Honors College Thesis
Submitted in partial fulfillment of the requirements for graduation
from the Honors College

An introduction to applied quantum computing for undergraduate computer science students

Angel Bajracharya
Kees Leune
Sukun Li, Matthew Wright
5/8/2024

Table of Contents

Abstract **1**
Introduction **2**
Relevance **3**
Preparing the Next Generation of Quantum Computer Scientists 3
Undergraduate Curriculums 4
Harnessing Quantum Advantage 4
Relevance in the Technology Industry 5
Interdisciplinary Impact 6
Research Problem **6**
Objectives and Scope **7**
Methodology **8**
Literature Review 9
History of Computing **10**
Timeline of Classical Computing 10
Timeline of Quantum Computing 13
Classical Computing Fundamentals **15**
Boolean Algebra and Logic Gates 15
Von Neumann Architecture 17
Time Complexity 18
Quantum Computing Fundamentals **19**
Definition of a Quantum Computer **19**
Superposition 20
Entanglement 21
Quantum Parallelism 22
Oracles 22
Quantum Gates **23**
Hadamard Gate 23
CNOT gate 24
Quantum Algorithms **25**
Deutsch-Jozsa Algorithm 25
Grover's Algorithm 26

Shor’s Algorithm 26
Quantum Software 27
IBM Quantum Experience 27
IBM Qiskit 28
Google Cirq 29
Microsoft Quantum Development Kit and Q# 30
Experimental Design 31
Design of Lecture 31
Design of Lab Packet 33
Research Findings 37
Lab Packet Findings: 38
Interpretation 41
Future Considerations 42
Conclusion 43
Appendices 45
Appendix A: Lecture Slides 45
Appendix B: Recording and transcript of lecture 55
Appendix C: Source Code 55
Appendix D: Lab Packet 55
Appendix E: Feedback Survey 61
References 62

Abstract

With quantum technologies reaching unprecedented heights, the necessity for a larger quantum workforce becomes imperative. My research examines how to introduce quantum computing to undergraduate computer science students without specialized math and physics background knowledge. Although quantum computing has been a concept that has been around a long time, quantum computer science as a discipline is still in its infancy. This urges further research on the topic of educating the younger generation on this computing paradigm. Unlike the classical computing paradigm that is used by modern computers, quantum computing uses principles from quantum mechanics such as entanglement and superposition to increase efficiency in solving computational problems. Certain problems can be solved with less resources such as time and space when using quantum computing algorithms as opposed to classical computing algorithms. My research aims to conduct a quasi-structured literature review that outlines the best types of algorithms to teach to undergraduate computer science students. My research was used to create a lecture plan for an undergraduate computer science course at Adelphi University. This research contributes to the growing demand for accessible quantum computing education by young computer scientists seeking to navigate the continuously evolving technology industry.

Introduction

Classical computers refer to the modern devices that are governed by principles of classical physics and used in everyday life. Quantum computers are specialized computers that utilize principles of quantum mechanics such as superposition and entanglement (Bernhardt, 2019, p. 182). The recent rapid development of quantum technology urges further research on the discipline of quantum computer science. While there is a vast corpus of scholarly research on the physics and mathematical understanding of quantum computing, there is a gap in the discussion about how the computer science community will take advantage of the breakthroughs that quantum computing promises, particularly in the education of undergraduate computer science students on the topic (Liu & Franklin, 2023). A crucial aspect of classical computer science education is the reinforcement of the algorithm. An algorithm is a well-defined procedure with a set of instructions that processes information. Algorithms allow students to apply concepts of classical computing to create a solution to a given problem. Just like we teach classical computing with algorithms, there are a handful of quantum algorithms to explore (Nielsen & Chuang, 2000, p.7). My study will highlight the best quantum algorithms to introduce to computer science students. These algorithms demonstrate the key principles of quantum computing and prove that there are instances where quantum computing is faster than classical computing.

Relevance

Preparing the Next Generation of Quantum Computer Scientists

Despite its potential computational capabilities, the quantum computer is still in its infancy (Mykhailova & Svore, 2020). It is at an early stage where goals such as curing cancer (Pflitsch, 2022) and climate change (Ladopoulos, 2023) are not yet feasible although heavily advertised in the media. It is important to note that the sources cited in the previous statement are not peer-reviewed academic sources but are included to illustrate fallacies in media on the topic of the benefits of quantum computing.

The field of quantum computer science, which applies quantum mechanics principles to information processing, is growing rapidly. There needs to be more efforts and resources to expose the younger generation of computer scientists to the quantum paradigm to prepare them for the new possibilities that quantum computing promises. The next generation of quantum computer scientists should be taught about quantum computing using credible sources and high-level organization rather than the non peer-reviewed sources that are currently available to them, which is why conducting research of this nature is so important.

Undergraduate Curriculums

This research specifically targets undergraduate students because they are at the crux of innovation and they should be given the opportunity to learn about this cutting edge technology. Educating computer scientists on quantum computing at its primitive stage will allow them to gain a strong foundational understanding of this new paradigm. Undergraduate students are the emerging tech workforce and it is essential that they are properly trained and equipped to make further developments in quantum technologies.

Harnessing Quantum Advantage

According to Ivan H. Deutsch (2020), the emergence of quantum computer science as a field is a manifestation of the second quantum revolution. The first quantum revolution introduced the field of quantum physics to the world and led to the development of transistors and modern computers. The second quantum revolution is occurring as researchers develop new ways to apply the power of quantum mechanics to the real world. Quantum computing harnesses the power of quantum mechanics to increase efficiency in solving computational problems (Deutsch, 2020).

The focus of my research is to make quantum computing more accessible to a younger generation because the realization of quantum advantage could have a significant impact on the technology industry. Quantum advantage is achieved when a quantum computer can solve problems faster than a classical computer (Bernhardt, 2019, p. 187).

Relevance in the Technology Industry

Leading technology companies such as IBM have met many milestones in quantum development. By the end of 2023, IBM released Condor, the largest quantum computer in the world at its time, with 1,121 functional qubits. As the leading company in quantum research, IBM promises significant developments in quantum technology within the next decade. With these timely developments, quantum computers hold great promise for exponential speedup in information processing (IBM, 2024). In addition, according to a blog post by Apple's Security Research team, Apple has released a new version of iMessage using the PQ3 protocol in February 2024. PQ3 is designed to provide end-to-end encryption and protect against attacks from future quantum computers (Apple Security Engineering and Architecture [SEAR], 2024). Although quantum computers do not have the capability of performing these attacks yet, Apple's focus on implementing preventative security measures demonstrates the promise and threat of quantum technology advancements. Recent spikes in the development of quantum technologies demands further research on quantum computing and the best pedagogical strategies to introduce this topic to undergraduate students with limited prior exposure to the topic.

Interdisciplinary Impact

Quantum computing is not only revolutionizing the landscape of the technology industry but also promising breakthroughs in other fields such as the sciences, the economy, and society (Seegerer et al., 2021). Quantum computer science is at the intersection of math, physics, and computer science (Mykhailova & Svore, 2020). Making further advances in this field also has the ability to affect other fields such as medicine and finance.

Research Problem

The goal of my research is to find the type of quantum computing problems that undergraduate computer science students can effectively solve. By addressing my research problem, this study aims to find the best approach to introduce the topic of quantum computing to undergraduate computer science students in a methodical, evidence-based manner. It also aims to contribute educational material and key insights to the growing field of quantum computer science.

Objectives and Scope

Currently, most quantum computing programs are designed in the mathematics or physics department where the material tends to focus heavily on quantum computing theory rather than application (Meyer et al., 2022). This research abstracts the heavily theoretical aspects away from quantum computing and aims to find a practical intersection of math, physics, and computer science that can allow computer science students to apply quantum computing as they would classical computing. The scope of my research is limited to the mathematical and physics concepts necessary for comprehending quantum computing principles.

Furthermore, most current quantum computing programs are taught at the graduate level, which impedes the accessibility of quantum computing (Liu & Franklin, 2023). This research will determine the appropriate educational standard for teaching quantum computing to undergraduate computer science students. The goal is to provide students with an introduction to the main principles of quantum computing and allow them to apply their knowledge practically by running and writing their own quantum programs. The students were expected to have prior knowledge of classical computing due to the prerequisite courses for the course that the introductory lecture was presented at.

Methodology

In order to assess the types of problems that computer science students are able to solve, a study will be conducted to assess the effectiveness of introducing educational material on the topic. The study will consist of a quasi-structured literature review of the existing educational materials designed for the topic of quantum computing followed by an experiment that implements the best concepts and algorithms needed to introduce quantum computing to undergraduate computer science students. The literature review contains scholarly works such as peer-reviewed research papers, articles, and academic books, curriculums, and syllabi. This research will be used to determine:

- The appropriate educational standard for instructing students–specifically in the computer science department–in the field of quantum computing.

- The prerequisites required (and not required) before introducing quantum computing to computer science undergraduate students.

Then, a gap analysis will be conducted to determine what material can be used to demonstrate quantum computing and a lecture will be designed to holistically address this gap in a classroom setting. Following the lecture will be a lab session designed to evaluate the students' engagement with the topic. The lecture and lab design will use Bloom’s Taxonomy (Adams, 2015) as its main guiding framework. Moreover, a feedback survey will be conducted to assess student opinion on the new topic and its presentation.

Literature Review

The first step in the research process was conducting a thorough literature review to identify key concepts, relevant quantum computing problems, and informative quantum algorithms that could be used to introduce the topic of applied quantum computing to undergraduate computer science students. The literature review consisted of a plethora of sources such as textbooks and quantum curriculums. The result of the literature review conducted was a lecture and lab presented in two classes with a length of an hour and fifteen minutes each. The lecture introduced the preliminary concepts needed to build a foundation of understanding for computer science students. The lab assessed the understanding and retentivity of concepts as well as gave the students an outlet to apply their knowledge.

History of Computing

The modern approach to processing information is known as classical computing. Classical computing relies on classical principles of physics and mathematics. The application of quantum mechanics on computation and information processing is what led to the emergence of quantum computing. The following is an overview of the origins and important milestones of classical and quantum computing.

Timeline of Classical Computing

In the 1930's, George Boole defined boolean logic as a method of representing logical decisions as either true or false (Bernhardt, 2019, p. 89). His work on boolean algebra laid the mathematical foundation for the classical bit, or "binary digit." A bit is an atomic unit of information that gets stored in classical computers that can either have a value of 1 or 0 which is associated with true and false respectively. According to Nielsen and Chuang (2000), another one of the key events that directly impacted the emergence of the computer science discipline was the Church-Turing Thesis, published in 1936. Alan Turing and Alonzo Church published a theoretical model for computation which served as the foundation for classical computation theory. The key assertion in their paper was that a problem could be considered "computable" by any computer if and only if it could be solved by a simple and imaginary machine called the Turing machine and conversely, a problem is computable by the Turing machine if and only if it is computable by some piece of hardware (Kaye et al., 2007, pp. 2-3). The main takeaway from this assertion was that a universal definition of a “computable” algorithm was established (Nielsen & Chuang, 2000, p. 4). Any algorithm used to solve a problem on the Turing machine can be computable on any computational device. While the Church-Turing thesis laid the groundwork for computation theory, the von Neumann architecture (Godfrey, 1992) was created as a theoretical model for the practical application of creating a general-purpose computer that would work as the Turing machine. The main components in the von Neumann model are the Central Processing Unit (CPU), the Main Memory Unit, and the Input/Output Device (Godfrey, 1992). The role of the von Neumann architecture will be discussed further in Von Neumann Architecture (see Figure 2).

ENIAC (Electronic Numerical Integrator and Computer) was built by J. Presper Eckert and John Mauchly in 1945, and it became the first programmable electronic computer (see Figure 1). The same inventors of the ENIAC designed the UNIVAC I (Universal Automatic Computer I), the first commercial classical computer, which belonged to the United States Census Bureau in 1951. After the 1960’s, rapid developments in the technology industry made computers more commonplace in households. In recent years, according to a 2024 survey by Parks

Associates, the average US household has seventeen computing devices connected to the Internet (Parks Associates, 2024). This shows the exponential growth and use of classical computing devices in everyday life.

