



SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Quantum Computation and Shor's Algorithm

av

Per Idenfeldt

2020 - No K41

Quantum Computation and Shor's Algorithm

Per Idenfeldt

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Olof Sisask

2020

Acknowledgments

I would like to thank Olof Sisask for his patience, attention to detail, and for always finding time to answer my questions in a pedagogic way.

Thank you to Pavel Kurasov for his constructive criticism and for pointing to the faults in the thesis.

Of course, thank you to my family for supporting me unconditionally.

Abstract

Quantum computation is a computing system that makes use of quantum mechanical phenomena to perform computation. A computer performing such computation is referred to as a quantum computer. Some computational problems, in particular integer factorization, are believed to be solved significantly faster on quantum computers.

The RSA cryptosystem makes use of the fact that integer factorization is considered hard for a classical computer. On a quantum computer, the integer factorization problem may be solved easily with Shor's algorithm. The central goal in this thesis is to understand the details of Shor's integer factorization algorithm. To accomplish this we provide a brief introduction to the field of quantum computation.

Contents

0.1	Introduction	2
1	Preliminaries	4
1.1	Postulates of quantum mechanics	4
1.1.1	Single qubits	5
1.1.2	Multiple qubits	7
1.1.3	Quantum gates	9
1.1.4	Measurement	10
1.2	Reformulation of the standard postulates of quantum mechanics	11
1.2.1	Partial measurements	11
2	Quantum Computation	15
2.1	The quantum circuit model	15
2.1.1	Single qubit quantum gates	15
2.1.2	Two qubit quantum gates	17
2.1.3	Graphical representation of quantum circuits	18
2.1.4	Universality	19
2.2	Reversible computation	22
2.2.1	A first iteration	23
2.2.2	Composing subcircuits	25
2.2.3	Quantum circuit complexity	27
3	The Finite Abelian Hidden Subgroup Problem	28
3.1	Algebraic representation of groups	28
3.2	Fourier basis and quantum Fourier transform	31
3.3	The Fundamental Theorem of Finite Abelian Groups	34
3.4	Solving the Finite Abelian Hidden Subgroup Problem	36
4	Quantum Algorithms	39
4.1	Simon's problem	39
4.2	Shor's algorithm	42
4.2.1	Pitfalls of order finding	51
4.2.2	Time complexity	52

0.1 Introduction

In 1994 Peter Shor developed the polynomial-time quantum algorithm for integer factorization that we today know as Shor's algorithm. This algorithm is remarkable in a number of ways. Perhaps most interesting, as of writing it remains unproven whether a polynomial-time classical analogue exists. That is, it's possible that there are natural computational problems for which a quantum computer is inherently faster than a classical one. Although exponential quantum speed up of a classical algorithm had previously been demonstrated with Simon's algorithm for Simon's problem, it was more difficult to envision an exact application of this discovery.

There is however no doubt in anyone's mind regarding the possible application of polynomial-time integer factorization. In fact, the assumption that there is no classical polynomial-time algorithm to the integer factorization problem is the very bedrock of the RSA cryptosystem. As of writing this, the RSA cryptosystem still has many uses. It is for example a very common cipher suite used in TLS protocols for establishing secure internet connections.

Recently, it was estimated that by optimizing modular exponentiation in Shor's algorithm, a 2048 bit RSA integer could be factorized in 8 hours with 20 million qubits[1]. This is the most commonly used RSA moduli size in use today. The number field sieve algorithm, which is regarded as the fastest classical integer factorization algorithm, took 2000 years of computing on a single core 2.2GHz AMD Opteron to factor a 768-bit RSA integer in 2009[2].

Post-quantum RSA

Even with all this in mind, there may still be a future for RSA. For one thing, there currently are no quantum computers with 20 million qubits. For context, IBM announced in October 2019 their most powerful quantum computer so far - a staggering 53 qubits[3]. Even then, with the appropriate amount of qubits, RSA may still be feasible. According to [3], it's estimated (preliminary) that trying to factor the product of two 4096-bit primes with Shor's algorithm would take 2^{100} operations. Although the process of using such primes in encryption took around 100 hours, it is perhaps still an alternative for high-security information.

Prerequisites and final remarks

The motivated reader needs no more than a strong grasp of the fundamental concepts in linear algebra. It is however recommended for the reader to be familiar with some analysis and group theory.

In the end, regardless of possible application, the field of quantum computation is a truly interesting one. Lying somewhere in between mathematics, computer science and physics, it contains beautiful and clever ideas from all three fields. Being very much a beginner himself, the author hopes he can illustrate some of the basic ideas of quantum computation as well explain some of the clever

algorithms that emerged from these ideas. The goal is to do this in a sufficient way as to understand the details of Shor's integer factorization algorithm.

Chapter 1

Preliminaries

In order to gain a good understanding of quantum algorithms we must first look to the building blocks which compose them. This chapter will explain the concept of a qubit, quantum measurement, quantum computation along with the technical background needed to formulate them.

We begin with some notation. Dirac's bra/ket notation is used throughout quantum physics to represent the *state* of a quantum system. A vector is represented by a symbol written inside a *ket*, such as $|v\rangle$.

That is, instead of writing \vec{v} for a vector we write $|v\rangle$. The vector representation of a *bra* $\langle v|$ is obtained from taking the *Hermitian conjugate* of $|v\rangle$

Example 1.0.1. Let $|v\rangle = \begin{bmatrix} 1+i \\ 2-i \end{bmatrix}$. The bra $\langle v|$ would then be $\begin{bmatrix} 1+i \\ 2-i \end{bmatrix}^\dagger = [1-i \quad 2+i]$.

Furthermore we represent the *Outer product* of two vectors $|a\rangle$ and $|b\rangle$ as $|a\rangle\langle b|$, and the *Inner product* of the same vectors as $\langle a|b\rangle$.

Example 1.0.2. Let $|v\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|w\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. Then $|v\rangle\langle w| = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$. The inner product $\langle w|v\rangle = \begin{bmatrix} 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0$.

With this notation in mind, we will begin presenting some of the basic concepts in quantum computation.

1.1 Postulates of quantum mechanics

Quantum mechanics is the mathematical framework we need to work within to explain concepts in quantum computation. This section introduces the postulates of quantum mechanics in a formal manner in terms of state vectors. We begin by introducing the most fundamental concepts of quantum computation

in combination with the associated postulate. In the next section we reformulate two of these postulates in terms of density operators. This new formulation comes in handy when dealing with certain scenarios in quantum computation.

1.1.1 Single qubits

In a classical computer, the bit is the elementary unit of information. The quantum analogue for this is the qubit. While a classical bit is either in the state 0 or 1, the state of the qubit can be represented by some linear combination of the two. We say this is a two state quantum system. Loosely speaking a qubit is a mixture of 0 and 1.

In order to manipulate qubits we have to have a suitable space to operate in. It so happens that we can use a complex vector space with inner product like the *Hilbert space* to achieve our goals.

Definition 1.1.1. *Hilbert space*

A complex finite vector space H that assigns a complex number to the inner product $\langle x, y \rangle$ for every pair of vectors $x, y \in H$ is called a *Hilbert space* if:

- The inner product is linear with respect to the first argument:

$$\langle ax_1 + bx_2, y \rangle = a\langle x_1, y \rangle + b\langle x_2, y \rangle$$

- The inner product is equal to the complex conjugate reverse inner product:

$$\langle x, y \rangle = \overline{\langle y, x \rangle}$$

- The inner product is positive definite, in other words:

$$\langle x, x \rangle \geq 0$$

and is 0 only when $x = 0$

- The inner product is anti-linear with respect to the second argument:

$$\langle x, ay_1 + by_2 \rangle = \bar{a}\langle x, y_1 \rangle + \bar{b}\langle x, y_2 \rangle$$

This follows from the other properties.

The following definition is important when defining the qubit. More generally, the state of a qubit is a special case of a quantum state. A quantum state is a vector in Hilbert space which we think of as assigning probabilities to certain outcomes in a system.

Definition 1.1.2. *Quantum state*

Assume some orthonormal basis $\mathcal{B} = \{|b_1\rangle, |b_2\rangle, \dots, |b_n\rangle\}$ of the Hilbert space \mathcal{H} . A quantum state is a vector

$$|\psi\rangle = \sum_{i=1}^n a_i |b_i\rangle$$

where $a_i \in \mathbb{C}$ for $i \in \{1, 2, \dots, n\}$ and $\sum_{i=1}^n |a_i|^2 = 1$.

As we shall discuss later, we interpret this definition as each part b_i in the basis representing the outcome of a measurement with the associated probability $|a_i|^2$.

In the case of the qubit we denote the two *basis states* as $|0\rangle$ and $|1\rangle$. This means that the state of the qubit can be any (complex)linear combination

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

We think of $|\alpha|^2$ and $|\beta|^2$ as being the probabilities of *measuring* the qubit as being in state $|0\rangle$ and $|1\rangle$ respectively. We refer to α and β as the (probability) *amplitudes*. Because $|\alpha|^2 + |\beta|^2 = 1$, geometrically we can think of this as a unit vector in two-dimensional complex space. More generally, if we have a quantum state $\sum_{i=1}^n a_i |b_i\rangle$ we may think of it as a unit vector in n -dimensional space. For the two dimensional case, we sometimes refer to the linear combination of basis states $|\alpha|^2 + |\beta|^2$ as a *superposition* when α and β are non-zero. When we talk about bases in this paper, it will always be assumed that they are *orthonormal*. **Unless explicitly stated otherwise, throughout the paper we will always assume**

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

We are now ready to introduce the first postulate of quantum mechanics.

Postulate 1 *Associated to any closed (physical) system is a complex vector space, a Hilbert space \mathcal{H} . We call this the state space of the system. The system is completely described by a unit vector in the state space $|\psi\rangle \in \mathcal{H}$.* unit vector

Our interpretation of this postulate in the context of qubits is that the state of the system (qubit) can be described by a quantum state $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$. We may describe the state of any closed system with such a state vector, but we cannot know what it looks like without measuring it (introducing outside interference) which in turn collapses the state. More about this in the measurement section.

Relative and global phase

There is a small caveat to postulate 1 worth mentioning regarding whether two vectors are different or not. We introduce two definitions in application should be regarded different or not

Definition 1.1.3. *Relative phase*

Let $\theta, \theta' \in \mathbb{R}$. Two qubit states $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$ and $|\phi\rangle = \alpha' |0\rangle + \beta' |1\rangle$ are said to differ by a relative phase if $\alpha = e^{i\theta} \alpha'$ and $\beta = e^{i\theta'} \beta'$ when $e^{i\theta} \neq e^{i\theta'}$.

Definition 1.1.4. *Global phase*

Let $\theta \in \mathbb{R}$. Two qubit states $|\psi\rangle$ and $|\phi\rangle$ are said to be equal up to a global phase if $|\psi\rangle = e^{i\theta} |\phi\rangle$.

For reasons that will become clear in the section about the reformulation of certain quantum postulates, we regard two states equal up to a global phase as being the same. We will only say informally that a relative phase difference implies a real physical difference between states. Meanwhile, global phase factors are seen as nonphysical.

Example 1.1.1. Consider the states $|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ and $|\psi_3\rangle = \frac{i}{\sqrt{2}}(|0\rangle + |1\rangle)$.

States $|\psi_1\rangle$ and $|\psi_2\rangle$ differ by relative phase and are thus not the same. State $|\psi_1\rangle$ and $|\psi_3\rangle$ are equal up to a global phase and we regard them as the same.

Remark The attentive and topologically minded reader will perhaps realize that this classification of similar states is equivalent to introducing the equivalence relation $|\psi\rangle \sim e^{i\theta} |\psi\rangle$, creating a quotient space \mathcal{H}/\sim known as the *projective Hilbert space*. This remark is not necessary to understand the rest of the paper.

Example 1.1.2. A very common geometric representation of the single qubit state is given by the Bloch sphere as represented in figure 1.1. The north pole represents being in state $|0\rangle$ and the south pole state $|1\rangle$.

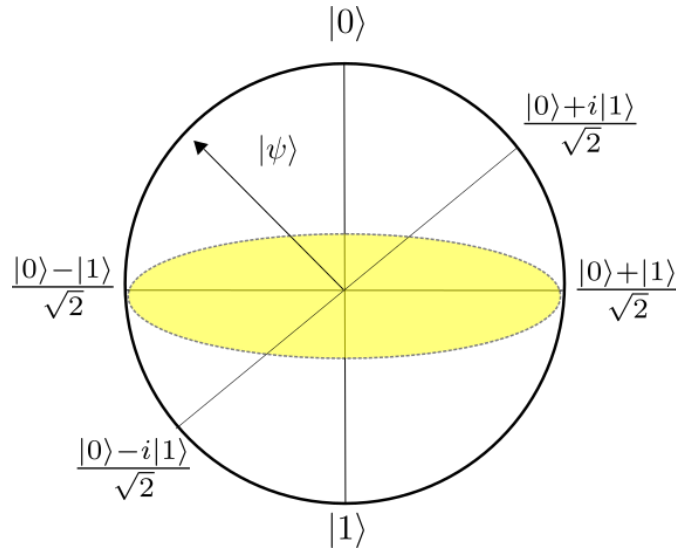


Figure 1.1: Bloch sphere representation for single qubit state

1.1.2 Multiple qubits

In a state space with n qubits, there are 2^n basis vectors, each represented as $|j_0 j_1 \dots j_{n-1}\rangle$ where $j_i \in \{0, 1\}$. The "components" of multiple qubits are of course, some sort of combination of single qubits. Formally, we shall use the following definition.

Definition 1.1.5. Tensor product

Let V and W be vector spaces with bases $\{|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle\}$ and $\{|w_1\rangle, |w_2\rangle, \dots, |w_m\rangle\}$ respectively.

The tensor product of V and W is written as $V \otimes W$ and is an nm -dimensional vector space spanned by elements of the form $v_i \otimes w_j$. This new vector space is called the product space. The operator in question \otimes is called the tensor operator.

We define the tensor operator by the following relations:

- $|v\rangle \otimes (|a_1\rangle + |a_2\rangle) = |v\rangle \otimes |a_1\rangle + |v\rangle \otimes |a_2\rangle$
- $(|b_1\rangle + |b_2\rangle) \otimes |w\rangle = |b_1\rangle \otimes |w\rangle + |b_2\rangle \otimes |w\rangle$
- $(a|v\rangle) \otimes |w\rangle = |v\rangle \otimes (a|w\rangle) = a(|v\rangle \otimes |w\rangle)$

Tensor products will only be talked about in the context of Hilbert spaces in this paper. The following definition will come in handy.

Definition 1.1.6. Inner product in product space

Let $\mathcal{H}_1 \otimes \mathcal{H}_2$ be a product space of two Hilbert spaces \mathcal{H}_1 and \mathcal{H}_2 . Also let $|v_1\rangle \otimes |v_2\rangle, |w_1\rangle \otimes |w_2\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$. We define the inner product $\langle |v_1\rangle \otimes |v_2\rangle \mid |w_1\rangle \otimes |w_2\rangle \rangle \equiv \langle v_1|w_1\rangle \cdot \langle v_2|w_2\rangle$.

