CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

A Taxonomy of Quantum Algorithms

The core ideas of existing quantum algorithms and their implications on cryptography

Master's thesis in Computer Systems and Networks

ANDERS STIGSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Master's thesis 2018

A Taxonomy of Quantum Algorithms

The core ideas of existing quantum algorithms and their implications on cryptography

ANDERS STIGSSON

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

A Taxonomy of Quantum Algorithms
The core ideas of existing quantum algorithms and their implications on cryptography
ANDERS STIGSSON

Supervisor: Elena Pagnin, Computer Science and Engineering
Supervisor: Pablo Picazo-Sanchez, Computer Science and Engineering
Examiner: Andrei Sabelfeld, Computer Science and Engineering

Master's Thesis 2018
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2018

A Taxonomy of Quantum Algorithms
The core ideas of existing quantum algorithms and their implications on cryptography
ANDERS STIGSSON
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

With quantum computers coming up as one of the fastest growing areas in multiple research areas, such as computer science and physics, a taxonomy of the existing quantum algorithms is necessary. However, before this thesis, no taxonomy which included many of the existing quantum algorithms could be found. We have filled that gap with this thesis. The result is a taxonomy with 31 algorithms. Each algorithm are classified into different groups depending on the characteristics and the core idea that the algorithm uses. We have focused on three different core ideas distributed as 33% using the Quantum Fourier Transform, 27% uses Amplitude Amplification and 15% uses Quantum Walks with the remaining 25% being classified as "Other". On top of this, we also discuss the security implications on the cryptographic schemes used today, once quantum computers become reality. A taxonomy about an area that expands as fast as quantum computing is never finished, but we believe that this thesis provides a good base for future work in the area. This thesis can also be used as an introduction to quantum computing for students with a base knowledge about computer science and mathematics.

Keywords: quantum computers, quantum algorithms, computer science, cryptography.

Acknowledgements

I would like to thank my supervisors Elena Pagnin and Pablo Picazo-Sanchez for the vast amount of help you have given me in a topic that neither of us had much previous knowledge about.

Anders Stigsson, Gothenburg, June 2018

Contents

List of Figures

List of Tables

1

Introduction

The principles of classical computers, or just computers as we know them, were first proposed by Alan Turing in 1936. In the 1950's more and more computers were built and nowadays almost everyone owns at least one computer. These classical computers use registers containing bits, where every bit can either store the value 0 or 1. These bits can then be manipulated using instructions, allowing computers to run algorithms and programs. The Church-Turing (or rather the extended Church-Turing) thesis states that:

> Any reasonable model of computation can be efficiently simulated on a standard model, e.g. a Turing Machine [1].

There is no doubt that Alan Turing and Alonzo Church believed that their thesis would be broad enough to catch all kinds of computers that could be built. However, when it comes to quantum computers this might not be true anymore. There are no absolute proofs that break the Church-Turing thesis, but the indications are definitely there. In this master thesis we will see that quantum computers can perform computations that are believed to take superpolynomial time, such as prime factorisation of integers and the discrete logarithm problem, in polynomial time. To be absolutely sure that quantum computers break the extended Church-Turing thesis, two things have to be determined: 1. that the task, e.g. integer factorisation, actually is a problem that takes exponential time on a classical computer, and 2. that a quantum computer is a reasonable model of computation.

A model of computation is said to be reasonable if the model is possible to use and build physically, i.e. something that can be used in the real world [1]. The model has to be digital, since an analog model needs infinite precision to be able to do computations and infinite precision is not physically realisable. Normal (digital) computers are, of course, reasonable in this sense, but the question one must ask is: are quantum computers reasonable? It is unclear whether we can say that a quantum computer is digital since the gates that are used in a quantum computer are unitary linear transformations, i.e. matrices, built up by complex numbers and thus are not digital. However, if it is possible to approximate the complex numbers so that infinite precision is not necessary, they might still be considered digital. The second property that must be fulfilled for quantum computers to be reasonable is that it should be possible to build and implement an arbitrary quantum gate on a large number of qubits.

Quantum computers were first proposed in 1980 by Benioff [2] and are based upon using quantum mechanical phenomena to do computations. Even though this was over 35 years ago, there are still no commercially available quantum computers for regular users. There are however companies, such as IBM, that are working on building larger quantum computers [3]. The big difference between quantum and classical computers are that the former runs algorithms on qubits (quantum bits). Qubits represent data "in a more powerful way" than classical bits, as we will see in Chapter 2. This gives quantum algorithms the possibility to solve problems which cryptosystems rely on to be unfeasible to solve.

Cryptography provides a way to hide information from any unintended recipient and has been used for centuries. Historically, the first and most famous scheme is due to the Roman Emperor Julius Caesar (Caesar cipher) in the 1st century BC. The idea behind this encryption method is to make messages unintelligible by shifting the letters in the alphabet three steps [4]. This was fairly secure at the Roman times, however the computational power of modern computers allows caesar-like ciphers to be broken in a very efficient way. This is due to the distribution of letters in the alphabet and how they are combined in meaningful words. For example, in English, E is the most common letter to be used, which means that a computer is able to find out the shift of the alphabet using statistical analysis of the ciphertext [4]. More recent schemes are based on problems that are believed to be computationally hard, i.e. take a superpolynomial amount of time to be solved using classical computing, for modern computers (further described in Section 2.7). Examples of such hard problems are solving the discrete logarithm in finite groups [5], and factoring an integer into prime factors [6].

Cryptography is part in our everyday life, e.g. whenever you go to a webpage that uses HTTPS, you are likely to use RSA to exchange keys with which the communication with the website is encrypted. RSA uses the previously mentioned problem of integer factorisation to prove that the cryptosystem is practically impossible to break. Should you not use RSA, the second most probable key exchange system is the Diffie-Hellman (DH) key exchange [7]. DH relies on another hard problem, namely the discrete logarithm problem. Unfortunately, Shor [8] developed a quantum algorithm that can break both of these problems in polynomial time.

The most widely known quantum algorithms are two algorithms that were developed in the 90's by Shor [8] and Grover [9] (see Chapter 3.3). To the best of our knowledge, there are no good taxonomies available that categorise all the other quantum algorithms developed so far. The aim of this thesis is to close this gap. We will look at whether all algorithms are based on the two previously mentioned algorithms, or if the authors have found other possibilities to speed up the algorithms. We will also investigate what the core ideas of the algorithms are, such as if they use a quantum version of the Fourier Transform. Finally we will describe what the implications on the cryptosystems that we use today will be when quantum computers become more and more available to companies, governments and ultimately normal users.

This thesis will begin with a background on quantum computing in Chapter 2. We will describe how the Quantum Fourier Transform, Quantum Walks and Amplitude

Amplification works in Chapter 3. The taxonomy is presented in Chapter 4. Finally, the conclusions of the thesis will be described and discussed in Chapter 5.

2

Background on Quantum Computing

2.1 Qubits

Qubits are the quantum analog to bits in a classical computer. Like classical bits, Qubits have a state (0,1), but in addition qubits can also have any state *between* 0 and 1. Intuitively, one can imagine that qubits can be both 0 and 1 at the same time. Quantum states are generally described using the Bra-Ket notation, $|\cdot\rangle$. Thus a single qubit is described as $|0\rangle$ and $|1\rangle$. The Bra-Ket notation is also sometimes called Dirac notation, named after the British physicist, and Nobel laureate, Paul Dirac [10]. The quantum states can be seen as vectors where

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.1}$$

These states can also be in a superposition which means that they can be both $|0\rangle$ and $|1\rangle$ with different probabilities at the same time (more details in Section 2.2). To find the value, the qubits must be measured. Once measured, the qubits collapse into 0 or 1 and lose their superposition. Intuitively, think of measurements as Schrödinger's cat. This thought experiment consists of a cat in a steel box with a piece of radioactive matter that has a 50% chance of decaying within 1 hour. This means that, after 1 hour, inside the steel box the cat is both alive and dead at the same time, until we open the box, i.e. measures the system, and if the cat is dead it will stay dead. If there are more qubits, there will also be more states, e.g. with two qubits the states will be $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$. In this case, the classical vectors one is computing on are:

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.2}$$

These states, with one or two qubits, are called the computational basis [10]. They can be seen as the basis of the vector space within which we do the calculations, since every vector in the space is a linear combination of these states. The basis

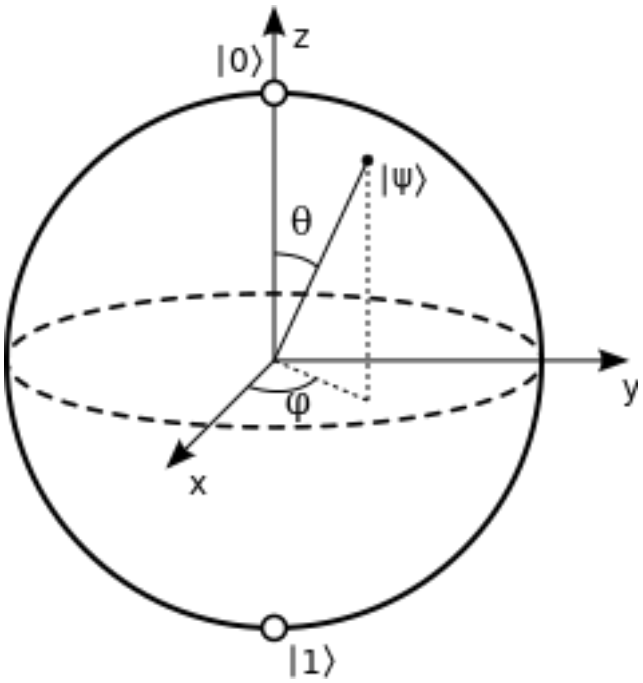


Figure 2.1: The Bloch sphere used to visualise qubit states [11].

vectors lie on a sphere, called the Bloch sphere, where $|0\rangle$ and $|1\rangle$ are at each end of the Z-axis, depicted in Figure 2.1.

2.2 Superpositions

As previously mentioned, qubits can be in two (or more) states at the same time called superpositions. Superpositions can be seen as linear combinations of states, such as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.3}$$

where α and β are complex numbers called the amplitude of the two states. An example of this can be when $\alpha = \beta = \frac{1}{\sqrt{2}}$ meaning that the states have the same amplitude. In the case where the amplitudes are equal for both $|0\rangle$ and $|1\rangle$, we say that they are in equal superposition and when the qubit is measured, the probability of collapsing into either state is equal. Such qubit can be seen as a truly random coin-flip where one of the states is heads and the other is tails. The probability that the qubit collapses into either 1 or 0 is equal to the magnitude of the amplitude squared, i.e. $P(|0\rangle) = |\alpha|^2 = (\frac{1}{\sqrt{2}})^2 = \frac{1}{2}$ and $P(|1\rangle) = |\beta|^2 = (\frac{1}{\sqrt{2}})^2 = \frac{1}{2}$ [12]. This tells us that in a true empirical example, half of the times we measure the qubit we will get 0 and half of the time we will get 1, whereas in reality we can not say with certainty that we would see the exact equal amount of 0 and 1 since we are working with probabilities. These probabilities must always add up to 1, i.e. $|\alpha|^2 + |\beta|^2 = 1$.

The amplitude or probability of a state can be shown in the sense of vectors as well

where we multiply the vectors with the amplitudes, i.e.

$$\alpha |0\rangle + \beta |1\rangle = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{2.4}$$

2.3 Phase

Phase is a word that is widely used in quantum mechanics with different meanings depending on the context. In the context of quantum computing, we encounter two different kinds of phases that are used: the global phase factor and the relative phase factor.

2.3.1 Global phase

A global phase factor is when the phase is the same upon both amplitudes such as $e^{i\theta}(|\alpha\rangle + |\beta\rangle)$. This means that the probability of measuring α or β does not change since for an arbitrary phase factor $e^{i\theta}$ the magnitude squared is 1 ($|e^{i\theta}|^2 = 1$), thus not making any difference in the measurement of the two states. This means that we can say that $e^{i\theta} |\psi\rangle = |\psi\rangle$ up to the global phase factor $e^{i\theta}$, if $|\psi\rangle$ is a state vector and θ is a real number [12]. Alas, from an observational point of view, these states are identical.

2.3.2 Relative phase

Two states can differ by a relative phase, such as the state that we will see in Section 2.4 when we apply a specific gate. In that case the states are $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$, such that the amplitude of $|1\rangle$ is $\frac{1}{\sqrt{2}}$ and $-\frac{1}{\sqrt{2}}$, i.e. they differ by a relative phase of -1. These two states can not be seen as identical in an observation point of view, even though the probabilities of seeing the two states are the same when working in the computational basis [10].

2.4 Gates

In classical computing there are certain logical gates that are used in a computer, e.g. AND-, OR-, NOT- and NAND-gates. Gates are also used in quantum computing but there are restrictions on how a gate is constructed. Before going in to depth on the restrictions, we must understand what a quantum gate is. The NOT-gate is an interesting gate to take as an example since it is rather easy to understand. Classically, the NOT-gate changes $0 \rightarrow 1$ and $1 \rightarrow 0$. The quantum NOT-gate should achieve the same behaviour in the quantum setting. Since we work in superpositions and states with amplitudes, what we want to do is to swap the amplitudes of $|0\rangle$

and $|1\rangle$, i.e. $(\alpha|0\rangle + \beta|1\rangle) \rightarrow (\beta|0\rangle + \alpha|1\rangle)$. Since the states can be seen as vectors, gates can be seen as matrices, and gate evaluation corresponds to matrix-vector multiplication. From this, we can see that the matrix X in Equation 2.5 solves the problem of swapping the amplitudes. This gate is called the **Pauli-X** or just the NOT-gate [12].

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, X|\psi\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \tag{2.5}$$

So we see that, in the quantum world, the gates are matrices. More precisely, for a single qubit, a gate is a 2×2 matrix. However, it is not possible to use any 2×2 matrix as a gate for meaningful quantum computations. The restriction is that the matrix must be unitary, which implies that it is square and complex. This restriction is due to working on a sphere, and if the matrix is unitary, the result of the multiplication will be a vector that is also on the sphere, thus making it possible to compute amplitudes and measure the qubits. Formally, for a matrix $U \in \mathbb{C}^{nxn}$ and the complex conjugate transpose of U, $U^\dagger$, a unitary matrix implies:

$$U^\dagger U = I \tag{2.6}$$

It is easy to see that this holds for the **Pauli-X** gate since $X^\dagger = X$

$$X = X^\dagger = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I \tag{2.7}$$

Thus, we can define the set of all possible gates in quantum computing as $G = \{U : U \in \mathbb{C}^{n \times n}, U^\dagger U = I\}$ [13]. Notice that there are possibilities to create several different custom gates according to one's needs. In what follows, we present the most common gates that will be used throughout this thesis.

The **Hadamard gate** (or Welsh-Hadamard gate) is denoted by the letter $H \in G$. This gate creates the equal superposition that we saw in Section 2.2 where both states had the same probability [12]. The matrix corresponding to this gate is

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \tag{2.8}$$

When we apply this gate to the input qubit in states $|0\rangle$ and $|1\rangle$ we will get some different results

$$\begin{aligned} H|0\rangle &\rightarrow \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \\ H|1\rangle &\rightarrow \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \end{aligned} \tag{2.9}$$

We can see that when applied to the input qubit in state $|1\rangle$, the amplitude becomes negative on the second row. However, this has no implication in regards to the probability, since the probability is the magnitude of the amplitude squared, so

negative amplitudes do not incur any changes in the probabilities [12]. The two states that we get when applying the Hadamard gate are sometimes denoted as $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$.

As with the Pauli-X matrix, $H^\dagger = H$ so the same calculations can be done and it is trivial to see that this matrix is unitary as well.