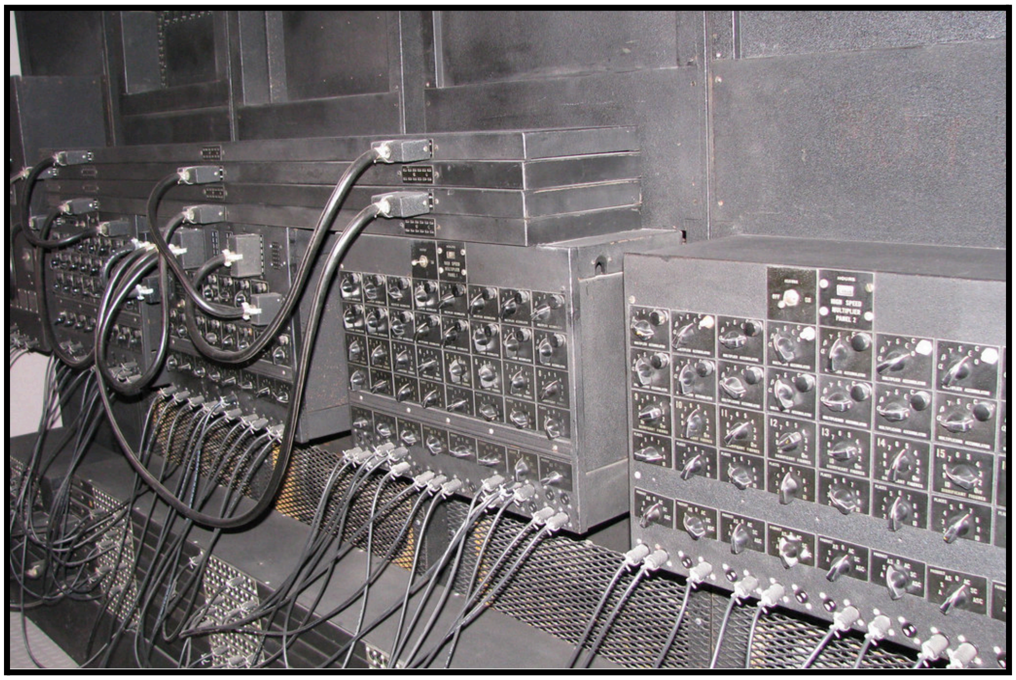


Figure 1. ENIAC (Electronic Numerical Integrator and Computer).

Timeline of Quantum Computing

According to Chen (2023), the origins of quantum computing can be traced back to the early 1900's when scientists such as Albert Einstein and Niels Bohr discovered the fundamental principles of quantum mechanics. Then, in the 1980's, the works of scientists like Richard Feynman, Paul Benioff, and David Deutsch led to the emergence of quantum computing as its own field of research. The 1990's were a period of growth for quantum algorithms that proved quantum advantage over classical (Chen, 2023). These algorithms included Deutsch-Jozsa, Shor's, and Grover's algorithm which get discussed further in the section Quantum Algorithms. Finally, from the year 2000 to present, there has been a race between the biggest technological powerhouses–such as IBM, Google, Microsoft, and Apple–to build a quantum computer that realizes the potential of quantum advantage.

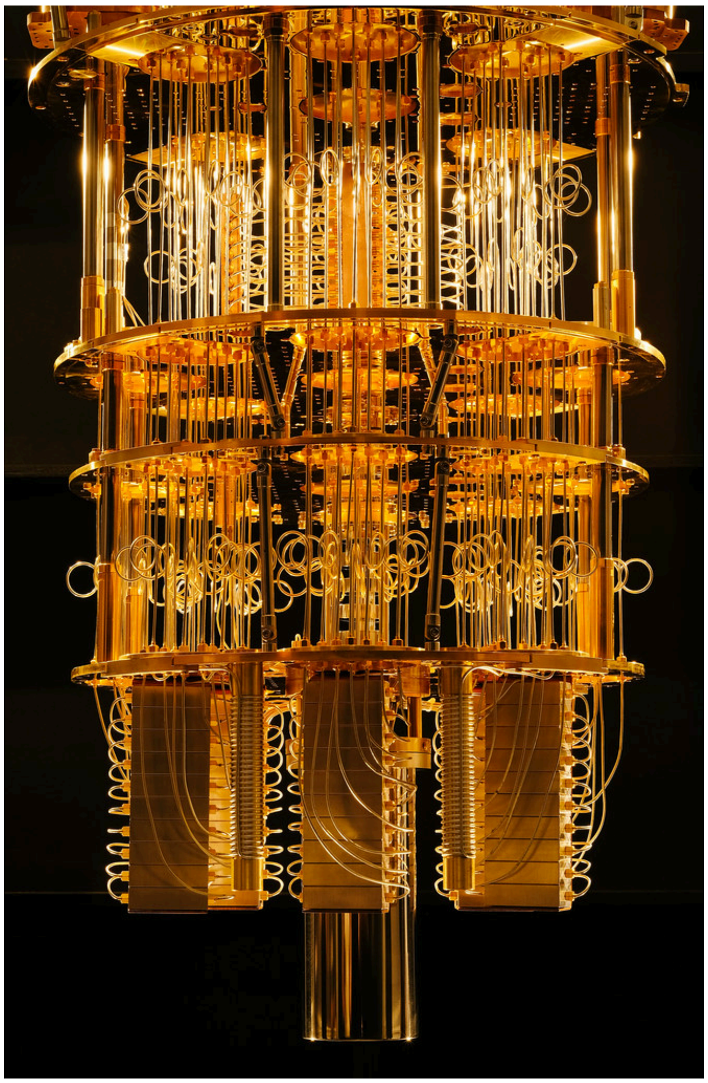


Figure 2. Quantum Computer.

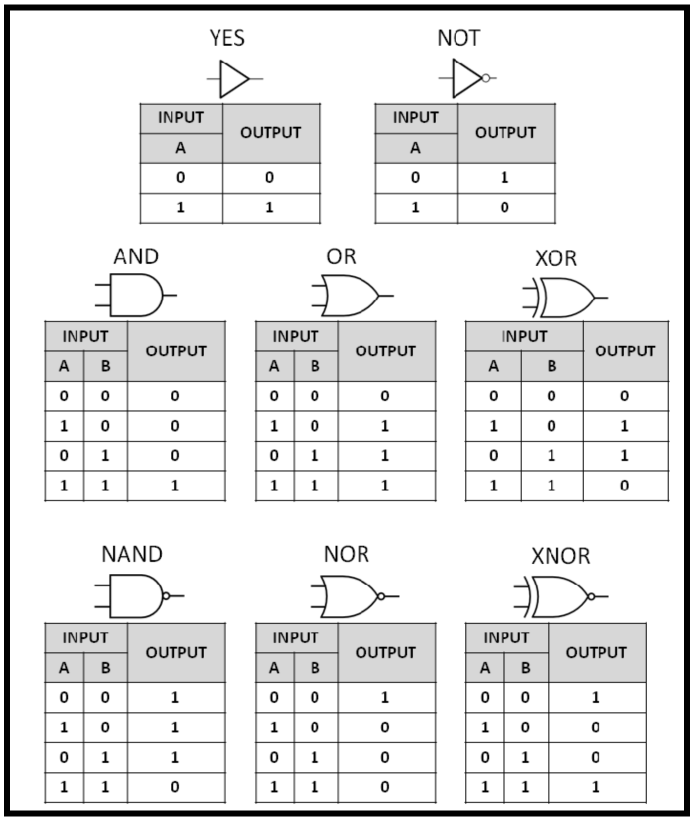
Classical Computing Fundamentals

Classical computing refers to the computing paradigm that exists in most if not all of the modern computing devices. In order to understand quantum computing, it is important to provide a review on the fundamentals of classical computing.

Boolean Algebra and Logic Gates

As discussed above, boolean algebra serves as the mathematical foundation for information processing. Boolean algebra also laid the mathematical foundation for logical gates (e.g., AND, OR, NOT, XOR) which allow information to be processed and manipulated in order to solve computational tasks. Each of the logical gates have corresponding truth tables that display the inputs and outputs.

Classical computing makes heavy use of if-else statements and iterative loops which all rely on boolean logic to compute. Classical algorithms are able to execute with reliable outputs because of these mathematical predefined logical truths. For instance, in this if-statement, `if (A AND B) → C`, the statement relies on the logical truth that the AND operator means that both A and B have to be true in order for C to be true. Logical gates are fundamental to manipulating logical bits and allowing computer algorithms to run.



YES

INPUT	OUTPUT
A	
0	0
1	1

NOT

INPUT	OUTPUT
A	
0	1
1	0

AND

INPUT		OUTPUT
A	B	
0	0	0
1	0	0
0	1	0
1	1	1

OR

INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	1

XOR

INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	0

NAND

INPUT		OUTPUT
A	B	
0	0	1
1	0	1
0	1	1
1	1	0

NOR

INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	0

XNOR

INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	1

Figure 3. Summary of the common Boolean logic gates.

Von Neumann Architecture

The von Neumann architecture (see Figure 3) is a high-level framework that organizes the components of a computer (Godfrey, 1992). It shows us how data moves within the computer. Data is stored in the memory and represented using binary numbers. The data, which is stored with instructions on what to do with it, is fetched by the control unit of the processor, also known as the Central Processing Unit (CPU) (Godfrey, 1992). The instruction dictates how the data gets processed. The instruction is executed in the arithmetic logic unit which manipulates the bits using logical gates to perform computational operations.

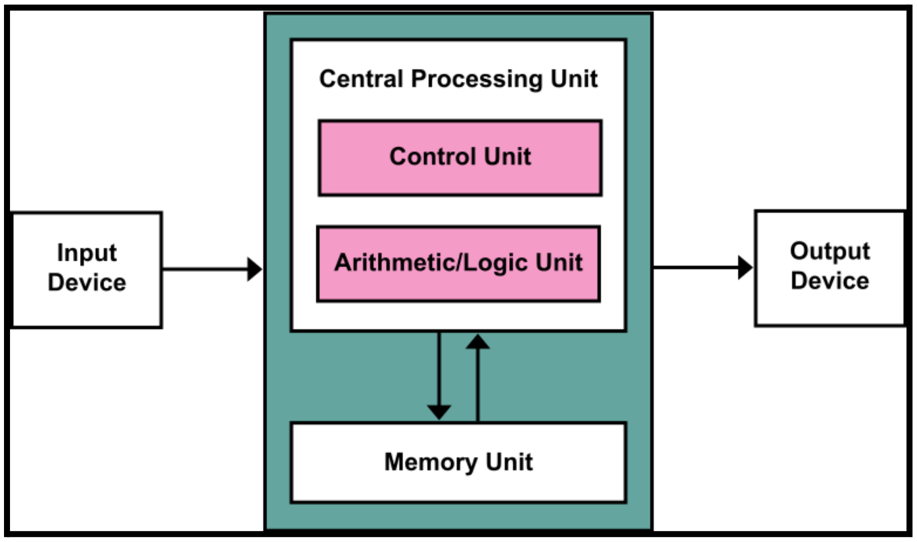


Figure 4. Von Neumann Architecture.

Time Complexity

One of the main branches of classical computing theory is the study of algorithms and the computational resources they take up (Nielson & Chuang, 2000, p.120). An algorithm is a well-defined procedure that processes information. In order to maximize the efficiency of computations, computer scientists attempt to measure the amount of resources that an algorithm takes. They calculate the time complexity–the time it takes for a computation to perform–and the space complexity–the memory a computer takes up to perform a computation. Each computing device may use a different amount of resources to execute the same algorithm which is why computer scientists use a "coarse" calculation of the complexity that only involves the term of the highest degree. In order to abstract the complexity of algorithms away from the physical architecture, computer scientists characterize complexity in terms of Big O notation (O) which represents the worst-case scenario, Big Omega notation (Ω) which represents the best-case scenario, and Big Theta notation (Θ) which is a balance between the two (Kaye et al., 2007, "Introduction and Background").

Quantum Computing Fundamentals

To understand quantum computer science, it is important to first understand the governing principles of quantum computing. Quantum computers encode information in the form of qubits, or quantum bits (Liu & Franklin, 2023). Quantum computing relies on two main concepts in quantum mechanics called superposition and entanglement (Seegerer et al., 2021).

