AN EXPOSITION ON PETER SHOR'S POLYNOMIAL-TIME FACTORING ALGORITHM AND ITS EFFECTS ON POST-QUANTUM CRYPTOGRAPHY

A Thesis

by

KRISTOPHER LONNIE WATKINS

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2019

Major Subject: Mathematics

ABSTRACT

When building cryptosystems, cryptographers focus on finding problems that are not believed to be solvable in polynomial-time. Some of the most popular problems they have found are the Discrete Logarithm Problem and Integer Factoring. The former is used in Diffie-Hellman Key Exachange (DHK) and El Gamal encryption, while the latter is used in RSA. El Gamal and DHK are both very popular, but RSA is more prevalent due to its efficiency. Nevertheless, it is plausible that in the next few decades, all three of these systems will likely be useless due to the advances made by Peter Shor in quantum computing.

This paper will explain the details of how Shor's algorithm works and how it accomplishes the above. It will also feature a redesign of the proof of Jeffrey Miller (1975) that efficiently reduces from order finding in a group of order N to factoring N. Hopefully, doing so will aid future students in their studies of quantum algorithms and Post-Quantum Cryptography.

DEDICATION

To Lonnie, Reba, Lynnette, Arnold, Gina, Robert A., Barabara, Jeremy, Kellie, Davean, Adam, Emilie, Brooke, Megan, Morgan, Rachel, Arpan, Pablo, John, Agniva, Kelly, Thomas, Andrew, Dr. Staron, Dr. Schielke, Dr. Yasskin, Dr. Jung, and Dr. Rojas. Thank you for all that you have done to get me to this point in my life.

ACKNOWLEDGMENTS

I would like to thank Dr. Rojas for his advice and guidance. His mentorship has led me to finding my favorite area of mathematics and my dream area to work in. The opportunity to learn from him has developed my math and research skills more than I could have imagined. I would also like to thank the Department of Mathematics, especially Monique Stewart and Dr. Howard, since there's no way I would have been in the position to graduate if it wasn't for all that they have done for the department and me.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professors J. Maurice Rojas and Riad Masri of the Department of Mathematics and Professor Mortari of the Department of Aerospace Engineering.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was supported by a job opportunity from the Texas A&M Math Circle by Dr. Yasskin.

NOMENCLATURE

$\mathcal{BQP}$	Bounded-Error Quantum Polynomial Time
lg	$\lg(x) = \log_2(x) =$
QFT	Quantum Fourier Transform
FFT	Fast Fourier Transform
$\gcd(x, y)$	Greatest Common Divisor of x and y
$\mathbb{F}_p$	The field containing $\{1, 2, \ldots p\}$
$\mathbb{Z}/\langle N\rangle$	The ring of integers mod N
U^*	The complex-conjugate transpose of the matrix U
$\lceil x\rceil$	The smallest integer greater than x
$\|x\|$	The order of x in a given group

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 Cryptography Overview

Whenever we use messaging services (email, text, etc.), banking services, or anything that deals with transferring information that we don't want to be public, we are using some sort of cryptosystem. These cryptosystems are made by cryptographers, and they use an encryption protocol that is appropriate for the information being shared. This means that the system must be fast enough for the users; i.e., we don't want to wait 10 minutes for our phones to encrypt a text message before it sends. It also means that the system must be secure enough to deal with the anticipated attack. For example, in cabinets we probably store something like basic foods, whereas in a fire safe we're going to store something like private documents (passport, social security card, etc.). We're not too concerned about someone breaking into our home just to steal the food, but we are concerned about the documents being stolen. Thus, we build fire safes more securely than cabinets.

When we make cryptosystems to protect information, we are generally basing the cryptosystems off of the fact that it's "hard" to solve some given problem. For our most secure systems we are going to choose problems that (hopefully) require exponential time to solve. Unfortunately, efficient cryptosystems that are *provably* this hard to break remain unknown. What is currently known is how to build useful cryptosystems whose breakage implies the resolution of well-known computational problems *expected* to be algorithmically hard in practice. In order to show these things and verify that the cryptosystem is being based on an appropriate problem, we need a fundamental understanding of complexity theory, abstract algebra, and number theory. Despite that, it's been shown that cryptography has been around since approximately 1900 BC. However, a new and potentially dangerous sub-field, quantum cryptography, arose in 1970. The danger comes from the potential to break current cryptosystems that are secure enough to be used by the NSA, FBI, DEA, etc.

My interest lies in the field of post-quantum cryptography which is the study of cryptography with the assumption that we have a stable quantum computer that can run quantum algorithms.

Fortunately for us, it's estimated that we still have 25-100 years before such a computer can be made which gives us time to study post-quantum cryptography and come up with alternatives to our current cryptosystems. One can get away without knowing the quantum mechanics that lie behind the quantum computer, but a decent knowledge of computer science is certainly needed for post-quantum cryptography. One side of post-quantum cryptography is aimed at proposing a method of encryption that can't possibly be cracked by classical or quantum computers. The other side is to take problems from current cryptosystems and prove that they belong to the complexity class $\mathcal{BQP}$. $\mathcal{BQP}$, Bounded-Error Quantum Polynomial Time, is the complexity class of decision problems that can be solved with a correct answer at least $2/3$ of the time. The latter is the side that I'm interested in, more specifically I'm interested in how we break famous cryptosystems such as RSA and how we create cryptosystems that are quantum-resistant. The following will give us an official definition to prove that Integer Factoring is indeed in $\mathcal{BQP}$.

Definition 1. *$\mathcal{BQP}$ is the class of decision problems for which there exists a uniform family of polynomial-size quantum circuits that can be solved with a correct answer greater than $2/3$ of the times the problem is attempted.*

1.2 Overview of RSA

RSA is a cryptosystem made by Rivest, Shamir, and Adleman that assumes that factoring an arbitrary integer is considered to be hard, classically. In outline we have two people, Alice and Bob, who are trying to communicate without their adversary, Eve, getting in the way. Alice secretly chooses two prime numbers, p and q. She then publishes $n = p{\cdot}q$ and r such that $\gcd(r, \phi(n)) = 1$ where ϕ is Euler's Totient function. Bob then chooses a message M such that $1 \leq M < n$ and checks that $\gcd(M, n) = 1$. Bob then computes $C \equiv M^r \pmod n$ and sends C to Alice. Alice can then find s such that $r \cdot s \equiv 1 \pmod{\phi(n)}$ and then use that to compute $C^s \equiv M \mod (n)$. In order to recover the message, Eve needs to find $\phi(n)$ which means that she needs to be able to factor n. Even our best algorithms (Elliptic Curve method, Quadratic Sieve, etc.) are unable to factor a general integer faster than $e^{\mathrm{O}((\log(N))^{1/3}(\log(\log(N)))^{2/3})}$ by [5]. This is good because that running time is exponential and thus doesn't put integer factoring in $\mathcal{P}$. This means that it's safe for our government and our servers to continue using RSA encryption.