This also works for vectors that are linear combinations of such vectors above, because of the distributive property.

Example 1.1.3. Let \mathcal{H} be a two dimensional Hilbert space with standard basis

$$\left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}.$$

The product space $\mathcal{H} \otimes \mathcal{H}$ has the basis

$$\left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}.$$

It's perhaps a little frustrating that the tensor product cannot be simplified further. We will usually be referring to these tensor products by their representation in the standard basis of the product space, sorted lexicographically.

Example 1.1.4. In example 1.1.3 by naming $\{|v_1\rangle, |v_2\rangle\} =$

$$\left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\},$$

we see that the basis provided for $\mathcal{H} \otimes \mathcal{H}$ is sorted lexicographically and has the following representation in the standard basis:

$$\left(\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right)$$

As you may have already guessed, this way of combining vectors in Hilbert spaces is how we're going to represent multiple qubits. Letting $|j_i\rangle \in \mathcal{H}_i$ for $i \in \{1, 2, \dots, n\}$ be single qubits, the composition of them has the following notation:

$$|j_0 j_1 \dots j_{n-1}\rangle = |j_0\rangle \otimes |j_1\rangle \otimes \dots \otimes |j_{n-1}\rangle.$$

where j_i is a two state quantum system, a single qubit. By definition 1.1.2, a system of n qubits is a quantum state represented by a 2^n dimensional vector.

Example 1.1.5. Consider the basis $\{|\Psi^+\rangle, |\Psi^-\rangle, |\Phi^+\rangle, |\Phi^-\rangle\}$ for a two-qubit system where:

$$\begin{aligned} |\Psi^+\rangle &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\ |\Psi^-\rangle &= \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \\ |\Phi^+\rangle &= \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \\ |\Phi^-\rangle &= \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \end{aligned}$$

These are called the *Bell states*, and the basis is called the *Bell basis*.

1.1.3 Quantum gates

The state vector of some state space is manipulated into other vectors of the same space through linear transformations. These operations are the quantum equivalent of the logic gates in classical computation. They are the building blocks with which we will later compose *quantum circuits*.

Definition 1.1.7. Quantum gate

A quantum gate is a complex unitary matrix.

Definition 1.1.8. Quantum circuit

A quantum circuit is a sequence of quantum gates.

Interestingly, the unitary constraint is sufficient for the definition of quantum gates. The unitary condition implies that every quantum gate is invertible, which in the context of quantum circuits is referred to as the circuit being *reversible*.

Remark Just as there is a small set of classical gates that can be used to compute some arbitrary function, there is a corresponding set of such gates in quantum computation. We call these set of gates *universal*. Universality and reversibility will be discussed in greater detail in chapter 2.

The following postulate describes how a quantum system changes with time.

Postulate 2 *The evolution of a closed quantum system is described by a quantum gate. This means that for state $|\psi_1\rangle$ at point t_1 in time and $|\psi_2\rangle$ at point t_2 in time, they are related to each other by some unitary matrix U :*

$$|\psi_1\rangle = U |\psi_2\rangle.$$

Where U is a function of t_1, t_2 .

This description of time evolution will serve as a good approximation of the framework for quantum computation we're constructing. The limitation being that we can only describe the system in discrete time. The continuous time formulation of the postulate leads to the introduction of the *Schrödinger equation*. Although very interesting, this is outside the scope of the paper and for our purposes not completely necessary to formulate the framework.

1.1.4 Measurement

For some arbitrary system, recall that a state vector $|\psi\rangle$ lives inside of a Hilbert space \mathcal{H} which is spanned by some basis. As mentioned earlier, if the Hilbert space has dimension n we will always set the basis as the standard basis for n -Euclidean space. By definition this means we may write any state vector as a linear combination of this basis,

$$|\psi\rangle = \sum z_j |j\rangle.$$

The act of measuring a state vector "collapses" it to one of the elements in this basis. This leads to the third postulate.

Postulate 3 *The probability of obtaining outcome j from a measurement with respect to the standard basis of the state $|\psi\rangle = \sum z_j |j\rangle$ is $|z_j|^2$.*

This is consistent with definition 1.1.2 which says that $\sum |z_j|^2 = 1$.

Notice how measurement is always with respect to a basis. In practice this will always be the standard basis, but it's worth noting that the result of the measurement depends entirely on the choice of basis. **We will always measure with respect to the standard basis in this paper.**

Example 1.1.6. Let $|\psi\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$. Measured in the standard basis, we would

have an equal probability $\frac{1}{2}$ of the outcome $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ as well as $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

If we instead measure with the *Hadamard basis* $\left\{ \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \right\}$, the outcome

would be $|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ with probability 1.

A more natural explanation of this will be given in the next section, where Postulate 3 is reformulated in terms of projection operators. Lastly we present Postulate 4, which is a more formal description of the composition of single qubits into multiple qubits described earlier.

Postulate 4 *Let \mathcal{H}_A and \mathcal{H}_B be Hilbert spaces. The composite state \mathcal{H}_{AB} is tensor product of its components $\mathcal{H}_A \otimes \mathcal{H}_B$.*

Not all states $|\psi_{AB}\rangle \in \mathcal{H}_{AB}$ can be described as $|\psi_A\rangle \otimes |\psi_B\rangle$ for $|\psi_A\rangle \in \mathcal{H}_A$ and $|\psi_B\rangle \in \mathcal{H}_B$. This is a bit surprising, since it's in a way saying that a system can't always be described by the sum of its parts. This isn't true for classical mechanics and seems to be an oddity of stepping into the quantum world. It's in this confusion that we introduce the concept of *entanglement*.

Definition 1.1.9. Entanglement

Any state $|\psi\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \dots \otimes \mathcal{H}_n$ which cannot be described by a tensor product $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \dots \otimes |\psi_n\rangle$ for $|\psi_i\rangle \in \mathcal{H}_i$ is referred to as entangled.

Given some thought, it's clear that the vast majority of the possible states in a given system are entangled. In fact, these states are what makes quantum computation so powerful and the ability to compute in parallel. The next section will deal with reformulating postulate 1 and 3 in order to give a more general understanding, and for postulate 3 giving a more solid mathematical understanding rather than dismissing it as quantum weirdness.

1.2 Reformulation of the standard postulates of quantum mechanics

In this section we will give a brief reformulation of two of the four postulates from quantum mechanics according to Landsberg[10]. This will be done in terms of density operators. We provide motivations for the reformulations.

1.2.1 Partial measurements

The state of a quantum system can be described by some vector in a Hilbert space \mathcal{H} . More specifically, a full description of the state of an n -qubit system is given by some vector $v \in (\mathbb{C}^2)^{\otimes n}$. In most algorithms however, we require some workspace registers which are not to be measured. Let $|\psi\rangle = \sum z_I |I\rangle$ be some state vector. Instead of simply saying that the probability of the outcome I being equal to $|z_I|^2$, we will define measurement with orthogonal projection operators into $\mathbb{C}|I\rangle$. The space spanned by $|I\rangle$.

Definition 1.2.1. Projection operator

Let \mathcal{M} be some linear subspace in $(\mathbb{C}^2)^{\otimes(n+m)}$, and let m be the amount of workspace qubits. A projection operator $\Pi_{\mathcal{M}}$ is a map

$$\Pi_{\mathcal{M}} : (\mathbb{C}^2)^{\otimes(n+m)} \rightarrow \mathcal{M}.$$

In the case of measurement $\mathcal{M} = |I\rangle \otimes (\mathbb{C}^2)^{\otimes m}$ where $I \in \{0, 1\}^n$. We call this special case an **orthogonal projection operator**. The formal description of the probability of a state being measured is given by the following proposition.

Proposition 1.2.1. *Let $|\psi\rangle \in (\mathbb{C}^2)^{\otimes(n+m)}$, and $I \in \{0, 1\}^n$. Then the probability that given $|\psi\rangle$ one measures $|I\rangle$, $p(|I\rangle | |\psi\rangle)$, is the following expression. $p(|I\rangle | |\psi\rangle) = \langle\psi| \Pi_{\mathcal{M}} |\psi\rangle$.*

Proof. Since $|\psi\rangle \in (\mathbb{C}^2)^{\otimes(n+m)}$, then $p(|I\rangle | |\psi\rangle)$ can be written as the sum of the probabilities $p(|\psi\rangle, |I\rangle |R\rangle)$ for all $R \in \{0, 1\}^m$ for some $I \in \{0, 1\}^n$. Equivalently,

$$p(|I\rangle | |\psi\rangle) = \sum_{R \in \{0, 1\}^m} p(|\psi\rangle, |I\rangle |R\rangle).$$

Each probability is the same as the inner product $\langle\psi|I\rangle |R\rangle \langle I| \langle R|\psi\rangle$. Thus,

$$\sum_{R \in \{0, 1\}^m} p(|\psi\rangle, |I\rangle |R\rangle) = \sum_{R \in \{0, 1\}^m} \langle\psi|I\rangle |R\rangle \langle I| \langle R|\psi\rangle.$$

This can be rewritten as

$$\langle\psi| (|I\rangle \langle I| \otimes Id_{(\mathbb{C}^2)^{\otimes m}}) |\psi\rangle$$

which we finally identify as

$$\langle\psi| \Pi_{\mathcal{M}} |\psi\rangle.$$

□

Let $Id_{\mathcal{H}}$ be the identity matrix for the Hilbert space \mathcal{H} . We are ready to reformulate the third postulate of quantum mechanics as

Postulate 3 - Measurements

A state is always measured with a corresponding collection of projection operators $\Pi_{\mathcal{M}_j}$ such that $\sum_k \Pi_{\mathcal{M}_k} = Id_{\mathcal{H}}$. The probability of state $|\psi\rangle$ being measured in state space \mathcal{M}_j is given by $\langle\psi| \Pi_{\mathcal{M}_j} |\psi\rangle$

This reformulation in terms of projection operators is useful for generalizing the concept of measurement. For example, it allows us to describe more precisely what it means to measure a state with respect to a certain basis. Simply let \mathcal{M}_j be the spaces spanned by the vectors in the basis which we wish to measure in respect to. As mentioned earlier, in this paper when we talk about measurement it is implied that $\mathcal{M}_j = \mathbb{C}|j\rangle$ for the standard basis vectors $|j\rangle \in \mathcal{H}$. There is a very natural explanation as to why two states equal up to global phase are regarded as equal. Consider the proposition.

Proposition 1.2.2. *Two states equal up to global phase are regarded as equal.*

Proof. With our new formulation of the standard postulates in terms of projection operators, consider for $\theta \in \mathbb{R}$

$$\langle\psi| e^{-i\theta} M_m^\dagger M_m e^{i\theta} |\psi\rangle = \langle\psi| M_m^\dagger M_m |\psi\rangle.$$

□

Mixed states

So far our efforts have gone to accurately describing the measurements and manipulations of *pure states*, that is to say states which can be represented as a vector in the tensor product of complex Hilbert spaces. We can however, come up with systems which have states that *cannot* be described in this way. An example of this kind of scenario is the following. Let $|\psi\rangle_1, |\psi\rangle_2$ be two states in the Hilbert space \mathcal{H} . Assume a qubit is either in state $|\psi\rangle_1$ or in state $|\psi\rangle_2$ with equal probability. We have introduced an additional thing to keep in mind when measuring the system, and we would like our measurement postulate to encapsulate this scenario. First of all we would like to define this kind of state that we just described. Consider the following two definitions.

Definition 1.2.2. Ensemble of pure states

Suppose we have n states $|\psi_i\rangle$ with an associated probability p_i . We will call $\{p_i, |\psi_i\rangle\}$ an ensemble of pure states.

Sometimes an ensemble of pure states is referred to as a **mixed state**, the idea being that a mixed state refers to both pure and mixed states[6]. Here we will treat them as disjoint. Mixed states cannot be represented as vectors in tensor products of complex Hilbert spaces. These states are instead described by matrices.

Definition 1.2.3. Density operator

For probabilities $0 \leq p_i \leq 1$, let $\{p_i, |\psi_i\rangle\}$ be some pure ensemble of states. The density operator for this system is defined as

$$\rho = \sum_s p_s |\psi_s\rangle \langle \psi_s|.$$

The density operator is sometimes referred to as the density matrix. A pure state $|\psi\rangle$ may be represented with this matrix notation as $|\psi\rangle \langle \psi|$. This way of describing states through density operators is more general since it also includes mixed states. Let's prove some properties about density operators.

Theorem 1.2.1. Let ρ be a density operator. The following facts are true:

- ρ is Hermitian.
- ρ is a positive operator, $\forall |\phi\rangle, \langle \phi| \rho |\phi\rangle \geq 0$.
- $\text{trace}(\rho) = 1$.

Proof. To prove that $\rho = \sum_s p_s |\psi_s\rangle \langle \psi_s|$ is Hermitian we need to show $\rho = \rho^\dagger$. From the properties of the Hermitian adjoint operator, we know $(\sum_i p_i |\psi_i\rangle \langle \psi_i|)^\dagger = \sum_i p_i (|\psi_i\rangle \langle \psi_i|)^\dagger = \sum_i p_i (\langle \psi_i|^\dagger |\psi_i\rangle^\dagger) = \sum_i p_i |\psi_i\rangle \langle \psi_i|$. This proves that ρ is Hermitian.

Let $|\phi\rangle$ be any arbitrary state. We can write $\langle \phi| \rho |\phi\rangle = \sum_i p_i \langle \phi| \psi_i\rangle \langle \psi_i| \phi\rangle =$

$$\sum_i p_i |\langle \phi | \psi_i \rangle|^2 \geq 0.$$

Lastly, it's simple to show that ρ has trace 1 since

$$\text{tr}(\rho) = \sum_i p_i \text{tr}(|\psi_i\rangle\langle\psi_i|) = \sum_i p_i = 1.$$

□

Lastly we note that a density operator is an *endomorphism* for the associated Hilbert space.

Definition 1.2.4. Endomorphism

A is linear operator Let \mathcal{H} be a vector space. An operator A that maps \mathcal{H} to itself is called an endomorphism of \mathcal{H} . The set of all such operators is written $\text{End}(\mathcal{H})$.

It's clear that a density operator ρ is an endomorphism because it maps \mathcal{H} to itself.

In order to better describe mixed states and motivated by the above definitions, we now reformulate the first postulate in terms of density operators. The final set of postulates are the following.

Postulate 1'

Associated to any closed system is a Hilbert space \mathcal{H} , known as the state space of the system and a density operator $\rho \in \text{End}(\mathcal{H})$ describing the state of the system.

Postulate 2

The evolution of a closed system is described by quantum gates acting on its associated density operator ρ by matrix multiplication.

dålig engelska While we are at it, measurement can also be reformulated in terms of these density operators.