Definition of a Quantum Computer

According to David Mermin (2007), the definition of a quantum computer cannot be reduced to a device that is governed by quantum mechanics principles because all physical phenomena are governed by quantum mechanics to some extent. There are several quantum mechanics concepts that work together to make a quantum computer work. Firstly, there is the principle of superposition which allows an individual qubit to be in more than one state simultaneously. The qubit can be in a state of one, zero, or a superposition of the two. Another concept that is instrumental in quantum computing is entanglement, or the intrinsic linking of multiple qubits to each other. The measurement of the state of one qubit will immediately impact the other entangled qubit. This is significant because processing an individual qubit can also give information about the other qubits entangled to it at the same time. A quantum computer is able to harness the principles of superposition and entanglement to increase the amount of information a computer can store exponentially in a phenomenon called quantum parallelism (Kaye et al., 2007, p. 94). As the name suggests, quantum parallelism allows for a quantum computer to have access to multiple states of qubits in parallel, which allows for the computer to discover multiple solutions for a quantum algorithm at the same time. Quantum parallelism may seem ideal in theory but achieving it is very difficult because it requires a state of coherence in the quantum computer. According to Mermin (2007), all physical interactions in a quantum computer need to be controlled to the last detail. Any environmental changes that would be normal in a classical computer could cause catastrophic results in a quantum computer, such as air particles bouncing off the system and small amounts of thermal energy. These catastrophic interactions result in decoherence. In order to reach a state of coherence, the quantum computer must be contained in a steel case at absolute zero temperature.

Superposition

In the traditional computing paradigm, computers store information in the form of binary bits that can represent two states, 0 or 1. Superposition allows the qubit to be in both states of 0 and 1 simultaneously. A popular analogy used to explain this phenomenon is a coin (Liu & Franklin, 2023). When you flip a coin, it can either land on heads or tails. This is like the classical bit that represents one of two states. However, if you are spinning the coin, in the moment that it is spinning, the coin is both heads and tails at the same time. This spinning state represents the revolutionary qubit. The ability to be in multiple states at the same time (until measured) is called superposition. The qubit can be in a state of one, zero, or a superposition of the two states. It is important to note that once the spinning is stopped, just like the coin stops on either heads or tails, the qubit also “stops” on 0 or 1. This is important to note because the qubit is only in a state of superposition until it is stopped for observation.

Entanglement

Entanglement represents the dependent nature of qubits where the outcome of one qubit is directly correlated to a qubit that is entangled to it. In the case where two classical bits are combined, you need to know both of the bits’ information to know what state it is in. For example, you can store 00, 01, 10, and 11. Quantum computing is able to harness the powers of superposition and entanglement to complete the same task with fewer operations due to the fact that more information can be represented by a smaller amount of qubits (Bernhardt, 2019, p. 141).

Quantum Parallelism

Quantum parallelism is achieved with the combination of superposition and entanglement (Kaye et al., 2007, p. 94). It refers to the ability for a quantum computer to explore multiple solutions simultaneously, unlike a classical solution that needs to iterate through the multiple solutions at separate instances. Quantum parallelism is what allows quantum programs to have an advantage over classical programs.

Oracles

Quantum algorithms make use of oracles. The oracle is represented by a black-box mechanism whose internal workings are hidden and unknown (Bernhardt, 2019, p. 145). Given an input, an oracle will return an output. The output indicates to the quantum circuit when a correct solution to a problem has been found.

Quantum Gates

Hadamard Gate

One of the most important quantum gates is the Hadamard gate (see Figure 5). It allows the qubit to go from a classical state to a state of superposition. While there are mathematical theories associated with this gate, this paper will not delve into them because the objective was to see whether students would be able to apply quantum gates with minimal knowledge on their standing in the physics and mathematics realms (see Figure 4).

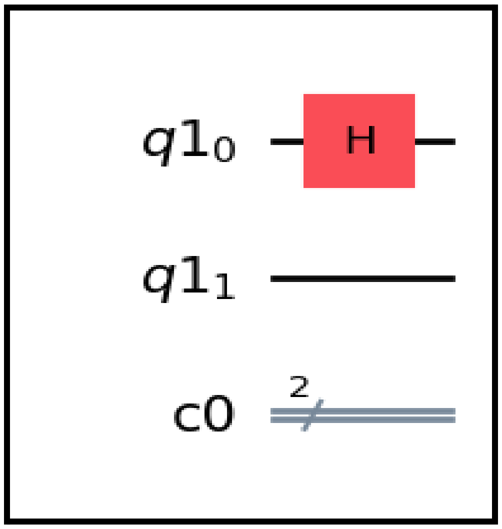


Figure 5. Circuit Diagram of Hadamard Gate Applied to Qubit.

CNOT gate

The CNOT gate represents an if-then in quantum computing. If the control qubit (qr[0]) is in state |0⟩, nothing happens to the target qubit (qr[1]). If the control qubit (qr[0]) is in state |1⟩, the target qubit (qr[1]) is flipped, i.e., if it was |0⟩ it becomes |1⟩, and vice versa. The classical equivalent to that is the XOR gate where true (1) if the number of true inputs is odd and false (0) if the number of true inputs is even.

Quantum Algorithms

My research focuses on the types of problems that computer scientists can solve using quantum computing. In the discipline of computer science, an algorithm is a set of instructions that a computer executes to perform computations. Algorithms serve as the solutions in the problem-solving process. Hence, my research considers several quantum algorithms that demonstrate quantum advantage in solving certain types of problems.

Deutsch-Jozsa Algorithm

The Deutsch-Jozsa algorithm was proposed by David Deutsch and Richard Jozsa in 1992. It is one of the earliest examples of a quantum algorithm that solves a problem exponentially faster than its classical solution (Deutsch & Jozsa, 1992). It solves the Deutsch-Jozsa problem of determining whether a function *f* is constant or balanced by sending an n-bit string through an oracle (Johansson & Larsson, 2017). As mentioned in the section Oracles, the oracle is represented by a black-box whose internal workings are hidden and unknown. Given an input, an oracle will return an output. If the function was constant, it would return all 0's or all 1's and if the function was balanced, it would return an equal number of 1's and 0's (Johansson & Larsson, 2017). The classical solution for this problem requires at least 2n-1+1 queries but the quantum solution, also known as the Deutsch-Jozsa algorithm requires 1 query to the oracle.

Grover’s Algorithm

Grover’s algorithm was developed by Lov Grover in 1996. This quantum algorithm allows for more efficient unordered searches compared to classical search algorithms (Kaye, et al., 2007, p. 154). For classical solutions, the computer iterates through the entire array until it finds the entry it is looking for. By using Grover’s algorithm, we reduce the time complexity from O(N) to O(√N). A quadratic speedup in database search is significant and can be applied to many real-life scenarios since search algorithms are used in most modern applications (Kaye, et al., 2007, p. 154).

Shor’s Algorithm

Shor’s algorithm was developed by mathematician Peter Shor in 1994. This algorithm factorizes large composite numbers into their prime factors rather than a classical algorithm (Kaye, et al., 2007). The RSA cryptosystem is a key component of most encryption algorithms used in modern computing devices, and it relies on the computational complexity of factoring large numbers into their primes (Kaye, et al., 2007, p. 130).

Quantum Software

IBM Quantum Experience

IBM is one of the leading contributors to the research and development of quantum computing. On May 4, 2016, IBM made history by releasing IBM Quantum Experience (IBM-QE), which was the first cloud-based quantum processor available for the general public. This allowed people to send quantum algorithms over the cloud to be run on a 5-qubit quantum computer (Mandelbaum, 2021). IBM continued to devote countless resources on furthering quantum computing development. The benefit of IBM-QE was that it allowed accessibility to quantum computing. Researchers, students, and the average public were able to remotely access quantum computers, which used to be restricted to specialized researchers. With more accessibility came more developments in the quantum computing field. The drawback was the limit in size of the quantum computer. The public was only given 5 qubits to experiment with. Quantum circuits could be run on the ideal processor or the real processor. The ideal processor ran quantum algorithms without any “unwanted disruptions.” The real processor ran quantum algorithms with the disruptions, simulating the decoherence in a real quantum computer. Despite a limit in the size of the quantum computer, IBM had accomplished a significant milestone that pushed quantum computing further to the public eye. IBM stands as a research and development powerhouse in the quantum computing industry. In recent years, IBM grants access to 100-qubit quantum systems to execute quantum algorithms by countless users. After unveiling the 400-qubit quantum processor in 2022, IBM revealed plans to release a quantum processor featuring 1,121 qubits in 2023 (Gambetta, 2023).

IBM Qiskit

Today, IBM hosts a plethora of cloud-based quantum systems such as IBM Quantum Composer, IBM Quantum Lab, and Qiskit. Qiskit is an open-source software development kit (SDK) that provides an importable library which allows working with quantum circuits, quantum operators, and primitive functions. It was released by IBM in 2017. It is written mainly in Python which makes the language accessible to most coders. There are numerous benefits to Qiskit. It is open-source and written in a commonly known programming language which makes it very accessible. There are numerous tutorials and documentation available which makes Qiskit a great educational tool. Specifically, in the scope of the research, a tool like Qiskit is a possible resource to introduce to undergraduate computer science students to teach them how to develop quantum algorithms without focusing too much on the theoretical aspects. Another benefit is that users can write and execute quantum algorithms on IBM's 100-qubit processor without concerns about the decoherence of the physical system. A slight disadvantage would be that the quantum processor used to run Qiskit programs has a limited amount of qubits. Although IBM has created a quantum processor featuring 400 qubits, the processor behind Qiskit only uses 100 qubits. This means that there is a limit to the computational resources available through Qiskit. However, it is still a valuable tool in quantum computing education.

Google Cirq

Shortly after IBM released Qiskit, Google released Cirq which is a similar Python-based library that was created for simulating quantum algorithms. Cirq currently provides the computational capabilities to run a quantum simulation of 20 qubits and supports the execution of quantum circuits on Google's quantum computers ("Cirq," n.d.).

Both Qiskit and Cirq are valuable resources in facilitating the education of quantum computing. They are easy to read and easy to integrate to numerous environments such as Jupyter Notebooks and Google Colaboratory (Mehar, 2024). They provide extensive documentation and beginner tutorials. They were both close contenders when it came to choosing the quantum platform to introduce to undergraduate computer science students.

Microsoft Quantum Development Kit and Q#

Another prominent quantum platform is Microsoft's Quantum Development Kit (QDK). As opposed to IBM's Qikit which relies on Python, Microsoft's QDK uses the Q# programming language. While Python is a general programming language that is easy to read, Q# is a high-level domain-specific language that uses statements and expressions like most classical programming and abstracts away from quantum states and circuits (Mehar, 2024). Research done by Mariia Mykhailova and Krysta Svore, two influential voices in the quantum computing industry, had a similar objective to my research where it aimed to focus on the software and programming aspect of quantum computing and had a target audience of undergraduate computer science. They introduced Q# and the Microsoft QDK to their curriculum. While these tools are appropriate for a 10-week long course, such as the one taught by Mykhailova and Svore, the scope of my research is significantly more limited. For a lecture and lab session, it was unfeasible to introduce Q#.

Experimental Design

Design of Lecture

The main goal in the lecture design process was to introduce key quantum computing concepts with a deviation from heavily theoretical focus. The goal was to choose as many topics as necessary to prepare the students to write and execute their own quantum algorithms. In order to enhance student engagement, the slides were designed to include a mixture of various modes of learning such as lecture-style presentations, lab exercises, and multimedia resources like images and GIFs (see Appendix A: Lecture Slides). The overarching objectives of the presentation were to go over the following:

- What is Quantum Computing?
  - Superposition
  - Entanglement
  - Quantum parallelism
  - Oracle - Black box mechanism
- Examples of Quantum Gates
  - Hadamard Gate
  - CNOT gate
- What is Qiskit?

- Demo: Hello World
- Quantum Algorithms
    - Deutsch-Jozsa
    - Grover’s
    - Shor’s

The educational materials were introduced as part of the existing course CSC (0145) 272 - Principles Of Programming Languages at Adelphi University taught by Professor Sukun Li. It was assumed that the students have taken the prerequisite courses: CSC (0145) 270 - Survey of Programming Languages, CSC (0145) 175 - Intermediate Computer Programming, and CSC (0145) 171 - Intro to Comp Programming (2023-24 University Bulletin, 2023).