1.3 Overview of El Gamal and Diffie-Hellman

There is another type of encryption known as El Gamal which is based off the Diffie-Hellman Key-Exchange Protocol which relies on the Discrete Logarithm Problem (DLP) being hard. For DLP, we are trying to find x such that $g^x \equiv b \pmod{n}$ when given $g, b,$ and n. We can apply this to an Alice, Bob, and Eve situation as follows. First, Alice chooses a prime p that is large enough that it's unfeasible to solve DLP in $\mathbb{F}_p$. Alice will secretly select a private key, a and compute $A \equiv g^a \pmod{p}$ where $g \in \mathbb{F}_p$. Bob will now need to send a message, $m \in \mathbb{Z}$, to Alice where $2 \leq m < p$. In order to encrypt m, Bob computes $b_1 \equiv g^k \pmod{p}$ and $b_2 \equiv mA^k \pmod{p}$ where k is a randomly selected element of $\mathbb{F}_p$. He will then send b_1 and b_2 to Alice and she will compute $(b_1^a)^{-1} \cdot b_2 \pmod{p}$ in order to get m. A huge downside here is that we are using 2-1 encryption. This means that our cipher-text has to be twice as long as our original message. Due to ease of use and speed, we primarily use RSA to encrypt information. There are various types of RSA, with the most secure version being RSA-2048 which uses numbers that are 2048 bits long (617 decimal digits).

1.4 Motivation for Studying Shor's Algorithm

In 1994 Dr. Peter Shor released his algorithm [7] for factoring a general integer with O$((\log(n))^2(\log(\log(n)))(\log(\log(\log(n)))))$ quantum gates. This put integer factoring in the complexity class of $\mathcal{BQP}$ and thus implied that we would be able to break RSA with a "good" quantum computer. By "good" quantum computer, we are talking about a quantum computer that is stable enough to run Shor's Algorithm and also utilizes over 4000 qubits. While experts predict that we are still 25-100 years from having such a quantum computer, it's still intimidating to think that our most effective method of encryption will be broken. Bernstein, Mosca, and others have made advances on Shor's Algorithm by decreasing the number of qubits which weakens the time complexity slightly. So it's reasonable that we could have access to a similar algorithm that will allow us to break RSA sooner than we expect.

1.5 Overview of how Shor's Algorithm Works

The quantum algorithm developed by Shor doesn't actually focus on factoring a number. Instead, it focuses on putting the order-finding problem in $\mathcal{BQP}$. This is done by utilizing phase estimation along with the quantum fourier transform and then applying some number-theoretical results about modular exponentiation, continued fractions, and rational numbers. Then with a polynomial reduction from order-finding to integer factoring, given by Dr. Jeffrey Miller in 1975, we get that integer factoring is also in $\mathcal{BQP}$. We'll proceed with a description of the polynomial reduction first.

2. REDUCTION FROM FINDING THE ORDER OF AN ARBITRARY GROUP ELEMENT TO FACTORING AN INTEGER

2.1 Order-Finding Problem

Before starting, we should note that $(\mathbb{Z}/\langle N\rangle)$ is the multiplicative group of all integers less than N that are co-prime to N, i.e., $\mathbb{Z}_8^* = \{1, 3, 5, 7\}$. The Order-Finding problem is the following: Let $N \in \mathbb{N}$ and $x \in (\mathbb{Z}/\langle N\rangle)$. Can we find the minimal $r \in \mathbb{N}$ such that $x^r \equiv 1 \pmod{N}$? Since N is the number that we wish to factor later on, we will also add on the constraint that N is odd and composite. we make N composite because if we were trying to factor a number, we would first use the AKS primality test to determine whether N is prime or not in polynomial time. If it is, then we have the factorization of N and the problem is trivial. We choose N to be odd, because if N were even, then $\frac{N}{2^k}$ is odd for some $k \in \mathbb{N}$, and the problem reduces to factoring $\frac{N}{2^k}$. It's also safe to say that N is not a perfect square because this would also make our factoring problem trivial since we can find the square root of number in polynomial time with Newton's Method. we can now tighten the bounds on x by saying that $2 < x < N - 1$ because of the new constraints on N.

2.2 Polynomial Reduction From Order-Finding to Integer Factoring

So now our goal is to try to get a non-trivial factor of N with success greater than $2/3$ of the time and the assumption that we know the order of x. The following theorem will give us a way of finding such a factor and then we can calculate the probability of the setting of the theorem actually happening.

Theorem 1. *Let $x \in (\mathbb{Z}/\langle N\rangle)$ such that $|x| = r$, $x^{r/2} \not\equiv -1 \pmod{N}$, and r is even. Then $\gcd(x^{r/2} - 1, N)$ gives a non-trivial factor of N in polynomial time.*

Proof.

$$\begin{aligned} r \text{ is even and } |x| = r &\implies x^r \equiv 1 \pmod N \\ &\implies x^r - 1 \equiv 0 \pmod N \\ &\implies (x^{r/2}+1)(x^{r/2}-1) \equiv 0 \pmod N \end{aligned}$$

Now suppose that $\gcd(x^{r/2}-1, N)$ gives a trivial divisor of N.

Case 1: Suppose $\gcd(x^{r/2}-1, N) = N$. Then $N \mid x^{r/2}-1 \implies x^{r/2}-1 \equiv 0 \pmod N$ which contradicts r being the order of x. Thus, $\gcd(x^{r/2}+1, N) \neq N$.

Case 2: Suppose $\gcd(x^{r/2}-1, N) = 1$. Then $x^{r/2}-1 \equiv 1 \pmod N \implies N \mid x^{r/2}+1 \implies x^{r/2}+1 \equiv 0 \pmod N$. However, this contradicts $x^{r/2} \not\equiv -1 \pmod N$. Thus, $\gcd(x^{r/2}-1, N) \neq 1$.

In either case, $\gcd(x^{r/2}-1, N)$ gives a non-trivial divisor of N. The Euclidean Algorithm can then be used to calculate the $\gcd$ in polynomial time. □

Before we find the probability of r being even and $x^{r/2} \not\equiv -1 \pmod N$, we'll need two lemmas. Beforehand, it's important to note that, by the Fundamental Theorem of Arithmetic, N has a unique prime factorization, i.e. $N = p_1^{\alpha_1} \cdot \ldots \cdot p_m^{\alpha_m}$ where every p_i is prime and m is the number of distinct prime factors of N. For notational purposes in the following lemmas, let $r_j = |x_j|$ and d_j be the largest integer such that $2^{d_j} | r_j$. Similarly d is the largest integer such that $2^d | r$.

Lemma 1. *Let $x \in (\mathbb{Z}/\langle N \rangle)$ such that $|x| = r$. If r is odd or $x^{r/2} \equiv -1 \pmod N$, then $d_j = c$ for some $c \in \mathbb{Z}$ and for all j such that $1 \leq j \leq m$.*

Proof. We know that every $r_j \mid r$ because

$$\begin{aligned} x^r &\equiv 1 \pmod{N} \\ &\equiv 1 \left(\bmod \left(\prod_{j=1}^{m} p_j^{\alpha_j}\right)\right) \\ &\equiv 1 \left(\bmod \left(p_j^{\alpha_j}\right)\right) \text{ for all } j \in \mathbb{Z}_m \setminus \{0\} \end{aligned}$$

Case 1: Suppose r is odd. Since we must have $r_j|r$ for all $j \in \mathbb{Z}_m \setminus \{0\}$ we also must have that every r_j is odd. Therefore $d_j = 0 = d$ for all $j \in \mathbb{Z}_m \setminus \{0\}$.

Case 2: Suppose r is even and $x^{r/2} \equiv -1 \pmod{N}$. Similar to case 1, we get that $x^{r/2} \equiv -1 \left(\bmod p_j^{\alpha_j}\right)$ for all $j \in \mathbb{Z}_m \setminus \{0\}$ which means that r_j doesn't divide $\frac{r}{2}$ for every $j \in \mathbb{Z}_m \setminus \{0\}$.

Claim: All of the d_j's are equal to each other.

