KU LEUVEN

FACULTY OF SCIENCE

Analysis and simulations of Shor's algorithm

Robbe Motmans

Supervisors: Prof. B. Preneel
Prof. F. Vercauteren

Mentors: C. Bootland
I. Iliashenko

Thesis presented in fulfillment of the requirements for the degree of Master of Science in Mathematics

Academic year 2017-2018

Preface

Foremost, I would like to sincerely thank my two mentors Carl Bootland and Ilia Iliashenko. Their knowledge and advice were of great value for me and I am very grateful for their time and patience. Next I want to thank my two supervisors Prof. Bart Preneel and Prof. Frederik Vercauteren for giving me the chance to make this thesis and for providing feedback after each presentation. I also want to thank Wouter Castryck who took care of all practical arrangements that come with a thesis. The next persons I want to thank are my two readers Dr. Jan Tuitman and Dr. Enrique Argones Rúa for taking time out of their agenda's to attend my presentations.

At last I want to thank my parents and the rest of my family who always supported me in my studies.

Abstract

Peter Shor developed two quantum algorithms that break two important and currently used cryptosystems, namely the RSA cryptosystem and the ElGamal cryptosystem. It breaks these cryptosystems by solving the two underlying mathematical problems efficiently. The first quantum algorithm solves the integer factorization problem while the second quantum algorithm solves the discrete logarithm problem, both do this in polynomial time.

To be able to understand the theoretical analysis of these algorithms one should be familiar with quantum computing. The first part of the thesis will explain the basics of quantum computing and quantum algorithms for this reason. The remaining part of this thesis we will focus on the factoring algorithm. To get a deeper understanding, we provided some simulations alongside the theoretical analysis to get a deeper understanding of the algorithm. At the end of the thesis the discrete logarithm algorithm is briefly explained.

List of abbreviations and list of symbols

$\phi(n)$	Euler Totient function of n
$\mathbb{Z}_n$	set of integers modulo n
$\mathbb{Z}_n^*$	multiplicative group of integers modulo n
$\langle g \rangle$	group generated by the integer g
$\lvert\psi\rangle$	ket-notation of a quantum state
$\langle\psi\rvert$	dual vector of $\lvert\psi\rangle$
α_i	complex amplitude of the classical state $\lvert i\rangle$
$\mathbb{C}^N$	N dimensional complex Hilbert space
$\otimes$	tensor product operator
$U^\dagger$	conjugate transpose or hermitian transpose of a transformation U
H	Hadamard gate
R_ψ	phase shift gate
CNOT	CNOT gate
$\oplus$	addition modulo 2
$H^{\otimes n}$	Hadamard gate tensored n times with itself
U_f	oracle for an arbitrary function f
F_N	discrete Fourier transform and quantum Fourier transform
$\lfloor x \rfloor$	integer part of a real number x

Contents

Chapter 1

Introduction

Quantum Computation is the field that investigates the computational power and possibilities of computers based on quantum mechanics, while classical computers are based on classical mechanics. The interest in quantum computers is due to the fact that it is possible to find quantum algorithms that are significantly faster than all known classical algorithms solving the same problem.

In this text we will begin by explaining the basic principles of quantum computing. After this we will describe some quantum algorithms. We start with the Deutsch-Jozsa algorithm [7] from 1992. This was one of the first quantum algorithms that illustrated the capabilities of quantum computers. Next we will describe Simon's algorithm [17] from 1997, which was the first quantum algorithm with an exponential improvement over the corresponding classical algorithm. After Simon's algorithm we will discuss Shor's algorithms [16]. Peter Shor described two quantum algorithms for two distinct mathematical problems. The first algorithm is a quantum algorithm for integer factorization while his second algorithm is for solving discrete logarithms. Since in cryptography it is assumed that these problems are computationally hard and are used as the basis of cryptographic systems like the RSA system or the ElGamal system, one could use a quantum computer to break those systems.

Our main focus will be on Shor's factoring algorithm because this algorithm has a larger impact than the discrete logarithm algorithm in the sense that it breaks the RSA system which is more widely used as the ElGamal system and because both algorithms are very similar. They use the same quantum techniques and operators to solve the two different problems. To get a better understanding and a deeper insight in the algorithm we will accompany the theoretical analysis by the results of numerous simulations. To perform these simulations we edited and wrote some programs using the C library ‘Libquantum’, which is a library for the simulation of quantum mechanics and quantum computing. This library has been written by Björn Butscher and Hendrik Weimer [2].

Chapter 2

RSA and ElGamal

2.1 Public-key cryptosystems

Before the paper [8] by Diffie and Hellman only symmetric key cryptosystems were used. One shortcoming of such cryptosystems is that the common key used has to be known to both sides of the communication channel and has to be secret. This means that we have to make sure there is secure key exchange.

Diffie and Hellman introduced the idea of public-key cryptography in their paper. In public-key cryptography each party has its own key pair, this pair consists of a public key and a private key. Anyone who wishes to send a message to somebody else takes this second person's public key, encrypts his message with it and sends it. The second person will be able to decrypt the message using his private key.

Public-key systems work because the public and private key are linked mathematically in such a way that knowing the public key does not give enough information about the corresponding private key to recover it in an efficient way. This requires a mathematical operation which is easy to compute in one way, used for the encryption, but hard to compute the other way such that the decryption without the private key is nearly impossible in the sense that performing this operation would take an exponential running time on a classical computer. Such mathematical functions are called trapdoor one-way functions for this reason. Two important public-key cryptosystems, that are used at present, will be discussed next. These are the RSA algorithm [14] and the ElGamal cryptosystem [9].

2.2 RSA

2.2.1 Key generation

The RSA algorithm is based on the difficulty of factoring large integers. To generate the public and secret key we start by choosing two random large primes p and q and calculate $n = pq$. Next, we take a random integer e which is coprime to $\phi(n) = (p-1)(q-1)$ and with e between 3 and $\phi(n)$. Often the numbers 3, 17 or 65537 are chosen because these numbers result in a faster encryption. The notation $\phi(n)$ is the Euler totient function, whose value is the number of positive integers less than n which are coprime to n and is thus also the number of elements in $\mathbb{Z}_n^*$. In other words, we take an integer e such that

$$\gcd(e, (p-1)(q-1)) = 1,$$

with $\gcd(a, b)$ the greatest common divisor of a and b.

Now we compute d, with d the multiplicative inverse of e in the group $\mathbb{Z}_{\phi(n)}^*$. This means that d must satisfy

$$ed = 1(\text{mod } \phi(n)).$$

Now we have obtained the integers e, d and n the public key used for encryption is (e, n) and the secret key used for decryption is (d, n).

2.2.2 Encryption and decryption

Let a message M be an integer between 0 and $n-1$. We encrypt this message M by taking the eth power modulo n of M.

$$C \equiv M^e (\text{mod } n).$$

The resulting integer is called the ciphertext and will be denoted C. We decrypt C by taking the dth power modulo n of C:

$$M \equiv C^d (\text{mod } n).$$

Because of the construction of e and d and using that $M^{\phi(n)} \equiv 1(\text{mod } n)$ we have that

$$C^d \equiv (M^e)^d (\text{mod } n) \equiv M^{ed} (\text{mod } n) \equiv M (\text{mod } n) = M,$$

which is necessary to recover the original message.

2.3 ElGamal

The second public-key cryptosystem we discuss is the ElGamal cryptosystem which is based on the Discrete Logarithm problem. The Discrete Logarithm problem is the following:

Problem. *Let G be an multiplicative abelian group with generator g. Given g and $h \in G$, find an x, such that $g^x = h$.*

2.3.1 Key generation

We start by randomly choosing a large prime p and an element g of the multiplicative group $\mathbb{Z}_p^*$ of prime order q. These parameters create a finite abelian group $G = \langle g \rangle$ of order q with generator g. Now we select a random integer b between 1 and $q - 1$. With this integer we calculate $h = g^b (\text{mod } p)$.

The public key that is used for encryption is (p, g, h) and the secret key used for decryption is b.

2.3.2 Encryption and decryption

Let a message M be an integer between 0 and $p - 1$. To encrypt our message we start by selecting a random integer k. With this integer we compute

$$C_1 = g^k (\text{mod p}),$$
$$C_2 = Mh^k (\text{mod } p).$$

The resulting ciphertext is then $C = (C_1, C_2)$. The decryption of the obtained ciphertext goes as follows

$$\tfrac{C_2}{(C_1)^b} (\text{mod } p) = \tfrac{Mh^k}{g^{bk}} (\text{mod } p) = \tfrac{Mg^{bk}}{g^{bk}} (\text{mod } p) = M.$$

Chapter 3

Basics of Quantum Computing

In this chapter we will discuss the basic principles of quantum mechanics and quantum computing, which can be found, for example, in the textbooks [20, 13, 18].

3.1 Postulates of quantum mechanics

3.1.1 The superposition principle

Consider a physical system with N distinguishable classical or basic states. By a classical or basic state we mean a state in which the system can be found if we observe it. The *superposition* principle says that a quantum system can also be placed in a linear combination of these classical states with complex coefficients.

Let us denote the classical states of our quantum system by $|0\rangle, |1\rangle, ..., |N-1\rangle$. The quantum state can be written as

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle + ... + \alpha_{N-1}|N-1\rangle,$$

where α_i is a complex number such that $\sum_i |\alpha_i|^2 = 1$. The different α_i's are called the *amplitudes* of the classical state $|i\rangle$. The quantum state of an N-state system can thus be described by N complex numbers. This means we can write the state of a quantum system as an N-dimensional vector

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ . \\ . \\ \alpha_{N-1} \end{pmatrix}.$$

We have already used another notation (called *Dirac's ket* notation) where the state is represented as a linear superposition of the classical states but actually this is just another way of writing a vector. Namely let

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \\ . \\ . \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \\ . \\ . \\ 0 \end{pmatrix}, ..., |N-1\rangle = \begin{pmatrix} 0 \\ 0 \\ . \\ . \\ 1 \end{pmatrix}.$$

Mathematically, the classical states $|0\rangle, |1\rangle, ..., |N-1\rangle$ are represented by mutually orthogonal unit vectors in an N dimensional complex vector space with inner product (called a Hilbert space). In other words, they form an orthonormal basis for this space. A quantum state $|\psi\rangle$ is a unit vector in this space.

Dirac's ket notation is in fact just one part of Dirac's bra-ket notation. The column vector $|v\rangle$ of the Hilbert space $\mathbb{C}^N$ is referred to as a ket-v. The dual vector of $|v\rangle$ has the following notation:

$$\langle v| = |v\rangle^\dagger = \overline{|v\rangle}^T = (\overline{v_0}\ \overline{v_1}\ ...\ \overline{v_{N-1}}).$$

This dual vector is called a bra-v. The inner product of two vectors
$|v\rangle = v_0|0\rangle + ... + v_{N-1}|N-1\rangle$ and $|w\rangle = w_0|0\rangle + ... + w_{N-1}|N-1\rangle$ of $\mathbb{C}^N$ is then defined as

$$\langle v|w\rangle = \begin{pmatrix} \overline{v_0} & \overline{v_1} & ... & \overline{v_{N-1}} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ . \\ . \\ w_{N-1} \end{pmatrix} = \overline{v_0}w_0 + \overline{v_1}w_1 + ... + \overline{v_{N-1}}w_{N-1}.$$

3.1.2 The measurement principle

Performing a *measurement* on a quantum state does not mean we can measure the complex amplitudes α_i. Instead, if we measure a state $|\psi\rangle$ we will get one and only one classical state $|j\rangle$ with probability $|\alpha_j|^2$. Thus observing a quantum state induces a probability distribution on the classical states, given by the squared norms of their amplitudes. This is the reason behind the restriction $\sum_i |\alpha_i|^2 = 1$.

