SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Quantum Computing: From Shor's Algorithm to The Hidden Subgroup Problem

av

Emilia Dunfelt

2020 - No K36

Quantum Computing: From Shor's Algorithm to The Hidden Subgroup Problem

Emilia Dunfelt

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Sven Raum

2020

Abstract

The following paper aims to present the basic theory of quantum computing, Shor's algorithm, the graph isomorphism problem, and their connection to the hidden subgroup problem. A mathematical approach is taken to these concepts, in contrast to a physical or computer scientific focus. Each topic is introduced with an emphasis on Hilbert spaces, analytic number theory, representation theory, and group theory. First, Shor's factoring algorithm is presented and summarized in detail, giving a complete overview of the factoring process. Then, after introducing basic notions of representation theory of finite groups, the standard method of quantum algorithms is presented in the context of the hidden subgroup framework. Finally, the graph isomorphism problem is discussed, and a brief overview of the current status of this problem is given.

Acknowledgements

I would like to thank my supervisor Sven Raum, for introducing me to this fascinating and compelling topic, and for the many enlightening conversations along the way helping me navigate the intersection of mathematics, computer science, and physics. I also want to express my sincere gratitude to my family for all the support throughout the process of writing this project.

Table of contents

SECTION 1
INTRODUCTION

A major area of research today is the development of algorithms that can be efficiently implemented using methods of quantum computing. There is, for good reason, an expectation that these algorithms may offer exponential speed-up compared to their classical counterparts. In 1994, Peter Shor presented a probabilistic quantum algorithm providing this sought after speed-up to the factorization problem. This result had a significant impact on the area, and the interest both in solving the technical difficulties introduced by quantum mechanics and in how the underlying mathematics could be used to develop new algorithms utilizing similar techniques increased over the following years.

Just last year, Google claimed [AAB+19] in an article published in *Nature*, to have achieved so-called "quantum supremacy" in solving a problem on a 53-bit quantum computer in a mere 2.5 days, as opposed to their estimation of 10,000 years to solve the same problem on a (classical) supercomputer. The initial claim was disputed by IBM [PGMG19], the other major player at the forefront of the research in the field. Nonetheless, this shows that the quick developments and active research of recent years are only the beginning of a new area of computing which promises to advance the limits of computation as we know it today.

At the core of this area is a deep mathematical framework known as the hidden subgroup problem, which, in combination with the Fourier transform gives the very foundation of quantum algorithms such as Shor's algorithm. In its most general form, this framework can be applied to problems that are not yet solved, such as the graph isomorphism problem. Consequently, further progress in this field of research promises not only significant advances in computer science and physics but also in mathematics.

The main purpose of this text is to present the basics of quantum computing along with two of the most significant problems that are known to be, or suspected to be, efficiently solvable using quantum techniques: the factoring problem and the graph isomorphism problem. We focus on the mathematical aspects of this theory, with an emphasis on how the underlying mathematics provides the greater structure of quantum computing. This includes the theory of Hilbert spaces, representation theory, group theory, and Fourier transforms.

The preliminaries in Section 2 covers the theory of Hilbert spaces, time complexity, and continued fractions. Section 3 presents the basics of quantum computing and the discrete Fourier transform, these are tools that will be used heavily throughout the text. In Section 4 Shor's factoring algorithm is presented in its entirety, and an example of how it can be used to factor an integer is given. This section is followed by a presentation of the hidden subgroup problem, its relation to quantum computing, and the so-called standard method for different groups. Finally, in Section 6, the graph isomorphism problem is presented, as well as theory regarding the representations of the symmetric group S_n.

SECTION 2
PRELIMINARIES

§ 2.1. Hilbert Spaces. In what follows, we assume the reader to be familiar with the basics of linear algebra: vector spaces, inner products and matrix operations. We here present some extended definitions and theorems necessary to understand later sections. Proofs and detailed explanations can be found in [Hal87].

Definition 2.1. A *Hilbert space* is a real or complex complete inner product space.

When considering a finite dimensional inner product space, which will be the case throughout this text, the completeness criterion follows from the completeness of $\mathbb{R}^n$ and uniqueness of norms.

Definition 2.2. A *linear transformation* on a vector space $\mathcal{V}$, is a function $T : \mathcal{V} \longrightarrow \mathcal{V}$ such that for all $v_1, v_2 \in \mathcal{V}$, and scalars α, β, the following equality is satisfied:

$$T(\alpha v_1 + \beta v_2) = \alpha T(v_1) + \beta T(v_2).$$

Choosing a basis for the n-dimensional vector space $\mathbb{C}^n$, we identify with each linear transformation from $\mathbb{C}^n$ to itself a matrix in $M_n(\mathbb{C})$, with respect to the chosen basis. A certain subset of linear transformations will be exceptionally useful in studying quantum gates.

Definition 2.3. A linear transformation U between Hilbert spaces is called *unitary* if it is a bijection preserving the norm, or equivalently, if

$$U^* = U^{-1}.$$

We want to be able to create the product of two vector spaces, thus forming a larger vector space, similar to forming the direct sum. We will denote this operation, called the *tensor product*, by $\otimes$. Intuitively, we would want the tensor product $\mathcal{U} \otimes \mathcal{V}$ of two spaces to contain elements $u \otimes v$, that are products of elements $u \in \mathcal{U}$ and $v \in \mathcal{V}$ in some sense. Most naturally, the dimension of the product space should be equal to $\dim(\mathcal{U}) \cdot \dim(\mathcal{V})$, and we also want this product to satisfy some linearity properties that would be reasonable to expect.

In order to give a useful, formal definition of such a product, we make the following definitions.

Definition 2.4. A scalar valued function y on the vector space $\mathcal{V}$ is called a *linear functional* if, for every $v_1, v_2 \in \mathcal{V}$ and scalars α and β, it is true that

$$y(\alpha v_1 + \beta v_2) = \alpha y(v_1) + \beta y(v_2).$$

The set of all linear functionals form a vector space called the *dual space* of $\mathcal{V}$, and is denoted by $\mathcal{V}^*$.

The dimension of the dual space is in fact equal to the dimension of the vector space, a consequence of the following theorem:

Theorem 2.5. *Let $\mathcal{V}$ be a vector space with basis $\{v_1, \ldots, v_n\}$, then there exists a unique basis $\{w_1, \ldots, w_n\}$ in $\mathcal{V}^*$ such that $w_j(v_i) = \delta_{ij}$. This is called the dual basis.*

Finally, we consider the equivalence of linear functionals on the direct product of two vector spaces. This theory can also easily be extended to a product $\mathcal{V}_1 \times \ldots \times \mathcal{V}_n$, of n finite dimensional vector spaces. Note that this is not the same as the tensor product that we have yet to define.

Definition 2.6. Let $\mathcal{U}, \mathcal{V}$ be vector spaces over the same field and form their direct product $\mathcal{W} = \mathcal{U} \times \mathcal{V}$, which is also a vector space. A scalar valued function w on $\mathcal{W}$ is called a *bilinear form* or *bilinear functional*, if

$$w(\alpha u_1 + \beta u_2, v) = \alpha w(u_1, v) + \beta w(u_2, v),$$

and

$$w(u, \alpha v_1 + \beta v_2) = \alpha w(u, v_1) + \beta w(u, v_2),$$

for all $u, u_1, u_2 \in \mathcal{U}$, $v, v_1, v_2 \in \mathcal{V}$ and scalars α, β.

We are now ready to give the definition of the tensor product of vector spaces. Note that there are many possible ways to define the tensor product, some more complicated than others. For our purposes of working with finite-dimensional vector spaces we stick with the definition of [Hal87].

Definition 2.7. The *tensor product* $\mathcal{U} \otimes \mathcal{V}$ of two finite-dimensional vector spaces over the same field is a space of linear functionals on the space of all bilinear forms on $\mathcal{U} \times \mathcal{V}$, such that for any $u \in \mathcal{U}$ and $v \in \mathcal{V}$ the tensor product $u \otimes v$ is defined by

$$w \longmapsto w(u, v),$$

for every bilinear form w.

For the tensor space $\mathcal{U} \otimes \mathcal{V}$ we have the following theorem, which shows that our hopes that the dimension be multiplicative is satisfied.

Theorem 2.8. *If $\{x_1, x_2, \ldots, x_n\}$ and $\{y_1, y_2, \ldots, y_m\}$ are bases of $\mathcal{U}$ and $\mathcal{V}$, respectively, then the set $\{x_i \otimes y_j \mid i = 1, \ldots, n;\ j = 1, \ldots, m\}$ is a basis of $\mathcal{U} \otimes \mathcal{V}$.*

We also consider the linear transformations on the space $\mathcal{U} \otimes \mathcal{V}$. So let $A : \mathcal{U} \longrightarrow \mathcal{U}'$ and $B : \mathcal{V} \longrightarrow \mathcal{V}'$ be two transformations. Then $C = A \otimes B$ is the linear trasformation from $\mathcal{U} \otimes \mathcal{V}$ to $\mathcal{U}' \otimes \mathcal{V}'$, with the adjoint C^*. By definition of the tensor product we then have that

$$\begin{aligned}
[C(u \otimes v)](w) &= (u \otimes v)(C^* w) \\
&= (C^* w)(u, v) \\
&= w(Au, Bv) \\
&= (Au \otimes Bv)(w),
\end{aligned}$$

for $u \in \mathcal{U}$ and $v \in \mathcal{V}$.

Example 2.9. To perhaps make these somewhat abstract definitions more concrete, we consider what the matrix $C = A \otimes B$ would look like in terms of A and B. So let these transformations be defined by matrices $A = (a_{ij})$ and $B = (b_{pq})$ and fix an ordering of the basis of $\mathcal{U} \otimes \mathcal{V}$, defined in Theorem 2.8, called the *lexicographical order*:

$$x_1 \otimes y_1, x_1 \otimes y_2, \ldots, x_1 \otimes y_m, x_2 \otimes y_1, \ldots, x_2 \otimes y_m, \ldots, x_n \otimes y_1, \ldots, x_n \otimes y_m.$$

This provides an identification between the linear transformations from $\mathcal{U} \otimes \mathcal{V} \to \mathcal{U} \otimes \mathcal{V}$ to $M_{mn}(\mathbb{C})$. Then,

$$C(x_j \otimes y_q) = A(x_j) \otimes B(y_q) = \left(\sum_i a_{ij} x_i\right) \otimes \left(\sum_p b_{pq} y_p\right) = \sum_{i,p} a_{ij} b_{pq} (x_i \otimes y_p).$$

We thus get the following matrix form of C, known as the *Kronecker product*:

$$C = \begin{pmatrix} a_{11}B & \ldots & a_{1n}B \\ a_{21}B & \ldots & a_{2n}B \\ \vdots & \vdots & \vdots \\ a_{n1}B & \ldots & a_{nn}B \end{pmatrix}.$$

The final concept necessary to continue is that of projections, and in particular orthogonal projections on Hilbert spaces.

Definition 2.10. Let $\mathcal{V} = M \times N$ be a vector space such that every $v \in \mathcal{V}$ can be written uniquely as $v = x + y$, where $x \in M$ and $y \in N$. Then the *projection* on M along N is the transformation P defined by $Pv = x$.

Given a subspace M of a finite dimensional inner product space $\mathcal{V}$, the Projection Theorem [Hal87] Chapter 66, states that $\mathcal{V}$ can be decomposed into the direct product of M and $M^{\perp}$. So every $v \in \mathcal{V}$ can be written uniquely as $v = x + y$, where $x \in M$ and $y \in M^{\perp}$. In particular this is true for Hilbert spaces.

It can thus be shown that projections on Hilbert spaces are both *idempotent*, meaning that $P = P^2$, but also self adjoint, meaning that $P = P^*$. With these concuding remarks, we state the following definition.

Definition 2.11. A linear operator $P : \mathcal{V} \to \mathcal{V}$ on a Hilbert space $\mathcal{V}$ is called an *orthogonal projection* if $P = P^2 = P^*$.

§ 2.2. Asymptotic Notation.

We briefly discuss the basic notions of algorithmic complexity and order notation, for a more in-depth discussion, the reader can consult Chapter 3 of [CLRS09]. When discussing the efficiency of algorithms it is fundamental to be able to compare the amount of resources needed to run the computation. Resources here usually refers to time or memory. We aim to describe the amount of resources needed in terms of the input size, often as a function $T(n)$, if the input is of size n. However, determining this function is often quite difficult, which is why we usually consider the *asymptotic complexity* instead. We make the following definition:

Definition 2.12 (Order Notation)**.** Let f and g be positive functions of n. If there exists constants c and N such that

$$f(n) \leq cg(n),$$

for all $n \geq N$, we say that f is *big-$\mathcal{O}$ of g*, and write $f(n) = \mathcal{O}(g(n))$.