Proof of Claim: Suppose not. Then $\exists$ distinct $i, j \in \mathbb{Z}_m \setminus \{0\}$ such that $i \neq j$ and $d_i \neq d_j$. WLOG suppose $d_i < d_j$. We know that $r_j \mid r$ and $r_i \mid r$ by Lagrange's Theorem. We know that $r_j = 2^{d_j} q_j$ for some odd, positive integer q_j, and $r_i = 2^{d_i} q_i$ for some odd, positive integer q_i. Since $r_j \mid r$ and $r_i \mid r$, we have that $r = 2^{d_j} q_j k_j$ for some $k_j \in \mathbb{Z}$ and $r = 2^{d_i} q_i k_i$ for some $k_i \in \mathbb{Z}$. Thus, $2^{d_j} \mid r$ and $q_i \mid r$. Since q_i is odd, we must have that 2^{d_j} and q_i are co-prime. Therefore,

$$\begin{aligned} 2^{d_j} q_i \mid r &\implies 2^{d_j - d_i + d_i} q_i \mid r \\ &\implies 2^{d_j - d_i} 2^{d_i} q_i \mid r \\ &\implies 2^{d_j - d_i} r_i \mid r \\ &\implies r_i \mid \frac{r}{2} \text{ which is a contradiction.} \end{aligned}$$

Therefore, $d_i = d_j$ for all $i, j \in \mathbb{Z}_m \setminus \{0\}$. Since they are all equal we also have that $d_i = d$ for all $i \in \mathbb{Z}_m \setminus \{0\}$. ☐

Lemma 2. *Let* p *be an odd prime and* h *is the largest power of 2 that divides* $\phi(p^\alpha)$ *where* ϕ *is the Euler* ϕ *function. Then with probability* $\frac{1}{2}$, 2^h *divides the order,* y*, modulo* p^α *of a random* $x \in \mathbb{Z}^*_{p^\alpha}$*.*

Proof. Let $x \in \mathbb{Z}^*_{p^\alpha}$. As a property of the Euler ϕ function, we know that $\phi(p^\alpha) = p^{\alpha-1} \cdot (p -$

1). Since p is odd we know that $\phi(p^\alpha)$ is even which means that $h \geq 1$. $\mathbb{Z}^*_{p^\alpha}$ is cyclic under multiplication so $x = g^k \pmod{p^\alpha}$ where g is a generator of $\mathbb{Z}^*_{p^\alpha}$ and $k \in [0, \phi(p^\alpha)] \cap \mathbb{Z}$.
Case 1: Suppose k is odd. Then we have

$$\begin{aligned} x^y = 1 \pmod{p^\alpha} &\implies g^{ky} = 1 \pmod{p^\alpha} \\ &\implies \phi(p^\alpha) \mid ky \\ &\implies 2^h \mid ky \\ &\implies 2^h \mid y, \text{ since } k \text{ is odd} \end{aligned}$$

Case 2: Suppose k is even. Then $x = g^k \pmod{\phi(p^\alpha)} \implies x^{\phi(p^\alpha)/2} \pmod{(\phi(p^\alpha))} = 1 \pmod{\phi(p^\alpha)}$. This means that $y \mid \phi(p^\alpha)/2$. Since h is the largest integer such that $2^h \mid \phi(p^\alpha)$ we know that 2^h doesn't divide $\phi(p^\alpha)/2$. Therefore 2^h doesn't divide y.
Since the probability of $k \in [0, \phi(p^\alpha)] \cap \mathbb{Z}$ being odd is $\frac{1}{2}$, we have that the probability of 2^h dividing y is $\frac{1}{2}$. □

Theorem 2. *Let* $x \in (\mathbb{Z}/\langle N \rangle)$ *such that* $|x| = r$. *Then the probability that* r *is even and* $x^{r/2} \not\equiv -1 \pmod{N}$ *is greater than or equal to* $1 - 2^{-m}$ *where* $N = p_1^{\alpha_1} \cdot \ldots \cdot p_m^{\alpha_m}$.

Proof. PROB(r is even and $x^{r/2} \not\equiv -1 \pmod{N}$) $= 1-$PROB(r is odd or $x^{r/2} = -1 \pmod{N}$) so it suffices to prove that PROB(r is odd or $x^{r/2} = -1 \pmod{N}$) $= 2^{-m}$. By Lemma 1, we know that these conditions can only be met if all of the d_j's are equal to d. The probability of this happening for just one of the d_j's is at most $\frac{1}{2}$ by Lemma 2.
Therefore PROB(r is odd or $x^{r/2} = -1 \pmod{N}$) $\leq 2^{-m} \implies$ PROB(r is even and $x^{r/2} \not\equiv -1 \pmod{N}$) $\geq 1 - 2^{-m}$. □

Now we have shown that given the order of an arbitrary element in a group of order N, we can get a non-trivial factor of N with at least $3/4$ chance of being correct. We have at least a 75% chance because N is a composite number that is not a perfect square, so it must have at least two different prime factors. Thus, $m \geq 2$ so $1 - 2^{-m} \geq \frac{3}{4}$. This means that if the Order-Finding problem is in $\mathcal{BQP}$, then so is integer factoring. Before showing how to place the Order-Finding problem in $\mathcal{BQP}$, we'll first take a look at some of the rudimentary concepts associated with the mathematics of quantum computing.

3. CLASSICAL COMPUTING VS. QUANTUM COMPUTING AND THE MATH INVOLVED

3.1 Differences Between Classical and Quantum Computing

Suppose we have a number $N \in \mathbb{N}$. We can represent N in binary notation as N is an element of $\{0, 1\}^k$ where $k = \lfloor \lg(N) + 1 \rfloor$ (the number of bits required to represent N in binary notation without a sign bit). This representation is independent of whether we are using a classical or quantum computer. In both scenarios we have registers that store these bits/qubits. The differences between classical and quantum computing come from how these registers act when measured and how the bits/qubits work. These differences also help us to understand why we are so interested in the complexity class $\mathcal{BQP}$ when it comes to quantum computing.

When representing N in classical computing, what we really have is a register that contains k flip flops. You can think of some sequence $x_1 x_2 \ldots x_k$ where all of the x_j's are light switches. When we turn a light switch on we get a 1 for the x_j and when we turn a light switch off we get a 0 for the x_j. Anybody who has ever tried to play with a normal light switch in their house, would know that the light switch can't be off and on at the same time. Thus, we can only represent one number at a time with this sequence of x_j's. The upside is that we can flip light switches on and off while observing the different light settings without destroying anything. In the language of classical computing, this means that we can measure our register with a non-destructive measurement so that we can continue using that register later on if need be.

In quantum computing our register, $|\Psi_k\rangle$, is made up of k qubits. A huge upshot for quantum computing is that $|\Psi_k\rangle$ doesn't just represent one number. Instead, it represents every number $y \in \mathbb{Z}$ such that $0 \leq y \leq 2^k - 1$ (we can represent negative numbers and rational numbers as well by using the same methods that classical computing uses). We do this by putting each qubit into a superposition of states that will give us some number with high probability when we perform our measurements. However when we perform this measurement we use "destructive" measurement. This prevents us from accessing a lot of information from the register.

