THE HIDDEN SUBGROUP PROBLEM - REVIEW AND OPEN PROBLEMS

CHRIS LOMONT, CYBERNET

Abstract. An overview of quantum computing and in particular the Hidden Subgroup Problem are presented from a mathematical viewpoint. Detailed proofs are supplied for many important results from the literature, and notation is unified, making it easier to absorb the background necessary to begin research on the Hidden Subgroup Problem. Proofs are provided which give very concrete algorithms and bounds for the finite abelian case with little outside references, and future directions are provided for the nonabelian case. This summary is current as of October 2004.

Contents

Date: Oct 2004.
clomont@cybernet.com, clomont@math.purdue.edu.

1. Introduction

The main purpose of this paper is to give a self contained explanation of the Hidden Subgroup Problem in quantum computing. A second goal is to bring the interested reader to the forefront of research in the area, so that a wider audience can attack the problems. The final goal is to present this at a level accessible to graduate students in math, physics, and computer science. Prerequisites are some abstract algebra, linear algebra, and an understanding of (classical) computation. However almost any mathematically inclined reader should be able to learn something from this presentation.

1.1. **Importance.** The importance of the Hidden Subgroup Problem (from now on labelled the HSP) is that it encompasses most of the quantum algorithms found so far that are exponentially faster than their classical counterparts. Research in this area is centered on extending the families of groups for which the HSP can be efficiently solved, which may improve other classically inefficient algorithms, such as determining graph isomorphism or finding the shortest vector in a lattice. Finally, there are many group theoretic algorithms that are more efficient on a quantum computer, such as finding the order of a finite group given a set of generators.

1.2. **History.** In 1994, Shor [114], building on the work of Deutsch [37] and Simon [118], found a quantum algorithm that could factor integers exponentially faster than any known classical method, and opened the floodgates on quantum computing research. Efficient integer factoring breaks the ubiquitous RSA cryptosystem. Shor also gave an algorithm solving the Discrete Log Problem (DLP), which is used in several other cryptosystems. Kitaev [77] noted that these algorithms as well as others fit in a framework of finding subgroup generators from a group using a function that "hides" the subgroup, and thus the Hidden Subgroup Problem was born. For more history, the book by Chuang and Neilsen [29] contains a wealth of information, as well as the quantum physics archives at http://arxiv.org/archive/quant-ph.

1.3. **Notation.** Here we fix some notation used throughout this paper. All logs are base 2 unless otherwise specified. $\mathbb{C}$ denotes the field of complex numbers. $\mathbb{Z}$ is the ring of integers, and for a positive integer N we let $\mathbb{Z}_N$ denote the ring of integers mod N. For each integer $N > 0$ let $\omega_N = \exp(2\pi i/N)$, a principal N^{th} root of unity. Quantum mechanics specific notation is in section 2 and appendix C.

1.4. **Layout.** The layout of this paper is as follows. Section 2 covers the necessary quantum mechanics and notation used therein. It also introduces a quantum computing model well suited to present the rest of the topics in this paper. Section 3 explains the algorithm solving the abelian case of the HSP efficiently, describing in detail the mathematics making it work. Section 4 generalizes the examples from section 3 to give the a more general form of the HSP, suitable for any finite group. Section 5 covers recent results, and what is currently known about the HSP, as well as quantum algorithms for other group related problems. Section 6 concludes. Much of the background and details are included in numerous appendices, giving details on topics such as the necessary background for the graph isomorphism reduction, generating groups from random samples, number theory results, etc.

2. Quantum Computing Model

2.1. **The Rules and Math of Quantum Mechanics.** Here we define the rules of quantum mechanics (from a mathematical perspective). Details can be seen in Appendix C.

First some notation used in quantum mechanics. We define the following symbols:

$|\psi\rangle$ represents a column vector in some complex Hilbert space V, of finite dimension for this paper. For this section, let this dimension be N. Quantum mechanics forces us to use an orthonormal basis for V, so we fix the orthonormal *standard basis* $\mathcal{B} = \{|0\rangle, |1\rangle, \ldots, |N-1\rangle\}$. Then $\langle\psi|$ denotes the conjugate transpose row vector, often viewed as the dual to $|\psi\rangle$ with respect to $\mathcal{B}$. For example, we compute as follows:

If $|\psi\rangle = \sum_i a_i|i\rangle$, then $\langle\psi| = \sum_i a_i^*\langle i|$, where $*$ denotes complex conjugation. $\langle i|\ |j\rangle$, written $\langle i|j\rangle$, equals 1 if $i = j$, otherwise $\langle i|j\rangle$ equals 0. A basis for linear operators on V can be written as a $\mathbb{C}$-linear combination of the operators $|i\rangle\langle j|$, which is the matrix with a 1 in the (i, j) entry, and 0's elsewhere. Thus any linear operator A on V in the basis $\mathcal{B}$ can be written in the form $A = \sum_{i,j} a_{i,j}|i\rangle\langle j|$, which is the matrix with the value $a_{i,j}$ in the i, j entry, and acting on the left of a column vector $|\psi\rangle$. $\langle\psi|A|\phi\rangle$ is the inner product of ψ and $A|\phi\rangle$. Later the basis will often be indexed with elements from a group G, viewed as fixing an orthonormal basis and an injection mapping elements of G to this basis.

2.1.1. *The Postulates of Quantum Mechanics.* Now on to the physical content of quantum mechanics, abstracted to a mathematical formalism. The content of quantum mechanics can be summarized by 4 postulates, which we take as the definition of quantum mechanics. They are:[1]

Quantum Mechanics Postulate 1: State Space: Associated to an isolated physical system is a complex vector space with inner product (a Hilbert space) known as the *state space* of the system. The system is completely described by its *state vector*, which is a unit vector in the system's state space.

[1]Postulates are taken verbatim from Neilsen and Chuang [29].

Quantum Mechanics Postulate 2: State Evolution: The evolution of a *closed* quantum system is described by a *unitary transformation*[2]. That is, the state of a system $|\psi\rangle$ at time t_1 is related to the state $|\psi'\rangle$ at time t_2 by a unitary operator U which depends only on the times t_1 and t_2,

$$|\psi\rangle = U|\psi'\rangle \tag{1}$$

Quantum Mechanics Postulate 3: State Measurement: Quantum measurements are described by a collection $\{M_m\}$ of *measurement operators*. These are operators acting on the state space of a system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the system is $|\psi\rangle$ immediately before the measurement, then the probability that result m occurs is given by

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle \tag{2}$$

and the state of the system after the measurement is

$$\frac{M_m|\psi\rangle}{\sqrt{p(m)}} \tag{3}$$

The measurement operators satisfy the *completeness equation*

$$\sum_m M_m^\dagger M_m = I \tag{4}$$

Quantum Mechanics Postulate 4: State Combining: The state space of a composite physical system is the tensor product of the state spaces of the component systems. Moreover, if we have systems numbered 1 through n, and system number j is prepared in the state $|\psi_j\rangle$, then the joint state of the total system is $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \ldots |\psi_n\rangle$.

We will explain briefly how these postulates are used in practice for quantum computing.

2.1.2. *Qubits and Operators.* Analogous to the bit being the basic block in classical computing, the qubit is the basic building block in quantum computing. Formally we define

Definition 2.1 (Qubit)**.** *A* **qubit** *(or quantum-bit) is a unit vector in* $\mathbb{C}^2$*. We fix an orthonormal basis of column vectors denoted as* $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ *and* $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$*, corresponding to classical bits 0 and 1.*

Definition 2.2 (State vector)**.** *The* **state** *of a quantum system is a (column) vector in some vector space, written* $|\psi\rangle$*.*

By postulate 4, we can combine single qubits as follows.

[2]Recall a unitary operator U satisfies $UU^\dagger = I = U^\dagger U$ where $\dagger$ is conjugate transpose. In particular, unitary operators are invertible, implying quantum computation is *reversible*, which differs significantly from classical computing.

2.1.3. *Qubits Galore.* Similar to concatenating n classical bits to "bitstrings", we concatenate qubits to get larger systems. Two qubits form a space spanned by the four vectors

$$|0\rangle \otimes |0\rangle, \quad |0\rangle \otimes |1\rangle, \quad |1\rangle \otimes |0\rangle, \quad \text{and } |1\rangle \otimes |1\rangle \tag{5}$$

where the tensor product is the usual vector space tensor. See Appendix C for details. Shorthand for the above expressions is

$$|00\rangle, \quad |01\rangle, \quad |10\rangle, \quad \text{and } |11\rangle \tag{6}$$

Now we can check the second basis element (dictionary ordering)

$$|01\rangle = |0\rangle \otimes |1\rangle \tag{7}$$

$$= \tbinom{1}{0} \otimes \tbinom{0}{1} \tag{8}$$

$$= \begin{pmatrix} 1\binom{0}{1} \\ 0\binom{0}{1} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \tag{9}$$

and we get the second usual basis element of $\mathbb{C}^4$. This works in general; that is, the vector corresponding to the state $|n\rangle$ where n is a binary number, is the $(n+1)^{\text{th}}$ standard basis element. We frequently use decimal shorthand: $|32\rangle$ is the 33rd standard basis vector in some space which would be clear from context.

Thus the the state of an n-qubit system is a unit vector in $\mathbb{C}^{2^n}$. Note that the state of n classical bits is described by n elements each either 0 or 1, while the state of n qubits requires 2^n complex numbers to describe. Thus it seems qubits contain much more "information" than classical bits. Unfortunately we cannot retrieve all this "information" from the state; we are limited by quantum mechanics due to the fact that measuring the state destroys information.

2.1.4. *Measurement.* The final operation we need to understand about qubits is measurement, the process of getting information out of a quantum state. There are several equivalent ways to think about it. We will cover the easiest to understand, intuitively and mathematically. However, to gain precise control over measurements, often one has to resort to an equivalent, yet more complicated, measurement framework[3], which we do not discuss here. See Nielsen and Chuang [29, Ch. 2].

We will do our measurements in the *computational* basis $\{|0\rangle, |1\rangle, \ldots, |2^n - 1\rangle\}$ over an n-qubit system. Suppose we have the state $|\psi\rangle = \sum_{j=0}^{2^n-1} a_j |j\rangle$, which is a unit vector in $\mathbb{C}^{2^n}$. Measuring in the computational basis has the following effect: it returns the state $|j\rangle$ with probability $p_j = |a_j|^2$, and after the measurement, the state becomes $|\psi'\rangle = |j\rangle$. Thus measuring "collapses" the waveform, returning a state with probability the square of its coefficient (*amplitude*), and the resulting state is the one returned by the measurement. Thus from a given state, we return one answer depending on the basis we measure, and destroy all other information about the state.

Finally we note that cascaded measurements (one after the other) can always be replaced by a single measurement.

[3]This is the "Positive Operator-Valued Measure" (POVM) formalism.

2.1.5. *The No Cloning Theorem.* As an example of the using above postulates, we prove an important and surprising theorem:

Theorem 2.3. ***The No Cloning Theorem.*** *It is impossible to build a machine that can clone any given quantum state.*

This is in stark contrast to the classical case, where we copy information all the time. It is the tip of the iceberg for the differences between quantum and classical computing.

Proof. Suppose we have a machine with two slots: A for the quantum state $|\psi\rangle$ to be cloned, and B in some fixed initial state $|s\rangle$, and the machine makes a copy of the quantum state A. By the rules of quantum mechanics, the evolution U is unitary, so we have

$$|\psi\rangle \otimes |s\rangle \xrightarrow{U} |\psi\rangle \otimes |\psi\rangle \tag{10}$$

Now suppose we have two states we wish to clone, $|\psi\rangle$ and $|\varphi\rangle$, giving

$$\begin{aligned} U\left(|\psi\rangle \otimes |s\rangle\right) &= |\psi\rangle \otimes |\psi\rangle \\ U\left(|\varphi\rangle \otimes |s\rangle\right) &= |\varphi\rangle \otimes |\varphi\rangle \end{aligned}$$

Taking the inner product of these two equations, and using $U^\dagger U = I$:

$$\begin{aligned} \left(\langle\varphi| \otimes \langle s|\right) U^\dagger U \left(|\psi\rangle \otimes |s\rangle\right) &= \left(\langle\varphi| \otimes \langle\varphi|\right)\left(|\psi\rangle \otimes |\psi\rangle\right) \\ \langle\varphi|\psi\rangle\langle s|s\rangle &= \langle\varphi|\psi\rangle\langle\varphi|\psi\rangle \\ \langle\varphi|\psi\rangle &= \left(\langle\varphi|\psi\rangle\right)^2 \end{aligned}$$

This has solutions if and only if $\langle\varphi|\psi\rangle$ is 0 or 1, so cloning cannot be done for general states.[4] □

2.2. Efficient Quantum Computation.

2.2.1. *Quantum Computing.* Quantum sates are transformed by applying unitary operators to the state. So where classical computing can be viewed as applying transforms to n-bit systems, quantum computation proceeds by constructing an n-qubit machine, applying unitary operators to the state until some desired state is found, and then measuring the result. This paper will avoid the physical construction of such machines, and focus on the unitary transformations that seem likely to be physically realizable, and the computational outcomes of such systems. Again, for an introduction to the physical issues, see [29, Ch. 7] and the references therein.

2.2.2. *Circuit Model.* Similar to being able to construct any classical circuit with NAND gates, there are finite [5] sets of quantum gates that allow the construction of any unitary operator to a desired precision. Kitaev [78] shows that these approximations can be done with minimal overhead, allowing quantum computation to be modelled with simple "quantum circuits". A final note on quantum circuits is that Deutsch's Quantum Turing Machine [37] and the circuit model used more recently were shown equivalent by Yao [128]. We will use a few quantum gates that operate on 1,2, or 3 qubits at a time, defined later. The intuitive description is that

[4] There is a lot of research on precisely what can be cloned, how to approximate cloning, and what other limitations there are to duplicating quantum states.

[5] There are many ways to choose them. See for example [8].

quantum computations are built of quantum circuits, which are composed of quantum gates, and each quantum gate operates on only a few qubits at a time. This statement mirrors the classical one with "quantum" removed and qubits replaced with bits.

2.2.3. *Quantum Circuit Size.* In loose terms, efficient classical computations are done on small circuits, in the sense that as the problem size grows, the size of the circuit required to solve the problem grows at a certain rate, usually bounded polynomially in the size of the problem. We want to make this precise in the quantum context.

The following is just a mathematically precise way to say our "elementary operations" only operate on a few qubits at a time, which is desirable since it makes quantum computation physically plausible. Some definitions:

Definition 2.4. *Given a 2^n-dimensional vector space V with basis $\mathcal{B}$, and a $2^m \times 2^m$ matrix U with $m \leq n$, an* **expansion of U relative to $\mathcal{B}$** *is any matrix of the form $G(U \otimes I_{2^{n-m}})G^{-1}$ where G permutes the basis, and I_k is the $k \times k$ identity matrix.*

This just says each expansion of U operates on m of the n qubits in a n-qubit machine. In general m will be small, n will vary, and we will build computations by composing these operators.

Definition 2.5. *Given a 2^n-dimensional vector space V, an orthonormal basis $\mathcal{B}$ of V, and a finite set $\mathcal{U} = \{U_1, U_2, \ldots U_k\}$ of unitary matrices of dimensions dividing 2^n, then the set of* **elementary operations relative to** *$(\mathcal{B},\mathcal{U})$ consists of all expansions of the U_i relative to $\mathcal{B}$.*

This just allows us to consider all operations on any subset of n qubits generated from our initial set of "elementary operations". Note U unitary and $\mathcal{B}$ orthonormal implies expansions of U relative to $\mathcal{B}$ are unitary.

For our use V will be the state space of a quantum system, clear from context, and $\mathcal{B}$ will be the standard orthonormal basis of V. We fix a specific generating set $\mathcal{U}_\tau = \{H, CNOT, CCNOT, P\}$ relative to such a fixed basis to be the matrices

$$H = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \text{ the Hadamard matrix} \tag{11}$$

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \text{ the controlled NOT} \tag{12}$$

$$CCNOT = (a_{ij}) \text{ with } a_{ii} = 1, i = 1, ..., 6, a_{87} = a_{78} = 1, \text{ the rest } = 0, \text{ the controlled controlled NOT} \tag{13}$$

$$P = \begin{pmatrix} e^{i\frac{\theta}{2}} & 0 \\ 0 & e^{-i\frac{\theta}{2}} \end{pmatrix} \text{ the phase matrix, where } \cos\theta = \frac{3}{5}. \tag{14}$$

For any $n > 2$ and using the standard basis $\mathcal{B}$ defined earlier, the elementary operations from this set of 4 matrices generates a group dense in $U(2^n)$, the space of legal quantum operations on an n-qubit machine[6]. So from now on one can

[6]From chapter 4 exercises in [29].

assume these 4 matrices and associated elementary operations are the legal set of elementary operations on any n-qubit machine. The definitive paper on elementary gates for quantum computing is [8].

Definition 2.6. *A* **quantum circuit** *is a unitary matrix built from composing elementary operations from* $\mathcal{U}_\tau$

Now mathematically, quantum computing becomes the following. We have an initial state $|0\rangle$ in the n-qubit space $\mathbb{C}^{2^n}$. Applying unitary transformations that are products of the elementary transformations, we want to obtain a quantum state (unit vector $|\psi\rangle$) that, when measured, has a high probability of returning some useful answer. We want to know how "efficient" such transformation are. We restrict legal quantum operations to those obtained from the elementary operations from some finite set, such as $\mathcal{U}_\tau$.

Definition 2.7. *The* **size** *of a quantum circuit will be the minimal number of elementary operations composed to obtain it.*

This gives us a way to measure the complexity of a quantum operation. From here on we can assume all quantum operation complexities are measured against our set of elementary operations coming from $\mathcal{U}_\tau$ and a corresponding V and $\mathcal{B}$ taken from context.

Often it is possible to rearrange the elementary operations and obtain the same quantum circuit. For example if adjacent operations affect disjoint sets of qubits, these two operations can be swapped obtaining the same circuit (the matrices commute). Similar to parallelizing classical circuits, this reordering allows us to partition the sequence of elementary operations into ordered lists of operations, where within each list a qubit is affected by at most one operation. This leads to the notion of depth:

Definition 2.8. *The* **depth** *of a quantum circuit is the minimal length of a partition of the ordered elementary operations composing the circuit into ordered lists where each qubit is affected at most once per list.*

As a result, we always have **depth**$\leq$**size**.

To parallel the quantum to classical terminology, we sometimes call a state (or part of a state) a quantum register. Physically a quantum state is basically constructed using n particles which can be either of two states 0 or 1 when measured. If we take a subset of these particles, and operate on them, it is convenient to call this subset a register.

Definition 2.9. *A* **register** *in a quantum computer is a subset of the total set of qubits. We often write* $|a\rangle|b\rangle$ *to denote that the first register is in state* $|a\rangle$ *and the second in state* $|b\rangle$*, the number of qubits in each set being understood from context.*

2.2.4. *Efficient Quantum Computation.* Most of this paper is concerned with *efficient* quantum computation. Sometimes this has two components: needing an efficient quantum process, and an efficient classical computing method to post-process the data output from the quantum process to obtain the desired answer. We will see these two are (often) separate issues.

Given a problem to solve on a quantum computer, we need a way to represent the problem as a quantum state, a unitary operation U built from elementary

operations to convert this quantum state to a final state, and a way to process the final state to obtain the desired answer. Although a precise definition of "efficient" takes us too far afield, we will make it precise in special cases throughout this paper. The general idea is that as the "size" of the input grows (the number of qubits required to represent the problem, say), the size of the necessary quantum operator U should grow polynomially in the size of the input.

An example: suppose we want to determine the order of finite abelian groups given a generating set for each one. Given a group $|G|$, we can represent each element using roughly $\log|G|$ qubits. To call a quantum algorithm efficient for this problem would mean the size of the quantum circuit computing the order of G should be of size polynomial in $\log|G|$, as G varies throughout the *family* of finite abelian groups.

As a final technical point, we require what is called a "uniform class of algorithms," meaning that, for a problem of size n, there is a Turing machine that given n, can produce the circuit description in number of steps equal to a polynomial in n. This ensures that we can (in theory) construct an explicit machine to solve each problem in time polynomial in the size of the problem.

For more information on quantum complexity, see [18, 31].

2.2.5. *A Note on Probabilistic Algorithms.* Quantum computers are probabilistic, meaning that algorithms tend to be of the form "Problem A is solved with probability 80%." For those used to thinking that algorithms solve problems with certainty (such as algorithms encountered in a first algorithms class), note that probabilistic algorithms suffice in practice. We just run the experiment a few times, and take the majority result. This returns the correct answer with probability exponentially close to 1 in the number of trials. Precisely we use the following theorem:

Theorem 2.10 (The Chernoff Bound)**.** *Suppose $X_1, X_2, \ldots, X_n$ are independent and identically distributed random variables, each taking the value 1 with probability $1/2+\epsilon$ and 0 with probability $1/2-\epsilon$. Then*

$$p\left(\sum_{i=1}^{n} X_i \leq \frac{n}{2}\right) \leq e^{-2\epsilon^2 n}. \tag{15}$$

Thus the majority is wrong very rarely. For example, we will make most algorithms succeed with probability 3/4, so our $\epsilon = 1/4$. Although it sounds like a lot, taking 400 repetitions of the algorithm causes our error to drop below 10^{-20}, at which point it is more likely our computer fails than the algorithm fails. And since the algorithms we are considering are usually exponentially faster than classical ones, there is still a net gain in performance. If we do 1000 runs, our error drops below 10^{-55}, at which point it is probably more likely you'll get hit by lightning while reading this sentence than the algorithm itself will fail. For completeness, here is a proof of the Chernoff Bound.