We often analyze algorithms from a *worst-case* standpoint, assuming that it takes as long as possible to complete, and thus derive the upper bound $g(n)$.

Depending on what kind of function g is, we can compare the complexity of different algorithms. We here list, from best to worst, the most common functions to consider when analyzing time and space complexities:

- $g(n) = 1$: constant time
- $g(n) = n$: linear time
- $g(n) = n^k$: polynomial time
- $g(n) = \mathcal{O}(e^{\varepsilon n})$, for all $\varepsilon > 0$: sub-exponential time
- $g(n) = e^{kn}$: exponential time

§ **2.3. Continued Fractions.** Perhaps surprisingly, the theory of continued fractions plays a significant role in the post processing of Shor's algorithm which we will see in the next section. In this section, we give a brief overview of relevant results in this area, following the presentation given in [HW08].

Definition 2.13. The expression

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\dots + \cfrac{1}{a_n}}}},$$

where $n \geq 0$ and a_i is a positive integer for each i, is called a *finite continued fraction*. This expression is also denoted by $[a_0, a_1, a_2, \dots, a_n]$.

Definition 2.14. The expression $[a_0, a_1, \dots, a_i]$, $0 \leq i \leq n$, is called the ith *convergent* to the continued fraction $[a_0, a_1, \dots, a_n]$.

Throughout this section, we reserve the $[a]$-notation for denoting the continued fraction expansion with just one term, not to be confused with the notation used in other sections where this represents the integer part of a.

The convergents can be calculated using the following recursive formula.

Theorem 2.15. *Let $[a_0, a_1, \dots, a_n]$ be a finite continued fraction. If p_i and q_i are defined recursively by*

$$\begin{aligned} &p_0 = a_0, \quad p_1 = a_1 a_0 + 1, \quad p_i = a_i p_{i-1} + p_{i-2} \qquad (2 \leq i \leq n), \\ &q_0 = 1, \quad q_1 = a_1, \quad q_i = a_i q_{i-1} + q_{i-2} \qquad (2 \leq i \leq n). \end{aligned}$$

then

$$[a_1, a_1, \dots, a_i] = \frac{p_i}{q_i}$$

is the ith convergent.

Theorem 2.16. *Every positive rational number can be uniquely represented by a finite continued fraction.*

Now, to actually determine this continued fraction expansion of a given rational number $x \in \mathbb{Q}$, we use a process similar to the familiar Euclidean algorithm. We now demonstrate this process and then analyze its time complexity which will be important later on.

Taking $x \in \mathbb{Q}$, let $a_0 = [x]$, so that $x = a_0 + \varepsilon_0$, where $0 \leq \varepsilon_0 < 1$. Now, if $\varepsilon_0 \neq 0$ we write

$$\frac{1}{\varepsilon_0} = a_1', \quad [a_1'] = a_1, \quad a_1' = a_1 + \varepsilon_1,$$

again where $0 \leq \varepsilon_1 < 1$. Similarly, if $\varepsilon_1 \neq 0$, we have that

$$\frac{1}{\varepsilon_1} = a_2' = [a_2'] + \varepsilon_2 = a_2 + \varepsilon_2,$$

where $0 \leq \varepsilon_2 < 1$. We continue this process, in each step obtaining another $a_i' = 1/\varepsilon_{i-1} > 1$ and corresponding positive integer a_i, for all $i \geq 1$. Eventually, a point is reached when $\varepsilon_i = 0$, say for $i = n$, and a system of equations is obtained:

$$\begin{aligned} x &= a_0 + \varepsilon_0 \\ \frac{1}{\varepsilon_0} &= a_1 + \varepsilon_1 \\ \frac{1}{\varepsilon_1} &= a_2 + \varepsilon_2 \\ &\vdots \\ \frac{1}{\varepsilon_{n-1}} &= a_n + \varepsilon_n, \end{aligned}$$

where $0 \leq \varepsilon_i < 1$, for each $0 \leq i < n$. It follows that $x = [a_0, a_1, \ldots, a_n]$ is the continued fraction expansion of x.

As promised, we now demonstrate how this process is actually the Euclidean algorithm in disguise. Since x is rational, write $x = \frac{p}{q}$, where $p, q \in \mathbb{Z}$, $q > 1$, and we have the initial relation $p = a_0 q + \varepsilon_0 q$. Denote the term $\varepsilon_0 q$ by q_1, this is the remainder of the division p/q. So we can write

$$a_1' = \frac{q}{q_1} = a_1 + \varepsilon_1,$$

from which we get the relation $q = a_1 q_1 + \varepsilon_1 q_1$, where $q_2 = \varepsilon_1 q_1$ is the remainder of q divided by q_1. Continuing in the same fashion, we obtain a strictly decreasing sequence of integers, $q, q_1, q_2, \ldots, q_n$. This yields the following system of equations:

$$\begin{aligned} p &= a_0 q + q_1, \quad 0 < q_1 < q \\ q &= a_1 q_1 + q_2, \quad 0 < q_2 < q_1 \\ &\vdots \\ q_{n-2} &= a_{n-1} q_{n-1} + q_n, \quad 0 < q_n < q_{n-1} \\ q_{n-1} &= a_n q_n, \end{aligned}$$

which is immediately recognized as the Euclidean algorithm, calculating the greatest common divisor of p and q. This also implies that the complexity of the continued fraction algorithm is the same as the Euclidean algorithm.

As mentioned previously, the q_i's are strictly decreasing, and in fact, it is true that $q_{i+2} < \frac{1}{2}q_i$, meaning that they are halved after two iterations.

This is certainly true if $q_{i+1} \leq \frac{1}{2}q_i$, and if $q_{i+1} > \frac{1}{2}q_i$ we have that

$$q_{i+2} = q_i - q_{i+1} < q_i - \frac{1}{2}q_i = \frac{1}{2}q_i.$$

Now if $2^{k-1} < p < 2^k$ the algorithm terminates after at most $2k$ iterations and thus $\mathcal{O}(\log p)$ division steps are needed to find $\gcd(p, q)$ or the continued fraction expansion of $\frac{p}{q}$, where $p > q$.

The final result that we will need is the following theorem, an elementary result of number theory. A proof can be found in [HW08], Theorem 184.

Theorem 2.17. *Let $x \in \mathbb{Q}$, and $p, q \in \mathbb{Z}$, $q \neq 0$. If*

$$\left| x - \frac{p}{q} \right| < \frac{1}{2q^2},$$

then p/q is a convergent of x.

This theorem will be used later, along with the continued fraction algorithm, to determine the convergent p/q.

SECTION 3
Quantum Computing

§ 3.1. Foundations of Quantum Computing. Quantum computing differs quite significantly from classical computing. In this section we introduce basic notions in this area, and go over the three phases of quantum computation: the input, the computation and the output. Similar to the classical case, quantum computations are operations performed on registers consisting of bits, but these computations, registers and bits are very different from their classical counterparts.

Definition 3.1. A unit vector in a Hilbert space is called a *quantum state*. A state in $\mathbb{C}^2$ is called a qubit, and we distinguish the vectors

$$e_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad e_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

called the *fundamental states*.

Here we instantly recognize the difference to the definition of classical bits. While the state of a classical bit is either 0 or 1, a qubit is any unit vector in $\mathbb{C}^2$. Furthermore, the state of a qubit is described by two complex numbers and thus contains far more information than the state of a classical bit.

Theorem 3.2. *The vector $v = \alpha e_0 + \beta e_1$ in $\mathbb{C}^2$ is a qubit for all α, β such that $|\alpha|^2 + |\beta|^2 = 1$.*

Proof. This is true by the definition of states, since $||v|| = \sqrt{|\alpha|^2 + |\beta|^2} = 1$, so v is indeed a unit vector. □

It is worth noting that a qubit $v = \alpha e_0 + \beta e_1$ is said to be in a *superposition of states* if both α and β are nonzero.

As the reader is probably familiar with, an n-bit register in classical computing corresponds to a string of n bits, in quantum computing, however, we define an n-bit register as follows.

Definition 3.3. An n-qubit *composite system*, or an *n-qubit register*, is the Hilbert tensor product of the n individual qubit systems, $(\mathbb{C}^2)^{\otimes n}$.

By Theorem 2.8, for any Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$, with bases $\{x_1, x_2, \dots, x_n\}$ and $\{y_1, y_2, \dots, y_m\}$, we have that $\mathcal{H}_1 \otimes \mathcal{H}_2$ is a Hilbert space with basis $\{x_i \otimes y_j \mid i = 1, \dots, n; j = 1, \dots, m\}$.

We can define a basis of fundamental states for this composite system as the tensor product of the fundamental states of each subsystem: $\{e_{i_1} \otimes e_{i_2} \otimes \dots \otimes e_{i_n}\}$, where $(e_{i_1}, \dots, e_{i_n}) \in (\mathbb{Z}/2\mathbb{Z})^n$, a binary digit of length n. Thus we may write the basis elements as $\{e_0, e_1, e_2, \dots, e_{2^n-1}\}$, and they can be interpreted as binary sequences. Observe that this might lead to ambiguity if we fail to pay attention to which Hilbert space a product belongs to, for example $e_0 \otimes e_0 \in (\mathbb{C}^2)^{\otimes 2}$ and $e_0 \in \mathbb{C}^2$.

Theorem 3.4. *A vector $v = \sum_{i=0}^{2^n-1} \alpha_i e_i$ is a state in a composite system $(\mathbb{C}^2)^{\otimes n}$ if*

$$\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1.$$

Proof. Again, it is clear that v is a unit vector in $(\mathbb{C}^2)^{\otimes n}$ since $||v|| = \sqrt{\sum |\alpha_i|^2} = 1$. $\square$

Now we move on to discuss how quantum computations are performed on an input consisting of a quantum state. First of all, we want a computation to transform one state into another, so the norm has to be preserved. These are then the unitary transformations from a Hilbert space $(\mathbb{C}^2)^{\otimes n}$ to itself, in quantum computing commonly referred to as *quantum gates*. Since unitary transformations are invertible, this implies that quantum computations are reversible, in contrast to classical computations. We now provide the reader with a few examples of simple quantum gates.

Example 3.5. (Hadamard Transform) The *Hadamard transform*, $H_1 : \mathbb{C}^2 \to \mathbb{C}^2$, on a single-qubit system is defined by the following transformation

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

We see that it transforms the two fundamental states in the following way

$$H_1 e_0 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}(e_0 + e_1) \quad \text{and} \quad H_1 e_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}}(e_0 - e_1).$$

More generally, the Hadamard transform on a n-qubit register is defined recursively as the n-fold tensor product of H_1:

$$H_n = H_{n-1} \otimes H_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix}.$$

In what follows we use superscripts to denote the Hilbert space to which a state belongs. Now consider the following intermediary result:

Lemma 3.6. *It is true that*

$$(e_0^1 + e_1^1)^{\otimes n} = \sum_{i=0}^{2^n - 1} e_i^n.$$

Proof. First note that

$$\begin{aligned}(e_0^1 + e_1^1)^{\otimes 2} &= (e_0^1 + e_1^1) \otimes (e_0^1 + e_1^1) \\ &= e_0^1 \otimes e_0^1 + e_0^1 \otimes e_1^1 + e_1^1 \otimes e_0^1 + e_1^1 \otimes e_1^1 \\ &= \sum_{i=0}^{2^2-1} e_i^2.\end{aligned}$$

This is our base case for induction. Now assume that $(e_0^1 + e_1^1)^{\otimes k} = \sum_{i=0}^{2^k-1} e_i^k$. Then,

$$\begin{aligned}(e_0^1 + e_1^1)^{\otimes k+1} &= (e_0^1 + e_1^1) \otimes (e_0^1 + e_1^1)^{\otimes k} \\ &= (e_0^1 + e_1^1) \otimes \sum_{i=0}^{2^k-1} e_i^k \\ &= \sum_{i=0}^{2^k-1} e_0^1 \otimes e_i^k + \sum_{i=0}^{2^k-1} e_1^1 \otimes e_i^k,\end{aligned}$$

by assumption. Note that in the final sum we append a 0 to the beginning of each binary sequence in the first sum, and a 1 to the terms in the second sum. This leaves the first sum unchanged, and increases the value of the binary sequences in the second sum by a power of 2. Hence this expression is equal to

$$\sum_{i=0}^{2^k-1} e_i^k + \sum_{i=0}^{2^{k+1}-1} e_{i+2^k}^{k+1} = \sum_{i=0}^{2^{k+1}-1} e_i^{k+1}.$$

The result follows by induction. □

Now returning back to the Hadamard transform, we consider an interesting property, that we will see the usefulness of later: applying the H_n to the zero-state $e_0^{\otimes n}$ we obtain the following state:

$$\begin{aligned} H_n((e_0^1)^{\otimes n}) = (H_1 e_0^1)^{\otimes n} &= \left(\frac{1}{\sqrt{2}}\right)^n \underbrace{\left[(e_0^1 + e_1^1) \otimes (e_0^1 + e_1^1) \otimes \ldots \otimes (e_0^1 + e_1^1)\right]}_{n \text{ times}} \\ &= \frac{1}{2^{n/2}} \sum_{i=0}^{2^n-1} e_i^n, \end{aligned}$$

by the preceding Lemma. This is a superposition of all fundamental states of the composite system $(\mathbb{C}^2)^{\otimes n}$.