Postulate 3'

Measurement is described by a collection of projection operators $\Pi_{\mathcal{M}_j}$ such that $\sum_k \Pi_{\mathcal{M}_k} = Id_{\mathcal{H}}$. The probability that ρ is measured in state \mathcal{M}_j is given by $\text{tr}(\Pi_{\mathcal{M}_j}\rho)$. Upon measurement, the state collapses to \mathcal{M}_j .

Postulate 4 *Let \mathcal{H}_A and \mathcal{H}_B be Hilbert spaces. The composite state \mathcal{H}_{AB} is tensor product of its components $\mathcal{H}_A \otimes \mathcal{H}_B$.*

This summary concludes the section of quantum computation preliminaries.

Chapter 2

Quantum Computation

Chapter 2 presents the building blocks needed to construct basic quantum circuits. The concept of reversibility and universality will be explained in terms of our mathematical framework. Universality in this context meaning the construction of arbitrary functions from a given set of quantum gates. We want to show that classical circuits can be made reversible with no significant efficiency loss. Even though classical circuits may perhaps best be left to classical computers for other reasons, this results tells us in some sense that quantum computers are at least as powerful as classical ones.

2.1 The quantum circuit model

This section deals with describing the model of quantum circuits.

2.1.1 Single qubit quantum gates

We can do a surprising amount of things with single quantum gates, which we will refer to as *1-gates*. Three types of 1-gates are especially important, consider these three rather arbitrary looking types of gates.

$$R(\beta) = \begin{bmatrix} \cos \beta & \sin \beta \\ -\sin \beta & \cos \beta \end{bmatrix}, T(\alpha) = \begin{bmatrix} e^{i\alpha} & 0 \\ 0 & e^{-i\alpha} \end{bmatrix}, K(\delta) = e^{i\delta}I.$$

Definition 2.1.1. *Using the box above, let δ, β, α be real numbers. We refer to $K(\delta)$ as the phase shift gate, $R(\beta)$ as the rotation gate and $T(\alpha)$ as the phase rotation gate.*

The reason these three operations are special is that we can write any single qubit quantum gate in terms of them.

In order to prove this, we start with a lemma:

Lemma 2.1.1. Any unitary matrix $U = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}$ can be expressed as $U = \begin{bmatrix} e^{i\phi_{00}} \cos \beta & e^{i\phi_{01}} \sin \beta \\ -e^{i\phi_{10}} \sin \beta & e^{i\phi_{11}} \cos \beta \end{bmatrix}$ for $\phi_{ij}, \beta \in \mathbb{R}$.

Proof. The unitary condition $UU^\dagger = I$ implies

$$\begin{cases} |u_{00}|^2 + |u_{01}|^2 = 1 \\ u_{00}\overline{u_{10}} + u_{01}\overline{u_{11}} = 0 \\ |u_{11}|^2 + |u_{10}|^2 = 1 \end{cases}$$

This in turn implies that $|u_{00}| = |u_{11}|$ and $|u_{01}| = |u_{10}|$. This means we can rewrite the coefficients u_i in terms of sine and cosine for some angle β ,

$$Q = \begin{bmatrix} e^{i\phi_{00}} \cos(\beta) & e^{i\phi_{01}} \sin(\beta) \\ -e^{i\phi_{10}} \sin(\beta) & e^{i\phi_{11}} \cos(\beta) \end{bmatrix}.$$

We also see that $\phi_{00} + \phi_{11} = \phi_{01} + \phi_{10}$ because Q is a unitary matrix, so $QQ^\dagger = I$. \square

Theorem 2.1.2. Any single qubit quantum gate U can be written as $U = K(\delta)T(\alpha)R(\beta)T(\gamma)$ for some $\alpha, \beta, \delta \in \mathbb{R}$.

Proof. According to lemma 2.1.1 we can write $U = \begin{bmatrix} e^{i\phi_{00}} \cos \beta & e^{i\phi_{10}} \sin \beta \\ -e^{i\phi_{01}} \sin \beta & e^{i\phi_{11}} \cos \beta \end{bmatrix}$.

Consider that $K(\delta)T(\alpha)R(\beta)T(\gamma) =$

$$\begin{bmatrix} e^{i(\delta+\alpha+\gamma)} \cos \beta & e^{i(\delta+\alpha-\gamma)} \sin \beta \\ -e^{i(\delta-\alpha+\gamma)} \cos \beta & e^{i(\delta-\alpha-\gamma)} \sin \beta \end{bmatrix}.$$

It's easy to convince yourself that $\phi_1 = \delta + \alpha + \gamma, \phi_2 = \delta + \alpha - \gamma, \phi_3 = \delta - \alpha - \gamma, \phi_4 = \delta - \alpha - \gamma$ and that this system has a solution. \square

Example 2.1.1. The Hadamard gate

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

can be decomposed as $K(0)T(0)R(\frac{\pi}{4})T(0) =$

$$II \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} I.$$

In other words, just a special case of the rotation gate R .

2.1.2 Two qubit quantum gates

Two qubit quantum gates are a very important addition to our model, because with 1-gates only we cannot describe entanglement. Entanglement of two qubits was defined in chapter 1. The way we achieve entanglement is through a set of gates referred to as *control gates*. A control gate specifies a *control qubit* and a *target qubit*. We will explain what this means after introducing the following notation.

Definition 2.1.2. *We can represent quantum gates by specifying where the basis vectors are mapped to,*

$$\begin{aligned} |00\dots 0\rangle &\mapsto |a_0\rangle \\ |01\dots 0\rangle &\mapsto |a_1\rangle \\ &\dots \\ |11\dots 1\rangle &\mapsto |a_n\rangle \end{aligned}$$

or the matrix representation

$$|00\dots 0\rangle \langle a_0| + |01\dots 0\rangle \langle a_1| + \dots + |11\dots 1\rangle \langle a_n|$$

for $|a_i\rangle \in \mathcal{H}^n$.

Suppose we have states $|a_0\rangle, \dots, |a_n\rangle \in \mathcal{H}$. In the standard basis, a control gate acts on the target qubit with a quantum gate Q if the control qubit is in state $|1\rangle$. If the control qubit is in state $|0\rangle$, the target qubit is not acted upon. We express this as

$$\bigwedge Q = |0\rangle \langle 1| \otimes I + |1\rangle \langle 1| \otimes Q.$$

The \bigwedge means that it is a controlled gate, the Q tells us what kind. The above is therefore a controlled Q -gate.

Example 2.1.2. A very common controlled 2-gate is the controlled-NOT gate, which we may express in the following ways $C_{NOT} = \bigwedge X = |0\rangle \langle 1| \otimes I + |1\rangle \langle 1| \otimes X$

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

Here X is the single qubit *NOT* gate

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{2.1}$$

2.1.3 Graphical representation of quantum circuits

The definition of a quantum circuit was introduced in chapter 1 as a sequence of quantum gates. We will give some common graphical notation to describe quantum circuits.

The figure below describes a possible single wire quantum circuit.

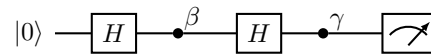


Figure 2.1: A single wire quantum circuit

Going through the symbols in order, $|0\rangle$ is the state the qubit is in before entering the circuit. Quantum gates are represented by boxes with symbols on them. Of course H is the Hadamard gate, and we apply H to $|0\rangle$. It follows that the qubit at β has the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. After yet again applying H to the qubit, we return to the state $|0\rangle$ because the Hadamard gate is its own inverse. Lastly, **measurement is represented by the last box**. The circuit of course outputs $|0\rangle$. Figure 2.1 is a single qubit quantum circuit. **We represent multiple qubits by drawing wires in parallel, and multiple qubit gates as boxes or symbols intersecting into multiple wires.** More common notation is summarized in the figure below.












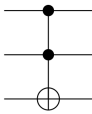
Operator	Gate(s)	Matrix
Pauli-X (X)	 \oplus	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)	 	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP	 	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

Figure 2.2: Common quantum gates and their graphical representation. Made by Rxtreme and shared under the Creative Commons Attribution-Share Alike 4.0 International license.

2.1.4 Universality

The reader may be familiar with the classical result that arbitrary (classical) functions may be calculated with a combination of *AND*, *OR*, and *NOT* gates. Informally, we will refer to such a set as of gates as universal, in the sense that

all other gates may be constructed from them.

Considering the context, it's natural to ask, what would be the analogous result in quantum computation? As it turns out, we can construct a similar non-discrete set containing only 1-gates and C_{NOT} gates. We shall follow the construction in Benenti, Casati and Strini[11].

Theorem 2.1.3. *Let U be a quantum gate, then $\wedge U$ gate may be decomposed into 1-gates and C_{NOT} gates.*

Proof. We would like to make the decomposition $\wedge U = (\wedge K(\delta)) (\wedge U')$ where $U' = T(\alpha)R(\beta)T(\gamma)$.

It's possible to implement $\wedge K(\delta)$ using only single qubit gates in the following way:

$$\begin{aligned} \wedge K(\delta) &= |0\rangle\langle 0| \otimes I + e^{i\delta} |1\rangle\langle 1| \otimes I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{i\delta} & 0 \\ 0 & 0 & 0 & e^{i\delta} \end{bmatrix} = \\ &= \begin{bmatrix} 1 & 0 \\ 0 & e^{i\delta} \end{bmatrix} \otimes I = \left(K\left(\frac{\delta}{2}\right) T\left(\frac{-\delta}{2}\right) \right) \otimes I. \end{aligned}$$

Implementing $\wedge U'$ is perhaps not as intuitive, and makes use of the following type of quantum gates.

$$\begin{aligned} U_0 &= T(\alpha)R\left(\frac{\beta}{2}\right), \\ U_1 &= R\left(\frac{-\beta}{2}\right)T\left(\frac{-(\gamma+\alpha)}{2}\right), \\ U_2 &= \left(\frac{\gamma-\alpha}{2}\right). \end{aligned}$$

We can now implement $\wedge U'$ in terms of these 1-gates and C_{NOT} gates by

$$\wedge U' = (I \otimes U_0)(I \otimes U_1)(I \otimes U_2).$$

Finally, we combine the two to get our final result

$$\wedge U = \left(\left(K\left(\frac{\delta}{2}\right) T\left(\frac{-\delta}{2}\right) \right) \otimes I \right) (I \otimes U_0)(I \otimes U_1)(I \otimes U_2).$$

□

We illustrate this circuit in figure 2.3.

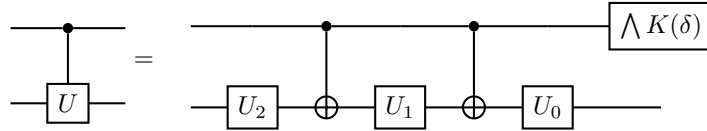


Figure 2.3: $\wedge U$ gate decomposed into 1-gates and C_{NOT} gates

Recall the Toffoli gate from figure 2.2. The Toffoli gate is actually universal in classical computation, a result we will not prove here. By showing that we may implement such a gate with only 1-gates and C_{NOT} gates, we also encompass classical computations.

The details in the proofs of the following two theorems will not be included, a quantum wire and an intuitive explanation will be offered.

Theorem 2.1.4. *The Toffoli gate may be implemented from C_{NOT} gates and 1-gates.*

Proof. Let $V = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$. We already know we can write $\wedge V$ in terms of C_{NOT} gates and 1-gates. The following circuit implements the Toffoli gate.

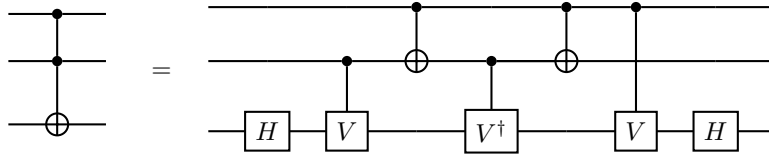


Figure 2.4: Quantum circuit illustrating the general procedure in the proof of theorem 2.1.4.

The details of verifying this consists of confirming that the 8×8 matrix representation of the circuit maps the basis vectors to the correct position. This is quite messy, so hopefully an intuitive explanation will suffice.

If the control gates do not trigger, the idea is that the other gates should cancel each other out. This leaves the state unchanged. Otherwise, the gates will not cancel out and transform the state accordingly. \square

Let's extend some notation. We will refer to a control gate with k control qubits with the notation $\wedge_k U$ for some gate U . In other words, we perform U if and only if all k control qubits are 1. Furthermore $\wedge_x^i U$ is the control gate for target qubit i and pattern x .

Example 2.1.3. Let $|b_0 b_1 b_2\rangle$ be a three-qubit system. The Toffoli gate uses two qubits as control qubits and another as the target qubit. In our new notation, this can be written as $\wedge_{110}^2 X$. It can also be written as $\wedge_{111}^2 X$, since only the values of the control qubits decide the gate.

We're interested in being able to construct $\wedge_k U$ gates, and we can do so only with Toffoli and $\wedge U$ gates.

Theorem 2.1.5. *Let U be any 1-gate and k some integer. We can decompose $\bigwedge_k U$ in terms of Toffoli and $\bigwedge U$ gates.*

Proof. Once again, the proof will consist of the graphical circuit representation. We need k workspace qubits to store the result of the previous Toffoli gate.

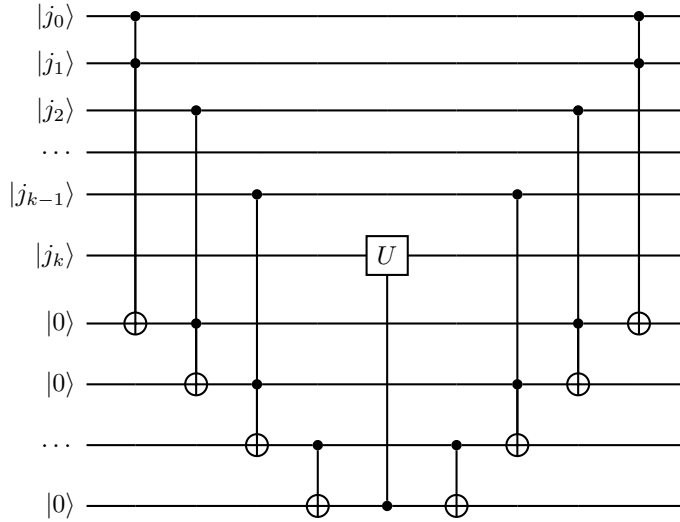


Figure 2.5: Quantum circuit illustrating the general procedure in the proof of theorem 2.1.5.

□

The final argument, which shows that any arbitrary matrix U may be written in terms of the gates we've dealt with, is perhaps a bit involved. The intuitive explanation goes something like this: Any unitary transformation is just a rotation. To transform a 2^n dimensional vector, we can do so with a sequence of rotation in 2 dimensions.

For the formal argument, we refer to [11].

Theorem 2.1.6. *Any arbitrary quantum gate may be decomposed in terms of C_{NOT} gates and $R(\beta), T(\alpha), K(\delta)$ gates for $\beta, \alpha, \delta \in \mathbb{R}$.*

Proof. See [11].

□

2.2 Reversible computation

Looking back to postulate 2, we see that our qubits are transformed through unitary matrices, which we refer to as quantum gates. One basic property of these transformation is that they're invertible, the direct consequence being that our quantum circuit is *reversible*. That is, the matrix that describes the circuit

is invertible.