The **Pauli-Z** is another interesting gate, which makes no changes if the input is $|0\rangle$ but flips the sign on the input $|1\rangle \rightarrow -|1\rangle$ [10]. The matrix for this operation is

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.10}$$

The circuits below, with input on the left side and output on the right, show the functionality of these 3 gates on inputs that are in superpositions of states:

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

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

$$\alpha|0\rangle + \beta|1\rangle \text{ ——} \boxed{H} \text{——} \quad \alpha\tfrac{|0\rangle+|1\rangle}{\sqrt{2}} + \beta\tfrac{|0\rangle-|1\rangle}{\sqrt{2}} = \tfrac{\alpha+\beta}{\sqrt{2}}|0\rangle + \tfrac{\alpha-\beta}{\sqrt{2}}|1\rangle$$

The **CNOT-gate** (ControlledNOT) is another important gate that is analogous to the classical XOR-gate with the result of the XOR stored in the second qubit [10]. It takes two qubits as input, one which is the control qubit and the other which will be the target qubit. If the control qubit is 1, we apply an X-gate to the target qubit and if the control qubit is 0, nothing happens. The matrix for the CNOT is:

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

As we can see, the upper left corner of the matrix is just the identity matrix, which act on the control qubit, so that we don't change that qubit, and the bottom right corner is the X-gate that act on the target qubit.

$$CNOT|01\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = |01\rangle$$

$$CNOT|10\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = |11\rangle$$

$$CNOT\,|00\rangle = |00\rangle$$

$$CNOT\,|11\rangle = |10\rangle$$

The **phase gate (S)** looks rather similar to the Pauli-Z [10]. However, instead of changing the sign of the $|1\rangle$-input, it adds a phase factor i, i.e. $|0\rangle \rightarrow |0\rangle$ and $|1\rangle \rightarrow i\,|1\rangle$. The matrix for this gate is:

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

2.5 Entanglement

Two entangled qubits are connected to each other in a quantum mechanical sense. This means that when we perform actions on one qubit, there will be changes in the other qubits characteristics as well.

Intuitively, to understand entanglement of two qubits, we can think of the qubits as two balls with different colours (e.g. blue and green). We put the two balls in two identical containers and shuffle the containers around. At this moment, before opening any box, it is the same probability that when I choose a box it contains the blue and the green ball. After opening one container, the disposition of the balls is fully determined and we know exactly what the other container contains without opening it. We could say that when "measuring" the first ball, we also measure the second ball so that it "collapses" to the colour, since from the beginning the items in the two containers were in superpositions of the two balls. This is similar to Schrödinger's cat since after we have shuffled the containers, each of the containers have a ball that is both blue and green, but as soon as we open the container we know which colour it is and also which colour the other ball is.

Equation 2.11 shows a Bell state, also known as an EPR-pair from possibly the three most famous researchers in the field of quantum mechanics; Einstein, Podolsky and Rosen. These two qubits in the state $|\psi\rangle$ are entangled which we can see since if we measure the first qubit and get a 0, the second qubit must also be a 0 since the only state where the first qubit is 0 is $|00\rangle$ and vice versa if we measure 1, the other qubit is 1 [12].

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{2.11}$$

2.6 An example - Quantum Teleportation

By using entangled states, we can perform an interesting phenomenon that is called quantum teleportation. The goal of quantum teleportation is to move a quantum

state between two points that are in different locations, even far away from each other (in May 2010 Jin *et al.* [14] managed to send a state 16km with 89% accuracy). Let us assume that we have two persons, Alice and Bob, that meet up once and exchange their telephone numbers and also create a Bell-state with two qubits. They then move far away from each other and take one of the qubits each with them. Now, Alice wants to send a quantum state $|\psi\rangle$ to Bob, but she can not inspect the state since it would collapse and no longer be a quantum state if she did. Thus, it is impossible for her to describe the state to Bob in any way. However, Alice can perform operations on her part of the Bell-state together with the state $|\psi\rangle$ and measure them to get one of four different results, namely $00, 01, 10$ or 11 and communicate the result of the operations to Bob. After obtaining the result, Bob can recreate the state $|\psi\rangle$ without any issue. Below follows an example of the calculations that are done, these can be seen as the general way to perform quantum teleportation even though parameters might change, e.g. another Bell-state can be used instead of the chosen one below [12].

Example 2.6.1. The Bell-state that Alice and Bob shared is $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$, where we say that Bob has the second qubit, i.e. $|0\underline{\mathbf{0}}\rangle$ and $|1\underline{\mathbf{1}}\rangle$. The state $|\psi\rangle$ is in some superposition in the computational basis, i.e. $\alpha|0\rangle + \beta|1\rangle$. This means that the state of all qubits is

$$|\psi_0\rangle = |\psi\rangle|\beta_{00}\rangle = \frac{1}{\sqrt{2}}\Big[\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|00\rangle + |11\rangle)\Big] \tag{2.12}$$

Alice then applies a CNOT-gate to her two qubits, with the control on the qubit in state $|\psi\rangle$ and the target on her qubit in the Bell-state. This would change the state into

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}\Big[\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|10\rangle + |01\rangle)\Big] \tag{2.13}$$

After this, Alice applies a Hadamard-gate to the state $|\psi\rangle$ to achieve

$$|\psi_2\rangle = \frac{1}{2}\Big[\alpha(|0\rangle + |1\rangle)(|00\rangle + |11\rangle) + \beta(|0\rangle - |1\rangle(|10\rangle + |01\rangle)\Big] \tag{2.14}$$

which can be rewritten to

$$\begin{aligned}|\psi_2\rangle = \frac{1}{2}\Big[&|00\rangle(\alpha|0\rangle + \beta|1\rangle) + |01\rangle(\alpha|1\rangle + \beta|0\rangle) \\ &+ |10\rangle(\alpha|0\rangle - \beta|1\rangle) + |11\rangle(\alpha|1\rangle - \beta|0\rangle)\Big]\end{aligned} \tag{2.15}$$

We can break this equation into four different terms depending on what Alices' qubits are measured to, i.e. the two qubits outside the parentheses. If Alices' qubits are in state $|00\rangle$ then Bob's qubit is $\alpha|0\rangle + \beta|1\rangle$, which is the state $|\psi\rangle$ that we

wanted to teleport from the beginning. However, if Alice has another constellation of her qubits when she measures, Bob will have to apply some gate (or gates) to his qubit to obtain the state that was to be teleported. The gate to be applied depending on Alice's qubits can be seen in Table 2.1.

Alice	Bob	Gate
$\|00\rangle$	$\alpha\|0\rangle + \beta\|1\rangle$	No gate
$\|01\rangle$	$\alpha\|1\rangle + \beta\|0\rangle$	X-gate
$\|10\rangle$	$\alpha\|0\rangle - \beta\|1\rangle$	Z-gate
$\|11\rangle$	$\alpha\|1\rangle - \beta\|0\rangle$	X-gate then Z-gate

Table 2.1: How to do the last step to teleport the state from Alice to Bob

To understand what gate to apply, we can easily see that when Alice has $|01\rangle$, and we apply the X-gate to Bob's qubits, we change the amplitudes $\alpha|1\rangle + \beta|0\rangle$ to $\alpha|0\rangle + \beta|1\rangle$ as we described in Section 2.4.

2.7 Algorithms and time calculations

To analyse the computations required by algorithms we often speak about the time and space that is needed to perform the algorithm. Usually, algorithms are either constant, polynomial or exponential in the size of their input. Formally, an algorithm with an n-bit input has either one of the following running times:

- $\mathcal{O}(C)$ for a constant $C > 0$
- $\mathcal{O}(n^x) : x < n$ (polynomial)
- $\mathcal{O}(c^n)$ (exponential)

$\mathcal{O}(\cdot)$ means that the growth rate of the running time of an algorithm is bounded asymptotically by the expression inside the $\mathcal{O}()$. When we discuss problems that are computationally hard, we usually mean problems that can not be solved faster than in exponential time. There exist algorithms that have a larger running time than the three above, e.g. $\mathcal{O}(n!)$, but in the scope of this thesis we will only discuss the three above.

3

Taxonomy classes

In this chapter, we will describe the quantum Fourier transform (QFT) which plays a fundamental role in many of the quantum algorithms publicly available today, including Shor's algorithm. We will also describe the quantum analog to random walks, the Quantum Walks. Lastly, amplitude amplification will be shown in terms of the Grover's algorithm.

