a universal gate. A universal gate is one from which any other logic gate can be made. In other words, any circuit can be broken down and rephrased using only the universal logic gate and no others. The NOR gate is an example of another universal logic gate.

1.3.2 Entanglement

Quantum entanglement is often understood as a solely quantum mechanical phenomenon, as there is nothing of this sort that occurs in classical physics. When two particles, or qubits, are entangled, they remain connected so that actions performed on one, affect the other, despite being separated by arbitrary distances. This is relevant because, from a classical or a macro point of view, the two particles separated at large distances would have no way of interacting with each other. However, through entanglement, these particles far away from each other now can interact through the facets of quantum mechanics. Two entangled qubits are now one system, so thinking of them as separate systems is wrong. Although they are still *physically* distinct or separate, they can now be manipulated, and interact with the environment, as one system. Therefore, by interacting with one qubit you change the state of the whole system. Or, by manipulating one particle in a particular manner, you may be able to deduce or mathematically predict the state of the other qubit at the instant you interact with the first particle. Albert Einstein once described this phenomenon of instantaneous communication at a distance, "spooky."

1.3.3 Quantum Logic Gates

A quantum logic gate is an input-output operation whose inputs and outputs are discrete quantum variables, such as spin. Quantum computers differ from classical computers in that they can maintain quantum-mechanical coherence: an electron can be in a coherent superposition of spin up and spin down, and a quantum bit can be in a coherent superposition of 0 and 1 [8]. Similar to the classical single-bit gate, *single-qubit gates* each take in a single input and produce a single output. However, the inputs and outputs are much more complicated for quantum logic gates, as each state can be any point on the Bloch sphere and has infinitely many options to end up at after passing through the gate. Some of the simplest single-qubit gates are the Pauli-X Gate, Pauli-Z Gate, and the Hadamard Gate.

$$\alpha|0\rangle + \beta|1\rangle \xrightarrow{\text{Pauli-X}} \alpha|1\rangle + \beta|0\rangle$$

$$\alpha|0\rangle + \beta|1\rangle \xrightarrow{\text{Pauli-Z}} \alpha|0\rangle - \beta|1\rangle$$

$$\alpha|0\rangle + \beta|1\rangle \xrightarrow{\text{Hadamard}} \alpha\frac{|0\rangle+|1\rangle}{\sqrt{2}} + \beta\frac{|0\rangle-|1\rangle}{\sqrt{2}}$$

Figure 2. Three of some of the simplest single-qubit logic gates. Input states are shown on the left of the arrow, quantum logic gates above the arrow, and output states on the right [4].

As we begin discussing two-qubit gates, it starts to get more and more complex as we consider both quantum superposition and entanglement. Two-qubit gates can be either entangling or non-entangling. The two-qubit gates that are entangling, are considered universal quantum gates. "A quantum gate is said to be universal if copies of it can be wired together to make circuits to evaluate any desired classical logic function (that is, a quantum universal gate is also a classical universal gate), and to enact any desired unitary transformation on a set of quantum variables" [8]. In other words, these entangling two-qubit gates are considered universal, as they alone can represent and rephrase any possible quantum algorithm. Thus, entanglement is crucial since the universal two-qubit gates require entanglement to function. Some examples of two-qubit gates are the Controlled Not Gate (entangling), Controlled Phase Gate (entangling), Swap Gate (*not*-entangling), and the $\sqrt{Swap}$ Gate (entangling). However, it is important to note that the gates labeled 'entangling,' only have the *ability* to entangle. That is to say, they do not always entangle. It also depends on the state of the input, or what is sent through the gate.

(not-entangling) $\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle \xrightarrow{\text{SWAP}} \alpha|00\rangle + \beta|10\rangle + \gamma|01\rangle + \delta|11\rangle$

(entangling)

$\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle \xrightarrow{\text{cNOT}} \alpha|00\rangle + \beta|01\rangle + \gamma|11\rangle + \delta|10\rangle$

$\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle \xrightarrow{\text{π-cPHASE}} \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle - \delta|11\rangle$

$\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle \xrightarrow{\sqrt{SWAP}} \alpha|00\rangle + \frac{1}{2}\beta[(1+i)|01\rangle + (1-i)|10\rangle] + \frac{1}{2}\gamma[(1-i)|01\rangle + (1+i)|10\rangle] + \delta|11\rangle$

Figure 3. Four common two-qubit logic gates. Input states are shown on the left, and output states on the right.

Entanglement is important for the functionality of two-qubit gates since the combined state cannot be written as two separate qubit states. Although the qubits are in a superposition, measuring one will collapse the original superposition and change the state of the other. This is the "spooky action at a distance" that upset so many physicists in the early 20th century [9].

1.3.4 Quantum Circuits and Algorithms

From quantum gates, we can build more complex quantum gates as well as collections, models, and sequences for quantum computation. A *quantum algorithm* involves distinctly 'quantum' notions such as entanglement and superpositions. Quantum algorithms are most commonly modeled, or represented, by quantum circuits. A *quantum circuit* is a computational routine consisting of coherent quantum operations on quantum data, such as qubits. It is an ordered sequence of quantum gates, measurements, and resets [9].

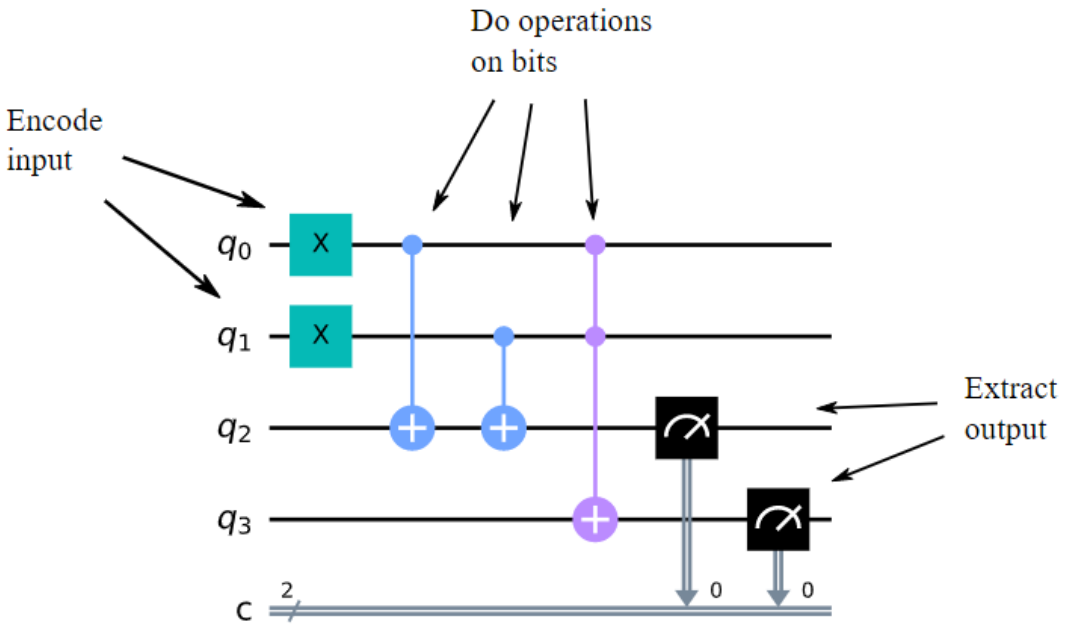

Figure 4: Quantum circuits are commonly used as a model for quantum algorithms. Qubits are represented as q's (left). Operations or gates are shown in blue and purple (middle). The two black boxes with the white symbol represent measurement or output extraction (right) [9].