Classical logic gates can also be reversible, by the following definition.

Definition 2.2.1. *A logic gate, represented by a function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$, is called reversible if f is bijective.*

It's worth noting that the representation of a reversible logic gate is just a permutation. Classical circuits, which are sequences of logic gates and can thus be represented as functions, are clearly not all reversible. Consider the OR gate for example. By knowing that the output is 1, one still can't deduce what the two input bits were. It's tempting to disregard reversibility as a quirk of quantum computation, but consider that a quantum algorithm may need to perform classical subroutines. Modular exponentiation is an important classical subroutine present in Shor's algorithm. In the original paper, Shor even referred to modular exponentiation as "The bottleneck of the quantum factoring algorithm". We would of course like this bottleneck to be as efficient as possible.

The object of this section is to show that any classical circuit can be made reversible with no significant efficiency loss, according to Rieffel and Polak[5].

2.2.1 A first iteration

Fact: *The AND and NOT operations form a universal set of logic gates for classical circuits.*

As to not delve too far into logic and functional completeness, we will not prove this fact. We can assume that every circuit we construct is only composed out of AND and NOT operations. These circuits are not necessarily reversible, as illustrated by the example below. We use notation \neg to represent logical negation and \wedge to represent logical conjunction.

Example 2.2.1. Figure 2.6 illustrates a classical circuit reusing input registers to store intermediate calculations.

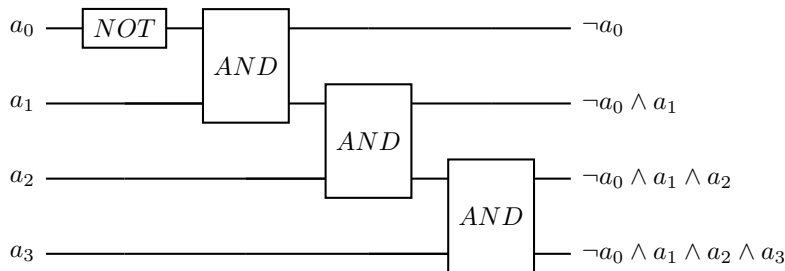


Figure 2.6: A classical irreversible circuit.

A common technique to make a classical circuit reversible is to add an additional output bit for each AND gate. For a circuit with s bits and t gates, we

at most need t additional bits to make it reversible. This means replacing the AND gates with Toffoli gates.

Example 2.2.2. The circuit in example 2.2.1 has been made reversible below.

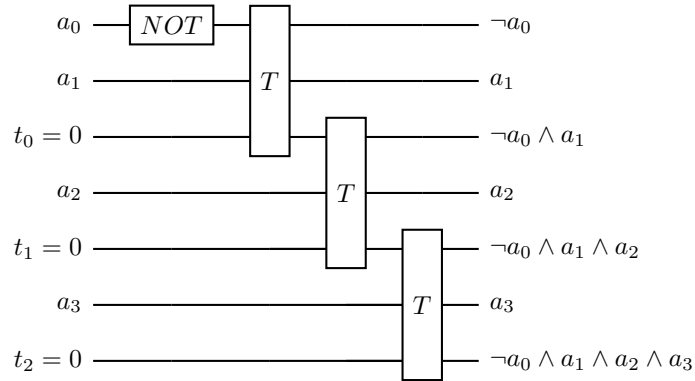


Figure 2.7: A reversible classical circuit

This approach of simply substituting AND gates with Toffoli gates won't do. Worst case scenario we double the amount of bits used, which can hardly be called space efficient. We would like to somehow reuse the bits carrying the intermediate results. Just plain resetting them to 0 is of course not a reversible action. Instead we aim to *uncompute* them. For a circuit with s bits and t gates, we need at most t additional gates to do this.

Example 2.2.3. Let's uncompute the intermediate bits in the circuit of example 2.7, so that they can be reused later in the circuit. Our workspace bits will be called t_0, t_1, t_2 .

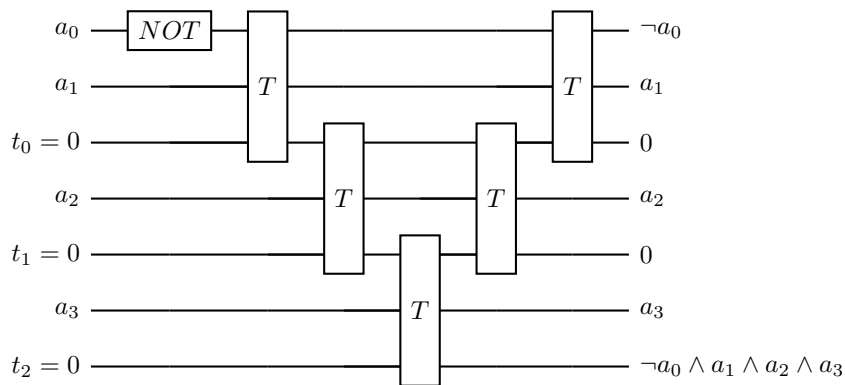


Figure 2.8: A reversible classical circuit

Substituting Toffoli gates and uncomputing intermediate bits is the naive approach. This is $\mathcal{O}(s + t)$ memory. Next we will show a more efficient construction.

2.2.2 Composing subcircuits

The more efficient construction involves partitioning the circuit C , with t gates and s bits, $r = \lceil t/s \rceil$ sub circuits. We will refer to each as C_i for $1 \leq i \leq r$, such that $C = C_1 C_2 \dots C_r$. The inefficient way of making this circuit reversible as shown in the above examples would concretely look like this.

1. For each C_i , make it reversible by substituting AND gates with Toffoli gates. Call the reversible subcircuit R_i . This new subcircuit has at most s more bits.
2. Copy the values used in later parts of the computation to an output register. Once again, this adds at most s bits.
3. Perform the sequence of gates from step 1 in reverse order, to reset all bits except the output register bits to their input values. By doing this, all intermediate bits have been uncomputed to 0 and can be reused.

The idea behind our new, more efficient construction is now the following: Combine the circuits R_i in a smart way. By choosing to uncompute a subcircuit, we add more gates but reduce the need to create new intermediate bits.

Theorem 2.2.1. *Every classical circuit C with t gates and s bits can be made reversible using $\mathcal{O}(t^{\log_2 3})$ gates and $\mathcal{O}(s \log t)$ bits.*

Proof. Let $C = C_1 C_2 \dots C_r$ be a circuit with t gates and s bits. For simplicity, we will prove only the case where $r = \lceil t/s \rceil = 2^k$ for some $k \in \mathbb{Z}^+$. Also let $r_i = 2^i$. Let $R = R_1 R_2 \dots R_r$ be the inefficient reversible circuit, made from C with the steps specified above. Consider the recursively defined transformation $\mathcal{B} : R \rightarrow R'$ for circuits R and R' .

$$\begin{aligned} \mathcal{B}(R_1 R_2 \dots R_{r_{i+1}}) &= \mathcal{B}(R_1 R_2 \dots R_{r_i}) \mathcal{B}(R_{r_i+1} R_{r_i+2} \dots R_{r_{i+1}}) (\mathcal{B}(R_1 R_2 \dots R_{r_i}))^{-1} \\ \mathcal{B}(R) &= R. \end{aligned}$$

For a reversible circuit R represented by some function $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$, R^{-1} is represented by f^{-1} . The act of applying R^{-1} is what we refer to as 'uncomputing' the circuit. The last term $(\mathcal{B}(R_1 R_2 \dots R_{r_i}))^{-1}$ acts on the same bits as the first $\mathcal{B}(R_1 R_2 \dots R_{r_i})$, and does not require additional space.

The point of this transformation is that it uncomputes every subcircuit R_i except R_r , which outputs the result of the circuit. We know $\mathcal{B}(R_1 R_2 \dots R_{i+1})$ uses at most s more bits than $\mathcal{B}(R_1 R_2 \dots R_i)$, meaning that we may find a bound for the amount of bits used recursively in the following way. Let $S(i)$ mean the space requirement for each of the steps $1 \leq i \leq k = \log_2 r$. Then we know by the previous observation that $S(i) \leq S(i-1) + s$. We also know $S(1) \leq 2s$.

This of course means that $S(k) \leq (k+1)s = s(\log_2 r + 1)$. Thus if we have t steps, we need $\mathcal{O}(s \log_2 t)$ space. As for the gates, we solve it similarly. Let $T(i)$ be the number of circuits executed by $\mathcal{B}(R_1, \dots, R_{r_i})$. Then $T(i) = 3T(i-1)$ because we have 3 computations in (1.15). Also $T(1) = 1$ because $\mathcal{B}(R) = R$. Since we assumed that $r = 2^k$, we see that $T(2^k) = 3^k = 3^{\log_2 r} = r^{\log_2 3}$. We have shown that we need $\mathcal{O}(r^{\log_2 3})$ gates, which ends the proof. \square

An illustration of this way of combining subcircuits is shown below.

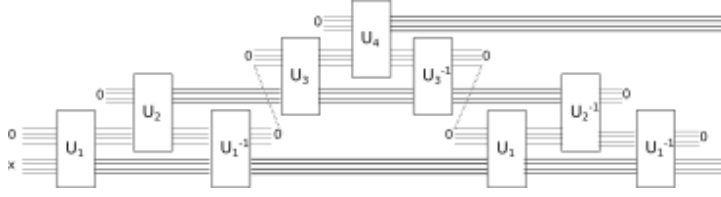


Figure 2.9: Reversible circuit that reuses bits

Example 2.2.4. For figure 2.9, we can get the same result by using the recursive transformation \mathcal{B} defined in the proof of theorem 2.2.1.

$$\begin{aligned} \mathcal{B}(U_1 U_2 U_3 U_4) &= \mathcal{B}(U_1 U_2) \mathcal{B}(U_3 U_4) (\mathcal{B}(U_1 U_2))^{-1} = \\ \mathcal{B}(U_1) \mathcal{B}(U_2) (\mathcal{B}(U_1))^{-1} \mathcal{B}(U_3) \mathcal{B}(U_4) (\mathcal{B}(U_3))^{-1} (\mathcal{B}(U_1) \mathcal{B}(U_2) (\mathcal{B}(U_1))^{-1})^{-1} &= \\ U_1 U_2 U_1^{-1} U_3 U_4 U_3^{-1} U_1 U_2^{-1} U_1^{-1}. \end{aligned}$$

Instead of arbitrarily deciding that we may only have $r = 2^k$ partitions, inspired by the technique used in the last proof we can make a recursively defined transformation that partitions the sequence of circuits into m equally large parts instead of only 2. By doing this, we attain another bound.

Theorem 2.2.2. Let C be a classical circuit with t gates and s bits. It has a reversible counterpart with $\mathcal{O}(t^{1+\epsilon})$ gates and $\mathcal{O}(s \log t)$ bits.

Proof. Suppose $r = m^k$ for $m, k \in \mathbb{Z}^+$, and let $r_i = m^i$. Examine the notation $\vec{R}_{x,i} = R_{1+(x-1)r_i} R_{2+(x-1)r_i} \dots R_{x r_i}$. Convince yourself that $\vec{R}_{1,i+1} = \vec{R}_{1,i} \vec{R}_{2,i} \dots \vec{R}_{m,i}$.

Consider the following recursive transformation $\mathcal{B} : R \rightarrow R'$ for reversible circuits R and R' .

$$\begin{aligned} \mathcal{B}(R_{1,i+1}) &= \mathcal{B}(\vec{R}_{1,i} \vec{R}_{2,i} \dots \vec{R}_{m,i}) = \\ \mathcal{B}(\vec{R}_{1,i}) \mathcal{B}(\vec{R}_{2,i}) \dots \mathcal{B}(\vec{R}_{m,i}) \mathcal{B}(\vec{R}_{m-1,i})^{-1} (\vec{R}_{m-2,i})^{-1} \dots (\vec{R}_{1,i})^{-1}, \\ \mathcal{B}(R) &= R. \end{aligned}$$

In each step, one of the m parts $\vec{R}_{k,i}$ is replaced by $2m-1$ transformations. Since $r = m^k$, the recursion takes k steps. This implies that we have a total of $(2m-1)^k$ reversible subcircuits R_i . Rewriting this in terms of r gives us

$$(2m-1)^{\log_m r} = r^{\log_m 2m-1} \approx r^{\log_m 2m} = r^{1 + \frac{1}{\log_2 m}}$$

Any reversible circuit R_i constructed from C_i may be made reversible with at most $3s$ gates, according to the steps outlined at the start of section 2.2.2. Because of this, we once again define $T(t)$ as the total number of gates for a reversible circuit of t gates and say

$$T(t) \approx 3s \left(\frac{t}{s} \right)^{1 + \frac{1}{\log_2 m}} < 3t^{1 + \frac{1}{\log_2 m}}.$$

For any $\epsilon > 0$, we can choose m large enough so that we only need $\mathcal{O}(t^{1+\epsilon})$ gates for a reversible construction. The amount of bits remains the same as from the previous proof, $\mathcal{O}(s \log_2 t)$. \square

From classical to quantum

With this theorem, it's clear that any classical circuit can be computed in a reversible way with comparable efficiency. Because we may construct any reversible classical circuit from AND and Toffoli gates, the implementation translates to quantum circuits since both gates may be implemented as quantum gates. Thus they can be used in quantum circuits without significant efficiency loss.

2.2.3 Quantum circuit complexity

In the coming chapters we will showcase quantum algorithms which solve problems in polynomial time which classical algorithms may only solve in exponential. Before moving on we will provide the reader with an incomplete, but for our purposes sufficient definition of the time complexity of a quantum circuit. We make the following definitions.

Definition 2.2.2. Simple quantum gate

A simple quantum gate is a gate A is one of the gates $R(\beta)$, $T(\alpha)$, $K(\delta)$ or C_{NOT} for some parameter $\alpha, \beta, \delta \in \mathbb{R}$.

The time complexity of a quantum circuit is defined by how many simple quantum gates it has.

Definition 2.2.3. Time complexity

A quantum circuit is said to have time complexity $\mathcal{O}(f(n))$ where n is the amount of input qubits and $f(n)$ is the amount of simple gates used.

Chapter 3

The Finite Abelian Hidden Subgroup Problem

In order to give context to the definition of the quantum Fourier transform, we would like to take a section to change the scope of what we're trying to solve. The context is given by the problem of finding generators to a subgroup H of a finite Abelian group G , where H is implicitly defined by a function on G . In order to be more formal, we need the definition.

Definition 3.0.1. *Constant within each coset, distinct between each coset*

Assume a group G and a subgroup $G > H$, let gH be some coset for $g \in G$. A function $f : G \rightarrow S$ for some set S is said to be constant within each coset and distinct between each coset if $f(\tilde{g}_1) = f(\tilde{g}_2)$ if and only if $\tilde{g}_1, \tilde{g}_2 \in gH$.

Below is a more formal description of our problem. för tidigt för order finding

Definition 3.0.2. Hidden Subgroup Problem

Assume a group G and a function $f : G \rightarrow S$ where S is some finite set. Suppose f defines a subgroup H such that f is constant within each coset and distinct between each one.

