QUANTUM COMPUTING: AN INTRODUCTION

ARAV BHATTACHARYA

1. Introduction

Quantum computing is a very common buzzword in the modern-day tech landscape, which makes sense because it's a powerful tool that can do everything a classical computer can and more. And while there are many difficulties in implementing quantum computing which have prevented it from going into mainstream use at this point, the mathematical theory behind quantum computing gives us insight into why so many people have predicted that it will be the dominant computing paradigm in the future. Surprisingly, one does not need an in-depth prior knowledge of quantum mechanics to learn this rich theory!

In this paper, we will begin by defining basic aspects of quantum computers and providing a general understanding of each of these aspects. Then, we will prove that quantum computers can be used to simulate deterministic classical computers, concluding with a look at an algorithm that runs faster on quantum computers than on classical computers.

2. Background

This paper assumes that the reader is familiar with linear algebra and the theories of classical computation and complexity. A background in quantum mechanics is not necessary for understanding this paper.

3. The Qubit

The most basic unit in classical computing is the bit. Similarly, the basic unit of quantum computers is called the qubit, a portmanteau of the words quantum and bit. We can think of qubits as mathematical objects used in quantum computing. Although qubits are really physical objects, treating them as just mathematical objects gives us a basis for studying them in sufficient detail without delving into the physical complexities of quantum mechanics.

Definition 3.1. A *qubit* is a basic unit of quantum information: the quantum analogue of the classic binary bit physically realized with a two-state device.

Though qubits have many differences from bits, there are some basic similarities. Bits are in one of two states: either 0 or 1. Qubits also have two so-called *computational basis states*, $|0\rangle$ and $|1\rangle$. But qubits don't just have these two states. In simplest terms, qubits are like unit vectors in a 2-dimensional complex vector space. Thus we have more options for qubits' states: qubits can take many different states that are linear combinations of $|0\rangle$ and $|1\rangle$ with magnitude 1, i.e.

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

where α, $\beta \in \mathbb{C}$.

Date: December 12, 2022.

But what do these states mean? Unlike classical bits, qubits' states cannot directly be measured. When you measure any state of the form $\alpha|0\rangle + \beta|1\rangle$, the state "collapses" down to $|0\rangle$ with probability $|\alpha|^2$ or down to $|1\rangle$ with probability $|\beta|^2$. The state can only collapse down to either $|0\rangle$ or $|1\rangle$, so the probabilities of collapsing to either computational basis state must add to 1. Hence we must have that $|\alpha|^2 + |\beta|^2 = 1$. This equation also makes sense if we think of our qubit as a 2-dimensional vector, since its norm must be equal to 1. To see the geometric intuition for qubit states, visit [blo] (note that the notation on this site is slightly different from what's used in this paper: $|\uparrow\rangle$ is used in place of $|0\rangle$ and $|\downarrow\rangle$ is used in place of $|1\rangle$).

Example. If a qubit takes on either of the computational basis states, it will only collapse down to its state when measured. Ergo, measuring $|1\rangle$ will always give $|1\rangle$ and measuring $|0\rangle$ will always output $|0\rangle$.

Example. A qubit may have the state $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, but you would not be able to tell that the qubit is in this state. When you measure this qubit, the measurement would be $|0\rangle$ with probability $\frac{1}{2}$ or $|1\rangle$ with probability $\frac{1}{2}$.

4. Multiple Qubits

Now, while one qubit already has many interesting properties, quantum computing relies on our ability to work with multiple qubits. So how do we represent multiple qubits? To answer this question, we once again revisit the classical case of multiple bits: n classical bits can take on one of 2^n possible states from the set of n-digit binary strings, which we will represent as

$$\{0,1\}^n = \{0^n, 0^{n-1}1, \ldots, 1^n\}.$$

Similarly, a quantum computer with n qubits can be represented with a 2^n-dimensional complex vector space, where in each state $|\phi\rangle$ our qubits can take on any linear combination of states $|x\rangle$, where $x \in \{0,1\}^n$. In other words, we can represent $|\phi\rangle$ as

$$|\phi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle.$$

Now, similar to the case of a single qubit, the probability that we get $|x\rangle$ when we measure all the qubits in our computer is $|\alpha_x|^2$. It follows that

$$\sum_{x\in\{0,1\}^n} |\alpha_x|^2 = 1,$$

which implies that the vector representing the qubits' states is a unit vector, once again paralleling the case of a single qubit.

