Quantum Computing and Shor's Factoring Algorithm

Department of Mathematics and Statistics

Nisrine Sqalli

April 2025

Abstract

This paper presents what I have learned during my MATH 470 Honours Undergraduate Research Project taken in Winter 2025 with Professor Brent Pym as my supervisor. In this report, I present the framework of quantum computing as an introduction to the field. Then, I present four quantum algorithms that lead up to Shor's factoring algorithm.

Contents

1 Introduction

The theory of quantum computation is an important field of research because of its numerous applications in cryptography and more generally its new computational properties. The most mentioned application of quantum computation is its capacity to break our current methods of cryptography, namely the RSA cryptosystem. It results in a goal to develop new kinds of cryptography using quantum computers. Another reason for why quantum computers are very interesting is the computational power and the storage capacity that it offers which can often times surpass Classical Computers. In this paper, I present the most famous quantum algorithm: Shor's Factoring Algorithm. Peter Shor presented this algorithm in 1994 in his paper *Algorithms for quantum computation: discrete logarithms and factoring* [4].

Chapter 2 and 3 we will present the basics of Quantum Physics and Quantum Computation necessary to understand the rest of the paper. Then, from Chapter 4 to 6, we will work our way through Shor's Factoring Algorithm, which we will present in Chapter 7.

The report is heavily based on *Quantum Computation and Quantum Information* by Michael A. Nielsen and Isaac L. Chuang [2] which presents the important results of Quantum Computation and Information. Section 2.3 results from additional research mostly based on Chapter 1 of *Classical and Quantum Information* by Dan C. Marinescu, and Gabriela M. Marinescu [1]. Additionally, for a deeper understanding of the quantum phase estimation and order finding algorithms, *An algorithm based on quantum phase estimation for the identification of patterns* by Dimitris Ntalaperas, Andreas Kalogeropoulos and Nikos Konofaos [3], was used.
In this paper I aim to present the necessary background needed to understand Shor's Factoring Algorithm.

2 Setup

In quantum computing, we process and store information using quantum states. One of the principles of Quantum Mechanics is that a particle can be in a superposition of states. What does it mean? Let's say we are interested in the energy of a particle. The superposition principle states that the particle can either have a definite energy, in which case we say that it is a definite state, or the particle is in a superposition of definite energies in which case we can consider that it has different values of energy. Only when we measure it, the particle becomes a particle in a definite state, ie. with a definite energy. We can, alternatively, focus on the particle's position, momentum or spin. This principle of superposition opens the door to various new computational tools and techniques.

2.1 The Qubit

A bit in Classical Computation is the smallest unit of data in a computer. It either takes the values 0 or 1 and all of classical computation is based off of that. Its analogue in Quantum Computation is the *quantum bit*, or *qubit* for short, which is a quantum state. There are two definite states in which a qubit can be in, denoted by $|0\rangle$ and $|1\rangle$. We will also call them the basis states. We use the word basis because any qubit can be written as a linear combination of these two states, in other words, they span our whole space. Therefore, any arbitrary qubit $|\psi\rangle$ can be expressed as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2.1}$$

where α and β are complex number which we call amplitudes. For the future, it'll be useful to also think of a qubit as a vector with complex entries:

$$|\psi\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix} \tag{2.2}$$

This new view of computation is compatible with the study of quantum physics where particles are in a superposition of different definite states. An important postulate of Quantum Mechanics is that we can measure the state of a particle. However, it is impossible to measure a superposition, that is, to know both α and β. When we measure the state of a particle, the result will either yield $|0\rangle$ or $|1\rangle$, a definite state. Some people call this **the collapse of the wavefunction upon measuring**. An important question arises, what is the probability of measuring either one of the definite states? The Copenhagen interpretation of Quantum Physics tells us that the probability of measuring the state $|0\rangle$ is $|\alpha|^2$ and the probability of measuring $|1\rangle$ is $|\beta|^2$. It follows that

$$|\alpha|^2+|\beta|^2= 1, \tag{2.3}$$

since the only possible outcomes are $|0\rangle$ and $|1\rangle$.

2.2 The Mathematical Framework

The normalization condition in equation 2.3 is very restrictive on our amplitudes, but is it really necessary? In this section we will answer this question and understand a little bit better the mathematical framework of quantum computing.

Quantum states live in Hilbert space. A Hilbert space is a vector space over the field of complex numbers $\mathbb{C}$ equipped with an inner product. It can have an infinite dimension, but in quantum computation, we will exclusively consider finite dimensional Hilbert Spaces. First, let's consider a 2 dimensional Hilbert space, $\mathbb{C}^2$, in which single qubit states live. Saying that quantum states live in a Hilbert space doesn't necessarily mean that any element of the Hilbert space can be a qubit. The simplest example would be the zero element. No qubit is

nothing, and more precisely, the zero element doesn't satisfy equation 2.3. In addition, many other elements of $\mathbb{C}^2$ don't satisfy equation 2.3, so do we only consider the ones that do satisfy it? Notice that in terms of the probabilistic interpretation of quantum mechanics, multiplying $|\psi\rangle$ by $e^{i\theta}$ for any real number θ doesn't change the information we have on the state:

$$|\psi\rangle \mapsto |\psi'\rangle = e^{i\theta}|\psi\rangle = e^{i\theta}\alpha|0\rangle + e^{i\theta}\beta|1\rangle = \alpha'|0\rangle + \beta'|1\rangle,$$

$|\psi'\rangle$ still yields

$$|\alpha'|^2 + |\beta'|^2 = 1, \tag{2.4}$$

Therefore, there is an equivalence between state that relate in that way: they both represent the same quantum state. Can we take this a step further? To answer, consider the transformation

$$|\psi\rangle \mapsto |\psi'\rangle = re^{i\theta}|\psi\rangle. \tag{2.5}$$

$|\psi'\rangle$ no longer respects Equation 2.3. However, to access the original information about the amplitudes (which also corresponds to the probability), it suffices to normalize $|\psi'\rangle$. And this new qubit represents the same quantum states. We don't actually need multiple representation of the same quantum state, therefore we can create an equivalence relation $\sim$ between the states that are related by Equation 2.5, and only consider one representative of each equivalence class, one that satisfies Equation 2.3. Therefore, we will say that any element of $\mathcal{H}_2 := (\mathbb{C}^2 \setminus \{0\})/\sim$ represents a possible qubit state.

2.3 Multiple Qubit System

So far we have talked about single qubits. What if I want to consider a group of qubits as a single system, or, single state. In this section, we will talk about *multiple qubit systems*. Let's start with a 2 qubit system. We want to consider the qubits $|\psi\rangle$ and $|\phi\rangle$ as one system. One may think that to do that we can simply add them together and work with that, however this is wrong. Adding two quantum states has no physical meaning. The system of the 2 qubits is the tensor product of the two. So the system of both $|\psi\rangle$ and $|\phi\rangle$ is

$$|\psi\rangle \otimes |\phi\rangle \in \mathcal{H}_2 \otimes \mathcal{H}_2 =: \mathcal{H}_4.$$

The tensor product is a tool used to join two vector spaces. Here we used it to join $\mathcal{H}_2$ and $\mathcal{H}_2$. The tensor product of a vector spaces of dimension m and n has dimension $m \times n$, therefore $\mathcal{H}_4$ has dimension 4. If $|\psi\rangle$ or $|\phi\rangle$ is in a superposition of definite states we can easily compute the resulting superposition by applying the usual rules of multiplication: associativity and distributivity. Suppose that

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad \text{and} \quad |\phi\rangle = \gamma|0\rangle + \delta|1\rangle \tag{2.6}$$

Then the 2-qubit system consisting of $|\psi\rangle$ and $|\phi\rangle$ can be expressed as:

$$\begin{aligned} |\psi\rangle \otimes |\phi\rangle &= (\alpha|0\rangle + \beta|1\rangle) \otimes (\gamma|0\rangle + \delta|1\rangle) \\ &= (\alpha|0\rangle \otimes \gamma|0\rangle) + (\alpha|0\rangle \otimes \delta|1\rangle) + (\beta|1\rangle \otimes \gamma|0\rangle) + (\beta|1\rangle \otimes \delta|1\rangle) \\ &= \alpha\gamma(|0\rangle \otimes |0\rangle) + \alpha\delta(|0\rangle \otimes |1\rangle) + \beta\gamma(|1\rangle \otimes |0\rangle) + \beta\delta(|1\rangle \otimes |1\rangle) \end{aligned} \tag{2.7}$$

This notation can quickly become heavy, which is why, for clarity, we will usually omit the $\otimes$ and sometimes "merge" the kets:

$$|0\rangle \otimes |1\rangle = |0\rangle\,|1\rangle = |01\rangle$$

We will often use the second and third notation depending on the context and with clearness in mind. Therefore we will write Equation 2.7 as

$$|\psi\rangle \otimes |\phi\rangle = \alpha\gamma\,|00\rangle + \alpha\delta\,|01\rangle + \beta\gamma\,|10\rangle + \beta\delta\,|11\rangle \tag{2.8}$$

We can equivalently use a vector notation to write the state $|\psi\rangle \otimes |\phi\rangle$:

$$|\psi\rangle \otimes |\phi\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix} \otimes \begin{pmatrix}\gamma\\ \delta\end{pmatrix} = \begin{pmatrix}\alpha\gamma\\ \alpha\delta\\ \beta\gamma\\ \beta\delta\end{pmatrix} \tag{2.9}$$

Equation 2.9 clearly shows that $\mathcal{H}_4$ has dimension 4. Therefore, in a two qubit system, any quantum state can be expressed as a linear combination of $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. In other words, while for a single qubit system, the basis states $|0\rangle$ and $|1\rangle$ span the space, the basis states that span the space of 2 qubit systems are $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. Similarly for a 3 qubit system $|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle$, and $|111\rangle$ span the space. When n becomes big enough this notation can also become heavy. So, instead of writing the states in that way, we will sometimes write their base 10 equivalent:

$$\begin{aligned} 00_{(2)} &= 0_{(10)},\\ 01_{(2)} &= 1_{(10)},\\ 10_{(2)} &= 2_{(10)},\\ 11_{(2)} &= 3_{(10)}, \end{aligned}$$

Therefore we can also write Equation 2.8 as:

$$|\psi\rangle \otimes |\phi\rangle = \alpha\gamma\,|0\rangle + \alpha\delta\,|1\rangle + \beta\gamma\,|2\rangle + \beta\delta\,|3\rangle \tag{2.10}$$

But remember that sometimes we need to know the state on each individual qubit so we will more often than not use the base 2 notation.
For an n-qubit system there will be 2^n basis states that span $\mathcal{H}_{2^n} := \mathcal{H}_2^{\otimes n}$, which are simply the binary expansion of $\{|0\rangle, |1\rangle ...., |2^n - 1\rangle\}$:

1-qubit system: $|0\rangle, |1\rangle$

2-qubit system: $|00\rangle, |01\rangle, |10\rangle, |11\rangle \sim |0\rangle, |1\rangle, |2\rangle, |3\rangle$

3-qubit system: $|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle$
$\sim |0\rangle, |1\rangle, |2\rangle, |3\rangle, |4\rangle, |5\rangle, |6\rangle, |7\rangle$

We can differentiate the states $|0\rangle$ and $|1\rangle$ from a n-qubit system and a m-qubit system depending on the context.