Proof. Consider a sequence $(x_1, x_2, \ldots, x_n)$ containing at most $n/2$ ones. The probability of such a sequence is maximized when it contains $\lfloor n/2 \rfloor$ ones, so

$$p\left(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n\right) \leq \left(\frac{1}{2}-\epsilon\right)^{\frac{n}{2}} \left(\frac{1}{2}+\epsilon\right)^{\frac{n}{2}} \tag{16}$$

$$= \frac{(1-4\epsilon^2)^{\frac{n}{2}}}{2^n}. \tag{17}$$

There can be at most 2^n such sequences, so

$$p\left(\sum_{i=1}^{n} X_i \le \frac{n}{2}\right) \le 2^n \times \frac{(1-4\epsilon^2)^{\frac{n}{2}}}{2^n} = (1-4\epsilon^2)^{\frac{n}{2}}. \tag{18}$$

From calculus, $1 - x \le \exp(-x)$, so

$$p\left(\sum_{i=1}^{n} X_i \le \frac{n}{2}\right) \le e^{-4\epsilon^2 n/2} = e^{-2\epsilon^2 n} \tag{19}$$

□

3. The Abelian Hidden Subgroup Problem

We will detail the Hidden Subgroup Problem (HSP), starting with some illustrative and historically earlier examples, before covering the most general cases and research problems. The simplest groups considered are the finite cyclic groups, followed by finite abelian groups. Kitaev [77] examines a similar problem over finitely generated abelian groups, but we will not cover that here. The finite abelian case was first used to spectacular effect by Shor [114] and Simon [118]. The HSP for finite nonabelian groups is currently researched for the reasons given in sections 4 and 5.

Related to the HSP over finite groups is the Abelian Stabilizer Problem, in Kitaev [77].

3.1. Definition of the Hidden Subgroup Problem.

In order to set the stage for the rest of the paper, we make a general definition of the Hidden Subgroup Problem, which we will abbreviate HSP for the rest of this paper, and then attempt to determine for which groups G and subgroups H we can solve the HSP efficiently. We will also discuss partial results on groups for which efficient HSP algorithms are not known.

Definition 3.1 (Separates cosets)**.** *Given a group G, a subgroup $H \le G$, and a set X, we say a function $f : G \to X$ **separates cosets** of H if for all $g_1, g_2 \in G$, $f(g_1) = f(g_2)$ if and only if $g_1 H = g_2 H$.*

Definition 3.2 (The Hidden Subgroup Problem)**.** *Let G be a group, X a finite set, and $f : G \to X$ a function such that there exists a subgroup $H < G$ for which f separates cosets of H. Using information gained from evaluations of f, determine a generating set for H.*

For any finite group G, a classical algorithm can call a routine evaluating $f(g)$ once for each $g \in G$, and thus determine H with $|G|$ function calls. A central challenge of quantum computing is to reduce this naive $O(|G|)$ time algorithm to $O(\text{poly}(\log |G|))$ time (including oracle calls and any needed classical post-processing time). This can be done for many groups, which gives the exponential speedup found in most quantum algorithms.

We assume an efficient encoding of G and X to basis states of our quantum computer. We also assume a quantum "black-box" that operates in unit time for performing the unitary transform $U_f|g\rangle|x\rangle = |g\rangle|x \oplus f(g)\rangle$, for $g \in G$, $x \in X$, and $\oplus$ bitwise addition on the state indices.

3.2. **The Fast Fourier Transform.** The Fast Fourier Transform (FFT) of Cooley and Tukey [34] reduced the cost of doing Fourier transforms from the naive $O(n^2)$ down to $O(n \log n)$, allowing a large class of problems to be attacked by computers. Mikhail Atallah[7] remarked the FFT is the most important algorithm in computer science. The success of the FFT is that so many other problems can be reduced to a Fourier transform, from multiplication of numbers and polynomials to image processing to sound analysis to correlation and convolution[8]. More references are Beth[19], Karpovsky[72], and Maslen and Rockmore [93].

Most, if not all, quantum algorithms that are exponentially faster than their classical counterparts rely on a quantum Fourier transform (QFT), and much of the rest of this document deals with the QFT. For more information beyond this paper on the QFT see Ekert and Jozsa[42], Hales and Hallgren[56], and Jozsa[68].

Just as the FFT turned out to be a big breakthrough in classical computing, exploiting the QFT so far is the central theme in quantum algorithms. The main reason quantum algorithms are exponentially faster is the QFT can be done exponentially faster than the classical FFT. However there are limitations due to the probabilistic nature of quantum states.

3.3. **The Basic Example.** Fix an integer $N > 1$. Let X be a finite set, and let $G = \langle \mathbb{Z}_N, + \rangle$ be the additive group of integers mod N. Suppose we have a function (set map) $f : G \to X$ such that there is a subgroup $H = \langle d \rangle$ of G, such that f is constant on H and distinct on cosets of H, that is, f separates cosets of H. Let $M = |H|$. We assume we have a quantum machine[9] capable of computing the unitary transform on two registers $f : |x\rangle|y\rangle \to |x\rangle|f(x) \oplus y\rangle$, where $\oplus$ is (qu)bitwise addition[10]. We do not assume we know M or d or H; we only know G and have a machine computing f. We want to determine a generating set for H, calling the "black-box" function f as few times as possible. For now we ignore the size of the quantum circuit and focus on the math making the whole process work. Later we will deal with efficiency.

Definition 3.3 (Quantum Fourier Transform (QFT))**.** *The quantum Fourier transform F_N is the operator on a register with $n \geq \log N$ qubits given by*

$$F_N = \frac{1}{\sqrt{N}} \sum_{j,k=0}^{N-1} e^{\frac{2\pi i j k}{N}} |k\rangle\langle j| \tag{20}$$

Note later we will define the QFT over other groups, so this one is actually the *cyclic* QFT.

The $\frac{1}{\sqrt{N}}$ factor is required to make this a unitary transformation[11], so it is a valid quantum transformation. Map the group, which we view as integers added mod N, into the basis of the quantum state, that is, $G = \{|0\rangle, |1\rangle, \ldots, |N-1\rangle\}$ and

[7] Private comment.

[8] Lomont [87] has shown that there can be no quantum correlation or convolution algorithms that parallel the quantum Fourier transform.

[9] Recall $|x\rangle|y\rangle$ merely means $|x\rangle \otimes |y\rangle$ and is used as shorthand.

[10] Check this is unitary, thus an allowable quantum operation.

[11] Homework!

$H = \{|0\rangle, |d\rangle, |2d\rangle, \ldots, |(M-1)d\rangle\}$. Compute on two registers:

$$|0\rangle|0\rangle \xrightarrow{F_N \text{ on 1st}} \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle|0\rangle \tag{21}$$

$$\xrightarrow{\text{apply } f} \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle|f(j)\rangle \tag{22}$$

Measuring the second register to obtain some value $f(j_0)$ collapses the state, leaving only those values in the first register that have $f(j_0)$ in the second register, namely the coset $H + j_0$. This is where we needed that f separates cosets of H. This "entanglement" is not present in classical computation, and seems to be one source of the increased computational power of quantum computing, another source being the ability to do computations on 2^n state coefficients in parallel by manipulating only n qubits. We now drop the second register which remains $|f(j_0)\rangle$.

$$\xrightarrow{\text{measure}} \frac{1}{\sqrt{M}} \sum_{h \in H} |j_0 + h\rangle \tag{23}$$

$$= \frac{1}{\sqrt{M}} \sum_{s=0}^{M-1} |j_0 + sd\rangle \tag{24}$$

$$\xrightarrow{\text{apply } F_N} \frac{1}{\sqrt{M}} \sum_{s} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i (j_0 + sd)k}{N}} |k\rangle \tag{25}$$

$$= \frac{1}{\sqrt{MN}} \sum_{k=0}^{N-1} e^{\frac{2\pi i j_0 k}{N}} |k\rangle \sum_{s=0}^{M-1} e^{\frac{2\pi i s d k}{N}} \tag{26}$$

Using $\frac{d}{N} = M$, evaluate the geometric series

$$\sum_{s=0}^{M-1} e^{\frac{2\pi i s d k}{N}} = \sum_{s=0}^{M-1} \left(e^{\frac{2\pi i k}{M}}\right)^s \tag{27}$$

$$= \begin{cases} 0 & \text{if } M \nmid k \\ M & \text{if } M \mid k \end{cases} \tag{28}$$

So in expression 26, only those values of k that are multiples of M remain, simplifying to the superposition

$$|\psi_f\rangle = \frac{1}{\sqrt{d}} \sum_{t=0}^{d-1} e^{\frac{2\pi i j_0 t M}{N}} |tM\rangle \tag{29}$$

Now measuring at this point gives a multiple of M in $\{0, M, \ldots, (d-1)M\}$ with uniform probability. All that remains is to repeat this to get several multiples of M, and to take the GCD to obtain M with high probability. Computing the GCD with the Euclidean algorithm[12] has complexity $O(\log^2 |N|)$, where $\log |N|$ is the number of digits in N.

To estimate how many trials we need, suppose we have obtained k multiples of M, say the (possibly repeated) multiples $t_1, \ldots, t_k \in T = \{0, 1, \ldots, d-1\}$. We want to estimate the probability that $\gcd(t_1, t_2, \ldots, t_k) = 1$, which would guarantee we

[12]This is the oldest known algorithm [79].

would obtain the true value of M, and hence determine H properly. By lemma E.3 in appendix E,

$$\text{prob}\left(\gcd(t_1, t_2, \ldots, t_k) = 1\right) \geq 1 - \left(\frac{1}{2}\right)^{k/2}$$

Thus a few runs of the algorithm determines H with high probability, for any size N and d. To understand the complete cost of the algorithm, we need the computational cost of the QFT, which is shown next in section 3.4. Then we show how these pieces can be used to find hidden subgroups in any finite abelian group in section 3.5, and finally in section 3.6 we show some applications.

Above we assume infinitely precise values in the operations making the QFT. Since this is not physically reasonable, work has been done to cover the case of slight errors in the precision of the computations. Kitaev [77] and the error correction methods of Calderbank and Shor [27] are good places to start, and show that it is still possible to sample multiples of M with high probability even with errors in the QFT, so the process works.

3.4. **Computing the Fourier Transform on $\mathbb{Z}_N$ Efficiently.** In this section we want to show how to compute the quantum Fourier transform F_N on the cyclic group $\mathbb{Z}_N$ efficiently, or at least approximate it to as high a precision as necessary. We will do this in two steps: first we do it for the case $N = 2^n$, and then use this in the second step to do it for general N. F_N will be used to construct HSP algorithms for general finite abelian groups. We make the next definition for general groups, but reserve the more general QFT definition until section 4.3.

Definition 3.4. *A family of quantum circuits $\{U_i\}$ computing the quantum Fourier transform over a family of finite groups $\{G_i\}$ is called* **efficient** *if U_i has size polynomial in $\log |G_i|$ for all i.*

Efficient quantum circuits for the Fourier transform over $\mathbb{Z}_N$ are well studied. Kitaev [77] gives an approximate method. Mosca and Zalka [100] use "amplitude amplification" [25] to give an exact method, but claim it is unlikely to be of practical use. Mosca's [98] thesis and Hales' thesis [55] both contain circuit descriptions. Hales and Hallgren [57] give the algorithm used in appendix A for the general case. For practical implementations of Shor's algorithm the "semiclassical" version given by Griffiths and Niu [52] would probably be the best known choice. Cleve and Watrous [33] have given parallel algorithms, showing even more speed increases. Shor [114] did the cyclic case for "smooth" values of N, and Coppersmith [35] gave an efficient algorithm for the case $N = 2^n$ as well as an approximate version. Brassard and Høyer [23] show how to solve Simon's problem, and have a useful framework for analyzing the general finite abelian HSP.

It has been said [59] that "The efficient algorithm for the abelian HSP is folklore." This section attempts to clear that up with precision.

3.4.1. *Reduction to Odd Order and 2^n Order.* As mentioned in Mosca's thesis [98, Appendix A.4], it is a fact that the Fourier transform F_N over a composite $N = AB$, with $(A, B) = 1$, can be computed efficiently from the efficient Fourier transforms over A and B. We show this briefly.

We assume $(A, B) = 1$, and we have efficient QFT algorithms for F_A and F_B. Let U_B be the unitary transform $|x \bmod A\rangle \xrightarrow{U_B} |xB \bmod A\rangle$, and similarly $|y \bmod$

$B\rangle \xrightarrow{U_A} |yA \bmod B\rangle$. Both U_A and U_B are efficiently computable, since they are just multiplication, followed by a remainder operation.

The main idea comes from the ring isomorphism $\mathbb{Z}_N \cong \mathbb{Z}_A \times \mathbb{Z}_B$, given in one direction by $j \to (j \bmod A, j \bmod B)$, and in the other direction by $(j_1, j_2) \to j_1 BB^{-1} + j_2 AA^{-1}$, where $AA^{-1} \equiv 1 \mod B$ and $BB^{-1} \equiv 1 \mod A$. These statements required $(A, B) = 1$. With this notation it is instructive to check

$$F_N = (U_B \otimes U_A)(F_A \otimes F_B). \tag{30}$$

This reduces the general QFT over $\mathbb{Z}_N$ for general N to the cases $N = 2^n$ and N odd. Finding QFT algorithms with time complexity of $O(\text{poly} \log N)$ for each case thus results in such an algorithm for any N, since U_A and U_B are efficient.

Thus for our purposes it is enough to show how to compute F_N efficiently for N a power of two and for N odd.

3.4.2. *The Case $N = 2^n$.* We start with the easiest case: $N = 2^n$. We show an explicit construction of the Fourier transform F_N, where $N = 2^n$. This presentation follows [29, Ch. 5], which in turn is adapted from sources mentioned in their book.

We use the notation from section 3.3, specialized to the case $N = 2^n$. We write the integer j in binary as $j = j_1 2^{n-1} + j_2 2^{n-2} + \cdots + j_n 2^0$, or in shorthand, as $j = j_1 j_2 \dots j_n$. We also adopt the notation $0.j_l j_{l-1} \dots j_m = j_l/2 + j_{l+1}/4 + \cdots + j_m/2^{m-l+1}$. Note the Fourier[13] operator F_N sends a basis element $|j\rangle$ to $\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} |k\rangle$. The inverse transform has ω_N^{-1} instead of ω_N. Then we can derive a formula giving an efficient way to compute the Fourier transform:

$$F_N|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi i j k}{2^n}} |k\rangle \tag{31}$$

$$= \frac{1}{\sqrt{N}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i j \sum_{l=1}^{n} k_l 2^{-l}} |k_1 k_2 \dots k_n\rangle \tag{32}$$

$$= \frac{1}{\sqrt{N}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \tag{33}$$

$$= \frac{1}{\sqrt{N}} \bigotimes_{l=1}^{n} \left[\sum_{k_l=0}^{1} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \right] \tag{34}$$

$$= \frac{1}{\sqrt{N}} \bigotimes_{l=1}^{n} \left[|0\rangle + e^{2\pi i j 2^{-l}} |1\rangle \right] \tag{35}$$

$$= \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1} j_n}|1\rangle\right) \dots \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right)}{\sqrt{N}} \tag{36}$$

where in the last step we used $\exp\left(2\pi i j 2^{-l}\right) = \exp\left(2\pi i j_0 j_1 \dots j_{n-l}.j_{n-l+1} \dots j_n\right) = \exp\left(2\pi i 0.j_{n-l+1} \dots j_n\right)$. Using this expression, we exhibit a quantum circuit (unitary operator) using $O((\log N)^2)$ elementary operations that transforms the state $|j\rangle$ into the one shown in equation 36.

[13] Note that the Fourier coefficients can be viewed as group homomorphisms $\omega_N^k : \mathbb{Z}_N \to \mathbb{C}^*$, taking $a \to \omega_N^{ka}$. This viewpoint generalizes well.

We need two types of unitary[14] operations, $H^{(a)}$ and $R_k^{(a,b)}$, where a and b index the qubits in the quantum machine, as follows[15]: Let $H^{(a)} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$ be the standard Hadamard operator, applied to qubit a, and let $R_k^{(a,b)}$ be the operator on qubits a and b given by

$$R_k^{(a,b)} = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & \omega_{2^k}\end{pmatrix}$$

where $\omega_N = e^{\frac{2\pi i}{N}}$ is the standard primitive N^{th} root of unity. $R_k^{(a,b)}$ has the effect of multiplying the phase of the $|1\rangle$ component of qubit b by ω_{2^k} if and only if qubit a is $|1\rangle$, and is called a controlled phase change. For example, looking at the two-qubit state,

$$(\alpha|0\rangle + \beta|1\rangle)\ |1\rangle \xrightarrow{R_5^{(2,1)}} \left(\alpha|0\rangle + \beta e^{2\pi i/2^5}|1\rangle\right)|1\rangle \tag{37}$$

Note each $H^{(a)}$ and $R_k^{(a,b)}$ is a local interaction on the quantum state, so we will count the number of them needed to implement a Fourier transform.

Apply to the state $|j_1 j_2 \dots j_n\rangle$ the operator $R_n^{(n,1)} R_{n-1}^{(n-1,1)} \dots R_2^{(2,1)} H^{(1)}$. We have

$$|j_1 j_2 \dots j_n\rangle \xrightarrow{H^{(1)}} \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_1}|1\rangle\right)|j_2 j_3 \dots j_n\rangle \tag{38}$$

$$\xrightarrow{R_2^{(2,1)}} \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2}|1\rangle\right)|j_2 j_3 \dots j_n\rangle \tag{39}$$

$$\dots \tag{40}$$

$$\xrightarrow{R_2^{(n,1)}} \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right)|j_2 j_3 \dots j_n\rangle \tag{41}$$

This required n local operations.

Apply to the state $|j_2 j_3 \dots j_n\rangle$ the operator $R_n^{(n,2)} R_{n-1}^{(n-1,2)} \dots R_2^{(3,2)} H^{(2)}$, which changes only the second qubit, resulting similarly in

$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right)\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_2 \dots j_n}|1\rangle\right)|j_3 j_4 \dots j_n\rangle \tag{42}$$

which required $n-1$ operations. Repeating this process uses $1+2+\dots+n = \frac{n(n+1)}{2}$ local operations and results in the state

$$\frac{\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right)\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_2 \dots j_n}|1\rangle\right)\dots\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)}{\sqrt{N}} \tag{43}$$

Noting this is similar to equation 36, we finish the Fourier transform by reversing the order of the qubits with approximately $\lfloor \frac{n}{2} \rfloor$ unitary qubit swaps. Thus the total number of operations, each affecting at most 2 qubits, is $O(n^2) = O(\log^2 N)$. We get an exact F_N transform with this method.

[14] A careful reader should check these are unitary.

[15] Note Chuang and Nielsen in [29] denote R_k as a single qubit operator, ours is what they would call a controlled R_k.

Most discussions avoid the following point. Notice as N grows, so the number of basic operations R_k grows as $\log N$, and this seems like cheating. If for each $N = 2^n$ we use only H and R_n, we may construct R_m, $0 \le m \le n$ as $(R_n)^{(n-m)}$, thus upping the complexity to $O(\log^3 N)$, which seems more fair from a complexity standpoint. Also, the $H^{(a)}$ were in list of elementary operations from section 2.2, but the $R_k^{(a,b)}$ were not. We remark they can be approximated in a manner leaving the overall QFT circuit efficient.

So this shows how to get an exact transform in $O(\log^2 N)$ or $O(\log^3 N)$ operations, depending on one's viewpoint. Since physical realizations will have error, we would be fine just approximating the QFT, a viewpoint detailed in Coppersmith [35], where he shows how to approximate the transform within any $\epsilon > 0$ in time $O(\log N(\log\log N + \log 1/\epsilon))$. See appendix A for more information on this result.

3.4.3. *The Case N Odd.* We use the algorithm over powers of 2 to get one for an odd N. The details of the proof are lengthy, and are left to Appendix A. The main result however gives

Theorem A.17. *Given an odd integer $N \ge 13$, and any $\sqrt{2} \ge \epsilon > 0$. Then F_N can be computed with error bounded by ϵ using at most $\left\lceil 12.53 + 3\log\frac{\sqrt{N}}{\epsilon}\right\rceil$ qubits. The algorithm has operation complexity*

$$O\left(\log\frac{\sqrt{N}}{\epsilon}\left(\log\log\frac{\sqrt{N}}{\epsilon} + \log 1/\epsilon\right)\right) \tag{44}$$

The induced probability distributions $\mathcal{D}_v$ from the output and $\mathcal{D}$ from $F_N|u\rangle \otimes |\psi\rangle$ satisfy

$$|\mathcal{D}_v - \mathcal{D}| \le 2\epsilon + \epsilon^2 \tag{45}$$

This says we can approximate the QFT very well. For odd $N < 13$ we can also design circuits using the methods in the proof. It is not currently known how to construct an exact QFT for odd cyclic groups, so this is as good as it (currently) gets.

3.4.4. *Final result: the Cyclic HSP Algorithm.* Combining sections 3.4.2 and 3.4.3 with the reasoning in section 3.3, we end up with the cyclic HSP algorithm:

The Hidden Subgroup Algorithm, Cyclic Abelian Case

Given: The group $G = \mathbb{Z}_N$ for a positive integer N, and a quantum black-box that evaluates a function $f : |x\rangle|y\rangle \to |x\rangle|f(x) \oplus y\rangle$, which we assume requires constant time[16].

Promise: f is constant on a subgroup $H = \langle d\rangle$ of G, and is distinct on cosets of H.

Output: The integer d, in time $O(\log^2 N)$ with probability at least $\frac{3}{4}$, and using at most $O(\text{poly}(\log N))$ qubits.

We proceed as follows

(1) Do the following steps for 8 trials, obtaining samples $t_1, t_2, ..., t_8$.
   (a) On the initial state $|0\rangle|0\rangle$ apply the quantum Fourier transform F_N (as earlier), with an approximation error of at most $\epsilon = 0.01$.

[16]Even if the time to compute f is not constant, if f can be computed efficiently, the overall algorithm is still efficient since f is called only a few times.

(b) Apply f in constant time.
(c) Sample the registers in constant time, obtaining t_j, a multiple of $M = |H|$.

(2) Compute $M = \gcd(t_1, t_2, \ldots, t_8)$ using the Euclidean algorithm[17] in time $O(\log^2 N)$.
(3) Output the answer $d = N/M$.