In a quantum circuit, the steps to solve the problem are quantum gates performed on one or more qubits, as mentioned in the previous section. Thus, the circuit is concluded with a measurement on one or more qubits. A difference with a classical algorithm is that a quantum algorithm is always reversible. This means that if measurements are not a part of the circuit, a reverse traversal of the quantum circuit will undo the operations brought about by a forward traversal of that circuit [9]. In other words, if the system remains unmeasured, unobserved, then a reversal of the operations in the quantum circuit can be carried out which will undo the state change that happened in the forward direction. However, if measurement is a part of the circuit, reversibility is no longer possible.

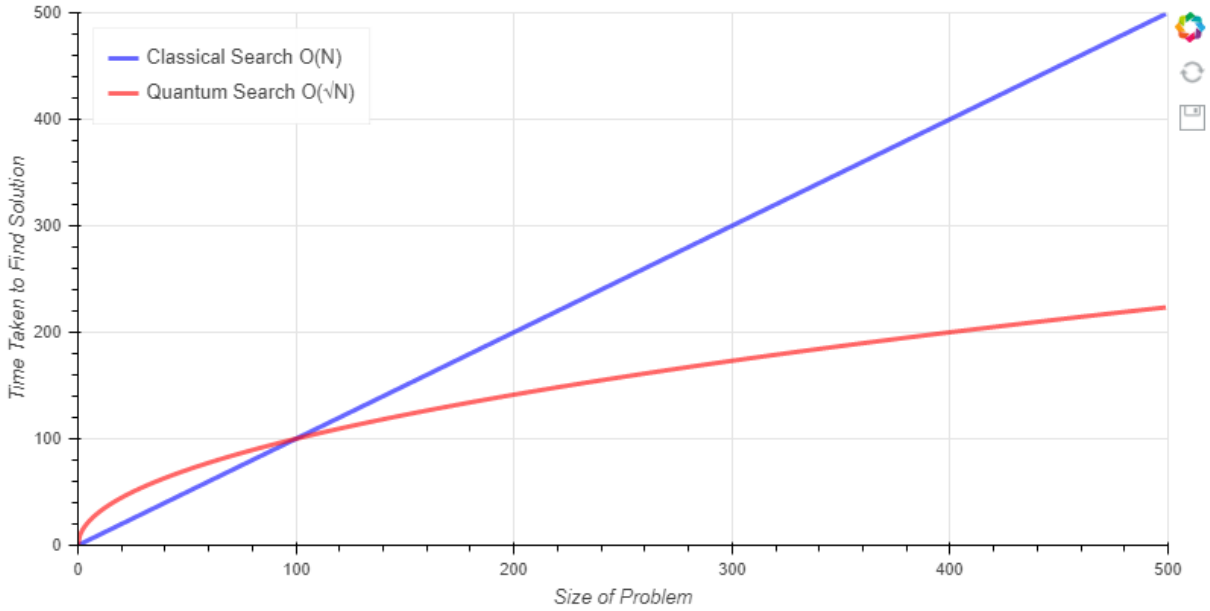

Figure 5. The time taken to find a solution is plotted against the size of the problem. Problem is represented as an unstructured search of *N* items (Plot was taken from https://qiskit.org/) [9].

Figure 5 illustrates the potential (and actual) speed-up of quantum algorithms. For a given problem size greater than 100, Grover's quantum algorithm reduces the time taken to find a solution from N to $\sqrt{N}$. By using this algorithm on certain problems, quantum computers can find a solution faster than a classical computer. Grover's algorithm provides a *quadratic* speed-up [9]. However, as you may have observed, there seems to be some unanticipated or strange behavior *below* a problem size of 100. Even though Grover's algorithm has quadratic speedup, the time required to find a solution *below* a certain problem size (in this case a problem size of 100) is greater for Grover's algorithm than the classical algorithm. Additionally, some quantum algorithms may offer speed up only for very large problem sizes, or potentially not at all (where the quantum algorithm line stays above the classical algorithm, and the two plot functions never cross). In other words, there are many cases for which quantum algorithms will prove to be less efficient than classical algorithms. This suggests that classical computers may still be better at solving *some* computational problems than quantum computers.

Efficient quantum algorithms are designed so you do not have to read out the entire output state repeatedly. If a quantum algorithm is poorly designed, it will have to run 2^n times or more in order to produce a distribution of the probability of all possible paths to the answer (where n is the number of qubits). It is a misconception that quantum computers will run each possible path at once through the existence of many possible worlds, therefore resulting in a speedup. They can, however, run all possible paths to the answer at once, but that may all be lost upon measurement, depending on the quality of the algorithm. Quantum algorithms have the potential to work fast, but only with a well-designed algorithm. "If we have n qubits, we will need to keep track of 2^n complex amplitudes. The vectors representing the states of the qubits will grow exponentially with the number of qubits. This is the reason quantum computers with large numbers of qubits are so difficult to simulate. A modern laptop can easily simulate a general quantum state of around 20 qubits but simulating 100 qubits is too difficult for the largest supercomputers [9].

With the application of quantum physics in computing, we can extend our knowledge and capabilities beyond the classical computer. Quantum algorithms provide much promise to the field of computation, but not in the sense that they will replace all classical algorithms. Problems that are fundamentally unsolvable by classical algorithms cannot be solved by quantum algorithms either (this will be ventured into more depth in the Computational Complexity section of this paper). Quantum computers are valuable because they can solve *some* problems significantly faster than classical algorithms. The best-known examples are Shor's algorithm and Grover's algorithm. Shor's algorithm is a quantum algorithm for integer factorization. It can solve this problem exponentially faster than the best-known classical algorithm. As we inquired into earlier, Grover's algorithm can search an unstructured database or unordered list quadratically faster than the best classical algorithm with this purpose [10]. We have built a conceptual framework of quantum computing by building up from the qubit to quantum logic gates to finally quantum algorithms. Table 1 below presents a brief list of some current methods of creating qubits.