3.1 Quantum Fourier Transform

The quantum Fourier transform is analogous to the discrete Fourier transform used in classical computing. What it does is to transform the entire quantum state from the position space to the momentum space. This enables us to perform necessary steps for many algorithms, such as the period-finding step in Shor's algorithm for factoring. The classical Fourier transform finds periodicity in functions, while the QFT finds periodicity in wavefunctions [15]. The quantum Fourier transform in itself is a unitary matrix consisting of powers of ω, where $\omega = e^{\frac{2\pi i}{N}}$, and a normalisation factor in front of the matrix which is $\frac{1}{\sqrt{N}}$, $N = 2^n$, where n is the number of qubits.

Example 3.1.1. We show how the QFT matrix are built for 2 and 3 qubits, with emphasis on the case with 2 qubits. For 2 qubits, $N = 2^2 = 4$ and $\omega = e^{\frac{2\pi i}{4}} = i$, which gives the following roots of unity: $\omega^0 = 1$, $\omega^1 = i$, $\omega^2 = i^2 = -1$, $\omega^3 = i^3 = -i$ [15]. So when we fill in the matrix it will look like the matrix in Equation 3.1.

$$QFT_4 = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & i^2 & i^3 \\ 1 & i^2 & i^4 & i^6 \\ 1 & i^3 & i^6 & i^9 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \tag{3.1}$$

If we extend the transform to 3 qubits, the roots of unity become less nice numbers, compared to the numbers in the case with 2 qubits. Because of this, the matrix

for the quantum Fourier transform with 3 qubits is shown only with ones and the 7 different powers of ω, as seen in Equation 3.2 [12].

$$QFT_8 = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{bmatrix} \tag{3.2}$$

Now, to implement a quantum Fourier transform, we must be able to create a quantum circuit that perform the transform. It is actually not very hard and it basically contains two different gates, with a few changes along the way [15]. First off, we will use the Hadamard gate, which we've already seen in Section 2.4, and we will need a controlled-R gate, which is a rotation gate that uses the same kind of control that the CNOT-gate does, but instead of a bit-flip it performs a rotation according to the matrix in Equation 3.3.

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{-2\pi i}{2^k}} \end{bmatrix} \tag{3.3}$$

Now we need to apply the Hadamard gate to the first qubit, then apply controlled-R_2 with control on the second qubit, controlled-R_3 with control on the third qubit and so on. We then continue this on all qubits as we see in the circuit for 3 qubits in Figure 3.1.

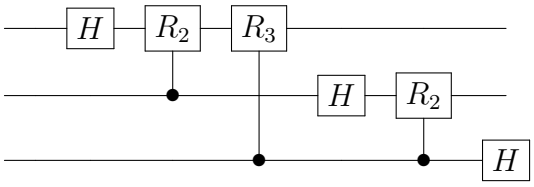


Figure 3.1: The quantum Fourier transform on 3 qubits.

Should there instead be 4 qubits, we would have added an R_4 gate with control on the fourth qubit on the first wire, an R_3 gate on the second wire and so on.

After running the circuit, we will have the transformed state if we read the qubits backwards, that is if we used to have little-endian, we need to read them as big-endian and vice versa to get the correct result of the quantum Fourier transform. This could be done with a gate called the SWAP-gate, but since it is equally easy to read the qubits "backwards", we have omitted the SWAP-gates from the circuit [12].

3.2 Quantum Walks

Quantum walks are the quantum counterpart to classical random walks. They can be of the discrete kind, where every step is a discrete time step, or of the continous kind, where the steps are continuosly applied. Both of these can also be on different underlying structures, such as a line or a graph. In this section we will focus on the discrete quantum walks on a line. First, a short introduction to classical random walks and then we will follow that up with the quantum walks.

3.2.1 Classical Random Walks

A random walk is a stochastic process where random steps decide the outcome of the process. To perform a classical random walk on a line, all we need to have is an initial position and a coin, potentially biased but most likely unbiased. The easiest to understand would be an integer line with the inital position at 0 and the next position to the left of 0 is -1, and right of 0 is +1. To make a step in a random walk we need to flip the coin, and depending on whether it is heads or tails, move to the position one step left or right respectively. The mean after T steps is 0 and the standard deviation from the inital position, given an infinite line, is the square root of T.

3.2.2 Discrete Quantum Walks

Aharanov and Davidovich [16] wrote what is generally accepted as the first paper with quantum walks as the main topic. The setup for discrete quantum walks is quite similar to what we had in classical random walks. We need a "coin" and we need a "walker", which are analogous in intuition to the coin and the position. However, since we now are in the quantum world, the coin can not be a normal coin and the position of the walker is not known until we measure the system. Instead we have the unitary operator S in Equation 3.4.

$$S = |0\rangle\langle 0| \otimes \sum_i |i+1\rangle\langle i| + |1\rangle\langle 1| \otimes \sum_i |i-1\rangle\langle i| \tag{3.4}$$

S transforms the state $|0\rangle \otimes |i\rangle$ to $|0\rangle \otimes |i+1\rangle$ and $|1\rangle \otimes |i\rangle$ to $|1\rangle \otimes |i-1\rangle$, where the first $|\cdot\rangle$ is the coin state and the $|i\rangle$ is the walker at position i [17].

The first step of the quantum walk is to rotate the coin space, which is analogous to the coin-flip in the classical walk and thus we will call it a coin-flip as well. We want the coin-flip to be unbiased, i.e. if we measure the coin state after one iteration of the walk, we should get the same probability of going left or right, $\frac{1}{2}$, as we had with the classical walk if we used an unbiased coin. Therefore we need to use a balanced unitary coin, e.g. the Hadamard coin H:

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

As we know from the introduction to quantum computing in Chapter 2, the Hadamard gate creates equal probabilities for the two states and here we use that knowledge to create a balanced coin for the coin-flip. To show that the Hadamard coin is balanced we can apply it to an initial setup for a quantum walk. In Equation 3.5, the first $|0\rangle$ is as before the coin, upon which we apply the Hadamard, and the $|0\rangle$ after the tensor product ($\otimes$) is the position state.

$$|0\rangle \otimes |0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle \tag{3.5}$$

In Equation 3.6, we see what happens when we apply the S-operator after applying the Hadamard and we will notice that the amplitude of moving to the left or to the right, i.e. the position being $|1\rangle$ or $|-1\rangle$ when we measure is $\frac{1}{\sqrt{2}}$ and thus the probability is $\frac{1}{2}$ [17].

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle \xrightarrow{S} \frac{1}{\sqrt{2}}(|0\rangle \otimes |1\rangle + |1\rangle \otimes |-1\rangle) \tag{3.6}$$

If we were to measure the coin in each step throughout the walk, we would just have the classical random walk, since at each step we would either walk left or right and no quantum phenomena would be involved in the algorithm. Instead we want to apply H and S a number of times before measuring the entire system to see where the walker ended up. To make it easier to understand we will name a transformation, U, where U^T denotes the quantum walk for T steps [17]:

$$U = S \cdot (H \otimes I) \tag{3.7}$$

In Equation 3.8 we can see how the system evolves after applying U three times to the initial state $|1\rangle \otimes |0\rangle$.

$$\begin{enumerate}
\item $\frac{1}{\sqrt{2}}(|0\rangle \otimes |1\rangle - |1\rangle \otimes |-1\rangle)$
\item $\frac{1}{2}(|0\rangle \otimes |2\rangle - (|0\rangle - |1\rangle) \otimes |0\rangle + |1\rangle \otimes |-2\rangle)$
\item $\frac{1}{2\sqrt{2}}(|0\rangle \otimes |3\rangle + |1\rangle \otimes |1\rangle + |0\rangle \otimes |-1\rangle - 2\,|1\rangle \otimes |-1\rangle - |1\rangle \otimes |-3\rangle)$
\end{enumerate} \tag{3.8}$$

Table 3.1 shows the probabilites of being in the different positions (i) after T steps. We see that the probabilities skew to the left which has to do with the initial state. If we would have used $|0\rangle \otimes |0\rangle$ as initial state, the skew would have been to the

right instead [17]. To solve this, we can use an equal superposition of $|0\rangle$ and $|1\rangle$ as initial state of the coin.