The sample population was a class of twenty-six students and the study lasted over two class sessions, one on Tuesday and the other on Thursday of the same week, taking one hour and fifteen minutes each. The lecture was presented in the first session and data collection from this session consisted of qualitative insights of student engagement and interactions with students during class time. The lab packet was presented in the second session. Data collection from this session consisted of an assessment of the artifacts produced by students during this lab session and observation of student engagement with the material during the lab session.

Fourteen lab packet submissions were collected during the study due to them being classified as extra credit and only two responses were collected from the feedback survey. The sample size was too small to make any overarching claims that represent a diverse sample of undergraduate computer science students. However, the study provided valuable insights that could be used to develop an in-depth curriculum and specifically demonstrated high engagement with the topic in Adelphi's computer science community. Moreover, further plans for the development of my research could include enlarging the sample size that this introductory lecture and lab session gets presented to.

Design of Lab Packet

The overarching objectives of the lab packet were to:

- Setup environment
- Execute Quantum Hello World
- Analyze Deutsch-Jozsa Algorithm

The selected development environment for the lab was crucial in the decision making process. The final selection was to use Qiskit–the free, open-source quantum software development kit provided by IBM–on a Google Colab environment. This choice was made to ensure easy access to a Python notebook so that students could practice executing individual cells and familiarize themselves with the results of each line.

The selection of the algorithms that would be included as lab exercises was also crucial in the decision making process. The students were introduced to four algorithms in the lecture: Quantum Hello World, Deutsch-Jozsa, Shor's, and Grover's. The Quantum Hello World algorithm was chosen as an introduction to writing quantum algorithms. This algorithm was simple enough to recreate in the span of a lab session, yet it displayed the crucial principles of quantum computing that the lecture discussed. Although the second algorithm, Deutsch-Jozsa, did not have many real-life applications, it fulfilled its purpose by being another simple algorithm that demonstrated quantum principles. The other two algorithms, Shor's and Grover's, were too complex to introduce in two class sessions, although their real-life applications matched the types of problems that computer science students are used to encountering.

The structure and design of the lab packet was influenced by Bloom's taxonomy, a pedagogical framework that contains a hierarchy of learning objectives that educators aim to meet when designing assessments (Adams, 2015). The hierarchy consists of the following levels: Create, Evaluate, Analyze, Apply, Understand, and Remember (Adams, 2015). The sequence of levels ranges from the highest to the lowest order of critical thinking. In addition to a section that aided in setting up a coding environment for quantum algorithms, the lab packet comprised two parts: “Implement Quantum Hello World” and “Understanding the Deutsch-Jozsa Algorithm.”

The premise of the first section, “Implement Quantum Hello World” was to recreate the provided code in the student’s own coding environment. This satisfied the highest order of critical thinking, “Create.” Each code cell that the students were asked to recreate was followed with a question to further assess their critical thinking skills. The question, “What quantum property is associated with the application of a Hadamard gate on a qubit?” assessed students’ ability of recalling concepts covered in the lecture as we had discussed the Hadamard gate in class. Questions like “In this quantum circuit, the control qubit is q0 and the target qubit is q1. What are the possible measurement outcomes of this CNOT gate?” and “Explain the role of the classical bits in this quantum circuit.” and “What does the histogram plot represent? Interpret the results of the plot. Does it match what you expected the outcomes to be?” satisfied the “Understand,” “Apply,” and “Analyze” criteria. Students were encouraged to use the concepts taught in the lecture and to apply it to produce answers to these questions. So for the first question, students had to demonstrate an understanding of the CNOT gate in order to predict the measurement outcomes. For the second question, the students were asked to observe the circuit that resulted from executing the cell and determine the role of the classical bits. This question not only relied on the student's ability to interpret the circuit visualization but also to demonstrate understanding of the concept of qubit measurement which was discussed in the lecture. The third question required the students to interpret the findings from a histogram that was being plotted that kept count of the measurement outcomes. They were required to apply their understanding of the concept of measurement to answer this question. Essentially, the structure and order of the questions in the lab packet were intentional and they guided the students from going though the different stages of critical thinking from the bottom-up. Once the foundation was set for the syntax of the Qiskit library, the second section of the lab, "Understanding the Deutsch-Jozsa Algorithm" was introduced.

Research Findings

An overarching observation made was that student interest and engagement with the topic of quantum computing as it related to computer science was high. This can be determined from the interactions with the students during the two class sessions. In the lecture session, students demonstrated engagement by answering guiding questions asked during the presentation. Some guiding questions included, “What have you heard about quantum computing before this class?” and “What is the von Neumann architecture?” For the first question, students were able to give answers with a degree of comprehensiveness, offering a useful contrast for evaluating their understanding after the lecture. Generally, they answered that they had heard of the term but did not know how the technology worked. This set the scene for the remainder of the lecture where they would be gaining that knowledge. The second question was used to connect the concepts learned by students about organizing classical computer components earlier in the semester with the manner in which quantum computers organize their components

In the lab session, students demonstrated high levels of understanding of and engagement with the material in one-on-one interactions. They asked clarification questions and demonstrated understanding of the concepts.

Lab Packet Findings:

- 26 total students in the class
- 14 students uploaded submissions to Moodle, the platform that students use to turn in their homework assignments
- Out of those 14 students, 5 students only submitted their code (in either a .py or .ipynb file) and didn't submit a lab packet.
- 9 students submitted lab packets.
    - 100% of the students who submitted their lab packet were able to correctly identify that the Hadamard gate demonstrated the quantum property of superposition.
        - This demonstrates that they were able to remember a key concept introduced in the lecture.
    - 78% of students were able to apply their knowledge from the lecture to deduce that the role of the classical bits was to store the results after measuring the qubits.
        - The concept of measurement is vital to understanding superposition. Once a qubit that's in superposition gets measured, it collapses to a classical state. Being able to deduce the role of classical bits by looking at the code "**circuit.measure(qr, cr)**" and looking at the circuit diagram that

        resulted from this code cell demonstrates the students' ability to apply lecture concepts to executed code.

- 78% of students were able to correctly interpret the histogram that recorded the measurement outcomes of the qubits after they had Hadamard gates and CNOT gates applied to them.
    - This shows that the lecture successfully met the objective of educating students on quantum gates such as the Hadamard and CNOT gates.
- 45% of students were able to correctly explain the function of the oracle in the Deutsch-Jozsa algorithm. The majority of their responses reached a consensus that the constant and balanced oracle was able to determine the nature of the input n-bit string.
    - Some students were able to retain the concept of the oracle as it was taught in the lecture and some were not.
- 56% of the students were able to translate the pseudocode provided into executable code that integrated with Qiskit terminology.
    - Part of the goal of the lab packet was to allow students to run quantum algorithms in their own environments. Being able to correctly translate the steps into code proves that students were able to use their prerequisite knowledge of

        writing code for classical algorithms and apply it by creating their own quantum code.

- 89% of students demonstrated the ability to read the circuit diagram and correctly identify the main difference between the constant and balanced oracle. The general consensus was that the main difference between the oracles was that CNOT gates were applied in the balanced oracle but not the constant oracle.
    - This demonstrated students’ ability to read quantum circuit diagrams and correctly interpret them.

Interpretation

These research findings suggest high student interest and engagement with the topic of quantum computing specifically as it relates to computer science. During the lecture session, student responses to guiding questions created a baseline understanding of quantum computing that was furthered throughout the rest of the lecture.

Students performed exceptionally well in the first section “Quantum Hello World,” demonstrating a strong grasp of key quantum properties like superposition and entanglement; key quantum gates like the Hadamard and CNOT gate; and applying these concepts to a simple and reproducible quantum algorithm. However, student responses were varied in the “Understanding the Deutsch-Jozsa Algorithm” section, indicating a need for further clarification on the concepts of oracles and the Deutsch-Jozsa problem.

Overall, the lecture and lab effectively engaged students and provided a solid understanding of the fundamental concepts of quantum computing. However, further improvements can be made to reinforce certain concepts such as the Deutsch-Jozsa algorithm.

Future Considerations

Given the data, it is safe to assert that introducing quantum computing principles to computer science students can enable them to begin implementing quantum algorithms as early as Day 2. All students were able to successfully recreate the Quantum Hello World algorithm. Students demonstrated a need for further clarification for the Deutsch-Jozsa algorithm.

Although the student responses gave meaningful insights, they were limited in number. Obtaining data from the entire class of twenty-six students would have been beneficial, especially if the lab packet were assigned as a graded assignment rather than solely offered as extra credit.

Next steps could include designing a semester-long course on applied quantum computing using the same frameworks as the lecture in my research. The lecture created for the purpose of my research could be included into this syllabus.

Conclusion

In conclusion, my research sought to address the question: What types of quantum computing problems can undergraduate computer science students solve? The experiment consisted of a lecture that would effectively introduce quantum computing to computer science students. Additionally, I aimed to test how well students can apply those concepts in solving real-world problems using quantum algorithms. Through a thorough literature review, I identified the core concepts that needed to be addressed to build a foundation for students to start coding and designed a lecture plan that aligned with those core concepts. The key objectives of the lecture was to introduce:

- What is Quantum Computing?
    - Superposition
    - Entanglement
    - Quantum parallelism
    - Oracle - Black box mechanism
- Examples of Quantum Gates
    - Hadamard Gate
    - CNOT gate
- What is Qiskit?

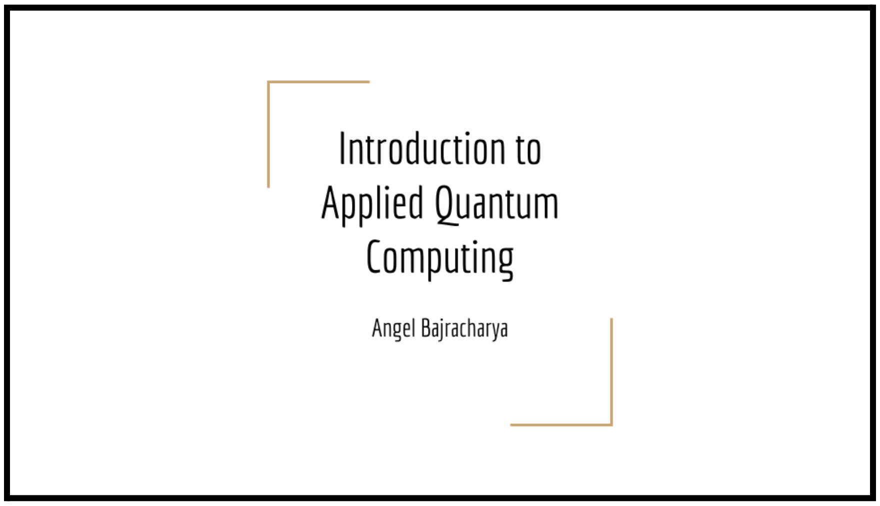
- Demo: Hello World
- Quantum Algorithms
    - Deutsch-Jozsa algorithm
    - Grover’s algorithm
    - Shor’s algorithm

Initially, I hypothesized Deutsch Jozsa, Shor's, and Grover's algorithms as potential solutions to my original research problem. However, the research findings revealed that implementing either of these introductory algorithms was not feasible within the scope of the study. Nonetheless, the design of the lecture and lab packet helped foster understanding of and interest in the revolutionary paradigm of quantum computing. The findings of this research underscore the importance of a structured approach to teaching quantum computing concepts. My research contributes to the growing body of literature on quantum computing education by offering an approach that emphasizes hands-on learning and practical applications. Furthermore, leveraging Qiskit as a framework for simulating quantum behavior proved to be successful, enabling students to conduct quantum computations via the cloud. Future considerations should include increasing the sample size of students and increasing the duration of the study in order to gain deeper insights.