Example 3.7. Another interesting gate is the NOT-gate on a single qubit system, which maps e_0 to e_1 and vice versa, as compared to the logical NOT that negates the statement to which it is applied. This transformation is defined by

$$N = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The final phase of quantum computation is the *measurement* where we actually retrieve information from a state after a computation has been made.

Definition 3.8. A *projective measurement* on a finite dimensional qubit register $\mathcal{V}$, is a finite collection of nonzero orthogonal projections $P_1, P_2, \ldots, P_k$, where P_i is a projection of $\mathcal{V}$ onto $\mathcal{V}_i$ such that

$$\sum_{i=1}^{k} P_i = I_{\mathcal{V}}.$$

Let $v \in \mathcal{V}$ be a quantum state. In our model of quantum computing, applying a measurement results in a permanent change of the state to which it was applied. In particular, applying the measurement $P_1, \ldots, P_k$ on v results in an observation of $i \in \{1, \ldots, k\}$ with probability $p(i) = ||P_i(v)||^2$, and the state v is transformed to

$$\frac{P_i(v)}{||P_i(v)||}.$$

Furthermore, it follows by Pythagoras that this defines a probability measure because

$$\sum_{i=1}^{k} p(i) = \sum_{i=1}^{k} ||P_i(v)||^2 = \left\| \sum_{i=1}^{k} P_i(v) \right\|^2 = ||v||^2 = 1.$$

This means that every time we measure a quantum register, and observe i, the probability of obtaining any other outcome in a second measurement becomes zero. Hence, further measurements will not provide any additional information. Note that if i is observed, it must be that $p(i) \neq 0$, so $P_i v \neq 0$.

Example 3.9. Let $v = \alpha e_0 + \beta e_1$ be a single-qubit system. Then we have the projections

$$P_0 = e_0 \otimes e_0^* = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad \text{and} \quad P_1 = e_1 \otimes e_1^* = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

If we measure and observe 0, the system is transformed to the state

$$\frac{P_0 v}{||P_0 v||} = \frac{\alpha e_0}{\sqrt{|\alpha|^2}} = \frac{\alpha}{|\alpha|} e_0.$$

Similarly, if 1 is observed, the system is transformed into

$$\frac{P_0 v}{||P_0 v||} = \frac{\beta}{|\beta|} e_1.$$

§ 3.2. The Discrete Fourier Transform.

A central part of quantum computation is the use of the quantum Fourier transform. A special case of this unitary transformation occurs when we consider the cyclic group $\mathbb{Z}/N\mathbb{Z}$. This is called the *discrete quantum Fourier transform*, or *cyclic Fourier transform*.

Definition 3.10. Let A be an additive Abelian group, and B a set. A function $f : A \longrightarrow B$ is said to be *periodic* with period r if $f(a + r) = f(a)$, for every $a \in A$.

Let $L^2(\mathbb{Z}/n\mathbb{Z}) = \{f : \mathbb{Z}/n\mathbb{Z} \longrightarrow \mathbb{C}\}$ be the n-dimensional Hilbert space with inner product

$$\langle f, g \rangle = \sum_{x \in \mathbb{Z}/n\mathbb{Z}} f(x)\overline{g(x)} \quad \text{for } f, g \in L^2(\mathbb{Z}/n\mathbb{Z}).$$

An orthonormal basis of this space is the family of functions $\{\delta_x\}_{x \in \mathbb{Z}/n\mathbb{Z}}$ defined as

$$\delta_i(j) = \begin{cases} 1 & \text{if } i \equiv j \pmod{n} \\ 0 & \text{otherwise.} \end{cases}$$

We define the *characters* of $\mathbb{Z}/n\mathbb{Z}$ to be the set of homomorphisms $\{\chi_c\}_{c \in \mathbb{Z}/n\mathbb{Z}}$, from $\mathbb{Z}/n\mathbb{Z}$ to $\mathbb{C}^*$ such that $\chi_c(x) = e^{2\pi i c x/n}$. We will learn an alternative definition in Section 5.1 which applies to the more general case of an arbitrary finite group G.

Lemma 3.11. *Let $n \in \mathbb{N}$, then*

$$\sum_{k=0}^{n-1} e^{2\pi i k/n} = 0.$$

Proof. We have that

$$\left(e^{2\pi i/n} - 1\right) \sum_{k=0}^{n-1} e^{2\pi i k/n} = e^{2\pi i n/n} - 1 = 0,$$

and since $e^{2\pi i/n} \neq 1$, it must be that the sum is equal to zero. □

Theorem 3.12. *Let $\{\chi_0, \chi_1, \dots, \chi_{n-1}\}$ be the set of characters of $\mathbb{Z}/n\mathbb{Z}$, then the family*

$$C = \left\{ \frac{\chi_c}{\sqrt{n}} \right\}_{c \in \mathbb{Z}/n\mathbb{Z}}$$

form an orthonormal basis of $L^2(\mathbb{Z}/n\mathbb{Z})$.

Proof. Let $\chi_c/\sqrt{n}$ and $\chi_d/\sqrt{n}$ be two such characters. If $c = d$, by definition of the inner product of $L^2(\mathbb{Z}/n\mathbb{Z})$ we have that

$$\left\langle \frac{\chi_c}{\sqrt{n}}, \frac{\chi_c}{\sqrt{n}} \right\rangle = \frac{1}{n} \sum_{x \in \mathbb{Z}/n\mathbb{Z}} e^{2\pi i x c/n} \cdot e^{-2\pi i x c/n} = \frac{1}{n} \sum_{x \in \mathbb{Z}/n\mathbb{Z}} 1 = 1.$$

Now if $c \neq d$ we instead have that

$$\begin{aligned} \left\langle \frac{\chi_c}{\sqrt{n}}, \frac{\chi_d}{\sqrt{n}} \right\rangle &= \frac{1}{n} \sum_{x \in \mathbb{Z}/n\mathbb{Z}} e^{2\pi i x c/n} \cdot e^{-2\pi i x d/n} \\ &= \frac{1}{n} \sum_{x \in \mathbb{Z}/n\mathbb{Z}} (e^{2\pi i (c-d)/n})^x \\ &= \frac{1 - (e^{2\pi i (c-d)/n})^n}{n(1 - e^{2\pi i (c-d)/n})} = 0. \end{aligned}$$

□

Definition 3.13. The *discrete Fourier transform* is the transformation, $\mathcal{F} : L^2(\mathbb{Z}/n\mathbb{Z}) \longrightarrow L^2(\mathbb{Z}/n\mathbb{Z})$ defined by

$$[\mathcal{F}(f)](c) = \left\langle f, \frac{\chi_c}{\sqrt{n}} \right\rangle.$$

This function is commonly denoted by $\hat{f}$, and we have more explicitly that

$$\hat{f}(c) = \frac{1}{\sqrt{n}} \sum_{a \in \mathbb{Z}/n\mathbb{Z}} e^{\frac{-2\pi i a c}{n}} f(a).$$

Applying $\mathcal{F}$ to the basis of delta functions we obtain the matrix representation

$$Q_n = \left(\overline{\chi_c(x)}/\sqrt{n}\right)_{0 \leq c, x \leq n-1}.$$

Example 3.14. For $n = 2$ we obtain the familiar matrix

$$Q_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

which we recognize as the Hadamard transformation on a single qubit system. However, this does not extend to larger n, consider for example $n = 4$ which results in the matrix

$$Q_4 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix}.$$

SECTION 4
Shor's Algorithm

One of the most widely known and used algorithms in cryptography today is the so called RSA cryptosystem. The main reason for its widespread use being that the fastest algorithms to break the encryption runs in sub-exponential time, which is far worse than the desired polynomial time complexity. The hard underlying problem of RSA is the integer factorization problem, which is also of great mathematical interest. In 1994, Peter W. Shor [Sho97] introduced a quantum algorithm which makes it possible to factorize a large integer N in polynomial time. The fundamental idea of the algorithm is to reduce factorization to a problem of determining the order of elements in cyclic groups, and then utilize the properties of the discrete quantum Fourier transform to retrieve information about a subgroup generated by the element in question in the additive group $\mathbb{Z}/N\mathbb{Z}$. This information can then be used to classically obtain the generator for this subgroup, and thus the period of f.

§ 4.1. Reducing the Factorization Problem. Our first goal is to reduce the factorization problem to one that we can solve using a quantum computer. Let N be a large, odd integer, and pick a random integer $1 < y < N$. Using the Euclidean algorithm we can quickly determine whether y and N share a common factor, because if $1 < d = \gcd(y, N)$, then d is a common factor of N and y. However, if N is large, it is very unlikely that we find such a y, so we can assume this is not the case, and $\gcd(y, N) = 1$. Since y is invertible modulo N in this case we can consider it an element of the finite group $(\mathbb{Z}/N\mathbb{Z})^*$. Let r be the order of y in this group, so that $y^r = 1$. This means that $y^r - 1 = 0$ modulo N, and hence it must be that $N \mid (y^r - 1)$.

Now, if we are lucky and it happens that r is even, we have

$$y^r - 1 = (y^{r/2} - 1)(y^{r/2} + 1) = 0 \pmod{N},$$

and since $N \nmid (y^{r/2} - 1)$ it must be that N and $y^{r/2} + 1$ share at least one prime factor. So we can again use the Euclidean algorithm to determine $d = \gcd(N, y^{r/2} + 1)$. If $d = 1$ or $d = N$ the test has failed, and we cannot get any further with this choice of y, but if $d > 1$ we have once again been lucky and found a factor of N.

In order for this process to actually be useful, we require two things: first an efficient algorithm for finding the order r of y, and second we need to find a y of even order r, such that $y^{r/2} + 1$ is not divisible by N. For the first part we will require a quantum computer, but for the second we can rely on the following proposition to ensure that the probability of finding such a y is not too small.

Theorem 4.1. *Let N be a large odd integer with prime factorization $N = p_1^{s_1} p_2^{s_2} \dots p_m^{s_m}$, and let*

$$\{y \in (\mathbb{Z}/N\mathbb{Z})^* : r = ord(y) \equiv 0 \pmod{2},\ y^{r/2} + 1 \not\equiv 0 \pmod{N}\}.$$

Then this set contains at least

$$\varphi(N)\left(1 - \frac{1}{2^{m-1}}\right)$$

elements, where φ denotes Euler's function that counts the number of integers up to N that are that are relative prime to N.

Proof. First we determine the number of elements in $(\mathbb{Z}/N)^*$ that have odd order. By the Chinese remainder theorem $\mathbb{Z}/N \equiv \mathbb{Z}/p_1^{s_1} \times \mathbb{Z}/p_2^{s_2} \times \ldots \times \mathbb{Z}/p_m^{s_m}$. Hence picking a $y \in (\mathbb{Z}/N)^*$ is equivalent to picking $y_i \in (\mathbb{Z}/p_i^{s_i})^*$ for each $i \in \{1,\ldots,m\}$. Each $(\mathbb{Z}/p_i^{s_i})^*$ is cyclic of order $\varphi(p_i^{s_1}) = p_i^{s_i-1}(p_i-1)$. Half of these elements has odd order, so the total number of elements of odd order in $(\mathbb{Z}/N)^*$ is equal to

$$\prod_{i=1}^{m} \frac{1}{2}\varphi(p_i^{s_i}) = \frac{1}{2^m}\varphi(N).$$

Next, we find the number of elements y of order r such that $y^{r/2} \equiv -1 \pmod N$. For such y by the Chinese remainder theorem we have that $y^{r/2} \equiv -1 \pmod{p_i^{s_i}}$ for all i, so y has even order in $(\mathbb{Z}/p_i^{s_i})^*$. So there are at most $\frac{1}{2^m}\varphi(N)$ such y.