The probability of any one run returning a valid sample is at least $1 - (2\epsilon + \epsilon^2)$. We fix $\epsilon = 0.01$. We require 8 good samples, at which point the probability of them returning the correct GCD is at least $1 - (1/2)^4$, so the probability of success is then $(1 - (.0201))^8(15/16) > 3/4$. Oddly enough, the Euclidean Algorithm to compute the GCD requires more time than the QFT, and the result follows.

3.5. **The General Finite Abelian Group.** We want to generalize the cyclic case algorithm to all finite abelian groups. This discussion is a mixture of [23] and [36], with unified notation, and minor changes and corrections.

A basic result about finite abelian groups is the following structure theorem (Lang [83]):

Theorem 3.5. *Every finite abelian group G is a direct sum of cyclic groups.*

That is, $G \cong \mathbb{Z}_{N_1} \oplus \mathbb{Z}_{N_2} \oplus \ldots \mathbb{Z}_{N_k}$. Given generators for G, finding the N_i is hard classically, but Cheung and Mosca [28] (Theorem 5.23 below) give an efficient quantum algorithm to find the N_i. For example, given the cyclic group $\mathbb{Z}_N$, there is no known efficient classical algorithm to find the decomposition of the multiplicative group $\mathbb{Z}_N^*$ of integers relatively prime to N. Yet classically we can compute within this group efficiently.

So from now on, we assume we know the decomposition of our finite abelian group G, and can compute in G efficiently both classically (and hence) quantum mechanically.

Let $G = \mathbb{Z}_{N_1} \oplus \cdots \oplus \mathbb{Z}_{N_k}$ be a finite additive abelian group, and assume we have a function f from G to a finite set X, such that there is a subgroup $H < G$ such that f separates cosets of H as in section 3.1. Denote elements of G as k-tuples: $g = (g_1, \ldots, g_k)$, where we view g_j either as an integer mod N_j or an integer $\in \{0, 1, \ldots, N_j - 1\}$. Write $-g$ for the (additive) inverse of $g \in G$.

To generalize the cyclic group Fourier transform F_N to an arbitrary finite abelian group, we need some representation theory, specifically character theory, and to this area we now turn. See also section 4.2 for representation theory basics.

3.5.1. *Character Theory of Finite Abelian Groups.* To define a Fourier transform over G, we need to generalize the ω_N^{jk} terms from the cyclic case, basically by putting one such term for each entry in the k-tuple description of G.

Definition 3.6. *A* **character** *of a group G is a group homomorphism from G to the multiplicative group of nonzero complex numbers $\mathbb{C}^*$.*

Recall this is then just a map of sets $\chi : G \to \mathbb{C}^*$ such that

$$\chi(g_1 + g_2) = \chi(g_1)\chi(g_2) \tag{46}$$

[17]The GCD complexity follows from $O(\log N)$ time algorithms for division in [15] and that the most steps used in the Euclidean algorithm happens when the input is two consecutive Fibonacci numbers multiplied by an integer.

We think of G as having an additive structure, and $\mathbb{C}^*$ a multiplicative structure. From this simple definition we derive some tools and facts which will allow us to finish the HSP discussion for finite abelian groups.

Our first task is to describe all characters $\chi : G \to \mathbb{C}^*$. Denote the identity of G by $e = (0, 0, \ldots, 0)$; the identity of $\mathbb{C}^*$ is 1.

Let $\chi : G \to \mathbb{C}^*$ be a character (so $\chi(ng) = \chi(g)^n$ for any integer n and group element g). Let $\beta_1 = (1, 0, 0, \ldots, 0) \in G$, $\beta_2 = (0, 1, 0, 0 \ldots, 0) \in G, \ldots, \beta_k = (0, 0, \ldots, 0, 1) \in G$. Then for any element $g = (g_1, g_2, \ldots, g_k)$ we have

$$\chi(g) = \chi\left(\sum_{j=1}^{k} g_j \beta_j\right) \tag{47}$$

$$= \prod_{j=1}^{k} \chi(\beta_j)^{g_j} \tag{48}$$

so χ is completely determined by its values on the β_j. Since β_j has order N_j, $\chi(\beta_j)$ must have order dividing N_j, for each j. Then we must have[18] that $\chi(\beta_j) = \omega_{N_j}^{h_j}$ for some integer h_j. It is sufficient to consider $h_j \in \{0, 1, \ldots, N_j - 1\}$ since the values of $\omega_{N_j}^{h_j}$ are periodic, so any given character $\chi : G \to \mathbb{C}^*$ is determined by a k-tuple $(h_1, h_2, \ldots, h_k)$, which may be viewed as an element $h \in G$. This allows labelling each distinct character χ by an element of G: for each $g \in G$ define the character $\chi_g : G \to \mathbb{C}^*$ via $\chi_g(h) = \prod_{j=1}^{k} \omega_{N_j}^{g_j h_j}$, for $h \in G$. From this definition we notice that for all $g, h \in G$

$$\chi_g(h) = \chi_h(g) \tag{49}$$

$$\chi_g(-h) = \frac{1}{\chi_g(h)} \tag{50}$$

Let $\chi(G)$ denote the set of all such homomorphisms, which is a group under the operation $\chi_{g_1}\chi_{g_2} = \chi_{g_1+g_2}$ with identity χ_e. Then we prove

Theorem 3.7. *For a finite abelian group G, $\chi(G) \cong G$.*

Proof. From the discussion above, there is a set bijection between the two sets given (in one direction) by $\alpha : g \to \chi_g$, which is also a group isomorphism. The identity $e = (0, 0, \ldots, 0) \in G$ is sent to the identity $\alpha(e) = \chi_e$ in $\chi(G)$, and $\alpha(g_1 + g_2) = \chi_{g_1+g_2} = \chi_{g_1}\chi_{g_2} = \alpha(g_1)\alpha(g_2)$, making α a group homomorphism and a set bijection, thus an isomorphism. □

In the cyclic QFT algorithm, we sampled elements that were multiples of the generator of the subgroup H, and to generalize this to the finite abelian case where there may not be a single generator, we introduce *orthogonal elements*. For any subset $X \subseteq G$, we say an element $h \in G$ is *orthogonal* to X if $\chi_h(x) = 1$ for all $x \in X$. Then for any subgroup $H < G$ we define the *orthogonal subgroup*

$$H^\perp = \{g \in G | \chi_g(h) = 1 \text{ for all } h \in H\} \tag{51}$$

as the set of all elements in G orthogonal to H. $H^\perp$ is a subgroup of G as follows: the identity $e \in G$ is in $H\perp$ since $\chi_e(g) = 1$ for all $g \in G$, and if $a, b \in H^\perp$ then for any $h \in H$ we have $\chi_h(a - b) = \chi_h(a)/\chi_h(b) = 1$ so $a - b \in H^\perp$, and $H^\perp$ is a subgroup of G.

[18] Recall ω_N is a primitive N^{th} root of unity, from section 1.3.

Note 3.8. These orthogonal subgroups are not quite like orthogonal subspaces. For example, we could have nontrivial $H \cap H^\perp$, unlike the vector space example. Here is an example following [36] where $H = H^\perp \neq G$. Let $G = \mathbb{Z}_4$, $H = \{0, 2\}$. Then $H^\perp = \{(a) \in G | (i)^{ah} = 1 \text{ for all } (h) \in H\} = \{(a) | (-1)^a = 1\} = H$. This can be extended to give examples of varying weirdness.

Another useful fact is

Theorem 3.9. *Let G be a finite abelian group, and $\chi \in \chi(G)$ a fixed character, and χ_e the identity character sending $G \to 1$. Then*

$$\sum_{g \in G} \chi(g) = \begin{cases} |G| & \text{if } \chi = \chi_e \\ 0 & \text{if } \chi \neq \chi_e \end{cases} \tag{52}$$

Proof. Fix $G \cong \mathbb{Z}_{N_1} \oplus \cdots \oplus \mathbb{Z}_{N_k}$, and by theorem 3.7 fix $h \in G$ with $\chi = \chi_h$. Using the notation above,

$$\sum_{g \in G} \chi_h(g) = \sum_{g_1 \in \mathbb{Z}_{N_1}} \sum_{g_2 \in \mathbb{Z}_{N_2}} \cdots \sum_{g_k \in \mathbb{Z}_{N_k}} \prod_{j=1}^{k} \omega_{N_j}^{h_j g_j} \tag{53}$$

$$= \left(\sum_{g_1 \in \mathbb{Z}_{N_1}} \omega_{N_1}^{h_1 g_1} \right) \left(\sum_{g_2 \in \mathbb{Z}_{N_2}} \omega_{N_2}^{h_2 g_2} \right) \cdots \left(\sum_{g_k \in \mathbb{Z}_{N_k}} \omega_{N_k}^{h_k g_k} \right) \tag{54}$$

If some $\omega_{N_j}^{h_j} \neq 1$, then the geometric series $\sum_{g_j \in \mathbb{Z}_{N_j}} \left(\omega_{N_j}^{h_j} \right)^{g_j} = 0$, making the entire product 0. This happens if and only if $\chi_h \neq \chi_e$. If $\chi_h = \chi_e$ then the sum is $|G|$. □

We now prove some relations between H and $H^\perp$.

Theorem 3.10. *With the notation above,*

$$G/H \cong H^\perp \tag{55}$$

$$H^{\perp\perp} = H \tag{56}$$

Proof. Using theorem 3.7, we already have $H^\perp \cong \chi(H^\perp)$ and $\chi(G/H) \cong G/H$, so it is enough to prove $\chi(H^\perp) \cong \chi(G/H)$. For any element $g \in G$ let $\overline{g}$ denote the image in G/H under the projection map $\pi : G \to G/H$. Note that any character $\chi_{h'} \in \chi(H^\perp)$ coming from an element $h' \in H^\perp$ can also be viewed as a character on G, since h' is also in G. Then define a map $\alpha : \chi(H^\perp) \to \chi(G/H)$ via

$$(\alpha\chi)(\overline{g}) = \chi_{h'}(g)$$

where $\overline{g} \in G/H$ and g is any coset representative, i.e., $\overline{g} = g + H$. We will show α is a group isomorphism.

α is well defined since if g_1 and g_2 are different representations of the same coset $\overline{g_1} = \overline{g_2}$, then there is an $h \in H$ with $g_1 - g_2 = h$, giving $(\alpha\chi_{h'})(\overline{g_1}) = \chi_{h'}(g_1) * 1 = \chi_{h'}(g_1 + h) = \chi_{h'}(g_2) = (\alpha\chi_{h'})(\overline{g_2})$. For the identity $\chi_e \in \chi(H^\perp)$ and any $\overline{g} \in G/H$ we have $(\alpha\chi_e)(\overline{g}) = \chi_e(g) = 1$, so $\alpha\chi_e$ is the identity in $\chi(G/H)$. Also for $\overline{(g)} \in G$ $(\alpha(\chi_{h_1}\chi_{h_2}))(\overline{g}) = (\alpha(\chi_{h_1+h_2}))(\overline{g}) = \chi_{h_1+h_2}(g) = \chi_{h_1}(g)\chi_{h_2}(g) = ((\alpha\chi_{h_1})(\alpha\chi_{h_2}))(\overline{g})$, so α is a group homomorphism.

To show α is injective, suppose for some $h' \in H^\perp$ that $\alpha\chi_{h'}$ is the identity in $\chi(G/H)$. Take any $g \in G$. $\alpha\chi_{h'}(\overline{g}) = 1$ implies $\chi_{h'}(g) = 1$, and since this is for any $g \in G$, we have $\chi_{h'} = \chi_e$. $G \cong \chi(G)$ then gives $h' = e$, and thus α is injective.

Now all we need is to show alpha is surjective. Let $\overline{\chi} \in \chi G/H$. The composite map with the projection $\pi : G \to G/H$ gives a homomorphism $\chi = \overline{\chi} \circ \pi : G \xrightarrow{\pi} G/H \xrightarrow{\overline{\chi}} \mathbb{C}^*$, thus is a character, say χ_t, for some fixed $t \in G$. For $h \in H$ this evaluates to $\chi_t(h) = \overline{\chi}(\overline{e}) = 1$, so $t \in H^\perp$, and $\chi_t \in \chi(H^\perp)$. To show $\alpha\chi_t = \overline{\chi}$, let $\overline{g} \in G/H$, and compute: $(\alpha\chi_t)(\overline{g}) = \chi_t(g) = \overline{\chi}\pi(g) = \overline{\chi}(\overline{g})$. Thus α is surjective and thus a group isomorphism.

To show $H^{\perp\perp} = H$ start with the isomorphism already proven: $|G/H| = |H^\perp|$ gives $|G/H^\perp| = |H|$ and also implies $|G/H^\perp| = |H^{\perp\perp}|$, giving $|H| = |H^{\perp\perp}|$. Fix $h \in H$. By definition $H^{\perp\perp} = \{g \in G | \chi_g(h') = 1 \text{ for all } h' \in H^\perp\}$. In particular $\chi_h(h') = \chi_{h'}(h) = 1$ for all $h' \in H^\perp$ by the definition of $H^\perp$, so we have $h \in H^{\perp\perp}$, giving $H \subseteq H^{\perp\perp}$. Thus $H = H^{\perp\perp}$. □

3.5.2. *The General Finite Abelian Group Quantum Fourier Transform.* We continue the notation from the previous section. Similar to the cyclic QFT algorithm returning multiples of the generator of H (which is really the orthogonal subgroup), this general finite abelian QFT algorithm will return elements of the orthogonal subgroup $H^\perp$. We start with the Fourier transform.

We define three quantum operators over the group G: the *Fourier transform* F_G over G, the *translation operator* τ_t for a $t \in G$, and the *phase-change operator* ϕ_h for $h \in G$ as

$$F_G = \frac{1}{\sqrt{|G|}} \sum_{g,h \in G} \chi_g(h)|g\rangle\langle h| \tag{57}$$

$$\tau_t = \sum_{g \in G} |t+g\rangle\langle g| \tag{58}$$

$$\phi_h = \sum_{g \in G} \chi_g(h)|g\rangle\langle g| \tag{59}$$

Note that for cyclic $G = \mathbb{Z}_N$ the Fourier transform is the same as earlier in section 3.3, since then $\chi_h(g) = e^{\frac{2\pi i h g}{N}}$, and we recover the earlier algorithm.

First we check that the Fourier transform maps a subgroup H to its orthogonal subgroup $H^\perp$.

Theorem 3.11.

$$F_G|H\rangle = |H^\perp\rangle \tag{60}$$

Proof. Recall from the definition of a subset $|H\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |h\rangle$. Then

$$F_G|H\rangle = \frac{1}{\sqrt{|G|}} \sum_{g,h' \in G} \chi_g(h')|g\rangle\langle h'| \frac{1}{\sqrt{|H|}} \sum_{h \in H} |h\rangle \tag{61}$$

$$= \frac{1}{\sqrt{|G||H|}} \sum_{\substack{g,h' \in G \\ h \in H}} \chi_g(h')|g\rangle\langle h'|h\rangle \tag{62}$$

$$= \frac{1}{\sqrt{|G||H|}} \sum_{\substack{g \in G \\ h \in H}} \chi_g(h)|g\rangle \tag{63}$$

$$= \frac{1}{\sqrt{|G||H|}} \sum_{g \in G} \left(\sum_{h \in H} \chi_g(h) \right) |g\rangle \tag{64}$$

Now consider the coefficient $\sum_{h\in H}\chi_g(h)$ of the ket $|g\rangle$. The G character χ_g is also a character of H, so by theorem 3.9 the sum is 0 unless the character is the identity on H, in which case the sum is $|H|$. χ_g is the identity on H precisely when $\chi_g(h)=1$ for all $h\in H$, i.e., $g\in H^\perp$. So equation 64 becomes

$$\frac{1}{\sqrt{|G||H|}}\sum_{g\in H^\perp}|H||g\rangle = \sqrt{\frac{|H|}{|G|}}\sum_{g\in H^\perp}|g\rangle \tag{65}$$

$$= |H^\perp\rangle \tag{66}$$

where we used theorem 3.10 to get $\frac{|H|}{|G|}=\frac{1}{|H^\perp|}$. □

We also have

Theorem 3.12 (Commutative laws of the G-operators). *For every $h,t\in G$ we have*

$$\chi_h(t)\tau_t\phi_h = \phi_h\tau_t \tag{67}$$

$$F_G\phi_h = \tau_{-h}F_G \tag{68}$$

$$F_G\tau_t = \phi_t F_G \tag{69}$$

Proof. We prove the last one, which is the only one we explicitly use. The rest are similar. We use the identity $I=\sum_{g\in G}|g\rangle\langle g|$.

$$\begin{aligned}
F_G\tau_t &= \left(\frac{1}{\sqrt{|G|}}\sum_{g,h\in G}\chi_g(h)|g\rangle\langle h|\right)\left(\sum_{g'\in G}|t+g'\rangle\langle g'|\right)\\
&= \frac{1}{\sqrt{|G|}}\sum_{g,g',h\in G}\chi_g(h)|g\rangle\langle h|t+g'\rangle\langle g'|\\
&= \frac{1}{\sqrt{|G|}}\sum_{g,g'\in G}\chi_g(t+g')|g\rangle\langle g'|\\
&= \frac{1}{\sqrt{|G|}}\sum_{g,g'\in G}\chi_g(t)\chi_g(g')|g\rangle\langle g'|\\
&= \frac{1}{\sqrt{|G|}}\sum_{a,g,g'\in G}\chi_g(t)\chi_g(g')|a\rangle\langle a|g\rangle\langle g'|\\
&= \frac{1}{\sqrt{|G|}}\sum_{a,g,g'\in G}\chi_a(t)\chi_g(g')|a\rangle\langle a|g\rangle\langle g'|\\
&= \left(\sum_{a\in G}\chi_a(t)|a\rangle\langle a|\right)\left(\frac{1}{\sqrt{|G|}}\sum_{g,g'\in G}\chi_g(g')|g\rangle\langle g'|\right)\\
&= \phi_t F_G
\end{aligned}$$

□

Then the algorithm becomes:

(1) Apply the quantum Fourier transform[19] to the first register of the zero state on two registers:

$$|0\rangle|0\rangle \xrightarrow{F_G} \frac{1}{\sqrt{|G|}}\sum_{g\in G}|g\rangle|0\rangle$$

obtaining a superposition over all elements of G.

(2) Apply the coset separating function f:

$$\xrightarrow{f} \frac{1}{\sqrt{|G|}}\sum_{g\in G}|g\rangle|f(g)\rangle$$

and as before, f constant and distinct on cosets allows the simplification

$$\begin{aligned} &= \frac{1}{\sqrt{|T|}}\sum_{t\in T}|t+H\rangle|f(t)\rangle \\ &= \frac{1}{\sqrt{|T|}}\sum_{t\in T}\tau_t|H\rangle|f(t)\rangle \end{aligned}$$

where $T=\{t_1,\dots,t_m\}$ is a transversal (*set of coset representatives*) for H in G.

(3) Apply the Fourier transform F_G to the first register, and apply theorems 3.11 and 3.12

$$\begin{aligned} \xrightarrow{F_G} &\frac{1}{\sqrt{|T|}}\sum_{t\in T}F_G\tau_t|H\rangle|f(t)\rangle \\ = &\frac{1}{\sqrt{|T|}}\sum_{t\in T}\phi_t F_G|H\rangle|f(t)\rangle \\ = &\frac{1}{\sqrt{|H^\perp|}}\sum_{t\in T}\phi_t|H^\perp\rangle|f(t)\rangle \end{aligned}$$

We used that $|T| = |G|/|H| = |H^\perp|$ by theorem 3.10. Note we could have measured the second register as in the cyclic case, but a fact called "The Principle of Deferred Measurement" allows us to measure at the end[20].

(4) Measure the first register, obtaining a random element (uniformly distributed) of $H^\perp$. Note that the phase ϕ_t does not affect amplitudes, so we could measure the second register first if we desired, fixing a t_0, as mentioned in the previous step.

This algorithm returns uniformly distributed random elements of $H^\perp$. Since $(H^\perp)^\perp = H$, determining a generating set for $H^\perp$ determines H uniquely. The following discussion comes from [36], with details not mentioned there to make the results precise.

Theorem D.1 in appendix D proves that choosing $t+\lceil\log|G|\rceil$ uniformly random elements of a finite group G will generate G with probability greater than $1-\frac{1}{2^t}$.

---

[19] Usually the *inverse* transform is applied here, but this has the same effect for the $|0\rangle$ state. [36, Lemma 8] allows quicker setting of these superposed states with high probability.

[20] As you will see we still get the desired outcome whether or not we measure twice, or only once at the end.

For the moment assume we have chosen a generating[21] set $g^1, g^2, \ldots, g^t$ for $H^\perp$. We want to find efficiently a generating set for H, finishing the algorithm. Since $H^{\perp\perp} = H$, an element $h \in H$ if and only if $\chi_h(h'_j) = 1$ for all $j = 1, 2, \ldots, t$. Next we make these relations linear.

Let $d = \text{LCM}\{N_1, N_2, \ldots, N_k\}$. Set $\alpha_l = d/N_l$, giving $\omega_{N_l} = \omega_d^{\alpha_l}$. Then $\chi_h(g^j) = \prod_{l=1}^k \omega_d^{\alpha_l h_l g_l^j} = 1$ if and only if $\sum_{l=1}^k \alpha_l h_l g_l^j \equiv 0 \pmod{d}$. So to find elements of H, we find random solutions to the system of t linear equations

$$
\begin{array}{rcl}
\alpha_1 g_1^1 X_1 + \alpha_2 g_2^1 X_2 + \cdots + \alpha_k g_k^1 X_k & \equiv & 0 \pmod{d} \\
\alpha_1 g_1^2 X_1 + \alpha_2 g_2^2 X_2 + \cdots + \alpha_k g_k^2 X_k & \equiv & 0 \pmod{d} \\
\vdots & & \vdots \\
\alpha_1 g_1^t X_1 + \alpha_2 g_2^t X_2 + \cdots + \alpha_k g_k^t X_k & \equiv & 0 \pmod{d}
\end{array} \tag{70}
$$

We do the following. Run the algorithm $T = t_1 + \lceil \log |G| \rceil$ times, giving elements $g^1, g^2, \ldots, g^T \in H^\perp$. Since $H^\perp \subseteq G$, these elements generate $H^\perp$ with probability $p_1 \geq 1 - 1/2^{t_1}$. We want to sample solutions to the system of equations 70 randomly and uniformly, to get $S = t_2 + \lceil \log |G| \rceil$ samples of H, which would generate H with probability $p_2 \geq 1 - 1/2^{t_2}$. To sample the solutions, view the equations in matrix form $AX \equiv 0 \pmod{d}$, and then compute the Smith normal form[22] of A, that is, a diagonal matrix D such that $D = UAV$ with U and V being integer valued invertible matrices. Then we can uniformly randomly find solutions to $DY \equiv 0 \pmod{d}$ by solving simple linear congruences, and then compute $X = VY$, which is a uniformly randomly selected solution to the system of equations 70. This determines generators of H with probability at least $(1 - \frac{1}{2^{t_1}})(1 - \frac{1}{2^{t_2}})$.