2.4 Bra-ket Notation

We will first focus on single qubit systems. In the previous sections we have been using the symbol $|\rangle$ to denote a quantum state. We call the state $|\psi\rangle$ a ket. It can generally be though of as a complex valued 2D vector for a single qubit state, as seen in Equation 2.2. However, we have to be careful, a single ket can represent any qubit system, that is, it can represent multiple qubits. A ket $|\rangle$ is always associated with a bra $\langle|$ which can be though of as a row vector. The bra version of $|\psi\rangle$ is $\langle\psi|$. We can go from one to the other by taking the complex conjugate (*) transpose (T) which we'll denote by the superscript †:

$$\langle\psi| = |\psi\rangle^\dagger = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}^\dagger = \begin{pmatrix}\alpha^* & \beta^*\end{pmatrix}$$

It can also be written in the following way:

$$\langle\psi| = \alpha^* \langle 0| + \beta^* \langle 1|$$

This notation is called the bra-ket notation or the Dirac notation. Let $|\psi\rangle$ and $|\phi\rangle$ be as we defined them in Equation 2.6. We define the inner product on our 2-dimensional vector space as follows:

$$\langle\psi|\phi\rangle = \alpha^*\gamma + \beta^*\delta \tag{2.11}$$

For an n-dimensional space, if we have

$$|a\rangle = \begin{pmatrix}a_1\\ a_2\\ \vdots\\ a_n\end{pmatrix} \quad \text{and} \quad |b\rangle = \begin{pmatrix}b_1\\ b_2\\ \vdots\\ b_n\end{pmatrix}, \tag{2.12}$$

the generalization of this inner product is

$$\langle a|b\rangle = \sum_{i=1}^{n} a_i^* b_i \tag{2.13}$$

Additionally, notice that taking the complex conjugate will give us the reverse inner product $\langle b|a\rangle$. This inner product is easily seen to be equivalent to a matrix multiplication between the row vector $\langle a|$ and the column vector $|b\rangle$.

$$\langle a|b\rangle = \begin{pmatrix}a_1^* & a_2^* & \dots & a_n^*\end{pmatrix} \begin{pmatrix}b_1\\ b_2\\ \vdots\\ b_n\end{pmatrix} = \sum_{i=1}^{n} a_i^* b_i$$

Let's see what happens if we use the basis notation as in Equation 2.6 to compute the inner product:

$$\begin{aligned}\langle\psi|\phi\rangle &= (\alpha^* \langle 0| + \beta^* \langle 1|)(\gamma |0\rangle + \delta |1\rangle)\\ &= \alpha^*\gamma \langle 0|0\rangle + \alpha^*\delta \langle 0|1\rangle + \beta^*\gamma \langle 1|0\rangle + \beta^*\delta \langle 1|1\rangle\end{aligned}$$

Comparing this result and Equation 2.11 and equating the coefficients will give us

$$\langle 0|1\rangle = \langle 1|0\rangle = 0 \quad \text{and} \quad \langle 0|0\rangle = \langle 1|1\rangle = 1 \tag{2.14}$$

Doing the same manipulation for an n-dimensional space will yield

$$\langle i|j\rangle = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad \text{for } i, j = \{1, 2, \cdots, n-1\}, \tag{2.15}$$

where δ_{ij} is the *Kronecker Delta* symbol.
Equation 2.15 is the orthonormality property of the basis states described. We will use it multiple times to prove results.
Notice that the probability amplitudes discussed in section 2.1, which are simply the vector elements, are related to the probability of observing the state in the basis states it is associated with. If we go back to equation 2.1 and use the properties showed in equation 2.14 we have:

$$\alpha = \langle 0|\psi\rangle \quad \text{and} \quad \beta = \langle 1|\psi\rangle \tag{2.16}$$

This can be generalized to any qubit system. That is the probability amplitude of $|\psi\rangle$ associated with the basis state $|j\rangle$, with $0 \leq j \leq n-1$, is $\langle j|\psi\rangle$. There are many other properties and benefits coming from the bra-ket notation but they are not necessary to understand the content of the paper, therefore, we will not go further than that.

2.5 Logic Gates

A logic gate is what permits us to manipulate the nature of our qubits directly. It can be modeled by a function that transforms a qubit into another. The logic gates that depend only on a single qubit are called *Single Qubit Operations* and they can be represented by 2×2 matrices. A logic gate is entirely determined by its action on the basis states $|0\rangle$ and $|1\rangle$. The first and second columns of the matrix are the images of $|0\rangle$ and $|1\rangle$ under the transformation, respectively. For example, it is a very useful tool to be able to switch the amplitudes α and β:

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

This transformation is analogous to the NOT gate in classical computing (which maps $0 \mapsto 1$ and $1 \mapsto 0$), which is why we call this gate the Quantum NOT gate. Its matrix representation (denoted X) is

$$X \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{2.17}$$

since $|0\rangle \mapsto |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, and $|1\rangle \mapsto |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.
The X gate is part of a group of three matrices ($X, Y$and $, Z$) called *the Pauli*

Matrices. They play a key role in Quantum Physics.

$$Y \equiv \begin{pmatrix} 0 & -i \\ -i & 0 \end{pmatrix}, \tag{2.18}$$

$$Z \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.19}$$

Another crucially important single qubit gate is the Hadamard gate which transforms a basis state into a perfect superposition (equal probability of measuring $|0\rangle$ and $|1\rangle$) of the basis states. So

$$|0\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$|1\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

It follows that the matrix representation of the Hadamard gate is

$$H \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.20}$$

We can write the action of the Hadamard gate on a definite qubit in the following way:

$$|k\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + e^{\pi i k}\,|1\rangle), \quad k \in \{0, 1\}, \tag{2.21}$$

since $e^{\pi i k}$ is equal to 1 when $k = 0$ and -1 when $k = 1$.

Now that we have tools to manipulate single qubits, it would be useful to generalize this to multiple qubit systems where we want to change the nature of a qubit depending on the value of another. The simplest example is the CNOT gate which takes two input qubits: the *controlled* qubit and the *target* qubit. If the controlled qubit is 0 then the target qubit is untouched, but if the controlled qubit is 1 then the X gate (NOT gate) acts on the target qubit (it is flipped):

$$|00\rangle \mapsto |00\rangle\,,$$
$$|01\rangle \mapsto |01\rangle\,,$$
$$|10\rangle \mapsto |11\rangle\,,$$
$$|11\rangle \mapsto |10\rangle$$

The states $|00\rangle$ and $|01\rangle$ are unchanged since the first qubit (being the controlled one) is $|0\rangle$ for both. However for the states $|10\rangle$ and $|11\rangle$, the second qubit is flipped (acted on by X) since the controlled qubits are $|1\rangle$. The CNOT gate is then represented by the following 4×4 matrix.

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.22}$$

The full name of the CNOT gate is the "*controlled-NOT*" gate. It is part of a group of gates called the controlled Gates. Controlled gates act on 2 qubit systems and are associated to single qubit gates: if U is a single qubit gate then there exists an associated controlled-U gate. The controlled gate takes two input qubit: one qubit is the *controlled* one, while the other is the *target* one. As we mentioned earlier, we only need to know how our gates transform the basis states to specify them and know everything there is to know about them. So let's see how an arbitrary controlled-U gate acts on the basis states. The rule is that if the controlled qubit is in the state $|0\rangle$ then the second qubit is untouched. However, if the controlled qubit is in the state $|1\rangle$ then the second qubit is acted on by the single qubit operator U. Let's say that the first qubits constitute the controlled ones, while the second ones are the targets.

$$\begin{aligned}
|00\rangle &\mapsto |0\rangle\,|0\rangle\,,\\
|01\rangle &\mapsto |0\rangle\,|1\rangle\,,\\
|10\rangle &\mapsto |1\rangle\,(U\,|0\rangle),\\
|11\rangle &\mapsto |1\rangle\,(U\,|1\rangle)
\end{aligned}$$

This is exactly what we did for the $CNOT$ gate earlier where $U = X$. Note that we are free to chose which qubits are the controlled one and which ones are the targets, we simply have to be consistent.