Thus, the order of the set of good y's is at least

$$\varphi(N) - 2\cdot\frac{1}{2^m}\varphi(N) = \varphi(N)\left(1 - \frac{1}{2^{m-1}}\right).$$

□

This means that in the worst case scenario when N only has two prime factors, we are still able to find a "good" y with probability $\frac{1}{2}$, and if there are more prime factors the process is even more likely to succeed quickly. So by randomization it appears that the factorization problem has been reduced to a problem of determining the order of elements in $(\mathbb{Z}/N\mathbb{Z})^*$.

§ **4.2. The Quantum Subroutine.** First and foremost, we want to somehow encode the information of a periodic function to a qubit register. So let N be a large integer such that $2^{n-1} < N^2 \leq 2^n$, and $l = \lceil \log_2 N \rceil$. Now let $1 < y < N$ be an integer such that $\gcd(y, N) = 1$ with order r in $(\mathbb{Z}/N\mathbb{Z})^*$, and consider the function $f : \mathbb{Z}/2^n\mathbb{Z} \to \mathbb{Z}/2^l\mathbb{Z}$ given by

$$x \mapsto y^x \pmod N,$$

which is not necessarily periodic. We choose $\mathbb{Z}/2^n\mathbb{Z}$ in the definition of f in order to make sure the algorithm succeeds with a good probability in Theorem 4.2 later. The codomain is chosen to be the smallest power of 2 neccessary to capture the order of y.

We define the transformation $U_f : \mathcal{V}_n \otimes \mathcal{V}_l \to \mathcal{V}_n \otimes \mathcal{V}_l$ taking $e_i \otimes e_j$ to $e_i \otimes e_{f(i)+j}$. Here e_i takes values among the fundamental states of $\mathcal{V}_n$, and $e_{f(i)+j}$ takes all values among the fundamental states of $\mathcal{V}_l$ exactly once while j varies over $\{0,\ldots,2^l-1\}$. Hence, U_f is applied to all the fundamental states of the tensor product $\mathcal{V}_n \otimes \mathcal{V}_l$, and permutes these elements, so it is unitary. This transformation encodes f since we retrieve f by taking the tensor of $e_i \in \mathcal{V}_n$ with the zero state $e_0 \in \mathcal{V}_l$, so that

$$U_f(e_i \otimes e_0) = e_i \otimes e_{f(i)}.$$

In this way, we obtain access to the world of classical functions on a quantum computer through what is called a *quantum oracle*.

We may use this oracle together with the previously mentioned Hadamard transform to encode the function f as follows:

$$\begin{aligned} U_f(H_n(e_0^{\otimes n}) \otimes e_0^{\otimes l}) &= U_f\left(\frac{1}{2^{n/2}}\sum_{i=0}^{2^n-1} e_i \otimes e_0^{\otimes l}\right) \\ &= \frac{1}{2^{n/2}}\sum_{i=0}^{2^n-1} U_f(e_i \otimes e_0^{\otimes l}) \\ &= \frac{1}{2^{n/2}}\sum_{i=0}^{2^n-1} e_i \otimes e_{f(i)}. \end{aligned}$$

This entangled state now contains information about all the function values of f. The trickier part, and where the findings of Shor really come into play, is when we turn to the problem of extracting information from this entanglement, without collapsing the state. As mentioned previously, making multiple measurements of the same kind on the same register yields no further information.

The first step to extracting information from this state is to measure the second register, obtaining a random state e_j in the image of f. So let $j \in \mathbb{Z}/2^l\mathbb{Z}$ and $P_j : \mathcal{V}_l \to \mathcal{V}_l$ be the orthogonal projection onto the line $\mathbb{C}e_j$. Then $\{\mathrm{id}_{\mathcal{V}_n} \otimes P_j\}_{j\in\mathbb{Z}/2^l\mathbb{Z}}$ form a measurement. Applying this measurement we find that the probability of observing j is equal to

$$\left\|(\mathrm{id}_{\mathcal{V}_n} \otimes P_j)\left(\frac{1}{2^{n/2}}\sum_{i=0}^{2^n-1} e_i \otimes e_{f(i)}\right)\right\|^2 = \left\|\frac{1}{2^{n/2}}\sum_{i\in f^{-1}(j)} e_i \otimes e_j\right\|^2 = \frac{1}{2^n}|f^{-1}(j)|.$$

So if we do observe j, it must be that the preimage

$$f^{-1}(j) = \{j + kr \mid 0 \le j < r, k = 0, 1, \dots, K-1\},$$

where $K = \min\{k : j + kr \ge 2^n\}$, is nonempty. Denote by v the state before measurement, then after observing j, the system v is in the state

$$\begin{aligned} \frac{(\mathrm{id}_{\mathcal{V}_n} \otimes P_j)(v)}{||(\mathrm{id}_{\mathcal{V}_n} \otimes P_j)(v)||} = \left(\frac{1}{\sqrt{|f^{-1}(j)|}}\sum_{i\in f^{-1}(j)} e_i\right) \otimes e_j &= \left(\frac{1}{\sqrt{K}}\sum_{k=0}^{K-1} e_{j+kr}\right) \otimes e_j \\ &= \left(\sum_{i=0}^{2^n-1} \psi_j(i) e_i\right) \otimes e_j, \end{aligned} \tag{4.2.1}$$

where

$$\psi_j(i) = \begin{cases} \frac{1}{\sqrt{K}} & \text{if } r \mid (i-j) \\ 0 & \text{otherwise.} \end{cases}.$$

Next, we apply the discrete Fourier transform (see Section 3.2) to the first register of (4.2.1) which yields

$$(\mathcal{F} \otimes \mathrm{id}_{\mathcal{V}_l}) \left(\sum_{i=0}^{2^n-1} \psi_j(i) e_i \right) \otimes e_j = \left(\sum_{k=0}^{2^n-1} \hat{\psi}_j(k) e_k \right) \otimes e_j.$$

Let $P_c : \mathcal{V}_n \longrightarrow \mathcal{V}_n$ be the orthogonal projection onto the line $\mathbb{C}e_c$, and apply the measurement $\{P_c \otimes \mathrm{id}_{\mathcal{V}_l}\}_{c \in \mathbb{Z}/2^n\mathbb{Z}}$. Then the probability of observing c is equal to

$$||\hat{\psi}_j(c) e_c \otimes e_j||^2 = |\hat{\psi}_j(c)|^2 = \left| \left\langle \psi_j, \frac{\chi_c}{\sqrt{2^n}} \right\rangle \right|^2 = \frac{1}{K 2^n} \left| \sum_{k=0}^{K-1} e^{\frac{2\pi i c (j+kr)}{2^n}} \right|^2. \tag{4.2.2}$$

The following theorem tells us that we now have a reasonable chance of determining the period r.

Theorem 4.2. *The probability of observing $c \in \mathbb{Z}/2^n\mathbb{Z}$, with the property that there exists an integer s such that $0 \leq s < r$ with $gcd(s,r) = 1$ and*

$$\left| \frac{c}{2^n} - \frac{s}{r} \right| < \frac{1}{2r^2},$$

is at least

$$\frac{4}{\pi^2} \cdot \frac{\varphi(r)}{r} \left(1 - \left(\frac{\pi r}{2 \cdot 2^n} \right)^2 \right). \tag{4.2.3}$$

Now we see that given an integer s, and a corresponding c such that

$$\left| \frac{c}{2^n} - \frac{s}{r} \right| < \frac{1}{2 \cdot 2^n},$$

our former choice of $\mathbb{Z}/2^n\mathbb{Z}$ in the definition of f now makes sense since this inequality implies that

$$\left| \frac{c}{2^n} - \frac{s}{r} \right| < \frac{1}{2r^2}.$$

With this theorem, all parts are in place for Shor's algorithm, and we will prove it shortly. The remaining steps rely on the theory of continued fractions, encountered in Section 2.3, as we will see in Section 4.3.

However, if f happens to be periodic, we can make use of a shortcut in this process using the following Lemma.

Lemma 4.3. *Let r be a factor of $n \in \mathbb{N}$, and $f : \mathbb{Z}/n\mathbb{Z} \longrightarrow \mathbb{C}$ a periodic function of period r. Then $\hat{f}(c) = 0$ for all $c \neq 0 \pmod{\frac{n}{r}}$.*

Proof. We see that

$$\begin{aligned}
[\mathcal{F}(f)](c) &= \frac{1}{\sqrt{n}} \sum_{x \in \mathbb{Z}/n\mathbb{Z}} f(x) \chi_{-c} \\
&= \frac{1}{\sqrt{n}} \sum_{a=0}^{r-1} \sum_{b=0}^{\frac{n}{r}-1} f(a+br) \chi_{-c}(a+br) \\
&= \frac{1}{\sqrt{n}} \sum_{a=0}^{r-1} f(a) \chi_{-c}(a) \sum_{b=0}^{\frac{n}{r}-1} \chi_{-c}(br).
\end{aligned}$$

Now since n/r does not divide c, we have that $\chi_c(1) = e^{2\pi i c/n} \neq 1$, and the result follows by Lemma 3.11. □

In practice, this Lemma assures that the probability (4.2.2) is zero for any $c \neq 0 \pmod{\frac{n}{r}}$, which means that we will only observe states that are multiples of $\frac{n}{r}$. Hence, in the periodic case we necessarily get the equality $|\frac{c}{2^n} - \frac{s}{r}| = 0$ for some integer s. Then we can simply read off r since $\frac{c}{2^n}$ is known.

After the following intermediary lemma we are ready to prove Theorem 4.2, making no assumptions on the periodicity of f.

Lemma 4.4. *Let $n \in \mathbb{N}$ and $f : \mathbb{R} \to \mathbb{C}$ be a function given by*

$$f(\alpha) = \left| \sum_{n=0}^{N} e^{i\alpha n} \right|^2 .$$

Then f is monotonically decreasing on $[0, \pi/N]$.

Proof. We write

$$f(\alpha) = \sum_{m,n=0}^{N} e^{i\alpha(m-n)},$$

and obtain

$$\begin{aligned} \frac{df}{d\alpha} f(\alpha) &= \sum_{m,n=0}^{N} i(m-n)e^{i\alpha(m-n)} \\ &= \sum_{k=1}^{N} k(N-k+1)i(e^{i\alpha k} - e^{-i\alpha k}) \\ &= \sum_{k=1}^{N} -k(N-k+1)\mathrm{Im}(e^{i\alpha k}). \end{aligned}$$

For $\alpha \in [0, \pi/N]$, we have that $\mathrm{Im}(e^{i\alpha k}) \geq 0$ for all $k \in \{1, \ldots, N\}$. So $\frac{df}{d\alpha} \leq 0$ in this interval. This shows that f is monotonically decreasing. □

Proof. (Theorem 4.2) It follows by the statement of the theorem that

$$|cr - s2^n| < \frac{r}{2}, \tag{4.2.4}$$

and we denote the difference $cr - s2^n$ by ε.

The conditions on K implies that $|Kr - 2^n| < r$. This is the final relation we need, and we are now ready to begin the proof of Theorem 4.2.