Note that $F_G = \otimes_{j=1}^k F_{N_j}$, so we compute it by using the cyclic case algorithm from section 3.4.4, with the time complexity listed there. Choosing $t_1 = t_2 = \lceil \log |G| \rceil + 1$ gives a probability of success at least $1 - \frac{1}{|G|}$. After obtaining the system of equations 70, we compute D and V in time $O(\log |G| \log\log |G|)$ as in [120]. Then we sample the resulting system $O(\log |G|)$ times, and convert the answers to solutions to 70, totaling a time $O(\text{poly}(\log |G|))$.

Thus we have proven the following (partially stated in Ettinger and Høyer [45], theorem 2.2.)

Theorem 3.13 (Finite abelian HSP algorithm)**.** *Given a finite abelian group G, a finite set X, and a function $f : G \to X$ that separates cosets of H for some subgroup $H < G$, then there exists a quantum algorithm that outputs a subset $S \subseteq H$ such that S is a generating set for H with probability at least $1 - 1/|G|$. The algorithm uses $O(\log |G|)$ evaluations of f, and runs in time polynomial in $\log |G|$ and in the time required to compute f, using a quantum circuit of size $O(\log |G| \log\log |G|)$.*

3.6. **The Standard Problems.** Now that we can efficiently find hidden subgroups of finite abelian groups, we show a few examples of how to use the algorithms. For a longer list of examples, see [29, Figure 5.5]. We merely mention some algorithms that fall into this framework: Deutsch's algorithm [37] (modified by Cleve), Deutsch and Jozsa's algorithm [38], Simon's algorithm [118], Shor's factoring and discrete

[21] Here the exponent does not denote power, but is used since later we will use subscripts on these elements.

[22] [120] shows how to compute the Smith Normal $D = UAV$ form of an $m \times n$ integer matrix A mod d in time $O(n^2 m)$, and recover the U and V in time $O(n^2 m \log^c(n^2 m))$, for some constant $c > 0$.

log algorithms [115], hidden linear function algorithms, and the abelian stabilizer algorithm [77]. Now on to two examples.

3.6.1. *Simon's Algorithm.* Simon's [118] algorithm distinguishes a trivial subgroup from an order 2 subgroup over the additive group $\mathbb{Z}_2^n$. He showed that a classical probabilistic oracle requires *exponentially* many (in n) more oracle queries than a quantum algorithm to distinguish the two subgroup types with probability greater than $1/2$, giving a major boost to the argument that quantum computers may be more powerful than classical ones. He posed the following problem in 1994 (modified somewhat to fit our discussion):

GIVEN a function $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ with $m \geq n$, and such that there is a constant $s \in \mathbb{Z}_2^n$ for which $f(x) = f(x')$ if and only if $x = x' \oplus s$, where $\oplus$ is componentwise (binary) addition.

FIND s.

Here the subgroup is $H = \{0, s\} < G = \mathbb{Z}_2^n$, and so we can find it quickly with high probability using the algorithm from theorem 3.13. However, to solve this classically, one may have to call f $O(|G|)$ times, evaluating f on many points, to find the value s.

3.6.2. *Shor's Factoring Algorithm.* Shor [114] generalizes Simon's algorithm to obtain an integer factorization (and discrete log) algorithm. A good explanation is also in [69]. Integer factorization is classically very hard (see Lenstra and Pomerance [71]), and is the basis of the widely used public key cryptography algorithm RSA. Shor's Integer Factorization Algorithm reduces to finding the order r of an integer $x \bmod N$, that is, the smallest r such that $x^r \equiv 1 \bmod N$. We wish to factor a composite integer $N > 0$, and it suffices to find a non-trivial solution to $x^2 \equiv 1 \bmod N$, then $x+1$ or $x-1$ is a factor of N. A randomly chosen y relatively prime to N is likely to have even order, giving the solution $x = y^{(r/2)}$. All of this, except the order finding part, is efficient classically. Thus the hard part of the problem is to find the order of a given x modulo N. In other words, $f(a) = x^a \bmod N$, so $f(a + r) = f(a)$ for all a, and the HSP finds the generator r of the subgroup $\langle r \rangle = H < G = \mathbb{Z}_N$.

3.7. **Conclusion.** In conclusion, we have shown that for any finite abelian group G, and any efficiently computable function f that separates cosets of some subgroup $H < G$, we can efficiently find a generating set for H with high probability. This was summarized in theorem 3.13.

In the process of doing this we isolated a few items needed to construct an efficient HSP algorithm for a group G:

(1) An efficient way is needed to compute the quantum Fourier transform over the group G. This evolved from the simple Fourier transform, through a more abstract one involving character theory, and in the general setting will involve representation theory[23] to define the Fourier transform over nonabelian groups.
(2) An efficient way is needed to compute the coset separating function f. For Shor's algorithm this is raising an integer to a power mod N, which is efficient classically. Simon's algorithm had bitwise addition as the function, which also is efficient classically.

[23] See section 4.2 for representation theory basics.

(3) Finally, these HSP algorithms needed some post processing to extract the desired information from the randomly sampled elements of the orthogonal subgroup. This will turn out to be hard for nonabelian groups. Groups are known with efficient quantum Fourier transforms, but no known polynomial time algorithm is available to reconstruct hidden subgroups.

With that said, let's begin analyzing the general (nonabelian case).

4. The General Hidden Subgroup Problem

Why do we want to find hidden subgroups of nonabelian groups? An efficient abelian HSP algorithm yielded an integer factoring algorithm which is exponentially faster than any known classical algorithm. Similarly, finding efficient HSP algorithms over certain nonabelian groups would yield algorithms faster than any known classical ones for several important problems, two of which we now explain.

4.1. **Importance.**

One of the main reasons much research has been done into the HSP problem for nonabelian groups is the desire to find an efficient algorithm for the Graph Isomorphism problem: when are two graphs isomorphic? This algorithm has eluded researchers for over thirty years [81, 94]. Appendix B shows equivalences between several graph related algorithms, and describes several reductions. One reduction shown in appendix B gives that if the HSP could be solved efficiently for the symmetric group S_n, then we would have a polynomial time algorithm for the Graph Isomorphism Problem.

Another reason is that an efficient algorithm for solving the HSP for the dihedral group D_n would yield a fast algorithm for finding the shortest vector in a lattice, first shown by Regev [107]. This would yield another algorithm whose classical counterpart is much less efficient than the quantum version. Finding the shortest lattice vector has many uses, including applications to cryptography.

Before we cover the nonabelian HSP, we need to generalize the QFT algorithm, which is what the rest of this section will do. Then section 5 will list the main results known so far for the nonabelian HSP.

4.2. **Representation Theory Overview.**

To generalize the abelian QFT algorithm, we need the nonabelian analogue of the Fourier transform. The method explained in section 3.5 shows the general machinery: we need representations of the group G. What follows is a brief overview of representation theory, which can be seen in detail in either of the excellent texts Fulton-Harris [63] or Serre [113]. We only cover enough of the definitions and facts to define precisely the quantum Fourier transform for finite groups. Some definitions and facts:

Representation. A representation ρ of a group G is a group homomorphism $\rho : G \to GL(V)$ where V is a vector space over a field $\mathbb{F}$. For our purposes G will be finite, V will be finite dimensional of (varying) dimension d, and the field $\mathbb{F}$ will be the complex numbers $\mathbb{C}$. Fixing a basis of V, each $g \in G$ gives rise to a $d \times d$ invertible matrix $\rho(g)$, which we can take to be unitary. The *dimension* d_ρ of the representation is the dimension d of V. We will often use the term ***irrep*** as shorthand for an irreducible representation.

We say two representations $\rho_1 : G \to GL(V)$ and $\rho_2 : G \to GL(W)$ are *isomorphic* when there is a linear vector space isomorphism $\phi : V \cong W$ such that for all $g \in G$ and $v \in V$, $\rho_1(g)(v) = \rho_2(g)(\phi(v))$. In this case we write $\rho_1 \cong \rho_2$.

Irreducibility. We say a subspace $W \subseteq V$ is an *invariant* subspace of a representation ρ if $\rho(g)W \subseteq W$ for all $g \in G$. Thus the zero subspace and the total space V are invariant subspaces. If there are no nonzero proper subspaces, the representation is said to be *irreducible*.

Decomposition. When a representation does have a nonzero proper subspace $V_1 \subsetneq V$, it is always possible to find a complementary invariant subspace V_2 so that $V = V_1 \oplus V_2$. The restriction of ρ to V_i is written ρ_i, and these give representations $\rho_i : G \to GL(V_i)$. Then $\rho = \rho_1 \oplus \rho_2$, and there is a basis of V so that each matrix $\rho(g)$ is in block diagonal form with a block for each ρ_i.

Complete reducibility. Repeating the decomposition process, we obtain for any representation a decomposition $\rho = \rho_1 \oplus \cdots \oplus \rho_k$, where each representation ρ_i is irreducible. This is unique up to permutation of isomorphic factors.

Complete set of irreducibles. Given a group G, there are a finite number of irreducible representations up to isomorphism. We label this set $\hat{G}$. Then we have the fact

$$|G| = \sum_{\rho \in \hat{G}} d_\rho^2. \tag{71}$$

Characters. To a representation ρ is associated a *character* χ_ρ defined by $\chi_\rho(g) = \mathbf{tr}(\rho(g))$, where **tr** is the trace of the matrix. It is basis independent. An alternative, equivalent description is that a character is a group homomorphism $\chi : G \to \mathbb{C}^*$ where $\mathbb{C}^*$ denotes complex numbers of unit length, and the operation in C is multiplication, as we saw in section 3.5. Characters are fixed on conjugacy classes, which follows easily from the second definition: $\chi(hgh^{-1}) = \chi(h)\chi(g)\chi(h^{-1}) = \chi(g)$.

Orthogonality of characters. For two functions $f_1, f_2 : G \to \mathbb{C}$, there is a natural *inner product* $\langle f_1, f_2 \rangle_G = \frac{1}{|G|} \sum_{g \in G} f_1(g) f_2(g)^*$ where $*$ denotes complex conjugation. The main fact is: given the character χ_ρ of a representation ρ and the character χ_i of an irreducible representation ρ_i, the inner product $\langle \chi_\rho, \chi_i \rangle_G$ is exactly the number of times the representation ρ_i appears in the decomposition of ρ into irreducibles. Taking each ρ as unitary simplifies the inner product to

$$\langle \chi_\rho, \chi_i \rangle_G = \frac{1}{|G|} \sum_{g \in G} \chi_\rho(g) \chi_i(g^{-1})$$

Orthogonality of the second kind. Let C be a conjugacy class of G. Since a character χ_ρ is fixed on a conjugacy class, let this value be $\chi_\rho(C)$. Then

$$\sum_{\rho \in \hat{G}} |\chi_\rho(C)|^2 = \frac{|G|}{|C|}$$

The Regular Representation. Take $\dim V = |G|$, and fix a basis of V indexed by elements of G, labelling the basis as e_g. Then the regular representation $\rho_G : G \to GL(V)$ is defined by G permuting the basis elements, i.e., $\rho_G(g)e_x = e_{gx}$, extended $\mathbb{C}$-linearly. Thus the dimension of the regular representation is $|G|$. Another way to view this representation is as the group algebra $\mathbb{C}[G]$.

The regular representation contains as subrepresentations every irreducible representation of G. If $\rho_1, \ldots, \rho_k$ are all the possible irreducible representations of G, then

$$\rho_G = \rho_1^{\oplus d_{\rho_1}} \oplus \cdots \oplus \rho_k^{\oplus d_{\rho_k}}$$

that is, each irreducible ρ_i is contained exactly d_{ρ_i} times. This yields the important relation in equation 71. Taking the character associated to this gives, for $g \in G$, the "regular character"

$$\chi_G(g) = \sum_{\rho \in \hat{G}} d_\rho \chi_\rho(g) = \begin{cases} 0 \text{ if } g \neq e \\ N \text{ if } g = e \end{cases}, \tag{72}$$

where the last equality is obtained by noting that $\rho(g)$ acts on $\mathbb{C}[G]$ by permuting basis elements, so the trace is 0 if $g \neq e$ (all basis elements are permuted by any non-identity element g, so the diagonal is all 0's) and is otherwise N.

The Induced Representation. Given a representation $\rho : H \to \mathrm{GL}(W)$ of a subgroup H in a group G, we can define a way to extend this to a representation on G written $\mathbf{Ind}_H^G \rho : G \to \mathrm{GL}(V)$, unique up to isomorphism. The idea is to make copies of W for each coset of H in G, and let cosets permute the copies. So let $\Lambda = \{e, \tau_1, \ldots, \tau_k\}$ be a complete set of coset representatives, and let $V = \oplus_{\tau \in \Lambda} W_\tau$. Then any $g \in G$ can be written $g = \tau_g h_g$ for some representative $\tau_g \in \Lambda$ and $h_g \in H$, which acts on V via $\tau_g h_g \left(\oplus W_\tau\right) = \oplus h_g W_{\tau_g \tau}$.

For representation theory on various groups, most notably the symmetric group S_n, see James and Kerber [67], Kerber [75, 76], and Simon [117]. A package for constructive representation theory is [41].

4.3. **The General Fourier Transform.** With the machinery above, we can define the general Fourier transform which works for any finite group, abelian or nonabelian.

Definition 4.1 (Fourier Transform over a finite group)**.** *Let G be a finite group of order N, $f : G \to \mathbb{C}$ any map of sets. For an irreducible representation ρ of G of dimension d_ρ, define* ***the Fourier transform of*** *f* ***at*** *ρ to be*

$$\hat{f}(\rho) = \sqrt{\frac{d_\rho}{N}} \sum_{g \in G} f(g)\rho(g) \tag{73}$$

Let $\hat{G}$ be a complete set of irreducible representations of G. We define ***the inverse Fourier transform of*** *$\hat{f}$ to be*

$$f(g) = \sqrt{\frac{1}{N}} \sum_{\rho \in \hat{G}} \sqrt{d_\rho} tr\left(\hat{f}(\rho)\rho(g^{-1})\right) \tag{74}$$

To ensure this definition makes sense, we check that the $f(g)$ in the definition of the inverse is actually the f we started with, by substituting the definition of $\hat{f}$ in the definition for the inverse, and swapping the order of summation, obtaining

$$\frac{1}{N} \sum_{g' \in G} f(g') \sum_{\rho \in \hat{G}} d_\rho \mathrm{tr}\left(\rho(g'g^{-1})\right) = f(g), \tag{75}$$

where we note the rightmost sum is 0 by equation 72 unless $g' = g$, in which case that sum is N, so the equality follows. Thus the definition agrees with the initial f.

To understand this as a Fourier transform, we associate f and $\hat{f}$ with vectors in $\mathbb{C}^N$, and examine the map $\Gamma : f \to \hat{f}$. To do this, fix an ordering of $G =$

$\{g_1, g_2, \ldots, g_N\}$, and then f is equivalent to a vector we also label f,

$$f = (f(g_1), f(g_2), \ldots, f(g_N)) \in \mathbb{C}^N.$$

To view $\hat{f}$ as a vector in $\mathbb{C}^N$, we need more choices. Fix an ordering $\hat{G} = \{\rho_1, \rho_2, \ldots, \rho_m\}$, let $d_k = d_{\rho_k}$, and for each $\rho_k : G \to \mathrm{GL}(\mathbb{C}^{d_k})$ fix a basis of $\mathbb{C}^{d_k}$, so each $\hat{f}(\rho_k)$ is a $d_k \times d_k$ matrix. Since $\sum_\rho d_\rho^2 = N$, there are N matrix entries, which we order. For brevity label the matrix entry $\hat{f}(\rho_k)_{ij} = \hat{f}_{ijk}$. Then we can associate $\hat{f}$ with a vector

$$\hat{f} = \left(\hat{f}_{111}, \hat{f}_{121}, \ldots, \hat{f}_{d_N d_N m}\right) \in \mathbb{C}^N.$$

Viewing $\Gamma : f \to \hat{f}$ as a map from $\mathbb{C}^N$ to itself, it is not hard to show Γ is linear. It is a good exercise to show Γ is a unitary transformation when viewed this way.

Note in the finite abelian case each irreducible representation is one dimensional, so each $d_\rho = 1$, and then the only representations are given by the characters in section 3.5. Then Γ becomes the finite abelian Fourier transform, and this definition generalizes the definition given earlier.

4.4. **The Standard HSP Algorithm - Quantum Fourier Sampling.** We now cover what is called the standard algorithm for finding hidden subgroups of a given group. The complexity and qubit requirements depend on the group in question; we will cover what is known in section 5. This section follows Hallgren [59] and Grigni, Schulman, Vazirani, and Vazirani [53].

The process about to be described is called **Quantum Fourier Sampling**, or **QFS** for short. It is the process of preparing a quantum state in a uniform superposition of states indexed by a group, then performing an oracle function, then a quantum Fourier transform, and finally sampling the resulting state to gather information about subgroups hidden by the oracle.

We first note the standard finite abelian group case can be summarized as:
[**Algorithm 1**]

(1) Compute $\frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle |f(g)\rangle$ and measure the second register $f(g)$. The resulting superposition is then $\frac{1}{\sqrt{|H|}} \sum_{h \in H} |ch\rangle |f(ch)\rangle$ for some uniformly chosen coset cH of H.
(2) Compute the Fourier transform of the coset state, obtaining in the first register
$$\sum_{\rho \in \hat{G}} \frac{1}{\sqrt{|G||H|}} \sum_{h \in H} \rho(ch)|\rho\rangle$$
where $\hat{G}$ is the set of (irreducible) representations[24] $\{\rho : G \to \mathbb{C}\}$.
(3) Measure the register, and observe a representation ρ. This gives information about H.
(4) Classically process the information from the previous step to determine the hidden subgroup H.

We can generalize this to handle the nonabelian and abelian cases in one framework via

[**Algorithm 2**]

[24] In the abelian case these are the same as the characters.

(1) Compute $\frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle |f(g)\rangle$ and measure the second register $f(g)$. The resulting superposition is then $\frac{1}{\sqrt{|H|}} \sum_{h \in H} |ch\rangle |f(ch)\rangle$ for some uniformly chosen coset cH of H.
(2) Compute the Fourier transform of the coset state, obtaining in the first register

$$\sum_{\rho \in \hat{G}} \sum_{i}^{d_\rho} \sum_{j}^{d_\rho} \frac{\sqrt{d_\rho}}{\sqrt{|G||H|}} \left(\sum_{h \in H} \rho(ch) \right)_{i,j} |\rho, i, j\rangle$$

where $\hat{G}$ is the set of (irreducible) representations $\{\rho : G \to \mathbb{C}\}$.
(3) ***Weak form***: Measure the register, and observe a representation ρ. This gives information about H.
Strong form: Measure the register, and observe a representation ρ as well as matrix indices i and j. This gives information about H.
(4) Classically process the information from the previous step to determine the hidden subgroup H.

This algorithm gives information useful for finding generators of the hidden subgroup H. Ignoring the problem of engineering the physical quantum computer, there are three theoretical obstacles to making this algorithm efficient for a given family of nonabelian groups. They are:

(1) We need an efficient way to compute the QFT over the groups in question, similar to the way that equation 36 led to an efficient quantum circuit computing the QFT over $\mathbb{Z}_{2^n}$. Beals [14] constructs an efficient QFT for the symmetric groups, and Diaconis and Rockmore [39] construct efficient classical Fourier transforms over many other groups. For more information on the QFT see [90, 92, 91, 89] and section 5. Efficient QFT quantum circuits are not known for all finite groups.
(2) We need to choose a basis for the irreducible representations $\rho \in \hat{G}$. For the abelian case, the irreducible representations are one dimensional characters, so the basis choices are canonical, so this step is trivial. However, in the nonabelian case some bases may give better results. For example, it is known the standard method cannot solve the HSP over S_n if the basis choice is random - it will take a clever basis choice for the irreducibles to obtain an efficient algorithm.
(3) We need an efficient way to reconstruct the subgroup generators for H from the irreducible representations returned. For the abelian case this is efficient since they are canonical and computing the GCD and solving linear systems mod d are efficient classically as explained in section 3.5. However, this reconstruction is harder in the nonabelian case. For example, Ettinger, Høyer, and Knill [46] have shown only polynomially many calls in $\log |G|$ to the oracle distinguishes subgroups for any group G information theoretically, but it is currently unknown how to extract generators for H without *exponential* classical postprocessing time.

One immediate question is if the weak and strong forms are equivalent. Section 5 shows the strong form can distinguish between certain subgroups which the weak form cannot. The reason is roughly that conjugate subgroups determine the

same statistics on representations, but not on rows and columns, which gives more information. However, there are still cases where the weak form is good enough.

The next question is to ask which groups have efficient HSP algorithms, and are there any groups for which the HSP cannot be solved efficiently?

These questions are ongoing research problems, and there are partial results showing which groups are likely to be efficiently solvable, and some negative results showing limitations of this approach. The next section covers many known results and current research directions.