Example of destructive measurement: Suppose we have a 2 qubit register $|\Psi_2\rangle = |a\rangle\,|b\rangle$ such that $|a\rangle = a_1\,|0\rangle + a_2\,|1\rangle$ and $|b\rangle = b_1\,|0\rangle + b_2\,|1\rangle$ where $a_1, a_2, b_1, b_2 \in \mathbb{C}$. Every qubit must have a norm of 1, i.e. $|a_1|^2 + |a_2|^2 = |b_1|^2 + |b_2|^2 = 1$. The multiplication of $|a\rangle$ and $|b\rangle$ is just the tensor product of their respective vectors, i.e. $|a\rangle\,|b\rangle = a_1b_1\,|00\rangle + a_1b_2\,|01\rangle + a_2b_1\,|10\rangle + a_2b_2\,|11\rangle$. Now suppose we measure $|a\rangle$ and we get $|a\rangle = 1$. Now we have $|a\rangle\,|b\rangle = a_2b_1\,|10\rangle + a_2b_2\,|11\rangle$. However, this will not satisfy the norm requirement unless $a_1 = 0$. This means that we are most likely changing our coefficients to get $|a\rangle\,|b\rangle = a_2'b_1'\,|10\rangle + a_2'b_2'\,|11\rangle$ to satisfy the norm requirement.

3.2 How to Visualize and Understand Qubits

The superposition of a quantum state is represented as

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

The coefficients of the computational bases, namely α_0 and α_1, represent the up-spin and down-spin of a particle respectively. The only criteria on the coefficients are that they are complex numbers with modulus (distance from $(0,0) \in \mathbb{C}$) no greater than 1 and that the sum of the squares of their moduli is equal to 1.

One way of visualizing this is by using the Bloch Sphere from [6]. In order to do this we will see that every $|\Psi\rangle$ can be written as below, simply by factoring out the phase of α_0.

$$|\Psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$

The following shows that this is an acceptable definition by the criteria of the coefficients:

Criterion 1: Clearly the coefficients are complex numbers. The image of the cosine function gives us that $|\cos\left(\frac{\theta}{2}\right)| \leq 1$. To find the modulus of the second coefficient we have $|e^{i\phi}\sin\left(\frac{\theta}{2}\right)| = |e^{i\phi}| \cdot |\sin(\frac{\theta}{2})| = (\cos^2(\phi) + \sin^2(\phi)) \cdot |\sin(\frac{\theta}{2})| = \sin(\frac{\theta}{2}) \leq 1$ by the image of the sine function.

Criterion 2: The sum of the squares of their moduli is given by

$$\left|\cos\left(\frac{\theta}{2}\right)\right|^2+\left|e^{i\phi}\sin\left(\frac{\theta}{2}\right)\right|^2=\left|\cos\left(\frac{\theta}{2}\right)\right|^2+\left(\left|e^{i\phi}\right|\cdot\left|\sin\left(\frac{\theta}{2}\right)\right|\right)^2=\cos^2\left(\frac{\theta}{2}\right)+\sin^2\left(\frac{\theta}{2}\right)=1$$

Now if we take $(x, y, z) = (\sin(\theta)\cos(\phi), \sin(\theta)\cos(\phi), \cos(\theta))$ then we get a point on the unit sphere where the closer we are to the north or south pole, the higher probability we have of measuring a 0 or 1 respectively. This can all be seen in the following picture:

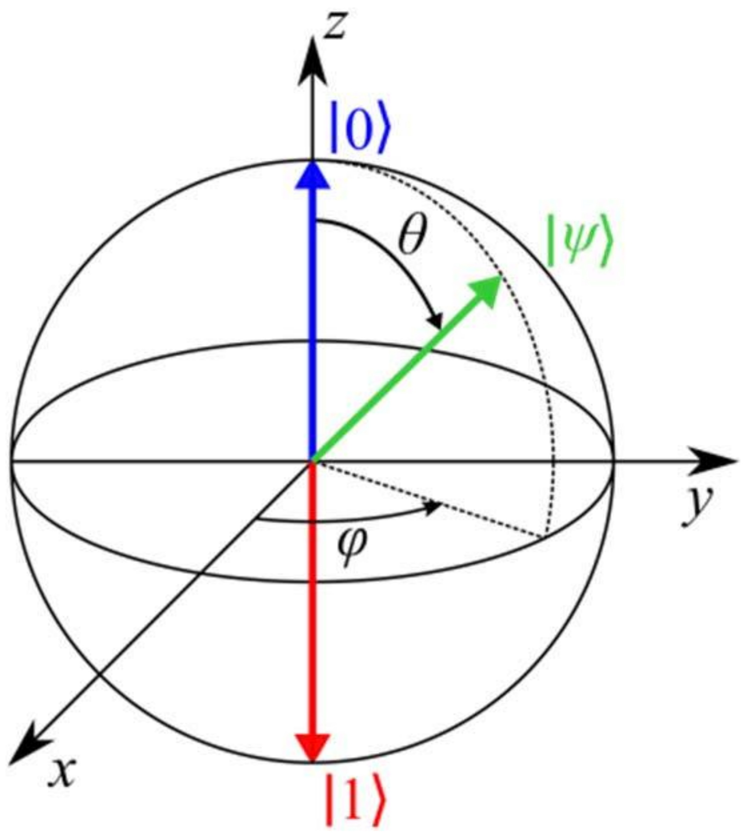


Figure 3.1: Representation of a qubit on the Bloch Sphere adapted from [6]

4. PHASE ESTIMATION

4.1 The Problem for Phase Estimation

The motivation for the following is that if we can place the phase estimation problem in $\mathcal{BQP}$ then we can place the order-finding problem in $\mathcal{BQP}$. Phase Estimation is a quantum procedure to finding the eigenvalue of a given unitary operator U and a given eigenvector $|\psi\rangle$. Since a unitary operation preserves the norm of a vector, the corresponding eigenvalue of U is just an exponential complex function with real value ϕ such that $0 \leq \phi < 1$. Thus we can state the problem we are trying to solve as follows: Given a unitary operator U and it's eigenvector $|\psi\rangle$, find an arbitrarily close estimation for ϕ such that $0 \leq \phi < 1$ and $U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle$. Now note that $e^{2\pi i\phi} = \cos(2\pi\phi) + i\sin(2\pi\phi) \implies \phi \in [0,1)$. We can write the binary approximation of ϕ as $\tilde{\phi} = 0.\phi_1\phi_2\ldots\phi_t = \frac{\phi_1}{2} + \frac{\phi_2}{2^2} + \ldots + \frac{\phi_t}{2^t}$ where the size of t is dependent on how close we want the approximation to be, how many qubits our quantum computer can use, and the range of error that we want (this will be explained later when we analyze phase estimation procedure). Note that if $\phi = \tilde{\phi}$ for some t then ϕ is called a Dyadic number.

4.2 Solving Phase Estimation

It's important to note that the following is not an actual quantum algorithm. That is, phase estimation relies on the fact that we have "black-boxes" (also known as oracles) that are capable of presenting us with $|\psi\rangle$ and U. That being said, we will refer to phase estimation as a procedure for how to solve for ϕ, given the above-mentioned black-boxes. We start with two registers named R_1 and R_2. In R_1 we will have t qubits that are set equal to $|0\rangle$. We will refer to these qubits as $|x_i\rangle$ such that $1 \leq i \leq t$. In R_2 we will store $|\psi\rangle$ and apply controlled-U operations. R_2 will not be important in the sense that we will never measure it. Afterwards, we apply the circuit model given on the next page.