Consider the probability described in (4.2.2). It can be simplified as follows:

$$\frac{1}{K2^n} \left| \sum_{k=0}^{K-1} e^{\frac{2\pi i c(j+kr)}{2^n}} \right|^2 = \frac{1}{K2^n} \left| \sum_{k=0}^{K-1} e^{\frac{2\pi i ckr}{2^n}} \right|^2 .$$

We want to determine a lower bound of this expression. First, we rewrite this expression on the form

$$\frac{1}{K2^n}\left|\sum_{k=0}^{K-1} e^{\frac{2\pi ickr}{2^n}}\right|^2 = \frac{1}{K2^n}\left|\sum_{k=0}^{K-1} e^{\frac{2\pi ik(\varepsilon+s2^n)}{2^n}}\right|^2 = \frac{1}{K2^n}\left|\sum_{k=0}^{K-1} e^{\frac{2\pi i\varepsilon k}{2^n}}\right|^2.$$

Now by (4.2.4) and Lemma 4.4 we have that

$$\frac{1}{K2^n}\left|\sum_{k=0}^{K-1} e^{\frac{2\pi i\varepsilon k}{2^n}}\right|^2 > \frac{1}{K2^n}\left|\sum_{k=0}^{K-1} e^{\frac{i\pi rk}{2^n}}\right|^2.$$

This can be further simplified using geometric series as follows

$$\frac{1}{K2^n}\left|\sum_{k=0}^{K-1}\left(e^{\frac{i\pi r}{2^n}}\right)^k\right|^2 = \frac{1}{K2^n}\left|\frac{\left(e^{\frac{\pi ir}{2^n}}\right)^K - 1}{e^{\frac{\pi ir}{2^n}} - 1}\right|^2.$$

Consider for a moment the expression $|\cos 2x + i\sin 2x - 1|^2$. By the double angle formulae it follows that

$$\begin{aligned}|\cos 2x + i\sin 2x - 1|^2 &= |-2\sin^2 x + 2i\sin x\cos x|^2\\ &= |2i\sin x(\cos x + i\sin x)|^2\\ &= 4\sin^2 x.\end{aligned}$$

So by writing $e^{\pi ir/2^n}$ in trigonometric form we get the following expression

$$\frac{1}{K2^n}\left|\frac{\cos\frac{K\pi r}{2\cdot 2^n} + i\sin\frac{K\pi r}{2\cdot 2^n} - 1}{\cos\frac{\pi r}{2\cdot 2^n} + i\sin\frac{\pi r}{2\cdot 2^n} - 1}\right|^2 = \frac{1}{K2^n}\frac{\sin^2\frac{K\pi r}{2\cdot 2^n}}{\sin^2\frac{\pi r}{2\cdot 2^n}}.$$

Now, using the properties of K mentioned previously we have that

$$\frac{K\pi r}{2\cdot 2^n} > \frac{\pi}{2}\cdot\frac{2^n - r}{2^n} = \frac{\pi}{2}\left(1 - \frac{r}{2^n}\right).$$

Since $r/2^n$ is small we can use Taylor expansions and obtain the following expression

$$\sin^2\left(\frac{K\pi r}{2\cdot 2^n}\right) > \sin^2\left(\frac{\pi}{2}\left(1 - \frac{r}{2^n}\right)\right) \geq 1 - \left(\frac{\pi r}{2\cdot 2^n}\right)^2.$$

We also have that

$$\sin^2\left(\frac{\pi r}{2\cdot 2^n}\right) \leq \left(\frac{\pi r}{2\cdot 2^n}\right)^2,$$

so that

$$\begin{aligned}\frac{1}{K2^n}\frac{\sin^2\frac{K\pi r}{2\cdot 2^n}}{\sin^2\frac{\pi r}{2\cdot 2^n}} &\geq \frac{1}{K2^n}\left(1 - \left(\frac{\pi r}{2\cdot 2^n}\right)^2\right)\cdot\left(\frac{2\cdot 2^n}{\pi r}\right)^2\\ &= \frac{4}{\pi^2 r}\cdot\frac{2^n}{Kr}\left(1 - \left(\frac{\pi r}{2\cdot 2^n}\right)^2\right)\\ &> \frac{4}{\pi^2 r}\left(1 - \left(\frac{\pi r}{2\cdot 2^n}\right)^2\right).\end{aligned}$$

Finally, since the number of s relatively prime to r is equal to $\varphi(r)$, the probability that gcd(s, r) $= 1$ for a given $0 < s \leq r$ is equal to $\varphi(r)/r$. The number of possible values to observe in the second register is r. Hence, the probability of observing a c corresponding to such an s is at least

$$\frac{4}{\pi^2} \cdot \frac{\varphi(r)}{r} \left(1 - \left(\frac{\pi r}{2 \cdot 2^n}\right)^2\right).$$

□

Furthermore, from [Apo76] Theorem 13.14, we have the bound

$$\frac{\varphi(r)}{r} > \frac{1}{e^\gamma \log_e \log_e r + \frac{2.50637}{\log_e \log_e r}}, \tag{4.2.5}$$

where $\gamma \approx 0.5772$ denotes Euler's constant. This, as we will see in the following section, proves that the algorithm has a great probability of success if it is repeated several times.

§ 4.3. Post Processing. By Theorem 4.2 we know that at the end of the quantum subroutine of Shor's algorithm, an integer c is observed such that

$$\left|\frac{c}{2^n} - \frac{s}{r}\right| < \frac{1}{2r^2},$$

where $0 \leq s < r$ are integers such that $\gcd(s, r) = 1$, with high probability. From this relation we wish to determine the period r. Since c and 2^n are known at this stage, we recognize this as the same expression as that of Theorem 2.17, where $x = \frac{c}{2^n}$. So $\frac{s}{r}$ is a convergent of $\frac{c}{2^n}$. Consequently, by examining the convergents of $\frac{c}{2^n}$ we find r. As seen in Section 2.3 this can be done in polynomial time on 2^n.

§ 4.4. Time Complexity. What really makes Shor's algorithm of interest is its superior time complexity compared to other, classical, factoring algorithms such as the quadratic sieve or the number field sieve. We have already seen that Shor's algorithm is probabilistic in the sense that we pick a y with good properties with a certain probability, and the success of the entire algorithm hinges on this choice. Luckily, as we saw in Theorem 4.1, the probability of picking a y with the desired properties is at least $\frac{1}{2}$. In principle, this means that in the worst case scenario when N is the product of two primes, after choosing only 7 different y 's the probability of a good choice is $1 - (1/2)^7 \approx 0.99$.

As we saw in Section 4.2, there is also another point of uncertainty that lies in how many times we expect to have to run the quantum subroutine. However, since there is a lower bound on $\varphi(r)/r$, given by (4.2.5), we can determine how many times the quantum algorithm needs to be performed to succeed with a probability of at least $\frac{2}{3}$ for a given size of N and r. For example, if $N = r = 10^{100}$ are fairly large numbers, after 28 tries the probability of success is at least

$$1 - \left(1 - \frac{4}{\pi^2} \cdot \frac{1}{e^\gamma \log_e \log_e 10^{100} + \frac{2.50637}{\log_e \log_e 10^{100}}} \left(1 - \left(\frac{\pi \cdot 10^{100}}{2 \cdot 2^{665}}\right)^2\right)\right)^{28} \approx 0.67994 \geq \frac{2}{3}.$$

In practice, given the sizes of N and r, we can determine the number of tries neccesary to suceed with the desired probability in the same way.

However, if the algorithm is really slow, having to run it several times can be a big issue. Therefore, it is of interest to us to determine how much time the algorithm requires from start to finish.

Let N be a large integer such that $2^{n-1} < N^2 \leq 2^n$ and $l = \lceil \log N \rceil$. As was already established, the Euclidean algorithm that is used to find the greatest common divisor of N and a randomly chosen $y < N$, runs in $\mathcal{O}(\log N) = \mathcal{O}(l)$ time. So the classical part preceding the quantum subroutine runs quickly.

For the quantum subroutine we first apply the Hadamard transform to the first register, and encode the information of f by applying the quantum oracle to the double register. The Hadamard gate requires $\mathcal{O}(l)$ steps, and using an algorithm for modular exponentiation, as described in Chapter 31 of [CLRS09], requires $\mathcal{O}(l^3)$ operations. Next, the discrete Fourier transform is applied to the first register, which requires $\mathcal{O}(l^2)$ additional steps [BBDR04]. The final part of the algorithm, that uses the theory of continued fractions, has the same running time as the Euclidean algorithm, so it has complexity $\mathcal{O}(l)$.

This analysis, together with the expected number of times we need to run the algorithm to get a probability of success greater than or equal to $2/3$, shows that the complete process has time complexity $\mathcal{O}((\log N)^3 \log\log N)$. This means that Shor's algorithm runs in polynomial time on the size of N, which is a huge improvement to the fastest classical algorithms in use today.

§ 4.5. The Algorithm.

We have now covered each part of Shor's algorithm, and hopefully the reader is convinced that it provides us with a probabilistic method of successfully factoring large integers, with time complexity $\mathcal{O}((\log N)^3 \log\log N)$.

Now has come the time to summarize the algorithm with all of its parts, and provide an illustrating example of the process, factoring $N = 15$. We start with a summary of the process from start to finish, with an arbitrary integer N, such that $2^{n-1} < N^2 \leq 2^n$.

Pre-processing.

1. Pick, at random, an integer $y < N$.
2. Calculate $\gcd(y, N) = d$. If $d \neq 1$, the algorithm terminates, otherwise continue.
3. Let $f : \mathbb{Z}/2^n\mathbb{Z} \longrightarrow \mathbb{Z}/2^l\mathbb{Z}$, be the function mapping x to $y^x \pmod N$, where $l = \lceil \log N \rceil$.

Quantum Subroutine.

4. Prepare the two quantum registers $V_n \otimes V_l$ in the zero-state $e_0^{\otimes n} \otimes e_0^{\otimes l}$.
5. Apply the Hadamard transform H_n to the first register.
6. Encode the function values of f by applying the quantum oracle U_f to the second register. The machine is now in the state

$$\frac{1}{2^{n/2}} \sum_{i=0}^{2^n-1} e_i \otimes e_{f(i)}.$$

7. Measure the second register.

8. Apply the discrete Fourier transform $\mathscr{F}$ to the first register.
9. Measure the first register, obtaining the state e_c.

Post-processing.

10. Use the continued fraction algorithm to obtain the set, S, of convergents for $\frac{c}{2^n}$.
11. Pick a random $\frac{s}{r} \in S$, where $\gcd(s, r) = 1$. Check that $r = 0 \pmod 2$ and that $y^{r/2} + 1 \neq 0 \pmod N$, if this is not true, return to step 1, otherwise continue. If no such s exists, return to step 4.
12. Compute $\gcd(y^{r/2} + 1, N)$ and $\gcd(y^{r/2} - 1, N)$, obtaining at least one prime factor of N.

Example 4.5. Take $N = 15 = 3 \cdot 5$, noting that $15^2 = 225 \leq 2^8$, as the integer we want to factor. We first pick a random element $y < N$, for the sake of example we pick $y = 2$. As expected, $\gcd(2, 15) = 1$, and the algorithm continues. We are thus to determine the period of the function $f(x) = 2^x \pmod{15}$. Since this is a small example, we can already determine this value, and for purposes of demonstration we note that $f(4) = 2^4 = 1 \pmod{15}$.

Next, we initiate the double quantum register and apply the Hadamard transform, and then the quantum oracle, obtaining the state

$$\frac{1}{\sqrt{256}} \sum_{i=0}^{255} e_i \otimes e_{f(i)} = \frac{1}{\sqrt{256}} \left(e_0 e_1 + e_1 e_2 + e_2 e_4 + e_3 e_8 + \ldots \right).$$

Now, measuring the second register, we obtain either e_1, e_2, e_4 or e_8 with probability $\frac{1}{4}$. Say e_1 is observed. Then, using what we know of the order of f, $K = \min\{k : 1 + 4k \geq 2^8\} = 64$. After measurement and applying $\mathscr{F}$ to the first register, the system is in the state:

$$\left(\sum_{i=0}^{255} \hat{\psi}_1(k) e_k \right) \otimes e_1 = \frac{1}{\sqrt{256}} \sum_{k=0}^{255} \sum_{c=0}^{255} e^{\frac{2\pi i k c}{256}} e_c \otimes e_1,$$

where $c \in \mathbb{Z}/256\mathbb{Z}$ is observed with probability

$$\frac{1}{64 \cdot 256} \left| \sum_{k=0}^{63} e^{\frac{2\pi i c(1+4k)}{256}} \right|^2.$$

As we see from Figure 1, the most probable values are $c = 0, 64, 128$ and 192, each observed with probability approximately equal to $\frac{1}{4}$.