We saw that any logic gate can be represented by a matrix, but can any matrix represent a quantum logic gate? The answer is no. As we said earlier, the norm of a qubit needs to be one to be in accordance with the probabilistic view of quantum mechanics. Therefore we want this to be unchanged by a logic gate transformation. Notice that Equation 2.3 can be rewritten as:

$$\langle\psi|\psi\rangle = 1 \tag{2.23}$$

Therefore we want this to be unchanged by any logic gate transformation U, that is, we want

$$\begin{aligned}
&(U\,|\psi\rangle)^\dagger (U\,|\psi\rangle) = 1\\
\iff &(\langle\psi|\,U^\dagger)(U\,|\psi\rangle) = 1\\
\iff &\langle\psi|\,U^\dagger U\,|\psi\rangle = 1
\end{aligned} \tag{2.24}$$

Additionally, we want the space to be non-disrupted. This is more complex idea but it can be simply seen as the preservation of the inner product if both states are similarly transformed:

$$\langle\psi|\,U^\dagger U\,|\phi\rangle = \langle\psi|\phi\rangle \tag{2.25}$$

It turns out that equation 2.24 follows if equation 2.25 is true. The preservation of the inner product is precisely the definition of unitary transformations. Therefore, we will exclusively consider logic gate transformations that are unitary transformations: there is a bijection between unitary matrices and logic

gates, ie. any unitary matrix is a plausible logic gate and any logic gate can be represented by a unitary matrix. An equivalent definition of unitary transformations is $U^\dagger U = UU^\dagger = I$, in which case we clearly see that 2.25 is satisfied. Two very important properties of logic gates emerge from this bijection.

1. Logic gates output as many qubits as they take as input
2. Logic gates' transformation are reversible

The second property will be of great use to us, as we shall see in Chapter 5. It is worth it to note that these properties are not shared by classical computing.

3 Quantum Circuits

Logic gates are the building blocks of Quantum Algorithms. We have already discussed some of the most useful ones in section 2.5, and in this chapter we will see how to use them and how to represent them.

Let's first represent the logic gate seen earlier in circuits. We will start with the Pauli matrices X, Y and Z acting an arbitrary state. Equations 2.17, 2.18 and 2.19 can equivalently be represented in the following way

$$\alpha|0\rangle + \beta|1\rangle \quad \boxed{X} \quad \beta|0\rangle + \alpha|1\rangle$$

$$\alpha|0\rangle + \beta|1\rangle \quad \boxed{Y} \quad -i\beta|0\rangle - i\alpha|1\rangle$$

$$\alpha|0\rangle + \beta|1\rangle \quad \boxed{Z} \quad \alpha|0\rangle - \beta|1\rangle$$

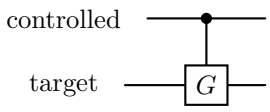
Similarly, the Hadamard gate (equation 2.20) is represented in the following way:

$$\alpha|0\rangle + \beta|1\rangle \quad \boxed{H} \quad \alpha\frac{|0\rangle+|1\rangle}{\sqrt{2}} + \beta\frac{|0\rangle-|1\rangle}{\sqrt{2}}$$

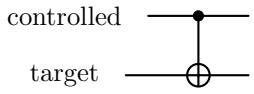
A single line represents a single qubit, which is in accordance with the fact that these gates take single qubits as input. To represent a multiple qubit gate G we would draw multiple lines intersecting with a single box. For example, for a gate that takes 2 qubits as input, we represent it in the following way:

$$\begin{matrix}|\psi\rangle \\ |\phi\rangle\end{matrix} \quad \boxed{G} \quad \Big\} G(|\psi\rangle|\phi\rangle)$$

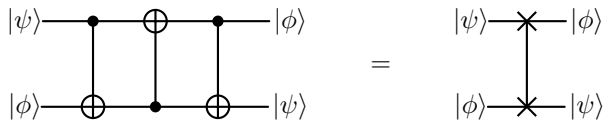
A controlled gate CG will be represented in such a way that we know which qubit is the controlled one and which one is the target.



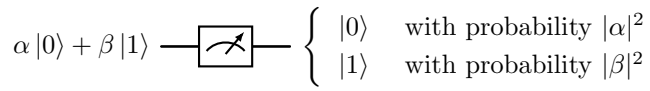
As a shortcut, since the controlled-NOT gate is the most basic and most used controlled gate will simply denote it in this way:



Another useful operation, which we will mention later on, is the swap operation. It allows us to swap the states of qubits. It can be done by solely using consecutive CNOT gates but interchanging the controlled and the target qubit:



The second circuit is a shortcut notation for the swap operation.
Finally, the act of measurement, which will collapse the state to a definite one will be denoted in this way:



4 The Quantum Fourier Transform

The first algorithm that will be presented is the quantum Fourier transform. It permits us to transform qubits with respect to the known *discrete Fourier transform.*

4.1 The Fourier Transform

The Quantum Fourier Transform acting on an arbitrary basis state in a vector space of dimension N is defined to be

$$|j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k/N} |k\rangle , \tag{4.1}$$

Therefore, its action on an arbitrary states $|\psi\rangle$ is

$$|\psi\rangle = \sum_{j=0}^{N-1} x_j |j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} x_j e^{2\pi ijk/N} |k\rangle , \tag{4.2}$$

So as an example, on a single qubit system, where $N = 2$, the basis states are mapped to:

$$|0\rangle \mapsto \frac{1}{\sqrt{2}} \sum_{k=0}^{1} e^{2\pi i(0)k/2} |k\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \tag{4.3}$$

$$|1\rangle \mapsto \frac{1}{\sqrt{2}} \sum_{k=0}^{1} e^{2\pi i(1)k/2} |k\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{4.4}$$

Notice that this is simply the transformation carried by the Hadamard gate. However, it gets more interesting when we transform states in higher dimensions. We are not solely interested in transforming a single qubit, we want to generalize this to an n-qubit state. We have seen in earlier chapters that a definite n-qubit state can be represented by a single ket $|j\rangle$ where j is a number with an n-digit binary expansion. An n-qubit system lives in an 2^n-dimensional space so $N = 2^n$. We define the QFT on basis states of this system in the same way:

$$|j\rangle \to \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi ijk/2^n} |k\rangle \tag{4.5}$$

Here $|k\rangle$ runs from $|k\rangle = |0\rangle$, which corresponds to $|\underbrace{000\ldots00}_{\text{n}}\rangle$, to $|k\rangle = |2^n - 1\rangle$ which corresponds to $|\underbrace{111\ldots11}_{\text{n}}\rangle$. Let's work out the Quantum Fourier Transform for a 2 qubit state to better understand what is happening. We first need to know how the basis states are transformed. Remember that for a 2 qubit system, the basis states are:

$$\begin{aligned}
|0\rangle &\sim |00\rangle \quad (j = 0),\\
|1\rangle &\sim |01\rangle \quad (j = 1),\\
|2\rangle &\sim |10\rangle \quad (j = 2),\\
|3\rangle &\sim |11\rangle \quad (j = 3)
\end{aligned}$$

Now, let's apply the formula (Equation 4.5):

$$
\begin{aligned}
|0\rangle \to & \frac{1}{2}\sum_{k=0}^{3} e^{2\pi i(0)k/4}\,|k\rangle = \frac{1}{2}\sum_{k=0}^{3}|k\rangle = \frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle)\\
|1\rangle \to & \frac{1}{2}\sum_{k=0}^{3} e^{2\pi i(1)k/4}\,|k\rangle = \frac{1}{2}\sum_{k=0}^{3} e^{\pi ik/2}\,|k\rangle = \frac{1}{2}(|0\rangle + i\,|1\rangle - |2\rangle - i\,|3\rangle)\\
|2\rangle \to & \frac{1}{2}\sum_{k=0}^{3} e^{2\pi i(2)k/4}\,|k\rangle = \frac{1}{2}\sum_{k=0}^{3} e^{\pi ik}\,|k\rangle = \frac{1}{2}(|0\rangle - |1\rangle + |2\rangle - |3\rangle)\\
|3\rangle \to & \frac{1}{2}\sum_{k=0}^{3} e^{2\pi i(3)k/4}\,|k\rangle = \frac{1}{2}\sum_{k=0}^{3} e^{3\pi ik/2}\,|k\rangle = \frac{1}{2}(|0\rangle - i\,|1\rangle - |2\rangle + i\,|3\rangle)
\end{aligned}
$$

Therefore the matrix representation of the Quantum Fourier Transform for an 2-qubit system is

$$
F = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}. \tag{4.6}
$$

Now, to prove that performing the Quantum Fourier Transform on a quantum computer is possible, we need to show that the transformation is Unitary.

Proposition 4.1. *The Quantum Fourier Transform is Unitary*

Proof. Let's first denote the quantum Fourier transform (QFT) in Equation (4.1) to be the function $\mathcal{F} : \mathcal{H}_N \to \mathcal{H}_N$ such that,

$$
\mathcal{F}\,|j\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi ijk/N}\,|k\rangle\,. \tag{4.7}
$$

We can represent the quantum Fourier transform by a $N \times N$ matrix that we'll denote F. Since we know how the basis states are mapped using Equation 4.2, we can express F as follows

$$
F = \begin{pmatrix} \vdots & \vdots & & \vdots \\ \mathcal{F}\,|0\rangle & \mathcal{F}\,|1\rangle & \cdots & \mathcal{F}\,|N-1\rangle \\ \vdots & \vdots & & \vdots \end{pmatrix} \tag{4.8}
$$