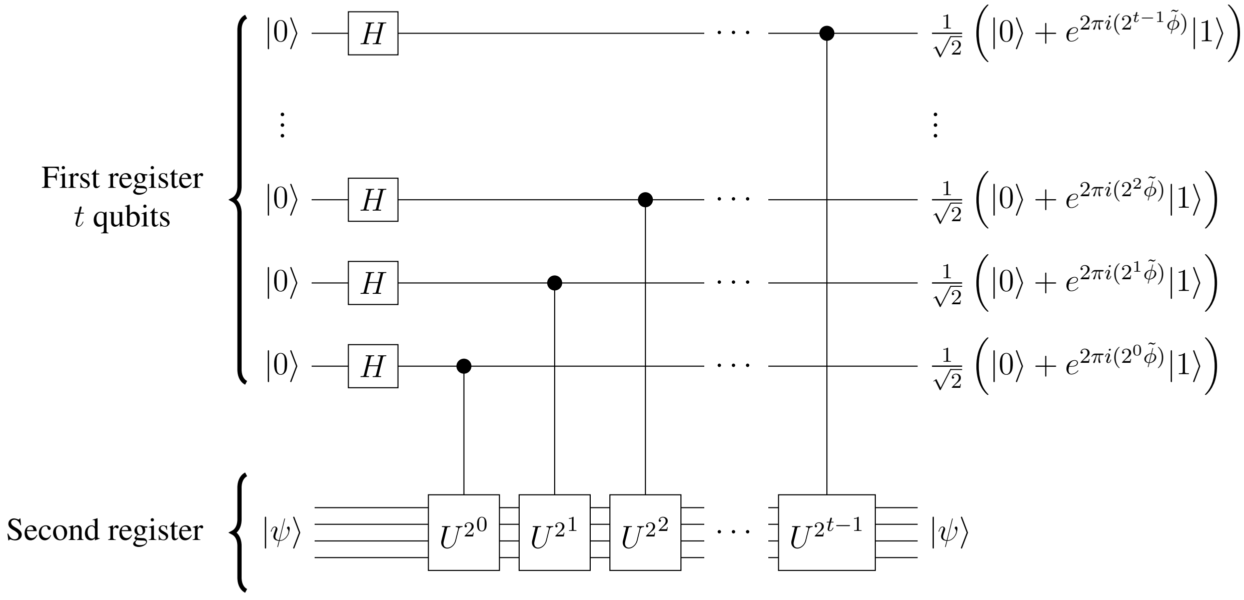

First register
t qubits
Second register
$|0\rangle$
H
$|\psi\rangle$
U^{2^0}
U^{2^1}
U^{2^2}
$U^{2^{t-1}}$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(2^{t-1}\tilde{\phi})}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(2^{2}\tilde{\phi})}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(2^{1}\tilde{\phi})}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(2^{0}\tilde{\phi})}|1\rangle\right)$

Figure 4.1: Phase Estimation Circuit Model adapted from [6]

In the above circuit model we have H, the Hadamard gate, is the quantum gate with matrix representation given by $H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. The Hadamard gate just takes a qubit and puts it into superposition with an equal chance of measuring 0 or 1. We can verify that the Hadamard gate is indeed a unitary operator by making quick use of the following theorem.

Proposition 1. *Let U be a matrix of size $n \times n$ where the rows of U are denoted $u_1, u_2, \ldots, u_n$. If every pair of distinct row vectors is orthogonal and $\vec{u}_i \cdot \overline{\vec{u}_i} = 1$ for all $1 \leq i \leq n$, then U is unitary.*

Going forward, when a matrix is stated to be "clearly unitary", it is because a quick computation via Theorem 3 will show that it is indeed unitary. Now as much as we would love to just state that a the multiplication of a unitary matrix with a suitable vector preserves the norm of the vector, such a statement requires proof. In the proof we'll use the following lemma.

Lemma 3. *Let U be a $n \times n$ matrix and v be a column vector in $\mathbb{C}^n$. Then $(Uv)^* = v^* U^*$.*

Proof. Let u_{ij} denote the entries in U, u_i denote the i-th row of U and let v_i denote the entries of v. Note that $\overline{u_i}$ is equal to the row vector u_i where we have taken the complex conjugate of each entry.

$$Uv = \begin{bmatrix} u_1 \cdot v \\ u_2 \cdot v \\ \vdots \\ u_n \cdot v \end{bmatrix} \implies (Uv)^* = \begin{bmatrix} \overline{u_1 \cdot v} & \overline{u_2 \cdot v} & \dots & \overline{u_n \cdot v} \end{bmatrix}$$

$$v^*U^* = \begin{bmatrix} \overline{u_1} \cdot \overline{v} & \overline{u_2} \cdot \overline{v} & \dots & \overline{u_n} \cdot \overline{v} \end{bmatrix} = (Uv)^*$$

□

Corrolary 1. *Let U be a $n \times n$ unitary matrix and v be a row-vector in $\mathbb{C}^n$. Then Uv preserves the norm of v, that is, $\|v\| = \|Uv\|$*

Now going back to the circuit model for phase estimation, we can see how the controlled-U gates work. The black dots in the circuit model represent which qubit is being acted on by each controlled-U gate. Essentially if a qubit is in the state $|0\rangle$ then nothing will happen, but if a qubit is in the state $|1\rangle$ then we will adjust the coefficient of the qubit. Since we still have to satisfy the norm requirements for a quantum state, we are just changing the phase of qubit, i.e. changing the exponent in the coefficient $e^{2\pi i}$. We can now represent the first register as

$$|R_1\rangle = \bigotimes_{k=0}^{t-1} \tfrac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i 2^k \tilde{\phi}} |1\rangle \right) = \tfrac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2k\pi i \tilde{\phi}} |k\rangle$$

Using the binary approximation of $\tilde{\phi}$ from 4.1, we can substitute for $\tilde{\phi}$ to get

$$|R_1\rangle = \tfrac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2k\pi i [.\phi_1 \phi_2 \dots \phi_t]} |k\rangle \tag{4.1}$$

An important thing to note is that we only used O(t) gates in register 1 of the phase estimation circuit model (we'll talk about register 2 later). This is key because it fits with our definition of

$\mathcal{BQP}$ (it also fits the probability of error part of the definition which will be explained later). Next we'll take a look at the Quantum Fourier Transform (QFT) to show how to obtain $\tilde{\phi}$ from $|R_1\rangle$.

5. QUANTUM FOURIER TRANSFORM

5.1 Deriving and Developing Definitions of the QFT

For those who are already familiar with Fourier Transforms such as the Discrete Fourier Transform (DFT) you may think of the QFT as the quantum analogue of DFT^{-1}. For others, we'll define the QFT in multiple ways below. As always, in quantum computing we should check that the QFT is a unitary operator before we do anything with it. To do so, it's best to start with the following definition.

Definition 2. *Let $|X_n\rangle$ be a register consisting of n qubits denoted by $x_1, x_2, \ldots, x_n$. Then the matrix representation of the QFT acting on $|X_n\rangle$ is given by*

$$QFT(|X_n\rangle) = \frac{1}{\sqrt{2^n}} \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & e^{\frac{2\pi i}{2^n}} & e^{\frac{4\pi i}{2^n}} & \ldots & e^{\frac{2(2^n-1)\pi i}{2^n}} \\ 1 & e^{\frac{4\pi i}{2^n}} & e^{\frac{8\pi i}{2^n}} & \ldots & e^{\frac{4(2^n-1)\pi i}{2^n}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{\frac{2(2^n-1)\pi i}{2^n}} & e^{\frac{4(2^n-1)^{\pi} i}{2^n}} & \ldots & e^{\frac{2(2^n-1)^2)\pi i}{2^n}} \end{bmatrix}$$

If we want the QFT to act on a single basis state given by $|x_1 x_2 \ldots x_n\rangle$ then we can look at the entries of the i-th row, where $i = x_n + 2x_{n-1} + \ldots 2^{n-1}x_1$, to get the coefficients of $x_1, x_2, \ldots x_n$.