The Hidden Subgroup Problem (HSP) consists of finding H . This is done by finding a subset that generates H .

The difficult part of Shor's problem can be reformulated in terms of the HSP for finite Abelian groups. It is in the context of solving HSP for finite Abelian groups that we introduce the Quantum Fourier transform.

3.1 Algebraic representation of groups

We start off with some definitions from group theory. Definitions are taken from [5].

Definition 3.1.1. Algebraic representation

Let G be any group and X a complex vector space. We denote $GL(X)$ as the group of invertible, linear mappings that carry X into itself.

A representation is a group homomorphism

$$\begin{aligned}\chi : G &\rightarrow GL(X) \\ g &\rightarrow \chi(g).\end{aligned}$$

In this section, we mostly care about Abelian groups. In the case of Abelian groups we will only need to consider group homomorphisms $\chi : G \rightarrow \mathbb{C}$ where \mathbb{C} is the multiplicative group of complex numbers. Because every Abelian group can be decomposed into a product of cyclic groups (as we will show later) we will mostly be dealing with $G = \mathbb{Z}_n$ and products of \mathbb{Z}_n . The representations for \mathbb{Z}_n are called *characters*.

For \mathbb{Z}_n we can form the complete set of representations

$$\chi_j : x \rightarrow \exp\left(\frac{2\pi i}{n} jx\right)$$

for each $j \in \{0, 1, \dots, n-1\}$. This is because we map the identity element 0 to the identity element 1. Furthermore we must map 1 to one of the n roots of unity since $n \equiv 0$ in \mathbb{Z}_n . After all, a representation χ is a homomorphism such that $\chi(1)^n = \chi(1 + 1 + \dots + 1) = \chi(n) = \chi(0) = 1$. This implies that there can be no more than n representations of \mathbb{Z}_n and that they also must be of the form above. When dealing with products of \mathbb{Z}_n , say $\mathbb{Z}_{n_1} \times \dots \times \mathbb{Z}_{n_k}$, the complete set of representations is given by

$$\chi((g_1, \dots, g_k)) = \chi_1(g_1) \dots \chi_k(g_k),$$

where χ_1, \dots, χ_k may be any of the representations for $\mathbb{Z}_{n_1}, \dots, \mathbb{Z}_{n_k}$ respectively. The complete set of characters of a group G with the operation of pointwise multiplication, $\chi(g) = \chi_1(g) \circ \chi_2(g) = (\chi_1 \chi_2)(g) \forall g \in G$, is called the *dual group* \widehat{G} with the inverses given by $\chi_i(g) = \frac{1}{\chi_i(g)} \forall g \in G$. If H is a subgroup of G , then the following is a subgroup of \widehat{G} :

$$H^\perp = \{\chi \in \widehat{G} \mid \chi(h) = 1 \forall h \in H\}.$$

We see that it's a subgroup by using the subgroup test, $(\chi\omega^{-1})(h) = 1$ for all $\chi, \omega \in \widehat{G}$ and for all $h \in H$. The subgroup H^\perp has $|G : H|$ members because it's the representations of G which map all elements of H to 1. The reason we care about H^\perp is because of the fairly simple property given below.

Proposition 3.1.1. *Let G be a group and $G > H$. Then $(H^\perp)^\perp \cong H$.*

Proof. We mentioned above that H^\perp has $|G : H|$ elements if $G > H$. It follows that $(H^\perp)^\perp$ has $|G : H^\perp| = |H|$ elements.

Next, we show that every element of H is contained in $(H^\perp)^\perp$,

$$\begin{aligned}(H^\perp)^\perp &= \{g' \in G \mid g'(\chi_g) = 1, \forall g \in H^\perp\} \\ &= \{g' \in G \mid \chi_g(g') = 1, \forall g \in H^\perp\}.\end{aligned}$$

By definition, all elements of H have this property. Thus $(H^\perp)^\perp \cong H$. \square

In order to define the quantum Fourier transform we need a fundamental lemma called Schur's lemma.

Lemma 3.1.1. Schur's Lemma

Let χ_1 and χ_2 be elements of \widehat{G} for some finite group G where $\chi_1 \neq \chi_2$. Then,

$$\sum_{g \in G} \chi_1(g) \overline{\chi_1(g)} = |G|$$

and

$$\sum_{g \in G} \chi_1(g) \overline{\chi_2(g)} = 0.$$

Proof. The first statement is obvious when you consider $\overline{\chi_1(g)} = \frac{1}{\chi_1(g)} \forall g \in G$. For the second statement we reason in the following way. Since $\chi_1 \neq \chi_2$ we can take an element $h \in G$ such that $\chi_1(h) \neq \chi_2(h)$. Then:

$$\begin{aligned} \chi_1(h) \sum_{g \in G} \chi_1(g) \overline{\chi_2(g)} &= \sum_{g \in G} \chi_1(h) \chi_1(g) \overline{\chi_2(g)} \\ &= \sum_{g \in G} \chi_1(hg) \overline{\chi_2(h^{-1}hg)} \\ &= \sum_{g \in G} \chi_1(g) \overline{\chi_2(h^{-1}g)} \\ &= \sum_{g \in G} \chi_1(g) \chi_2(h) \overline{\chi_2(g)} \\ &= \chi_2(h) \sum_{g \in G} \chi_1(g) \overline{\chi_2(g)}. \end{aligned}$$

Since $\chi_1(h) \neq \chi_2(h)$, we conclude that

$$\sum_{g \in G} \chi_1(g) \overline{\chi_2(g)} = 0.$$

\square

A useful corollary to this is Schur's lemma for subgroups.

Corollary 3.1.1.1. Schur's lemma for subgroups *Let G be a finite Abelian group with subgroup H , and χ some representation of G . Then*

$$\sum_{h \in H} \chi(h) = \begin{cases} |H|, & \text{if } \forall h \text{ in } H \chi(h) = 1 \\ 0, & \text{otherwise} \end{cases}$$

Proof. Apply Schur's lemma restricted to χ , since it's also a representation of H . \square

Next we will be defining the quantum Fourier transform. fixa prop 3.0.2.

3.2 Fourier basis and quantum Fourier transform

For every group $G = \mathbb{Z}_n$, $|G| = n$, we will associate an n -dimensional complex vector space \mathbb{C}^n with the standard basis \mathcal{G} . Given standard ordering, we denote each basis vector by an element in the group G in the following way,

Let $\mathcal{G} = \{|g_0\rangle, |g_1\rangle, \dots, |g_{n-1}\rangle\}$ be the standard basis.

Consider the following set: $\mathcal{B} = \{|e_i\rangle \mid i \in G\}$ such that

$$|e_i\rangle = \frac{1}{\sqrt{n}} \sum_{g \in G} \overline{\chi_i(g)} |g\rangle.$$

Proposition 3.2.1. \mathcal{B} is an orthonormal basis for \mathbb{C}^n

Proof. Consider the inner product of two elements $|e_i\rangle$ and $|e_j\rangle$ where $i \neq j$,

$$\langle e_i | e_j \rangle = \left(\frac{1}{\sqrt{n}} \sum_{g \in G} \overline{\chi_i(g)} |g\rangle \right) \left(\frac{1}{\sqrt{n}} \sum_{h \in G} \overline{\chi_j(h)} |h\rangle \right) \quad (3.1)$$

$$= \frac{1}{n} \sum_{g \in G} \sum_{h \in G} \overline{\chi_i(g)} \overline{\chi_j(h)} \langle g | h \rangle \quad (3.2)$$

$$= \frac{1}{n} \sum_{g \in G} \overline{\chi_i(g)} \chi_j(g) = 0 \quad (3.3)$$

Also $\langle e_i | e_i \rangle = 1$ because

$$\langle e_i | e_i \rangle = \overline{\left(\frac{1}{\sqrt{n}} \sum_{g \in G} \overline{\chi_i(g)} |g\rangle \right)} \left(\frac{1}{\sqrt{n}} \sum_{g \in G} \overline{\chi_i(g)} |g\rangle \right) \quad (3.4)$$

$$= \frac{1}{n} \sum_{g \in G} \chi_i(g) \overline{\chi_i(g)} \langle g | g \rangle \quad (3.5)$$

$$= \frac{1}{n} \sum_{g \in G} \chi_i(g) \overline{\chi_i(g)} \quad (3.6)$$

$$= \frac{n}{n} = 1. \quad (3.7)$$

□

The equalities (3.3) and (3.7) are true because of Schur's Lemma. Now finally we are ready to define the quantum Fourier transform over Abelian groups

Definition 3.2.1. *The quantum Fourier transform for Abelian groups* The quantum Fourier transform is the transformation \mathcal{F} that maps the basis vector $|g\rangle$ to $|e_g\rangle \forall g \in G$. In other words,

$$\mathcal{F} = \sum_{g \in G} |e_g\rangle \langle g|.$$

A general n -qubit quantum circuit implementation of the quantum Fourier transform is given in figure 3.1. Here we use the notation $[0.x_1x_2 \dots x_k] = \frac{[x_1x_2 \dots x_k]}{2^k}$ where $[x_1x_2 \dots x_k]$ is the binary representation of some positive integer. We let

$$R_m = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^m}} \end{bmatrix} \quad (3.8)$$

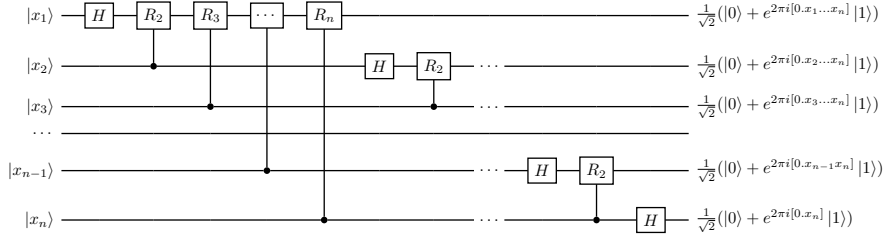


Figure 3.1: A quantum circuit implementation of the quantum Fourier transform for n qubits.

Recall our description of controlled gates in chapter 2.1.2 and figure 2.2. Loosely speaking, these are of great importance in the quantum Fourier transform because they provide a means of entanglement of the qubits. This cannot be done with only 1-gates. To see that the quantum Fourier transform can be written in the form shown in figure 3.1, we need a bit of algebra.

Proposition 3.2.2. *Writing $|j\rangle = |j_1j_2 \dots j_n\rangle$, for some n qubit system where we identify each basis vector $|j\rangle$ with the standard basis, the quantum Fourier transform may be expressed in the following way:*

$$|j_1j_2 \dots j_n\rangle \rightarrow \frac{(|0\rangle + e^{2\pi i 0.j_n} |1\rangle) \otimes (|0\rangle + e^{2\pi i 0.j_{n-1}j_n} |1\rangle) \otimes \dots \otimes (|0\rangle + e^{2\pi i 0.j_1 \dots j_n} |1\rangle)}{2^{n/2}}.$$

Proof. By definition of the quantum Fourier transform,

$$|j\rangle \rightarrow \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i jk/2^n} |k\rangle$$

for which the right-hand side can be written as the product of sums in terms of the fractional binary notation of k ,

$$\begin{aligned} & \frac{1}{2^{n/2}} \sum_{k_1=0}^1 \dots \sum_{k_n=0}^1 e^{2\pi i j (\sum_{l=1}^n k_l 2^{-l})} |k_1 \dots k_n\rangle \\ &= \frac{1}{2^{n/2}} \sum_{k_1=0}^1 \dots \sum_{k_n=0}^1 \bigotimes_{l=1}^n e^{2\pi i j (k_l 2^{-l})} |k_l\rangle. \end{aligned}$$

The tensor product is distributive, so we can rewrite it as

$$\begin{aligned} & \frac{1}{2^{n/2}} \bigotimes_{l=1}^n \left[\sum_{k_l=0}^1 e^{2\pi i j k_l 2^{-l}} |k_l\rangle \right] = \frac{1}{2^{n/2}} \bigotimes_{l=1}^n \left[|0\rangle + e^{2\pi i j 2^{-l}} |1\rangle \right] \\ & = \frac{(|0\rangle + e^{2\pi i 0 \cdot j_n} |1\rangle) \otimes (|0\rangle + e^{2\pi i 0 \cdot j_{n-1} j_n} |1\rangle) \otimes \dots \otimes (|0\rangle + e^{2\pi i 0 \cdot j_1 \dots j_n} |1\rangle)}{2^{n/2}}. \end{aligned}$$

Which is what we wanted to prove. \square

Example 3.2.1. A circuit implementation of the quantum Fourier Transform is given below.

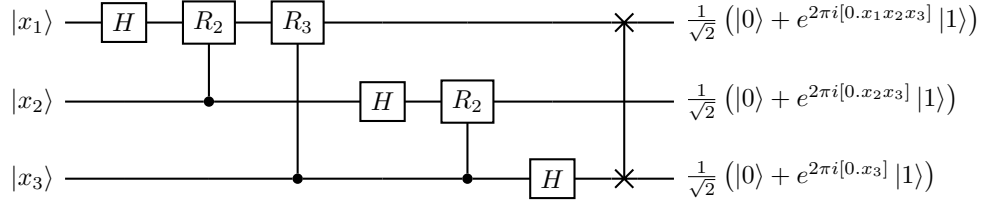


Figure 3.2: A quantum circuit implementation of the quantum Fourier transform for $n = 3$ qubits.

We swap the first and last qubit to obtain the correct output order.

Example 3.2.2. The n – qubit quantum Fourier transform may also be expressed as a unitary matrix,

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \dots & \omega^{(n-1)^2} \end{bmatrix}$$

where $\omega = e^{\frac{2\pi i}{n}}$.

Every unitary matrix has its inverse. In the case of the quantum Fourier transform we will always refer to it simply as the inverse quantum Fourier transform.

Definition 3.2.2. Inverse quantum Fourier transform

The inverse quantum Fourier transform, or the inverse quantum Fourier transform is

$$\mathcal{F}^{-1} = \sum_{g \in G} |g\rangle \langle e_g|.$$

Complexity of the quantum Fourier transform

The quantum Fourier transform is what provides the exponential speedup for most quantum algorithms. The circuit in figure 3.1. requires $(n+1)+\dots+2+1$ Hadamard and controlled phase shift gates. Using big-O notation, the quantum Fourier transform requires $\mathcal{O}((n+1)+\dots+2+1) = \mathcal{O}(n^2)$ simple gates. Compare this with the best known time complexity of the discrete (classical) Fourier transform $\mathcal{O}(n2^n)$, the speed up is exponential. This makes the quantum Fourier transform an incredibly powerful tool to have when designing quantum algorithms.

The catch here is of course that one does not actually know the amplitudes when using the quantum Fourier transform. After all, postulate 3 tells us that by measuring the state we inevitably also collapse it.