5. Quantum Circuits

Just like in classical computing, we have logic gates and circuits in quantum computing. However, the gates involved in quantum computing are slightly different from those involved in classical computing. This is because quantum gates must define a map from one infinite space of states to another, while classical gates simply define a map from finitely many input states to finitely many output states. Now, keeping in mind that we only feed unit vectors as input to quantum gates, we want quantum gates to obey three properties:

(1) Quantum gates should behave linearly on quantum states.

(2) The output of a quantum gate should be a unit vector.
(3) An orthogonal set of states should map to an orthogonal set of states.

The first of these properties follows empirically. The reader should justify in their own mind that properties (2) and (3) are also desirable.

It turns out that we can satisfy all three properties by representing quantum gates with unitary matrices, i.e. matrices U that satisfy

$$U^{\mathbf{H}}U = I = UU^{\mathbf{H}},$$

where $U^{\mathbf{H}}$ is the Hermitian adjoint matrix of U (see [Axl97] for more information on the adjoint operator).

Here are some quantum gates that operate on a single qubit:

Example. The quantum equivalent of the classical `NOT` gate, which operates on a single qubit system, is represented by the 2 by 2 matrix

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

Example. The Hadamard transform is given by

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

The Hadamard gate takes its input halfway to where the `NOT` gate takes it, though

$$I = H^2 \neq X.$$

Example. The phase transforms are given by

$$P(\psi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\psi} \end{bmatrix}.$$

These transforms take a single qubit and rotate the coefficient of $|1\rangle$ by ψ radians in the complex plane.

The last few examples in this section will be of operators on multiple qubits.

Example. To get an operator that flips the k^{th} qubit in a set of n qubits, we can use a controlled-not, or `CNOT`. The `CNOT` operator can be defined by an $2n$ by $2n$ permutation matrix where only the $2k^{\text{th}}$ and $(2k+1)^{\text{th}}$ columns are swapped from the n by n identity matrix.

Example. A controlled-U operator on n qubits applies a quantum gate U to the k^{th} qubit and does nothing to the other qubits. The reader can verify that this operator is unitary.

Example. The quantum Fourier transform (QFT), the quantum analogue of the classical discrete Fourier transform, is easiest to define by its action on N orthonormal basis states $|0\rangle, \ldots, |N-1\rangle$. Letting j be free in $\{0, \ldots, N-1\}$, we can define QFT as the gate satisfying

$$|j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |k\rangle.$$

As the astute reader will notice, the QFT is simply the discrete Fourier transform with different notation. Interestingly, QFT can be efficiently decomposed into controlled-Hadamard gates and controlled-phase gates, though the proof is too technical for this paper. If you are interested in reading this proof, check out [NC00].

6. Simulating Classical Computers

The complexity class **BQP** contains all problems that can be decided by a quantum computer in polynomial time, within some bounded error. In this section, we prove the following:

Theorem 6.1. *Any classical circuit can be simulated on a quantum computer. In other words,*

$$\boldsymbol{P \subseteq BQP}.$$

It seems trivial that we could simulate classical computers on quantum computers. After all, it seems like we could just measure all of our qubits and then feed them into a classical circuit. However, we encounter one major obstacle: unitary matrices (and hence quantum gates) are invertible by definition, but not all classical gates are invertible. Thus we cannot simply plug qubits into classical gates. As such we must construct a different way to simulate classical circuits. We will do so using the following lemma, which we will not prove within this paper.