Quantum Computing Approach	Qubit System	Entanglement Interaction
Nuclear Magnetic Resonance (NMR)	Nuclear spin in a molecule (liquid state) or solid state	Spin-spin coupling
Trapped Ions	Internal spin states of trapped ions	Phonons, photons, "head ions"
Neutral Atoms in Optical Lattices	Internal or motional states of neutral atoms	Electric or magnetic dipole-dipole interaction, cold collisions
Cavity Quantum Electrodynamics	Internal or motional states of atoms in cavities, photons in cavities	Atom-photon interaction
Optical	Polarization modes of photons	Prior entanglement through parametric down conversion
Solid State	Spin or charge in quantum dots, P "impurities" in Si	Heisenberg exchange interaction
Superconducting	Charge, flux, or energy levels in a superconducting circuit using a Josephson junction	NMR interaction

Table 1. List of quantum computing approaches. This list is not all-inclusive [11].

1.4 Selected *Myths* about Quantum Computers, Resolved.

Myth 1: When will quantum computing smartphones come out?

This is a common misconception of the usage of quantum computing. One way to answer this, is that unless an everyday task is found that cannot be done efficiently with a classical computer, they *most likely* will not be coming out. Not because it is impossible, rather, that it would be meaningless. There is simply no use in attempting to make quantum computing smartphones. All conceivable utility and everyday usage of smartphones are much more efficient with classical computing than quantum computing methods. For the most part, classical computers are just as efficient, and in most cases more efficient at solving the *easy* problems than quantum computers. *Easy* problems may include basic arithmetic, sorting, etc. Smartphones only consist of these *easy* problems. So, it would be a waste of resources and effort to always have a quantum computer on our person. The main use for this 'sci-fi' form of computing, is to create better and better simulations of our natural world. Principally, to simulate and model quantum mechanics and quantum systems. With these enhanced methods of simulation, we can run tests and experiments that will potentially lead to groundbreaking discoveries and push boundaries in many areas of human knowledge including physics, biology, chemistry, medicine, and much more. (Not to mention, the physical size of most practical quantum computers is enormous and would require great efforts to stay powered, stay accurate, and avoid measurement collapse for computation to take place.)

Myth 2: Quantum computers are just *better* classical computers.

Quantum computers are not simply *better* versions classical computers. They are not just a newer version of your laptop 20 years down the line. Quantum computing is a wholly *new way* of computing that utilizes quantum phenomena such as superposition and entanglement to carry out computation. Quantum computing moves away from the strict 0 or 1 state boundaries and into the realm of probability and complex wave amplitudes to manipulate and store information in a much different fashion than its classical counterpart. This new method of computing extends us beyond our current capabilities of computing that were previously limited by classical computers.

Myth 3: Quantum Supremacy

Quantum supremacy was claimed by Google with their superconducting computer in 2019. But what did they *really* claim? The phrase was coined in 2012 by John Preskill, a theoretical physicist at Caltech, to describe the point at which quantum computers can do things that classical computers simply cannot [12]. Google claimed that their computer accomplished a task in 200 seconds where it would take the world's fastest classical supercomputer more than 10,000 years [13]. However, this does not come without controversy. There is much debate over what quantum supremacy *actually* means. After Google's announcement, computer scientists at IBM have countered that their most powerful supercomputer, called Summit, could complete the same task in 2.5 days rather than 10,000 years [12]. IBM's counter shows that Google's claim of quantum supremacy

may not be fully deserving as there exists a classical computer to complete the said task in a reasonable amount of time, 2.5 days, as opposed to 10,000 years. Nonetheless, the leaps and bounds made in the field of quantum computing, both in academic research as well as in industry, have been remarkably exciting and promising.

Myth 4: Parallel processing in *one* system using multiple universes.

It is widely talked about and assumed that quantum computers operate by utilizing parallel processing in *one* system, which is why quantum computers are so fast. This is not the case. Computer scientist, Dr. Scott Aaronson clarifies, that a quantum computer would not let you try all answers in parallel and instantly pick the best one. That is simply too good to be true. You can make a superposition over all possible outcomes, but once you measure it, you are just going to get a random answer. So, for any hope of speed up, you need to exploit the concept of interference. This allows you to get the different paths leading to a given wrong answer to *destructively* interfere and cancel each other out while the paths leading to the right answer *constructively* interfere and add up. This was accomplished and implemented in a quantum algorithm for factoring integers (Shor's Algorithm).
This idea of parallel processing in one system assumes Hugh Everett's many-worlds interpretation of quantum mechanics. That each parallel computation occupies a separate world in order to complete the computation in much less time. However, while this is a popular theory of quantum mechanics and a conception to quantum computing, it is not widely accepted, "it is just easier to think about quantum algorithms as parallel computations performed in parallel worlds" rather than coming up with the *why* in why quantum computing works the way it does [14].

Myth 5: Quantum computers are just *faster* classical computers.

Quantum computers are not necessarily *faster* than classical computers. While it is true that there are certainly speed-ups, for specific problems, over classical computers, quantum computers are only faster than classical computers for *some* types of problems. To understand this more deeply, I explain this idea further in the Computational Complexity section of this paper.

Myth 6: Will quantum machines become conscious?

As philosopher David Chalmers states as the *Law of Minimization of Mystery*: "Consciousness is mysterious and quantum mechanics is mysterious, so maybe the two mysteries have a common source" [15]. Maybe the quantum computer is just an artificial creation of our concept of what human consciousness is. Or maybe, this is not the right question at all. Maybe, we should be asking whether quantum computing algorithms will provide a drastic advancement in the field of artificial intelligence to the point of emergent consciousness. Will the potentiality of quantum algorithm 'speed-ups' over classical computing algorithms exponentially outpace current classical methods of machine learning? Or is there something within us that cannot be programmed? Will there be a point where these machines or systems become conscious in the same way as us? Or is consciousness inherent to biological life forms? Unfortunately, we do not possess the

capability to answer these questions objectively. Although empirical science has not caught up to this point, there has been interesting work done in many of these fields related to computer science, physics, and philosophy of mind.

II Computational Complexity

"Physics is the universe's operating system."

- Steven R. Garman

2.1 Computability and Complexity

In this section, I will venture into a field of theoretical computer science, *computational complexity*, involving diverse mathematical approaches to classify types of computable problems. This unique field of complexity is not to be confused with the conception of complexity used in other fields such as biology or cosmology (complexity in terms of cellular automata or universe expansion) [16]. Computational complexity is concerned with the fundamental capabilities and limitations of computers [3]. The theory is interested in how complex something gets as the input approaches a very large number. I will begin by discussing necessary quantitative information and concepts of computational complexity, and then utilize them to make sense of important and often misconstrued details about quantum computing.