3.3 The Fundamental Theorem of Finite Abelian Groups

3.1.1. G is abelian As mentioned earlier, when dealing with algebraic representations of groups our main interests are products of cyclic groups. The reason for this being that, as it turns out, every finite Abelian group can be decomposed into a direct product of cyclic groups of prime order.

Seeing how this theorem is such a fundamental component in solving the HSP, we will dedicate this section to building up some background to the theorem in order to prove it. The proofs and theorems that follow are taken from Judson[8]. Consider the following definitions, the first one being a generalization of Abelian groups.

Definition 3.3.1. Generated group

Let S be a set of elements $g_i \in G$ for some Abelian group G . The smallest subgroup H of G containing all the elements in S is the group **generated** by S .

Proposition 3.3.1. Let H be the subgroup in the Abelian group G generated by the set $S = \{g_1, g_2, \dots\}$. Then $h \in H \Leftrightarrow h = g_{i_1}^{\alpha_1} g_{i_2}^{\alpha_2} \dots g_{i_n}^{\alpha_n}$ for some $n \in \mathbb{N}$ and $\alpha_1, \dots, \alpha_n \in \mathbb{Z}$. In other words, when h is a product of generators to some exponent.

Proof. It's clear that elements of the form $g_{i_1}^{\alpha_1} g_{i_2}^{\alpha_2} \dots g_{i_n}^{\alpha_n}$ are in H , since it's the smallest subgroup of G that contains all elements in S . It remains to show the other direction of the arrow.

Let K be the set of all elements of the form $g_{i_1}^{\alpha_1} g_{i_2}^{\alpha_2} \dots g_{i_n}^{\alpha_n}$. Clearly this set contains all elements in S , so if K is a group (with respect to multiplication) then $H = K$ since we already showed $K \subset H$. The identity element is contained in this set, since $g_{i_k}^0 = 1$. The inverse of any element $g_{i_1}^{\alpha_1} g_{i_2}^{\alpha_2} \dots g_{i_n}^{\alpha_n}$ is just $g_{i_1}^{-\alpha_1} g_{i_2}^{-\alpha_2} \dots g_{i_n}^{-\alpha_n}$, and it's clear that it's closed under the group operation. Associativity follows from the multiplication operation. \square

Definition 3.3.2. p -group

A p -group is a group G such that $|g| = p^\alpha$ for some α for every $g \in G$, where $|g|$ refers to the order of the element g in G .

With the second definition in mind, we will prove a special case of the fundamental theorem of finite Abelian groups.

Proposition 3.3.2. *Every finite Abelian group is isomorphic to a direct product of p -groups.*

Proof. Suppose G is a finite Abelian group. The case $|G| = 1$ is trivial. Let $|G| > 1$ and $|G| = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_n^{\alpha_n}$ for some unique prime decomposition. Define G_i to be the set of all $g \in G$ such that the order of g is p_i^α for some $\alpha \geq 0$.

We want to show that $G \cong G_1 \times G_2 \times \dots \times G_n$, since this is a direct product of p -groups. By Lagrange's theorem, the order g of an element of a group divides the order of the group. Write

$$|g| = p_1^{\beta_1} p_2^{\beta_2} \dots p_n^{\beta_n}$$

for some element in G . Now let $a_i = |g|/p_i^{\beta_i}$, which means $\gcd(a_1, a_2, \dots, a_n) = 1$. Then there are coefficients b_i such that $a_1 b_1 + a_2 b_2 + \dots + a_n b_n = 1$. With this in mind, consider

$$g = g^{a_1 b_1 + a_2 b_2 + \dots + a_n b_n} = g^{a_1 b_1} g^{a_2 b_2} \dots g^{a_n b_n}.$$

For each term in this product, notice how $g^{a_i b_i p_i^{\beta_i}} = g^{|g| b_i} = e$. This of course means that each $g_i = g^{a_i b_i}$ belongs to its respective subgroup G_i , and that every g has a decomposition $g = g_1 g_2 \dots g_n$.

It remains to show that this decomposition is unique. Let $g = h_1 h_2 \dots h_n$. Then $e = g_1 g_2 \dots g_n (h_1 h_2 \dots h_n)^{-1} = g_1 h_1^{-1} g_2 h_2^{-1} \dots g_n h_n^{-1}$. We observe that every $g_i h_i^{-1}$ must be of order p_i^α for some α . This means that the order of $g_1 h_1^{-1} g_2 h_2^{-1} \dots g_n h_n^{-1}$ is the greatest common product of the orders $g_i h_i^{-1}$, which of course is e . This proves uniqueness. Thus $G \cong G_1 \times G_2 \times \dots \times G_n$. \square

Lemma 3.3.1. *Every finite Abelian p -group G can be written $\langle g \rangle \times H$ for some $g \in G$ of maximal order and for some subgroup H of G .*

Proof. Let's proceed by induction. If $|G| = p^n$ for the case $n = 1$, then we can write G as $\langle g \rangle \times H$ for some generator g and $H = \{e\}$. Now assume the lemma is true for all $1 \leq k < n$, and let g be an element in G of maximal order, say $|g| = p^m$. This means for all $a \in G$, $a^{p^m} = e$. We know that $\langle g \rangle \neq G$ since otherwise we would be done. We can assume that there is an element $h \notin \langle g \rangle$ of smallest possible order. We define $H = \langle h \rangle$. Next we want to show that these groups are disjoint except for the identity element, $\langle g \rangle \cap H = \{e\}$. To show this, it actually suffices to show that $|H| = p$.

We know $|h^p| = \frac{|h|}{p}$, so $h^p \in \langle g \rangle$ because h is by definition the element with the smallest order that is not in $\langle g \rangle$. Thus $h^p = g^x$ for some integer x . Then the following is true,

$$(g^x)^{p^{m-1}} = (h^p)^{p^{m-1}} = h^{p^m} = e.$$

Also $|g^r| \leq p^{m-1}$. Because of this, g^r cannot generate $\langle g \rangle$. Notice that r is a multiple of p , say $r = ps$, so $h^p = g^r = g^{ps}$. Let $a = g^{-s}h$, clearly a is not in $\langle g \rangle$ because h is not in $\langle g \rangle$. Consider how

$$a^p = g^{-sp}h^p = g^{-r}h^p = h^{-p}h^p = e.$$

Thus a is an element with order p such that $a \notin \langle g \rangle$. But h is by definition such an element with the smallest order, so all elements that are not in $\langle g \rangle$ must have order p . Thus $|H| = p$. Finally we will show that $|gH|$ has the same order as $g \in G$. Assume by contradiction that $|gH| < |g| = p^m$. This means

$$H = (gH)^{p^{m-1}}H$$

which means $g^{p^{m-1}} \in \langle g \rangle \cap H = \{e\}$. But then the order of g can't be p^m . Thus gH has maximal order in G/H . Using the correspondence theorem (we will not prove this) and by our induction hypothesis,

$$G/H \cong \langle gH \rangle \times K/H.$$

for $K < G$ containing H . Let $b \in \langle g \rangle \cap K$, this implies that $bH \in \langle gH \rangle \cap K/H = \{H\}$ and $b \in \langle g \rangle \cap H = \{e\}$. It follows that $G = \langle g \rangle K \implies G \cong \langle g \rangle \times K$. \square

Finally we are ready to present the fundamental theorem. It follows easily from the previous results.

Theorem 3.3.2. Fundamental Theorem of Finite Abelian Groups
Every finite Abelian group is isomorphic to a product of cyclic groups of the following form,

$$\mathbb{Z}_{p_1}^{\alpha_1} \times \mathbb{Z}_{p_2}^{\alpha_2} \times \dots \times \mathbb{Z}_{p_n}^{\alpha_n}.$$

Proof. Every finite Abelian group is isomorphic to a product of p-groups

$$G_1 \times G_2 \times \dots \times G_n.$$

Here $|G_i| = p_i^{\alpha_i}$ for the corresponding element in the prime decomposition of $|G| = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_n^{\alpha_n}$.

By the lemma above, each G_i is isomorphic to $\langle g \rangle \times H$ for some maximal order element g and subgroup $H \in G$. Of course, H being a subgroup of G implies that it also is a p-group. Applying the same lemma to H we can further decompose this until $H = e$. This is true because G is finite. Doing this for each component G_i gives us our desired decomposition. \square

3.4 Solving the Finite Abelian Hidden Subgroup Problem

In this section we will give a general procedure to solving the finite Abelian HSP. As reference we will be using[5].

Assume we are given a finite Abelian group G where $|G| = n$, and a function f that implicitly defines H as in definition 3.0.1. Let the two n -qubit input be $|\psi_0\rangle = |0\rangle^{\otimes n} \otimes |0\rangle^{\otimes n}$, the first n qubits we refer to as the first register and the other n as the second register. It's common to refer to the the gate which computes the function f as U_f , or as "the oracle". It acts on two registers like

$$|x\rangle |0\rangle \rightarrow |x\rangle |f(x)\rangle.$$

Rather than finding H itself, the solution finds generators for H^\perp , which can later be used to find H since $(H^\perp)^\perp \cong H$. The solution consists of four steps.

Initialization

The $H^{\otimes n}$ gate is applied to the first register,

$$|\psi_1\rangle = \frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle |0\rangle^{\otimes n}.$$

Applying the oracle

We simply apply the oracle to the state in order to store the function output in the second register. We end up with a sum of cosets,

$$|\psi_2\rangle = U_f |\psi_1\rangle = \frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle |f(g)\rangle.$$

Notice that the state has become entangled.

Measuring second register

By measuring the second register, the first register collapses into some element in the same coset as the measured register,

$$|\psi_3\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} \tilde{g}h.$$

This measurement really just yields a random $f(\tilde{g})$ for some $\tilde{g} \in G$, since each coset has the same probability of being chosen.

The quantum Fourier transform

Apply the quantum Fourier transform to $|\psi_3\rangle$:

$$\begin{aligned} |\psi_4\rangle = \mathcal{F}(|\psi_3\rangle) &= \mathcal{F}\left(\frac{1}{\sqrt{|H|}} \sum_{h \in H} |\tilde{g}h\rangle\right) \\ &= \frac{1}{\sqrt{|G||H|}} \sum_{h \in H} \sum_{g \in G} \chi_g(\tilde{g}h) |g\rangle. \end{aligned}$$

χ_g is a group homomorphism, so we can also write it

$$\begin{aligned} & \frac{1}{\sqrt{|G||H|}} \sum_{h \in H} \sum_{g \in G} \chi_g(\tilde{g}) \chi_g(h) |g\rangle \\ &= \frac{1}{\sqrt{|G||H|}} \sum_{g \in G} \chi_g(\tilde{g}) \left(\sum_{h \in H} \chi_g(h) \right) |g\rangle. \end{aligned}$$

From Schur's lemma for subgroups (3.3.1.1.), we know that $\sum_{h \in H} \chi_g(h) \neq 0$ only when $\chi_g(h) = 1 \forall h \in H$. By definition, all such g are members of H^\perp . So measuring the state at this point returns an index g from some random element χ_g in H^\perp , which is precisely what we want. We already know that every finite Abelian group is isomorphic to some kind of product of cyclic groups. If we let G be such a product $(\mathbb{Z}_{p_1}^{\alpha_1}, \dots, \mathbb{Z}_{p_n}^{\alpha_n})$, measuring $|\psi_4\rangle$ would return a product of such indexes (g_1, \dots, g_n) where $g_i \in \mathbb{Z}_{p_i}^{\alpha_i}$.

In the next section, we will finally make use of this generalization. We will see that some important quantum algorithms, including Shor's algorithm, may be formulated in terms of the finite Abelian HSP.

Chapter 4

Quantum Algorithms

In this chapter we present two important quantum algorithms. First we include Simon's algorithm to show a common approach for quantum algorithms. Second we show Shor's algorithm. Order finding, perhaps conceptually the hardest part of Shor's algorithm, can be seen as a finite Abelian HSP. Simon's problem is also such a special case. We take time to develop tools that better our understanding of order finding, like *quantum phase estimation*.

The goal of this chapter is to provide the reader with an in depth understanding of these quantum algorithms in the context of showcasing quantum computational techniques. This means that the classical parts of the algorithms, while important, will not be our focus.

4.1 Simon's problem

While having a solution to Simon's problem isn't very interesting in itself, the techniques used in the solution definitely are and eventually came to inspire Shor's algorithm. Consider the definition, which we will use in the problem formulation.

Definition 4.1.1. XOR operation

Let $x = x_1x_2 \dots x_k, y = y_1y_2 \dots y_k \in \mathbb{Z}_2^k$. The XOR between these elements $x \oplus y$ is an element in $c = c_1c_2 \dots c_k \in \mathbb{Z}_2^k$ such that $c_i = x_i + y_i \pmod 2$.

Definition 4.1.2. Simon's problem

Let $f : \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^n$ be a two-to-one function such that there exists an $s \in \mathbb{Z}_2^n, s \neq 0$ with the following property:

$f(x) = f(y) \Leftrightarrow x \oplus y \in \{0, s\}$. Where \oplus is the XOR operation.

Simon's problem is finding s .

Example 4.1.1. A two-to-one function f is given by

$$\begin{aligned} 00 &\rightarrow 11 \\ 10 &\rightarrow 10 \\ 01 &\rightarrow 10 \\ 11 &\rightarrow 11 \end{aligned}$$

We can visually verify that $f(10) = f(01) = 10$. This means that $10 \oplus 01 = 11 \in \{00, s\}$. Thus the solution is $s = 11$.

For cases $n = 2$, it's easy to find a solution. This becomes exponentially harder as n increases.

Classically, the lower bound for solving this problem is $\mathcal{O}(2^{N/2})$ steps for $f : \{0, 1\}^N \rightarrow \{0, 1\}^N$ [9]. The quantum solution offers exponential speed-up, which is undeniably impressive.

Simon's problem can be formulated in terms of the finite Abelian HSP. Set $G = \mathbb{Z}_2^n$ with the XOR operation \oplus and a subgroup $H = \{0^{\otimes n}, s\}$. Of course, f is the function that implicitly defines the subgroup H , which we must find generators for.

Just to illustrate, we follow the procedure outlined in the previous chapter.

Simon's algorithm

Assume we have a case of Simon's problem and we are provided with a function $f : \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^n$. The following probabilistic algorithm returns the solution s .

Input: Two zero registers $|0\rangle^{\otimes n} |0\rangle^{\otimes n}$. Assume we know an efficient implementation of the oracle function f, U_f .