Table 3.1: Probabilities of being in position i (columns) after T steps (rows).

T \ i	-3	-2	-1	0	1	2	3
0				1			
1			$\frac{1}{2}$		$\frac{1}{2}$		
2		$\frac{1}{4}$		$\frac{1}{2}$		$\frac{1}{4}$	
3	$\frac{1}{8}$		$\frac{5}{8}$		$\frac{1}{8}$		$\frac{1}{8}$

Whereas the classical walk had a standard deviation from the initial position of $\sqrt{T}$, for quantum walks the standard deviation from the initial position is T, which means that we walk quadratically further with quantum walks as compared to classical walks, thus giving us a possible speed-up for algorithms [17].

3.3 Grover's Algorithm

Grover's algorithm [9] is one of the most famous quantum algorithms. Its popularity is due to the fact that it can be used to solve one realistic problem. By using superpositions, amplitude inversion and inversion about the mean, Grover's algorithm can find a sought element in an unstructured search space of size N with $\mathcal{O}(\sqrt{N})$ function calls to an oracle (realised as a black-box). On a classical computer, finding an element in an unstructured database would take $\mathcal{O}(N)$ function calls. The algorithm is a sequence of 3 steps with a final loop and the remaining subsections follow this structure.

3.3.1 Initialisation

The first step of Grover's algorithm is to put all qubits in equal superpositions, i.e. all outputs have the same amplitude, and thus the same probability. This is computationally achieved by applying a Hadamard gate to all the qubits, which will leave the qubits in the state

$$\frac{1}{\sqrt{N}} \sum |\psi\rangle \tag{3.9}$$

Visually, the result of applying the Hadamard gate can be seen in Figure 3.2 which shows the amplitudes of all qubits.

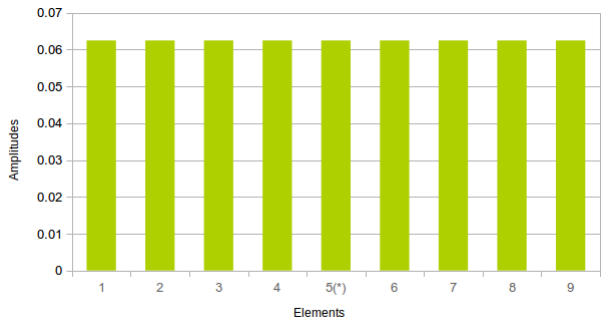


Figure 3.2: Equal superposition of all qubits, with 5(*) indicating that this is the element we look for.

3.3.2 Inverse amplitude

The next step of Grover's algorithm is using an oracle that will change the sign of the amplitude on the sought element. This is done by having the oracle add a -1 phase shift on the element that we search for. More formally, the oracle will output $|\psi\rangle$ for all $\psi \neq \psi^*$ and $-|\psi\rangle$ for $\psi = \psi^*$, where ψ^* is the target element. Therefore the amplitude of the sought element will be the same as the other qubits, except that it will be negative. The way to show this mathematically is with the following equation

$$\frac{1}{\sqrt{N}} \sum (-1)^{f(x)} |\psi\rangle \tag{3.10}$$

where $f(x) = 0$ if $x \neq \psi^*$ and $f(x) = 1$ if $x = \psi^*$. Figure 3.3 shows the amplitudes after this step.

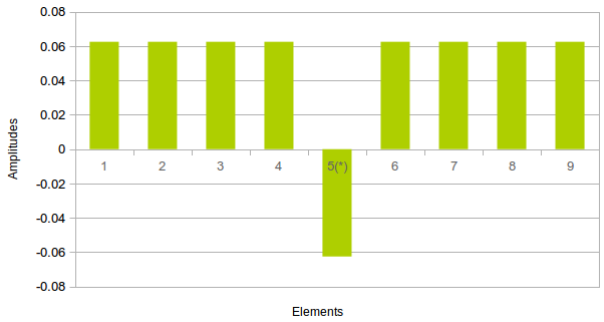


Figure 3.3: Step 2 of Grover's algorithm: Amplitude inversion. After inversing the amplitude of the sought element.

3.3.3 Inverse about mean

The last step to Grover's algorithm is to somehow make the amplitude of the sought element become larger than the amplitudes of any other element in the database. This procedure increases the probability that when measuring, we will get the element we are looking for. To achieve this, we need to change the amplitude of all qubits so that they are inverted about the mean. Grover called this step the *diffusion transform*. For every amplitude $\alpha \in \mathbb{C}$, we replace α with $2\mu - \alpha$, where μ is the mean. This will be the same as moving the amplitude as far above (or below) as it were below (or above) the mean before the application of this step. For the element with an already negative α, the amplitude will instead be $2\mu - (-\alpha) = 2\mu + \alpha$ which means that it will be larger than all the other amplitudes. The resulting chart over amplitudes can be seen in Figure 3.4.

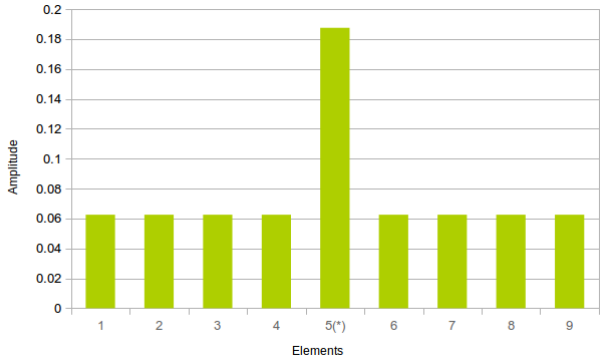


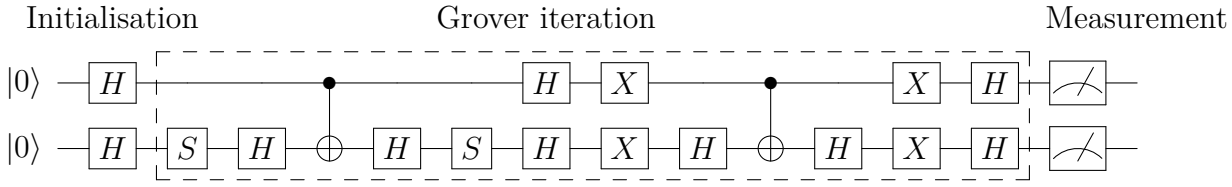
Figure 3.4: Grover's algorithm step 3: Inversion about the mean. Amplitudes after the inversion about the mean.

3.3.4 Number of iterations

As we could see in Figure 3.4, there is still a good chance that we won't get the value that we want since element 5 still only have probability 0.2. To solve this we need to repeat some steps of the algorithm a number of times to make the amplitude even higher, giving a higher probability that we measure the value we want. The steps that should be repeated are the amplitude inversion and the inverse about mean, also known as the Grover iteration. To make sure that the probability is high enough, i.e. close to 1, these steps need to be repeated $\sqrt{N}$ times in general. However, should we repeat these steps too many times, the probability would once again become lower and lower. Since we use the mean to inverse around, the problem arises when the amplitude of the target element becomes very large. If the amplitude is large, when we invert the amplitude of the target element, it will drag the mean below zero because of the large negative value. This means that when we try to invert about the mean, the probability will be smaller than it was before we applied this iteration of the algorithm.

3.3.4.1 Special Case - Single Shot Grover

There exists an interesting special case of Grover's search algorithm when there are two qubits ($n = 2$ and $N = 4$) and only one state that are marked, i.e. we are only looking for one state [18]. The states in this case are $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$. As an example, let us assume that we are looking for $|01\rangle$. It turns out that to find this state with certainty (100% probability), we only need to do the iteration one time [18]. The circuit for the above problem, i.e. searching for $|01\rangle$, looks like this:



The meter furthest to the right on the circuit indicates that the qubits should be measured. When we run this circuit and measure it, we will achieve $|01\rangle$ every time, which is exactly what we wanted.

4

Taxonomy of Quantum Algorithms

In this chapter we present the results of this thesis. The taxonomy contains 31 algorithms, which is about half of the algorithms available in the list at NIST.