To begin, I would like to point out the difference between computability and complexity. The onset of the idea of computability, or computing theory, dates back to the 1930s involving some of the pioneers of modern computer science Alan Turing, Alonzo Church, and Kurt Gödel. The objective of computability theory is to classify problems by those that are solvable and those that are not [3]. However, it turns out that most of the problems we actually want to solve are known to be solvable using our conventional method of computing. So, the better question to ask, is which problems are *efficiently* or *feasibly* computable? This is the general focus of computational complexity theory. The goal of complexity theory is to classify problems as easy ones and hard ones [3]. This seems easy enough, but what is the distinguishing component between easy and hard problems? We have arrived at the core of computational complexity. Researchers have devised a scheme to classify problems according to their computational difficulty. For example, problems for which you can derive a perfect method to arrive at a solution are considered easy. Problems for which you can only find a 'good enough' method to arrive at a solution may be considered more difficult (in other words, it may take a long time for the method to find the solution). Problems are considered hard when there is no known method for arriving at a solution. One last paramount consideration when examining complexity, is the question, "What makes some problems computationally hard and others easy?" As we move deeper and deeper into the makeup of any theory, the epistemological nature of the subject matter at hand is inevitable. And it is no different for computability and complexity. Approaching the realm of unanswerable problems, researchers have no definitive answer to the question posed above. However, with the creation of a system of classification (complexity classes), we can utilize a method to provide evidence that determines which problems are computationally easy, more difficult, or hard, even if we are unable to *prove* the underlying nature and objectivity.

2.2 Complexity Classes

Computational complexity is a subfield of computer science that studies the resources (i.e., time, space, and randomness) needed to solve computational problems [16]. In our examination, we will focus on one resource, time. Consider a simple example of adding two numbers. Adding 12 and 56, we presume it takes a defined amount of time. However, if the numbers are twice as long, 1234 and 5678, then it will take twice as long to compute since there is twice as much information. For instance, in computing this problem by hand, you will not be surprised when it takes twice as long to write down twice as many numbers. This is an example of a problem that takes *linear* time to compute. On the other hand, multiplication takes *quadratic* time. When multiplying numbers that are twice as long, it now takes four times as long to compute due to the number of steps it takes to calculate a multiplication problem. Linear and quadratic time problems are some of the most basic examples of complexity exhibiting the resources required (time). In the following sections, these types of problems, as well as many others, are classified into categories that computer scientists call *complexity classes.*

Complexity classes are categories of related computational problems organized by the computational difficulty to solve the problem concerning certain computational resources (in our case, time). An agreeable comparison to complexity classes in computer science is the elements of the periodic tables in chemistry. Complexity classes are understood as the basis and foundation of understanding for higher-level concepts and emergent properties as the complexity of the subject increases. There are 'must know' or common complexity classes which we will touch upon in this paper, however, there are around 500 classes discovered thus far (See *Complexity Zoo* [17]) .They are When considering time as a resource for computable problems, it is necessary to recognize the implications of efficiency. We will see in the following section how the divergence of required resources leads to more or less efficient methods of solving problems.

2.2.1 Polynomial-Time (P)

The simplest complexity class is P, which stands for *Polynomial-Time.* P is the class of all problems solvable with a Turing machine in polynomial-time [18] (i.e., classical computer). In other words, polynomial-time problems can be solved with a conventional computer. A conventional computer is anything from a smartphone to a laptop, to even a coin press machine. Polynomial-time problems are commonly known as the computational basis for sorting, basic arithmetic, sending an email, streaming music, and phone apps like Angry Birds. Problems that are computable in polynomial-time use a number of steps that grow only in proportion to the size of the input raised to a *fixed* power (n^x, where x is a fixed number).

$$n^x$$

n = input

x = fixed number

Problems that are solvable in *linear* and *quadratic* time are types of polynomial-time problems. To illustrate *linear* time, let us return to our example of adding two numbers. By increasing the input to twice its original size, the time it takes to solve that problem increases to twice its original duration. Mathematically, this is written as n^1, where *n* is the input. For *quadratic* time, multiplying two numbers is written mathematically as n^2. It would take *four* times as long to solve a problem that has its input increased by twice. Although n^5, n^{10}, even n^{50} time problems

are also possible, linear and quadratic time problems are the simplest and most common examples of computable problems in polynomial-time.

$$n^1 \ \textit{linear time}$$
$$n^2 \ \textit{quadratic time}$$

2.2.2 Exponential-Time (EXP) and Efficiency

Similar to polynomial-time problems at first glance is the set of problems solvable in *exponential-time.* Also known as EXP. Problems in EXP take exponentially more time to solve than polynomial-time problems. So far, we have seen polynomial-time problems that increase only like the size of the input. This can be rather substantial as the input approaches a very large number, or the fixed power is an already large number. However, it is a completely different world when it comes to exponential-time. Every time you add one more data point to the input, the time needed by the algorithm doubles. Or triples, or quadruples, and so on with respect to its base. For EXP problems the base is a fixed number raised to the input as a power.

$$c^n$$
c is a fixed number, where $c > 1$
n is the input (or variable)

As you can imagine, the time (resources) required for exponential-time problems drastically exceed what is required for polynomial-time problems. Computer scientist, Scott Aaronson explains this polynomial/exponential distinction as occupying a "middle ground" between two other types of gaps. On one side (the polynomial side), there are small quantitative gaps (i.e., between n steps and 2n steps), and on the other side (the exponential side), the gap is the difference between finite steps and an exponentially large number since the increase in output due to an increase in input is exponentially large (i.e., between 2^n and 2^{2n}). This representation of the differences in gap sizes of steps constitutes time efficiency. Problems on the polynomial side, with small quantitative steps, are much more efficient to compute than problems on the exponential side, with exponentially larger steps. "Theoretical computer scientists generally call an algorithm 'efficient' if its running time can be upper bounded by any polynomial function of n, and 'inefficient' if its running time can be lower-bounded by any exponential function of n" [16]. The smaller, or more finite, the gap size, the more efficient an algorithm is at solving a problem. The efficiency of an algorithm is important because it determines which problems are necessarily solvable in a reasonable amount of time (a reasonable amount of time being the lifetime of a human being).

2.2.3 Nondeterministic Polynomial Time (NP)

The next complexity class we will discuss is NP, *Non-Deterministic Polynomial Time.* Contrary to its appearance, NP does **not** stand for nonpolynomial-time. In other words, the NP problems are not simply the problems that do not belong in P. The solution to the computational problems in NP can be *recognized* or *verified* in polynomial time, even though finding the *actual* solution might be very hard, or seemingly impossible [16]. To put it another way, if there exists a solution, then there is an efficient way to *check* the solution. This is different than P. In P, if there is a solution to a problem, there is an efficient way to both check the solution *and* find the solution. However, these approaches, or *algorithms*, come in all forms. Some are extremely efficient,

requiring minimal resources to get to the solution, while others are slower and less efficient. The ones that are slower and less efficient are oftentimes replaced by other more efficient algorithms as researchers discover new methods and ways of computing. An example of an NP problem is factoring. By asking: "Does the number 47436923 have at least 4 non-trivial divisors?" If one were to compute this by hand, it would be extremely tedious and time-consuming and should be given some sort of award for the effort. However, this is not such an easy task, in terms of complexity resources, for a computer to compute also. Now, if you or the computer were *given* the 4 divisors, the task is made dramatically easier since all that is needed is to *check* if the divisors are indeed factors. If this were a polynomial-time problem, the method of *finding* the factors would be as easy and efficient as simply *checking* if the given numbers are factors. NP problems, in general, take exponentially more time to find a solution than polynomial-time problems.