For more reading on the (classical) computation of FFT's over finite groups, see Babai and Ronyai [5], Baum [9], Baum and Clausen [10, 11, 12], Baum, Clausen, and Tietz [13], Rockmore [108, 109, 110], and Terras [122].

5. Nonabelian Results

5.1. **Overview.** In this section we present results about the HSP over finite nonabelian groups. Throughout this section we fix notation: G is a member of a family of finite groups $\mathbf{G} = \{G_i\}$ that should be clear from context, and H is a subgroup of G. The size n of the problem is $n = \lceil \log |G| \rceil$ or sometimes $n = O(\log |G|)$, also clear from context. We say a quantum algorithm is *efficient* in either case if the circuit size is polynomial in n as G varies through the family.

We also divide families of groups into three classes (following Moore, Rockmore, Russell, and Schulman[97]):

- **I. Fully Reconstructible**. Subgroups of a family of groups $\mathbf{G} = \{G_i\}$ are *fully reconstructible* if the HSP on G_i can be solved with probability $> \frac{3}{4}$ by a quantum circuit of size polynomial in $\log |G_i|$.
- **II. Measurement Reconstructible**. Subgroups of a family of groups $\mathbf{G} = \{G_i\}$ are *measurement reconstructible* if the solution to the HSP on G_i is determined information-theoretically using the fully measured result of a quantum circuit of size polynomial in $\log |G_i|$.
- **III. Query Reconstructible**. Subgroups of a family of groups $\mathbf{G} = \{G_i\}$ are *query reconstructible* if the solution to the HSP for G_i is determined by the quantum state resulting from a quantum circuit of size polynomial in $\log |G_i|$, in the sense that there is a POVM that yields the subgroup H with constant probability. There is no guarantee that this POVM can be implemented by a small quantum circuit.

A primary goal of quantum algorithm research is to move groups into lower numbered classes, and the driving force is to place all finite groups in class I. Currently very few group families are class I, but we will see all finite groups are in class III, with some moving up to class II and I. This is contrasted with what we saw above: all finite abelian groups are in class I. We will see examples of each of the three classes below.

5.2. **A Necessary Result.** In order to find an efficient quantum algorithm for a given family, it is necessary that $O(\text{poly}(n))$ oracle queries suffices. Fortunately this has been shown possible for any finite group by Ettinger, Høyer, and Knill[46, 47]. They prove that polynomially many oracle queries in n distinguishes subgroups information theoretically. They do this by creating the state

$$|\psi\rangle = \frac{1}{\sqrt{|G|^m}} \sum_{(g_1,g_2,\ldots,g_m)\in G^m} |g_1, g_2, \ldots, g_m\rangle |f(g_1), f(g_2), \ldots, f(g_m)\rangle \tag{76}$$

which requires m oracle queries. Taking $m = \lceil 4n + 2 \rceil$ results in a state from which H can be extracted with high probability, unfortunately requiring $O(|G|)$ operations to do so. Precisely, they prove

Theorem 5.1. *Let G be a finite group, and f an oracle function on G which separates a subgroup H. Then there exists a quantum algorithm that calls the oracle function $\lceil 4 \log |G| + 2 \rceil$ times and outputs a subset $X \subseteq G$ such that $X = H$ with probability at least $1 - 1/|G|$.*

So for any finite group G and subgroup H it is possible to gather enough information to determine H using only $O(\text{poly}(\log |G|))$ queries of f, thus placing all finite groups in class III. Their proof is reproduced in section 5.7 since it is foundational.

5.3. **The Dihedral Group D_N.** Many attempts have been made to find an efficient HSP algorithm for the dihedral groups. One reason is that it one of the "simplest" nonabelian groups and is easily studied. Another reason is that they have exponentially many (in n) subgroups of small order, making classical algorithms infeasible[25]. A better reason is that an efficient HSP algorithm for the dihedral groups gives efficient algorithms for solving some classically hard lattice problems [107], which is covered below. Recall $\mathbb{Z}_N$ is a cyclic group on N elements[26]. Then we define the dihedral group $D_N = \mathbb{Z}_2 \ltimes \mathbb{Z}_N$ with $2N$ elements and with relations

$$x^N = y^2 = yxyx = 1. \tag{77}$$

5.3.1. *Equivalent Problems.* Before we start on dihedral group algorithms, we remark Kuperberg [82] lists equivalences between the Dihedral HSP (DHSP) and other problems. Precisely we define the DHSP as finding a hidden subgroup H that is either trivial or generated by a reflection $H = \langle x^s y \rangle$. This is equivalent to the general problem of determining subgroups of D_N as we outline below in section 5.3.2.

Next we define the *abelian hidden shift problem* to be: given an abelian group A, a set S, and two injective functions $f, g : A \to S$ that differ by a hidden shift s

$$f(v) = g(v + s) \tag{78}$$

and are otherwise distinct, then determine s (using quantum oracles f and g).

The DHSP is equivalent to the abelian hidden shift problem with $A = \mathbb{Z}_N$. If we define $h : D_N \to S$ by

$$h(x^n) = f(n) \qquad h(x^n y) = g(n), \tag{79}$$

then h hides the reflection $x^s y$. Solving the DHSP for h gives s, solving the shift problem. Conversely, given the DHSP $h : D_N \to S$, define f, g as in equation 79. Then a solution to the abelian hidden shift problem for f and g determines s, which determines the subgroup H hidden by h.

Generally, the a solution to the HSP on $G = \mathbb{Z}_2 \ltimes A$ where $\mathbb{Z}_2$ acts by inversion on A is equivalent to the abelian hidden shift problem on A.

The *cyclic hidden reflection problem* is: $h : \mathbb{Z}_N \to S$ satisfies

$$h(n) = h(s - n) \tag{80}$$

[25]For example, it takes exponentially many evaluations of f just to determine if H is nontrivial with probability bounded above $1/2$. This holds for the reasons in Simon[119]

[26]We could abstractly call C_N the cyclic group on N elements, but then $C_N \cong \mathbb{Z}_N$, not always canonically. We choose the concrete $\mathbb{Z}_N$.

and otherwise takes distinct values. We want to find s. This problem is equivalent to the DHSP; we show it equivalent to the abelian hidden shift problem as follows.

It reduces to the shift problem by defining the ordered pairs

$$f(n) = (h(-n), h(-n-1)) \qquad g(n) = (h(n), h(n+1)). \tag{81}$$

We need pairs to ensure f and g are injective. Then $f(n) = g(s+n)$ and are distinct otherwise, giving the reduction.

Conversely, if $f, g : \mathbb{Z}_N \to S$ are injective and

$$f(n) = g(s+n) \tag{82}$$

then we can define the unordered pairs

$$h(n) = \{f(-n), g(n)\}. \tag{83}$$

which reduces the hidden reflection problem to the shift problem. Note $h(n) = \{f(-n), g(n)\} = \{g(s-n), f(n-s)\} = \{f(-(s-n)), g(s-n)\} = h(s-n)$.

5.3.2. *Dihedral Results.* Now we cover what is known about the DHSP.

Ettinger and Høyer [45] show an algorithm that produces data sufficient to determine any hidden subgroup H in a dihedral group D_N, but it is unknown if this data can be post processed in $O(\text{poly}(n))$ time to reconstruct the subgroup H. This is stronger than the result in [43] since it returns the classical data from the quantum state. [43] only constructed a state determining H, but required exponential time to extract that information to classical information. Their algorithm exploits the normality of the (abelian) cyclic group $\mathbb{Z}_N < D_N$, and uses the abelian QFT to gather information which is then extended to determine the subgroup H. They reduce to the case of finding a subgroup H generated by a reflection. They prove

Theorem 5.2. *Let f be a function that separates H in the dihedral group D_N. There exists a quantum algorithm that uses $\Theta(\log N)$ evaluations of f and outputs a subset $X \subseteq H$ such that X is a generating set for H with probability at least $1 - \frac{2}{N}$.*

Following Ettinger and Høyer [45], we outline the proof that it is sufficient to solve the DHSP for the simpler case where H is either trivial or generated by a reflection. We want to find the hidden subgroup $H < D_N$, where we view D_N as the semidirect product $\mathbb{Z}_N \rtimes \mathbb{Z}_2$. Using the abelian QFT algorithm, we find $H_1 = H \cap \{\mathbb{Z}_N \times \{0\}\}$, which is normal in D_N. Then we work on the quotient group $D_N/H_1 \cong D_M$ with $M = [\mathbb{Z}_N \times \{0\} : H_1]$, and find H/H_1 which is either generated by a reflection $r + H_1$ or is trivial. Precisely,

Theorem 5.3. *Let f be a function that separates H in the dihedral group D_N, and suppose we are promised that $H = \{0\}$ is trivial or $H = \{0, r\}$ is generated by a reflection r. Then there exists a quantum algorithm that given f, outputs either "trivial" or the reflection r. If H is trivial , the output is always trivial, otherwise the algorithm outputs r with probability at least $1 - \frac{1}{2N}$. The algorithm uses at most $89 \log_2(N) + 7$ evaluations of f and it runs in time $O(\sqrt{N})$.*

Finally, Kuperberg [82] gives a subexponential time quantum algorithm for solving the dihedral HSP, using time and query complexity $O(\exp(C\sqrt{\log N}))$ for D_N. This is much better than the classical query complexity of $O(\sqrt{N})$. Unfortunately

this algorithm requires $\Theta(\exp(C\sqrt{\log N}))$ quantum space. Variants of this algorithm also work for the abelian hidden shift problem described above and for the hidden substring problem[27]. The main results are

Theorem 5.4. *There is an algorithm that finds a hidden reflection in the dihedral group $G = D_N$ (of order $2N$) with time and query complexity $O(\exp(C\sqrt{\log N}))$.*

Theorem 5.5. *The abelian hidden shift problem has an algorithm with time and query complexity $O(\exp(C\sqrt{n}))$ where n is the length of the output, uniformly for all finitely generated abelian groups.*

(Note this is even true for infinite groups; we only need finitely generated!)

Corollary 5.6. *The $N \hookrightarrow 2N$ hidden substring problem has an algorithm with time and query complexity $O(\exp(C\sqrt{\log N})$.*

5.4. **Groups with an Efficient QFT.** Next we turn to some other groups with an efficient QFT. To use the standard weak or strong form of the algorithm, we need to be able to compute efficiently the Fourier transform of a function over a given group. So in this section we list some of the groups for which efficient quantum Fourier transform algorithms are known.

Zalka [129] gives an algorithm for the HSP on wreath product groups $G = \mathbb{Z}_2^n \wr \mathbb{Z}_2$. The idea is similar to Ettinger and Høyer [45], in that it finds generators for an abelian subgroup in the desired subgroup, and then extends it.

Høyer [64] shows how to construct QFT for many groups: quaternions, a class of metacyclic[28] groups (up to phase), and a certain subgroup E_n of the orthogonal group $O(2^n)$ useful for quantum error correction [26].

Beth, Püschel, Rötteler, [20] show how to do the QFT efficiently on a class of groups - solvable 2 groups containing a cyclic normal subgroup of index 2 ($|G|$ is a power of 2 and solvable): They give reference to the fact for $n \geq 3$ there are exactly 4 isomorphism classes of such nonabelian groups of order 2^{n+1} with a cyclic subgroup of order 2^n:

- the dihedral group $D_{2^{n+1}} = \langle x, y \mid x^{2^n} = y^2 = 1, \ \ yxyx = 1 \rangle$,
- the quaternion group $Q_{2^{n+1}} = \langle x, y \mid x^{2^n} = y^4 = 1, \ \ y^3xyx = 1 \rangle$,
- the quasi-dihedral group $QD_{2^{n+1}} = \left\langle x, y \mid x^{2^n} = y^2 = 1, \ \ yxy = x^{2^{n-1}-1} \right\rangle$,
- the group $QP_{2^{n+1}} = \left\langle x, y \mid x^{2^n} = y^2 = 1, \ \ yxy = x^{2^{n-1}+1} \right\rangle$.

Beals [14] shows how to compute the QFT over S_n in time $O(\text{poly}(n))$, by adapting the methods of Clausen [30] and Diaconis-Rockmore [39] to the quantum setting.

Moore, Rockmore, and Russell [96] show how to construct efficient quantum Fourier transform circuits of size $O(\text{poly} \log |G|)$ for many groups, including

- the Clifford groups $\mathbb{CL}_n$,
- the symmetric group, recovering Beals algorithm [14],
- wreath products $G \wr S_n$, where $|G| = O(\text{poly}(n))$,
- metabelain groups (semidirect products of two abelian groups), including metacyclic groups such as the dihedral and affine groups, recovering the algorithm of Høyer [64],

[27] See the paper for a precise definition. It is basically a string matching algorithm.

[28] A group G is *metacyclic* if it contains a cyclic normal subgroup H such that the quotient group G/H is cyclic.

- bounded extensions of abelian groups such as the generalized quaternions, recovering the algorithm of Püschel et al. [20].

Their results also give subexponential size quantum circuits for the linear groups $\mathrm{GL}_k(q)$, $\mathrm{SL}_k(q)$, $\mathrm{PGL}_k(q)$, $\mathrm{PSL}_k(q)$, for a fixed prime power q, finite groups of Lie type, and the Chevalley and Weyl groups. Unfortunately, defining *polynomially uniform*, *adapted diameter*, *homothetic*, and *multiplicity* would take us too far afield; see their paper for details. These have to do with certain group items being efficiently computable. But we state their two main theorem anyway:

Theorem 5.7. *If G is a polynomially uniform group with a subgroup tower $G = G_m > G_{m-1} > \cdots > 1$ with adapted diameter D, maximum multiplicity M, and maximum index $I = \max_i[G_i : G_{i-1}]$, then there is a quantum circuit of size $\mathrm{poly}(I \times D \times M \times \log|G|)$ which computes the quantum Fourier transform over G.*

Theorem 5.8. *If G is a homothetic extension of H by an abelian group, then the quantum Fourier transform of G can be obtained using $O(\mathrm{poly}\log|G|)$ elementary quantum operations.*

5.5. **HSP Algorithms and Groups.**

5.5.1. *Group Definitions I.* H is a subgroup of G; let $N(H)$ or $N_G(H)$ be the *normalizer* of H in G. Let M_G be the intersection of all normalizers in G, i.e., $M_G = \bigcap_{H \le G} N(H)$. M_G is a subgroup of G and can be taken to be the size of how nonabelian G is ($[G : M_G] = 1$ for abelian groups). H^G is the largest subgroup of H that is normal in G, and is called the *normal core* of H.

Definition 5.9 (Wreath Product)**.** *The* **wreath product** *of two finite groups G and H is defined as follows. For $|H| = n$, view H a subgroup of the symmetric group S_n on n items. Let $P = G \times \cdots \times G$ be the direct product of n copies of G. The wreath product $G \wr H$ of G with H is a semidirect product $P \rtimes H$ with multiplication*

$$(g_1, \ldots, g_n; \tau)\,(g'_1, \ldots, g'_n; \tau') = \left(g_{\tau'(1)} g'_1, \ldots, g_{\tau'(n)} g_n; \tau\tau'\right) \tag{84}$$

That is, the permutations in H are composed as usual, but the right permutation permutes the left factors of P and then the n-tuple is multiplied pointwise. It is instructive to verify this operation forms a group.

5.5.2. *Normal Subgroups Can be Found in Any Group.* Hallgren, Russell, and Ta-Shma (2002) [59] prove that the natural extension of the abelian case algorithm finds H^G efficiently, the normal core of H. This also gives that normal subgroups can be found efficiently by the standard (weak or strong version of the) algorithm. In particular, this allows finding hidden subgroups in Hamiltonian groups (groups whose subgroups are all normal); the nonabelian Hamiltonian groups are of the form $\mathbb{Z}_2^k \times B \times Q$, where Q is the 8 element quaternion group and B is an abelian group with exponent[29] b coprime with 2. See Rotman [111, Exercise 4.28]. They show the probability of measuring a representation ρ is independent of the coset of H.

[29]Recall the exponent a of a group G is the smallest integer a such that $g^a = e$, the identity, for every element $g \in G$, if such an integer exists.

Theorem 5.10. *The probability of measuring the representation ρ in Algorithm 2 of section 4.4 is $d_\rho \frac{|H|}{|G|}$ times the number of times ρ appears in $\mathbf{Ind}_H^G \mathbf{1}_H$.*

They also obtain:

Theorem 5.11. *Let H be an arbitrary subgroup of G, and let H^G be the largest subgroup of H that is normal in G. With probability at least $1-2\exp(-\log_2|G|/8)$, H^G is determined by observing $O(\log|G|)$ independent trials of QFS.*

In fact, if $\rho_1, \ldots, \rho_m$ are the representations sampled by m repetitions of the algorithm, then $H^G = \bigcap_i \ker \rho_i$ with high probability.

They also show that weak QFS does not distinguish between order 1 and 2 subgroups in S_n:

Theorem 5.12. *For S_n, there is a subgroup H_n so that the weak QFS does not distinguish (even information theoretically) the case that the hidden subgroup is trivial from the case the hidden subgroup is H_n. Specifically, the distributions induced on representations in these two cases have exponentially small total variation distance.*

Theorem 5.13. *Let H be an arbitrary subgroup of G, and let H^G be the largest subgroup of H that is normal in G. With probability at least $3/4$, H^G is uniquely determined by observing $m = O(\log|G|)$ independent trials of Algorithm 2 of section 4.4 when H is the hidden subgroup. When H is normal, $H^G = H$, and this determines H.*

5.5.3. *"Almost Abelian" Subgroups Can be Found and Measuring Rows is Strong Enough.* Grigni, Schulman, Vazirani, and Vazirani [53] show another class of groups for which the HSP has an efficient quantum solution - what they call "almost abelian" groups. These are groups for which the intersection $M(G)$ of all the normalizers of all subgroups of G is large. For $n = \log|G|$, they require $[G : M(G)]$ (called the Baer norm [97]) to be of order $\exp O(\log^{1/2} n)$, and then the HSP can be solved if the QFT can be performed efficiently. In particular they show that the subgroups of the semidirect product $\mathbb{Z}_m \ltimes \mathbb{Z}_3$ for m a power of 2 can be found efficiently.

Another useful result in their paper shows that measuring both the row and column in the strong form of the QFT gives no more information than measuring just one of them (depending on how one lets the irreps act - left or right). This follows from the quantum mechanical requirement that the irreps are unitary matrices, and thus each matrix row (or column) has the same norm, which gets "absorbed."

Most importantly, they show that even using the strong form *with a random basis* for the irreps, the strong QFS algorithm cannot distinguish between the case of a trivial subgroup and an order two subgroup without exponentially many oracle queries.

The restriction on the size of $M(G)$ was extended by Gavinsky [51] to allow $[G : M(G)]$ to be of size $O(\text{poly}(n))$, allowing the corresponding HSP to be solved efficiently if the QFT over G can be. These groups are labelled "poly-near-hamiltonian groups." A final algorithm in this paper shows how to solve the HSP efficiently on poly-near hamiltonian groups *even when the QFT over the group G is not known to be efficient*, by using QFS over a hamiltonian group, which was shown to be efficient by a result from above.

5.5.4. *Strong is Indeed Stronger.* Moore, Rockmore, Russell, and Schulman [97] show that the strong form is indeed stronger, by exhibiting semidirect products $\mathbb{Z}_q \ltimes \mathbb{Z}_p$ (the *q-hedral* groups, which include the affine groups $A_p \cong \mathbb{Z}_p^* \ltimes \mathbb{Z}_p$) , where $q|(p-1)$ and $q = p/\text{polylog}(p)$, such that the strong form can determine hidden subgroups efficiently, but the weak form and "forgetful" abelian form cannot. They also prove a closure property for the class of groups over which the HSP can be solved efficiently:

Theorem 5.14. *Let H be a group for which hidden subgroups are fully reconstructible, and K a group of size polynomial in $\log|H|$. Then hidden subgroups in any extension of K by H, i.e. any group G with $K \triangleleft G$ and $G/K \cong H$, are fully reconstructible.*

They also place some groups in class I.

Theorem 5.15. *Let p be a prime, q a positive integer, and $G = \mathbb{Z}_q \ltimes \mathbb{Z}_p$. Then*

1. *if q is prime and $q = (p-1)/\text{polylog}(p)$, then subgroups of G are fully reconstructible (class I),*
2. *if q divides $p-1$, then hidden conjugates of H in G are fully reconstructible (class I) if H has index $\text{polylog}(p)$,*
3. *if q divides $p-1$, then hidden conjugates of H in G are measurement reconstructible (class II),*
4. *if q divides $p-1$, then subgroups the q-hedral groups G are measurement reconstructible (class II). In particular, the subgroups of the affine groups $A_p = \mathbb{Z}_{p-1}^* \ltimes \mathbb{Z}_p$ are measurement reconstructible (class II).*

For another direction studying the HSP over infinite groups, see Lomonaco and Kauffman [85]. They consider a version of the HSP for finding periods of functions over the real numbers $\mathbb{R}$, although it is not clear if these could be physically implemented due to $\mathbb{R}$ being an infinite set. They have a good overview of the HSP in [84].

Rötteler and Beth [112] give an efficient algorithm solving the HSP on wreath products $W_n = \mathbb{Z}_2^n \wr \mathbb{Z}_2$ (like Zalka) by giving quantum circuits for the QFT and showing how to reconstruct the subgroup efficiently from samples. It uses $O(n)$ queries of f and $O(\text{poly}(n))$ classical post processing time, putting these groups in Class I. It is similar to the method of Ettinger and Hoyer.

In [43] Ettinger and Høyer construct a quantum observable for the graph isomorphism problem. Given two graphs of n vertices and an integer m, they define a quantum state on $O(mn)$ qubits, that when observed, outputs "yes" with certainty if the graphs are isomorphic and "no" with probability at least $1 - \frac{n!}{2^m}$ if they are not isomorphic. It is unknown if this observable can be implemented efficiently.

Cleve and Watrous [33] show how to reduce the complexity and size of the QFT for $\mathbb{Z}_{2^n}$.