Lemma 6.2. *The classical* `NAND` *gate (which is essentially an* `AND` *gate followed by a* `NOT` *gate) is a universal gate, meaning that any classical circuit can be simulated (albeit perhaps really slowly) by applying multiple* `NAND` *gates.*

Proof of Theorem 6.1. Since we know by the preceding lemma that `NAND` gates are universal, we just need to find a way to simulate a `NAND` gate via a quantum circuit. We can do this by using a Toffoli gate, which operates on 3 qubits and is defined by the following actions on computational basis states:

$$\begin{aligned} |111\rangle &\mapsto |110\rangle, \\ |110\rangle &\mapsto |111\rangle, \\ |x\rangle &\mapsto |x\rangle \qquad \forall x \notin \{110, 111\}. \end{aligned}$$

The reader can verify that the action of the Toffoli gate on each computational basis state matches the action of the `NAND` gate on its analogous classical state. Thus we have shown that `NAND` gates can be simulated on quantum computers, and by the universality of `NAND` gates we see that we can simulate any classical circuit on a quantum computer. ■

7. Fast Quantum Algorithms

Quantum algorithms aren't just useful for simulating classical algorithms. Quantum algorithms can also solve certain problems faster than any known classical algorithm. One problem where this is the case is the problem of prime factorizing an integer. The fastest known classical factoring algorithm, the general number field sieve, runs in sub-exponential time (which is slower than polynomial time). However, Shor's algorithm is a quantum algorithm that runs in polynomial time. In this section, we will give a brief overview of Shor's factoring algorithm.

Shor's factoring algorithm has two parts: a classical part and a quantum part. The classical part of Shor's algorithm reduces the factorization of N to a problem about finding the period of a function, while the quantum part uses QFT to find the period, speeding up the process of solving the problem of factoring N.

In the classical part of Shor's algorithm, we guess a candidate a with $1 < a < N$ that can help us factor N. We then check if a and N are relatively prime via the Euclidean algorithm.

If they are not, we can stop as their common factor divides N. Now comes the quantum step, where we use the QFT to find the least r such that

$$a^r \equiv 1 \mod N.$$

If $a^{r/2}$ is a nontrivial root of 1 modulo N, then we know that $\gcd(a^{r/2} \pm 1, N)$ are nontrivial factors of N, so we're done. Otherwise, we start over with another guess of N.

Shor's algorithm encapsulates the benefits of quantum computing outside of simply simulating classical computers. And it's not the only one: quantum search algorithms can search over an unsorted space in sub-linear time [NC00].

8. Recap

Quantum computing is a powerful tool that could become the standard computing paradigm of the future. While quantum computing relies heavily on the physics behind quantum mechanics, a high-level understanding of quantum computing can be gleaned simply through the use of linear algebra. Quantum computers can simulate classical computers without significant slowdown, and seem to be able to solve some problems faster than classical computers can. Modern cryptography is mainly based on public-key cryptosystems, which all rely on the difficulty of factoring integers. But due to Shor's algorithm, current cryptosystems would be need to be replaced with better (quantum) cryptosystems if quantum computing was to become widespread.

9. Acknowledgements

The author would like to thank Sherry Sarkar and Simon Rubinstein-Salzedo for insightful guiding conversations.

References

[Axl97] Sheldon Jay Axler. *Linear Algebra Done Right*. Undergraduate Texts in Mathematics. Springer, New York, 1997. URL: http://linear.axler.net/.

[blo] Bloch sphere representation of quantum states for a spin $\frac{1}{2}$ particle. URL: https://www.st-andrews.ac.uk/physics/quvis/simulations_html5/sims/blochsphere/blochsphere.html.

[NC00] M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information*. Cambridge Series on Information and the Natural Sciences. Cambridge University Press, 2000. URL: https://books.google.com/books?id=aai-P4V9GJ8C.

Email address: bhattacharya.arav05@gmail.com