Here, F_{kj} is the coefficient in front of the ket $|k\rangle$ in $\mathcal{F}|j\rangle$, in other terms,

$$\begin{aligned} F_{kj} &= \langle k| \mathcal{F} |j\rangle \\ &= \langle k| \left(\frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} e^{2\pi i j l/N} |l\rangle \right) \\ &= \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} e^{2\pi i j l/N} \langle k|l\rangle \\ F_{kj} &= \frac{1}{\sqrt{N}} e^{2\pi i j k/N}, \end{aligned}$$

since $\langle k|l\rangle = \delta_{kl}$, where δ_{kl} is the Kronecker Delta function, since we are working on an orthogonal basis.
Proving that the QFT is a unitary transformation is equivalent to proving that its matrix representation is unitary. Therefore, we want to prove that $FF^\dagger = F^\dagger F = I$, where $F^\dagger$ is the conjugate transpose matrix of F and I is the identity matrix. Let's first write down the matrix elements of $F^\dagger$:

$$F^\dagger_{kj} = F^*_{jk} = \frac{1}{\sqrt{N}} e^{-2\pi i k j/N}$$

Now we can compute the matrix elements of $F^\dagger F$ in the following way:

$$\begin{aligned} (F^\dagger F)_{kj} &= \sum_{n=0}^{N-1} F^\dagger_{kn} F_{nj} \\ &= \sum_{n=0}^{N-1} \left(\frac{1}{\sqrt{N}} e^{-2\pi i k n/N} \right) \left(\frac{1}{\sqrt{N}} e^{2\pi i j n/N} \right) \\ &= \frac{1}{N} \sum_{n=0}^{N-1} e^{2\pi i j n/N - 2\pi i k n/N} \\ (F^\dagger F)_{kj} &= \frac{1}{N} \sum_{n=0}^{N-1} e^{2\pi i (j-k) n/N} \end{aligned}$$

If $j = k$, then $e^{2\pi i (j-k) n/N} = e^0 = 1$. It follows that

$$(F^\dagger F)_{kj} = \frac{1}{N} \sum_{n=0}^{N-1} 1 = \frac{N}{N} = 1,$$

therefore the diagonal elements of $F^\dagger F$ are ones. Now for the second case, if

$j \neq k$, we can use the formula for the sum of terms of a geometric series:

$$\begin{aligned}
(F^\dagger F)_{kj} &= \frac{1}{N}\sum_{n=0}^{N-1} e^{2\pi i(j-k)n/N} \\
&= \frac{1}{N}\sum_{n=0}^{N-1} \left(e^{2\pi i(j-k)/N}\right)^n \\
&= \frac{1}{N}\left(\frac{\left(e^{2\pi i(j-k)/N}\right)^N - 1}{e^{2\pi i(j-k)/N} - 1}\right) \\
&= \frac{1}{N}\left(\frac{\left(e^{2\pi i}\right)^{j-k} - 1}{e^{2\pi i(j-k)/N} - 1}\right) \\
&= \frac{1}{N}\left(\frac{(1)^{j-k} - 1}{e^{2\pi i(j-k)/N} - 1}\right) \\
&= \frac{1}{N}\left(\frac{1-1}{e^{2\pi i(j-k)/N} - 1}\right) \\
&= 0.
\end{aligned}$$

It follows that the non diagonal elements of $F^\dagger F$ are zeros. We just proved that $(F^\dagger F)_{kj} = \delta_{kj}$ which is the definition on the identity matrix. Similarly,

$$\begin{aligned}
(FF^\dagger)_{kj} &= \sum_{n=0}^{N-1} F_{kn} F^\dagger_{nj} \\
&= \sum_{n=0}^{N-1} \left(\frac{1}{\sqrt{N}} e^{2\pi ikn/N}\right)\left(\frac{1}{\sqrt{N}} e^{-2\pi ijn/N}\right) \\
(FF^\dagger)_{kj} &= \frac{1}{N}\sum_{n=0}^{N-1} e^{2\pi i(k-j)n/N},
\end{aligned}$$

and the proof becomes essentially the same.
Therefore $FF^\dagger = F^\dagger F = I$, which proves that the QFT in a unitary transformation. □

Since we now know that we can perform the transformation on a quantum computer, the challenge is to figure out a circuit that does it. To do that we need a better formula for the transformation. Equation 4.2 is written as a sum which is not the natural way of writing an n-qubit system. The goal is to write the transformed state $|j\rangle$ as the tensor product of single qubit states (which can and will be in a superposition). To do so, we will take advantage of the binary expansion of j.
Let j be an n bit long number (n digits in its binary expansion). As a reminder

we defined the quantum Fourier transform of $|j\rangle$ as

$$\mathcal{F}\,|j\rangle = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i j k/2^n}\,|k\rangle \tag{4.9}$$

Since k runs from 0 to $2^n - 1$ the binary expansion of k can always be written with n digits. Let $k_1 \dots k_{n-1}k_n$ be the binary expansion of k such that $k = \sum_{l=1}^{n} k_l 2^{n-l}$. Then

$$\frac{k}{2^n} = \frac{\sum_{l=1}^{n} k_l 2^{n-l}}{2^n} = \sum_{l=1}^{n} k_l 2^{-l},$$

and the sum that runs over all possible k's can be rewritten as consecutive sums that all possible values of k_l, namely 0's and 1's. Therefore, Equation 4.9 can be rewritten as

$$\mathcal{F}\,|j\rangle = \frac{1}{2^{n/2}} \sum_{k_1,\dots,k_n=0}^{1} e^{2\pi i j\left(\sum_{l=1}^{n} k_l 2^{-l}\right)}\,|k\rangle$$

Since $e^{a+b} = e^a \times e^b$, we can rewrite the sum in the exponential as a product. Additionally the state $|k\rangle$ is simply $|k_1 \dots k_n\rangle = \bigotimes_{l=1}^{n} |k_l\rangle$. The rest of the algebraic manipulation is simply a matter of factoring and expanding sums.

$$\begin{aligned}
\mathcal{F}\,|j\rangle =& \frac{1}{2^{n/2}} \sum_{k_1,\dots,k_n=0}^{1} \left(\prod_{l=1}^{n} e^{2\pi i j k_l 2^{-l}}\right)\left(\bigotimes_{l=1}^{n} |k_l\rangle\right) \\
=& \frac{1}{2^{n/2}} \sum_{k_1,\dots,k_n=0}^{1} \left(\bigotimes_{l=1}^{n} e^{2\pi i j k_l 2^{-l}}\,|k_l\rangle\right) \\
=& \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left(\sum_{k_l=0}^{1} e^{2\pi i j k_l 2^{-l}}\,|k_l\rangle\right) \\
=& \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left(|0\rangle + e^{2\pi i j 2^{-l}}\,|1\rangle\right) \\
\mathcal{F}\,|j\rangle =& \frac{1}{2^{n/2}} (|0\rangle + e^{2\pi i j/2}\,|1\rangle)(|0\rangle + e^{2\pi i j/2^2}\,|1\rangle)\dots(|0\rangle + e^{2\pi i j/2^n}\,|1\rangle)
\end{aligned} \tag{4.10}$$

Now, it is time to make use of the binary expansion of j: $j = \sum_{m=1}^{n} j_m 2^{n-m} = j_1 \dots j_n$. In Equation 4.10 we divide j by 2^l with l going from 1 to n. When dividing by 2^l, we shift the j_m's into decimal places, similarly to what happens when we divide by 10 in base 10. The integer part of $j/2^l$ will be absorbed into the exponential, which will just result in a multiplication by one, therefore, we only care about the decimal part of $j/2^l$. Dividing by 2 will move j_n to the first decimal place, dividing by 4 will move j_n and j_{n-1} to the second and first decimal places respectively, and it goes on until the whole expansion is shifted.

Therefore we can finally rewrite the Fourier Transform of $|j\rangle$ as a product:

$$\mathcal{F}|j\rangle = \frac{\left(|0\rangle + e^{2\pi i(0.j_n)}|1\rangle\right)\left(|0\rangle + e^{2\pi i(0.j_{n-1}j_n)}|1\rangle\right)\ldots\left(|0\rangle + e^{2\pi i(0.j_1 j_2 \ldots j_n)}|1\rangle\right)}{2^{n/2}}, \tag{4.11}$$

where $0.j_l j_{l+1}\ldots j_m = j_l/2 + j_{l+1}/2^2 + \ldots + j_m/2^{m-l+1}$ is the binary fraction. This product representation is much more easier and natural to work with since we can directly see the state of each individual qubit that was transformed.

4.2 The Circuit for the Quantum Fourier Transform

Using the product representation of the Fourier Transform (Equation 4.11), we can explicitly show what each input qubit will be transformed into. Given $|j\rangle = |j_1 j_2 \ldots j_n\rangle$, we want:

$$\begin{aligned}
|j_1\rangle &\mapsto \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.j_n)}|1\rangle) \\
|j_2\rangle &\mapsto \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.j_{n-1}j_n)}|1\rangle) \\
&\vdots \\
|j_n\rangle &\mapsto \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.j_1 j_2 \ldots j_n)}|1\rangle)
\end{aligned}$$

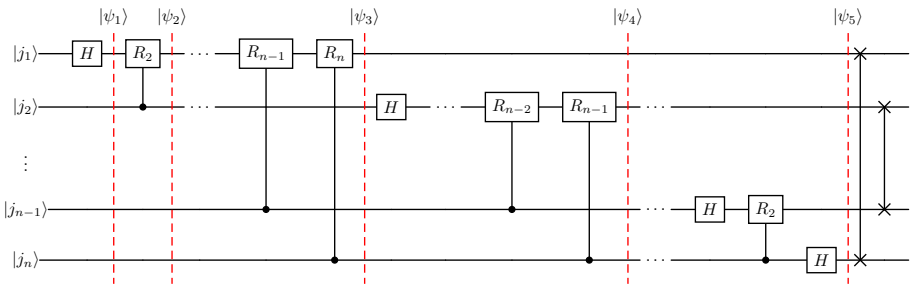
Notice that here that the transformed $|j_k\rangle$'s form the desired Fourier transform of $|j\rangle$. We have simply decomposed the product representation of the Fourier transform. The circuit will use primarily two single qubit logic gates: the Hadamard gate (Equation 2.20) and the R_k gate. The R_k gate is represented by the following unitary matrix:

$$R_k \equiv \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix} \tag{4.12}$$

It will also be useful to write down how the R_k gate acts on the basis states $|0\rangle$ and $|1\rangle$:

$$\begin{aligned}
|0\rangle &\xmapsto{R_k} |0\rangle \\
|1\rangle &\xmapsto{R_k} e^{2\pi i/2^k}|1\rangle
\end{aligned}$$

We can give a motivation as to why we would use those logic gates. The Hadamard gate transforms a definite state into a perfect superposition while the R_k gate add a phase to $|1\rangle$. Notice that each transformed $|j_l\rangle$ is in a perfect superposition with a added phase to $|1\rangle$.
The circuit performing the transform is given below.