For example, let's look at the QFT acting on a register of 3 qubits. The corresponding matrix would be given by

$$\frac{1}{2\sqrt{2}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \frac{1+i}{\sqrt{2}} & i & \frac{-1+i}{\sqrt{2}} & -1 & \frac{-1-i}{\sqrt{2}} & -i & \frac{1-i}{\sqrt{2}} \\ 1 & i & -1 & -i & 1 & i & -1 & -i \\ 1 & \frac{-1+i}{\sqrt{2}} & -i & \frac{1+i}{\sqrt{2}} & -1 & \frac{1-i}{\sqrt{2}} & i & \frac{-1-i}{\sqrt{2}} \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & \frac{-1-i}{\sqrt{2}} & i & \frac{1-i}{\sqrt{2}} & -1 & \frac{1+i}{\sqrt{2}} & -i & \frac{-1+i}{\sqrt{2}} \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & \frac{1-i}{\sqrt{2}} & -i & \frac{-1-i}{\sqrt{2}} & -1 & \frac{-1+i}{\sqrt{2}} & i & \frac{1+i}{\sqrt{2}} \end{bmatrix}$$

From this we can gather that

$$\text{QFT}(|100\rangle) = \frac{1}{2\sqrt{2}}\left(|000\rangle - |001\rangle + |010\rangle - |011\rangle + |100\rangle - |101\rangle + |110\rangle - |111\rangle\right)$$

We can also develop a circuit model for the QFT acting on 3 qubits, which can be seen below.

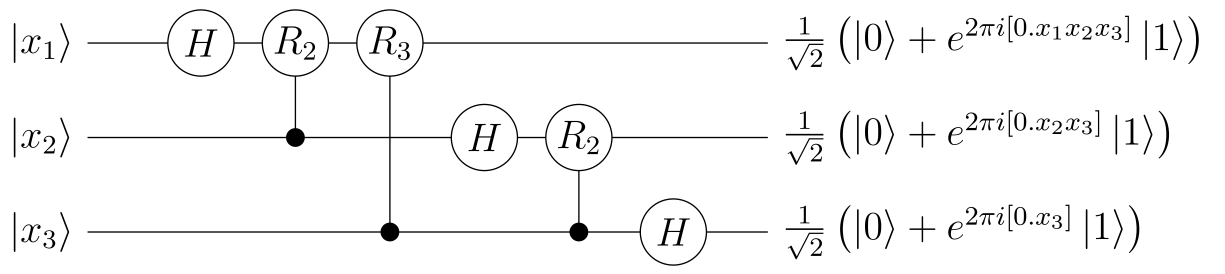

$|x_1\rangle$
$|x_2\rangle$
$|x_3\rangle$
H
R_2
R_3
H
R_2
H
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_1x_2x_3]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_2x_3]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_3]}|1\rangle\right)$

Figure 5.1: Quantum Fourier Transform on 3 qubits adapted from [6]

In the above circuit model we have the Hadamard gate once again showing up and then the R_k's are called controlled-phase gates where $R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{\pi i 2^{-k+1}} \end{bmatrix}$. Once again, the black dots represent the R_k gate acting on the bit on the corresponding line. If the bit is equal to $|0\rangle$ then nothing happens and if it's in state $|1\rangle$ then we change the phase by a factor of $\pi i 2^{-k+1}$. Now we want to try to develop the tensor product equation and the summation equation for n qubits so that we can transfer that into a circuit model for n qubits. Once we do that we will notice a very shocking relation between our circuit model for phase estimation and our circuit model for the QFT. The derivation of the tensor product and summation equation can be seen as following by starting with the map given by the matrix in definition 2.

$$\begin{aligned}
\text{QFT}(|X_n\rangle) &= \frac{1}{\sqrt{2^n}}\left(|0\rangle + e^{2\pi i[.x_n]}|1\rangle\right)\left(|0\rangle + e^{2\pi i[.x_{n-1}x_n]}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i[.x_1x_2\cdots x_n]}|1\rangle\right)\\
&= \frac{1}{\sqrt{2^n}}\bigotimes_{j=1}^{n}\left(|0\rangle + e^{2\pi i[.x_jx_{j+1}\cdots x_n]}\right)\\
&= \frac{1}{\sqrt{2^n}}\bigotimes_{j=1}^{n}\left[\sum_{k_j=0}^{1} e^{2\pi i X_n k_j 2^{-j}}|k_j\rangle\right]\\
&= \frac{1}{\sqrt{2^n}}\sum_{k_1=0}^{1}\cdots\sum_{k_n=0}^{1}\bigotimes_{j=1}^{n} e^{2\pi i X_n k_j 2^{-j}}|k_j\rangle\\
&= \frac{1}{\sqrt{2^n}}\sum_{k_1=0}^{1}\cdots\sum_{k_n=0}^{1} e^{2\pi i X_n\left(\sum\limits_{j=1}^{n} k_j 2^{-j}\right)}|k_1k_2\cdots k_j\rangle\\
&= \frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{2\pi i X_n k 2^{-n}}|k\rangle \text{ note that } |k\rangle \text{ should be written using binary notation}
\end{aligned}$$

The tensor product equation is what we use to get our circuit model, but the summation equation will be restated as another definition of the QFT for computational purposes.

Definition 3. *Let X_n be a positive integer written in binary notation with n qubits, denoted by $x_1, x_2, \dots x_n$ in order from left to right. Then QFT$(X_n) = \frac{1}{\sqrt{2^n}}\sum\limits_{k=0}^{2^n-1} e^{2\pi i X_n k 2^{-n}}|k\rangle$. Note this is equivalent to Definition 2 and the following circuit model.*

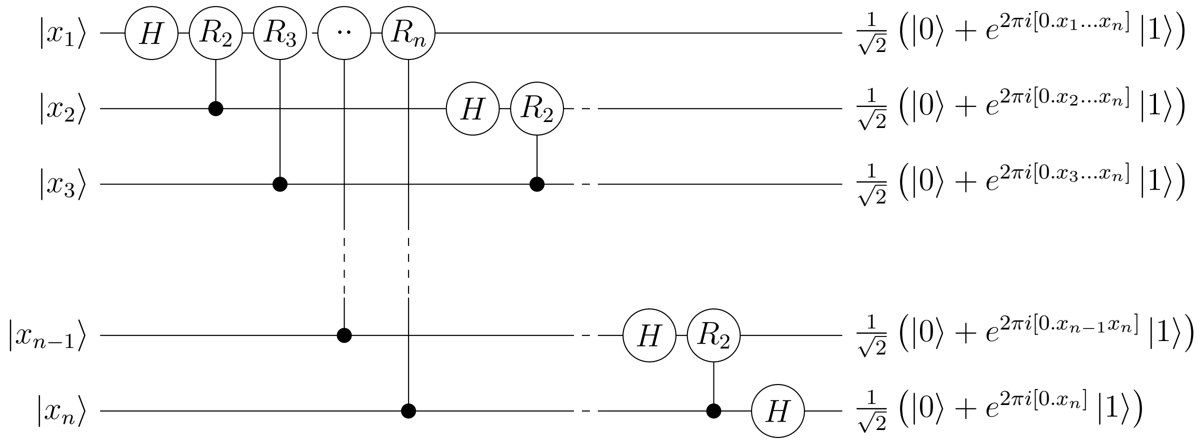

$|x_1\rangle$
$|x_2\rangle$
$|x_3\rangle$
$|x_{n-1}\rangle$
$|x_n\rangle$
H
R_2
R_3
R_n
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_1\ldots x_n]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_2\ldots x_n]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_3\ldots x_n]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_{n-1}x_n]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_n]}|1\rangle\right)$