Say $c = 64$ is observed. Then

$$\left| \frac{64}{256} - \frac{s}{r} \right| < \frac{1}{2 \cdot r^2} = \frac{1}{32}.$$

The fraction $\frac{64}{256}$ has convergents 0 and $\frac{1}{4}$, and since 0 is clearly not a valid option, we get

$$\frac{s}{r} = \frac{1}{4}.$$

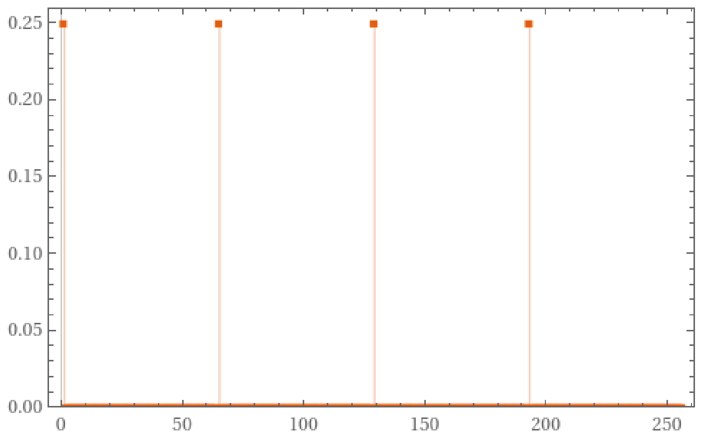


Figure 1: Probabilities of observing c between 0 and 255.

This choice passes our checks since 1 and 4 are relatively prime, $4 = 0 \pmod 2$, and $2^{4/2} + 1 = 5 \pmod{15}$.

Computing $\gcd(2^2 + 1, 15) = 3$ and $\gcd(2^2 - 1, 15) = 5$, we obtain two unique factors of 15 in this run of the algorithm. Hence, we have successfully factored $N = 15$.

SECTION 5
The Hidden Subgroup Problem

Now that we have seen an example of a quantum algorithm that successfully solves a problem to which no efficient classical algorithm is known, it is reasonable to ask whether it is possible to generalize the process, and perhaps provide tools to help solve other mathematical problems as well.

The period finding problem of last section is actually a special case of a broader problem called the *hidden subgroup problem* which can be stated as follows.

Definition 5.1. Let G be a group, H a subgroup, and X a finite set. Suppose $f : G \to X$ is a function such that for all $g_1, g_2 \in G$, the equality $f(g_1) = f(g_2)$ holds if and only if $g_1H = g_2H$. Then H is the subgroup of G hidden by f, and *the hidden subgroup problem* is to find H, given f. We say that f *separates cosets* of H.

In fact, many different mathematical problems can be reduced to the hidden subgroup problem, which makes efficient algorithms for solving this problem for different families of groups very useful. We now show how the factorization problem of the previous section falls under this category.

Example 5.2. Recall the reduction of the factorization problem of Section 4.1, where we wished to factor the integer N. We then introduced the periodic function $f : \mathbb{Z}/2^n\mathbb{Z} \to \mathbb{Z}/2^l\mathbb{Z}$, taking x to y^x modulo N. Let G be the additive group $\mathbb{Z}/2^n\mathbb{Z}$, r the period of f, and H the subgroup generated by r. Then H is the hidden subgroup of G, since periodicity implies that f is constant on elements of H, and distinct otherwise. So Shor's algorithm is effectively an algorithm for solving the hidden subgroup problem over cyclic groups.

In Section 6 we will investigate the hidden subgroup problem over symmetric groups, and how this could help solve the well known graph isomorphism problem.

§ 5.1. Representation Theory Basics.

Section 3.2 introduced the discrete Fourier transform, which we could use to solve the Abelian hidden subgroup problem. For other families of groups, however, we will need a more general version of the quantum Fourier transform. This section presents the necessary foundations of representation theory.

Recall that if $\mathcal{V}$ is a vector space of dimension n over a field F, then $GL(\mathcal{V})$ is the automorphism group of $\mathcal{V}$, that is, the group of all nonsingular linear transformations from $\mathcal{V}$ to itself, called the general linear group. Associated with this group is the group $GL_n(F)$, of matrices obtained by fixing a basis of $\mathcal{V}$ of size n.

Definition 5.3. Let G be a group, and $\mathcal{V}$ a vector space over a field F. A *representation* of G, is a homomorphism $\rho : G \to GL(\mathcal{V})$. The *degree* of the representation is the dimension of $\mathcal{V}$.

Two representations ρ and ρ' of the same group G in the vector spaces $\mathcal{V}$ and $\mathcal{V}'$ respectively, are said to be *isomorphic* if there exists an isomorphism $f : \mathcal{V} \to \mathcal{V}'$ satisfying

$$f \circ \rho(g) = \rho'(g) \circ f,$$

for all $g \in G$.

Example 5.4. An important example, that we will see the significance of later, is the so called *regular representation* of a group G. This is the representation $\lambda : G \longrightarrow GL(K^G)$ defined on the basis $\{e_g\}_{g\in G}$ of the vector space $\mathscr{V}$ over the field K, by

$$\lambda_g(e_h) = e_{gh}.$$

The degree of λ_g is equal to the order of G.

Definition 5.5. Let $\rho : G \longrightarrow GL(\mathscr{V})$ be a representation of a group G on the vector space $\mathscr{V}$, and $\mathscr{W}$ be a subspace of $\mathscr{V}$. If $\mathscr{W}$ satisfies $\rho(g)\mathscr{W} \subseteq \mathscr{W}$ for every $g \in G$, then $\mathscr{W}$ is an *invariant* subspace of ρ.

Note that the subspaces $\mathscr{V}$ and $\{0\}$ are invariant since $\rho(g)\mathscr{V} = \mathscr{V}$ and $\rho(g)\{0\} = \{0\}$ for all $g \in G$. These are called the trivial invariant subspaces. We thus distinguish the nontrivial invariant subspaces with the following definition.

Definition 5.6. A representation with no nontrivial invariant subspaces is said to be *irreducible*.

As the name suggests, we can uniquely decompose a representation ρ into irreducible parts. However, in what follows we require the representation space to be over either $\mathbb{R}$ or $\mathbb{C}$ to guarantee that an inner product is defined.

Theorem 5.7. *If $\rho : G \longrightarrow GL(\mathscr{V})$ is a representation of a finite group G on the vector space $\mathscr{V}$, and $\mathscr{V}_1$ is a nontrivial invariant subspace of $\mathscr{V}$, then there exists a complementary invariant subspace $\mathscr{V}_2$ of $\mathscr{V}$ such that $\mathscr{V} = \mathscr{V}_1 \oplus \mathscr{V}_2$.*

Proof. Let $\langle \cdot, \cdot \rangle_0$ be the inner product on $\mathscr{V}$. We define the inner product

$$\langle u, v \rangle = \sum_{g\in G} \langle \rho(g)u, \rho(g)v \rangle_0,$$

also on $\mathscr{V}$, where $u, v \in \mathscr{V}$. Then

$$\langle \rho(g)u, \rho(g)v \rangle = \sum_{h\in G} \langle \rho(hg)u, \rho(hg)v \rangle_0 = \langle u, v \rangle,$$

so this inner product is invariant. Then taking $\mathscr{V}_2 = \mathscr{V}_1^{\perp}$, we obtain a complementary and invariant subspace of $\mathscr{V}$ such that $\mathscr{V} = \mathscr{V}_1 \oplus \mathscr{V}_2$. □

It follows that we can decompose the space $\mathscr{V}$ into a direct sum of invariant subspaces, $\mathscr{V} = \mathscr{V}_1 \oplus \mathscr{V}_2 \oplus \ldots \oplus \mathscr{V}_n$, for which we have the irreducible representations $\rho_i : G \longrightarrow GL(\mathscr{V}_i)$. This gives us a decomposition of the representation $\rho = \rho_1 \oplus \rho_2 \oplus \ldots \oplus \rho_n$. This property is called *complete reducibility*.

In Section 3.2 we introduced the characters of the group $\mathbb{Z}/n\mathbb{Z}$ as the set of homomorphisms from $\mathbb{Z}/n\mathbb{Z}$ to $\mathbb{C}^*$ taking x to $e^{2\pi icx/n}$. Now has come the time to reintroduce this concept in the context of what we now know.

Definition 5.8. Let $\rho : G \longrightarrow GL(\mathcal{V})$ be a representation. The *character* of this representation is the complex-valued function on G defined by

$$\chi_\rho(g) = \mathrm{Tr}(\rho(g)).$$

This is the trace of the matrix associated with $\rho(g)$.

Theorem 5.9 (Schur's Lemma). *Let $\rho_1 : G \longrightarrow GL(\mathcal{V})$ and $\rho_2 : G \longrightarrow GL(\mathcal{W})$ be **irreducible** representations of G, and $\varphi : \mathcal{V} \longrightarrow \mathcal{W}$ a map such that $\rho_2(g) \circ \varphi = \varphi \circ \rho_1(g)$ for all $g \in G$, then*

(i) If φ is not an isomorphism, then $\varphi = 0$.

(ii) If $\mathcal{V} = \mathcal{W}$, then $\varphi = \lambda \cdot I$, where $\lambda \in \mathbb{C}$ and I is the identity.

Proof. To prove *(i)*, first consider $\mathrm{Ker}(\varphi) = \{v \in \mathcal{V} \mid \varphi(v) = 0\}$. This subspace is invariant since for any $v \in \mathrm{Ker}(\varphi)$ and $g \in G$ we have that $(\varphi \circ \rho_1(g))(v) = (\rho_2(g) \circ \varphi)(v) = 0$. Since $\mathcal{V}$ is irreducible, and $\mathrm{Ker}(\varphi) \subseteq \mathcal{V}$, either $\mathrm{Ker}(\varphi) = \mathcal{V}$ or $\mathrm{Ker}(\varphi) = 0$. Unless $\varphi = 0$, we have that $\mathrm{Ker}(\varphi) = 0$. Similarly, the subspace $\mathrm{Im}(\varphi) = \{w \in \mathcal{W} \mid \varphi(v) = w, v \in \mathcal{V}\}$, is also invariant since $\mathcal{W}$ is invariant. So $\mathrm{Im}(\varphi) = \mathcal{W}$, which means that φ is an isomorphism.

Next, suppose $\mathcal{V} = \mathcal{W}$, and let $\lambda \in \mathbb{C}$ be an eigenvalue of φ. This value exists because $\mathbb{C}$ is algebraically closed. Taking $\varphi - \lambda I$ in place of φ in the previous argument and noting that this is not an isomorphism proves *(ii)*. □

This important "lemma" has several consequences that provide important steps forward. Although not advanced, we omit the proofs of these corollaries, which can all be found in [Ser77]. In what follows we let $\rho_1 : G \longrightarrow GL(\mathcal{V})$ and $\rho_2 : G \longrightarrow GL(\mathcal{W})$ be irreducible representations of G as in Theorem 5.9.

Corollary 1. *Let h be a linear map from $\mathcal{V}$ into $\mathcal{W}$, and let*

$$h^0 = \frac{1}{|G|} \sum_{g \in G} \rho_2^{-1}(g) h \rho_1(g).$$

Then the following is true:

(i) If ρ_1 and ρ_2 are not isomorphic, then $h^0 = 0$.

(ii) If $\mathcal{V} = \mathcal{W}$ and $\rho_1 = \rho_2$, then $h^0 = \frac{1}{n} Tr(h)$, where $n = dim(\mathcal{V})$.

Now, assuming ρ_1 and ρ_2 are given in matrix form as

$$\rho_1(g) = (r_{i_1 j_1}(g)), \quad \text{and} \quad \rho_2(g) = (r_{i_2 j_2}(g)).$$

Let h be given in matrix form by $x_{i_2 i_1}$, and h^0 by $x^0_{i_2 i_1}$. Then

$$h^0 = x^0_{i_2 i_1} = \frac{1}{|G|} \sum_{g \in G, j_1, j_2} r_{i_2 j_2}(g^{-1}) x_{j_2 j_1} r_{i_1 j_1}(g),$$

from which the following two corollaries can be derived.

Corollary 2. *In case (i) of Corollary 1, we have*

$$\frac{1}{|G|}\sum_{g\in G} r_{i_2j_2}(g^{-1})r_{j_1i_1}(g) = 0,$$

for all i_1, i_2, j_1, j_2.

Corollary 3. *In case (ii) of Corollary 1, we have*

$$\frac{1}{|G|}\sum_{g\in G} r_{i_2j_2}(g^{-1})r_{j_1i_1}(g) = \frac{1}{n}\delta_{i_2i_1}\delta_{j_2j_1} = \begin{cases} \frac{1}{n} & \textit{if } i_1 = i_2 \textit{ and } j_1 = j_2 \\ 0 & \textit{otherwise.} \end{cases}$$

Example 5.10. Another interesting example arise when we consider irreducible representations of Abelian groups. As we will see, in this case, the representations are actually equal to the characters. So let $\rho : G \longrightarrow GL(\mathcal{V})$ be an irreducible representation. We want to show that the dimension of $\mathcal{V}$, and thus the degree of ρ, is equal to 1. For $g, h \in G$ we have that

$$\rho(g)\rho(h) = \rho(gh) = \rho(hg) = \rho(h)\rho(g),$$

so taking $\varphi = \rho$ in Schur's Lemma, it follows that $\rho = \lambda I$, for some constant $\lambda \in \mathbb{C}$. Hence, ρ is a scalar, and it must be that $\dim(\mathcal{V}) = 1$.