Let's go through each step and understand why the circuit indeed works. The first step is applying the Hadamard gate to the first qubit[1]:

$$|j_1\rangle \overset{H}{\longmapsto} \frac{1}{\sqrt{2}}(|0\rangle + e^{\pi i j_1}|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1}|1\rangle) \tag{4.13}$$

Therefore the whole system is in the state

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1}|1\rangle)\,|j_2 \dots j_n\rangle \tag{4.14}$$

$|\psi_2\rangle$ is then obtained by applying the controlled-R_2 gate on $|j_1\rangle$ where $|j_2\rangle$ is the controlled qubit (unchanged). Therefore, if $j_2 = 0$, then the first qubit is unchanged, and if $j_2 = 1$, then we add a phase $e^{2\pi i/2^2}$ to the ket $|1\rangle$ for the first qubit (Equation 4.13). We can generalize this by the following statement

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1}|1\rangle)\,|j_2\rangle \overset{C-R_2}{\longmapsto} \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i j_2/2^2} e^{2\pi i 0.j_1}|1\rangle)\,|j_2\rangle\,,$$

since $e^{2\pi i j_2/2^2}$ equals 1 if $j_2 = 0$ and equals $e^{2\pi i/2^2}$ if $j_2 = 1$, as wanted. With a little algebraic manipulation we can do the following:

$$\begin{aligned}
\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i j_2/2^2} e^{2\pi i 0.j_1}|1\rangle)\,|j_2\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.0 j_2} e^{2\pi i 0.j_1}|1\rangle)\,|j_2\rangle \\
&= \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i (0.j_1 + 0.0 j_2)}|1\rangle)\,|j_2\rangle \\
&= \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i (0.j_1 j_2)}|1\rangle)\,|j_2\rangle
\end{aligned}$$

Therefore

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i (0.j_1 j_2)}|1\rangle)\,|j_2 \dots j_n\rangle \tag{4.15}$$

The next step is to simply repeat this process by applying the controlled-R_3, R_4 until R_n where the controlled qubit is $|j_k\rangle$. Each time, it adds a j_k to the binary fraction in the exponential of the first qubit. At the end we obtain

$$|\psi_3\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i (0.j_1 j_2 j_3 \dots j_n)}|1\rangle)\,|j_2 \dots j_n\rangle \tag{4.16}$$

[1] Refer back to Equation 2.21 if need be to understand the formula used for the Hadamard transformation.

We already begin to see the result. Notice that the first qubit is exactly in the same state as the last qubit in Equation 4.11. The other steps are very similar to what we have already worked out. For $|j_2\rangle$, we first apply the Hadamard gate, then apply a series of controlled-R_k gate with k going from 2 to $n-1$ where the controlled qubit corresponds to $|j_k\rangle$. Therefore, we get

$$|\psi_4\rangle = \frac{1}{2^{2/2}}(|0\rangle + e^{2\pi i(0.j_1\dots j_n)}|1\rangle)(|0\rangle + e^{2\pi i(0.j_2 j_3\dots j_n)}|1\rangle)|j_3\dots j_n\rangle \tag{4.17}$$

We proceed to do this for every qubit each time reducing the number of controlled-R_k. The result is

$$|\psi_5\rangle = \frac{1}{2^{2/2}}(|0\rangle + e^{2\pi i(0.j_1\dots j_n)}|1\rangle)(|0\rangle + e^{2\pi i(0.j_2\dots j_n)}|1\rangle)\dots(|0\rangle + e^{2\pi i(0.j_n)}|1\rangle) \tag{4.18}$$

Notice that, up to ordering, this the same state as in Equation 4.11. The only thing left to do is to swap the states of the qubits to get the desired order. After the swaps operations we finally get

$$\frac{\left(|0\rangle + e^{2\pi i(0.j_n)}|1\rangle\right)\left(|0\rangle + e^{2\pi i(0.j_{n-1}j_n)}|1\rangle\right)\dots\left(|0\rangle + e^{2\pi i(0.j_1 j_2\dots j_n)}|1\rangle\right)}{2^{n/2}}, \tag{4.19}$$

which is the Fourier transform of $|j\rangle$!

In our case, the Fourier transform is not useful in itself but we will soon see how it may be used to solve computational problems. We will not explicitly use the Quantum Fourier Transform but rather the Inverse Quantum Fourier Transform. We know that it exists since, as we said before, any quantum transformation is reversible.

5 Quantum Phase Estimation Algorithm

A very useful algorithm is the quantum phase estimation algorithm which is used in many quantum algorithms. The goal of this algorithm is to estimate the eigenvalue of a given eigenstate of a unitary operator U.
Let $|u\rangle$ be an eigenstate of U with eigenvalue $e^{2\pi i\varphi}$[2]:

$$U|u\rangle = e^{2\pi i\varphi}|u\rangle \tag{5.1}$$

Our goal is to estimate φ.

[2]Since U is unitary, it preserves the norm of the input state. Therefore, the norm of the eigenvalue is 1, which is why we say that the eigenvalue of $|u\rangle$ is of the form $e^{2\pi i\varphi}$.

Quantum Phase Estimation Algorithm	
Input	An operator and its eigenvector : U, $\|u\rangle$
Output	Estimation of the eigenvalue by estimating φ

Table 1: Input and Output table of the Quantum Phase Estimation Algorithm (QPEA)

The strategy to achieve our goal is to transform a given set of multiple qubits into a set of qubits that encodes φ. To do so, we use 2 sets of qubits, that we call registers.

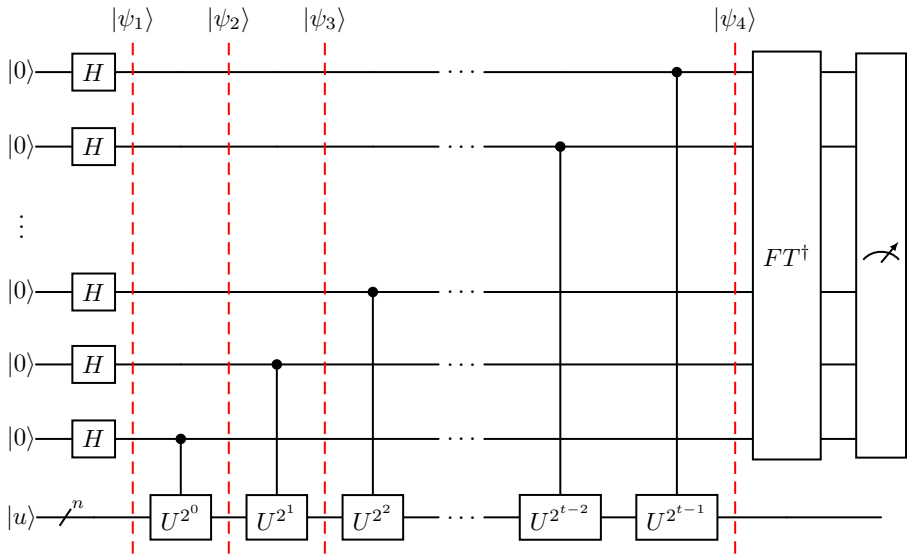
1. The first register contains t qubits, initially in the $|0\rangle$ state. The choice of t depends on how accurately we want to estimate φ.
2. The second register is in the states $|u\rangle$ and contains the number of qubits necessary to store $|u\rangle$ [3].

So our input is a $(t+n)$-qubit state:

$$|\psi_0\rangle = |0\rangle^{\otimes t} |u\rangle , \tag{5.2}$$

where $|0\rangle^{\otimes t} = |\underbrace{00\dots0}_{t}\rangle$.

The Quantum Circuit of the Phase Estimation Algorithm is presented below



Let's go through each step of the algorithm.

[3] If U is a single qubit operator, we only need one qubit in the second register, however, if U takes as input an n qubit system, we need n qubits in the second register

$|\psi_1\rangle$ is obtained by applying the Hadamard gate to each of the qubits in the first register and leaving the second register alone. Therefore

$$
\begin{aligned}
|\psi_1\rangle &= \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)^{\otimes t} |u\rangle \\
|\psi_1\rangle &= \frac{1}{2^{t/2}} (|0\rangle + |1\rangle)^{\otimes t} |u\rangle \\
|\psi_1\rangle &= \frac{1}{2^{t/2}}(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)\cdots(|0\rangle + |1\rangle)\,|u\rangle
\end{aligned} \tag{5.3}
$$

The second step is applying the controlled-U gate to $|u\rangle$ where the last qubit of the first register is the controlled qubit (colored in green in Equation 5.3). The controlled qubit is then in the state[4] $|0\rangle + |1\rangle$. Let's first focus on how the system composed of the controlled qubit and $|u\rangle$ is transformed. We can first decompose it as a superposition of two $n+1$ qubit states:

$$
(|0\rangle + |1\rangle)\,|u\rangle = |0\rangle\,|u\rangle + |1\rangle\,|u\rangle \tag{5.4}
$$

Acting the controlled-U gate on $|0\rangle\,|u\rangle$ results in the identity operator, while acting it on $|1\rangle\,|u\rangle$ results in acting U on the second qubit $|u\rangle$.

$$
\begin{aligned}
|0\rangle\,|u\rangle + |1\rangle\,|u\rangle \xmapsto{Controlled-U} & |0\rangle\,|u\rangle + |1\rangle\,(U\,|u\rangle) \\
&= |0\rangle\,|u\rangle + |1\rangle\,(e^{2\pi i\varphi}\,|u\rangle) \\
&= |0\rangle\,|u\rangle + e^{2\pi i\varphi}\,|1\rangle\,|u\rangle \\
&= (|0\rangle + e^{2\pi i\varphi}\,|1\rangle)\,|u\rangle
\end{aligned}
$$