Figure 5.2: Quantum Fourier Transform on n qubits adapted from [6]

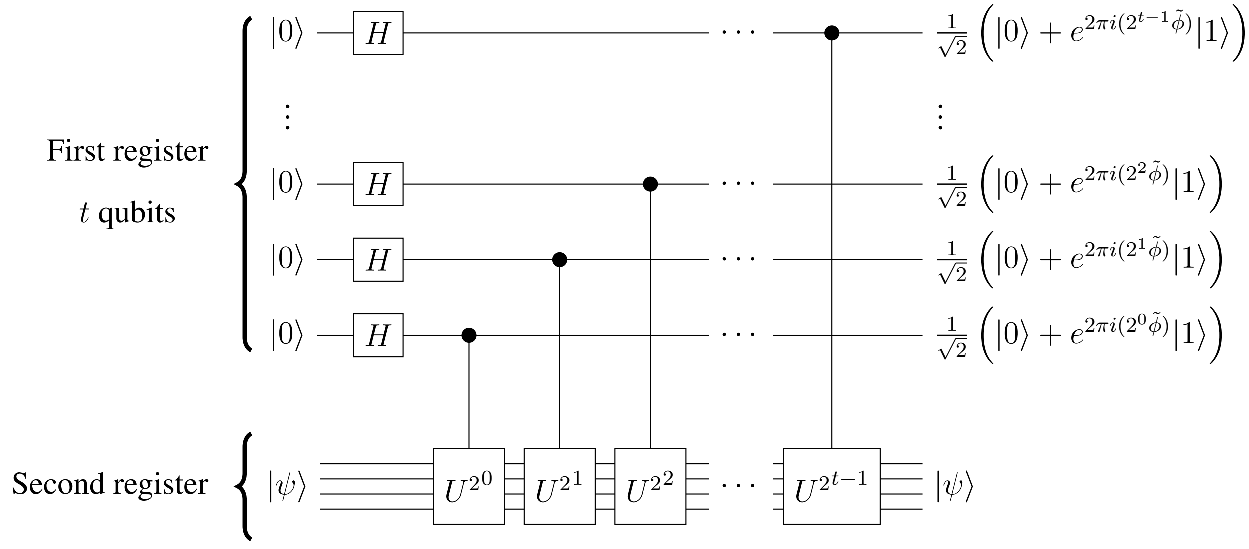
5.2 Making Use of the QFT

Now suppose we wanted make use of awful notation and instead of using X_n and x_j's for the QFT, we chose to use $\tilde{\phi}_n$ and $\tilde{\phi}_j$'s for the QFT. We would then see that if we perform a bit reversal after letting the QFT act on ϕ_n, we would indeed end up with the exact same output as the phase estimation circuit model. Since the QFT is clearly a unitary operator, we know that QFT^{-1} exists and is a unitary operator as well. Thus, we take our output for the phase estimation model, act on it with the QFT^{-1}, and perform a measurement to retrieve $\tilde{\phi}$ (the estimate for the eigenvalue of the unitary operator we were given). Suppose our acceptable range of error from ϕ is 2^{-n}, i.e. we want $\phi - \tilde{\phi} \leq 2^{-n}$. It turns out that if we take $t = n + \lceil \log(2 + \frac{1}{2\varepsilon}) \rceil$ then we can get a successful estimate of ϕ with probability at least $1 - \varepsilon$. Since we are going to be acting on the phase estimation circuit model, we should note that we will have t qubits in the QFT^{-1}. Thus we will have t Hadamard gates, $t-1$ R_2 gates, $t-2$ R_3 gates, $\ldots$, and 1 R_t gate. So in total we will have $\mathrm{O}(t^2) = \mathrm{O}(n^2)$ gates for the QFT^{-1}. Note that we're not including the bit reversal in this because the amount of gates needed for that is negligible in comparison.

6. MODULAR EXPONENTIATION AND CONTINUED FRACTIONS

6.1 What Happened in Register 2 of Phase Estimation

In order to discuss what happened in register 2 of phase estimation, we'll take another look at the circuit model for phase estimation.



First register
t qubits
Second register
$|0\rangle$
H
$|\psi\rangle$
U^{2^0}
U^{2^1}
U^{2^2}
$U^{2^{t-1}}$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(2^{t-1}\tilde{\phi})}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(2^{2}\tilde{\phi})}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(2^{1}\tilde{\phi})}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(2^{0}\tilde{\phi})}|1\rangle\right)$
$|\psi\rangle$

Again let $|R_1\rangle$ denote the contents of the first register. Then the purpose of the controlled-U operations is to take the state $|R_1\rangle\,|\psi\rangle$ to the state $|R_1\rangle\,\left|x^{R_1}\psi \mod (N)\right\rangle$. So, essentially, we just want to multiply the second register $|\psi\rangle$ by $x^{R_1} \mod (N)$. So if we can find the number of gates it takes to compute $x^{R_1} \mod (N)$, then we can combine the number of gates from the first register to find out how many gates are actually used in Phase Estimation. Note that the following can be improved drastically, but since our only concern is placing integer factoring in $\mathcal{BQP}$, we're only concerned with making sure that we have a family of polynomial size circuits. Remember that we are using t qubits in $|R_1\rangle$ where $t = n + \lceil \log(2 + \frac{1}{2\varepsilon})\rceil$. We can compute $x^{R_1} \mod (N)$ with $\mathrm{O}(n)$ squaring operations if we use repeated squaring. Each of these squaring operations will have a cost of $\mathrm{O}(n^2)$ if we use the grade-school multiplication algorithm. Thus, we will not need any more than $\mathrm{O}(n^3)$ gates to compute $x^{R_1} \mod (N)$. Combining this with the number of gates from the first register, we get that phase estimation only requires $\mathrm{O}(n^3)$ gates.

6.2 Why is the Eigenvalue of the U in Phase Estimation Important?

To recap what has been performed so far, we have used a black-box and the phase estimation circuit to set up the phase estimation problem. Afterwards we applied a bit reversal to the QFT and then computed the QFT^{-1} of the output of the phase estimation circuit. The output of all of this is $\tilde{\phi}$ which is our t qubit approximation of the original eigenvalue ϕ. Remember the whole point of this was to get the order of an arbitrary element of $(\mathbb{Z}/\langle N\rangle)$ with high probability so that we could apply our polynomial reduction. It turns out that if we just apply the continued fractions algorithm to $\tilde{\phi}$ then we will do exactly that.

6.3 Continued Fractions Algorithm

Before we apply such an algorithm to $\tilde{\phi}$, we need to verify that there is a finite continued fraction for $\tilde{\phi}$. Remember, $\tilde{\phi} = [.\phi_1\phi_2 \ldots \phi_t] = \frac{\phi_1}{2} + \frac{\phi_2}{4} + \ldots \frac{\phi_t}{2^t}$. Therefore $\tilde{\phi}$ is a finite sum of rational numbers so $\tilde{\phi}$ is a rational number. With the following theorem we will verify that $\tilde{\phi}$ has a finite continued fraction representation which means that the algorithm will terminate. However, since there are multiple ways to represent continued fractions, we'll make use of the following definition.