4.1 Oracular or Non-oracular

We begin with a simple categorisation where the algorithms have been classified as either oracular or non-oracular. Some of the algorithms may have parts that are using an oracle and parts that are not, these are classified according to how the list at NIST classify them. The results of this classification can be seen in Table 4.1, where the algorithms in the left column are non-oracular and the algorithms in the right column are oracular.

Table 4.1: The first classification of the algorithms, where the left column contains non-oracular algorithms and the right column contains the oracular algorithm.

Non-oracular Algorithms	Oracular Algorithms
Shor's Algorithm[8]	Grover's Algorithm[9]
Principal Ideal[19]	Numerical Gradient Estimation[20]
Matrix Product Verification [21]	Hidden Shift Problem[22]
SAT-Solving[23]	Polynomial Interpolation[24]
Bernstein-Vazirani[25]	Ordered Search[26]
Formula Evaluation[27]	Matrix Commutativity[28]
Group Representations[29]	Curvelet Transform[30]
Viterbi Decoding[31]	Statistical Differences[32]
Hallgren's Algorithm[19]	Hash Collision and Claws[33]
Gauss sums[34]	Basin Hopping[35]
Subset Sum[36]	Secure MACs[37]
Abelian Hidden Subgroup[12]	Pattern Matching[38]
Deutsch-Jozsa[39]	Element Distincness[40]
Exponential Congruences[41]	Hidden Non-Linear Structures[42]
Decoding of Simplex Code[43]	Group Isomorphism
Linear Systems of Equations[44]	

4.2 Distribution of Core Ideas

Table 4.2: The classification regarding which core idea(s) each algorithm uses, such as Quantum Fourier Transform (QFT), Amplitude Amplification (AA), Quantum Walks (QW) and other if none of the above apply.

Algorithm	QFT	AA	QW	Other
Shor's Algorithm[8]	X			
Hallgren's Algorithm[19]	X			
Principal Ideal[19]	X			
Gauss sums[34]	X			
Matrix Product Verification [21]		X	X	
Subset Sum[36]		X	X	
SAT-Solving[23]		X		
Abelian Hidden Subgroup[12]	X			
Bernstein-Vazirani[25]				X
Deutsch-Jozsa[39]				X
Formula Evaluation[27]				X
Exponential Congruences[41]	X	X		
Group Representations[29]	X			
Decoding of Simplex Code[43]				X
Viterbi Decoding[31]		X		
Linear Systems of Equations[44]	X			
Grover's Algorithm[9]		X		
Hash Collision and Claws[33]		X		
Numerical Gradient Estimation[20]	X			
Basin Hopping[35]		X		
Hidden Shift Problem[22]	X			
Secure MACs[37]				X
Polynomial Interpolation[24]	X			
Pattern Matching[38]		X		
Ordered Search[26]				X
Element Distincness[40]			X	
Matrix Commutativity[28]		X	X	
Hidden Non-Linear Structures[42]	X		X	
Curvelet Transform[30]	X			
Group Isomorphism[45]	X			X
Statistical Differences[32]				X

After this categorisation, we will now look at different core ideas of the algorithm, such as if they use the Quantum Fourier Transform (QFT), amplitude amplification and/or quantum walks. In Table 4.2, we see what core idea each algorithm depends on. We can, for example, see that Grover's algorithm uses the amplitude amplification, as we saw in Chapter 3.3. There are also algorithms that use multiple different core ideas, such as the algorithm for the subset sum problem which uses both am-

plitude amplification and quantum walks in the solution, represented with multiple X's in their row. The distribution of the different core ideas can be seen in Figure 4.1.

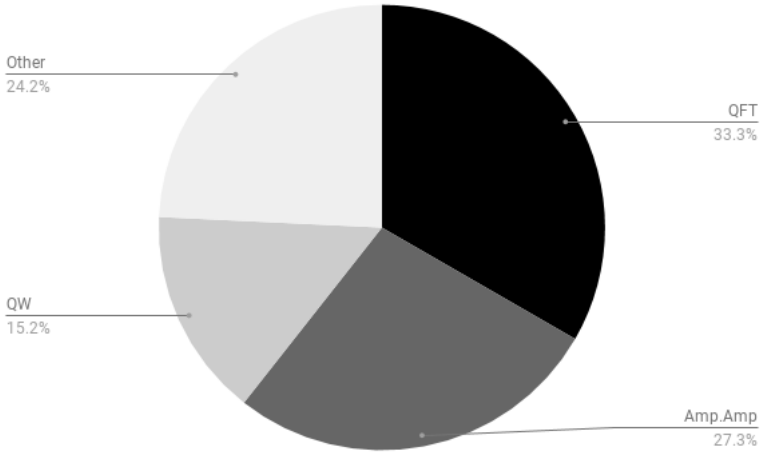


Figure 4.1: The distribution of core ideas used in the algorithms in this taxonomy.

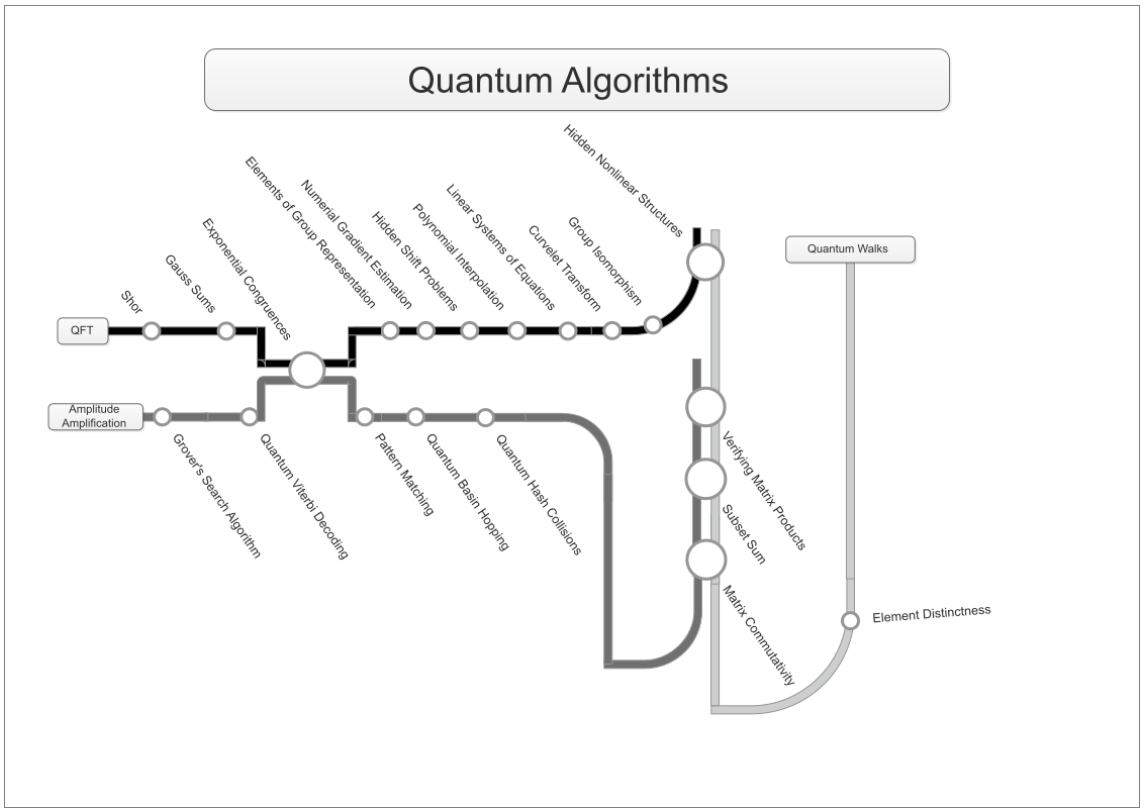


Figure 4.2: The taxonomy visualised as a subway map.

To visualise the amount of algorithms that use more than one of the core ideas, we have built a subway map, where each stop represents one algorithm, see Figure 4.2.