2.2.4 NP-Complete

Currently, there does not exist an efficient algorithm to solve every problem in NP. These computational problems without a method for finding a solution are called NP-complete problems. The class of NP-complete problems is considered to be a set of some of the hardest problems in modern computer science since there is no known method of arriving at a solution. However, "if a polynomial time algorithm exists for any of these problems, all problems in NP would be polynomial time solvable" [3]. In other words, if there exists a polynomial-time algorithm for one of these problems, we will also know that there exists a polynomial-time algorithm for all NP-complete problems. "An efficient algorithm for an NP-complete problem would mean that computer scientists' present picture of the classes P, NP and NP-complete was utterly wrong, because it would mean that every NP problem (including all the NP-complete ones) was actually a P problem" [5]. In other words, does P=NP? Does there exist a polynomial-time solution to all NP problems we simply have not discovered yet? The P=NP question is one of the most debated topics in computer science. Yet, there is no conclusive answer. This debate has many future implications in topics including mathematics, cryptography, machine learning, and artificial intelligence.

2.3 The Quantum Realm of Complexity

Now that we have surveyed some background information of computational complexity, I will shift toward a discussion about the relevance of quantum computers and their computational capabilities. Namely, quantum computers' capabilities in terms of complexity. First, I would like to note, that the background in computational complexity is in no way exhaustive or thorough. I have only provided the information I deemed necessary to move forward with the topic at hand and invite the reader to look further into resources that have been cited in this section for a more in-depth understanding. The goal of this section is to offer clarity into a new way of computing while employing terminology and concepts of computational complexity included in the prior section. Two commonly misconstrued assumptions or hopes of quantum computers will be posed and examined to provide insight and accuracy by utilizing the information brought forth in this paper.

1) Are quantum computers faster than classical computers?

To answer this question, we must clarify what is meant by 'fast.' I will do this by first restating the question in terms of computational complexity: "Can quantum computers solve computational problems in less time than classical computers?" The answer is yes, and it has been shown through the quantum algorithm created by Peter Shor. Shor's algorithm factors an n-bit integer in $\sim n^2$ steps on a quantum computer. Due to this feat of science, it is known that quantum computers can solve some NP (non-deterministic polynomial-time) problems in P (polynomial-time). Classical computers require an exponential (c^n) amount of time to find a solution to a factoring problem (NP), where a quantum computer using Shor's algorithm takes a polynomial amount (n^x). So, in the case of factoring, quantum computers can solve problems in less time than classical computers. Or put more accurately, quantum computers are computationally more efficient than classical computers. There are other examples of quantum computers showcasing impressive efficiency, such as Grover's algorithm which provides *quadratic* 'speedup'. However, the importance of Shor's algorithm, is that it provided a key piece of evidence that switching from classical to quantum computers would enlarge the class of problems solvable in polynomial-time [16]. Unfortunately, the P=NP debate still goes on, but Shor's algorithm and other 'speedup' examples of this sort do offer great insight and outlook on the field of quantum computing.

NOTE: It is a minority position to think that factoring is an NP-complete problem. So, for the scope of this paper, I decided to take the position of Dr. Scott Aaronson along with many other scientists on this topic of P=NP by assuming that factoring is *not* an NP-complete problem.

2) Can quantum computers solve problems not solvable by classical computers?

First, the types of computable problems for both classical and quantum computers are the same. So, in saying a problem is unsolvable, or physically and empirically devoid of a method of finding a solution, it would be unsolvable for both versions of computers. What most closely resembles this type of problem, is the complexity class, NP-complete. There is no known efficient algorithm for solving these problems. Aaronson points out, "contrary to myth, quantum computers are not known to be able to solve efficiently the very hard class called NP-complete problems" [16]. Therefore, quantum computers are not known to be able to solve the classically unsolvable. If it were the case that quantum computers have the capability to solve these sorts of problems while classical computers did not, that would open up a whole new can of worms in terms of computability, complexity theory, and the P=NP debate. "Many computer scientists now conjecture not only that P $\neq$NP but also that quantum computers cannot solve NP-complete problems in polynomial time" [16].

In conclusion, the speed and efficiency of quantum computing in comparison to classical computing show the true nature of the promise of quantum computing in full transparency. "This speed advantage is so significant that many researchers believe that no conceivable amount of progress in classical computation would be able to overcome the gap between the power of a classical computer and the power of a quantum computer" [4]. The understanding of even minimal basics of computational complexity outlined in this paper provides an in-depth and rounded comprehension of how the computing power and capability of quantum algorithms go up against

classical computing algorithms. Finally, we have quantitatively and qualitatively achieved an understanding as to why researchers are so interested in the field of quantum computing and why there is so much for hope for the field in the future.

III Philosophical Considerations

"In quantum computing, no one has any quantitative idea of where the theory could break down and the computer would stop working - which leads to the conjecture that maybe it won't stop working…" – Scott Aaronson

3.1 Quantum Computing and Consciousness

As research in scientific fields approaches cutting-edge discoveries that may affect our very understanding of the universe or how we live our everyday lives, hope, and promise of *far-out* ideas may start to take shape. In this section, we will explore some philosophical, and some speculative, views on the outlook of quantum computing and what could theoretically be to come.

Since quantum computing harnesses the capabilities of quantum mechanics, will it allow for a better understanding of quantum mechanics itself? With a better understanding of quantum mechanics, we may be presented with a radical paradigm shift in our scientific understanding of the world, but also a dramatic shift in our understanding of the very reality of our existence. There is already increasing popularity of attempts to integrate quantum physics as sort of pragmatic metaphors and analogies of everyday life. Particularly in the spiritual, or new-thought communities. If quantum computing can bring about experimental methods to test our metaphysical understanding and interpretations of quantum mechanics in a new light, it may present a radical perspective shift of the entire world.

We will also explore what quantum computing may mean in terms of the advancement of technology and artificially intelligent systems. Will it realize faster and better machine learning algorithms to the point of artificial intelligent systems emerging conscious? Or do the quantum machines themselves represent human consciousness? Lastly, how do we know if consciousness is computable at all?

3.2 Experimental Metaphysics

The capacity of quantum computing is derived from the current understanding we have of quantum mechanics. The underlying concepts of superposition and entanglement are key ideas, but not at all sufficient to explain the inner workings of quantum computing. Each idea in quantum mechanics seems to draw us down a metaphysical rabbit hole with no end in sight. Nobody seems to have any clue as to what is *actually* going on with quantum mechanics and why it works the way we currently understand it.