Now, since $\mathcal{V} \cong \mathbb{C}$ in this case, we can again return to the discussion in Section 3.2 and note that they in fact send every $g \in G$ to an nth root of unity, where $n = |G|$.

Recall the standard inner product on functions, also introduced in Section 3.2. We can use the more general form of this,

$$(f, h) = \frac{1}{|G|}\sum_{g\in G} f(g)\overline{h(g)},$$

where f, h are complex valued functions on G, and show a more general statement of Theorem 3.12.

Theorem 5.11. *The characters of irreducible representations are orthonormal.*

Proof. Let ρ_1, ρ_2 be irreducible representations with characters χ_1 and χ_2. Let the associated matrix to ρ_1 be $r_{ij}(g)$, and the associated matrix to ρ_2 be $s_{ij}(g)$. Then χ_1 is given by $\mathrm{Tr}(r_{ij}(g)) = \sum r_{ii}(g)$, and $\chi_2 = \sum s_{jj}(g)$. Fixing an invariant inner product such that the representation ρ unitary, it is easy to check that $\chi_\rho(g^{-1}) = \overline{\chi_\rho(g)}$, and by Corollary 2 we thus have

$$(\chi_1, \chi_2) = \frac{1}{|G|}\sum_{g\in G} \chi_1(g)\overline{\chi_2(g)} = \frac{1}{|G|}\sum_{g\in G} \chi_1(g)\chi_2(g^{-1}) = 0.$$

□

Now we can relate the characters of a representation with the decomposition of the associated vector space.

Theorem 5.12. *Let $\rho : G \longrightarrow GL(\mathcal{V})$ be a representation with character χ, and suppose $\mathcal{V}$ decomposes into irreducible representations:*

$$\mathcal{V} = \mathcal{W}_1 \oplus \ldots \oplus \mathcal{W}_n.$$

Then, if $\mathcal{W}$ is the representation space of an irreducible representation with character ϕ, the number of $\mathcal{W}_i$ in the decomposition of $\mathcal{V}$ equivalent to $\mathcal{W}$ equals (ϕ, χ).

Proof. Let ρ_i be the representation with space $\mathcal{W}_i$, and χ_i its character. By choosing an appropriate matrix for $\rho_1 \oplus \ldots \oplus \rho_n$, we have that $\phi = \chi_1 + \ldots + \chi_n$. By the orthonormality of the characters, the terms of the sum

$$(\phi, \chi) = \sum_{i=1}^{n} (\phi_i, \chi_i)$$

are only equal to 1 if $\mathcal{W}_i \neq \mathcal{W}$, and 0 otherwise. □

Example 5.13. Recall that we for an element g in a group G define the conjugacy class $Class(g) = \{h^{-1}gh \mid h \in G\}$. It is easy to see that the characters are constant on the conjugacy classes of G, that is

$$\chi(h^{-1}gh) = \chi(g),$$

by the familiar formula $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$. This makes the characters into what we call *class functions*, functions constant on conjugacy classes. In fact, one can take Theorem 5.11 even further and show that the irreducible characters form an orthonormal basis of the space of class functions. It can also be shown that the number of conjugacy classes is equal to the dimension of the space of class functions, so it follows that the number of conjugacy classes is equal to the number of irreducible characters.

Returning to Example 5.4, denote the character of the regular representation λ by χ_λ. For $g = 1$, we have that $\chi_\lambda(1) = \mathrm{Tr}(\lambda_1) = \mathrm{Tr}(I) = |G|$, since all the basis elements are fixed. Otherwise, $\chi_\lambda(g) = \mathrm{Tr}(\lambda_g) = 0$, since there are no fixed points and all basis elements are permuted by g. From this reasoning the following theorem follows almost immediately.

Theorem 5.14. *Let G be a group with irreducible characters $\chi_1, \ldots, \chi_n$ with degrees $d_1, \ldots, d_n$. Then every irreducible representation ρ_i of G is contained in the regular representation with multiplicity equal to its degree d_i, and the degrees satisfy*

$$\sum_{i=1}^{n} d_i \chi_i(g) = \begin{cases} 0 & \text{if } g \neq 1 \\ |G| & \text{otherwise.} \end{cases}$$

In particular, it is true that

$$\sum_{i=1}^{n} d_i^2 = |G|.$$

Proof. The first statement follows from Theorem 5.12, computing

$$(\chi_\lambda, \chi_i) = \frac{1}{|G|}\sum_{g\in G}\chi_\lambda(g)\overline{\chi_i(g)} = \overline{\chi_i(1)} = d_i.$$

From this it follows that

$$\chi_\lambda = \sum_{i=1}^{n} d_i\chi_i(g),$$

so if $g = 1$, this sum is equal to $|G|$, in particular $\sum d_i^2 = |G|$, and otherwise it is 0 as desired. □

This result concludes our section on basic representation theory, necessary to give insight into a more general approach to solving the hidden subgroup problem.

§ 5.2. The Standard Method.

Now, with this final piece of the puzzle in place, we can generalize the Fourier transform, which previously was only introduced for cyclic groups.

Definition 5.15. Let G be group of order N which is finite, $f : G \to \mathbb{C}$ a function, and ρ an irreducible representation of G of dimension d_ρ. Then the *Fourier transform of f at ρ* is the linear map defined by

$$\hat{f}(\rho) = \sqrt{\frac{d_\rho}{N}}\sum_{g\in G} f(g)\rho(g). \tag{5.2.1}$$

We define the *inverse Fourier transform of $\hat{f}$* to be the linear map

$$f(g) = \sqrt{\frac{1}{N}}\sum_{\rho\in\hat{G}}\sqrt{d_\rho}\,\mathrm{Tr}\left(\hat{f}(\rho)\rho(g^{-1})\right), \tag{5.2.2}$$

where $\hat{G}$ is the set of irreducible representations of G.

For our purposes this definition of the inverse quantum Fourier transform is sufficient. However, it is worth mentioning that it is really a map between vector valued function spaces, $\hat{G}$ and $\coprod_{\rho\in\hat{G}}\mathrm{End}(\mathcal{V}_\rho)$. To check that formula (5.2.2) indeed gives the inverse, it suffices to check that it holds if $f(h) = \delta_{g,h}$, for $g, h \in G$. We then have that $\hat{f}(\rho) = \sqrt{\frac{d_\rho}{N}}\rho(g)$, and the inverse Fourier transform equals

$$\begin{aligned}\sqrt{\frac{1}{N}}\sum_{\rho\in\hat{G}}\sqrt{d_\rho}\,\mathrm{Tr}\left(\sqrt{\frac{d_\rho}{N}}\rho(g)\rho(h^{-1})\right)\\ &= \frac{1}{N}\sum_{\rho\in\hat{G}} d_\rho\,\mathrm{Tr}\left(\rho(gh^{-1})\right)\\ &= \frac{1}{N}\sum_{\rho\in\hat{G}} d_\rho\,\chi_\rho(gh^{-1}).\end{aligned}$$

By Theorem 5.14 this sum is equal to 1 if $g = h$, and 0 otherwise, so this expression is exactly $f(h)$, as desired.

Note that after a choice of basis, each irreducible representation ρ can be identified with a $d_\rho \times d_\rho$ matrix. By Theorem 5.14 we have that $|G| = \sum_{\rho\in\hat{G}} d_\rho^2$. This means that the number of bits required to store the function values that determine f is equal to the number of bits necessary to store the matrix entries of ρ. Specifically, the Fourier transform does not change the size of the register to which it is applied.

Now, relating this to the discrete Fourier transform that we saw in Definition 3.13, we note that by Example 5.10 the representations of Abelian groups are equal to the characters. Hence the general quantum Fourier transform of (5.2.1) reduces to

$$\hat{f}(\rho) = \frac{1}{\sqrt{N}} \sum_{a\in\mathbb{Z}/N\mathbb{Z}} f(a)\chi_\rho(a).$$

The second thing that it is time to generalize is the quantum procedure that we used to solve the hidden subgroup problem. As we saw, the key step in the quantum subroutine was the use of the discrete Fourier transform, and with our newly gained knowledge of the necessary theory to define the *general* Fourier transform, we can generalize this process. We refer to this as the *standard method* for solving the hidden subgroup problem.

So let G be an arbitrary group with a subgroup H hidden by the oracle function $f : G \longrightarrow X$, and let $\rho \in \hat{G}$ where $\hat{G}$ is the set of all irreducible representations of G. We associate the formal symbol ρ, i, j with the matrix of the irreducible representation ρ, and the i, j element of this matrix. Then the method can be summarized in the following steps:

1. Prepare a double register in the zero state $e_0^{\otimes|G|} \otimes e_0^{\otimes|X|}$
2. Apply the Hadamard transformation to the first register.
3. Encode the information of f by applying the oracle U_f to the second register, obtaining the state
$$\frac{1}{\sqrt{|G|}} \sum_{g\in G} e_g \otimes e_{f(g)}.$$
4. Measure the second register which encodes the function values, obtaining the value $f(c)$ for some $c \in G$. This collapses the first register onto the coset cH, resulting in the state
$$\frac{1}{\sqrt{|H|}} \sum_{h\in H} e_{ch} \otimes e_{f(ch)}.$$
Note that $f(ch) = f(c)$ for all $h \in H$.
5. Apply the quantum Fourier transform to the coset state, which yields
$$\sum_{\rho\in\hat{G}} \sqrt{\frac{d_\rho}{|G||H|}} \sum_{i,j} \sum_{h\in H} \rho(ch)_{ij} e_{\rho,i,j}$$
in the first register.

6. Measuring this register of subspaces defined by the elements of $\hat{G}$, we obtain an irreducible representatiton ρ, which gives information about H and its generators.
7. Use the information of step 6 to find a generating set of H, identifying the hidden subgroup.

Hopefully the reader recognize the similarities between this standard method and the steps of the quantum subroutine seen in Section 4.5. As we saw in Theorem 4.2, when uncovering the hidden subgroup generated by a single element, the information given by the quantum subroutine had a high probability of being useful, and resulting in a situation where the generator could be recovered during post processing. So the information obtained in step 6 was a state e_c, where $c \in \mathbb{Z}/2^n\mathbb{Z}$. This value c corresponds to one specific irreducible character χ_c, or, equivalently a representation ρ_c. From this information, in this case without really needing any of the more advanced techniques discussed in this section, we could extract c, and by the theory of continued fractions obtain r, the generator of the hidden subgroup.

In the next section we will take a look at yet another problem that can be reduced to the hidden subgroup problem, and possibly be solved using this standard method.

SECTION 6
The Graph Isomorphism Problem

As we saw in previous sections, the Abelian hidden subgroup problem could be used to efficiently solve the factoring problem using the discrete quantum Fourier transform. If we instead consider the hidden subgroup problem for a non-Abelian group such as the symmetric group S_n, we obtain the *graph isomorphism problem* as a special case.

§ 6.1. Graph Isomorphism and Automorphism Groups. Let $G = (V, E)$ denote a graph, with nonzero vertex set V and edge set $E \subseteq V \times V$. Throughout this section, we assume all graphs to be simple and undirected, so that $(v, v) \notin E$ and (u, v) is identified with (v, u), for all $u, v \in V$. We also let $Graph(V)$ denote the set of graphs on the vertex set V, and $Sym(V)$ be the symmetric group on V.

Definition 6.1. The graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are said to be isomorphic if there exists a bijection $\pi : V_1 \longrightarrow V_2$ such that $(u, v) \in E_1$ if and only if $(\pi(u), \pi(v)) \in E_2$.

Definition 6.2. Let G_1 and G_2 be two graphs. The problem of determining whether G_1 and G_2 are isomorphic in polynomial time is called the *graph isomorphism problem.*

As mentioned, this problem can also be translated into the hidden subgroup problem which we will see shortly after introducing one more definition and a subsequent result.