We use the fact that $|u\rangle$ is an eigenstate of U with eigenvalue $e^{2\pi i\varphi}$ (Equation 5.1). We conclude that

$$
(|0\rangle + |1\rangle)\,|u\rangle \xmapsto{Controlled-U} (|0\rangle + e^{2\pi i\varphi}\,|1\rangle)\,|u\rangle \tag{5.5}
$$

Going back to $|\psi_2\rangle$, we have:

$$
|\psi_2\rangle = \frac{1}{2^{t/2}}(|0\rangle + |1\rangle)\cdots(|0\rangle + |1\rangle)(|0\rangle + e^{2\pi i\varphi}\,|1\rangle)|u\rangle \tag{5.6}
$$

The third step is applying the controlled-U^2 gate to $|u\rangle$ where the second to last qubit of the first register is the controlled qubit (colored in green in Equation 5.6). Applying the controlled-U^2 will yield:

$$
\begin{aligned}
(|0\rangle + |1\rangle)\,|u\rangle \xmapsto{Controlled-U^2} & |0\rangle\,|u\rangle + |1\rangle\,(U^2\,|u\rangle) \\
&= |0\rangle\,|u\rangle + |1\rangle\,(Ue^{2\pi i\varphi}\,|u\rangle) \\
&= |0\rangle\,|u\rangle + |1\rangle\,(e^{2\pi i\varphi}U\,|u\rangle) \\
&= |0\rangle\,|u\rangle + |1\rangle\,(e^{2(2\pi i\varphi)}\,|u\rangle) \\
&= (|0\rangle + e^{2(2\pi i\varphi)}\,|1\rangle)\,|u\rangle
\end{aligned}
$$

[4] We will omit the normalization factor for simplicity

We conclude that

$$|\psi_3\rangle = \frac{1}{2^{t/2}}(|0\rangle + |1\rangle)\cdots(|0\rangle + e^{2(2\pi i\varphi)}|1\rangle)(|0\rangle + e^{2\pi i\varphi}|1\rangle)|u\rangle \tag{5.7}$$

We simply have to repeat this step t times as shown in the circuit, changing the controlled qubit each time and applying the controlled-U^{2^j} gate to $|u\rangle$. Let's consider the general case where we apply the controlled-U^{2^j} on the target qubit $|u\rangle$ and controlled qubit $|0\rangle + |1\rangle$:

$$\begin{aligned}(|0\rangle + |1\rangle)|u\rangle \xmapsto{Controlled-U^{2^j}} &|0\rangle|u\rangle + |1\rangle(U^{2^j}|u\rangle) \\ =&|0\rangle|u\rangle + |1\rangle(e^{2^j(2\pi i\varphi)}|u\rangle) \\ =&(|0\rangle + e^{2^j(2\pi i\varphi)}|1\rangle)|u\rangle \end{aligned} \tag{5.8}$$

So we can now write down the $|\psi_4\rangle$:

$$|\psi_4\rangle = \frac{1}{2^{t/2}}(|0\rangle + e^{2^{t-1}(2\pi i\varphi)}|1\rangle)(|0\rangle + e^{2^{t-2}(2\pi i\varphi)}|1\rangle)\cdots(|0\rangle + e^{2^0(2\pi i\varphi)}|1\rangle)|u\rangle \tag{5.9}$$

Now let's take a step back and see what is happening. We can assume WLOG that $0 < \varphi < 1$ since the function $e^{2\pi ix}$ is 1-periodic. Therefore, we can express φ by its binary fraction $\varphi = 0.\varphi_1 \ldots \varphi_t \ldots$. Note that the expansion may be infinite. Similarly to what we did for the Fourier transform, when we derived the product representation of the transform, when φ is multiplied by 2^k , the expansion gets shifted k places to the left and we can disregard the integer part. Therefore we can rewrite Equation 5.9 as

$$|\psi_4\rangle = \frac{1}{2^{t/2}}(|0\rangle + e^{2\pi i0.\varphi_t}|1\rangle)(|0\rangle + e^{2\pi i0.\varphi_{t-1}\varphi_t}|1\rangle)\cdots(|0\rangle + e^{2\pi i0.\varphi_1\ldots\varphi_t}|1\rangle)|u\rangle \tag{5.10}$$

If we look close enough, this is exactly what we would get if we were to Fourier transform $|\varphi_1\varphi_2\ldots\varphi_t\rangle$. Applying the inverse Fourier transform and measuring each of the states in the first register will give us $\tilde{\varphi} := \varphi_1\varphi_2\ldots\varphi_t$. Now $\tilde{\varphi}$ is an approximation of φ when the binary fraction expansion of φ is more than t bits. This explains why we call this algorithm the phase *estimation* algorithm, and why earlier, we said that the value t depends on how accurately we want to estimate φ. The bigger t, the better the approximation.

6 Quantum Order Finding

The order finding algorithm is an application of the Phase Estimation Algorithm and will also be useful later on. Let's understand what it does and how it does it. First, let's define the order.

Definition 6.1. *Given two positive integers x and N, with $x < N$ and $\gcd(x, N) = 1$, we say that the **order** of x is the smallest integer r satisfying*

$$x^r = 1 \bmod N \tag{6.1}$$

The goal of the algorithm is to find the order of x modulo N given x and N.

Quantum Order Finding Algorithm	
Input	Two co-prime integer : x, N, with $x < N$
Output	Estimation of the order r

Table 2: Input and Output table of the Quantum Order Finding Algorithm (QOFA)

For the rest of the chapter we will consider an x and N that satisfy the conditions. The strategy that we'll adopt is finding a unitary operator whose eigenvalue (for a certain eigenstate) inscribes r, and apply the phase estimation algorithm.
We will call that operator U. It will act on L-qubit systems where $L := \lceil \log N \rceil$ ie. L is the number of digits needed to write N in base 2. With this choice, the input state $|y\rangle$ satisfies $0 \leq y \leq 2^L - 1$.
For $N \leq y \leq 2^L - 1$, we define U as the identity operator. For $0 \leq y < N$, we define U as:

$$U|y\rangle = |xy \bmod N\rangle \tag{6.2}$$

Let's prove that U to be a plausible logic gate.

Proposition 6.1. *U is unitary*

Proof. To prove that U is unitary we will focus on its matrix representation. We know from linear algebra that, given two square matrices A and B, if $AB = I$, then $BA = I$. Since U is a $2^L \times 2^L$ matrix, it suffices to show that $U^\dagger U = I$ to prove that U is unitary.
Since U is the identity operator for $N \leq y \leq 2^L - 1$, we only care for how $|y\rangle$ is transformed when $0 \leq y < N$. Like we did earlier to prove that the Fourier transform in unitary, we have that, for $1 \leq j, k \leq N$:

$$\begin{aligned}
(U^\dagger U)_{kj} &= \langle k| (U^\dagger U) |j\rangle \\
&= (\langle k| U^\dagger)(U|j\rangle) \\
&= (U|k\rangle)^\dagger (|xj \bmod N\rangle) \\
&= (|xk \bmod N\rangle)^\dagger (|xj \bmod N\rangle) \\
&= \langle xk \bmod N | xj \bmod N\rangle \\
&= \delta_{(xk \bmod N)(xj \bmod N)}
\end{aligned}$$

Since x and N are relatively prime, x has an inverse modulo N. We'll denote it x^{-1}. It follows that

$$xk \equiv xj \bmod N \iff x^{-1}xk \equiv x^{-1}xj \bmod N \iff k \equiv j \bmod N$$

Since $1 \leq j, k \leq N$ then the statement is equivalent to $k = j$. Therefore

$$(U^\dagger U)_{kj} = \delta_{kj}$$

Therefore U is unitary. □

Now that we have a candidate for our operator, it is time to find eigenvectors that encode r. Fortunately for us, we again have a candidate (and even more than one). Let us consider the following states:

$$|u_s\rangle := \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(\frac{-2\pi isk}{r}\right) |x^k \bmod N\rangle, \quad 0 \leq s \leq r-1, \tag{6.3}$$

Proposition 6.2. *For any s satisfying $0 \leq s \leq r-1$, $|u_s\rangle$ is an eigenstate of U with eigenvalue $e^{2\pi is/r}$:*

$$U|u_s\rangle = \exp\left(\frac{2\pi is}{r}\right) |u_s\rangle \tag{6.4}$$

Proof. This will be a long derivation and can be skipped by the reader. I chose to still include it because it mixes different techniques that have often come up in the proofs in this paper.
We will start by simply transforming $|u_s\rangle$ by U using the definitions. For the sake of readability we will omit the mod$N$ from the kets but note that the values in the kets are modulo N.

$$\begin{aligned} U|u_s\rangle &= U\left(\frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(\frac{-2\pi isk}{r}\right) |x^k\rangle\right) \\ &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(\frac{-2\pi isk}{r}\right) U|x^k\rangle \\ &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(\frac{-2\pi isk}{r}\right) |x^{k+1}\rangle \end{aligned}$$

Before doing the proof we already know what the eigenvalue will be $e^{2\pi is/r}$ so

we will multiply by $e^{2\pi is/r}e^{-2\pi is/r} = 1$ and see what happens.

$$\begin{aligned}
U\left|u_s\right\rangle &= \frac{1}{\sqrt{r}}\exp\left(\frac{2\pi is}{r}\right)\exp\left(\frac{-2\pi is}{r}\right)\sum_{k=0}^{r-1}\exp\left(\frac{-2\pi isk}{r}\right)\left|x^{k+1}\right\rangle \\
&= \frac{1}{\sqrt{r}}\exp\left(\frac{2\pi is}{r}\right)\sum_{k=0}^{r-1}\exp\left(\frac{-2\pi is}{r}\right)\exp\left(\frac{-2\pi isk}{r}\right)\left|x^{k+1}\right\rangle \\
&= \frac{1}{\sqrt{r}}\exp\left(\frac{2\pi is}{r}\right)\sum_{k=0}^{r-1}\exp\left(\frac{-2\pi is(k+1)}{r}\right)\left|x^{k+1}\right\rangle \\
&= \frac{1}{\sqrt{r}}\exp\left(\frac{2\pi is}{r}\right)\sum_{k=1}^{r}\exp\left(\frac{-2\pi isk}{r}\right)\left|x^{k}\right\rangle \qquad (6.5)
\end{aligned}$$