The attempts to understand the metaphysical nature of quantum mechanics are commonly known as the interpretations of quantum mechanics. Most common is the Copenhagen Interpretation, which formed around the work of Niels Bohr and Werner Heisenberg in Copenhagen during the 1920s [19]. It encompasses the ideas of measurement, wavefunction collapse, and irreversibility. Put very roughly, it includes that "quantum mechanics is an extremely effective tool for predicting measurement results that takes the configuration of the measuring

apparatus (described classically) as input, and produces probabilities for the possible measurement outcomes (described classically) as output" [19].

Alternatively, there is the widely debated Many Worlds Interpretation (MWI) proposed by Hugh Everett in 1957. The MWI asserts that there is no 'wavefunction collapse.' Instead, the wavefunction is universal and objectively real and that measurement or observation does not collapse all possibilities to one observable result, therefore annihilating all unobserved possibilities. Rather, each possible quantum state is physically realized in some way. Loosely put, at the time of 'measurement,' instead of a collapse, it can be thought of as a split. Where each possible outcome is physically realized in distinct *worlds.* That is to say, "every physically possible outcome of a measurement actually occurs in some branch of the quantum state, but as an inhabitant of a particular branch of the state, a particular observer only sees one outcome" [19]. As the observer, we inhabit only one particular branch of the quantum state, and therefore are only able to observe one outcome. Whereas a theoretical observer in a different branch of the quantum state may observe a different possible outcome.

How does this relate to quantum computing? Skeptics may ask, "If no one knows why quantum computers are superior to classical ones, how can we be sure that they are, indeed, superior?" [20]. It is inevitable for the question to arise as to *why* quantum computing algorithms work so efficiently compared to their classical counterpart. This is where one may come across two different groups of scientists in the field. One group will constantly dodge the question, claiming that the metaphysical nature of quantum mechanics is not relevant to the promise and experimental results of the field. The other group, however, stands firm to a particular stance. A leader in the field of quantum computing, David Deutsch is of the latter. Deutsch is a long-standing supporter of the MWI through his many years of research in the field of theoretical physics and quantum computing. Deutsch argues, if solutions are evaluated for each of its possible inputs simultaneously, then the Quantum Parallelism Theory (QPT) proposed by Armond Duwell must be true [21]. To account for the consistency in accepting this theory it is also necessary to accept the Many Worlds Interpretation of quantum mechanics. "For Deutsch, a quantum computer in superposition, like any other quantum system, exists in some sense in many classical universes simultaneously. These provide the physical arena within which the computer effects its parallel computations" [22].

One promise of quantum computers is, with the integration of quantum mechanics in its framework of computing, we will be able to understand more of the inner workings and nature of quantum mechanics itself. This leads to the conjecture that quantum computing may be a potential method to understanding and experimentally test interpretations of quantum mechanics. In other words, *will quantum computers prompt experimental metaphysics?* Can we experimentally test interpretations of quantum mechanics through simulations of quantum computers or simulations of quantum mechanics on quantum computers to determine the underlying mysterious nature of this questionable field of empirical science? Is there an answer to the measurement problem? The best answers to these questions are only speculation as of now. One would need to consider the computer's architecture before making any metaphysical conclusions, the architecture, in particular, being the realization of a large-scale quantum computer [22]. We are currently only at small-scale quantum computing. However, there are extremely promising theoretical capabilities at the large-scale, but the implementation of scaling quantum computers at this size to exhibit this breathtaking architecture and astounding capabilities will be the challenge of the millennia.

3.3 Can Machines Think?

The notion of artificial intelligence and conscious machines are widely popular in modern culture due to the progression and prospect of advancing technology. The science fiction genre has given us the HAL 9000 computer in 2001: A Space Odyssey, C-3PO and R2D2 in Star Wars, Ava in Ex Machina, Samantha in Her, Dolores in Westworld, and much more. It is no question that this is a fascinating subject, but what does it actually mean for a computer or an advanced artificially intelligent machine to think? In the famous 1950 paper, *Computing Machinery and Intelligence* [23], Alan Turing proposed a test to consider the question, "Can machines think?" He called it the "Imitation Game."

> It is played with three people, a man (A), a woman (B), and an interrogator (C) who may be of either sex. The interrogator stays in a room apart from the other two. The object of the game for the interrogator is to determine which of the other two is the man and which is the woman. He knows them by labels X and Y, and at the end of the game he says either "X is A and Y is B" or "X is B and Y is A" [23].

The interrogator asks questions directed at either person A or person B, and each person can respond however they choose. Now, consider if one person is replaced with a machine, and the interrogator is tasked to determine which is a real person and which is a machine. The central point of this test is to determine if a machine can convince someone of being human rather than a machine. It essentially tests how the machine performs in conversation. Turing also adds, that if a machine passes the test, we should *conclude* that it can think. But if it does not pass, then we should not draw any conclusion about its thinking capacity. Therefore, he offers the test as a *sufficient* condition for thinking, not a necessary one.

3.3.1 Two Objections to Turing's Test for Thinking

1) Lady Lovelace's objection

> "The Analytical Engine has no pretensions to originate anything. It can do whatever we know how to order it to perform." – Memoir of Lady Lovelace (1842) [23]
>
> Lady Lovelace argues that the fact that a machine can pass the Turing Test only shows that the machine has good programming, not that it thinks. Therefore, the Turing Test only shows the intelligence of the programmers. Lady Lovelace proposes a necessary condition for thinking: the machine must also evidence some sort of originality or creativity. There must emerge something *new*, in order to diverge from the fact that it was meant to do whatever action is carried out due to its programming.
>
> Turing responds by questioning the validity of what is to be accepted as original. How could you tell if something original or creative was created in the first place? "Who can be certain that 'original work' that he has done was not simply the growth of the seed planted in him by teaching, or the effect of following well-known general principles?" [23].

2) Argument from consciousness

> In accepting an entity as thinking, it is not enough for the system to simply *behave* as if it is thinking. The entity must actually *be* thinking and there is no way to determine this. We accept ourselves as thinking entities because of our ability to introspect. One may also consult the famous quote of philosopher René Descartes, cogito, ergo sum, or "I think,

therefore I am." We can doubt any information about the content of our existence but cannot doubt the fact that we exist. We cannot introspect the minds other than our own. Therefore, we cannot accept the Turing Test as viable to determine a being's capability to think from the fact that we can never know what it is like to be the machine.

Turing responds by pointing out that the logic of this argument can only lead to solipsism. Solipsism is the view that the self is all that can ever be known in the universe; therefore, you and you alone only exist. This objection implies that you cannot know that other people are thinking or conscious beings in the same way that you cannot know that a machine is a thinking or conscious being. Turing argues that it is reasonable to accept that a machine that passes the Turing Test is thinking, rather than deny other people's ability of thinking [23].