- **Initialization:** Applying the Hadamard gates,

$$\frac{1}{2^{n/2}} \sum_{x \in \mathbb{Z}_2^n} |x\rangle |0\rangle^{\otimes n}.$$

- **Oracle:** The application of the Oracle U_f entangles the two registers.

$$\frac{1}{2^{n/2}} \sum_{x \in \mathbb{Z}_2^n} |x\rangle |f(x)\rangle.$$

- **Measurement:** By measuring the second register, we are left with a state of the form

$$\frac{1}{\sqrt{2}} (|x\rangle + |x \oplus s\rangle)$$

- **Quantum Fourier transform:** The representations for \mathbb{Z}_2^n can be written $\chi_i(j) = (-1)^{ij}$ for $i = 0, 1$. Note that the quantum Fourier transform

in this case is the Hadamard gate, and that we compute the elements x, z, s as their numeric value in the exponent.

$$\frac{1}{2^{n+1}} \sum_{z \in \mathbb{Z}_2^n} \left((-1)^{xz} + (-1)^{(x \oplus s)z} \right) |z\rangle$$

By applying Shur's lemma for subgroups, we know that the measurement of the state returns an element z_i of $H^\perp = \{z | (-1)^{s \cdot z} = 1 \forall z \in H\}$. This can also be seen by the following reasoning:

The only non-zero output of measuring the first register is the case where $(-1)^{xz} = (-1)^{(x \oplus s)z}$. This implies

$$x \cdot z = (x \oplus s) \cdot z \tag{4.1}$$

$$x \cdot z = x \cdot z \oplus s \cdot z \tag{4.2}$$

$$s \cdot z = 0 \pmod{2}. \tag{4.3}$$

Equation (4.2) is a property of the XOR operation.

Each time this process is repeated we obtain some random z_i . We wish to obtain $n - 1$ unique elements $z_i \in H^\perp$, how many times should we repeat the process to obtain this with good accuracy? If we repeat the procedure n times, the probability of choosing n unique such z_i is given by

$$\prod_{k=1}^n \left(1 - \frac{k}{2^n} \right) > \left(1 - \frac{n}{2^n} \right)^n.$$

Given n is large, this will be very close to 1. Thus we can expect to get a correct answer by repeating the algorithm $\mathcal{O}(n)$ times. It remains to solve the system of equations,

$$\begin{cases} s \cdot z_1 = 0 \pmod{2} \\ s \cdot z_2 = 0 \pmod{2} \\ \dots \\ s \cdot z_{n-1} = 0 \pmod{2} \end{cases}.$$

With $n - 1$ independent equations there are two possible solutions. The trivial solution $s = 0$ and the non-zero solution s . The system of equations may be solved with Gaussian elimination, which is $\mathcal{O}(n^2)$ steps. The quantum part takes $\mathcal{O}(n)$ steps we repeat the process roughly nk times, where k is the number of simple gates.

Theorem 4.1.1. *Simon's algorithm can solve Simon's problem in $\mathcal{O}(n^2)$ time.*

Proof. The Hadamard gates are $\mathcal{O}(n)$. The oracle is just the XOR operation, which may be done $\mathcal{O}(n)$ classically. In other words, the quantum part of Simon's algorithm is $\mathcal{O}(n)$. The bottleneck is the classical Gaussian elimination part which is $\mathcal{O}(n^2)$. \square

A quantum circuit implementation of Simon's algorithm is included in the figure below.

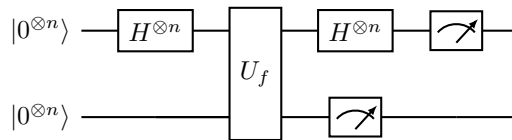


Figure 4.1: A quantum circuit illustrating the quantum part of Simon's algorithm.

Intuitively, we can see this algorithm as a way of gathering "clues" about H without actually knowing H . We find these clues as elements of a group H^\perp . After having a sufficient amount of "clues", one stops the process. Hopefully this has illustrated the power of seeing problems as special cases of the finite Abelian HSP. We will further illustrate this through Shor's algorithm.

4.2 Shor's algorithm

Unlike Simon's problem, the solution to the integer factorization problem definitely holds interest. For clarity and consistency, we will describe the problem.

Definition 4.2.1. Integer factorization problem *Given a composite number N , find the prime decomposition $N = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$. This necessarily exists because of the fundamental theorem of arithmetic.*

This is an overview of the algorithm:

For some composite number N the idea of the classical part is to find the order r of some number $1 < a < N$ such that $a^r = 1 \pmod N$. Pick another a if r ends up being odd. If a and N do not share any common factors then we may say that $(a^{r/2} + 1)(a^{r/2} - 1) = 0 \pmod N$. This is the reason we wanted an even r , so that $a^{r/2}$ would be a whole number. If neither $(a^{r/2} + 1)$ nor $(a^{r/2} - 1)$ is a multiple of N , it must mean at least one of them share a non trivial factor with N . We may find this easily with $\gcd(N, a^{r/2} + 1)$ and $\gcd(N, a^{r/2} - 1)$. Shor's algorithm solves this in two parts; A classical part and a quantum part.

The classical part

The objective of the classical part of the algorithm is to reduce the problem to the problem of *order finding*.

Definition 4.2.2. Order finding problem

Given a group G and an element $a \in G$, return the order of a .

The first three steps of the algorithm reduces the problem of factoring to the problem of order finding. Just like Simon's algorithm, Shor's algorithm is probabilistic and needs to be repeated several times in order to achieve a certain accuracy for the answer.

Shor's algorithm

It accomplishes this as follows.

1. Pick a random element $a \in \mathbb{Z}_n, a > 1$.
2. Calculate $\gcd(a, N)$ with the Euclidean algorithm
3. if $\gcd(a, N) \neq 1$ we're done (very unlikely).
4. Find the order r of a in \mathbb{Z}_n through the quantum subroutine.
5. If r is odd, go back to the first step.
6. If either $a^{r/2} + 1$ or $a^{r/2} - 1$ is a multiple of N , go back to the first step.
7. At least one of the two numbers $a^{r/2} + 1$ or $a^{r/2} - 1$ must have a non trivial factor of N . Calculate the following with the Euclidean algorithm, $\gcd(N, a^{r/2} + 1)$ and $\gcd(N, a^{r/2} - 1)$.

The only difficult part here is order-finding, which is handled by the quantum subroutine of the algorithm.

The quantum part

Finding the order of an element is no easy task classically. No classical algorithm is known to be able to do this polynomial time.

Thankfully this can be done much faster with the tools we have developed in the previous chapter. First we will describe the intuition of the solution in a high level overview. Order finding is the most important part of Shor's algorithm, and we will spend time going into the details of how it works. That being said, some classical results will be stated but not proven in order to not lose sight of the goal of the chapter. That is to showcase common quantum computational techniques for designing algorithms. We will show two different, at least in name, ways of thinking about the problem. Through *Quantum phase estimation* and as a finite Abelian HSP.

- Picking up from step 4 in the classical part, assume we have a suitable a . Let $f(x)$ be the function $f(x) = a^x \pmod N$.
- Apply U_f to the registers $|x\rangle |0\rangle$ so that we get a uniform superposition of all possible values $|x\rangle |f(x)\rangle$.
- We know $|x\rangle$ is entangled with $|f(x)\rangle$ By measuring the second register, the first register will be a uniform superposition of all $|x'\rangle$ such that $f(x') = f(x)$.
- All these values differ by a multiple of the period r . If only we could measure more than one value at one point, figuring out the period would be easier.

- Instead we apply the quantum Fourier transform. The intuition here is that we transform states into waves, that either constructively interfere to amplify the correct answer or destructively interfere to reduce the incorrect answers. After measuring one only obtains a sort of approximation of the correct answer. Even more abstractly, one can may get "clues" as to what the period r could be this way.
- Repeat this subroutine until one has sufficient "clues".

Quantum phase estimation

We will find the order of an element by using *quantum phase estimation*. By approximating some appropriately chosen phase through the process of constructive interference, one may get clues as to what the order is. More formally, this is the problem we're trying to solve.

Definition 4.2.3. Phase estimation problem Let some state $|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{i\phi k} |k\rangle$. The problem of finding an approximation $\tilde{\phi}$ of ϕ will be referred to as the *phase estimation problem*.

In this case, the problem will manifest itself as finding the eigenvalue $e^{2\pi i\phi}$ of some eigenvector $|u\rangle$ for some unitary matrix U .

Quantum phase estimation algorithm

Let $|u\rangle$ be an eigenvector to the unitary matrix U , with eigenvalue $e^{2\pi i\phi}$. for some unknown phase $\phi \in \mathbb{R}$.

1. **Initial state:** We initialize the state with two registers. The first register holds our estimation of ϕ in t for some appropriate t depending on the desired accuracy of the estimation. The second register holds our eigenvector $|u\rangle$.

$$|\psi_0\rangle = |0\rangle^{\otimes t} |u\rangle.$$

2. **Superposition:** Apply $H^{\otimes t}$ on the first register.

$$|\psi_1\rangle = \frac{1}{\sqrt{2^t}} (|0\rangle + |1\rangle) |u\rangle.$$

3. **Controlled operations:** Apply controlled- U^j operations as shown in figure 4.2. The resulting state is

$$|\psi_2\rangle = \frac{1}{\sqrt{2^t}} \left(|0\rangle + e^{2\pi i\phi 2^{t-1}} |1\rangle \right) \otimes \left(|0\rangle + e^{2\pi i\phi 2^{t-2}} |1\rangle \right) \otimes \dots \otimes \left(|0\rangle + e^{2\pi i\phi 2^0} |1\rangle \right) \otimes |u\rangle.$$

Motivation: Note how $U^{2^j} |u\rangle = U^{2^{j-1}} e^{2\pi i\phi} |u\rangle = \dots = e^{2\pi i 2^j \phi} |u\rangle$.

4. **Inverse quantum Fourier transform:** By applying the inverse quantum Fourier transform, we will get an estimation of ϕ .

$$|\psi_3\rangle = \left| \tilde{\phi} \right\rangle |u\rangle.$$

5. **Measurement** The first register is measured and we get a t -bit approximation $\tilde{\phi}$ of ϕ .

$$|\psi\rangle_4 = |\tilde{\phi}\rangle.$$

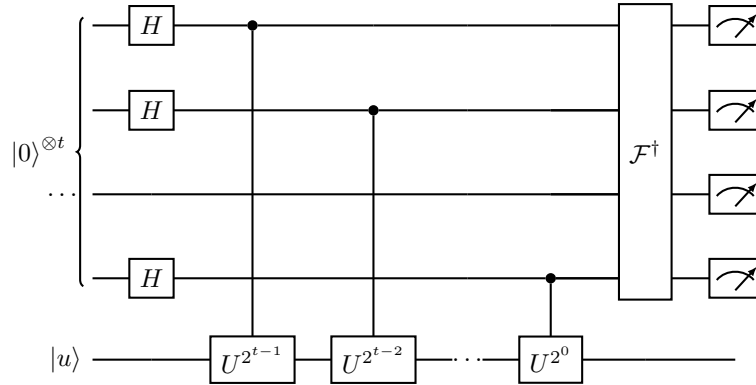


Figure 4.2: A solution to the phase estimation problem.

Order finding with quantum phase estimation

Let's show how phase estimation may be used to find the order of elements. For some element $a \in \mathbb{Z}_N$ where N is a L bit composite integer, we wish to find the order r . Consider the L -qubit operator

$$U|y\rangle \equiv \begin{cases} |a \cdot y \pmod N\rangle, & 0 \leq y \leq N-1 \\ |y\rangle, & N \leq y \leq 2^L-1 \end{cases}$$

Proposition 4.2.1. *The operator U defined above is unitary.*

Proof. One way to see this is to show $\|Ux\| = \|x\|$ for all $x \in \mathbb{C}^n$. In other words, show that U is an isometry with respect to the complex norm. Let $|b\rangle = \alpha_1|b_0\rangle + \dots + \alpha_k|b_k\rangle$ be some linear combination of basis states. Let $\alpha_i|b_i\rangle$ be one component in this linear combination, and $a \in \mathbb{Z}_n$. Note how either $U\alpha_i|b_i\rangle = \alpha_i|ab_i \pmod N\rangle$ or $U\alpha_i|b_i\rangle = \alpha_i|b_i\rangle$. Neither of which changes the length of the input vector $\alpha_i|b_i\rangle$. This implies that the size of the linear combination is preserved. Since all elements may be written as such a linear combination, it must mean that vector length is preserved with respect to the transformation. This means that U is unitary. \square

The following states are eigenvectors for U :

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i \frac{sk}{r}} |x^k \pmod N\rangle$$

for $0 \leq s \leq r - 1$. We can see this since

$$\begin{aligned} U |u_s\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i \frac{sk}{r}} |x^{k+1} \pmod N\rangle \\ &= \frac{1}{\sqrt{r}} \sum_{k=1}^r e^{-2\pi i \frac{s(k-1)}{r}} |x^k \pmod N\rangle \\ &= e^{2\pi i \frac{s}{r}} |u_s\rangle \end{aligned}$$

Why is this significant? Because the state $|u_s\rangle$ is a superposition of states which have the property that their representations differ by some multiple of r . However, we don't actually know the eigenvectors $|u_s\rangle$ because we don't know r . So how does one go about preparing the state $|u_s\rangle$? We circumvent this problem by initializing the second register as $|1\rangle^{\otimes n}$ and observing that

$$\frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} |u_j\rangle = |1\rangle^{\otimes n}.$$

The remaining problem is extracting information regarding our estimated phase $\frac{s}{r} = \left| \tilde{\phi} \right\rangle$.

Two cases

Before going any further, we let the reader know that there are two cases that we may find ourselves in while attempting to find the phase ϕ .

1. **The easy case:** ϕ has a representation $\phi = [0.\phi_1\phi_2 \dots \phi_t]$
2. **The general case:** ϕ does not have such a representation.

The approach for each case is equivalent, but differ in the probability of success. The same circuit shown in figure 4.2 is used in both cases. We will note some differences in the steps.

The easy case

For this scenario, assume that $\phi = [0.\phi_1\phi_2 \dots \phi_t]$.

With this assumption, the state $|\psi_2\rangle$ at step 3 is really just an application of the quantum Fourier transform $\mathcal{F}(|\psi_0\rangle)$. This can be seen from proposition 3.2.2. After applying the inverse quantum Fourier transform we recover the state $|\phi\rangle |u\rangle$. Upon measuring the first register we obtain $|\phi\rangle$. As already mentioned, we have prepared the input state such that $\phi = \frac{s}{r}$ for some $0 \leq s \leq r - 1$.

The general case

Assume ϕ cannot be expressed using t bits. As it turns out, we may still get a good approximation of ϕ with a high probability. This is foreshadowed by our

choice of notation $\tilde{\phi}$ when describing the order finding algorithm. The fact that we refer to this case as the general one is because most fractions between 0 and 1 cannot be represented as a t -bit binary fraction.

Theorem 4.2.1. *Let ϕ be such that it has no representation $\phi = [0.\phi_1 \dots \phi_t]$. To obtain an approximation of ϕ accurate to $2L + 1$ bits with probability $1 - \epsilon$ of success, one may choose t as*

$$t = 2L + 1 + \left\lceil \log_2 \left(2 + \frac{1}{2\epsilon} \right) \right\rceil$$

Where L is $\lceil \log_2(N) \rceil$.