A very important aspect about measuring a quantum state is that if one measures a quantum state and observes the classical state $|j\rangle$, the system will be in state $|j\rangle$ after the measurement. So every measurement after the first one will give you $|j\rangle$ as outcome. This implies that you cannot retrieve any more information about the amplitudes α_i.

3.1.3 A qubit

In classical computation, the most basic building block and the unit of information is a *bit*, a two state system that can either be 0 or 1. In quantum computation this unit is a *quantum bit* or *qubit*, a 2-state quantum system. The state space of such a system is then the complex Hilbert space $\mathbb{C}^2$. The state of a qubit is a unit vector in this space and can thus be written as $\begin{pmatrix} \alpha \\ \beta \end{pmatrix} \in \mathbb{C}^2$. We can also write this in Dirac notation as

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

with $\alpha, \beta \in \mathbb{C}$, $|\alpha|^2 + |\beta|^2 = 1$ and $|0\rangle, |1\rangle$ the orthonormal basis $\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

A physical example of a qubit is a hydrogen atom, which can be seen in Figure 3.1. Note that the electron of a hydrogen atom actually has an infinite number of energy levels, but that as long as we can isolate two of them, we can use these two as a qubit. Then the electron of a hydrogen atom can either be in its ground state or in an exited state. Before we measure, we don't know in which state it is. So if we represent the ground state as 0 and the exited state as 1, we can consider it a qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Measuring in the basis above yields 0 with probability $|\alpha|^2$ and 1 with probability $|\beta|^2$.

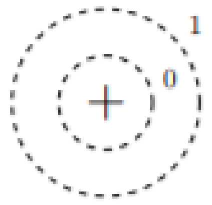

Figure 3.1: Qubit representation of a hydrogen atom

3.1.4 Phase

The amplitudes of a quantum state are complex numbers. In general, a complex number $z = x + iy$ can be represented as a point in the two dimensional plane, as can be seen in Figure 3.2. A complex number can also be written in the form $z = re^{i\theta}$, where r is the distance from the origin to the complex number in the plane and θ is the angle with the x-axis. This θ is called the *phase* of the complex number z.

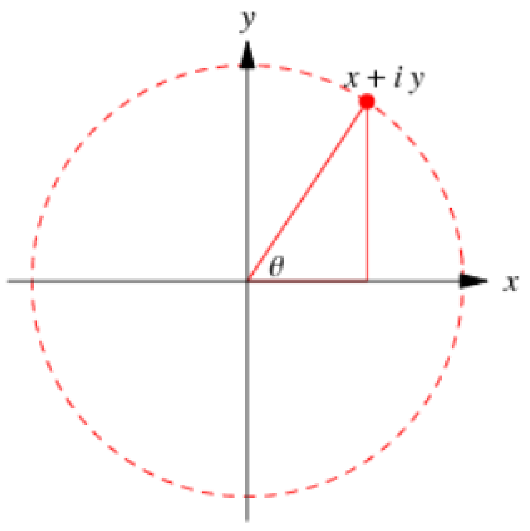

Figure 3.2: Complex plane

In quantum computing the terms phase and phase factor are used frequently. The term phase factor has two different meanings, depending on the context of the text. For example, consider the quantum state $e^{i\theta}|\psi\rangle$ where $|\psi\rangle$ is also a quantum state and θ is a real number. We say that the two states $e^{i\theta}|\psi\rangle$ and $|\psi\rangle$ are equal up to the *global phase factor* $e^{i\theta}$. For two quantum states that differ only in global phase, the probabilities for the outcome of the measurements are exactly the same. When a transformation is applied to two quantum states that differ only in a global phase factor the two outcomes of the transformation differ by the same phase factor. For these two reasons these two states are identical from an observational point of view. Therefore global phase factors are often ignored.

The second meaning is the one of the *relative phase factor*. Consider for example the two states

$$\tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{1}{\sqrt{2}}|1\rangle \text{ and } \tfrac{1}{\sqrt{2}}|0\rangle - \tfrac{1}{\sqrt{2}}|1\rangle.$$

The probability of measuring $|1\rangle$ is equal in both states, namely 1/2. However the amplitude of $|1\rangle$ is $1/\sqrt{2}$ for the first state while this amplitude is $-1/\sqrt{2}$ for the second state. We say that these two states are the same up to a relative phase factor because the $|0\rangle$ amplitudes are identical and the $|1\rangle$ amplitudes differ by a relative phase factor of -1. In general, we say that two amplitudes a and b differ by a relative phase factor if $a = e^{i\theta}b$ with θ a real number. We say that two states which contain the same basic states differ by a relative phase factor if at least one of the amplitudes of those states differ by such a phase factor. Relative phase factors cannot be ignored because they have an effect on the outcome of quantum transformations.

3.1.5 Multi-qubit systems

In classical computation we work with multiple bit systems. The same holds for quantum computation, where we work with *multi-qubit systems*. The state space of such a multi-qubit system is the tensor product of the state spaces of the component qubit systems. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state $|\psi_i\rangle$, then the joint state of the system is $|\psi_1\rangle \otimes |\psi_2\rangle \otimes ... \otimes |\psi_n\rangle$. For instance, a 2-qubit system has 4 basic states: $|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle$. Here for instance, $|0\rangle \otimes |1\rangle$ means that the first qubit is in state $|0\rangle$ and the second qubit is in state $|1\rangle$. The notation $|0\rangle \otimes |1\rangle$ is often abbreviated to $|01\rangle$. By the superposition principle, the quantum state of the 2-qubit system can be any linear combination of these four classical states:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \ ,$$

where $\alpha_{ij} \in \mathbb{C}, \sum_{ij} |\alpha_{ij}|^2 = 1$. This is again the Dirac notation, we can also write this as the unit vector in $\mathbb{C}^4$:

$$|\psi\rangle = \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix}.$$

In general, a n-qubit system has 2^n basic states, each in the form $|b_1 b_2 ... b_n\rangle$ with $b_i \in \{0, 1\}$. The mathematics of tensor products can be found in Appendix A.

Measuring a 2-qubit system $|\psi\rangle$ now reveals two bits of information. The probability of obtaining the state $|ij\rangle$ is $|\alpha_{ij}|^2$ with $i, j \in \{0, 1\}$. After the measurement, the state of the two qubits is $|ij\rangle$ or in other words the state of the first qubit is $|i\rangle$ and the state of the second is $|j\rangle$.

Instead of measuring the whole system, we can also measure parts of it. For example, when measuring the first qubit, the probability of obtaining 0 is $|\alpha_{00}|^2 + |\alpha_{01}|^2$ and the probability of obtaining 1 is $|\alpha_{10}|^2 + |\alpha_{11}|^2$. If the outcome of the measurement was 0, the system will then be in the new state:

$$|\psi\rangle_{new} = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}.$$

The new superposition is obtained by removing all the terms that are inconsistent with the outcome of the measurement. This new sum must then be renormalized to obtain a unit vector.

3.1.6 Evolution of quantum systems

A physical system changes in time, so the state vector of a quantum system also evolves. Quantum theory postulates that the evolution of the state vector of a closed quantum system is necessarily linear. In other words, the transformations can be represented by matrices.

Second, the state of a quantum system is represented by a unit vector of a Hilbert space. This means that every transformation has to preserve the norm of 1, where the norm of a qubit $|v\rangle$ is $\sqrt{\langle v|v\rangle}$. The evolution of a quantum system thus has to be *unitary.*

A linear transformation U is unitary iff $UU^\dagger = U^\dagger U = I$, where $U^\dagger$ is called the *hermitian transpose* (or *conjugate transpose*) of U. For a 2×2 matrix we have

$$U^\dagger = \begin{bmatrix} \overline{a} & \overline{c} \\ \overline{b} & \overline{d} \end{bmatrix} \text{ if } U = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$

In quantum computing, we refer to a unitary operator U acting on a single-qubit as a 1-qubit gate. This operator U is an operator on the 2-dimensional Hilbert space and can be represented by a 2×2 matrix. An operator can act on more than just one qubit; in general, we refer to an operator acting on n qubits as an n-qubit gate. This operator can be represented by a $2^n \times 2^n$ unitary matrix.

Some important single-qubit and multi-qubit gates are the following; their importance will become clear later in the text:

- Hadamard Gate: The Hadamard gate is a single-qubit gate and most (if not all) quantum algorithms make use of this gate. The Hadamard transform applied to one of the basic states $|0\rangle$ or $|1\rangle$ results in a superposition with equal probability of obtaining $|0\rangle$ and $|1\rangle$ when measuring the outcome, namely

$$\begin{aligned} H|0\rangle &= \tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{1}{\sqrt{2}}|1\rangle, \\ H|1\rangle &= \tfrac{1}{\sqrt{2}}|0\rangle - \tfrac{1}{\sqrt{2}}|1\rangle, \end{aligned}$$

  so that

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

- Phase shift gate: The phase shift gate is a single-qubit gate that leaves $|0\rangle$ unchanged and transforms $|1\rangle$ into $e^{i\psi}|1\rangle$, thus

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

- Controlled NOT (CNOT) gate: The CNOT gate is a 2-qubit gate. The first bit of a CNOT gate is called the "control bit", the second bit is called the "target bit". The target bit flips if and only if the control bit is equal to 1. The control bit always stays the same. The CNOT gate applied on a basic state $|a, b\rangle$ can be displayed as $\text{CNOT}|a, b\rangle = |a, a \oplus b\rangle$. The matrix notation for the CNOT gate is

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

When using the CNOT gate, this gate is represented by the following symbol:

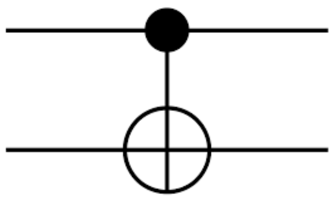

Figure 3.3: Representation CNOT gate

- Toffoli gate: The Toffoli gate is a doubly controlled NOT, it negates the third bit if and only if the first two bits are 1.

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

When we have an n-qubit system, it is also possible to perform an operation on a part of the qubits or even to perform multiple operations on different qubits at the same time. The resulting transformation at a given time step is denoted with the tensor product. For example, the CNOT-gate acting on the first two qubits corresponds to the unitary transformation CNOT$\otimes I_{n-2}$. Another example is when the Hadamard transformation is applied to each qubit, the resulting transformation is H tensored with itself n times. This is denoted $H_{2^n} = H^{\otimes n}$. We will exploit this transformation a lot. The mathematics of the tensor product of operators can also be found in Appendix A.