The result is similar to the expression of $|u_s\rangle$ given in Equation 6.3. The only differences are the value over which k ranges. For $|u_s\rangle$, k ranges from 0 to $r-1$, however, in Equation 6.5, k ranges from 1 to r. The trick here will be to start with $k = 0$ and subtract the $k = 0$ terms to conserve the equality, and, additionally subtract the $k = r$ term to stop the iteration at $k = r - 1$.

$$\begin{aligned}
U\left|u_s\right\rangle &= \frac{1}{\sqrt{r}}\exp\left(\frac{2\pi is}{r}\right)\left(\left(\sum_{k=1}^{r-1}\exp\left(\frac{-2\pi isk}{r}\right)\left|x^{k}\right\rangle\right) + \exp\left(-2\pi is\right)\left|x^{r}\right\rangle\right) \\
&= \frac{1}{\sqrt{r}}\exp\left(\frac{2\pi is}{r}\right)\left(\left(\sum_{k=1}^{r-1}\exp\left(\frac{-2\pi isk}{r}\right)\left|x^{k}\right\rangle\right) + \left|1\right\rangle\right) \\
&= \frac{1}{\sqrt{r}}\exp\left(\frac{2\pi is}{r}\right)\left(\left(\sum_{k=0}^{r-1}\exp\left(\frac{-2\pi isk}{r}\right)\left|x^{k}\right\rangle\right) + \left|1\right\rangle - e^{0}\left|x^{0}\right\rangle\right) \\
&= \frac{1}{\sqrt{r}}\exp\left(\frac{2\pi is}{r}\right)\left(\left(\sum_{k=0}^{r-1}\exp\left(\frac{-2\pi isk}{r}\right)\left|x^{k}\right\rangle\right) + \left|1\right\rangle - \left|1\right\rangle\right) \\
&= \frac{1}{\sqrt{r}}\exp\left(\frac{2\pi is}{r}\right)\left(\left(\sum_{k=0}^{r-1}\exp\left(\frac{-2\pi isk}{r}\right)\left|x^{k}\right\rangle\right)\right) \\
U\left|u_s\right\rangle &= \exp\left(\frac{2\pi is}{r}\right)\left|u_s\right\rangle
\end{aligned}$$

Here we have used the fact that r is the order of x modulo N therefore $|x^r\rangle = |x^r \bmod N\rangle = |1\rangle$. This proves that $|u_s\rangle$ is indeed an eigenstate of U with eigenvalue $e^{2\pi is/r}$. □

We have just found r eigenstates of U with eigenvalues that encode r. However, there is a small problem: You might have noticed that, in order to apply the phase estimation algorithm to U and $|u_s\rangle$, we need to be able to construct $|u_s\rangle$ for the second register. However, our eigenstates depend explicitly on r and r is our unknown. How are we supposed to construct $|u_s\rangle$ without knowing r? This is where a very interesting and clever trick comes in.

Proposition 6.3. *The sum of all eigenstates of U satisfying Equation 6.4, normalized, is the basis state $|1\rangle$:*

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle = |1\rangle \tag{6.6}$$

Proof. Let's explicitly compute the sum using the definition of $|u_s\rangle$ (Equation 6.3). For this proof as well we will omit the $\bmod N$ from the kets.

$$\begin{aligned}
\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle &= \frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}\left(\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}\exp\left(\frac{-2\pi isk}{r}\right)|x^k\rangle\right)\\
&= \frac{1}{r}\sum_{s=0}^{r-1}\sum_{k=0}^{r-1}\exp\left(\frac{-2\pi isk}{r}\right)|x^k\rangle\\
&= \frac{1}{r}\sum_{k=0}^{r-1}\left(\sum_{s=0}^{r-1}\exp\left(\frac{-2\pi isk}{r}\right)\right)|x^k\rangle\\
&= \frac{1}{r}\sum_{k=0}^{r-1}\left(\sum_{s=0}^{r-1}\exp\left(\frac{-2\pi ik}{r}\right)^s\right)|x^k\rangle
\end{aligned}$$

Now

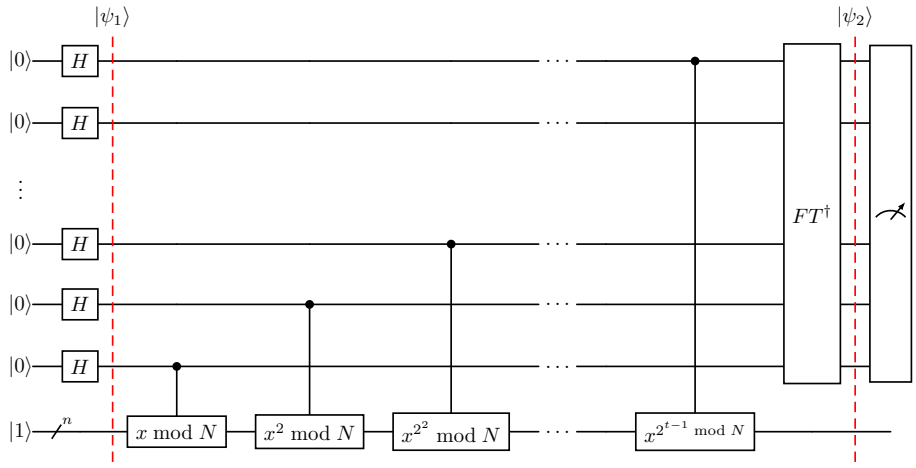
$$\sum_{s=0}^{r-1}\exp\left(\frac{-2\pi ik}{r}\right)^s$$

is simply the sum of the r first terms of a geometric series. The formula for geometric sums holds for $\frac{-2\pi ik}{r} \neq 0$, since we want the exponential to not be 1. Notice that k varies from 0 to $r-1$. Then, the condition is satisfied for all k's except for $k = 0$. So let's separate these cases and evaluate them separately:

$$\begin{aligned}
\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle &= \frac{1}{r}\sum_{k=0}^{r-1}\left(\left(\sum_{s=0}^{r-1}\exp\left(\frac{-2\pi ik}{r}\right)^s\right)|x^k\rangle\right)\\
&= \frac{1}{r}\left(\sum_{s=0}^{r-1}(\exp 0)^s\,|x^0\rangle + \sum_{k=1}^{r-1}\sum_{s=0}^{r-1}\exp\left(\frac{-2\pi ik}{r}\right)^s|x^k\rangle\right)\\
&= \frac{1}{r}\left(\sum_{s=0}^{r-1}|1\rangle + \sum_{k=1}^{r-1}\frac{\exp\left(\frac{-2\pi ik}{r}\right)^r - 1}{\exp\left(\frac{-2\pi ik}{r}\right) - 1}|x^k\rangle\right)\\
&= \frac{1}{r}\left(\sum_{s=0}^{r-1}|1\rangle + \sum_{k=1}^{r-1}0\right)\\
&= \frac{1}{r}r\,|1\rangle\\
\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle &= |1\rangle
\end{aligned}$$

□

Proposition 6.3 is incredibly important in the order finding algorithm. It permits us to input multiple eigenstates of our operator without explicitly knowing r. The trick here is to make the second register in the state $|1\rangle$, which is easy to do. Let's now see what will happen. First, we now can visualize what the algorithm will look like by simply applying the phase estimation algorithm:



Similarly to the Phase Estimation Algorithm:

$$|\psi_1\rangle = \frac{1}{2^{t/2}}(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)\cdots(|0\rangle + |1\rangle)\,|1\rangle \tag{6.7}$$

Now for a general analysis, let's see what happens when we act the controlled-U^{2^j} gate on the qubits of interest, but first, notice that we can rewrite them using Proposition 6.3:

$$\begin{aligned}(|0\rangle + |1\rangle)\,|1\rangle &= (|0\rangle + |1\rangle)\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle = |1\rangle \\ &= \frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}(|0\rangle + |1\rangle)\,|u_s\rangle\end{aligned}$$

Therefore, applying the controlled-U^{2^j} will result in a superposition of the states we described in the phase estimation algorithm:

$$C - U^{2^j}(|0\rangle + |1\rangle)\,|1\rangle \xmapsto{C-U^{2^j}} \frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}(|0\rangle + e^{2\pi i 2^j\left(\frac{s}{r}\right)}|1\rangle)\,|u_s\rangle$$

Notice that apart from the sum and the normalization factor, this is exactly what we had during the phase estimation algorithm, before the inverse Fourier transform. Therefore applying the inverse Fourier transform will result in a

perfect superposition of approximations of $\frac{s}{r}$:

$$|\psi_2\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\widetilde{s/r}\rangle \, |u_s\rangle \, , \tag{6.8}$$

where $\widetilde{s/r}$ is an approximation of s/r. Therefore when we measure the first register we will obtain $\widetilde{s/r}$ for some s between 0 and $r-1$. From $\widetilde{s/r}$, if s and r are co-prime, we can find r by applying the continued fraction algorithm[5]. The probability that s and r are co-prime is at least $\frac{1}{2}\log r > \frac{1}{2}\log N$. Therefore, if the continued fraction algorithm fails (which means that s and r have a common factor), we can simply repeat the algorithm $2\log N$ times, and with high probability, measure a s co-prime to r, in which case, the continued fraction algorithm succeeds and we have found our order.

7 Shor's Factoring Algorithm

We finally have arrived to a very interesting application of quantum computing: Shor's factoring algorithm. As the name indicates it, we want to find a non-trivial factor of a positive composite number N. To understand clearly what it entails, let's talk about number theory.