Proof. See the proof in Nielsen and Chuang[6]. □

The continued fraction expansion

Having obtained an approximation $\tilde{\phi} = \frac{\tilde{s}}{r}$, how should one extract information from this clue? We do this with *continued fractions*. Consider the lemma,

Lemma 4.2.2. *Let $\tilde{\phi}$ be the approximation of the rational number $\frac{s}{r}$ to t bits given as the output of our phase estimation algorithm. Then*

$$\left| \frac{s}{r} - \tilde{\phi} \right| \leq \frac{1}{2r^2}.$$

Proof. By setting $t = 2L + 1 + \lceil \log_2 \left(2 + \frac{1}{2\epsilon} \right) \rceil$ as in theorem 4.2.1., we see that

$$\left| \frac{s}{r} - \tilde{\phi} \right| \leq 2^{-2L-1} \leq \frac{1}{2r^2}. \quad (4.4)$$

This is because $r \leq N \leq 2^L$. □

This lemma is used in a fairly well known result,

Theorem 4.2.3. *Suppose $\frac{s}{r}$ is a rational number with the property*

$$\left| \frac{s}{r} - \tilde{\phi} \right| \leq \frac{1}{2r^2}.$$

It follows that $\frac{s}{r}$ is a convergent of the continued fraction for $\tilde{\phi}$, and may be computed in $\mathcal{O}(L^3)$ steps using the continued fractions algorithm.

Proof. Using the above lemma, we know $\tilde{\phi}$ has this property. The rest of the proof is outside the scope of this paper, and we refer to books such as An Introduction to the Theory of Numbers by Hardy, Wright, Wiles. □

The continued fractions algorithm

The idea of this algorithm is to describe rational numbers in terms of continued fractions.

$$[a_0, \dots, a_M] \equiv a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\dots + \frac{1}{a_M}}}}$$

for positive integers a_0, \dots, a_M . Rational numbers are represented by finite continued numbers. Let a be some real number and consider the following recursive process.

1. Split a into its integer and fractional part by

$$a = a_0 + r_1.$$

2. Invert the fractional part so that

$$a = a_0 + \frac{1}{\frac{1}{r_1}}.$$

3. Apply step one and two for $\frac{1}{r_1}$,

$$a = a_0 + \frac{1}{a_1 + \frac{1}{r_2}}.$$

4. Repeat until the decomposition into a continued fraction may be represented without a fraction. That is to say, using only integers.

Example 4.2.1. Let's apply the continued fractions algorithm on $\frac{15}{11}$.

$$\begin{aligned} \frac{15}{11} &= 1 + \frac{4}{11} \\ &= 1 + \frac{1}{\frac{11}{4}} \\ &= 1 + \frac{1}{2 + \frac{3}{4}} \\ &= 1 + \frac{1}{2 + \frac{1}{\frac{4}{3}}} \\ &= 1 + \frac{1}{2 + \frac{1}{1 + \frac{1}{3}}} \\ &= [1, 2, 1, 3]. \end{aligned}$$

The continued fraction algorithm can be done classically without the need for a quantum circuit. We only include this result to show later that it will not affect the time complexity of the algorithm. As to not lose focus of our goal of giving a comprehensive view of Shor's algorithm in the context of quantum computation, we will not prove this classic result.

Theorem 4.2.4. For rational numbers $\frac{p}{q}$, the continued fraction algorithm will complete in (at least) $\mathcal{O}(L^3)$

Proof. We refer to Loceff[12]. □

The order finding algorithm

We now have all the tools needed to show a final version of the order finding subroutine.

1. **Initialization:**

$$\rightarrow |\psi_0\rangle = |0\rangle^{\otimes t} |1\rangle^{\otimes L}$$

2. **Apply Hadamard on first register**

$$\rightarrow |\psi_1\rangle = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle |1\rangle$$

3. **Applying the oracle:**

$$\rightarrow |\psi_2\rangle = \frac{1}{\sqrt{r}2^t} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i j \frac{s}{r}} |j\rangle |u_s\rangle$$

4. **Inverse quantum Fourier transform:**

$$\rightarrow |\psi_3\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\tilde{\phi}\rangle |u_s\rangle$$

5. **Measure first register:**

$$\rightarrow \tilde{\phi}$$

6. **Continued fraction algorithm:**

$$\rightarrow r'$$

This algorithm will yield the correct answer with a certain probability. To increase the probability of attaining the correct answer, one repeats the algorithm a number of times to verify the correct answer. Hopefully, an example will help with illustrating the process we just outlined.

Example 4.2.2. This is how using Shor's algorithm to factor $N = 91$ might look.

We begin by arbitrarily choosing some integer, say $a = 3$. After confirming that $\gcd(91, 3) = 1$, we pick appropriate sizes for the two registers. Because $N < 2^7$, we would need $L = 7$ and $t = 2L + 1 = 15$ qubits in the second register. We will use these 15 qubits to approximate the phase ϕ . The input to the order finding

algorithm for order r will thus be $|0\rangle^{\otimes t} |1\rangle^{\otimes L}$, which after having applied the Hadamard gates to the first register becomes

$$(|0\rangle + |1\rangle + \dots + |2^{15}\rangle) \otimes |1\rangle.$$

After this it's time to apply our oracle, the controlled- U^{2^j} operations.

$$\frac{1}{\sqrt{2^{15}}} (|0\rangle |1\rangle + |1\rangle |3\rangle + |2\rangle |9\rangle + |3\rangle |27\rangle + |4\rangle |81\rangle + |5\rangle |61\rangle + \dots)$$

Notice that this can be rewritten as

$$\frac{1}{\sqrt{2^{15}}} ((|0\rangle + |6\rangle + |12\rangle + \dots) \otimes |1\rangle + (|1\rangle + |7\rangle + |13\rangle + \dots) \otimes |3\rangle + \dots)$$

If only we could look inside the circuit and see the state, we would be finished here. But alas, by observing the state we also collapse it. This wouldn't give us any worthwhile information. This is when we apply the inverse quantum Fourier transform to the first state and measure the second register, amplifying the amplitudes for $\frac{2^t}{6} \cdot s$ for $s \in \mathbb{Z}^+$. Thus we would have a high probability of obtaining the states which are approximate multiples of $\frac{2^t}{r} = \frac{2^{15}}{6}$. A rough illustration of this is given by figure 4.3.

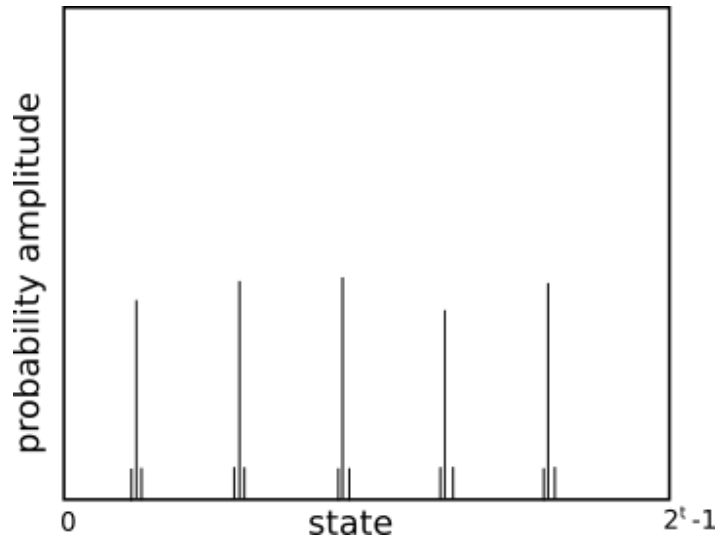


Figure 4.3: A rough illustration of what the state in example 4.2.2 may look like after applying the inverse quantum Fourier transform

After measuring, one obtains one such approximation. Say we obtain the state $|27307\rangle$, which is the integer closest to $2^{15} \frac{5}{6}$. This is the "clue" we use to

extract information about r through the continued fractions algorithm.

$$\frac{27307}{2^{15}} = \frac{27307}{32768} = \frac{1}{1 + \frac{1}{5 + \frac{1}{2730 + \frac{1}{2}}}}$$

After the first two convergents $\frac{1}{1}, \frac{5}{6}$ we see that the next one is very small since 2730 is large. This leads us to believe that the denominator yields the answer, $r = 6$. Let's check this.

$$3^6 \equiv 1 \pmod{91}.$$

We have found the order of $a = 3$. We see that r is even, and neither $3^3 + 1 = 28$ nor $3^3 - 1 = 26$ is a multiple of 91. Thus either 28 or 26 have a non-trivial factor with 91. This can easily be calculated with Euclid's algorithm.

$$\gcd(91, 28) = \gcd(28, 91 - 28 \cdot 3) = \gcd(7, 28 - 7 \cdot 4) = 7.$$

We see that $\frac{91}{7} = 13$. Since 13 is a prime, we have a complete factorization $91 = 7 \cdot 13$.

Using HSP

Another way to think about the order finding problem is as a finite Abelian HSP. Consider the following formulation. To make things easier, assume that we are dealing with the easy case. That is, the order r divides N . Consider, Group $G = \mathbb{Z}_N$ and a periodic function $f(x + r) = f(x)$ where r is the period to some element $a \in G$.

This function implicitly defines the subgroup $H = \{kr | k \in [0, \dots, N/r]\}$. We would like to find r , the generator of the subgroup H . Put $\chi_g(h) = e^{2\pi i \frac{gh}{N}}$ and

$$H^\perp = \{x | e^{2\pi i \frac{xh}{N}} = 1 \forall h \in H\} = \{x | xkr = 0 \pmod{N} \forall k \in [0, \dots, N/r]\}.$$

The procedure outlined in chapter 3 yields an $x \in H^\perp$. This element x satisfies $xkr = 0 \pmod{N} \forall k \in [0, \dots, N/r]$. This implies that $xr = 0 \pmod{N}$, so x is a multiple of N/r . We use this clue to compute r with the help of continued fractions.

Worth noting is that the quantum Fourier transform in step 4 in this case is the inverse quantum Fourier transform.

4.2.1 Pitfalls of order finding

There are some issues we haven't addressed regarding this approach of finding orders. First of all, the output of the phase estimation at step 5 outputs some m -bit estimation $\tilde{\phi} = \frac{s'}{r'}$ of ϕ . If $\gcd(s', r') \neq 1$, the denominator will not be an approximation of r but rather a divisor of r .

Numerator and denominator have common factors

How do we go about resolving the situation? The by far simplest way is simply realizing the following: It should be fairly unlikely that s' and r' share a factor larger than 1. The prime number theorem tells us that the probability of s being a prime smaller than r is given by $\frac{1}{\log r}$. Thus by repeating the algorithm $\log N$ times we will with a high probability observe a phase $\frac{s'}{r'}$ such that $\gcd(s', r') = 1$.

Oracle implementation

We have naively assumed that the oracle gate in our algorithm is some sort of black box, the implementation details of which we haven't bothered to verify. Thankfully, this abstraction doesn't turn out to be a problem since it's fairly straightforward to implement controlled $-U^{2^j}$ operations.

$$\begin{aligned} |a\rangle |b\rangle &\rightarrow |a\rangle U^{a_t 2^{t-1}} \dots U^{a_1 2^0} |b\rangle \\ &= |a\rangle \left| x^{a_t 2^{t-1}} \dots x^{a_1 2^0} b \pmod N \right\rangle \\ &= |a\rangle |x^a b \pmod N\rangle \end{aligned}$$

Here a is the content of the first register. Knowing the reversible techniques used in chapter 2, this is not hard for us. Simply reversibly compute $x^a \pmod N$ in a third register. In this same register, reversibly multiply the contents of the second register with the contents of the third register. Then just uncompute the third register using the techniques outlined in chapter 2. As for the modular exponentiation part, we first observe the following:

$$x^z \pmod N = \left(x^{a_t 2^{t-1}} \pmod N \right) \left(x^{a_{t-1} 2^{t-2}} \pmod N \right) \dots \left(x^{a_1 2^0} \pmod N \right).$$

We can use modular multiplication to compute $x^2 \pmod N$ and by repetition $x^{2^j} \pmod N$. Since we use $t = 2L + 1 + \lceil \log(2 + \frac{1}{2\epsilon}) \rceil = \mathcal{O}(L)$, we need $t - 1 = \mathcal{O}(L)$ squaring operations which in turn are $\mathcal{O}(L^2)$. This of course means we need a total of $\mathcal{O}(L^3)$ operations. We conclude that oracles may be implemented in polynomial time. Instead of replacing these black box abstractions in our circuits, we will keep them for simplicity sake. We are happy just knowing that they may be implemented reasonably.

4.2.2 Time complexity

The reason Shor's algorithm has garnered so much attention is because of its polynomial time complexity. Having illustrated the functionality of the algorithm previously, we will show that it is also tremendously fast.

- Hadamard gates are $\mathcal{O}(L)$.
- The oracle is implemented with modular exponentiation, which as shown before can be done in $\mathcal{O}(L^3)$ time.

- The inverse quantum Fourier transform of course has the same time complexity as the quantum Fourier transform, which is $\mathcal{O}(L^2)$.
- Recall that classically, theorem 4.2.4. states that the continued fraction algorithm completes in $\mathcal{O}(L^3)$.

Theorem 4.2.5. *Shor's algorithm computes a factor a of any integer N in $\mathcal{O}(L^4)$.*

Proof. The bottleneck of Shor's algorithm is modular exponentiation, which is $\mathcal{O}(L^3)$. Coupled with the reasoning above, and the fact that we repeat the algorithm approximately $\mathcal{O}(L)$ times implies that time complexity is $\mathcal{O}(L^4)$. \square

Closing words

In chapter 4 we have chosen to focus on only Simon's algorithm and Shor's algorithm. However, the techniques used in these algorithms can be found in many other quantum algorithms. The interested reader is may be interested in studying Deutsch's algorithm and Shor's discrete logarithm algorithm. These are both special cases of the finite Abelian HSP.

The techniques we have developed in earlier chapters will also suffice to study algorithms such as Grover's search algorithm. Due to time constraints the author has chosen not to include the aforementioned three algorithms, but encourages the reader to do their own research regarding these.

Bibliography

- [1] Gidney, Ekerå. *How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits.* *arXiv: 1905.09749.*
- [2] Kleinjung. *Factorization of a 768-bit RSA modulus.*
- [3] Pednault. *On "Quantum Supremacy"* <https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/> verified 2020-08-08.
- [4] Bernstein, Heninger, Lou, Valenta. *Post-quantum RSA.*
- [5] Rieffel, Polak. *Quantum Computing - A Gentle Introduction.*
- [6] Nielsen, Chuang. *Quantum Computation and Quantum Information.*
- [7] Shor. *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer.*
- [8] Judson. *Abstract Algebra - Theory and Applications.*
- [9] Cai, Qiu. *Optimal Separation in Exact Query Complexities for Simon's Problem.* *arXiv: 1610.01920*
- [10] Landsberg. *A Very Brief Introduction to Quantum Computing and Quantum Information Theory for Mathematicians.* *arXiv:1801.05893v1.*
- [11] Benenti, Casati, Rossini, Strini. *Principles of Quantum Computation and Information. A Comprehensive Textbook.*
- [12] Loceff. *A course in Quantum Computing Volume 1.*
- [13] Rué, Xambó. *Mathematical Essentials of Quantum Computing.*