3.2 Entanglement

In a quantum system of two or more qubits, multiple qubits can be entangled with each other. *Entanglement* is an important property which is unique for quantum systems. To explain entanglement we will start with two qubits $|\psi_1\rangle$ and $|\psi_2\rangle$, both in the zero state $|0\rangle$, and observe the creation of an EPR pair (named after Einstein, Podolsky and Rosen). We begin with the first qubit $|\psi_1\rangle$ and apply the Hadamard gate to it, we get

$$|\psi_1'\rangle = H|\psi_1\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle).$$

The joint state-space of the two qubits is then

$$|\psi_1'\psi_2\rangle = \tfrac{1}{\sqrt{2}}(|00\rangle + |10\rangle).$$

Next apply the CNOT gate to our two qubits

$$|\phi\rangle = \text{CNOT}|\psi_1'\psi_2\rangle = \tfrac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

This pair of qubits is now called entangled because it is impossible to decompose this state into the component states. Mathematically, this means that the two-qubit state $|\phi\rangle$ cannot be written as a tensor product $|\psi_1''\rangle \otimes |\psi_2''\rangle$ with $|\psi_1''\rangle$ and $|\psi_2''\rangle$ the state of the first qubit and the second qubit, respectively.

The remarkable thing about entanglement is that by measuring one qubit we can know the state of the second qubit without measuring it. Take for example the state above, the probability of observing 0 or 1 is both 1/2 and this for both the first and the second qubit. If we would measure for example the first qubit and obtain 0 as result, the state of the second qubit collapses into 0 without measuring the second state. So for the second qubit, the probability of 1/2 of observing 1 becomes 0 and the probability of observing 0 becomes 1 after we measured the first qubit. So in this case we know for sure that the state of the second qubit is 1 without measuring the qubit.

Chapter 4

Early Quantum Algorithms

In this chapter we will first discuss the concepts needed for quantum algorithms. After this we will explain two early quantum algorithms.

In quantum algorithms we often use the term register. A *register* is just the place where a certain sequence of qubits is held. To explain how a quantum computer can apply computational steps to its register of qubits we need a mathematical model for a quantum computer. Two equivalent models exist for this: the quantum Turing machine model [5] and the quantum circuit model [6]. These models are equivalent in the sense that they can simulate each other in polynomial time [21]. We will use the quantum circuit model, this model is mostly used by researchers because of it's simplicity.

A *quantum circuit* is a sequence of quantum gates. A quantum gate acts on a subset of the qubits, and leaves the rest unchanged. When different transformations are applied on different sets of qubits, this can be represented by the tensor product as seen in Section 3.1.6. In general, to perform a computation using a quantum circuit, we prepare an input state, apply the quantum circuit to this input state and perform a measurement on the result.

In classical computing, the most commonly known gates for classical circuits are the AND-gate, the OR-gate and the NOT-gate. These gates form a universal set, which means that for any boolean function $f : \{0,1\}^n \to \{0,1\}$ there is a circuit build out of these gates. There are, however, more compact universal sets for classical circuits, for example the NAND-gate on its own is universal. For quantum circuits there also exist such sets of elementary gates that are universal. This means that every unitary transformation can be built from these gates. One such set consists of three gates, namely the CNOT gate, the Hadamard gate and the phase shift gate $R_{\pi/4}$.

In most problems we want to solve there are some functions $f : \{0,1\}^n \to \{0,1\}^n$ we need to evaluate. Because all quantum gates are unitary it follows that all operations are reversible. This means that we cannot construct a quantum circuit that given x as input outputs $f(x)$, unless f is a bijection. Thus we need to convert a classical circuit for computing f into a reversible circuit. This can be done with Toffoli gates, the Toffoli gate on its own is a universal set for reversible computation. For example, the classical AND gate with input x and y gives $x \wedge y$, this can be replaced with a Toffoli gate where the three input bits are $|x\rangle$, $|y\rangle$ and $|0\rangle$, the output will then be $|x\rangle$, $|y\rangle$ and $|x \wedge y\rangle$.

In quantum algorithms, we often encounter an arbitrary function f that we need to calculate. As seen above, every function that can be implemented as a classical circuit can also be implemented as a quantum circuit. We do not need to build specific quantum circuits for evaluating those functions. For this reason the quantum circuit for computing a function f is given as an quantum oracle U_f. This is a "black box" that we are allowed to use in the algorithms without access to the circuit.

4.1 Deutsch-Jozsa algorithm

The first algorithm we discuss is the Deutsch-Jozsa algorithm [7], because this is one of the first algorithms that illustrated the capabilities of quantum computers.

Problem. *We have a function $f : \{0,1\}^n \to \{0,1\}$ which is either constant ($f(x) = 0$ or $f(x) = 1$ for all inputs $x \in \{0,1\}^n$), or else balanced, which means $f(x)$ equals 1 for exactly half the possible values of x, and $f(x)$ equals 0 for the other half. The goal is to find out whether $f(x)$ is constant or balanced.*

In the classical case, we will need to query $2^n/2 + 1$ times in the worst case, since we can for example receive $2^n/2$ 0's before getting our first 1. With the quantum algorithm below we will need to query only once.

We have an oracle U_f that on input $|x\rangle|y\rangle$ outputs $|x\rangle|y \oplus f(x)\rangle$, where x contains n qubits and y contains one qubit. In this algorithm and in the next algorithms we will use the $H^{\otimes n}$ transform. One way to define this transform on a random basic state $x \in \{0,1\}^n$ is:

$$H^{\otimes n}|x\rangle = \tfrac{1}{\sqrt{2^n}} \sum_{z=0}^{2^n-1} (-1)^{x \cdot z}|z\rangle,$$

where $x \cdot z = \sum_{k=1}^{n} x_k \cdot z_k$ and $|z\rangle$ is represented as its binary expansion. This will hold in the rest of the paper, a basic state will always be presented as its binary expansion.

We will use two registers, the first one with n qubits and the second one with one qubit. The input of the algorithm is $|0\rangle^{\otimes n}|1\rangle$. The notation $|0\rangle^{\otimes n}$ means $|0\rangle$ tensored n times with itself or in other words n consecutive $|0\rangle$ qubits. In the first step we apply $H^{\otimes(n+1)}$ to the entire state, the resulting state becomes:

$$\begin{aligned}|\psi_1\rangle &= H^{\otimes n}|0\rangle^{\otimes n}H|1\rangle \\ &= \sum_{x=0}^{2^n-1} \frac{1}{\sqrt{2^n}}|x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right).\end{aligned}$$

We now apply our transform U_f that implements $f(x)$ to obtain the new state:

$$|\psi_2\rangle = \sum_{x=0}^{2^n-1} \tfrac{1}{\sqrt{2^n}}|x\rangle \left(\tfrac{|f(x)\rangle - |1\oplus f(x)\rangle}{\sqrt{2}}\right).$$

We can rewrite this state as

$$|\psi_2\rangle = \sum_{x=0}^{2^n-1} \tfrac{(-1)^{f(x)}}{\sqrt{2^n}}|x\rangle \left(\tfrac{|0\rangle - |1\rangle}{\sqrt{2}}\right),$$

because if $f(x) = 0 \Rightarrow |f(x)\rangle - |1\oplus f(x)\rangle = |0\rangle - |1\rangle$ and if $f(x) = 1 \Rightarrow |f(x)\rangle - |1\oplus f(x)\rangle = |1\rangle - |0\rangle$. As last step we apply $H^{\otimes n}$ transform to the first register to obtain:

$$|\psi_3\rangle = \sum_{z=0}^{2^n-1} \sum_{x=0}^{2^n-1} \tfrac{(-1)^{f(x)+x\cdot z}}{2^n}|z\rangle \left(\tfrac{|0\rangle - |1\rangle}{\sqrt{2}}\right).$$

There were two possibilities, either $f(x)$ was constant or $f(x)$ was balanced. We will now look what the possible outcomes of the measurement of the first register are in both cases. We look at the probability of obtaining the state $|0\rangle^{\otimes n}$. The dot product $x \cdot z = 0$ for $z = 0$ so the summation over x becomes

$$\sum_{x=0}^{2^n-1} \tfrac{(-1)^{f(x)}}{2^n}|0\rangle^{\otimes n} \left(\tfrac{|0\rangle - |1\rangle}{\sqrt{2}}\right).$$

When $f(x)$ is constant the amplitude of the basis state $|0\rangle^{\otimes n}$ is equal to

$$\sum_{x=0}^{2^n-1} \frac{(-1)^{f(x)}}{2^n} = \pm 1.$$

This means that when the first n qubits are measured we have a probability of 1 to obtain the state $|0\rangle^{\otimes n}$.

If on the other hand $f(x)$ is balanced, the x summation over $|0\rangle^{\otimes n}$ is negative on half the x and positive on the other half and so cancels out. The probability of measuring $|0\rangle^{\otimes n}$ is equal to 0 in this case.

To summarize, when you measure the first n qubits of the final state $|\psi_3\rangle$ and obtain $|0\rangle^{\otimes n}$ the function $f(x)$ is constant. When any other basic state is obtained after the measurement the function $f(x)$ is balanced.

4.2 Simon's algorithm

The next algorithm we discuss is Simon's algorithm [17]. This algorithm was the first quantum algorithm with an exponential improvement over classical algorithms. Simon proved that the best classical algorithm takes exponential time to solve the problem stated below, namely it would need to make $O(\sqrt{2^n})$ function evaluations. His quantum algorithm takes only $O(n)$ function evaluations (here represented by an oracle) and a polynomial number of other operations.

Problem. *Given a 2-to-1 function $f : \{0,1\}^n \to \{0,1\}^n$ with the property that there is an $a \in \{0,1\}^n$ with $a \neq 0^n$ such that for all x: $f(x) = f(y)$ iff $y = x$ or $y = x \oplus a$, with $\oplus$ addition modulo 2. Find this a.*

In this algorithm we use an oracle U_f that on input $|x\rangle|0\rangle^{\otimes n}$ returns $|x\rangle|f(x)\rangle$, both x and $f(x)$ are n-bit strings. We again use two registers, both with n qubits this time. The input of the algorithm is the state $|0\rangle^{\otimes n}|0\rangle^{\otimes n}$ and we first apply the $H^{\otimes n}$ transform on the first register. We obtain the state

$$|\psi_1\rangle = \sum_{x=0}^{2^n-1} \frac{1}{\sqrt{2^n}}|x\rangle|0\rangle^{\otimes n}.$$

Next, we apply our oracle to obtain

$$|\psi_2\rangle = \sum_{x=0}^{2^n-1} \frac{1}{\sqrt{2^n}}|x\rangle|f(x)\rangle.$$

Now the algorithm measures the second register. Let b be the result of this measurement, then b is a random element in the image of f. The state of the first n qubits is now a superposition over exactly those values of x that are consistent with the outcome of the measurement. Suppose $x = j$ and $x = j \oplus a$ are the two values for which $f(x) = b$, then the resulting state is

$$|\psi_3\rangle = \tfrac{1}{\sqrt{2}}(|j\rangle + |j \oplus a\rangle).$$

On this state we apply the $H^{\otimes n}$ transform to obtain

$$\begin{aligned} |\psi_4\rangle &= \frac{1}{\sqrt{2^{n+1}}}\left(\sum_{i=0}^{2^n-1}(-1)^{j\cdot i}|i\rangle + \sum_{i=0}^{2^n-1}(-1)^{(j\oplus a)\cdot i}|i\rangle\right) \\ &= \frac{1}{\sqrt{2^{n+1}}}\left(\sum_{i=0}^{2^n-1}(-1)^{j\cdot i}(1+(-1)^{a\cdot i})|i\rangle\right). \end{aligned}$$

There are two cases. The first one is when $a \cdot i = 1$, in this case the amplitude becomes zero. This means that we cannot observe this basic state if we measure our n remaining qubits. The second case is when $a \cdot i = 0$, in this case the amplitude will be different from 0. Furthermore, each i such that $a \cdot i = 0$ has an equal probability of occurring. Therefore, each time we run the algorithm and perform a measurement we obtain a random i from $\{0, 1\}^n$ that gives us a bit of information about a. Namely, each time we obtain an equation of the form

$$i_1 a_1 \oplus \ldots \oplus i_n a_n = 0.$$

From linear algebra, we know that we can determine a uniquely once we have $n - 1$ linearly independent equations. So, we run this algorithm over and over until we have our $n - 1$ linearly independent equations. The solution of this system of equations can then be computed efficiently by Gaussian elimination.