Theorem 5.16. *For any m there is a quantum circuit that exactly computes the QFT modulo 2^m that has size $O(m(\log m)^2 \log\log m)$ and depth $O(m)$.*

Theorem 5.17. *For any m and ϵ there is a quantum circuit that approximates the QFT modulo 2^m that has size $O(m \log(m/\epsilon))$ and depth $O(\log m + \log\log(1/\epsilon))$.*

They give an upper bound.

Theorem 5.18. *Any quantum circuit consisting of one- and two- qubit gates that approximates the QFT with precision* $\frac{1}{10}$ *or smaller must have depth at least* $\log n$.

5.5.5. *Lattice Problems.* Regev [107] shows that an efficient algorithm solving the HSP for dihedral groups would result in efficient algorithms for solving the Unique Shortest Vector Problem (SVP) and the subset-sum problem. First we sketch some definitions. A *lattice* is the set of all integral linear combinations of k linearly independent vectors in $\mathbb{R}^k$. This set of k vectors is called the *basis* of the lattice. The SVP is the problem of finding the shortest nonzero vector in this lattice, given the basis. In the $f(k)$-unique-SVP we are given the promise that the shortest vector is shorter by at least a factor of $f(k)$ from all other non-parallel vectors. We also define the Dihedral Coset problem (DCP). The input to the DCP for the dihedral group D_N of order $2N$ is a tensor product of polynomially many (in N) registers, each with the state $|0, x\rangle + |1, (x + d \pmod N)\rangle$ for some arbitrary $x \in \{0, 1, \ldots, N-1\}$, and d is the same for all registers. The goal is to find d. We say the DCP has failure parameter α if each of the registers with probability at most $\frac{1}{(\log n)^\alpha}$ is in the state $|b, x\rangle$ for arbitrary b. We take $N = k$, so the dihedral group size is determined by the dimension of the lattice. The main theorem is then

Theorem 5.19. *If there exists a solution to the DCP with failure parameter* α *then there exists a quantum algorithm that solves the* $\Theta(k^{\frac{1}{2}+2\alpha})$*-unique-SVP.*

Thus an efficient Dihedral HSP algorithm would give an efficient $f(k)$-unique-SVP algorithm.

5.5.6. *Distinguishable Subgroups of* S_n. Kempe and Shalev [74] analyze which subgroups of S_n can be distinguished efficiently using QFS. $H < S_n$ is *primitive* if it is transitive, and does not preserve a non-trivial partition of the permutation domain. They show

Theorem 5.20. *Let* $H \neq A_n, S_n$ *be a subgroup of* S_n*, with* H *a primitive subgroup. Then* H *is indistinguishable.*

Theorem 5.21. *A subgroup* $H < S_n$ *with property* Υ *(below) can be efficiently distinguished from the identity subgroup using either the weak or strong standard method with random basis only if it contains an element of constant support (i.e., a permutation in which all but a constant number of points are fixed). Property* Υ *can be any of the following*

- H *is of polynomial size,*
- H *is primitive.*

They also show other properties Υ for which the statement is true, and conjecture it is true for all subgroups of S_n. If their conjecture is true, which amounts to proving the following conjecture, then QFS with random basis provides no advantage over classical search. The *minimal degree* of a subgroup $H < S_n$ is defined to be the minimal number of points moved by a non-identity element of H. The *support* of an element is the number of points moved. Then the conjecture is

Conjecture 5.22. *Every subgroup* $H < S_n$ *with non-constant minimal degree has at most* $n^{k/7}$ *elements of support* k.

5.6. **Black-Box Group Algorithms.**

5.6.1. *Black-box Group Algorithms.* Black-box groups were introduced by Babai and Szemerédi in 1984 [6]. In the context of *black-box groups*, each group element is encoded as a length $n = O(\log |G|)$ string, and we assume group operations (multiplication, inverse, identity testing) are preformed by a *group oracle* (or *black-box*) in unit time. If each element is represented by a unique string this is called the *unique encoding* model, otherwise it is *not unique encoding.* A black-box group without unique encoding augmented by an oracle that can recognize any encoding of the identity element in unit time can compare elements for equality in unit time. Any efficient algorithm in the context of black-box groups remains efficient whenever the group oracle can be replaced by an efficient process. It is provably impossible to compute group orders in polynomial time in size log of the group, even for abelian groups. This becomes possible using quantum algorithms, as we will see. A black-box group G is defined by a set of m generators, each of length n bits, i.e., $G = \langle g_1, g_2, \ldots, g_m \rangle$. The quantity mn is called the *input size* for the group. Throughout this section on black-box group algorithms we reserve n to denote the length of the strings representing the finite group G, and all groups are finite.

5.6.2. *Group Definitions.* To state results for black-box group algorithms we need more definitions. Given a group G and elements $g, h \in G$, we define the *commutator* of g and h, denoted $[g, h]$, to be $[g, h] = g^{-1}h^{-1}gh$, and for any two subgroups $H, K \leq G$ we write $[H, K]$ to denote the subgroup of G generated by all commutators $[h, k]$ for $h \in H$ and $k \in K$. The *derived subgroup* (also known as the *commutator subgroup*) of G is $G' = [G, G]$, and we write

$$\begin{aligned} G^{(0)} &= G, \\ G^{(j)} &= \left(G^{(j-1)}\right)', \text{ for } j \geq 1. \end{aligned}$$

A group G is said to be *solvable* if $G^{(m)} = \{1\}$ (the trivial group) for some value of m.

A *composition series* for G is a sequence of subgroups of $G = G_1 \triangleright G_2 \cdots \triangleright G_t = 1$ such that G_{i+1} is normal in G_i, and the *factor groups* G_i/G_{i+1} are simple. The factor groups G_i/G_{i+1} are unique up to isomorphism and ordering. Beals and Babai [4] define $v(G)$ as the smallest natural number v such that every nonabelian composition factor of G possesses a faithful permutation representation of degree at most v. Thus for a solvable group $v(G) = 1$ (solvable implies factor groups are cyclic, hence abelian, hence have only trivial irreducible representations). It is known that $v(G)$ is polynomially bounded in the input size in many important cases, such as permutation groups or matrix groups over algebraic number fields.

A *presentation* of G is a sequence $g_1, \ldots, g_s$ of elements generating G, together with a set of group expressions in variables $x_1, \ldots, x_s$ called *relations*, such that $g_1, \ldots, g_s$ generate G and the kernel of the homomorphism from the free group $F(x_1, \ldots, x_s) \to G$ given by $x_i \to g_i$ is the smallest normal subgroup of F containing the relations. This gives a non-canonical yet very concrete description of G as the set of "strings" of the g_i and equivalence relations on those strings. Note the generators in the presentation may differ from the original generators given for G.

A *nice representation* of a factor group G_i/G_{i+1} means a homomorphism from G_i with kernel G_{i+1} to either a permutation group of degree polynomially bounded in the input size + $v(G)$ or to $\mathbb{Z}_p$, where p is a prime dividing $|G|$.

The *exponent* of a group is the smallest integer m such that $g^m = e$ for all $g \in G$. Lagrange's theorem gives $m \leq |G|$.

An abelian group (family) is *smoothly abelian* if it can be decomposed into the direct product of a subgroup of bounded exponents and a subgroup of polylogarithmic size in the order of the group. A solvable group (family) is *smoothly solvable* if its derived series is of bounded length and has smoothly abelian factor groups.

A *constructive membership test* is the following: given pairwise commuting group elements $h_1, h_2, \ldots, h_r, g$ of a group G, either express g as a product of powers of the h_i's or report that no such expression exists.

5.6.3. *Results.* Our first result [28], the basis for many later ones, allows computing a canonical decomposition of a finite abelian group from a generating set in polynomial time, i.e.,

Theorem 5.23 (Cheung, Mosca)**.** *Given a finite abelian black-box group G with unique encoding, the decomposition of G into a direct sum of cyclic groups of prime power order can be computed in time polynomial in the input size by a quantum computer.*

Watrous [125] shows how to construct quantum certificates proving group nonmembership efficiently, and shows this is not possible classically.

Watrous [126] gives a polynomial-time quantum algorithm for computing the order of a solvable group, which gives polynomial-time algorithms for membership testing of an element in a subgroup, testing subgroup equality given two descriptions of the subgroups, and testing subgroup normality, each for solvable groups. The main result is

Theorem 5.24 (Group Order)**.** *Given a finite, solvable black-box group G, there exists a quantum algorithm that outputs the order of G with probability of error bounded by ϵ in time polynomial in the input size $+ \log(1/\epsilon)$. The algorithm produces a quantum state ϕ that approximates the state $|G\rangle = |G|^{-1/2} \sum_{g \in G} |g\rangle$ with accuracy ϵ in the trace norm metric.*

This result was also obtained using a different algorithm by Ivanyos *et. al.* [66] in a paper extending many of the black-box group results from Beals-Babai [4] to the quantum setting. They obtain

Theorem 5.25. *Let G be a finite black-box group with not necessarily unique encoding. Assume the following are given:*

(a) *an oracle for computing the orders of elements of G,*
(b) *an oracle for the constructive membership tests in elementary abelian subgroups of G.*

Then the following tasks can be solved by quantum algorithms of running time polynomial in the input size$+v(G)$:

(1) *constructive membership tests in subgroups of G,*
(2) *computing the order of G and a presentation for G,*
(3) *finding generators for the center of G,*
(4) *constructing a composition series $G = G_1 \rhd G_2 \rhd \cdots \rhd G_t = 1$ for G, together with nice representations of the composition factors G_i/G_{i+1},*
(5) *finding Sylow subgroups of G.*

The hypotheses (a) and (b) can be met in many cases. For example, using Shor's order finding method to compute element orders, they give:

Theorem 5.26. *Assume G is a black-box group with unique encoding. Then each task in theorem 5.25 can be solved in time polynomial in the input size + $v(G)$ by a quantum algorithm.*

Theorem 5.27. *Assume G is a black-box group with not necessarily unique encoding, and that N is a normal subgroup given as a hidden subgroup of G (i.e., there is a f hiding N). Then there are quantum algorithms each with running time polynomial in the input size + $v(G/N)$ that perform:*

- *all the tasks in theorem 5.25 for G/N,*
- *finding generators for N. In particular, we can find hidden normal subgroups of solvable black-box groups and permutation groups in polynomial time in input size + $v(G/N)$ (note we do not need an efficient QFT as in Hallgren* et. al. *[59]),*

If instead of giving N as a hidden subgroup, if N is given by generators, and N is solvable or of polynomial size, then all the tasks listed in theorem 5.25 can be solved for G/N in time polynomial in the input size + $v(G)$.

Theorem 5.28. *Let G be a black-box group with unique encoding. The HSP can be solved by a quantum algorithm in time polynomial in the input size + $|G'|$, the size of the commutator subgroup of G.*

This includes the wreath products $\mathbb{Z}_2^k \wr \mathbb{Z}_2$ of Rötteler and Beth [112].

A question remains: the above proofs only use the abelian QFT to get the results. Does using the nonabelian QFTs give better results?

Friedl *et. al.* [49] introduced the *Orbit Coset* problem as a generalization of the hidden subgroup and hidden shift[30] problems. Hidden shift was defined above in section 5.3.1. As mentioned there, when G is abelian, hidden shift is equivalent to the HSP in the semidirect product $G \rtimes \mathbb{Z}_2$.

Definition 5.29 (*Orbit Coset and Orbit Superposition*)**.** *Let G be a finite group acting on a finite set Γ of mutually orthogonal quantum states.*

- *Given generators for G and two quantum states $|\phi_0\rangle, |\phi_1\rangle \in \Gamma$, the problem* **Orbit Coset** *is to either reject the input if $G(|\phi_0\rangle) \cap G(|\phi_1\rangle) = \emptyset$, or output a generating set for $G_{|\phi_1\rangle}$ of size $O(\log |G|)$ and a $u \in G$ such that $|u \cdot \phi_1\rangle = |\phi_0\rangle$.*
- *Given generators for G and a quantum state $|\phi\rangle \in \Gamma$, the problem* **Orbit Superposition** *is to construct the uniform superposition $|G \cdot \phi\rangle = \frac{1}{\sqrt{|G(|\phi\rangle)|}} \sum_{|\phi'\rangle \in G(|\phi\rangle)} |\phi'\rangle$*

Theorem 5.30. *Let p be a fixed prime. Then*

- *the problem of hidden shift over $\mathbb{Z}_p^m$ can be solved in quantum polynomial time,*
- *the problem of Hidden Subgroup over $\mathbb{Z}_p^m \rtimes \mathbb{Z}_2$ can be solved in quantum polynomial time.*

This gives that $Z_p^m \rtimes \mathbb{Z}_2$ is class I for any prime p.

[30] Hidden shift is called hidden translation in their paper.

Theorem 5.31. *Let G be a smoothly solvable group and let α be a group action of G. When $t = (\log^{\Omega(1)} |G|) \log(1/\epsilon)$, Orbit Coset can be solved in G for α^t in quantum time* $\mathrm{poly}(\log |G|) \log(1/\epsilon)$ *with error ϵ.*

Using this they then show

Theorem 5.32. *Hidden shift can be solved over smoothly solvable groups in quantum polynomial time. HSP can be solved in solvable groups having smoothly solvable commutator subgroups quantum polynomial time.*

Fenner and Zhang [48] also address black-box group algorithms, obtaining efficient quantum algorithms for a few classically hard problems, by reducing them to Orbit Coset problems. The problems they study are Group Intersection (given two subsets S_1 and S_2 of a group, determine if the groups $\langle S_1 \rangle \cap \langle S_2 \rangle \neq \emptyset$), Coset Intersection (given two subsets S_1 and S_2 of a group and a group element g, determine if $\langle S_1 \rangle g \cap \langle S_2 \rangle \neq \emptyset$), and Double-Coset Membership (given two subsets S_1 and S_2 of a group and group elements g, h, determine if $g \in \langle S_1 \rangle h \langle S_2 \rangle$).

They obtain

Theorem 5.33. *Group Intersection over solvable groups can be solved efficiently in quantum polynomial time if one of the underlying solvable groups has a smoothly solvable commutator subgroup.*

Theorem 5.34. *Group Intersection over solvable groups is reducible to Orbit Superposition in quantum polynomial time.*

Theorem 5.35. *Coset Intersection and Double-Coset Membership over solvable groups can be solved in quantum polynomial time if one of the underlying groups is smoothly solvable.*

van Dam, Hallgren, and Ip [62] work on a hidden shift problem They first obtain a superposition result (ignoring the normalization constant):

Theorem 5.36. *Let $f : G \to \mathbb{C}$ be a complex valued function defined on the set G such that $f(x)$ has unit magnitude whenever $f(x)$ is nonzero. Then there is an efficient algorithm for creating the superposition $\sum_x f(x)|x\rangle$ with success probability equal to the fraction of x such that $f(x)$ is nonzero and that uses only two queries to the function f.*

The proof idea computes the state $\sum_x |x\rangle|f(x)\rangle$, tests if $f(x)$ is nonzero, moves the phase of $|f(x)\rangle$ into $|x\rangle$ to high precision, and then applies the second f to undo the first.

Let m be an integer, $m = p_1^{s_1} p_2^{s_2} \dots p_k^{s_k}$, then by the Chinese Remainder Theorem, $(\mathbb{Z}/m\mathbb{Z})^* \cong (\mathbb{Z}/p_1^{s_1}\mathbb{Z})^* \times (\mathbb{Z}/p_2^{s_2}\mathbb{Z})^* \dots (\mathbb{Z}/p_k^{s_k}\mathbb{Z})^*$. A multiplicative character χ on $\mathbb{Z}/m\mathbb{Z}$ can be written as $\chi(x) = \chi_1(x_1)\chi_2(x_2)\dots\chi_k(x_k)$ using this isomorphism, where $\chi_i(x_i)$ is a multiplicative character on $(\mathbb{Z}/p_i^{s_i}\mathbb{Z})^*$. We say χ is *completely nontrivial* if each χ_i is nontrivial. With this definition, they then solve some shifted character problems:

Theorem 5.37. *Given a nontrivial (resp. completely nontrivial) multiplicative character χ of a finite field $\mathbb{F}_q$ (where $q = p^r$ for some prime p) (resp. over $\mathbb{Z}/m\mathbb{Z}$), and a function f for which there is a shift s with $f(x) = \chi(x + s)$ for all $x \in \mathbb{F}_q$ (resp. $x \in \mathbb{Z}/m\mathbb{Z}$). Then there is an efficient quantum algorithm finding s with probability $1 - 1/q^2$ (resp. $(\frac{\phi(m)}{m})^3 = \Omega((\frac{1}{\log\log m})^3)$).*

In the case where m is unknown, this can still be done given a bound on m.

5.7. **Hidden Subgroups are Distinguishable.** In this section we show that at least information theoretically, it is possible to find any hidden subgroup H of a finite group G with only $\lceil 4\log|G|+2\rceil$ calls to the oracle function f, following [46] and done differently in [47]. Unfortunately, deducing H from the resulting quantum state requires exponential classical time, and it is still open for which groups this can be reduced to a polynomial time quantum algorithm. The idea is to create a quantum state that contains enough information to deduce H using few oracle calls, and then use $|G|$ applications of various measurements to this state to query each element of G. The technical work is to prove the measurements do not perturb the state too much, which would destroy information needed for later queries.

Precisely we prove:

Theorem 5.38. *Given a finite group G and an oracle function $f : G \to X$ to a set X, such that f separates cosets of a subgroup $H < G$ (f "hides" H). Then there exists a quantum algorithm that calls the oracle function $\lceil 4\log|G|+2\rceil$ times and outputs a subset $S \subseteq G$, such that $S = H$ with probability at least $1 - 1/|G|$.*

Proof. Fix a positive integer m. We work over the Hilbert space $\mathcal{H}$ of dimension $|G|^m$, with orthonormal basis indexed by m-tuples of elements of G. For any subset $S = \{s_1, s_2, \ldots, s_k\} \subseteq G$ let $|S\rangle$ be the normalized superposition $|S\rangle = \frac{1}{\sqrt{k}}(|s_1\rangle + \ldots |s_k\rangle)$. The first step is to prepare on $\mathcal{H}\otimes\mathcal{H}$ the state

$$\frac{1}{\sqrt{|G|^m}} \sum_{g_1,\ldots,g_m \in G} |g_1,\ldots,g_m\rangle|f(g_1),\ldots,f(g_m)\rangle \tag{85}$$

where we define $|f(g_i)\rangle = |g_i H\rangle$. Note this required m calls to the function f. Observing the second register leaves in the first register the state $|\Psi\rangle$ which is a tensor product of random left cosets of H, uniformly distributed. We ignore the second register for the rest of this proof. Let $|\Psi\rangle = |a_1 H\rangle \otimes \cdots \otimes |a_m H\rangle$ denote the first register, where the $a_i \in G$. For any (ordered) subset $\{b_1,\ldots,b_m\} \subseteq G$ and subgroup $K \leq G$ define

$$|\Psi(K,\{b_i\})\rangle = |b_1 K\rangle \otimes |b_2 K\rangle \otimes \cdots \otimes |b_m K\rangle \tag{86}$$

The key lemma, lemma 5.39, shows for $K \not\leq H$ that $\langle\Psi|\Psi(K,\{g_i\})\rangle$ is exponentially small for any m of the g_i.

Let $\mathcal{H}_K$ be the subspace of $\mathcal{H}$ spanned by all vectors of the form $|\Psi(K,\{g_i\})\rangle$ for all subsets $\{g_1,\ldots,g_m\} \subseteq G$. Let P_K be the projection operator[31] onto $\mathcal{H}_K$, and $P_K^\perp$ the projection onto the orthogonal complement of $\mathcal{H}_K$ in $\mathcal{H}$. Define the observable $A_K = P_K - P_K^\perp$, and fix an ordering $g_1, g_2, \ldots, g_{|G|}$ of G.

The algorithm then works as follows: First apply $A_{\langle g_1\rangle}$ to $|\Psi\rangle$, where $\langle g\rangle \leq G$ denotes the cyclic subgroup generated by $g \in G$. If the outcome is -1, then we know $g_1 \notin H$ with certainty, and if the outcome is $+1$ we know $g_1 \in H$ with high probability, by lemma 5.39. We then apply $A_{\langle g_2\rangle}$ to the state resulting from the first measurement. Continuing in this manner, we test all elements of G for membership in H by sequentially applying $A_{\langle g_2\rangle}, A_{\langle g_3\rangle}$, and so on to the resulting states of the previous measurements. Of course if we discover $g \in H$ then we can omit the tests for $g^j \in H$. Note we may have to apply $O(|G|)$ operations to test each element, making the algorithm complexity exponential in $\log|G|$. All that remains to show is that each measurement alters the state insignificantly with high probability, so

[31] Thus $P_K = \sum_{(b_1,\ldots,b_m)\in G^m} |\Psi(K,\{b_i\})\rangle\langle\Psi(K,\{b_i\})|$

that by the final operator $A_{\langle g_{|G|}\rangle}$ we have identified with high probability exactly which elements are in H and which are not.

We bound this probability of success. Let $|\Psi_0\rangle = |\Psi\rangle$. For $1 \leq i \leq |G|$, define the unnormalized states

$$|\Psi_i\rangle = \begin{cases} P_{\langle g_i\rangle}|\Psi_{i-1}\rangle & \text{if } g_i \in H \\ P^{\perp}_{\langle g_i\rangle}|\Psi_{i-1}\rangle & \text{if } g_i \notin H \end{cases} \tag{87}$$

By induction and the definition of the probabilities, $\langle\Psi_i|\Psi_i\rangle$ equals the probability that the algorithm given above answers correctly whether $g_j \in H$ for all $1 \leq j \leq i$. Now for all $0 \leq i \leq |G|$ let $|E_i\rangle = |\Psi\rangle - |\Psi_i\rangle$ denote the error between the original state and the desired state after testing $\langle g_i\rangle$.

Since $|\Psi_{|G|}\rangle = |\Psi\rangle - |E_{|G|}\rangle$, using $\langle E_{|G|}|E_{|G|}\rangle \leq \frac{|G|^2}{2^m}$ by lemma 5.40 and the triangle inequality gives that the probability for correctly determining all the elements of H is bounded below by $\langle\Psi_{|G|}|\Psi_{|G|}\rangle \geq 1 - \frac{2|G|}{2^{m/2}}$.