Definition 4. *A continued fraction for a number x is given by $[x_0, x_1, x_2, \ldots, x_n]$ such that*

$$x = [x_0, x_1, x_2, \ldots, x_n] = x_0 + \frac{1}{x_1 + \frac{1}{x_2 + \frac{1}{\ldots + \frac{1}{x_n}}}}$$

Theorem 3. *Every $x \in \mathbb{Q}$ has a finite continued fraction representation.*

Proof. Since $x \in \mathbb{Q}$ we can write $x = \frac{a}{b}$ such that $a, b \in \mathbb{Z}, b \neq 0,$ and $\frac{a}{b}$ is already reduced into its lowest terms. By the Euclidean algorithm we can write

$$a = x_1 b + r_1 \text{ s.t. } 0 \le r_1 \le b - 1; \tag{6.1}$$

$$b = x_2 r_1 + r_2 \text{ s.t. } 0 \le r_2 \le r_1 - 1; \tag{6.2}$$

$$r_1 = x_3 r_2 + r_3 \text{ s.t. } 0 \le r_3 \le r_2 - 1; \tag{6.3}$$

$$\vdots \tag{6.4}$$

$$r_{n-2} = x_n r_{n-1} \tag{6.5}$$

Dividing by the quotients in each step of the Euclidean algorithm will give us

$$\frac{a}{b} = x_1 + \frac{r_1}{b} \tag{6.6}$$

$$\frac{b}{r_1} = x_2 + \frac{r_2}{r_1} \tag{6.7}$$

$$\frac{r_1}{r_2} = x_3 + \frac{r_3}{r_2} \tag{6.8}$$

$$\vdots \tag{6.9}$$

$$\frac{r_{n-2}}{r_{n-1}} = x_n \tag{6.10}$$

If we back-substitute (6.10) into (6.9), (6.9) into (6.8), . . . , and (6.7) into (6.6) we will get

$$\frac{a}{b} = x_1 + \frac{1}{x_2 + \frac{1}{x_3 + \frac{1}{\dots + \frac{1}{x_{n-1} + \frac{1}{x_n}}}}}$$

Therefore, x must have a finite continued fraction representation since the Euclidean Algorithm must terminate in n steps for some $n \in \mathbb{N}$. $\square$

It's extremely important to note that this part of Shor's Algorithm is done classically because there is no extreme advantage from using a quantum computer at this time. We know that we can run the Euclidean algorithm in polynomial time so we must be able to find the continued fraction representation of $\tilde{\phi}$ in polynomial time (specifically we can do it with O(n^3) gates where n was our range of error on ϕ). Now we'll rewrite $\tilde{\phi} = \frac{a}{b}$ such that $a, b \in \mathbb{Z}$ and $b \neq 0$, since $\tilde{\phi}$ a rational number. We made $\frac{a}{b}$ such that $\phi - \frac{a}{b} \leq 2^{-n}$ so we have a convergent of the continued fraction of ϕ by Theorem 171 in [1]. The only way that b doesn't give us the order of an element is if we had some error in phase estimation or if a and b are not co-prime. For the first scenario remember that we can choose ε so that our chance of failure in phase estimation is suitable. For the second scenario, note that $a < b$ and the number of prime numbers less than b is greater than $\frac{b}{2\log(b)}$. Therefore, the probability that a is co-prime to b is greater than $\frac{1}{2\log(b)}$ which is greater than $\frac{1}{2\log(N)}$. So if we repeat the algorithm $2\log(N)$ times we will have a suitable chance at obtaining a and b such that $\gcd(a, b) = 1$.

7. SUMMARY AND CONCLUSIONS

7.1 Layout of the Entire Algorithm for Integer Factoring

The algorithm for integer factoring can be stated as follows:

Input: N such that $N \in \mathbb{N}$, $N = 2k + 1$ for some $k \in \mathbb{N}$, and N has at least 2 distinct prime factors.

Output: p such that p is a non-trivial factor of N

Number of Gates Needed: $\mathrm{O}(n^3)$ gates are needed where n is given by the range of error allowed on $\tilde{\phi}$ during phase estimation

Procedure:

1. Randomly choose $x \in \mathbb{Z}_N^*$. If $\gcd(x, N) \neq 1$, then return x.

2. Create two quantum registers $|R_1\rangle = |0\rangle$ and $|R_2\rangle = |1\rangle$ so that our quantum system is represented by $|0\rangle\,|1\rangle$

3. Create a superposition in the first register using a Hadamard gate on each qubit for $t = n + \lceil \log(2 + \frac{1}{2\varepsilon}) \rceil$ qubits. This gives us $|R_1\rangle = \frac{1}{\sqrt{2^t}} \sum\limits_{j=0}^{2^t-1} |j\rangle$. So our quantum system is represented by $\frac{1}{\sqrt{2^t}} \sum\limits_{j=0}^{2^t-1} |j\rangle\,|1\rangle$

4. Utilize a black-box and modular exponentiation on the second register to transform it into $|R_2\rangle = |x^j \mod (N)\rangle$. So our quantum system is represented by $\frac{1}{\sqrt{2^t}} \sum\limits_{j=0}^{2^t-1} |j\rangle\,|x^j \mod (N)\rangle$.

5. Utilize a bit reversal, the QFT^{-1}, and then a measurement to obtain $\tilde{\phi}$

6. Apply the continued fraction algorithm

7. Let b be the order (denominator) from the continued fraction algorithm.
   If $x^{b/2} = -1 \mod (N)$ or b is even then restart the algorithm by choosing x' from $\mathbb{Z}_N^* \setminus \{x\}$;
   Else, return $\gcd(x^{r/2} - 1, N)$ to retrieve the non-trivial factor of N.

7.2 Verifying Membership in $\mathcal{BQP}$

Recall that we only used O(n^3) gates in phase estimation and O(n^2) gates in the QFT (so similarly we use O(n^2) gates in the QFT^{-1}. So for steps 1 through 5 of our integer factoring algorithm, we have satisfied the gate requirement of $\mathcal{BQP}$. For steps 6 and 7 we note that both of these can be completed in polynomial time via the euclidean algorithm so they will not affect the algorithm's membership in $\mathcal{BQP}$. The only thing left to verify is that we can achieve a successful output with greater than $2/3$ success. We can choose our chance of error for phase estimation so that part is negligible. Similarly we can run the algorithm iteratively for the continued fraction portion so we can keep our success above $2/3$. As far as the polynomial reduction is concerned we know that the probability of success there is greater than $3/4$. Thus we have shown that with a polynomial family of gates we can achieve a non-trivial factor of N with probability of error less than or equal to $\frac{1}{3}$.

REFERENCES

[1] G. H. Hardy and E. M. Wright. “An Introduction to the Theory of Numbers”. In: *Oxford University Press* (1979), p. 136.

[2] J. Hui. *QC - Quantum Computing Series*. 2019. URL: https://medium.com/@jonathan_hui/qc-quantum-computing-series-10ddd7977abd.

[3] J. Pipher J. Hoffstein and J.H. Silverman. *An Introduction to Mathematical Cryptography*. Springer, 2014.

[4] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An introduction to quantum computing*. Oxford University Press, 2010.

[5] Jonathan Lee and Ramarathnam Venkatesan. “Rigorous Analysis of a Randomised Number Field Sieve”. In: *Journal of Number Theory* 187 (Dec. 2017). DOI: 10.1016/j.jnt.2017.10.019.

[6] M Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[7] P. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* (Oct. 1997), pp. 13–17.