The black line represents QFT, the dark grey line represents amplitude amplification and the light grey line represents quantum walks. Wherever two lines share a station, it means that the algorithm at that station uses both of the core ideas.

4.3 Implications to Cryptographic Schemes

It is interesting to have a taxonomy of quantum algorithms per se, since it is such a new area of research and the algorithms use ideas that are not used to the same extent in classical computing. However, even more important is to see what these quantum algorithms can be used for. Shor [8] showed that his quantum algorithm can be used to efficiently solve problems that are assumed to be computationally hard on classical computers such as integer factorisation and the discrete logarithm problem. As a consequence, most public key cryptosystems that have been developed in the last century result insecure if an adversary has access to a quantum computer. In this final part of the taxonomy, we highlight what the quantum algorithms can be used for in terms of attacking cryptographic schemes and thus affecting their security, which can be seen in Table 4.3. This means that when quantum computers are available to governments or companies, they will be able to weaken the security of what we now deem as secure cryptography. Shor's algorithm can solve the discrete logarithm problem and integer factorisation which are used to make RSA and Diffie-Hellman cryptosystem secure. Hallgren [19], who solves the principal ideal problem as well as Pell's equation, can break a cryptosystem called Buchman-Williams, and it is also possible to reduce the principal ideal problem to Shor's algorithm which means that it can break the same things as Shor. With the use of Grover's algorithm, Brassard *et al.* [33] break hash-based cryptography polynomially faster than classical computers do. Dam *et al.* [22] show that it is possible to break pseudo-random generators and homomorphic encryption superpolynomially faster than classical computers by solving the hidden shift problem. By solving polynomial interpolation, Childs *et al.* [24] can break Shamir's secret sharing scheme faster than classical computers, but only by a constant factor. Grover [9] can search through the key space used for symmetric keys quadratically faster than classical searches, which in theory means that it can break such cryptosystems (e.g. AES). However, for now the easy fix would be to increase the key size to make it practically infeasible even with Grover's algorithm. To be secure with public key cryptography, we would need to move to post-quantum cryptography which relies on problem believed to be unfeasible even with a quantum computer, meaning that we do not know of a quantum algorithm that can break the problem. An example of such a cryptosystem is lattice-based cryptography.

Table 4.3: The quantum algorithms that break certain cryptosystems.

Quantum Algorithm	**Cryptosystem**
Shor's Algorithm [8]	RSA, Diffie-Hellman
Hallgren's Algorithm [19]	Buchman-Williams, through reduction anything Shor breaks
Hash Collision and Claws [33]	Hash-based cryptography
Hidden Shift Problem [22]	Pseudo-random generators and homomorphic encryption
Polynomial Interpolation [24]	Shamir's Secret Sharing Scheme
Grover's Algorithm [9]	AES

5

Conclusion

We are used to classical computers, but quantum computers are being developed and will become a reality in a fairly short time. Despite the limited access, researchers have developed a serie of quantum algorithms with interesting properties. In particular, some quantum algorithms are faster in solving problems than algorithms for classical computers. Therefore it is interesting and important at this stage in time to have a better idea of how quantum algorithms are structured and what the core ideas behind them are. This is what this thesis does, explaining at a high level how quantum computing works, giving an idea of what the core tools used by quantum algorithms are and finally create a taxonomy to show what quantum algorithms depend on what core ideas and what algorithms affect classical cryptography.

We have attempted to fill the gap within quantum computing where no previous taxonomy exists. This has been done by reading through papers describing algorithms and finding which core idea they rely upon. We have described the three core ideas that the taxonomy focuses on so that they should be understandable for a computer scientist without prior knowledge of quantum mechanics.

The 31 algorithms that we have looked into are quite uniformly distributed between the different core ideas, with quantum walks as the least used idea (15%), amplitude amplification in second place (27%) and the quantum fourier transform (33%) as the most used.

Since this is an area in constant development there can be new ideas in the future, and this taxonomy should be updated. However, in this time this taxonomy catches the most famous algorithm together with some of the less known algorithms and shows if they are similar or if they have different base ideas to solve their problem. In the future, the distribution of core ideas amongst the algorithms might change, there might be new core ideas but for now this taxonomy shows the distribution well.

There are likely many problems which have not been solved with quantum algorithms, which will be solved in the future. Thus, for future work, the main work to do is to keep filling the taxonomy with algorithms as they are developed. The "Other" piece of the pie chart in this taxonomy would be interesting to do further work on to see if all 24% of it are different ideas or if some of these algorithms share another core idea than what we have shown in this taxonomy. Also, the cryptography part of this taxonomy is interesting to follow up on, with more algorithms being

developed that possibly affect the security of cryptosystems that we use.

Bibliography

[1] U. Vazirani, "Lecture: Extended Church-Turing Thesis", pp. 2–5, 2007.

[2] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines", *Journal of statistical physics*, vol. 22, no. 5, pp. 563–591, 1980.

[3] D. Gil (IBM), *The future is quantum*, 2017. [Online]. Available: `https://www.ibm.com/blogs/research/2017/11/the-future-is-quantum/` (visited on 12/12/2017).

[4] W. Stallings, *Cryptography and network security principles and practice*. Pearson, 2017, pp. 92–95, ISBN: 9781292158587.

[5] W. Diffie, W. Diffie, and M. E. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976, ISSN: 15579654. DOI: `10.1109/TIT.1976.1055638`.

[6] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978, ISSN: 00010782. DOI: `10.1145/359340.359342`. arXiv: `arXiv:1011.1669v3`. [Online]. Available: `http://portal.acm.org/citation.cfm?doid=359340.359342`.

[7] F. Kohlar, G. Horst, and S. Sch, "On the Security of TLS-DH and TLS-RSA in the Standard Model", pp. 1–50, 2013.

[8] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", pp. 124–134, 1995, ISSN: 0097-5397. DOI: `10.1137/S0097539795293172`. arXiv: `9508027 [quant-ph]`. [Online]. Available: `http://arxiv.org/abs/quant-ph/9508027%20http://dx.doi.org/10.1137/S0097539795293172`.

[9] L. K. Grover, "A fast quantum mechanical algorithm for database search", *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*, pp. 212–219, 1996, ISSN: 07349025. DOI: `10.1145/237814.237866`. arXiv: `9605043 [quant-ph]`. [Online]. Available: `http://portal.acm.org/citation.cfm?doid=237814.237866`.

[10] E. Strubell, "An Introduction To Quantum Algorithms", p. 35, 2011. [Online]. Available: `http://umcs.maine.edu/%7B~%7Dema/files/quantum%7B%5C_%7Dtutorial.pdf`.

[11] Wikimedia, *https://upload.wikimedia.org/wikipedia/commons/6/6b/Bloch_sphere.svg*, 2018.

[12] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. 2010, p. 702, ISBN: 1107002176. DOI: 10.1017/CBO9780511976667. arXiv: arXiv:1011.1669v3.

[13] A. Barenco, C. H. Bennett, R. Cleve, D. P. Divincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation", *Physical Review A*, vol. 52, no. 5, pp. 3457–3467, 1995, ISSN: 10502947. DOI: 10.1103/PhysRevA.52.3457. arXiv: 9503016 [quant-ph].

[14] X. M. Jin, J. G. Ren, B. Yang, Z. H. Yi, F. Zhou, X. F. Xu, S. K. Wang, D. Yang, Y. F. Hu, S. Jiang, T. Yang, H. Yin, K. Chen, C. Z. Peng, and J. W. Pan, "Experimental free-space quantum teleportation", *Nature Photonics*, vol. 4, no. 6, pp. 376–381, 2010, ISSN: 17494885. DOI: 10.1038/nphoton.2010.87. [Online]. Available: http://dx.doi.org/10.1038/nphoton.2010.87.

[15] Y. S. Weinstein, M. A. Pravia, E. M. Fortunato, S. Lloyd, and D. G. Cory, "Implementation of the quantum Fourier transform", *Physical Review Letters*, vol. 86, no. 9, pp. 1889–1891, 2001, ISSN: 00319007. DOI: 10.1103/PhysRevLett.86.1889. arXiv: 9906059 [quant-ph].