By choosing $m = \lceil 4\log|G| + 2\rceil$ the main theorem follows directly. □

Lemma 5.39. *Use the notation above. Let $K \leq G$. If $K \not\leq H$ then $\langle\Psi|P_K|\Psi\rangle \leq \frac{1}{2^m}$. If $K \leq H$ then $\langle\Psi|P_K|\Psi\rangle = 1$.*

Proof. Let $|H \cap K| = d$. Note that for all $g_1, g_2 \in G$ we have $|g_1H \cap g_2K| = d$ or $|g_1H \cap g_2K| = 0$. This implies that if $|g_1H \cap g_2K| = d$ then $\langle g_1H|g_2K\rangle = d/\sqrt{|H||K|}$. Therefore for any subset $\{b_1, \ldots, b_m\} \subseteq G$

$$\langle\Psi|\Psi(K,\{b_i\})\rangle = \begin{cases} \left(\frac{d}{\sqrt{|H||K|}}\right)^m & \text{if } |a_iH \cap b_iK| = d \text{ for } i = 1, 2, \ldots, m \\ 0 & \text{otherwise} \end{cases} \tag{88}$$

There exist exactly $(|H|/d)^m$ vectors of the form $|\Psi(K,\{b_i\})\rangle$ with $\langle\Psi|\Psi(K,\{b_i\})\rangle$ nonzero. Hence $\langle\Psi|P_K|\Psi\rangle = \left(\frac{|H|}{d}\right)^m \left(\frac{d^2}{|H||K|}\right)^m = \left(\frac{d}{|K|}\right)^m$. If $K \not\leq H$ then $d/|K| \leq 1/2$ and if $K \leq H$ then $d = K$. □

Lemma 5.40. *For all $0 \leq i \leq |G|$ we have $\langle E_i|E_i\rangle \leq \frac{i^2}{2^m}$.*

Proof. Proof by induction on i. Since $|\Psi_0\rangle = |\Psi\rangle$, by definition $|E_0\rangle = 0$. Now suppose $\langle E_i|E_i\rangle \leq \frac{i^2}{2^m}$. If $g_{i+1} \in H$, then $|\Psi_{i+1}\rangle = P_{\langle g_{i+1}\rangle}(|\Psi\rangle - |E_i\rangle) = |\Psi\rangle - P_{\langle g_{i+1}\rangle}|E_i\rangle$. Hence $\langle E_{i+1}|E_{i+1}\rangle \leq \langle E_i|E_i\rangle \leq \frac{i^2}{2^m}$. If $g_{i+1} \notin H$, then $|\Psi_{i+1}\rangle = P^{\perp}_{\langle g_{i+1}\rangle}(|\Psi\rangle - |E_i\rangle) = |\Psi\rangle - P_{\langle g_{i+1}\rangle}|\Psi\rangle - P^{\perp}_{\langle g_{i+1}\rangle}|E_i\rangle$. By lemma 5.39 we then have $\langle E_{i+1}|E_{i+1}\rangle = \langle\Psi|P_{\langle g_i\rangle}|\Psi\rangle + \langle E_i|E_i\rangle \leq \frac{1}{2^m} + \frac{i^2}{2^m} \leq \frac{(i+1)^2}{2^m}$. □

6. Conclusion

In conclusion, we have shown in great detail how to find hidden subgroups in any finite abelian group. This was shown to be efficient using a quantum computer, and is the basis for Shor's factoring algorithm, as well as many other exponentially faster quantum algorithms. The key ingredient was Fourier sampling - that is, doing a quantum Fourier transform on a state encoding the hidden subgroup, and then measuring (sampling) the resulting state to gather information used to compute the hidden subgroup generators.

Also, we described the nonabelian case of the HSP, using representation theory to define the Fourier transform over arbitrary finite groups, and then mimicking the abelian case in an attempt to solve the HSP efficiently for any finite group.

However this case is much harder, and only partial results are known, many of which we listed.

The main open problem in the field is finding an efficient quantum algorithm for the symmetric group S_n, which would yield an elusive (for over 30 years) efficient algorithm for determining graph isomorphism. However it seems that quantum Fourier sampling may not be up to the task since there are many negative results. Yet there is hope that a clever basis choice for the irreducible representations might turn this around. A second possibility, also seemingly remote, is finding a new quantum algorithm which does the trick, avoiding Fourier sampling completely.

6.1. **Other Quantum Algorithms.** There are many other areas where quantum algorithms are better than classical ones. One of the earliest algorithms was Grover's searching algorithm [54], which reduces the classical complexity of searching an unordered list of N items from $O(N)$ to a provably best quantum $\Theta(\sqrt{N})$ oracle queries[32]. See also [21]. This was exploited by [106] to make a quantum string matching algorithm much faster the best classical algorithms given in [80, 22].

Other quantum algorithms are found in [1, 24, 40, 58, 60, 61, 62, 70, 73, 124]. More quantum algorithm overviews are in [7, 17, 32, 50, 86, 99, 116]. Continuous variable algorithms are considered in [85, 104, 105]. A good point to start learning quantum error correction is [26].

Another interesting direction is taken by Orús, Latorre, and Martín-Delgado in [101, 102] where the authors notice an invariant of efficient quantum algorithms labelled "majorization," which they use to seek new algorithms.

A final direction is adiabatic quantum computation [123], another quantum computation computing model that may be physically realizable. It has recently been shown to be equivalent to the standard qubit model [3], but provides another viewpoint for quantum computation.

[32]Many authors claim $O(\sqrt{N})$ is the algorithm *time* complexity. A careful look shows $O(\sqrt{N}\ \log N)$ is a more reasonable time complexity.

These appendices contain results used above.

Appendix A. The Cyclic Quantum Fourier Transform over $\mathbb{Z}_N$

Here we give details on the cyclic QFT over $\mathbb{Z}_{2^n}$ and over $\mathbb{Z}_N$ for N odd.

A.1. **The Quantum Fourier Transform over $\mathbb{Z}_{2^n}$.** This section follows Coppersmith [35]. Since we already showed how to do the QFT over $\mathbb{Z}_{2^n}$ in section 3.4.2, we only have to cover the approximate QFT. The main result is

Theorem A.1. *Given an $\epsilon > 0$ and a positive integer n, let $N = 2^n$. Then there is a quantum circuit approximating the Fourier transform over $\mathbb{Z}_N$ using $O(\log N(\log\log N + \log(1/\epsilon)))$ 2-qubit operations. The approximated quantum state $|\phi\rangle$ differs from the true Fourier transformed state $|\psi\rangle$ by $\| \, |\phi\rangle - |\psi\rangle \, \| < \epsilon$.*

Proof. Let n be a positive integer. Let a, c be n-bit integers. The binary representations of a and c are

$$a = \sum_{i=0}^{n-1} a_i 2^i, \quad c = \sum_{i=0}^{n-1} c_i 2^i. \tag{89}$$

Let X, Y be arrays of size 2^n indexed by a or c. Let $\omega = \omega_{2^n} = \exp(2\pi i/2^n)$ be the standard 2^n root of unity.

The Fourier transform is defined as

$$Y_c = \frac{1}{\sqrt{2^n}} \sum_a X_a \omega^{ac} = \frac{1}{\sqrt{2^n}} \sum_a X_a \exp\left(\frac{2\pi}{2^n} ac\right), \tag{90}$$

or, in binary notation,

$$Y_c = \frac{1}{\sqrt{2^n}} \sum_a X_a \exp\left(\frac{2\pi}{2^n} \sum_{j,k=0}^{n-1} a_j c_k 2^{j+k}\right). \tag{91}$$

Whenever $j + k \geq n$, $\omega^{2^{j+k}} = 1$, so we drop those terms, giving the Fast Fourier Transform (FFT)

$$\text{(FFT)} \qquad Y_c = \frac{1}{\sqrt{2^n}} \sum_a X_a \exp\left(\frac{2\pi}{2^n} \sum_{\substack{0 \leq j,k \leq n-1 \\ j+k \leq n-1}} a_j c_k 2^{j+k}\right).$$

Now we approximate. Instead of the summation range having a $0 \leq j + k \leq n - 1$ bound, we parameterize on a positive integer $m < n$ and bound by $n - m \leq j + k \leq n - 1$, giving the Approximate Fast Fourier Transform (AFFT_m):

$$(\text{AFFT}_m) \qquad Y_c = \frac{1}{\sqrt{2^n}} \sum_a X_a \exp\left(\frac{2\pi}{2^n} \sum_{\substack{0 \leq j,k \leq n-1 \\ n-m \leq j+k \leq n-1}} a_j c_k 2^{j+k}\right).$$

The argument of "exp" in the AFFT differs from that in the FFT by

$$\frac{2\pi i}{2^n} \sum_{j+k<n-m} a_j c_k 2^{j+k}, \tag{92}$$

and is bounded in magnitude by

$$\begin{aligned}\left|\frac{2\pi i}{2^n}\sum_{\substack{0\le j,k\le n-1\\ j+k<n-m}} a_j c_k 2^{j+k}\right| &\le \frac{2\pi}{2^n}\sum_{0\le j<n-m} 2^j \sum_{0\le k<n-m-j} 2^k \\ &= \frac{2\pi}{2^n}\sum_{0\le j<n-m} 2^j\left(2^{n-m-j}-1\right) \\ &= \frac{2\pi}{2^n}\left((n-m)2^{n-m}-2^{n-m}+1\right) \\ &\le \frac{2\pi}{2^n} n 2^{n-m} \\ &= 2\pi n 2^{-m}.\end{aligned}$$

So the matrix entries of the AFFT differ from the FFT by a multiplicative factor of $\exp(i\delta)$, where $|\delta| \le 2\pi n 2^{-m}$. Let this error be $\exp(\delta_{j,k})$ in the (j,k) entry. From arc length on a circle, we have $|1-e^{i\delta}| \le |\delta|$.

To compute the error between the quantum states resulting from the FFT and AFFT, compute for any state $|\psi\rangle = \sum_j a_j|j\rangle$

$$\|(\mathrm{FFT}-\mathrm{AFFT}_m)|\psi\rangle\|^2 = \sum_{k=0}^{N-1}\left|\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}\omega_N^{jk}a_j(1-\exp(\delta_{j,k}))\right|^2 \tag{93}$$

$$\le (2\pi n 2^{-m})^2\sum_{k=0}^{N-1}\left|\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}\omega_N^{jk}a_j\right|^2 \tag{94}$$

$$= (2\pi n 2^{-m})^2\,\|\mathrm{FFT}|\psi\rangle\|^2 \tag{95}$$

$$= (2\pi n 2^{-m})^2\cdot 1 \tag{96}$$

Thus for any $\epsilon > 0$, taking $m \ge \log(2\pi) + \log n + \log(1/\epsilon)$ gives that

$$\|(\mathrm{FFT}-\mathrm{AFFT}_m)|\psi\rangle\| < \epsilon \tag{97}$$

Now we show how to compute the AFFT efficiently, similar to the method in section 3.4.2. Let $Q^{(J,K)}$ be the operation that multiplies the amplitude of those states with a 1 in positions J and K by a factor of $\omega^{2^{n-1-K-J}}$. This is similar to the $R_k^{(a,b)}$ defined for the QFT earlier. Let $H^{(J)}$ be the operation of applying the Hadamard matrix $\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$ to qubit J. Then check that the operation

$$H^{(0)}Q^{(0,1)}Q^{(0,2)}\dots Q^{(0,n-1)}H^{(1)}Q^{(1,2)}Q^{(1,3)}\dots Q^{n-2,n-1}H^{(n-1)} \tag{98}$$

performs the QFT as earlier. To perform the AFFT we drop those $Q^{(J,K)}$ with $K \ge J+m$, so it requires about nm 2-qubit operations. Taking $m = O(\log n + \log(1/\epsilon))$ to bound the error as required, we obtain the complexity bound. □

A.2. **The Quantum Fourier Transform over $\mathbb{Z}_N$, N Odd.** This section gives an algorithm to approximate the QFT over $\mathbb{Z}_N$ efficiently. The algorithm is from the Hales thesis [55] and the paper by Hallgren *et .al* [57] , but their proofs are incorrect. This section gives the proof from Lomont [88]. The end result is a proof of the correctness of their algorithm, with concrete bounds suitable for quantum

simulation instead of the asymptotic bounds listed in their papers. The final result is theorem A.17. The general idea of the algorithm is to make many copies of the initial state vector and perform a 2^n style QFT for a large value, and extract from this state period information for the original odd N. The proof requires a lot tedious work; it is more instructive to work through the algorithm until the general idea is clear.

A.2.1. *Notation and Basic Facts.* We fix three integers: an odd integer $N \geq 3$, $L \geq 2$ a power of 2, and $M \geq LN$ a power of 2. This gives $(M, N) = 1$, which we need later.

Some notation and facts to clarify the presentation:

- $\sqrt{-1}$ will be written explicitly, as i will always denote an index.
- For an integer $n > 1$, let $\omega_n = e^{2\pi\sqrt{-1}/n}$ denote a primitive n^{th} root of unity.
- Fact: $\left|1 - e^{\theta\sqrt{-1}}\right| \leq |\theta|$ as can be seen from arc length on the unit circle. If $-\pi \leq \theta \leq \pi$ we also[33] have $\left|\frac{\theta}{2}\right| \leq \left|1 - e^{\theta\sqrt{-1}}\right|$. Thus for real values α we have $|1 - \omega_M^\alpha| \leq \left|\frac{2\pi\alpha}{M}\right|$, etc.
- $\log n$ denotes log base 2, while $\ln n$ is the natural log. Since M and L are powers of two, $\lceil \log M \rceil = \lfloor \log M \rfloor = \lfloor \log M \rceil = \log M$, and similarly for L, but we often leave the symbols to emphasize expressions are integral.
- For a real number x, $\lceil x \rceil$ is the smallest integer greater than or equal to x, $\lfloor x \rfloor$ is the largest integer less than or equal to x, and $\lfloor x \rceil$ is the nearest integer, with ties rounding up[34]. We often use the three relations:

$$\begin{aligned} x - \frac{1}{2} \leq \lfloor x \rceil &\leq x + \frac{1}{2} \\ x - 1 < \lfloor x \rfloor &\leq x \\ x \leq \lceil x \rceil &< x + 1 \end{aligned}$$

- Indices: i and s will be indices from $0, 1, \ldots, N-1$. j will index from $0, 1, \ldots, L-1$. k will index from $0, 1, \ldots, M-1$. a and b will be arbitrary indices. t will index from a set C_s, defined in definition A.3 below.
- Given $i \in \{0, 1, \ldots, N-1\}$, let $i' = \left\lfloor \frac{M}{N} i \right\rceil$ denote the nearest integer to $\frac{M}{N} i$ with ties broken as above. Similarly for s and s'. Note $0 \leq i' \leq M-1$.
- For a real number x and positive real number n, let $x \bmod n$ denote the real number y such that $0 \leq y < n$ and $y = x + mn$ for an integer m. Note that we do not think of $x \bmod n$ as an equivalence class, but as a real number in $[0, n)$.
- $|u\rangle$ and $|v\rangle$ are vectors in spaces defined later, and given a vector $|u\rangle$ denote its coefficients relative to the standard (orthonormal) basis $\{|0\rangle, |1\rangle, \ldots, |n-1\rangle\}$ by $u_0, u_1, \ldots, u_{n-1}$, etc.
- For a real number x, let

$$|x|_M = \begin{cases} x \bmod M & \text{if} \quad 0 \leq (x \bmod M) \leq \frac{M}{2} \\ -x \bmod M & \text{otherwise} \end{cases}$$

[33] This range can be extended slightly.

[34] We could break ties arbitrarily with the same results.

Thus $0 \le |x|_M \le \frac{M}{2}$. Properties of this function are easiest to see by noting it is a sawtooth function, with period M, and height $M/2$.
- For an integer s set $\delta_s = \lfloor \frac{M}{N} s \rceil - \frac{M}{N} s$. Then $|\delta_s| \le \frac{1}{2}$.
- The (unitary) Fourier transform over a cyclic group of order N is denoted F_N. Thus if $|u\rangle = \sum_{i=0}^{N-1} u_i |i\rangle$, then $F_N|u\rangle = \frac{1}{\sqrt{N}} \sum_{i,s=0}^{N-1} u_i \omega_N^{is} |s\rangle$. We write $|\hat{u}\rangle = F_N|u\rangle$, with coefficients $\hat{u}_i$.
- $\sum_{i=0}^{N-1} |u_i|^2 = 1$ implies $\sum_i |u_i| \le \sqrt{N}$.

We define sets of integers which will play an important role:

Definition A.2. *For $i = 0, 1, \ldots, N-1$, let (i) denote the set of integers in the open interval $\left(i' - \frac{M}{2N} + \frac{1}{2}, i' + \frac{M}{2N} - \frac{1}{2}\right)$ taken* mod M. *Recall $i' = \lfloor \frac{M}{N} i \rceil$.*

The second definition we make precise is a division and remainder operation:

Definition A.3. *Given M, N as above. Set $\alpha = \lfloor \frac{M}{2N} + \frac{1}{2} \rfloor$, and $\beta = \lceil \frac{M}{2N} - \frac{3}{2} \rceil$. We define the map $\Delta : \{0, 1, \ldots, M-1\} \to \{0, 1, \ldots, N-1\} \times \{-\alpha, -\alpha+1, \ldots, \alpha\}$, as follows: for any $k \in \{0, 1, \ldots, M-1\}$, let $k \xrightarrow{\Delta} (s, t)$, via*

$$\begin{aligned} k' &= \left\lfloor k \frac{N}{M} \right\rceil \\ t &= k - \left\lfloor k' \frac{M}{N} \right\rceil \\ s &= k' \bmod N \end{aligned}$$

We extend this definition to a transform of basis elements $|k\rangle$ via

$$\Delta|k\rangle = |s\rangle|t + \alpha\rangle$$

and extend to all vectors by linearity.

Finally, from the image of Δ, define $C_s = \{\ t\ \mid\ (s,t) \in \text{Image } \Delta\}$ to be those values of t appearing for a fixed s. Thus $\sum_{k=0}^{M-1} |k\rangle \xrightarrow{\Delta} \sum_{s=0}^{N-1} \sum_{t \in C_s} |s\rangle|t + \alpha\rangle$.

We will show the integers $\{-\beta, \ldots, \beta\} \subseteq C_s \subseteq \{-\alpha, \ldots, \alpha\}$ for all s, which is why we defined β with the Δ definition. α and β remain fixed throughout the paper.

For the proofs to work, we need that the sets (i) are disjoint and have the same cardinality. Note also that the mod M condition gives $M-1, 0 \in (0)$ when $M > 3N$. We now show that the sets defined here have the required properties:

Lemma A.4. *For $i_1 \ne i_2 \in \{0, 1, \ldots, N-1\}$,*

$$|(i_1)| = |(i_2)| \tag{99}$$

$$(i_1) \bigcap (i_2) = \varnothing \tag{100}$$

Proof. Each set is defined using an interval of constant width, centered at an integer, so the sets will have the same cardinality. To show disjointness, for any integer a, take the rightmost bound $R_a = \lfloor \frac{M}{N} a \rceil + \frac{M}{2N} - \frac{1}{2}$ of an interval and compare it to

the leftmost bound $L_{a+1} = \left\lfloor \frac{M}{N}(a+1) \right\rceil - \frac{M}{2N} + \frac{1}{2}$ of the next interval:

$$L_{a+1} - R_a = \left\lfloor \frac{M}{N}(a+1) \right\rceil - \left\lfloor \frac{M}{N} a \right\rceil - \frac{M}{N} + 1 \tag{101}$$

$$\geq \left(\frac{M}{N}(a+1) - \frac{1}{2} \right) - \left(\frac{M}{N} a + \frac{1}{2} \right) - \frac{M}{N} + 1 \tag{102}$$

$$= 0 \tag{103}$$

giving that the open intervals are disjoint. Thus taking the integers in the intervals mod M remains disjoint (which requires $i_1, i_2 \leq N-1$). □

Note the image of Δ is not a cartesian product; the values t assumes depend on s, otherwise we would have that M is a multiple of N. In other words, the cardinality of C_s depends on s, with bounds given in the following lemma, where we show that our definition works and list some properties:

Lemma A.5. *Using the notation from definition A.3,*

1) the map Δ is well defined, and a bijection with its image,

2) $\alpha = \beta + 1$,

3) the sets of integers satisfy $\{-\beta, \ldots, \beta\} \subseteq C_s \subseteq \{-\alpha, \ldots, \alpha\}$ for all $s \in \{0, 1, \ldots, N-1\}$.

Proof. Given a k in $\{0, 1, \ldots, M-1\}$, let $\Delta(k) = (s, t)$. Clearly $0 \leq s \leq N-1$. Set $\alpha = \left\lfloor \frac{M}{2N} + \frac{1}{2} \right\rfloor$. To check that $-\alpha \leq t \leq \alpha$, note

$$\frac{N}{M} k - \frac{1}{2} \leq k' \leq \frac{N}{M} k + \frac{1}{2} \tag{104}$$

giving

$$\frac{M}{2N} + \frac{1}{2} \geq t = k - \left\lfloor \frac{M}{N} k' \right\rceil \geq -\left(\frac{M}{2N} + \frac{1}{2} \right) \tag{105}$$

and t integral allows the rounding operation. Thus the definition makes sense.

Next we check that both forms of Δ in the definition are bijections. Suppose $k_1 \neq k_2$ are both in $\{0, 1, \ldots, M-1\}$, with images $\Delta(k_r) = (s_r, t_r), r = 1, 2$. Let $k'_r = \left\lfloor \frac{N}{M} k_r \right\rceil, r = 1, 2$. Note $0 \leq k'_r \leq N$.