Appendices

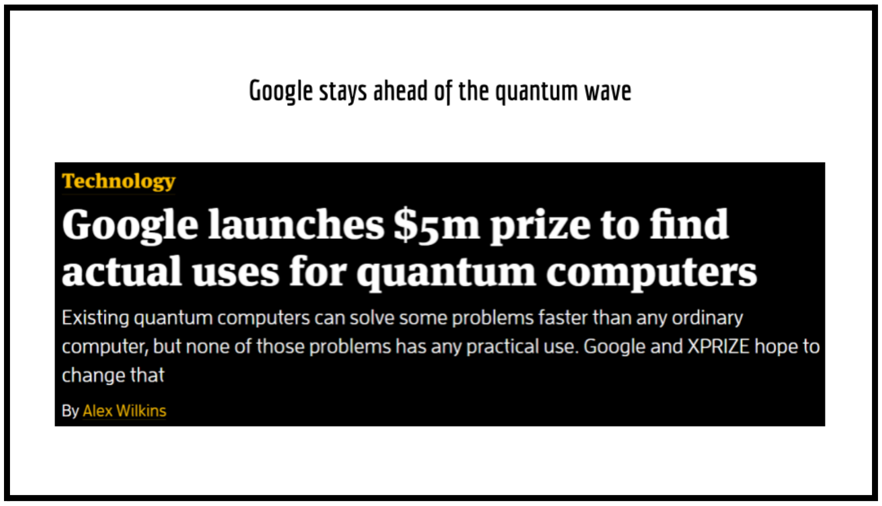
Appendix A: Lecture Slides

Lecture Slides



Introduction to Applied Quantum Computing

Angel Bajracharya



Google stays ahead of the quantum wave

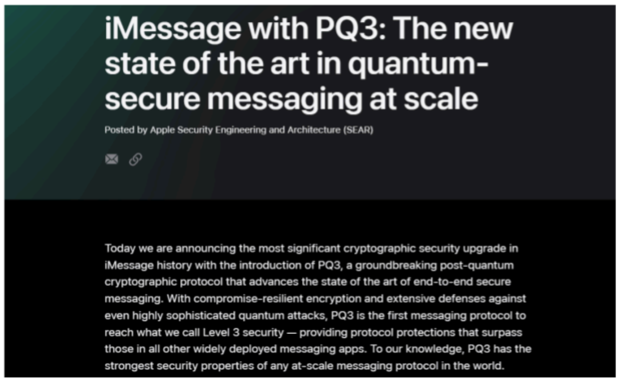
Technology

Google launches $5m prize to find actual uses for quantum computers

Existing quantum computers can solve some problems faster than any ordinary computer, but none of those problems has any practical use. Google and XPRIZE hope to change that

By Alex Wilkins

Apple stays ahead of the quantum wave

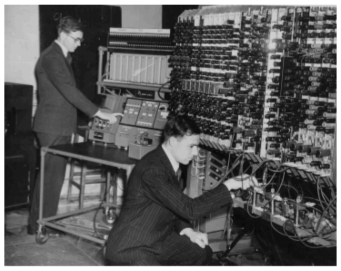

iMessage with PQ3: The new state of the art in quantum-secure messaging at scale

Posted by Apple Security Engineering and Architecture (SEAR)

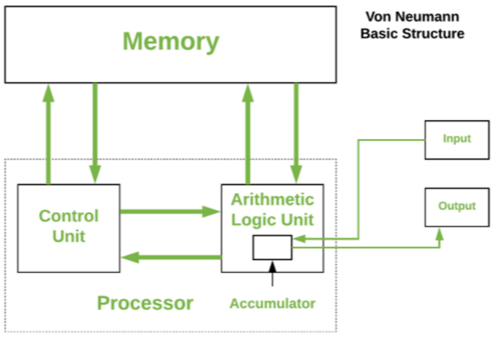
Today we are announcing the most significant cryptographic security upgrade in iMessage history with the introduction of PQ3, a groundbreaking post-quantum cryptographic protocol that advances the state of the art of end-to-end secure messaging. With compromise-resilient encryption and extensive defenses against even highly sophisticated quantum attacks, PQ3 is the first messaging protocol to reach what we call Level 3 security — providing protocol protections that surpass those in all other widely deployed messaging apps. To our knowledge, PQ3 has the strongest security properties of any at-scale messaging protocol in the world.

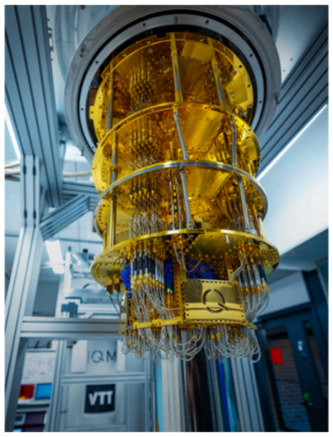
Objectives

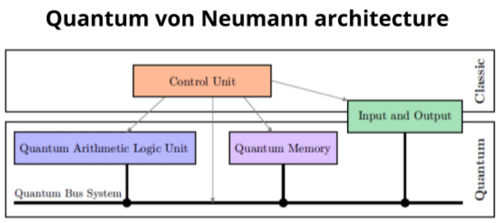
- What is Quantum Computing?
- Examples of Quantum Gates
  - Hadamard Gate
  - CNOT gate
- What is Qiskit?
- Demo: Hello World
- Quantum Algorithms
  - Deutsch-Jozsa
  - Grover's
  - Shor's


Computers before

Computers now



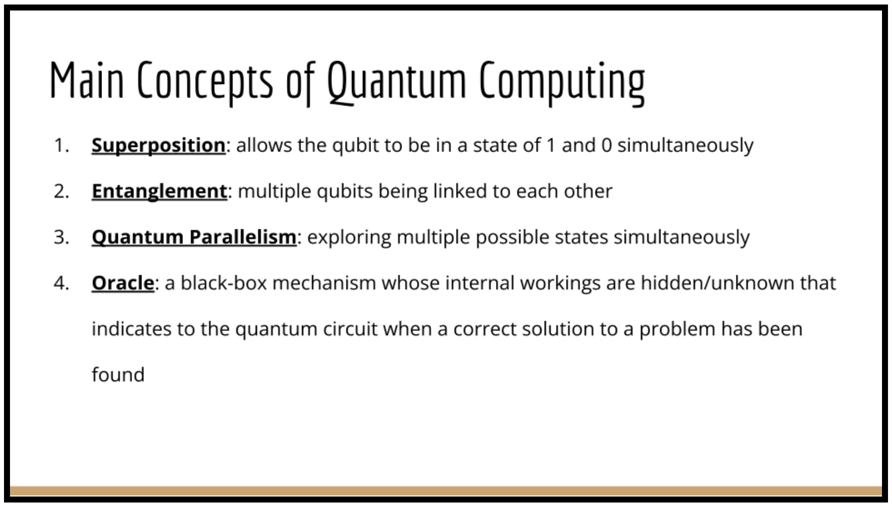




What is Quantum Computing?

Classical computing: stores data in the form of bits that can be represented by 1 or 0 ie. true or false.

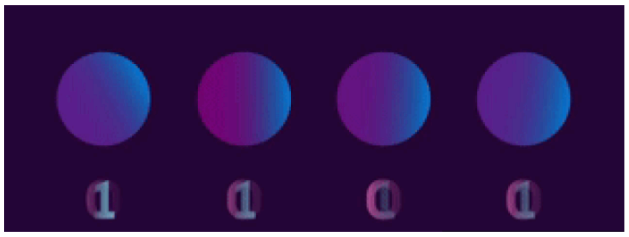
Quantum computing: stores data in the form of quantum bits, aka ***qubits***, and harnesses principles of quantum mechanics (such as superposition and entanglement) to cause computational speedups

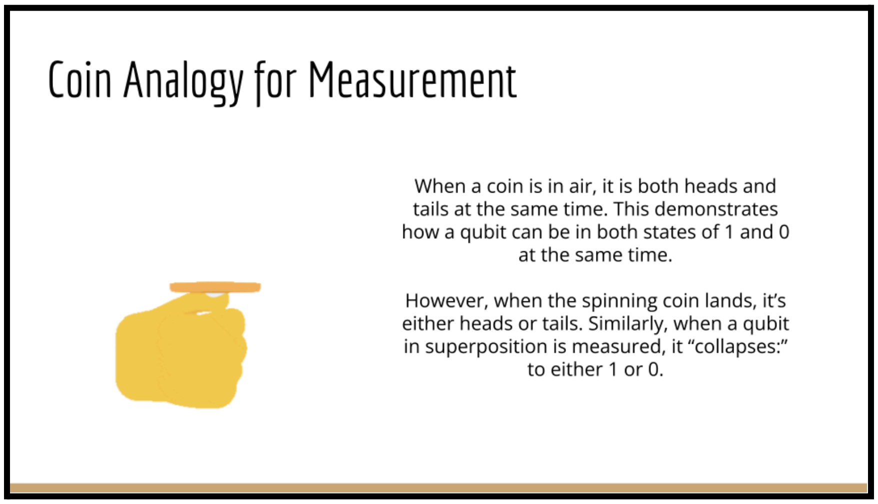


Main Concepts of Quantum Computing

1. **Superposition**: allows the qubit to be in a state of 1 and 0 simultaneously
2. **Entanglement**: multiple qubits being linked to each other
3. **Quantum Parallelism**: exploring multiple possible states simultaneously
4. **Oracle**: a black-box mechanism whose internal workings are hidden/unknown that indicates to the quantum circuit when a correct solution to a problem has been found

Superposition

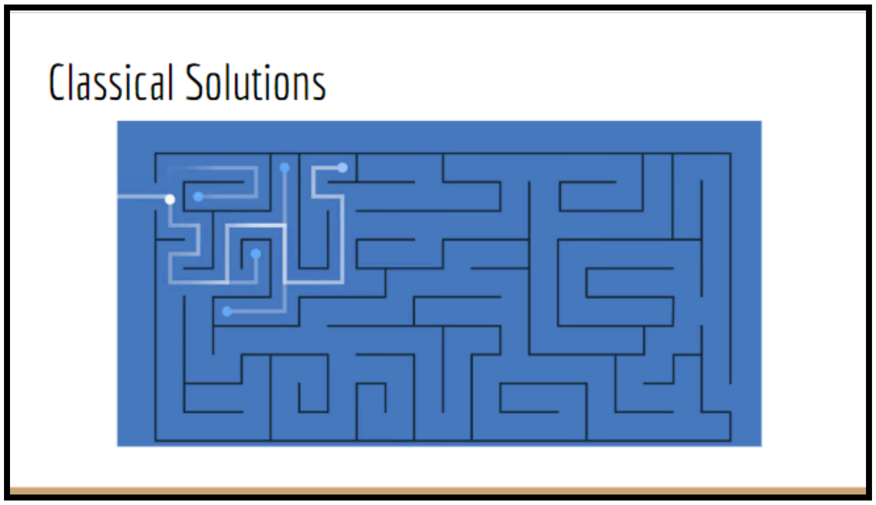




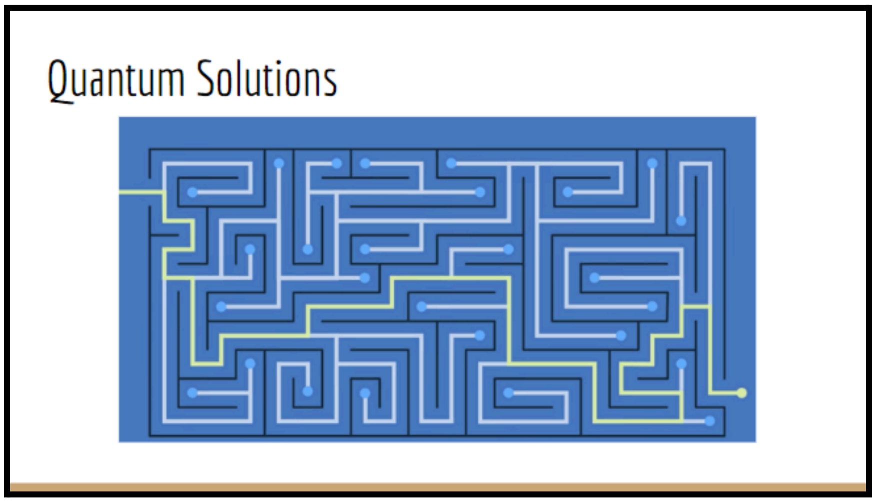
Coin Analogy for Measurement

When a coin is in air, it is both heads and tails at the same time. This demonstrates how a qubit can be in both states of 1 and 0 at the same time.