Chapter 5

Shor's algorithm

Two important quantum algorithms are Peter Shor's factoring algorithm and discrete logarithm algorithm [16, 15].

It has been known for a very long time that every integer $n \geq 2$ can be uniquely decomposed into a product of primes. This is stated in the fundamental theorem of arithmetic. Mathematicians have always been interested in the question of how to factor a random integer into this product of primes. The best currently known classical factoring algorithm is the number field sieve [12], which in order to factor an integer n takes asymptotic running time $O(\exp(c(\log n)^{1/3}(\log \log n)^{2/3}))$ for some constant c.

The discrete logarithm problem, as we have already briefly seen in Section 2.3, in a more concrete setting is the following: Suppose we have a prime p and a generator g of the multiplicative group $\mathbb{Z}_p^*$. The discrete logarithm of a number x is the integer r with $0 \leq r \leq p - 1$ such that $g^r \equiv x \bmod p$. The best classical algorithm is Gordon's adaptation of the number field sieve [10], which in order to find the discrete logarithm takes asymptotic running time $O(\exp((\log p)^{1/3}(\log \log p)^{2/3}))$.

Both of Shor's quantum algorithms take a polynomial amount of steps, namely the factoring algorithm takes asymptotically $O((\log n)^2(\log \log n)(\log \log \log n))$ steps along with a polynomial amount of time on a classical computer to convert the output of the quantum computer to factors of n. The discrete logarithm algorithm takes asymptotically $O((\log p)^2(\log \log p)(\log \log \log p))$ steps.

Much of modern cryptography is based on the conjecture that no fast (i.e. polynomial time) factoring algorithm or discrete logarithm algorithm exists. The most important cryptosystems based on these conjectures are the RSA and ElGamal cryptosystems. These would thus be broken if Shor's algorithms could be physically realized for sufficiently large integers.

In this chapter we start by discussing Shor's factoring algorithm. Shor uses a reduction from the factoring problem to the order finding problem, and uses a quantum algorithm to solve this problem. This reduction will be explained first. Next we will introduce the quantum Fourier transform, this transformation is an important aspect of both Shor's algorithms. After this we will give the order finding algorithm itself, followed by the discrete logarithm algorithm.

5.1 Reduction to Order-finding

In the multiplicative group $\mathbb{Z}_n^*$, for an element x, there is a least integer r such that $x^r \equiv 1(\text{mod } n)$. This integer r is called the order of the element x in this group. The order-finding problem is then defined as follows:

Problem. *Given a certain $x \in \mathbb{Z}_n^*$. Find the order of x.*

The reduction from factoring to order finding goes as follows. Suppose we want to find a factor of an odd integer n, given a method for order finding. We choose a random $x \in \{2, ..., n-1\}$ and try to find its order r in $\mathbb{Z}_n^*$.

If for the random x we choose it holds that $x \notin \mathbb{Z}_n^*$, or in other words if x and n are not coprime, the order of x in $\mathbb{Z}_n^*$ does not exist. In this case however we can easily find a factor by computing gcd(x,n), which can be done in $O((\log n)^2)$ time using the Euclidean algorithm.

If $x \in \mathbb{Z}_n^*$ with order r, we have:

$$x^r \equiv 1 \ (\text{mod } n) \iff$$
$$(x^{r/2}-1)(x^{r/2}+1) = x^r - 1 \equiv 0 \ (\text{mod } n).$$

This means that the numbers $\gcd(x^{r/2}-1, n)$ and $\gcd(x^{r/2}+1, n)$ are two factors of n.

This procedure can sometimes fail, namely when we choose x such that the order r is odd or if $x^{r/2} \equiv -1 \ (\text{mod } n)$. In the first case $r/2$ is not an integer while in the second case we could get the trivial factors 1 and n.

This procedure, when applied to a random x (mod n), gives us a nontrivial factor of n with probability at least $1 - 1/2^m$, with m the number of distinct odd prime factors of n. This is stated in Theorem 1.

Theorem 1. *Suppose $n = p_1^{\alpha_1} \cdots p_m^{\alpha_m}$ is the prime factorization of an odd composite positive integer. Let x be an integer chosen uniformly at random, subject to the requirement that $1 \leq x \leq n-1$ and x is coprime to n. Let r be the order of x modulo n. Then*

$$Prob(r \text{ is even and } x^{r/2} \neq -1(mod\ n)) \geq 1 - \tfrac{1}{2^m}.$$

This theorem and associating proof can also be found in Nielsen's and Chuang's book [13, Appendix 4].

5.2 Quantum Fourier Transformation

One transformation that is widely used in classical computing is the discrete Fourier transform. The notations we use in this paper are in accordance with [20, 16] and [13, Chapter 5]. The discrete Fourier transform takes as input a vector of complex numbers $x_0, ..., x_{N-1}$ and outputs a vector of complex numbers $y_0, ..., y_{N-1}$ defined by

$$y_k = \tfrac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{-2\pi ijk/N}.$$

The inverse discrete Fourier transform is then

$$x_k = \tfrac{1}{\sqrt{N}} \sum_{j=0}^{N-1} y_j e^{2\pi ijk/N}.$$

The inverse discrete Fourier transform can be represented by a matrix, which we denote F_N. The rows of this matrix will be indexed by $j \in \{0, ..., N-1\}$ and the columns of this matrix by $k \in \{0, ..., N-1\}$. The (j, k)-element of this matrix is then defined by $\frac{1}{\sqrt{N}} e^{2\pi ijk/N}$.

We can now define the *quantum Fourier transformation*. This transformation is of great importance for Shor's algorithm as we will see later on. The quantum Fourier transform is essentially the exact same transformation as the inverse discrete Fourier transform in the sense that it is represented by the same matrix. This means that the action on an arbitrary state can be written as

$$\sum_{j=0}^{N-1} x_j|j\rangle \to \sum_{k=0}^{N-1} y_k|k\rangle,$$

where the amplitudes y_k are the inverse discrete Fourier transform of the amplitudes x_j. Equivalently, the quantum Fourier transform on a basic state $|j\rangle$ is defined as

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

We will now show the unitarity of the quantum Fourier transform by constructing a quantum circuit that computes the quantum Fourier transform. In the following we assume $N = 2^n$. We write the state $|j\rangle$ using the binary representation $j = j_1j_2...j_n$ for $j = j_1 2^{n-1} + j_2 2^{n-2} + ... + j_n 2^0$. We also use the notation $0.j_lj_{l+1}...j_m$ to represent the binary fraction $j_l/2 + j_{l+1}/4 + ... + j_m/2^{m-l+1}$. The quantum Fourier transform can then be given using the following product representation:

$$|j_1, ..., j_n\rangle \to \tfrac{\left(|0\rangle+e^{2\pi i0.j_n}|1\rangle\right)\left(|0\rangle+e^{2\pi i0.j_{n-1}j_n}|1\rangle\right)\cdots\left(|0\rangle+e^{2\pi i0.j_1j_2\cdots j_n}|1\rangle\right)}{2^{n/2}}.$$

The equivalence of the product representation and the previous definition follows from the following calculations:

$$\begin{aligned}
|j\rangle &\to \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi ijk/2^n}|k\rangle \\
&= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi ij\left(\sum_{l=1}^{n} k_l 2^{-l}\right)}|k_1...k_n\rangle \\
&= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi ijk_l 2^{-l}}|k_l\rangle \\
&= \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left(\sum_{k_l=0}^{1} e^{2\pi ijk_l 2^{-l}}|k_l\rangle\right) \\
&= \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left(|0\rangle + e^{2\pi ij2^{-l}}|1\rangle\right) \\
&= \frac{(|0\rangle + e^{2\pi i0.j_n}|1\rangle)(|0\rangle + e^{2\pi i0.j_{n-1}j_n}|1\rangle)\cdots(|0\rangle + e^{2\pi i0.j_1j_2...j_n}|1\rangle)}{2^{n/2}}.
\end{aligned} \tag{5.1}$$

This product presentation makes it easier to describe an efficient quantum circuit for the quantum Fourier transformation. Such a circuit consists of multiple Hadamard gates and multiple phase shift gates $R_{2\pi/2^k}$, which we will also denote R_k. These R_k gates are used as controlled-R_k gates, where the control bit is the second bit in contrast to the previously seen CNOT gate where the control bit is the first bit. This means that the R_k will be applied if and only if the second bit is equal to 1. The circuit is shown in Figure 5.1.

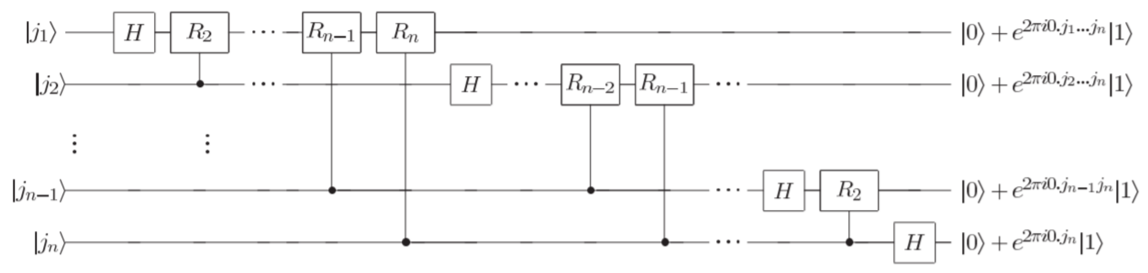
$|j_1\rangle$
$|j_2\rangle$
$|j_{n-1}\rangle$
$|j_n\rangle$
H
R_2
R_{n-1}
R_n
R_{n-2}
$|0\rangle + e^{2\pi i 0.j_1 \dots j_n}|1\rangle$
$|0\rangle + e^{2\pi i 0.j_2 \dots j_n}|1\rangle$
$|0\rangle + e^{2\pi i 0.j_{n-1} j_n}|1\rangle$
$|0\rangle + e^{2\pi i 0.j_n}|1\rangle$

Figure 5.1: Efficient circuit for the quantum Fourier transform

We will now show that this circuit indeed computes the quantum Fourier transform. We start with the state $|j_1, ..., j_n\rangle$. Applying the Hadamard gate to the first bit results in the state

$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_1}|1\rangle\right)|j_2, ..., j_n\rangle.$$

Applying the controlled-R_2 gate results in

$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2}|1\rangle\right)|j_2, ..., j_n\rangle.$$

After this we apply the controlled-R_3 until the controlled-R_n gate in this order, we obtain

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

We now perform a similar operation on the second bit. We start again with applying the Hadamard gate, followed by the controlled-R_2 through R_{n-1} to obtain the following state:

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

We repeat this procedure on each qubit and find

$$\frac{1}{\sqrt{2^n}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2 \dots j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right).$$

The last step we need to do to obtain the output of the quantum Fourier transform is to reorder the qubits. This can be done using multiple quantum swap operation. This quantum swap operation is explained in Appendix B.