3.4 Will Quantum Machines Become Conscious?

With a great deal of mystery behind both quantum mechanics and consciousness, they must be connected somehow, right? Since the inner workings of a quantum computer include the mystifying nature of quantum mechanics, maybe the quantum computer is just an artificial concoction of human consciousness. Maybe the feeling of connection we receive when we are surrounded by friends and family is simply the interactions of our individual wavefunctions of consciousnesses. Or maybe, we are not asking the right question. Should we be asking whether quantum machines will become conscious, or whether quantum computing algorithms will lead to rapid advancement in the field of artificial intelligence and machine learning to the point of emergent consciousness? There has been great progress in computer science regarding its advanced algorithms and approaches to machine learning. But will there be a point where these machines or systems become conscious in the same way as us? Will the potentiality of quantum algorithm 'speed-ups' over classical computing algorithms exponentially outpace current classical methods of machine learning, to the point of emergent consciousness? Or is there something within us that cannot be programmed? For the rest of this section, we will explore how consciousness (or thinking) can be defined and whether consciousness is at all computable. (Note: although there are slight differences between the terms *thinking* and *consciousness*, I will be using them somewhat interchangeably).

3.4.1 What *is* Consciousness?

Firstly, how can consciousness be defined? Due to its elusive nature, no objective definition has been discovered and if one is presented to you, you should so be inclined to deny it. Consciousness is subjective in the sense that you can only ever be fully aware of your own experience of consciousness. The best you can know about someone else's experience of consciousness is to assume that other people are experiencing consciousness the same way as you. Philosopher David Chalmers distinguishes consciousness as two different types of problems. The *easy problems* are the phenomena explained in terms of computational or neural mechanisms. Also known as a *functional* understanding of consciousness. The *hard problem* of consciousness is coming up with a reason as to why there is consciousness at all. Or why consciousness gives rise to an inner subjective experience [15]. The *hard problem* of consciousness has everyone perplexed and seems to elude any sort of actual, empirical, or objective explanation. So, for now, we will tackle the *easy problems.*

3.4.2 Machine-State Functionalism

One way to understand how the mind works is functionalism. Functionalism is a non-reductive theory for understanding mental states (such as pain, anger, hunger, fear, and love) by claiming that mental states are defined by their functional role rather than their internal composition. In other words, the function of pain, such as a wincing and a quick retraction of your hand after touching a hot pan, is simply what pain *is.* By focusing on functional roles, the theory allows us to make sense of our behaviors without limiting ourselves to the scope of our own inner subjective experiences.

One type of functionalism, relevant to our discussion about thinking machines, is *machine-state functionalism.* In this view, "…the functionalist sees mental states as something that mediates between inputs and outputs" [24]. The output of the machine, therefore, depends on the current state of the machine as well as the input. In other words, our reactions may differ for the same stimuli depending on our current mental state. I will most likely have a different reaction to a candy bar when I am hungry versus when I am satiated. The state of the machine can be a result of a complex algorithm following a particular set of internal rules, or as simple as a binary state of 0 or 1, two possible states. In her book, *Philosophy of Mind: the basics,* Amy Kind offers a very simple and clear example of a machine that operates with internal states, a coin press machine. This machine accepts only one- and two-euro coins as inputs but requires two euros to receive a pressed coin as an output. If you enter a one-euro coin and the machine does nothing, the machine is in the first internal state, S1, waiting for another euro coin input. If you enter a one-euro coin and the machine dispenses a pressed coin, then the machine was in the S1 state before you put the coin in, and is now in S2, waiting for a two-euro coin input [24]. This example shows how the same input can result in a completely different output depending on the internal state of the machine. Additionally, this concept can be extended to systems with increasing complexity such as artificially intelligent systems. Where each programmable response with respect to its algorithm represents a different functional role. Or further, as a functional understanding of how the human mind operates.

Although machine-state functionalism does not address the *hard problem* of consciousness, it gives an ample remark to focus on the *easy problem* of consciousness by offering a simple analogy to the way that our mind works. So, what is left out that this functional understanding of the mind does not appeal to knowledge of the mystery of inner subjective experience?

3.4.3 What is it like to be…me?

Functionalism omits an important aspect of mentality known as *qualia*. Qualia is often understood as perceptual states that have a phenomenal feel, or that there is something it is like to have such states [24]. As Thomas Nagel states in his popular 1974 paper, *What is it like to be a bat?*, "The fact that an organism has conscious experience at all means, basically, that there is something it is like to be that organism" [25]. But what constitutes an organism? Does a machine mind have qualitative aspects of experience in the same way as humans? If a machine operates only concerning a set of rules, an algorithm, written by a computer programmer, then why would the machine have qualia if it is simply a system following rules?

To illustrate this point, philosopher Ned Block offers the scenario of a homunculi-headed robot [24]. On the outside, the robot looks and acts like a completely regular human being, but on the inside are buttons and control tables manned by little humanoid creatures called homunculi. Each homunculus has their own individual responsibilities and tasks to complete. One may control

a lever to move the robot's limbs when given a specific input or instruction to do so, and another may press a button to generate a joyous facial expression. These homunculi all working together each with their own responsibilities would fulfill the robot's functional organization. At a macroscopic level, the behavior of this robot, controlled by little homunculi creatures would behave exactly like a normal human being. But would a robot of this sort have qualitative states? Would the robot actually be in pain if it were to stub its toe? Block's answer is no, as there is prima facie doubt whether the machine/homunculi system has any mental states at all. At first glance, it seems obvious that the robot would not have qualia the same as we do. But aren't we made up of genes and living bacterial creatures that abide by the rules and laws of nature?

3.4.4 Is Consciousness Computable?

Ned Block would argue although we may be able to conceptualize the brain as software, consciousness cannot be computed. This is a fairly common position held by those who firmly choose a side. Notable figures include physicist Roger Penrose and neuroscientist Christof Koch. However, many others withhold judgment on the matter altogether. In his book, *The Feeling of Life Itself,* Christof Koch claims that consciousness cannot be simulated. A computer may be able to simulate conscious behavior and perfectly mimic the synaptic and neuronal events that occur when somebody sees a face or hears a voice. This simulated behavior will be indistinguishable from those of a human [26]. However,

> It won't see an image; it won't hear a voice inside its circuitry; it won't experience anything. It is nothing but clever programming. Fake consciousness—pretending by imitating people at the biophysical level [26].

This returns us to the same ideas that arose with the Turing test and the homunculi-headed robot example. Alan Turing would argue that this *fake consciousness* should be taken for consciousness. Since we can only ever have knowledge of our own inner subjective experiences, we must, to some degree, assume the inner subjective experiences of others. And we should accept machine's *fake consciousness* as consciousness for the same reason that we accept others as having consciousness. However, what authority do we have to apply our concept of consciousness to an entity inherently different in material makeup and metaphysical origins? Can't it be said that machines are conscious in their own way? Or that even simpler systems are conscious but possibly less conscious than us? (See panpsychism for more information [27]) What are the constraints to understand consciousness? Is consciousness inherent and belongs only to biological life? Is consciousness a fundamental property of nature that exists outside the laws of physics (David Chalmers [15])? We may never understand the truth of the metaphysical reality of the mysteriousness of consciousness and what gives rise to experience. But the field of quantum computing seems to give hope in one way or another for a potential understanding of the human mind; whether that is understanding the mind as a quantum machine, or that quantum computing will allow for discoveries of efficient algorithms to model the human brain.