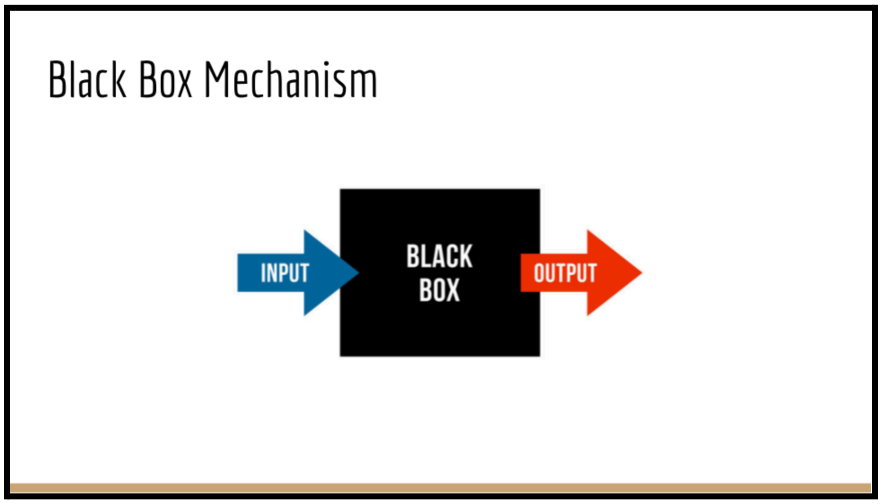
However, when the spinning coin lands, it's either heads or tails. Similarly, when a qubit in superposition is measured, it "collapses:" to either 1 or 0.



Classical Solutions

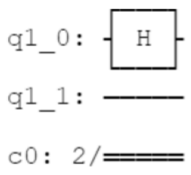


Quantum Solutions



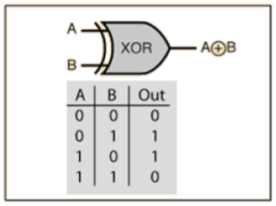
Black Box Mechanism

Ex: Hadamard Gate



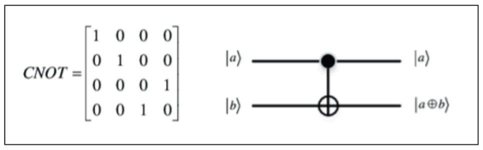
One of the most important quantum gates is the Hadamard gate. It allows the qubit to go from a classical state to a state of superposition.

Ex: CNOT Gate

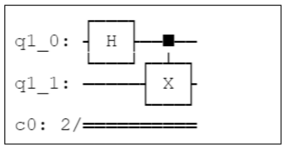


A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

Classical example



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

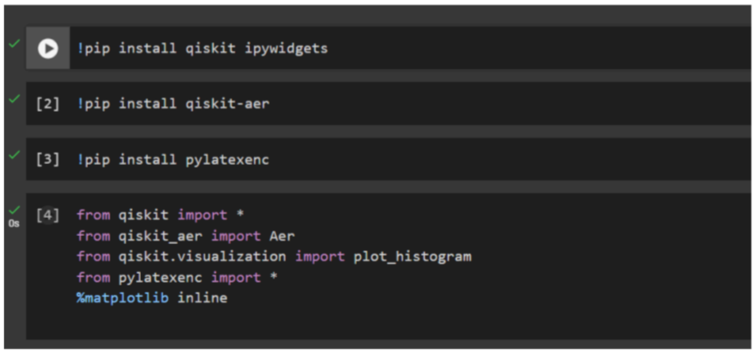


Quantum example

What is Qiskit?

- Qiskit it an open-source software development kit (SDK) that allows us to simulate quantum behavior without quantum computers.
- Easy to implement because it is Python-based
- You can actually run it on one of IBM’s quantum computers!

Hello World Demo



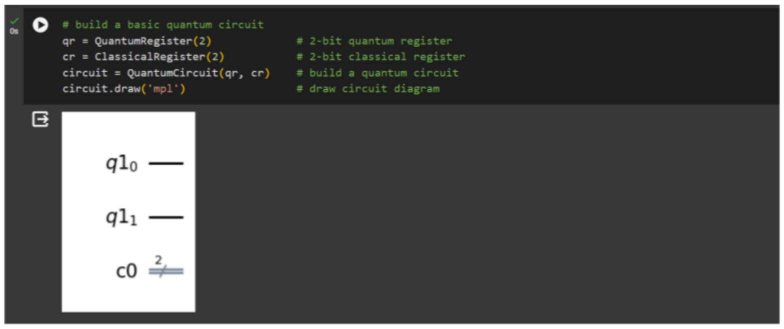
```
[1] !pip install qiskit ipywidgets
```

```
[2] !pip install qiskit-aer
```

```
[3] !pip install pylatexenc
```

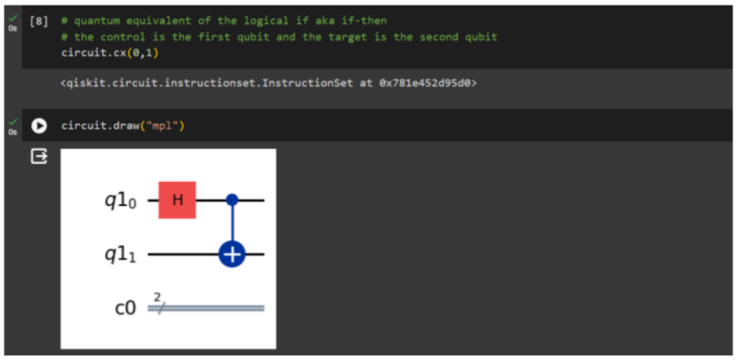
```
[4] from qiskit import *
    from qiskit_aer import Aer
    from qiskit.visualization import plot_histogram
    from pylatexenc import *
    %matplotlib inline
```

Hello World Demo cont.



```
# build a basic quantum circuit
qr = QuantumRegister(2)             # 2-bit quantum register
cr = ClassicalRegister(2)           # 2-bit classical register
circuit = QuantumCircuit(qr, cr)    # build a quantum circuit
circuit.draw('mpl')                 # draw circuit diagram
```

Hello World Demo cont.

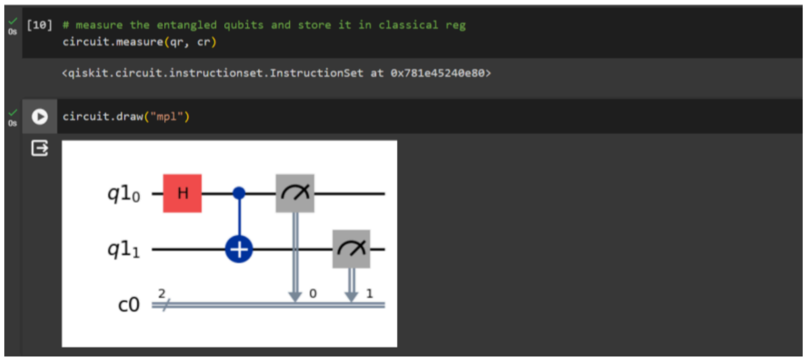


```
[8] # quantum equivalent of the logical if aka if-then
    # the control is the first qubit and the target is the second qubit
    circuit.cx(0,1)
```

```
<qiskit.circuit.instructionset.InstructionSet at 0x781e452d95d0>
```

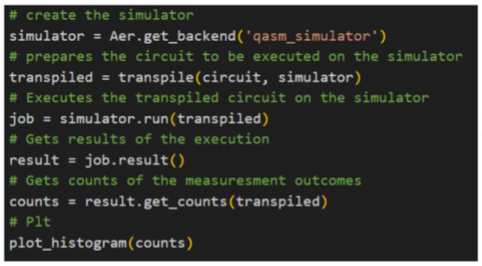
```
circuit.draw("mpl")
```

Hello World Demo cont.

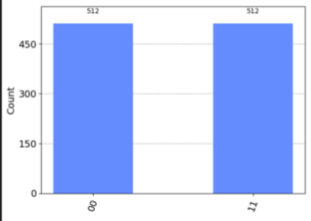


```
[10] # measure the entangled qubits and store it in classical reg
     circuit.measure(qr, cr)

     <qiskit.circuit.instructionset.InstructionSet at 0x781e45240e80>

     circuit.draw("mpl")
```

Hello World Demo cont.



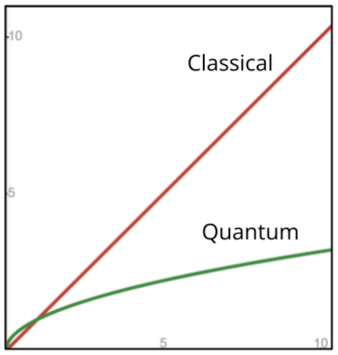
```
# create the simulator
simulator = Aer.get_backend('qasm_simulator')
# prepares the circuit to be executed on the simulator
transpiled = transpile(circuit, simulator)
# Executes the transpiled circuit on the simulator
job = simulator.run(transpiled)
# Gets results of the execution
result = job.result()
# Gets counts of the measurement outcomes
counts = result.get_counts(transpiled)
# Plt
plot_histogram(counts)
```



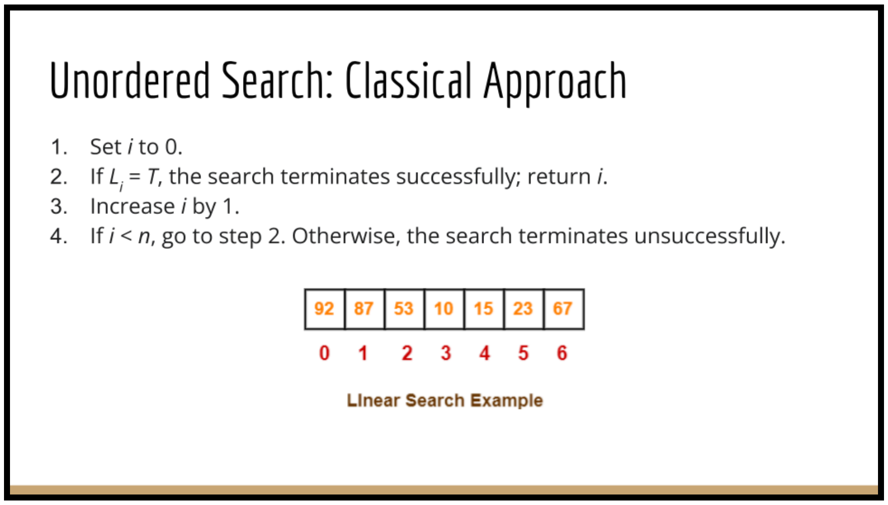
The qasm simulator will run the quantum circuit multiple times, also known as "shots," and count each measurement outcome. Default number of shots is 1024.

Grover's Algorithm

- Developed by computer scientist Lov Grover in 1996
- This quantum algorithm allows for more efficient searches in a database compared to classical search algorithms
- Difference in Time Complexity:
  - Classical Solution: O(N) where N is the size of the search space
  - Quantum Solution: O(√N)

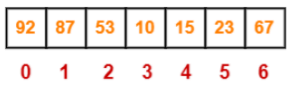


Difference in Time Complexity

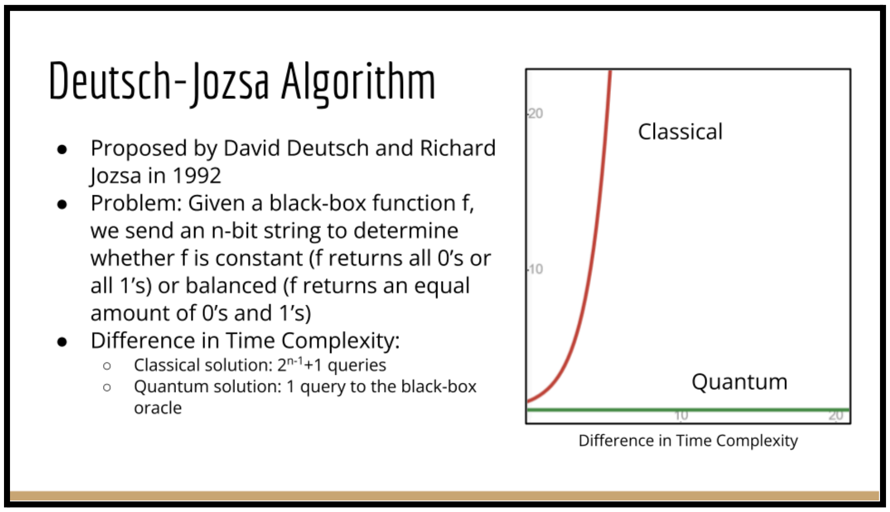


Unordered Search: Classical Approach

1. Set *i* to 0.
2. If $L_i = T$, the search terminates successfully; return *i*.
3. Increase *i* by 1.
4. If $i < n$, go to step 2. Otherwise, the search terminates unsuccessfully.