On the first qubit we use n quantum gates. On the second qubit we use $n-1$ quantum gates. On each of the following qubits we use one gate less. We see thus that the total number of gates is equal to $n + (n-1) + ... + 1 = \frac{n(n+1)}{2}$. At most $n/2$ swaps are required, therefore this circuit provides a $O(n^2)$ algorithm for the quantum Fourier transform.

For this construction we assumed that $N = 2^n$. Cleve [4] has shown the construction of F_N for all smooth numbers N. Where we consider an integer N smooth if his prime factors are at most $O(\log N)$.

5.3 Order-finding algorithm

In this section we will describe the algorithm for finding the order of $x \pmod n$ for x and n given. We begin by calculating $q = 2^l$ with $n^2 \leq q < 2n^2$. The integers x, n and q will be constant throughout the algorithm. We will use two registers, the first one contains l qubits. For the second register we need enough qubits to represent all integers up to n. Let us denote the number of qubits in the second register as k, then $k = l/2$ qubits if l is even and $k = \lfloor l/2 \rfloor + 1$ qubits if l is odd. The initial state is now $|0\rangle^{\otimes l}|0\rangle^{\otimes k}$, where $H^{\otimes l}$ is applied to the first l qubits. The resulting state is now

$$\frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a\rangle|0\rangle^{\otimes k}.$$

Now we apply an oracle U_f that on input $|a\rangle|0\rangle^{\otimes k}$ returns $|a\rangle|f(a)\rangle$ with $f(a) = x^a \pmod n$. This operation is called modular exponentiation and is the bottleneck of the algorithm which takes asymptotically $O((\log n)^2(\log\log n)(\log\log\log n))$ steps. A detailed analysis giving the quantum gates needed to implement this operation for the algorithm is given by Beckman [1].

Applying the oracle results in the state

$$\frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a\rangle|x^a \pmod n\rangle.$$

We now apply the quantum Fourier transform on the first register, this transforms $|a\rangle$ to

$$\frac{1}{\sqrt{q}}\sum_{c=0}^{q-1} e^{2\pi i\frac{ac}{q}}|c\rangle.$$

The resulting state is now

$$\frac{1}{q}\sum_{a=0}^{q-1}\sum_{c=0}^{q-1} e^{2\pi i\frac{ac}{q}}|c\rangle|x^a(\text{mod } n)\rangle.$$

This is our final state before our measurement. Measuring the second register gives some value $|x^k(\text{mod } n)\rangle$, where we can assume that $0 \leq k < r$. Because the order of x (mod n) is r, for each a of the form $a = jr + k$ it holds that $x^k(\text{mod } n) = x^a(\text{mod } n)$. Let m be the number of elements of $\{0, ..., q-1\}$ that satisfy this condition. The first register then collapses to the state

$$\frac{1}{q}\sum_{j=0}^{m-1}\sum_{c=0}^{q-1} e^{2\pi i\frac{(jr+k)c}{q}}|c\rangle.$$

The amplitude of a certain state $|c\rangle$ is then

$$\frac{1}{q}\sum_{j=0}^{m-1} e^{2\pi i\frac{(jr+k)c}{q}} = \frac{1}{q}e^{2\pi i\frac{kc}{q}}\sum_{j=0}^{m-1} e^{2\pi i\frac{jrc}{q}}.$$

We can ignore the global phase factor $e^{2\pi ikc/q}$ since it has magnitude 1. Using that $\sum_{j=0}^{m-1} z^j = (1-z^m)/(1-z)$ when $z \neq 1$, we obtain

$$\frac{1}{q}\sum_{j=0}^{m-1} e^{2\pi i\frac{jrc}{q}} = \frac{1}{q}\sum_{j=0}^{m-1}\left(e^{2\pi i\frac{rc}{q}}\right)^j = \begin{cases} \frac{m}{q} & \text{if } e^{2\pi i\frac{rc}{q}} = 1, \\ \frac{1-e^{2\pi i\frac{rcm}{q}}}{q(1-e^{2\pi i\frac{rc}{q}})} & \text{if } e^{2\pi i\frac{rc}{q}} \neq 1. \end{cases}$$

There are two distinct cases, either r divides q or r does not divide q.
Suppose r divides q, then $m = q/r$. In this case $e^{2\pi i\frac{rc}{q}} = 1$ iff c is a multiple of q/r. The probability to observe such a c is equal to $m^2/q^2 = 1/r^2$. Because there are exactly r such c and the second register can be r different value for each such c, there are r^2 states for which c is a multiple of q/r. Thus together they have a probability of 1. Thus we are left with a superposition where only the c that are integer multiples of q/r have non-zero amplitudes. This means that when we measure our final state we will obtain a random $c = bq/r$, with $0 \leq b < r$. Thus we find a c such that

$$\tfrac{c}{q} = \tfrac{b}{r}.$$

Both c and q are known values, so when b and r are coprime, we just need to write c/q in lowest terms and find our r. If b and r are not coprime we repeat the procedure above, measure our final state to find again a random c and calculate c/q until we find our r.

For example, we run the order finding algorithm 500 times with $n = 65$ and $x = 44$ and keep track of the observed values each time we measured $|c\rangle$. Figure 5.2 displays the number of times each value has been observed. In this case $q = 8192$ and $r = 4$ which means r divides q. We observe that there are four possible values for c that can be measured; namely 0, 2048, 4096 and 6144. All those values have an equal probability of being measured like we saw before.

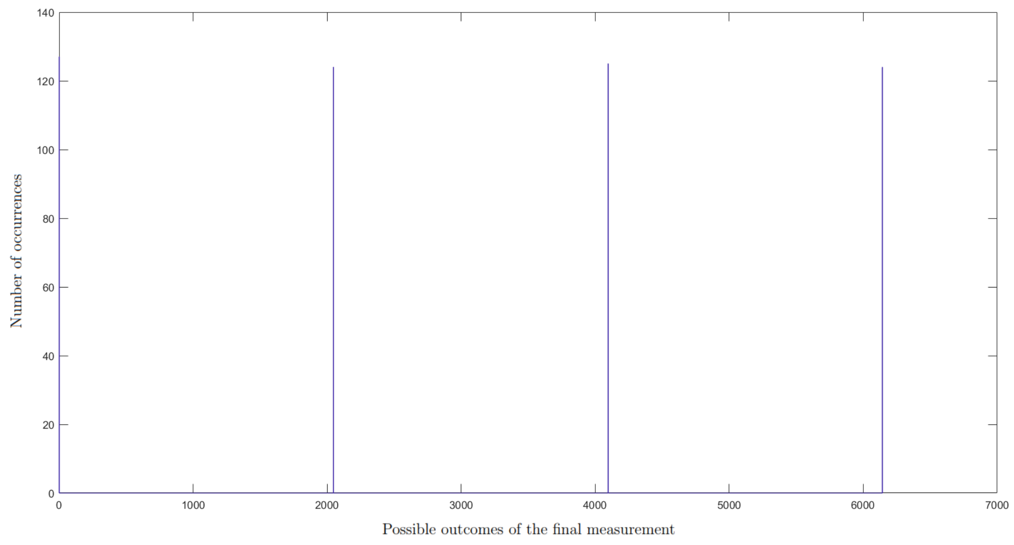

Figure 5.2: Frequency of the outcome of the final measurement when simulated 500 times for n = 65 and x = 44

The second case where r does not divide q is the more general case. In this case, q/r is no longer an integer. This means that $e^{2\pi i \frac{rc}{q}} \neq 1$, using $|1 - e^{ix}| = 2|\sin(x/2)|$ we obtain that the probability of observing a random c is equal to

$$\left| \frac{1-e^{2\pi i \frac{rcm}{q}}}{q(1-e^{2\pi i \frac{rc}{q}})} \right|^2 = \frac{\sin^2(\pi mrc/q)}{q^2 \sin^2(\pi rc/q)}.$$

If there is an integer d such that

$$-\tfrac{r}{2} \leq rc - dq \leq \tfrac{r}{2},$$

or in other words if the residue of rc modulus q is small enough, then this probability will be at least $1/3r^2$ [19].

Thus the probability of observing a certain state c where there is a d such that $-\frac{r}{2} \leq rc - dq \leq \frac{r}{2}$, or in other words c is close to an multiple of q/r, is at least $1/3r^2$. Dividing by rq results then in

$$\left|\frac{c}{q} - \frac{d}{r}\right| \leq \frac{1}{2q}.$$

Using a continued fraction expansion, explained in Appendix C, on c/q will then result in d/r. As for the case where r divided q, if d is not coprime with r, we repeat the procedure until we find our r.

We ran the order finding algorithm 10000 times to find the order of 20 in $\mathbb{Z}_{77}^*$, which is 10. In Figure 5.3 below, the number of times each state has been observed when measuring $|c\rangle$ is plotted. The observed values that resulted in the right factors of 77 are plotted in green, the values that didn't result in the factors are plotted in red. In this figure we can indeed see that with high probability we observe a value that is close to a multiple of q/r. We can also see that if the observed c is close enough to dq/r and d is coprime with r we indeed find the right order and thus the right factors.

In some cases we find a wrong order r' but we do find the right factors. This occurs when either $(x^{r'/2} + 1)$ or $(x^{r'/2} - 1)$ has a common factor with n. This happened in our simulations when we measured c close enough to $5q/r$. In this case the continued fraction expansion resulted in $1/2$ which means we take $r' = 2$ as possible order. This is the wrong order but the factors 7 and 11 were found because $(x^{r'/2} + 1) = (x^1 + 1) = 21$, which has 7 as common factor with 77. Unfortunately this is not common, most of the times when we find a wrong order r' this does not result in the right factors.