Definition 6.3. Let $G = (V, E)$ be a graph. The *automorphism group* $\mathrm{Aut}(G)$ of G, is defined by

$$\mathrm{Aut}(G) = \{\pi \in \mathrm{Sym}(V) \mid \forall u, v \in V;\ (u, v) \in E \iff (\pi(u), \pi(v)) \in E\}.$$

So the automorphism group of G consist of all permutations that respect the adjacency of edges in G.

Lemma 6.4. *Let $G = (V, E)$ be a graph, and $f : Sym(V) \longrightarrow Graph(V)$ be the function taking $\pi \in Sym(V)$ to the graph $\pi(G) = (V, (\pi \times \pi)(E))$. Then f hides the subgroup $H = Aut(G)$ of G.*

Proof. The group $\mathrm{Sym}(V)$ acts on $Graph(G)$. Given a graph G, the function f is the orbit map of G, taking G to its orbit under the group action. Then it is clear that the stabilizer of G is the automorphism group of G, so that the condition $f(\pi_1) = f(\pi_2)$ is equivalent to $\pi_1 H = \pi_2 H$, and f hides H. □

This result seems to imply that if we could find the generators of $\mathrm{Aut}(G)$, where $G = G_1 \amalg G_2$, we could solve the graph isomorphism problem via the hidden subgroup problem framework, which as we will now see, is indeed the case.

Theorem 6.5. *Let $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, be two connected graphs on n vertices, and $G = G_1 \amalg G_2 = (V, E)$ be the disconnected graph on the disjoint union of the sets V_1 and V_2. If the symmetric hidden subgroup problem can be solved in polynomial time, then the graph isomorphism problem of G_1 and G_2 is polynomially reducible to the hidden subgroup problem.*

Proof. By finding the generators of the automorphism group $\mathrm{Aut}(G)$, we can determine the existence of an element $\pi \in \mathrm{Aut}(G)$ that exchanges the vertices of G_1 and G_2, that is, such that $\pi(v) = u$ for $v \in V_1$ and $u \in V_2$. If such a "flip" exists, the graphs G_1 and G_2 are isomorphic, and we have solved the graph isomorphism problem.

By assumption, the hidden subgroup problem can be solved in polynomial time, so it is plausible that there are no more than $poly(n)$ generators of $\mathrm{Aut}(G)$. Thus, by simply checking each generator on each vertex of V_1 and V_2, we can determine if a flip exist in the automorphism group in $|V_1||V_2|poly(n) = n^2 poly(n)$ steps. □

In fact, it is not unreasonable to think that the assumption of the theorem is true, which would give us hope of solving the graph isomorphism problem by use of the quantum methods discussed in this text. By an unpublished result of Von Neumann we have the following Theorem, a sketched proof can be found in [CST89].

Theorem 6.6 (Von Neumann). *Let $G \subseteq S_n$ be a group and $d(G)$ be the number of generators of G. Then*

$$d(G) \leq max\left(2, \left[\frac{n}{2}\right]\right).$$

This gives us a polynomial bound on the number of generators of $\mathrm{Aut}(G)$, so a solution to the symmetric hidden subgroup problem would yield a solution to the graph isomorphism problem in polynomial time. However, as a solution to the former has not yet been discovered, this is still an area of active research, which will be further covered in Section 6.3.

§ 6.2. Representations of S_n . The last section introduced the relation between the symmetric group S_n and the hidden subgroup problem. As we know, the standard method presented in Section 5.2 relies on the representations of the group in question. We now present some of the interesting and relevant properties of the representations of the symmetric group. This section roughly follows [FH91], and details for the interested reader can be found in Chapter 4 of the book.

Consider the group S_n, we distinguish three particular representations of this group: the trivial, the alternating, and the standard representation. The first of these appears quite naturally as the representation $\rho : S_n \longrightarrow \mathbb{C}^*$ satisfying $\rho(\pi) = I$ for every $\pi \in S_n$.

The *alternating representation* can be realized if we consider the representation taking a permutation π to $\mathrm{sgn}(\pi)I$.

Finally, we can also consider the action of S_n on the standard basis $\{e_1, e_2, \dots, e_n\}$ of $\mathbb{C}^n$, by permuting the indices of the basis elements. Taking the subspace spanned by the vector $e_1 + e_2 + \ldots + e_n$ we obtain an invariant subspace since the action of S_n merely moves around the summands. This space then has a complementary subspace

$$V = \{(a_1, a_2, \dots, a_n) \in \mathbb{C}^n \mid a_1 + a_2 + \ldots + a_n = 0\},$$

which is also invariant, and thus irreducible. This then defines what is called the *standard representation* of S_n.

From abstract algebra we know that the cycle types of elements of S_n determine the conjugacy classes, and since the cycle types correspond to the partitions of n, the number of conjugacy classes is equal to the number of partitions of n. This observation leads to a particularly nice situation in the case of S_3.

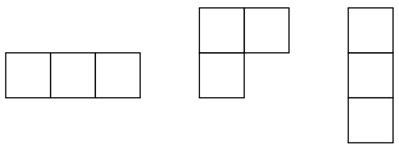
Example 6.7. Consider S_3, a symmetric group with three cycle types, or equivalently, three conjugacy classes. By Example 5.13 this means that S_3 has exactly three irreducible representations. The trivial and the alternating representation are both one-dimensional, so they must be irreducible, because they cannot contain any nonzero (proper) subspaces. Since there can only be one more irreducible representation of S_3, it has to be the standard representation which is 2-dimensional.

Now, for arbitrary n, we have other tools at our disposal to decompose the regular representation of the symmetric group into its irreducible representations.

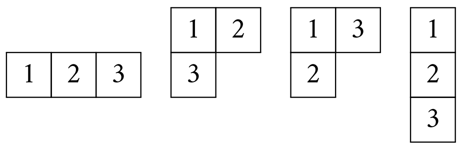
Given a partition of $n = \lambda_1 + \lambda_2 + \ldots + \lambda_k$, where the λs form a decreasing sequence, we define an associated *Young diagram*. It is a diagram consisting of n left-aligned boxes, with λ_i boxes in the ith row. In this way, the set of all Young diagrams of a given n is in bijection to the set of irreducible representations of S_n.

So we can determine the number of irreducible representations as the number of partitions of n, but we can also determine the order of each irreducible by introducing *Young tableaux*. Given a Young diagram, we assign an integer from $\{1, 2, \ldots, n\}$ to each box, such that the numbers are increasing from left to right and from top to bottom. There can be multiple possible such numberings, and it can be shown that the number of Young tableaux is equal to the degree of the corresponding representation.

Example 6.8. The Young diagrams for $n = 3$ looks as follows:



where the first corresponds to the trivial representation, the second to the standard representation, and the third to the alternating representation. It is simple to see that the Young tableaux in this case are:



which gives us the degree of each representation. We have that $1^2 + 2^2 + 1^2 = 6 = |S_3|$, and again we see the relation $|G| = \sum_{\rho \in \hat{G}} d_\rho^2$ appear.

When attempting to solve the symmetric hidden subgroup problem, we proceed according to the standard method presented in Section 5.2. This includes applying the Fourier transform to the symmetric group. So as a conclusion to this section on representations of symmetric groups, it might be interesting to see what such a transformation might look like when the information of the irreducible representations is to be stored in a real computer. We continue our discussion of S_3 from before:

Example 6.9. Consider the irreducible representations of S_3 we saw in Example 6.7. The trivial and alternating representations are one-dimensional, the former maps all permutations to (1), and the latter maps all transpositions to (-1) and all other elements to (1). The standard representation is more interesting. Let $\{e_1, e_2, e_2\}$ be the standard basis of $\mathbb{C}^3$, and fix the basis $\{e_1 - e_2, e_2 - e_3\}$ of the representation space V. From this we can derive the orthonormal basis $\frac{1}{\sqrt{6}}\{(\sqrt{3}, -\sqrt{3}, 0), (1, 1, -2)\}$. Then the matrices of the standard representation are:

$$\text{id} \mapsto I \qquad (1\ 2) \mapsto \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \qquad (1\ 3) \mapsto \begin{pmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & -\frac{1}{2} \end{pmatrix}$$

$$(2\ 3) \mapsto \begin{pmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{pmatrix} \qquad (1\ 2\ 3) \mapsto \begin{pmatrix} -\frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{pmatrix} \qquad (1\ 3\ 2) \mapsto \begin{pmatrix} -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & -\frac{1}{2} \end{pmatrix}$$

Fixing the order $S_3 = \{\text{id}, (1\ 2), (1\ 3), (2\ 3), (1\ 2\ 3), (1\ 3\ 2)\}$ of elements of S_3, we then get the following matrix representation:

$$Q_{S_3} = \frac{1}{\sqrt{6}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & -1 & 1 & 1 \\ \sqrt{2} & -\sqrt{2} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & 0 & -\frac{\sqrt{3}}{\sqrt{2}} & \frac{\sqrt{3}}{\sqrt{2}} & -\frac{\sqrt{3}}{\sqrt{2}} & \frac{\sqrt{3}}{\sqrt{2}} \\ 0 & 0 & -\frac{\sqrt{3}}{\sqrt{2}} & \frac{\sqrt{3}}{\sqrt{2}} & \frac{\sqrt{3}}{\sqrt{2}} & -\frac{\sqrt{3}}{\sqrt{2}} \\ \sqrt{2} & \sqrt{2} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

§ 6.3. Current and Future Developments. If we want to follow the general hidden subgroup-method given in Section 5, we still need to find a way to effectively compute the quantum Fourier transform over a symmetric group. In 1997, Robert Beals [Bea97] first constructed an efficient quantum Fourier transform over this family of groups. Since then, new results have been presented. An improved algorithm was given recently by Kawano and Sekigawa [KS16]. The complexity of their algorithm is $\mathcal{O}(n^3 \log n)$ on S_n, which is an improvement over the previously fastest algorithm by Moore, Rockmore and Russel [MR06] with complexity $\mathcal{O}(n^4 \log n)$.

To solve the graph isomorphism problem efficiently using the quantum Fourier transform it is also crucial that we can perform the post processing effectively, meaning that we need to be able to determine the hidden automorphism group and its generators from the representations returned by the quantum subroutine. In a manuscript originally released in 2015, László Babai [Bab17] claimed to have constructed a an algorithm solving the graph isomorphism problem. The suggested algorithm would run slower than polynomial algorithms but faster than sub-exponential algorithms. A mistake in the proof was discovered in 2017 by Harald Helfgott, but was corrected by Babai shortly after. The algorithm still remains unpublished, so a final confirmation of this result is yet to come.

References

[AAB+19] F. Arute, K. Arya, R. Babbush, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505 – 510, 2019.

[Apo76] Tom M. Apostol. *Introduction to Analytic Number Theory*. Springer-Verlag, 1976.

[Bab17] László Babai. Graph isomorphism. `http://people.cs.uchicago.edu/~laci/update.html`, 2017. Accessed: 2020-07-06.

[BBDR04] Michael Batty, Samuel L. Braunstein, Andrew J. Duncan, and Sarah Rees. Quantum algorithms in group theory, 2004.

[Bea97] Robert Beals. Quantum computation of fourier transforms over symmetric groups. *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 48 – 53, 1997.

[CLRS09] Thomas H. Cormen, Charles Eric. Leiserson, Ronald L. Rivest, and Clifford. Stein. *Introduction to Algorithms*. MIT Press, 3rd ed. edition, 2009.

[CST89] P.J. Cameron, R.G. Solomon, and A. Turull. Chains of subgroups in symmetric groups. *J. Algebra*, pages 235–265, 1989.

[FH91] William Fulton and Joe Harris. *Representation theory : a first course*. 1991.

[Hal87] Paul R. Halmos. *Finite-Dimensional Vector Spaces*. Springer-Verlag, 1987.

[HW08] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, 2008.

[KS16] Yasuhito Kawano and Hiroshi Sekigawa. Quantum fourier transform over symmetric groups — improved result. *Journal of Symbolic Computation*, 75:219 – 243, 2016.

[MRR06] Cristopher Moore, Daniel Rockmore, and Alexander Russell. Generic quantum fourier transforms. *ACM Trans. Algorithms*, 2:707–723, 2006.

[PGMG19] Edwin Pednault, John Gunnels, Dimitri Maslov, and Jay Gambetta. On quantum supremacy. `https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/`, 2019. Accessed: 2020-07-07.

[Ser77] Jean-Pierre Serre. *Linear Representations of Finite Groups*. Springer, 1977.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.