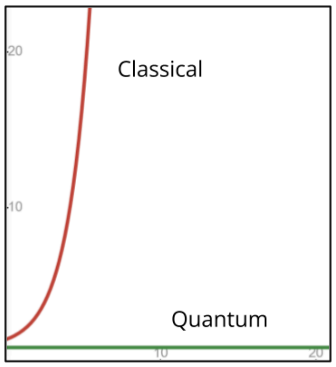


Linear Search Example

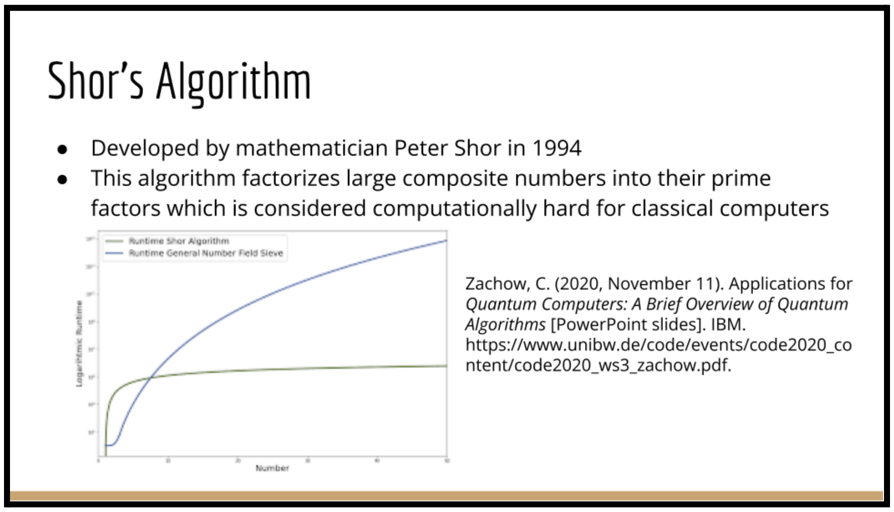


Deutsch-Jozsa Algorithm

- Proposed by David Deutsch and Richard Jozsa in 1992
- Problem: Given a black-box function f, we send an n-bit string to determine whether f is constant (f returns all 0's or all 1's) or balanced (f returns an equal amount of 0's and 1's)
- Difference in Time Complexity:
  - Classical solution: $2^{n-1}+1$ queries
  - Quantum solution: 1 query to the black-box oracle

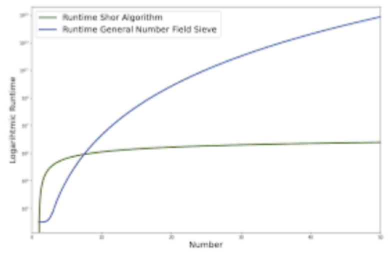


Difference in Time Complexity

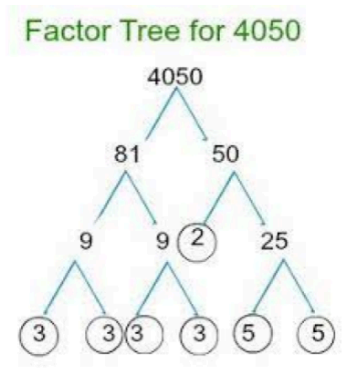


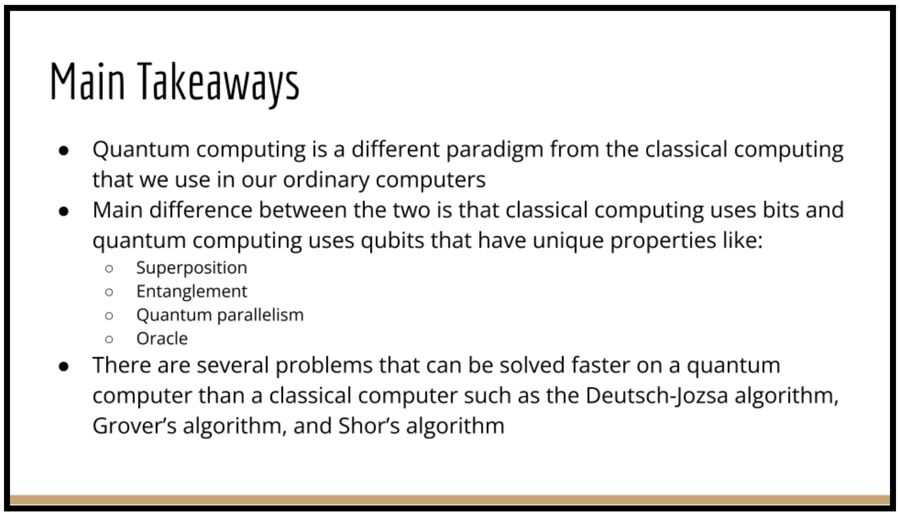
Shor's Algorithm

- Developed by mathematician Peter Shor in 1994
- This algorithm factorizes large composite numbers into their prime factors which is considered computationally hard for classical computers



Zachow, C. (2020, November 11). Applications for *Quantum Computers: A Brief Overview of Quantum Algorithms* [PowerPoint slides]. IBM. https://www.unibw.de/code/events/code2020_content/code2020_ws3_zachow.pdf.





Main Takeaways

- Quantum computing is a different paradigm from the classical computing that we use in our ordinary computers
- Main difference between the two is that classical computing uses bits and quantum computing uses qubits that have unique properties like:
    - Superposition
    - Entanglement
    - Quantum parallelism
    - Oracle
- There are several problems that can be solved faster on a quantum computer than a classical computer such as the Deutsch-Jozsa algorithm, Grover's algorithm, and Shor's algorithm

Appendix B: Recording and transcript of lecture

https://adelphiuniversity.zoom.us/rec/share/jBbkkfHsYbCvFH3od4M3ca87YdXj-gai6sNa5n89jI3neQ8ZtssF6pIDWx9dW4O8.xSmAtcJIAqxABJPV

Passcode: Y*E=7$%F

Appendix C: Source Code

https://github.com/abajr516/Quantum-Computing

Contains the implementation of the Quantum Hello World algorithm and the Deutsch-Jozsa Algorithm

Appendix D: Lab Packet

Lab Packet

(see next page)

Name: ______________________

Objectives

In this lab, we will delve into the world of quantum computing, exploring its principles, algorithms, and potential applications.

Main Goals:

1. Setup environment
2. Execute Quantum Hello World
3. Analyze Deutsch-Jozsa Algorithm

Setup Environment

In this lab, we will use Google Colab to code Python and run quantum algorithms. We'll install Qiskit directly into the Colab notebook to facilitate this process.

1. Open your Google Drive, click "New," select "More," then "Google Colaboratory" to open a new notebook.
2. Create a new code cell and run the following command to install Qiskit into the Colab environment.

```
!pip install qiskit ipywidgets
```

3. Run the following command to install the Qiskit Aer package which will be used to simulate the quantum circuit.

```
!pip install qiskit-aer
```

4. This is another dependency that needs to get installed to draw the circuit.

```
!pip install pylatexenc
```

5. Run the following lines of code to install the required packages.

```
from qiskit import *
from qiskit_aer import Aer
from qiskit.visualization import plot_histogram
%matplotlib inline
```

You have now successfully installed all the required dependencies for the implementation of quantum algorithms.

Implement Quantum Hello World

Run each box of code in *separate* code cells and in the order that they are in:

```
qr = QuantumRegister(2)              # 2-bit quantum register
cr = ClassicalRegister(2)            # 2-bit classical register
circuit = QuantumCircuit(qr, cr)     # build a quantum circuit
circuit.draw('mpl')                  # draw circuit diagram
```

Note: The "mpl" parameter within the draw() function is used to render the circuit using a uniform standard facilitated by Matplotlib. Its primary purpose is to enhance the readability of the circuit diagram.

```
# apply a Hadamard gate to the first qubit
circuit.h(0)
circuit.draw('mpl')
```

What quantum property is associated with the application of a Hadamard gate on a qubit?

```
# apply a CNOT gate to the 2 qubits
circuit.cx(0,1)
circuit.draw('mpl')
```

In this quantum circuit, the control qubit is q0 and the target qubit is q1. What are the possible measurement outcomes of this CNOT gate?

```
circuit.measure(qr, cr)
circuit.draw('mpl')
```

Explain the role of the classical bits in this quantum circuit.

```
# create the simulator
simulator = Aer.get_backend('qasm_simulator')
# prepares the circuit to be executed on the simulator
transpiled = transpile(circuit, simulator)
# executes the transpiled circuit on the simulator
job = simulator.run(transpiled)
# gets results of the execution
result = job.result()
# gets counts of the measurement outcomes
counts = result.get_counts(transpiled)
# plots histogram with counts of the measurement outcomes
plot_histogram(counts)
```

What does the histogram plot represent? Interpret the results of the plot. Does it match what you expected the outcomes to be?

STOP: Before you continue to the rest of the packet, go to File in your Google Colab, download as .py or .ipynb and submit in Moodle.

Understanding the Deutsch-Jozsa Algorithm

*The **Deutsch-Jozsa problem** is figuring out whether a function f is constant (outputting all 0's or all 1's) or balanced (outputting an equal amount of 1's and 0's) by passing an n-bit string and analyzing the outputs.*

You are not required to run this program

```
# Define the Deutsch-Jozsa algorithm function
# Parameters: oracle (constant, balanced) and n (number of bits in str)
# Return quantum circuit
def deutsch_josza_algorithm(oracle, n):
  # Create a quantum circuit with n+1 qubits and n classical bits
  dj_circuit = QuantumCircuit(n+1, n)
  # Apply X gate to the last qubit
  dj_circuit.x(n)
  # Apply Hadamard gate to the last qubit
  dj_circuit.h(n)
  dj_circuit.barrier()
  # Apply Hadamard gates to all the bits
  dj_circuit.h(range(n))
  dj_circuit.barrier()
  # Apply the oracle
  dj_circuit.compose(oracle, range(n+1), inplace=True)
  dj_circuit.barrier()
  # Apply Hadamard  gates to all qubits AGAIN
  dj_circuit.h(range(n))
  dj_circuit.barrier()
  # Measure the first n bits
  dj_circuit.measure(range(n), range(n))
  return dj_circuit

# Define the balanced oracle function
def balanced_oracle(n):
  oracle = QuantumCircuit(n+1)
  for qubit in range(n):
      oracle.cx(qubit, n)
  return oracle

# Define the constant oracle function
def constant_oracle(n):
  oracle = QuantumCircuit(n+1)
  return oracle
```

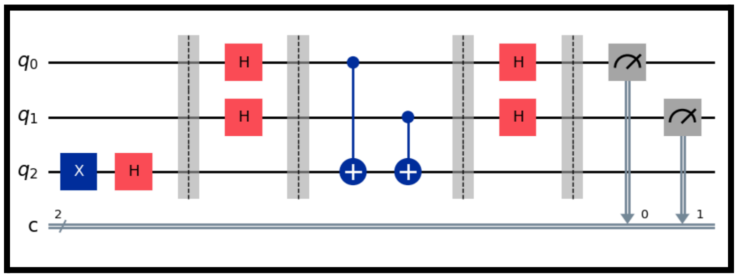
What is the function of an oracle in this algorithm? Explain the difference between the balanced and constant oracle.

You have been given the pre-defined functions that you will need for the Deutsch-Jozsa algorithm. How would you call these functions to answer the Deutsch-Jozsa problem for a 2-qubit string?