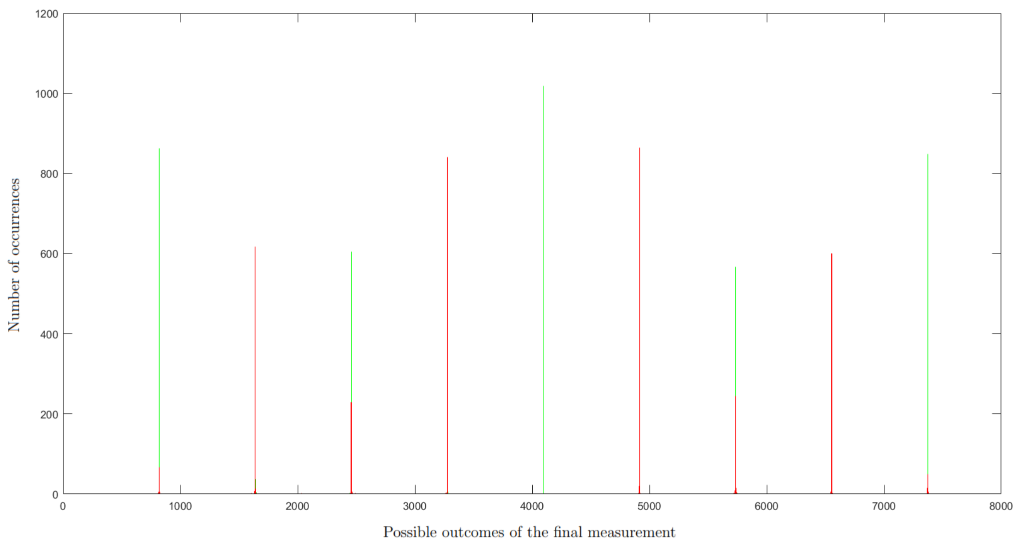

Figure 5.3: Frequency of the outcome of the final measurement when simulated 10000 times for n = 77 and x = 20

In Figure 5.4 we zoomed in on one of the spikes of the figure above, namely the spike between 5700 and 5750. In this figure we have a better view on the distribution of the observed values. As expected there are other values that do not satisfy the given condition that can be measured. We also observe that the closer the values are to the right c value that satisfies the bound, the higher their probability is to be the outcome of the measurement.

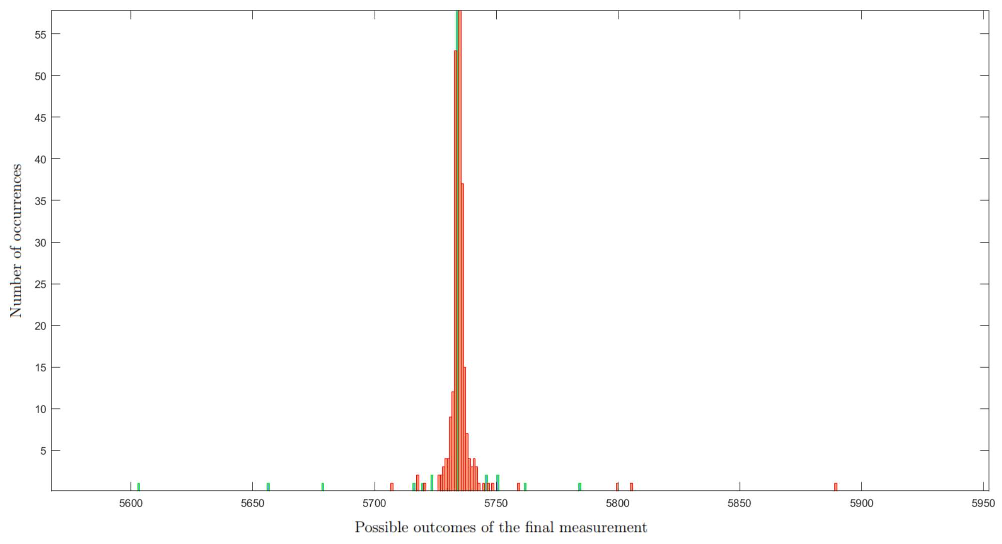

Figure 5.4: Frequency of the outcome of the final measurement when simulated 10000 times for n = 77 and x = 20 around 5750

The number of states $|c\rangle|x^k (\text{mod } n)\rangle$ for which we will find the right r can be approximated as follows. There are $\phi(r)$ possible values of d which are relative prime to r, where $\phi(r)$ is again the totient function. There are also r possible values for x^k because the order of x is r. This means that there are $r\phi(r)$ states $|c\rangle|x^k (\text{mod } n)\rangle$ which result in the right r. Since each of these states occur with probaility at least $1/3r^2$, we will find r with probability at least $\phi(r)/3r$. When we use the theorem that $\phi(r)/r > \sigma/\log\log r$ for some constant σ [11] we find r with probability at least $\sigma/\log\log r$. Thus by repeating the algorithm only $O(\log\log r)$ times we will find r with high probability. Because the order r of a random $x \in \mathbb{Z}_n^*$ is always smaller then n we will find the right factors of n after repeating the algorithm $O(\log\log n)$ times with high probability. To illustrate this we provided some plots of the number of times the algorithm had to run to find the right factors. The x-axis represents the number of times the algorithm had to be repeated before we found the right factors. The y-axis represents the frequency of the number of repetitions. In Figure 5.5 we used 5000 simulations and in Figure 5.6 and Figure 5.7 we used 1000 simulations.

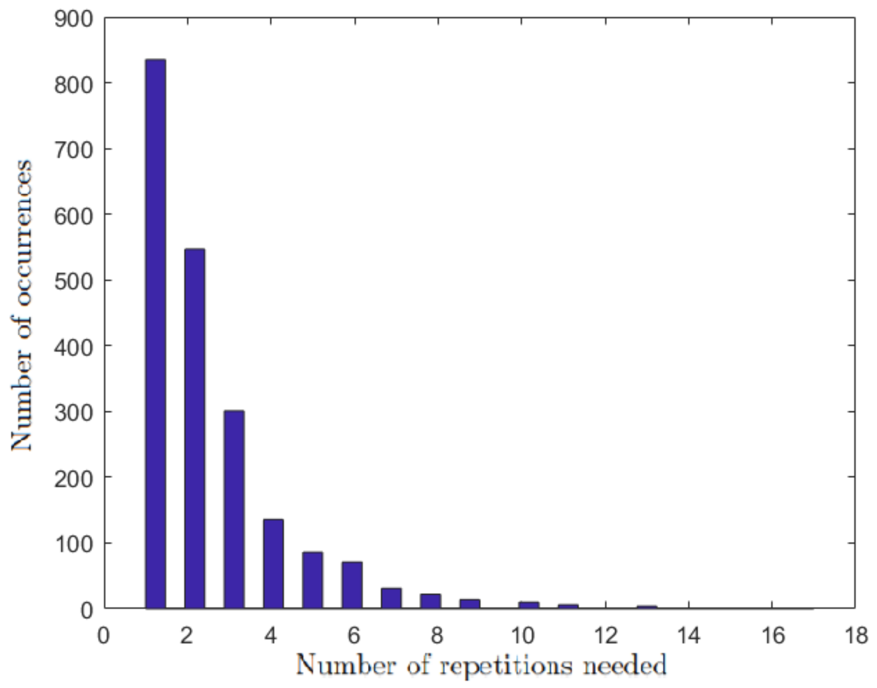

Figure 5.5: Frequency of repetitions needed before the right factors were found when simulated 5000 times for n=77

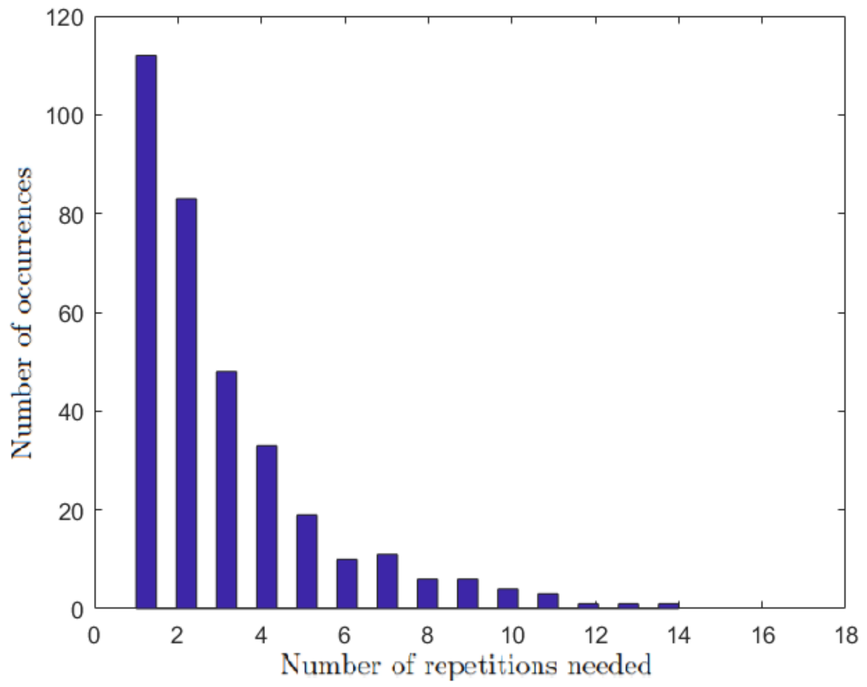

Figure 5.6: Frequency of repetitions needed before the right factors were found when simulated 1000 times for n=221

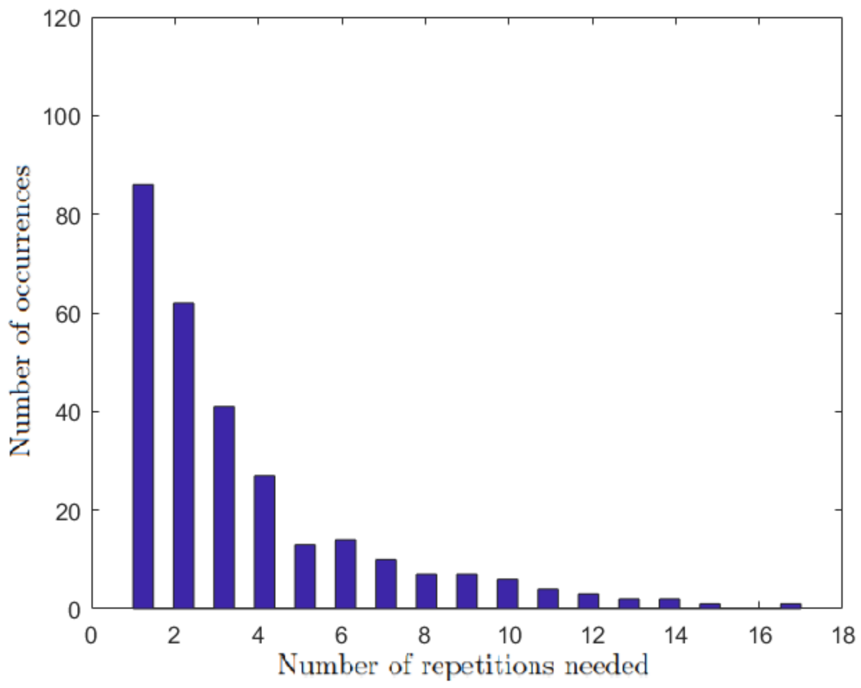

Figure 5.7: Frequency of repetitions needed before the right factors were found when simulated 1000 times for n=527

As expected we see that as the integer we want to factor increases, the expected number of runs before we find the factors also increase. The number of times we find the right factors after just one run decreases and the slope of the figures above will flatten out the higher the integer n becomes.

5.4 Minor improvements to the factoring algorithm

There are some minor improvements that can be done after the quantum part to find the right factors more often. This would reduce the amount of quantum computations which is always preferable assuming quantum computation is more expensive than classical computation.

First, when the observed state c does not result in the right factors, we could try the numbers close to c such as $c \pm 1, c \pm 2, ...$ In the case that c was not close enough to a multiple dq/r of q/r by only a small part where d and r are coprime these have a chance to be close enough to dq/r.