[16] N. Z. Y. Aharonov, L. Davidovich, "Quantum random walks", *Physical Review A*, vol. 48, no. 2, pp. 1687–1690, 1993, ISSN: 1050-2947. DOI: 10.1103/PhysRevA.48.1687. arXiv: 1207.4535v1.

[17] J. Kempe, "Quantum random walks: An introductory overview", *Contemporary Physics*, vol. 44, no. 4, pp. 307–327, 2003, ISSN: 00107514. DOI: 10.1080/0010715103100011077 6. arXiv: 0303081v1 [quant-ph].

[18] J. Vicary, "The Topology of Quantum Algorithms", 2012, ISSN: 1043-6871. DOI: 10.1109/LICS.2013.14. arXiv: 1209.3917. [Online]. Available: http://arxiv.org/abs/1209.3917%20http://dx.doi.org/10.1109/LICS.2013.14.

[19] S. Hallgren, "Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem", *Journal of the ACM*, vol. 54, no. 1, 1–es, 2007, ISSN: 00045411. DOI: 10.1145/1206035.1206039. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1206035.1206039.

[20] S. P. Jordan, "Fast quantum algorithm for numerical gradient estimation", *Physical Review Letters*, vol. 95, no. 5, pp. 1–4, 2005, ISSN: 00319007. DOI: 10.1103/PhysRevLett.95.050501. arXiv: 0405146 [quant-ph].

[21] H. Buhrman and R. Spalek, "Quantum Verification of Matrix Products", p. 15, 2004. DOI: 10.1145/1109557.1109654. arXiv: 0409035 [quant-ph]. [Online]. Available: http://arxiv.org/abs/quant-ph/0409035.

[22] W. van Dam, S. Hallgren, and L. Ip, "Quantum Algorithms for Some Hidden Shift Problems", *SIAM Journal on Computing*, vol. 36, no. 3, pp. 763–778, 2006, ISSN: 0097-5397. DOI: 10.1137/S009753970343141X. arXiv: 0211140 [quant-ph]. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/S009753970343141X.

[23] A. Ambainis, "Quantum search algorithms", pp. 1–12, 2005. arXiv: 0504012 [quant-ph]. [Online]. Available: http://arxiv.org/abs/quant-ph/0504012.

[24] A. M. Childs, W. van Dam, S.-H. Hung, and I. E. Shparlinski, "Optimal quantum algorithm for polynomial interpolation", *arXiv:1509.09271*, pp. 1–

16, 2015, ISSN: 18688969. DOI: 10.4230/LIPIcs.ICALP.2016.16. arXiv: 1509.09271v1. [Online]. Available: http://arxiv.org/abs/1509.09271.

[25] E. Bernstein and U. Vazirani, "Quantum Complexity Theory", *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1411–1473, 1997, ISSN: 0097-5397. DOI: 10.1137/S0097539796300921. [Online]. Available: http://epubs.siam.org/doi/10.1137/S0097539796300921.

[26] A. M. Childs, A. J. Landahl, and P. A. Parrilo, "Quantum algorithms for the ordered search problem via semidefinite programming", *Physical Review A - Atomic, Molecular, and Optical Physics*, vol. 75, no. 3, pp. 1–8, 2007, ISSN: 10502947. DOI: 10.1103/PhysRevA.75.032335. arXiv: 0608161v1 [arXiv:quant-ph].

[27] B. W. Reichardt, "Reflections for quantum query algorithms", no. 1, pp. 1–13, 2010. DOI: 10.1137/1.9781611973082.44. arXiv: 1005.1601. [Online]. Available: http://arxiv.org/abs/1005.1601.

[28] Y. Itakura, "Quantum algorithm for commutativity testing of a matrix set", *arXiv preprint quant-ph/0509206*, 2005. arXiv: 0509206v1 [arXiv:quant-ph]. [Online]. Available: https://arxiv.org/abs/quant-ph/0509206.

[29] S. P. Jordan, "Fast quantum algorithms for approximating some irreducible representations of groups", 2008. arXiv: 0811.0562. [Online]. Available: http://arxiv.org/abs/0811.0562.

[30] Y.-K. Liu, "Quantum Algorithms Using the Curvelet Transform", 2008. arXiv: 0810.4968. [Online]. Available: http://arxiv.org/abs/0810.4968.

[31] J. R. Grice and D. A. Meyer, "A quantum algorithm for Viterbi decoding of classical convolutional codes", *Quantum Information Processing*, vol. 14, no. 7, pp. 2307–2321, 2015, ISSN: 15700755. DOI: 10.1007/s11128-015-1003-3. arXiv: 1405.7479.

[32] S. Bravyi, A. W. Harrow, and A. Hassidim, "Quantum algorithms for testing properties of distributions", vol. 10598, pp. 1–21, 2009. arXiv: arXiv:0907.3920v1.

[33] G. Brassard, P. Hcyer, and A. Tapp, "Quantum Cryptanalysis of Hash and Claw-Free Functions*", vol. 20244, pp. 14–19, 1997.

[34] W. van Dam and G. Seroussi, "Efficient Quantum Algorithms for Estimating Gauss Sums", p. 11, 2002. arXiv: 0207131 [quant-ph]. [Online]. Available: http://arxiv.org/abs/quant-ph/0207131.

[35] D. Bulger, "Quantum basin hopping with gradient-based local optimisation", *Quant-Ph/0507193*, 2005. arXiv: 0507193v1 [arXiv:quant-ph]. [Online]. Available: http://arxiv.org/abs/quant-ph/0507193$%5Cbackslash$nhttp://www.arxiv.org/pdf/quant-ph/0507193.pdf.

[36] D. J. Bernstein, S. Jeffery, T. Lange, and A. Meurer, "Quantum algorithms for the subset-sum problem", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7932 LNCS, pp. 16–33, 2013, ISSN: 03029743. DOI: 10.1007/978-3-642-38616-9_2.

[37] D. Boneh and M. Zhandry, "Quantum-secure message authentication codes", *Lecture Notes in Computer Science (including subseries Lecture Notes in Ar-*

tificial Intelligence and Lecture Notes in Bioinformatics), vol. 7881 LNCS, pp. 592–608, 2013, ISSN: 03029743. DOI: 10.1007/978-3-642-38348-9_35.

[38] A. Montanaro, "Quantum Pattern Matching Fast on Average", *Algorithmica*, vol. 77, no. 1, pp. 16–39, 2017, ISSN: 14320541. DOI: 10.1007/s00453-015-0060-4. arXiv: 1408.1816.

[39] D. Deutsch and R. Jozsa, "Rapid Solution of Problems by Quantum Computation", *Proceedings: Mathematical and Physical Sciences*, vol. 439, no. 1907, pp. 553–558, 1992, ISSN: 09628444. [Online]. Available: http://www.jstor.org/stable/52182.

[40] A. Ambainis, "Quantum walk algorithm for element distinctness", vol. 0354, pp. 1–33, 2003, ISSN: 0272-5428. DOI: 10.1109/FOCS.2004.54. arXiv: 0311001 [quant-ph]. [Online]. Available: http://arxiv.org/abs/quant-ph/0311001.

[41] W. van Dam and I. E. Shparlinski, "Classical and Quantum Algorithms for Exponential Congruences", no. 1, 2008. arXiv: 0804.1109. [Online]. Available: http://arxiv.org/abs/0804.1109.

[42] A. M. Childs, L. J. Schulman, and U. V. Vazirani, "Quantum algorithms for hidden nonlinear structures", *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pp. 395–404, 2007, ISSN: 02725428. DOI: 10.1109/FOCS.2007.4389510. arXiv: 0705.2784.

[43] A. Barg and S. Zhou, "A quantum decoding algorithm of the simplex code", pp. 1–8,

[44] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations", 2009. arXiv: arXiv:0811.3171v3.

[45] F. Le Gall, "An Efficient Quantum Algorithm for some Instances of the Group Isomorphism Problem", *Stacs'10*, vol. 2010, p. 20, 2010, ISSN: 18688969. DOI: 10.4230/LIPIcs.STACS.2010.2484. arXiv: 1001.0608. [Online]. Available: http://arxiv.org/abs/1001.0608.