7.1 Arithmetic Background

One of the most famous result of number theory is the **Fundamental Theorem of Arithmetic**.

Theorem 7.1. ***(Fundamental Theorem of Arithmetic)*** *Any positive non-zero integer a can be expressed as the product of prime number (we include the empty product which is 1):*

$$a = p_1^{a_1} p_2^{a_2} ... p_n^{a_n}, \tag{7.1}$$

where $p_1, ..., p_n$ are distinct prime numbers, and $a_1, ..., a_n$ are positive integers. This factorization is unique.

We will not prove this theorem as I don't believe that it is necessary to achieve the goal of this paper.
The ultimate goal of the factoring algorithm is to find the prime factorization (Equation 7.1) of any given integer N. Obviously, we are not interested in factoring prime numbers since their prime factorization is themselves which is why we will turn our attention to **composite numbers**.

Definition 7.1. *We call $N \in \mathbb{N}$ a positive* ***composite number*** *if it can be written as a product of two natural numbers p and q different than 0 and 1. In*

[5] Not discussed in this paper but there are many books that explain it namely reference [2]

other words N is a composite number if it is non-zero, not 1 and not a prime number.

$$N = p \times q,$$

where, $p, q \notin \{0, 1\}$.

The primary goal of the Factoring Algorithm is to simply find one non-trivial factor of a given composite integer N. This factor doesn't need to be a prime number. Now given a factor q of N, we can divide N by q to find p repeat the algorithm on p and q if they are not prime. Doing this as much as needed until you have a collection of prime numbers will give you the prime factorization of N. So let's focus on how to simply find one factor of N.
In multiple cases we can easily factor it on a classical computer. Therefore, let's first go through those cases and then tackle the "non-classical" cases.

7.2 The Classical Cases of factoring

7.2.1 N is Even

First, the most trivial case is when N is even. If that is the case, the algorithm should simply return 2. This can be done on a classical computer by checking the last digit of the binary expansion of N. If it is 1 then N is odd and if it is 0 then N is even. The same is valid on a quantum computer. We simply measure the last qubit and if it is in the state $|1\rangle$ then N is odd and it is in the state $|0\rangle$ then N is even.

7.2.2 N is a Perfect Power

The second case is less trivial but doable on a classical computer. The claim is that there exists (and we will describe it) an efficient classical algorithm that determines whether $N = a^b$ for some integers $a \geq 1$ and $b \geq 2$. We define L as the number of digits in the binary expansion of N.
The algorithm is the following:

Algorithm 1: Finding if N is a perfect power

Data: Composite Number N
Result: Returns a if N can be written as $N = a^b$ (for some $a \geq 1$ and $b \geq 2$), otherwise returns **False**.

```
y ← log_2 N
for b ∈ {2, ..., L} do
    x ← y/b
    u_1 ← ⌊2^x⌋
    u_2 ← ⌈2^x⌉
    if u_1^b = N then
        return u_1
    if u_2^b = N then
        return u_2
return False
```

Why does this algorithm work? First, if $N = a^b$ then $b \leq L$, so we can check for each $b \in \{2, \ldots, L\}$ if there exists an a such that $N = a^b$. If we do not find such an a then it follows that N is not a perfect power. Therefore, to check for all possible values of b we do a loop that runs each case.

Proposition 7.1. *If $N = a^b$ then $b \leq L$.*

Proof. Since we don't consider the case $N = 1$,it follows that $a > 1$. Then 2^L is $L + 1$ bits long and N is L bits long, so

$$\begin{aligned} N < 2^L &\implies a^b < 2^L \\ &\implies b \log_2 a < L \log_2 2 \\ &\implies b \log_2 a < L \end{aligned}$$

Now since $a \geq 2 \implies \log_2 a \geq 1 \implies b \leq b \log_2 a \implies b \leq L$. □

We can argue and prove that in the case that N is a perfect power, the computation in the algorithm will output a. Notice that

$$2^x = 2^{y/b} = 2^{\frac{\log_2 N}{b}} = N^{1/b}$$

Therefore, when we find the closest integers to 2^x (u_1 and u_2) we find our possible values of $N^{1/b}$. Raising them to the power of b will give us N if indeed N is a perfect power for that b and we will have found our candidate for a. However, if N is not a perfect power then considering the closest integers near $N^{1/b}$ and then raising it to the power of b will move us away from N. Therefore, by Proposition 7.1, if this doesn't work for any $2 \leq b \leq L$ then N is not a perfect power and the algorithm returns **False**.

7.2.3 The Magic of Randomness

The last classical chance we have at finding a factor of N is of the randomness type. The trick here is to simply choose a random integer x that is in between 2

and N: $2 \leq x \leq N$. Then we compute the $\gcd(x, N)$. If $\gcd(x, N) \neq 1$ then we have found our factor! (miracle). If $\gcd(x, N) = 1$ then x and N are co-prime and we have the perfect setup for the order finding algorithm. We shall see in a moment how it helps us in our quest to find factors.

7.3 Reduction of Order Finding to Factoring

We have seen that applying the 3 previous step in our algorithm will ensure that N is an odd number that has multiple prime numbers (raised to different powers) in its prime number factorization. We also have an integer $2 \leq x < N$ which is co-prime to N. This is the perfect setting to execute the order finding algorithm. Therefore, let's say that we have the order r of x modulo N. Consider the following theorems.

Theorem 7.2. *Suppose N is a composite number L bits long, and y is a non-trivial solution to the equation*

$$y^2 = 1(\mathrm{mod} N) \tag{7.2}$$

in the range $2 \leq y \leq N$ (neither $y = 1(\mathrm{mod} N)$ nor $y = N - 1 = -1(\mathrm{mod} N)$). Then at least one of $\gcd(y - 1, N)$ and $\gcd(y + 1, N)$ is a non-trivial factor of N.

Proof. In this theorem we assume that x solves (non trivially) $x^2 = 1(\mathrm{mod} N)$ and is in the range $1 \leq x \leq N$. The non-trivial solution and the range conditions can be merge into the condition $1 < x < N - 1$. Notice that

$$\begin{aligned} x^2 = 1(\mathrm{mod} N) &\implies x^2 - 1 = 0(\mathrm{mod} N) \\ &\implies (x + 1)(x - 1) = 0(\mathrm{mod} N) \end{aligned}$$

Therefore at lease one of $x+1$ and $x-1$ has a common factor with N (different than 1). Since $1 < x < N-1$ then $x-1 < x+1 < N$ therefore the common factor cannot be N. We conclude that at least one of $\gcd(x - 1, N)$ and $\gcd(x + 1, N)$ is a non-trivial factor of N. □

Theorem 7.3. *Suppose $N = p_1^{a_1} p_2^{a_2} ... p_m^{a_m}$ is the prime factorization of an odd composite positive integer. Let x be an integer chosen uniformly at random, subject to the requirement that $1 \leq x \leq N - 1$ and x is co-prime to N. Let r be the order of x modulo N. Then*

$$P(r \text{ is even and } x^{r/2} \neq -1(\mathrm{mod} N)) \geq 1 - \frac{1}{2^m}$$

Let's see how the factoring algorithm follows directly from these two theorems. The first theorem states that, if you have a non-trivial solution to Equation 7.2, then you have found a factor of N. How can we find such a solution? A similar equation we have access to is from the definition of the order. We know that r satisfies

$$x^r = 1(\mathrm{mod} N). \tag{7.3}$$

The second theorem states that there is a high probability that the order r is even and that $x^{r/2} \neq -1 (\mod N)$. Let's assume that these statements are true. We can rewrite 7.3 as

$$(x^{r/2})^2 = 1 (\mod N) \tag{7.4}$$

Let's set $y = x^{r/2}$. Then y is a solution of equation 7.2 such that $y \neq -1 (\mod N)$. To show that y is a non-trivial solution it suffices to show that $y \neq 1 (\mod N)$. Recall the definition of the order: r is the smallest integer that satisfies $x^r = 1 (\mod N)$. Since $r/2 < r$ it follows that $y = x^{r/2}$ cannot be congruent to 1 modulo N. Therefore y is a non-trivial solution of equation 7.2. It follows that there is a high probability that at least one of

$$\gcd(x^{r/2} + 1, N) \quad \text{and} \quad \gcd(x^{r/2} + 1, N)$$

is a non-trivial factor of N!
Notice that in this scenario we have that N is an odd composite number with at least 2 distinct prime numbers in its prime factorization. Therefore, it follows from Theorem 7.3 that the probability that we find a non-trivial factor of N is at least $\frac{3}{4}$, which is not bad. Therefore, if the algorithm fails, it suffices to repeat the algorithm enough times to beat the odds.

8 Conclusion

The principle of superposition in Quantum Physics opened the door for a whole new field of computation: Quantum Computing. Quantum computers are susceptible of outperforming classical computers on many problems. In this paper we focused on the factoring problem which was solved by Peter Shor in 1994 [4] using quantum computing. At the time of writing, it is widely believed that there does not exist a polynomial time classical algorithm for factoring and the commonly used RSA cryptosystem is based on that fact. However, the number of qubits needed to actually perform the factoring algorithm on big numbers (the ones typically used in RSA encryption) is expressed in millions. Research is currently being conducted to reduce that number and to find new encryption methods unbreakable by quantum computers. Some scholars believe that quantum encryption is the way forward, while others believe that it is still possible to find new classical encryption methods. Only time will tell if new encryption methods see the light of day before the RSA cryptosystem becomes obsolete.

References

[1] D. C. Marinescu and G. M. Marinescu. Chapter 1 - preliminaries. In D. C. Marinescu and G. M. Marinescu, editors, *Classical and Quantum Information*, pages 1–131. Academic Press, Boston, 2012.

[2] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[3] D. Ntalaperas, A. Kalogeropoulos, and N. Konofaos. An algorithm based on quantum phase estimation for the identification of patterns. *Quant. Inf. Proc.*, 23(5):194, 2024.

[4] P. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.