Second and most important, if c/q is rounded off to d'/r' by continued fractions and the obtained r' does not result in the right factors because d and r were not coprime we could consider its small multiples $2r', 3r', ...$ to see if one of these is the actual order of x. This second technique will reduce the expected number of repetitions from $O(\log\log n)$ to $O(1)$ if the first $(\log n)^{1+\epsilon}$ multiples of r' are considered, with ϵ a constant.

5.5 Shor's algorithm as prime verification

Because Shor's algorithm finds the factors of an integer n in polynomial time Shor's algorithm by itself can also be used as a algorithm for prime verification. Because for a composite integer n there holds that you find his factors after $O(\log\log n)$ operations with high probability we can conclude that for a random integer m holds that if we have not found any factors after $O(\log\log m)$ operations this integer m is prime with high probability. There also exists algorithm for primality testing where the outcome if an integer is prime or not is certain. The fastest know algorithm is Chau's and Lo's quantum Pocklington-Lehmer $n-1$ primality test using Shor's algorithm [3]. This algorithm determines whether an integer n is prime or not in $O((\log n)^3(\log\log n)(\log\log\log n))$ operations. Verifying whether an integer is prime or not is important before you start the attempt to factor the integer. It would be a waste of time trying to factor a prime number. Especially when your factoring algorithm does not always result in the right factors and you have to repeat the algorithm a number of times before you find the right factors.

5.6 Discrete Logarithm algorithm

Suppose we have a prime p and a generator g of the multiplicative group $\mathbb{Z}_p^*$. The discrete logarithm of a number x is the integer r with $0 \leq r \leq p-1$ such that $g^r \equiv x \bmod p$. This algorithm finds r when x, g and p are given. First we will discuss the easy case where we assume that $p-1$ is smooth. In this case we can construct an efficient quantum circuit for the quantum Fourier transform as we mentioned at the end of Section 5.2.

The algorithm uses three registers and we start the algorithm by preparing the two first registers in the following state

$$\tfrac{1}{p-1}\sum_{a=0}^{p-2}\sum_{b=0}^{p-2}|a\rangle|b\rangle.$$

Next we apply modular exponentiation to compute $g^a x^{-b} (\mathrm{mod}\ p)$, which is stored in the third register. The resulting state is

$$\tfrac{1}{p-1}\sum_{a=0}^{p-2}\sum_{b=0}^{p-2}|a\rangle|b\rangle|g^a x^{-b} (\mathrm{mod}\ p)\rangle.$$

Next we apply the quantum Fourier transform F_{p-1} on the first two registers, mapping $|a\rangle$ and $|b\rangle$ to the following states

$$|a\rangle \to \tfrac{1}{\sqrt{p-1}}\sum_{c=0}^{p-2} e^{2\pi iac/(p-1)}|c\rangle,$$
$$|b\rangle \to \tfrac{1}{\sqrt{p-1}}\sum_{d=0}^{p-2} e^{2\pi ibd/(p-1)}|d\rangle.$$

This leaves us in the following quantum state

$$\tfrac{1}{(p-1)^2}\sum_{a=0}^{p-2}\sum_{b=0}^{p-2}\sum_{c=0}^{p-2}\sum_{d=0}^{p-2} e^{2\pi i(ac+bd)/(p-1)}|c\rangle|d\rangle|g^a x^{-b} (\mathrm{mod}\ p)\rangle.$$

This is the resulting state of the algorithm which will be measured. The probability of obtaining a certain state $|c\rangle|d\rangle|y\rangle$ with $y \equiv g^k (\mathrm{mod}\ p)$ is equal to

$$\left|\tfrac{1}{(p-1)^2}\sum_{a,b:a-rb=k} e^{2\pi i(ac+bd)/(p-1)}\right|^2.$$

Where the sum is taken over all the amplitudes of the states that are consistent with the outcome of the measurement, in other words all the states for which it holds that $a - rb \equiv k(\mathrm{mod}\ p-1)$. If we substitute the equation $a \equiv rb + k(\mathrm{mod}\ p-1)$ in the expression above we obtain

$$\left|\tfrac{1}{(p-1)^2}\sum_{b=0}^{p-2} e^{2\pi i(kc+b(d+rc))/(p-1)}\right|^2.$$

Now if $d + rc \not\equiv 0(\mathrm{mod}\ p-1)$, this sum is equal to zero because it is a sum over a set of $(p-1)$ roots of unity equally spaced on the unit circle. This means that the outcome of the measurement is a random $c(\mathrm{mod}\ p-1)$ and $d \equiv -rc(\mathrm{mod}\ p-1)$. When c is relative prime with $p-1$, we find r by division. Because all possible c's have the same probability, the chance that c and $p-1$ are relative prime is $\phi(p-1)/(p-1)$, with $\phi(p-1)/(p-1) > 1/\log(p)$. This means that we only need a number of runs polynomial in $\log p$ to obtain r with a high probability.

In general $p-1$ is not smooth, which means that we first have to calculate a smooth integer q that is close to p to be able to perform the quantum Fourier transform in polynomial time. More precise we calculate q as a power of 2 such that $p < q < 2p$. The algorithm uses three registers as in the easy case. We start the algorithm in the same way as the easy case, namely we prepare the state

$$\tfrac{1}{p-1}\sum_{a=0}^{p-2}\sum_{b=0}^{p-2}|a\rangle|b\rangle|g^a x^{-b}(\text{mod } p)\rangle.$$

Again as in the easy case our next step is applying the quantum Fourier transform on the first two registers sending $|a\rangle \to |c\rangle$ and $|b\rangle \to |d\rangle$, but now we apply F_q instead of F_{p-1}. That is, we take the state $|a\rangle|b\rangle$ to the state

$$\tfrac{1}{q}\sum_{c=0}^{q-1}\sum_{d=0}^{q-1} e^{2\pi i\frac{ac+bd}{q}}|c\rangle|d\rangle.$$

The resulting state is then

$$\tfrac{1}{(p-1)q}\sum_{a,b=0}^{p-2}\sum_{c,d=0}^{q-1} e^{2\pi i\frac{ac+bd}{q}}|c\rangle|d\rangle|g^a x^{-b}(\text{mod } p)\rangle.$$

This is the final state that will be measured. The probability of observing a state $|c\rangle|d\rangle|y\rangle$ with $y \equiv g^k(\text{mod } p)$ is

$$\left|\tfrac{1}{(p-1)q}\sum_{a,b:a-rb=k} e^{2\pi i(ac+bd)/q}\right|^2,$$

where the sum is again over all a and b with $a - rb \equiv k(\text{mod } p-1)$. Because the denominator in the exponent is now q instead of $p-1$ we cannot simply substitute $rb+k$ for a in the equation. Instead we use the following relation

$$a = br + k - (p-1)\left\lfloor \tfrac{br+k}{p-1}\right\rfloor,$$

and substitute this in the expression of the amplitude to obtain

$$\tfrac{1}{(p-1)q}\sum_{b=0}^{p-2} e^{2\pi i(brc+kc+bd-c(p-1)\left\lfloor\frac{br+k}{p-1}\right\rfloor)/q}.$$

From this expression we can take out the global phase factor $e^{2\pi ikc/q}$ because it does not change the probability. Next we split the resulting exponent into two parts

$$\tfrac{1}{(p-1)q}\sum_{b=0}^{p-2} e^{2\pi ibT/q}e^{2\pi iV/q}.$$

with

$$T = rc + d - \tfrac{r}{p-1}\{c(p-1)\}_q,$$
$$V = \left(\tfrac{br}{p-1} - \left\lfloor \tfrac{br+k}{p-1} \right\rfloor\right)\{c(p-1)\}_q.$$

The notation $\{a\}_q$ means the residue of a (mod q).

We now classify possible outputs as "good" or "bad" and will show that if we get enough "good" outputs we can find r. An output is "good" if it satisfies two conditions. The first condition is

$$|T| \leq \tfrac{1}{2},$$

in this case as b varies between 0 and $p-2$ the phase of the first exponential term only varies over at most half the unit circle because $p-2 < q$, namely it varies from 0 to at most π. This means that the sum of $p-1$ complex numbers where the phase varies from 0 to exactly π is a lower bound for $\sum\limits_{b=0}^{p-2} e^{2\pi ibT/q}$:

$$\sum_{b=0}^{p-2} e^{(\frac{\pi b}{p-2})} \leq \sum_{b=0}^{p-2} e^{2\pi ibT/q}.$$

The second condition is

$$\{c(p-1)\}_q \leq \tfrac{q}{20};$$

in this case $|V|$ is always at least $-\frac{q}{20}$ because $\left(\frac{br}{p-1} - \left\lfloor \frac{br+k}{p-1} \right\rfloor\right)$ can vary from -1 to 0, and thus the second exponential term is never farther than $e^{\pi i/10}$ from 1. The phase of this second exponential term varies from $-\pi/10$ to 0. We will now show that both conditions will hold with constant probability and a reasonable sample of c's for which the first condition holds will allow us to find r.

A lower bound on the probability to obtain a "good" output can then be found by putting the previously found phases together. This lower bound is given in following formula:

$$\left| \tfrac{1}{(p-1)q} \sum_{b=0}^{p-2} e^{(\frac{\pi b}{p-2} - \frac{\pi}{10})i} \right|^2.$$

Because $e^{i\theta} = \cos(\theta) + i\sin(\theta)$ and $|e^{i\theta}| = \sqrt{\cos(\theta)^2 + \sin(\theta)^2}$ a lower bound on this probability is

$$\left|\frac{1}{(p-1)q}\sum_{b=0}^{p-2}\sin\left(\frac{\pi b}{p-2}-\frac{\pi}{10}\right)\right|^2.$$

Replacing this sum by an integral we find that this probability is approximately equal to

$$\left(\frac{1}{(p-1)q}\int_0^{p-2}\sin\left(\frac{\pi b}{p-2}-\frac{\pi}{10}\right)db\right)^2,$$

Thus, after change of variables, we find that the probability of observing a state $|c\rangle|d\rangle|y\rangle$ that satisfying both conditions is at least

$$\left(\frac{1}{q\pi}\int_{-\pi/10}^{9\pi/10}\sin(b)db\right)^2,$$

which is roughly equal to $0.366/q^2 > 1/3q^2$. Now the number of pairs (c, d) such that the first condition holds is equal to the number of possible c's because for every c there is exactly one d such that the first condition holds. The number of possible c's such that the second condition holds is approximatly $q/20$. This means that there are $q/20$ pairs (c, d) that satisfy both conditions. There are $p - 1$ possible y's so this gives us approximatly $pq/20$ states $|c\rangle|d\rangle|y\rangle$. Multiplying this number of "good" states with the lower bound of the probability of each good states gives us the probability of obtaining any "good" state, which is at least $p/60q$ or $1/120$ since $q < 2p$. Note that each "good" c has a probability of at least $(p - 1)/(3q^2) \geq 1/6q$ of being observed because there are $p - 1$ possible values of y and only one possible value of b with which c can make a "good" state.