"Life is not a series of gig lamps symmetrically arranged; life is a luminous halo, a semi-transparent envelope surrounding us from the beginning of consciousness to the end." - Virginia Woolf

Conclusion

"People know what they do; frequently they know why they do what they do; but what they don't know, is what what they do does." – Michel Foucault

The purpose of this paper was to give quantum computing an accessible examination and resolve several *myths,* or popular misconceptions. We undertook this task by developing a scientific narrative including an exploration into the concepts and frameworks of classical computing, quantum mechanics, computational complexity, and finally, philosophy.

The conception of a classical computer, or Turing machine, in theory, gave a constructive framework to understand computability and the internal states of the machine, namely the bit. This framework provided the foundation to conceptually grasp the qubit when quantum mechanics was introduced. Although the *why* of quantum mechanics is and has always been, a topic of evasion, the pertinent notions of interference, superposition, and entanglement have contributed to the possibility of quantum information and its remarkable discoveries in the field of computation.

In providing clarity and insight into the distinction between the problem-solving capability of quantum computers and classical computers, we employed the theory of computational complexity. This theory presented a method to categorize computable problems based on their required resources. One resource of particular interest to us was time. This scheme of categorization addressed the common notion of quantum 'speed up' over classical computers. Quantum computers can solve specific computable problems in less time than classical computers, but as we found, this is not the case for all problems. That is to say, quantum computers are computationally more efficient for *some* problems than classical computers.

Lastly, it is believed that any problem solvable by a quantum computer is also solvable by a Turing machine. So, quantum computers are not known to be capable of solving problems that are not solvable by current classical computers. Although the obstacle of scalability is constantly in the back of our minds, the field brings exciting possibilities for the future of technology as further progress is continuously being made in quantum information and the experimental implementation of the creation of the computer itself.

References

[1] Deutsch, D. "Quantum Computational Networks." *Proceedings of the Royal Society of London*. Series A, Mathematical and Physical Sciences, vol. 425, no. 1868, 1989, pp. 73–90. JSTOR, doi: www.jstor.org/stable/2398494. Accessed 1 Mar. 2021.

[2] de Wolf, Ronald. Quantum Computing: Lecture Notes. University of Amsterdam. January 2012. Accessed 27 Feb 2021.

[3] Sipser, Michael. *Introduction to the Theory of Computation*. Course Technology Cengage Learning, 2013.

[4] Nielsen, Michael A., and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

[5] Aaronson, Scott. "The Limits of Quantum." *Scientific American*, vol. 298, no. 3, Mar. 2008, pp. 62–69. JSTOR, doi: https://www.jstor.org/stable/26000518?seq=1.

[6] Stark, Glenn. "Young's Double-Slit Experiment." *Encyclopædia Britannica*, Encyclopædia Britannica, Inc., 26 July 1999, www.britannica.com/science/light/Youngs-double-slit-experiment.

[7] Roell, Jason. "Demystifying Quantum Gates - One Qubit At A Time." *Medium*, Towards Data Science, 28 Feb. 2018, towardsdatascience.com/demystifying-quantum-gates-one-qubit-at-a-time-54404ed80640.

[8] Lloyd, Seth. "Almost Any Quantum Logic Gate is Universal." *Physical Review Letters*. Vol. 75, no. 2, 10 July 1995. pp. 346-349.

[9] The Qiskit Team. "Learn Quantum Computation using Qiskit." *Qiskit*, Data 100 at UC Berkeley, 2020, https://qiskit.org/textbook.html. Accessed 2 March 2021.

[10] Voorhoede, D. "What Is a Quantum Algorithm?" Quantum Inspire, QuTech, 4 Sept. 2018, www.quantum-inspire.com/kbase/what-is-a-quantum-algorithm/.

[11] Christandl, Katharina. "Advancing Neutral Atom Quantum Computing: Studies of One-Dimensional and Two-Dimensional Optical Lattices on a Chip." *Dissertation: The Ohio State University*. 2005.

[12] Netburn, Deborah. "What Google's 'Quantum Supremacy' Means for the Future of Computing." Phys.org, Phys.org, 25 Oct. 2019, phys.org/news/2019-10-google-quantum-supremacy-future.html.

[13] Arute, F., Arya, K., Babbush, R. et al. Quantum supremacy using a programmable superconducting processor. Nature 574, 505–510 (2019). https://doi.org/10.1038/s41586-019-1666-5.

[14] Deutsch, D., and R. Jozsa. 1992. “Rapid Solution of Problems by Quantum Computation,” Proc. Roy. Soc. Lond. A, 439: 553–58.

[15] Chalmers, David. “Facing Up to the Problem of Consciousness.” *Journal of Consciousness Studies*, 2(3):200-19, 1995.

[16] Aaronson, Scott. “Why Philosophers Should Care About Computational Complexity.” *Computability: Gödel, Turing, Church, and beyond*. MIT Press, 2012. https://arxiv.org/abs/1108.1791.

[17] Aaronson, Scott. “Complexity Zoo.” 18 November 2012. https://complexityzoo.net/Complexity_Zoo.

[18] Aaronson, Scott. *Quantum Computing since Democritus*. Cambridge University Press, 2013.

[19] Lewis, Peter J. “Interpretations of Quantum Mechanics.” *Internet Encyclopedia of Philosophy*, iep.utm.edu/int-qm/#H2.

[20] Levin, L., 2003. “Polynomial Time and Extravagant Models,” Problems of Information Transmission, 39: 2594–7.

[21] Duwell, A., 2007. “The Many-Worlds Interpretation and Quantum Computation,” Philosophy of Science, 74: 1007–18.

[22] Hagar, Amit and Michael Cuffaro, "Quantum Computing", The Stanford Encyclopedia of Philosophy (Winter 2019 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/win2019/entries/qt-quantcomp/>.

[23] Turing, Alan M. “Computing Machinery and Intelligence.” *Mind.* 49: 433-460. 1950.

[24] Kind, Amy. *Philosophy of Mind: the Basics*. Routledge, Taylor & Francis Group, 2020.

[25] Nagel, Thomas. “What is it like to be a bat?” *The Philosophical Review* , Oct., 1974, vol. 83, no. 4, pp. 435-450. doi: https://www.jstor.org/stable/2183914?seq=1.

[26] Koch, Christof. The Feeling of Life Itself*: Why Consciousness Is Widespread but Can't Be Computed*. MIT Press, 2020.

[27] Goff, Philip, William Seager, and Sean Allen-Hermanson, "Panpsychism", The Stanford Encyclopedia of Philosophy (Summer 2020 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/sum2020/entries/panpsychism/>.