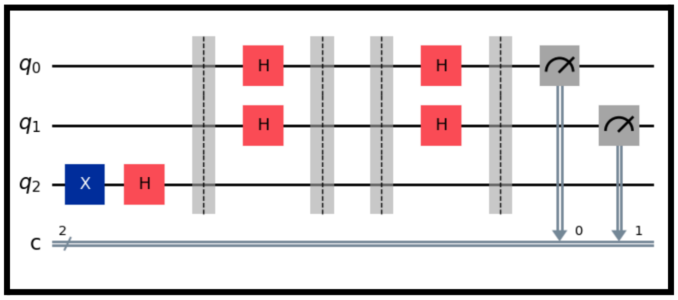
Write the code that will allow you to do the following:

1. Create a variable, `n`, that will equal to 2 to represent the 2-bit input string.
2. Create a variable, `oracle,` that will either be a balanced or constant oracle using `balanced_oracle(n)` or `constant_oracle(n)` respectively.
3. Create a Deutsch-Jozsa circuit using the `deutsch_josza_algorithm(oracle, n)` where the two arguments will be the oracle and n variable you created.
4. Draw the circuit that you created using the `draw()` function.

Depending on which oracle you decide to invoke, these are what your resulting circuits should look like:



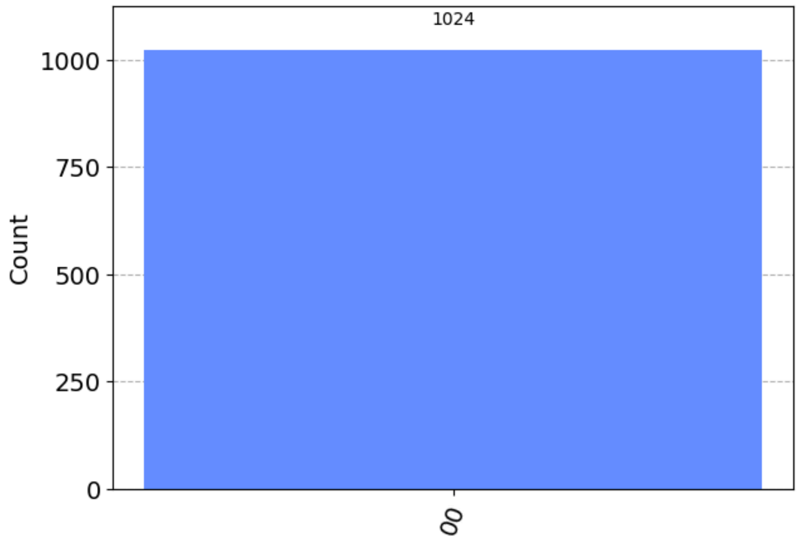
Balanced oracle



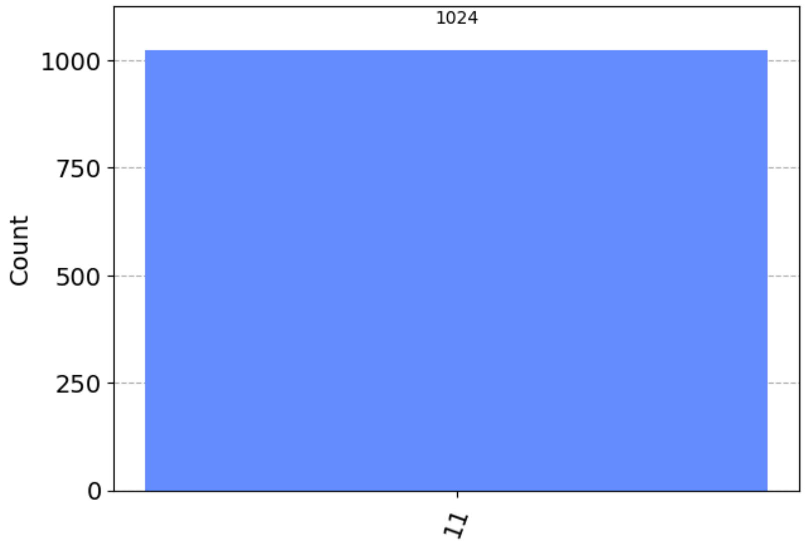
Constant oracle

How does the circuit diagram of a balanced oracle differ from that of a constant oracle in the Deutsch-Jozsa algorithm?

Depending on which oracle you decide to invoke, these are what the measurement outcomes from a quantum simulator should look like:



Constant Oracle



Balanced Oracle

How do the measurement outcomes of a balanced oracle differ from that of a constant oracle in the Deutsch-Jozsa algorithm? Were these the outcomes that we expected?

Appendix E: Feedback Survey

Interest in the New Topic:
On a scale of 1 to 5, please rate your level of interest in the newly introduced topic:

1. Not interested at all
2. Slightly interested
3. Neutral
4. Interested
5. Very interested

Understanding of the Topic:
How well do you feel you understood the new topic?

1. Very well
2. Moderately well
3. Somewhat well
4. Not very well
5. Not at all

Engagement with the Topic:
Did you find the topic engaging and thought-provoking?

1. Yes, very engaging
2. Somewhat engaging
3. Neutral
4. Not very engaging
5. Not at all engaging

Preferred Method of Instruction:
How do you prefer to learn about quantum computing? Please select all that apply:

1. Lecture-style presentations
2. Hands-on activities or labs
3. Multimedia resources (videos, animations, GIFs, images, etc.)
4. Other (please specify):

Suggestions for Improvement:

References

Adams N. E. (2015, July). *Bloom's taxonomy of cognitive learning objectives.* Journal of the Medical Library Association: JMLA, 103(3), 152–153. https://doi.org/10.3163/1536-5050.103.3.010.

Adelphi (2023, March). 2023-24 University Bulletin [Archived Catalog]. https://catalog.adelphi.edu/index.php.

Amellal, H., Meslouhi, A., & El Allati, A. (2018, October 10). *Effectiveness of quantum algorithms on classical computing complexities*. In Proceedings of the 3rd International Conference on Smart City Applications (SCA '18). Association for Computing Machinery, New York, NY, USA, Article 34, 1–3. https://doi-org.adelphi.idm.oclc.org/10.1145/3286606.3286811.

Apple Security Engineering and Architecture (SEAR). (2024, February 21). *iMessage with PQ3: The new state of the art in quantum-secure messaging at scale*. Apple Security Engineering and Architecture Blog. https://security.apple.com/blog/imessage-pq3/.

Bernhardt, C. (2019). Quantum Computing for Everyone. The MIT Press.

Chen, L. (2023, April 2). *A Brief History of Quantum Computing*. Quantumpedia-The Quantum Encyclopedia. https://quantumpedia.uk/a-brief-history-of-quantum-computing-e0bbd05893d0.

Deutsch, D., & Jozsa, R. (1992, December 8). *Rapid Solution of Problems by*

Quantum Computation. Proceedings: Mathematical and Physical Sciences, 439(1907), 553–558. http://www.jstor.org/stable/52182.

Deutsch, I. H. (2020, November 13). *Harnessing the Power of the Second Quantum Revolution*. American Physical Society. https://doi.org/10.1103/PRXQuantum.1.020101.

Gambetta, J. (2023, December 4). *The hardware and software for the era of quantum utility is here*. IBM. https://www.ibm.com/quantum/blog/quantum-roadmap-2033.

Godfrey, M. D. (1992). First Draft of a Report on the EDVAC by John von Neumann. https://web.archive.org/web/20130314123032/http://qss.stanford.edu/~godfrey/vonNeumann/vnedvac.pdf.

Google (n.d.). Cirq. https://quantumai.google/cirq.

IBM Research. (2018). IBM quantum computer [Photograph]. Flickr. https://www.flickr.com/photos/ibm_research_zurich/40786969122.

Johansson, N., Larsson, J. Å. (2017, August 12). *Efficient classical simulation of the Deutsch–Jozsa and Simon's algorithms.* Quantum Inf Process 16, 233. https://doi.org/10.1007/s11128-017-1679-7.

Johansson, N. & Larsson, J. A. (2019, August 15). *Quantum Simulation Logic, Oracles, and the Quantum Advantage*. Entropy, 21, 8, 800. https://doi.org/10.3390/e21080800.

Kapooht (2013). von Neumann Architecture. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Von_Neumann_Architecture.svg?uselang=fr.

Kaye, P., Laflamme, R., & Mosca, M. (2007). An introduction to quantum computing. OUP Oxford.

Ladopoulos, A. T. (2023, September, 5). *Quantum Leap: Revolutionizing Food Production with Quantum Computers*. Linkedin. https://www.linkedin.com/pulse/quantum-leap-revolutionizing-food-production-athanasios-t-ladopoulos/

Liu, J., & Franklin, D. (2023, March 3). *Introduction to Quantum Computing for Everyone: Experience Report*. In Proceedings of the 54th ACM Technical Symposium on Computer Science Education (Vol. 1, SIGCSE 2023, pp. 7). March 15–18, 2023, Toronto, ON, Canada. ACM, New York, NY, USA. https://doi.org/10.1145/3545945.3569836.

Mandelbaum, R. (2021, May 4). *Five years ago today, we put the first quantum computer on the cloud. Here's how we did it*. IBM. https://www.ibm.com/quantum/blog/quantum-five-years

Mehar, A. (2024, January 25). *Quantum Computing Frameworks: Cirq vs Qiskit.* Linkedin. https://www.linkedin.com/pulse/quantum-computing-frameworks-cirq-vs-qiskit-antematter-hs5sf/.

Mermin, N. D (2007). Quantum Computer Science: An Introduction. Cambridge University Press.

Meyer, J. C., Passante, G., Pollock, S. J., & Wilcox, B. R. (2022). *Today's interdisciplinary quantum information classroom: Themes from a survey of quantum information science instructors.* Physical Review. Physics Education Research, https://www.proquest.com/central/docview/2731423265/CCAE55CA22BF4C89PQ/9?accountid=8204.

Mykhailova, M., & Svore, K. M. (2020, February 26). Teaching Quantum Computing through a Practical Software-drive Approach: Experience Report. ACM Digital Library. https://dl-acm-org.adelphi.idm.oclc.org/doi/10.1145/3328778.3366952.

Nielsen, M. A. & Chuang I. L. (2011). Quantum Computation and Quantum Information. Cambridge University Press.

Parks Associates (2024, January 11). *At CES® 2024, Parks Associates announces new research showing average number of connected devices per US internet household reached 17 in 2023*. Parks Associates. https://www.parksassociates.com/blogs/press-releases/at-ces-2024-parks-associates-announces-new-research-showing-average-number-of-connected-devices-per-us-internet-household-reached-17-in-2023#:~:text=Parks%20

Associates%20today%20announced%20new,reached%2017%20in%20Q3%202023.

Pflitsch, M. (2022, December, 12). *Quantum Biology: How Quantum Computing Can Unlock A New Dimension of Treating Diseases*. Forbes. https://www.forbes.com/sites/forbestechcouncil/2022/12/12/quantum-biology-how-quantum-computing-can-unlock-a-new-dimension-of-treating-diseases/?sh=46cce1ef233a.

Seegerer, S., Michaeli T., & Romeike R. (2021, October 19). *Quantum Computing As a Topic in Computer Science Education*. In The 16th Workshop in Primary and Secondary Computing Education (WiPSCE '21). Association for Computing Machinery, New York, NY, USA, Article 13, 1–6. https://doi-org.adelphi.idm.oclc.org/10.1145/3481312.3481348.

SoniaLopezBravo, Bradben, tedhudek, dphansen, geduardo, & cjgronlund. (2023, June 9).

Quantum Computing History - Azure Quantum. Quantum computing history - Azure Quantum | Microsoft Learn. https://learn.microsoft.com/en-us/azure/quantum/concepts-overview.

SoniaLopezBravo, Bradben, tedhudek, dphansen, geduardo, & cjgronlund. (2024, January 12).

What are Q# and the Azure Quantum Development Kit? https://learn.microsoft.com/en-us/azure/quantum/overview-what-is-qsharp-and-qdk.

Terren in Virginia. (2008). ENIAC [Photograph]. Flickr. https://www.flickr.com/photos/8136496@N05/2196367188.