We must now recover r from such a pair (c, d) with

$$-\frac{1}{2q} < \frac{d}{q} + \frac{rc}{q} - \frac{r\{c(p-1)\}_q}{q(p-1)} < \frac{1}{2q}.$$

This can be rewritten as

$$-\frac{1}{2q} < \frac{d}{q} + r\left(\frac{c(p-1)-\{c(p-1)\}_q}{q(p-1)}\right) < \frac{1}{2q}.$$

The multiplier on r can be written as a fraction where the denominator is $p-1$ because q divides $c(p - 1) - \{c(p - 1)\}_q$. If we would then round d/q off to the nearest fraction where the denominator is also $p - 1$ and divide by the integer

$$c' = \frac{c(p-1)-\{c(p-1)\}_q}{q},$$

we obtain a candidate r. As with the factoring algorithm, we only need to repeat this procedure a polynomial amount of time to find the right discrete logarithm [15].

Chapter 6

Conclusion

Peter Shor developed two quantum algorithms that efficiently solve two mathematical problems, namely the factoring problem and the discrete logarithm problem, which are used in the RSA cryptosystem and the ElGamal cryptosystem. Both cryptosystems are still used at this moment. A fully functional quantum computer where Shor's algorithm could be implemented with large enough parameters would thus be able to break those systems.

Shor's factoring algorithm has a polynomial running time, where the bottleneck is the modular exponentiation operation. A more efficient quantum circuit to implement this operation would also improve the efficiency of the factoring algorithm in general. With a theoretical analysis we found that the algorithm needs only a polynomial amount of repetitions to find the right factors of an certain integer with high probability. This result was supported by the outcome of the simulations carried out using the 'Libquantum' library.

Shor's second algorithm, the discrete logarithm algorithm, was very similar. It used the same quantum transformation and has the same running time. As for the factoring algorithm we only need to repeat the algorithm a polynomial amount of times to find the discrete logarithm with high probability.

Chapter 7

Bibliography

[1] David Beckman, Amalavoyal N Chari, Srikrishna Devabhaktuni, and John Preskill. Efficient networks for quantum factoring. *Physical Review A*, 54(2):1034, 1996.

[2] Björn Butscher and Hendrik Weimer. C-library for simulating quantum computations. `http://www.libquantum.de/`. Accessed: 2018-02-20.

[3] HF Chau and H-K Lo. Primality test via quantum factorization. *International Journal of Modern Physics C*, 8(02):131–138, 1997.

[4] Richard Cleve. A note on computing fourier transforms by quantum programs. *preprint*, 1994.

[5] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proc. R. Soc. Lond. A*, 400(1818):97–117, 1985.

[6] David Deutsch. Quantum computational networks. *Proc. R. Soc. Lond. A*, 425(1868):73–90, 1989.

[7] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 439, pages 553–558. The Royal Society, 1992.

[8] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[9] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[10] Daniel M Gordon. Discrete logarithms in gf(p) using the number field sieve. *SIAM Journal on Discrete Mathematics*, 6(1):124–138, 1993.

[11] Godfrey Harold Hardy and Edward Maitland Wright. *An introduction to the theory of numbers*. Oxford university press, 1979.

[12] Arjen K Lenstra, Hendrik W Lenstra Jr, Mark S Manasse, and John M Pollard. The number field sieve. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 564–572. ACM, 1990.

[13] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011.

[14] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[15] Peter W Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 124–134. Ieee, 1994.

[16] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[17] Daniel R Simon. On the power of quantum computation. *SIAM journal on computing*, 26(5):1474–1483, 1997.

[18] Umesh V. Vazirani. *Lecture notes on quantum computing*. Cambridge university press, Spring 2009.

[19] Igor V Volovich. Quantum computing and shors factoring algorithm. *arXiv preprint quant-ph/0109004*, 2001.

[20] Ronald Michiel Wolf. *Quantum computing and communication complexity.* Institute for Logic, Language and Computation, 2001.

[21] Andrew Chi-Chih Yao. Quantum circuit complexity. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*, pages 352–361, 1993.

Appendix A

Tensor product

A single qubit is a unit vector in the Hilbert space $\mathbb{C}^2$. If we have multiple qubits, the state space is the tensor product of the state spaces of the component qubit systems. The tensor product captures the essence of superposition, that is if the first qubit $|\psi\rangle$ is in state $|\psi\rangle = a|0\rangle + b|1\rangle$ and the second qubit $|\phi\rangle$ in state $|\phi\rangle = c|0\rangle + d|1\rangle$ then each amplitude of the 2-qubit system should have something of $|\psi\rangle$ and something of $|\phi\rangle$. The tensor product expresses this.

Let V and W be vector spaces with bases $\{v_1...v_n\}$ and $\{w_1...w_m\}$, the tensor product $V \otimes W$ is a nm-dimensional vector space spanned by $\{v_i \otimes w_j : 1 \leq i \leq n, 1 \leq j \leq m\}$, a basis for the tensor product space.
Let x be an element of V and y be an element of W, i.e. $x = \sum\limits_{i=1}^{n} x_i v_i$ and $y = \sum\limits_{j=1}^{m} y_j w_j$. The element $x \otimes y$ of the tensor product space $V \otimes W$ is then of the form

$$\sum_{i,j} a_{ij} v_i \otimes w_j,$$

with $a_{ij} = x_i y_j$.

If we take our two qubits $|\phi\rangle$ and $|\psi\rangle$ as an example, the tensor product $|\phi\rangle \otimes |\psi\rangle = ac|0\rangle \otimes |0\rangle + ad|0\rangle \otimes |1\rangle + bc|1\rangle \otimes |0\rangle + bd|1\rangle \otimes |1\rangle$. This is a unit vector in the Hilbert space $\mathbb{C}^2 \otimes \mathbb{C}^2 = \mathbb{C}^4$. In general, we represent an n-particle system by n copies of $\mathbb{C}^2$ tensored together, the notation for this is $(\mathbb{C}^2)^{\otimes n} = \mathbb{C}^{2^n}$.

When multiple transformations are applied on different sets of qubits, the resulting transformation on the whole system is also represented with the tensor product. Suppose we have two unentangled states $|v\rangle$ and $|w\rangle$ of n and m qubits respectively. The state of the combined $n+m$-qubit system is $|v\rangle \otimes |w\rangle$ on $\mathbb{C}^{2^{n+m}}$. When a unitary transformation A is applied on $|v\rangle$ and a unitary transformation B is applied on $|w\rangle$, the state of the combined system becomes $A|v\rangle \otimes B|w\rangle$.

However, in most cases the two subsystems $|v\rangle$ and $|w\rangle$ will be entangled, this means that the combined state cannot be written as a tensor product (see section 3.2). So when the transformation A is applied on the first subsystem and the transformation B is applied on the second subsystem, we say that the operator $A \otimes B$ is applied on the combined system. $A \otimes B$ is defined as follows:

Let $|v_1\rangle, ..., |v_{2^n}\rangle$ be a basis for the first subsystem and $|w_1\rangle, ..., |w_{2^m}\rangle$ be a basis for the second subsystem, we can write $A = \sum_{i,j=1}^{2^n} a_{ij}|v_i\rangle\langle v_j|$ and $B = \sum_{k,l=1}^{2^m} b_{kl}|w_k\rangle\langle w_l|$ with a_{ij} the i, jth element and b_{kl} the k, lth element of A and B respectively. A basis for the combined system is then $|v_i\rangle \otimes |w_j\rangle$ and the operator $A \otimes B$ is

$$\begin{aligned} A \otimes B &= \left(\sum_{ij} a_{ij}|v_i\rangle\langle v_j|\right) \otimes \left(\sum_{kl} b_{kl}|w_k\rangle\langle w_l|\right) \\ &= \sum_{ijkl} a_{ij}b_{kl}|v_i\rangle\langle v_j| \otimes |w_k\rangle\langle w_l| \\ &= \sum_{ijkl} a_{ij}b_{kl}(|v_i\rangle \otimes |w_k\rangle)(\langle v_j| \otimes \langle w_l|). \end{aligned}$$

The matrix for $A \otimes B$ is then

$$\begin{bmatrix} a_{11}B & a_{12}B & ... \\ a_{21}B & a_{22}B & ... \\ ... & ... & ... \end{bmatrix},$$

where in the (i, j)th subblock, we multiply a_{ij} by the matrix for B.

Appendix B

Quantum swap operation

The quantum circuit in Figure B.1 accomplishes a simple but useful operation, namely it swaps the states of two qubits. This circuit consists of three CNOT gates where for the second CNOT gate the control bit and target bit are switched. To see that this circuit swaps the states of two qubits, note that the sequence of gates has the following sequence of effects on a certain basis state $|a, b\rangle$:

$$\begin{aligned} |a, b\rangle &\rightarrow |a, a \oplus b\rangle \\ &\rightarrow |a \oplus (a \oplus b), a \oplus b\rangle = |b, a \oplus b\rangle \\ &\rightarrow |b, (a \oplus b) \oplus b\rangle = |b, a\rangle. \end{aligned} \tag{B.1}$$

The circuit indeed swaps the states of two qubits.

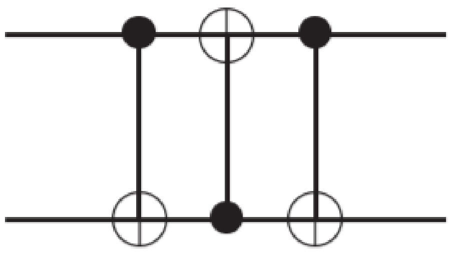

Figure B.1: Quantum circuit swapping two qubits

Appendix C

Continued fraction

We will explain continued fractions in the framework of Shor's algorithms. For a more detailed analysis we refer to [11, Chapter X].

Every positive rational number x can be written as an expression in the form

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{... + \cfrac{1}{a_n}}}}.$$

where a_0 is a non-negative integer and $a_1, ..., a_n$ are positive integers. Such an expression is called a continued fraction or continued fraction expansion of x and is uniquely defined for each different x under the condition $a_n > 1$. Another notation is $[a_0, a_1, ..., a_n]$.

Given x, the following procedure gives the continued fraction expansion of x:

$$a_0 = \lfloor x \rfloor, x_1 = 1/(x - a_0),$$
$$a_1 = \lfloor x_1 \rfloor, x_2 = 1/(x_1 - a_1),$$
$$a_2 = \lfloor x_2 \rfloor, x_3 = 1/(x_2 - a_2),$$
$$...$$

This procedure ends when we find a x_n which is an integer.

For example, take the rational integer $x = \frac{415}{93}$, we find the continued fraction as follows

$$a_0 = \lfloor \tfrac{415}{93} \rfloor = 4, x_1 = 1/(x - a_0) = \tfrac{93}{43},$$
$$a_1 = \lfloor \tfrac{93}{43} \rfloor = 2, x_2 = 1/(x_1 - a_1) = \tfrac{43}{7},$$

$$a_2 = \lfloor \tfrac{43}{7} \rfloor = 6, x_3 = 1/(x_2 - a_2) = 7,$$
$$a_3 = 7.$$

Thus we find the following continued fraction expansion

$$\tfrac{415}{93} = 4 + \cfrac{1}{2 + \cfrac{1}{6 + \cfrac{1}{7}}}.$$

The fraction $x_m = [a_0, a_1, ..., a_m]$, with $0 \le m \le n$, is called the *m-th convergent* of the continued fraction expansion of x. In Shor's algorithm we use the following theorem:

Theorem 2. *If x is a rational number and a and b are positive integers satisfying*

$$|\tfrac{a}{b} - x| < \tfrac{1}{2b^2},$$

then a/b is a convergent of the continued fraction of x.