Assume $(s_1, t_1) = (s_2, t_2)$. If $k'_1 = k'_2$, then

$$t_1 = k_1 - \left\lfloor \frac{M}{N} k'_1 \right\rceil = k_1 - \left\lfloor \frac{M}{N} k'_2 \right\rceil \tag{106}$$

$$\neq k_2 - \left\lfloor \frac{M}{N} k'_2 \right\rceil = t_2 \tag{107}$$

a contradiction. So we are left with the case $k'_1 \neq k'_2$. In order for $s_1 = s_2$ we have (without loss of generality) $k'_1 = 0, k'_2 = N$. But then $t_1 = k_1 \geq 0$ and $t_2 = k_2 - M \leq M - 1 - M = -1$, a contradiction. Thus Δ in the first sense is a bijection.

The second interpretation follows easily, since $-\alpha \leq t \leq \alpha$ gives $0 \leq t + \alpha \leq 2\alpha$. So the second register needs to have a basis with at least $2\alpha + 1$ elements, which causes the number of qubits needed[35] to implement the algorithm to be $\lceil \log M \rceil + 2$ instead of $\lceil \log M \rceil$.

[35] This is proven in theorem A.17.

To see $\alpha = \beta + 1$, bound $\alpha - \beta$ using the methods above, and[36] one obtains $2 > \alpha - \beta > 0$.

All integers between $\left\lfloor \frac{M}{N}(s+1) \right\rceil$ and $\left\lfloor \frac{M}{N} s \right\rceil$ inclusive must be of the form $t_1 + \left\lfloor \frac{M}{N} s \right\rceil$ for $t_1 \in C_s$ or of the form $t_2 + \left\lfloor \frac{M}{N}(s+1) \right\rceil$ for $t_2 \in C_{s+1}$. This range contains $\left\lfloor \frac{M}{N}(s+1) \right\rceil - \left\lfloor \frac{M}{N} s \right\rceil + 1 \geq \frac{M}{N}$ integers, and at most $\alpha + 1$ of these are of the form $t_2 + \left\lfloor \frac{M}{N}(s+1) \right\rceil$ with $t_2 \in C_{s+1}$. This leaves at least $\left\lceil \frac{M}{N} \right\rceil - \alpha \geq \frac{M}{2N} - \frac{3}{2}$ that have to be of the form $t_1 + \left\lfloor \frac{M}{N} s \right\rceil$ with $t_1 \in C_s$, implying $\beta \in C_s$. Similar arguments give $\pm\beta \in C_s$, thus $\{-\beta, \ldots, \beta\} \subseteq C_s \subseteq \{-\alpha, \ldots, \alpha\}$ for all s. □

Δ is efficient to implement as a quantum operation, since it is efficient classically [29, Chapter 4]. Finally we note that Δ, being a bijection, can be extended to a permutation of basis vectors $|k\rangle$, thus can be considered an efficiently implementable unitary operation.

We define some vectors we will need. For $i \in \{0, 1, \ldots, N-1\}$ define

$$\begin{aligned}
|A^i\rangle &= F_M F_{LN}^{-1} |Li\rangle \\
&= \frac{1}{\sqrt{LMN}} \sum_{k=0}^{M-1} \sum_{a=0}^{LN-1} \omega_N^{-ai} \omega_M^{ak} |k\rangle \\
|B^i\rangle &= |A^i\rangle \text{ restricted to integers in the set } (i) \\
&= \sum_{b \in (i)} A_b^i |b\rangle \\
&= \frac{1}{\sqrt{LMN}} \sum_{b \in (i)} \sum_{a=0}^{LN-1} \omega_N^{-ai} \omega_M^{ab} |b\rangle \\
|T^i\rangle &= |A^i\rangle \text{ restricted to integers outside the set } (i) \\
&= \sum_{b \notin (i)} A_b^i |b\rangle \\
&= |A^i\rangle - |B^i\rangle \\
&= \frac{1}{\sqrt{LMN}} \sum_{b \notin (i)} \sum_{a=0}^{LN-1} \omega_N^{-ai} \omega_M^{ab} |b\rangle
\end{aligned}$$

Think A^i for actual values, B^i for bump functions, and T^i for tail functions. Note that the coefficients B_b^i and T_b^i are just A_b^i for b in the proper ranges.

We also define three equivalent shifted versions of $|B^0\rangle$. Note that to make these definitions equivalent we require the sets (i) to have the same cardinality. Let $|S^i\rangle = \sum_{b \in (0)} B_b^0 |b + i'\rangle = \sum_{b \in (0)} A_b^0 |b + i'\rangle = \sum_{b \in (i)} A_{b-i'}^0 |b\rangle$, where each $b \pm i'$ expression is taken mod M. The $|S^i\rangle$ have *disjoint support*, which follows from lemma A.4, and will be important for proving theorem A.14.

A.2.2. *The Algorithm.* The algorithm takes a unit vector (quantum state) $|u\rangle$ on $\lceil \log N \rceil$ qubits[37], does a Fourier transform F_L, L a power of two, on another register containing $|0\rangle$ with $\lceil \log M \rceil - \lceil \log N \rceil + 2$ qubits, to create[38] a superposition, and

[36] $(M, N) = 1$ is used to get the strict inequalities.

[37] Recall logs are base 2.

[38] Note it may be more efficient to apply the Hadamard operator H to each qubit in $|0\rangle$.

then reindexes the basis to create L (normalized) copies of the coefficients of $|u\rangle$, resulting in $|u_L\rangle$. Then another power of two Fourier transform F_M is applied. The division Δ results in a vector very close to the desired output $F_N|u\rangle$ in the first register, with garbage in the second register (with some slight entanglement). The point of this paper is to show how close the output is to this tensor product. We use $\lceil \log M \rceil + 2$ qubits, viewed in two ways: as a single register $|k\rangle$, or as a $\lceil \log N \rceil$-qubit first register, with the remaining qubits in the second register, written $|s\rangle|t\rangle$. We note that merely $\lceil \log M \rceil$ qubits may not be enough qubits to hold some of the intermediate results. The algorithm is:

A.2.3. *The Odd Cyclic QFT Algorithm.*

$$|u\rangle|0\rangle \xrightarrow{F_L} \frac{1}{\sqrt{L}} \sum_{i=0}^{N-1} \sum_{j=0}^{L-1} u_i |i\rangle|j\rangle \tag{108}$$

$$\xrightarrow{\text{multiply}} \frac{1}{\sqrt{L}} \sum_{i,j} u_i |i + jN\rangle \tag{109}$$

$$= |u_L\rangle \tag{110}$$

$$\xrightarrow{F_M} \frac{1}{\sqrt{LM}} \sum_{i,j} \sum_{k=0}^{M-1} u_i \omega_M^{(i+jN)k} |k\rangle \tag{111}$$

$$\xrightarrow{\Delta} \frac{1}{\sqrt{LM}} \sum_{i,j} u_i \sum_{s=0}^{N-1} \sum_{t \in C_s} \omega_M^{(i+jN)\left(t + \left\lfloor \frac{M}{N} s \right\rceil\right)} |s\rangle|t + \alpha\rangle \tag{112}$$

$$= \frac{1}{\sqrt{N}} \sum_{i,s=0}^{N-1} u_i \omega_N^{is} |s\rangle \sqrt{\frac{N}{LM}} \sum_{t \in C_s} \sum_{j=0}^{L-1} \omega_M^{(i+jN)(t+\delta_s)} |t + \alpha\rangle \tag{113}$$

$$= |v\rangle \tag{114}$$

$|u_L\rangle$ is the vector that is L copies of the coefficients from $|u\rangle$, normalized. $|v\rangle$ is the algorithm output.

Notice that $F_N|u\rangle$ appears in the output in line 113, but the rest is unfortunately dependent on s and i. However the dependence is small: if C_s were the same for all s, if the δ_s, which are bounded in magnitude by $\frac{1}{2}$, were actually zero, and if the i dependence were dropped, then the output would leave $F_N|u\rangle$ in the first register. The paper shows this is approximately true, and quantifies the error.

A.2.4. *Initial Bounds.* We need many bounds to reach the final theorem, which we now begin proving.

Lemma A.6. *For integers $N > 2$, $M \geq 2N$, and any $i \in \{0, 1, \ldots, N-1\}$, $k \in \{0, 1, \ldots, M-1\}$, with $k \notin (i)$, we have*

$$\left| k - \frac{M}{N} i \right|_M \geq \frac{M}{2N} - 1 \tag{115}$$

Proof. The sets (i) are disjoint, so we do two cases. If $i = 0$, then $k \notin (0)$ implies

$$\frac{M}{2N} - \frac{1}{2} \leq k \leq M - \frac{M}{2N} + \frac{1}{2} \tag{116}$$

from which it follows that

$$\left|k - \frac{M}{N}0\right|_M \geq \frac{M}{2N} - \frac{1}{2} > \frac{M}{2N} - 1 \tag{117}$$

If $i \neq 0$, then either k is less than the integers in (i) or greater than the integers in (i), giving two subcases. Subcase 1:

$$0 \leq k \leq \left\lfloor \frac{M}{N}i \right\rceil - \frac{M}{2N} + \frac{1}{2} \leq \frac{M}{N}i - \frac{M}{2N} + 1 \tag{118}$$

implying

$$\frac{M}{2N} - 1 \leq \frac{M}{N}i - k \leq \frac{M}{N}i \leq M - \frac{M}{N} \tag{119}$$

which gives the bound. Subcase 2 is then

$$\frac{M}{N}i + \frac{M}{2N} - 1 \leq \left\lfloor \frac{M}{N}i \right\rceil + \frac{M}{2N} - \frac{1}{2} \leq k \leq M - 1 \tag{120}$$

which implies

$$\frac{M}{2N} - 1 \leq k - \frac{M}{N}i \leq M - 1 - \frac{M}{N}i \tag{121}$$

giving the bound and the proof. □

We now bound many of the $|A^i\rangle$ coefficients.

Lemma A.7. *For $k \in \{0, 1, \ldots, M-1\}$ and $i \in \{0, 1, \ldots, N-1\}$, with $\frac{k}{M} - \frac{i}{N}$ not an integer, then*

$$\left|A^i_k\right| \leq \sqrt{\frac{M}{LN}} \frac{2}{\pi \left|k - \frac{M}{N}i\right|_M} \tag{122}$$

Proof. We rewrite from the definition

$$A^i_k = \frac{1}{\sqrt{LMN}} \sum_{a=0}^{LN-1} \omega_M^{a\left(k - \frac{M}{N}i\right)} \tag{123}$$

(124)

which is a geometric series. By hypothesis, $\omega_M^{\left(k - \frac{M}{N}i\right)} \neq 1$, so we can sum as[39]

$$\left|A^i_k\right| = \frac{1}{\sqrt{LMN}} \left| \frac{1 - \omega_M^{LN\left(k - \frac{M}{N}i\right)}}{1 - \omega_M^{\left(k - \frac{M}{N}i\right)}} \right| \tag{125}$$

The numerator is bounded above by 2, and the denominator satisfies

$$\left|1 - \omega_M^{\left(k - \frac{M}{N}i\right)}\right| = \left|1 - \omega_M^{\left|k - \frac{M}{N}i\right|_M}\right| \tag{126}$$

$$\geq \frac{\pi \left|k - \frac{M}{N}i\right|_M}{M} \tag{127}$$

[39] Without this requirement, the sum would be LN, much different than the claimed sum. The hypotheses avoid the resulting divide by zero.

These together give

$$\left|A_k^i\right| \leq \sqrt{\frac{M}{LN}} \frac{2}{\pi \left|k - \frac{M}{N} i\right|_M} \tag{128}$$

□

Note our initial requirement that $(M, N) = 1$ is strong enough to satisfy the non-integral hypothesis in lemma A.7, except for the case $i = k = 0$, which we will avoid.

Next we bound a sum of these terms. We fix $\gamma = \frac{1}{2} - \frac{N}{M}$ for the rest of this paper.

Lemma A.8. *Given integers $N > 2$ and $M > 2N$, with N odd. Let $\gamma = \frac{1}{2} - \frac{N}{M}$. For a fixed integer $k \in \{0, 1, \ldots, M-1\}$,*

$$\sum_{\substack{i=0 \\ k \notin (i)}}^{N-1} \frac{1}{\left|k - \frac{M}{N} i\right|_M} \leq \frac{2N}{M} \left(\frac{1}{\gamma} + \ln \left| \frac{N-1}{2\gamma} + 1 \right| \right) \tag{129}$$

Proof. The minimum value of the denominator is at least $\frac{M}{2N} - 1$ by lemma A.6, and the rest are spaced out by $\frac{M}{N}$, but can occur twice[40] since the denominator is a sawtooth function going over one period, giving that

$$\sum_{\substack{i=0 \\ k \notin (i)}}^{N-1} \frac{1}{\left|k - \frac{M}{N} i\right|_M} \leq 2 \sum_{a=0}^{\frac{N-1}{2}} \frac{1}{\frac{M}{2N} - 1 + \frac{M}{N} a} \tag{130}$$

$$= \frac{2N}{M} \left(\frac{1}{\gamma} + \sum_{a=1}^{\frac{N-1}{2}} \frac{1}{\gamma + a} \right) \tag{131}$$

$$\leq \frac{2N}{M} \left(\frac{1}{\gamma} + \int_0^{(N-1)/2} \frac{1}{x + \gamma} dx \right) \tag{132}$$

$$= \frac{2N}{M} \left(\frac{1}{\gamma} + \ln \left| \frac{N-1}{2\gamma} + 1 \right| \right) \tag{133}$$

□

The generality of the above lemma would be useful where physically adding more qubits than necessary would be costly, since the lemma lets the bound tighten as $\frac{N}{M}$ decreases. However the following corollary is what we will use in the final theorem.

Corollary A.9. *Given integers $N \geq 13$ and $M \geq 16N$, with N odd. For a fixed value $k \in \{0, 1, \ldots, M-1\}$,*

$$\sum_{\substack{i=0 \\ k \notin (i)}}^{N-1} \frac{1}{\left|k - \frac{M}{N} i\right|_M} \leq \frac{4N \ln N}{M} \tag{134}$$

[40] Both [55] and [57] appear to overlook this fact.

Proof. Using lemma A.8, $M \geq 16N$ gives $\frac{1}{\gamma} \leq \frac{16}{7}$ and

$$\frac{1}{\gamma} + \ln\left|\frac{N-1}{2\gamma}+1\right| \leq \frac{16}{7} + \ln\left|\frac{8(N-1)}{7}+1\right| \tag{135}$$

$$= \ln\left(e^{\frac{16}{7}}\left(\frac{8(N-1)}{7}+1\right)\right) \tag{136}$$

$$\leq \ln\left(\frac{8}{7}\,e^{\frac{16}{7}}N\right) \tag{137}$$

$$\leq 2\ln N \tag{138}$$

where the last step required $N \geq \left(\frac{8}{7}\,e^{\frac{16}{7}}\right) > 11.2$. The corollary follows. □

Next we prove a bound on a sum of the above terms, weighted with a real unit vector. This will lead to a bound on the tails $\left\|\sum_i \hat{u}_i|T^i\rangle\right\|$.

Lemma A.10. *Given integers $N \geq 13$ and $M \geq 16N$, with N odd. For any unit vector $x \in \mathbb{R}^N$*

$$\sum_{k=0}^{M-1}\left|\sum_{\substack{i=0\\k\notin(i)}}^{N-1}\frac{x_i}{\left|k-\frac{M}{N}i\right|_M}\right|^2 \leq \frac{22N\ln^2 N}{M} + \frac{32N^3}{M^2} \tag{139}$$

Proof. We split the expression into three parts, the first of which we can bound using methods from [55] and [57], and the other two terms we bound separately.

Using the Δ operator from definition A.3, along with the values α and β defined there, and using lemma A.5, we can rewrite each k with $k = t + \left\lfloor \frac{M}{N}k' \right\rceil = t + \frac{M}{N}k' + \delta_s$. Since s differs from k' by a multiple of N, and the $|x|_M$ function has period M, in $\left|\frac{M}{N}(k'-i)+t+\delta_s\right|_M$ we can replace k' with s. Rewrite the left hand side of inequality 139 as

$$\sum_{k=0}^{M-1}\left|\sum_{\substack{i=0\\k\notin(i)}}^{N-1}\frac{x_i}{\left|k-\frac{M}{N}i\right|_M}\right|^2 = \sum_{s=0}^{N-1}\sum_{t\in C_s}\left|\sum_{\substack{i=0\\s\neq i}}^{N-1}\frac{x_i}{\left|\frac{M}{N}(s-i)+t+\delta_s\right|_M}\right|^2 \tag{140}$$

Letting $\Delta k = (s,t)$, note that $k \notin (i)$ if and only if $s \neq i$, which can be shown from the definitions and the rounding rules used earlier. To simplify notation, write $q^t_{i,s} = \frac{M}{N}(s-i)+t+\delta_s$. We have not changed the values of the denominators, so $|q^t_{i,s}|_M \geq \frac{M}{2N} - 1$ by lemma A.6 for all $i, (s,t)$ in this proof.

We want to swap the s and t sums, but we need to remove the t dependence on s. Again using lemma A.5, we can split the expression into the three terms:

$$\sum_{t=-\beta}^{\beta}\sum_{s=0}^{N-1}\left|\sum_{\substack{i=0\\ s\neq i}}^{N-1}\frac{x_i}{\left|q_{i,s}^t\right|_M}\right|^2 \tag{141}$$

$$+\sum_{s \text{ with } \alpha\in C_s}\left|\sum_{\substack{i=0\\ s\neq i}}^{N-1}\frac{x_i}{\left|q_{i,s}^\alpha\right|_M}\right|^2 \tag{142}$$

$$+\sum_{s \text{ with } -\alpha\in C_s}\left|\sum_{\substack{i=0\\ s\neq i}}^{N-1}\frac{x_i}{\left|q_{i,s}^{-\alpha}\right|_M}\right|^2 \tag{143}$$

Next we bound the first term 141. For a unit vector x and fixed t we rewrite the s,i sum as the norm of a square matrix P_t acting on x, so that the sum over s and i becomes

$$\|P_t x\|^2 = \sum_{s=0}^{N-1}\left|\sum_{\substack{i=0\\ s\neq i}}^{N-1}\frac{x_i}{\left|q_{i,s}^t\right|_M}\right|^2 \tag{144}$$

We also define similarly to each P_t a matrix Q_t which is the same except for minor modifications to the denominator:

$$\|Q_t x\|^2 = \sum_{s=0}^{N-1}\left|\sum_{\substack{i=0\\ s\neq i}}^{N-1}\frac{x_i}{\left|q_{i,s}^t-\delta_s\right|_M}\right|^2 \tag{145}$$

Note this matrix is circulant[41], since each entry in the matrix only depends on $s-i$. Also each entry is nonnegative[42]. Thus the expression is maximized by the vector $y=\frac{1}{\sqrt{N}}(1,1,\ldots,1)$ as shown in each of [55], [57], and [65]. Now we relate these matrix expressions. Recall $|q_{i,s}^t|_M \geq \frac{M}{2N}-1$ and $|\delta_s|\leq\frac{1}{2}$. Set $\lambda=\frac{N}{M-2N}$. Then we find lower and upper bounds

$$1-\lambda = 1-\frac{1}{2(\frac{M}{2N}-1)} \leq \frac{\left|q_{i,s}^t\right|_M-\frac{1}{2}}{\left|q_{i,s}^t\right|_M} \leq \frac{\left|q_{i,s}^t-\delta_s\right|_M}{\left|q_{i,s}^t\right|_M} \tag{146}$$

and

$$\frac{\left|q_{i,s}^t-\delta_s\right|_M}{\left|q_{i,s}^t\right|_M} \leq \frac{\left|q_{i,s}^t\right|_M+\frac{1}{2}}{\left|q_{i,s}^t\right|_M} \leq 1+\frac{1}{2(\frac{M}{2N}+1)} = 1+\lambda \tag{147}$$

Rewriting

$$\|P_t x\|^2 = \sum_{s=0}^{N-1}\left|\sum_{\substack{i=0\\ s\neq i}}^{N-1}\frac{x_i}{\left|q_{i,s}^t-\delta_s\right|_M}\frac{\left|q_{i,s}^t-\delta_s\right|_M}{\left|q_{i,s}^t\right|_M}\right|^2 \tag{148}$$

[41] That is, each row after the first is the cyclic shift by one from the previous row.

[42] $|q_{i,s}^t-\delta_s|_M \geq |q_{i,s}^t|_M-\frac{1}{2}\geq\frac{M}{2N}-\frac{3}{2}>0$ since $M>3N$

and using the bounds gives

$$(1-\lambda)^2 \|Q_t x\|^2 \le \|P_t x\|^2 \le (1+\lambda)^2 \|Q_t x\|^2 \tag{149}$$

Then since y maximizes $\|Q_t x\|^2$,

$$\|P_t x\|^2 \le (1+\lambda)^2 \|Q_t x\|^2 \le (1+\lambda)^2 \|Q_t y\|^2 \le \left(\frac{1+\lambda}{1-\lambda}\right)^2 \|P_t y\|^2 \tag{150}$$

giving that we can bound the leftmost term by $\left(\frac{1+\lambda}{1-\lambda}\right)^2$ times the norm at y. $\left(\frac{1+\lambda}{1-\lambda}\right)^2$ takes on values between 1 and $\frac{225}{169} \approx 1.33$ for $M \ge 16N$, better than the constant 4 in [55] and [57].

Combined with corollary A.9 this allows us to bound term 141:

$$\sum_{t=-\beta}^{\beta} \sum_{s=0}^{N-1} \left| \sum_{\substack{i=0 \\ s\neq i}}^{N-1} \frac{x_i}{|q_{i,s}^t|_M} \right|^2 \le \sum_t \frac{225}{169} \sum_{s=0}^{N-1} \left| \sum_{\substack{i=0 \\ s\neq i}}^{N-1} \frac{\frac{1}{\sqrt{N}}}{|q_{i,s}^t|_M} \right|^2 \tag{151}$$

$$\le (2\beta+1) \frac{225}{169} \frac{N}{N} \left(\frac{4N \ln N}{M}\right)^2 \tag{152}$$

$$\le \frac{M}{N} \frac{225}{169} \left(\frac{4N \ln N}{M}\right)^2 \tag{153}$$

$$\le \frac{22N \ln^2 N}{M} \tag{154}$$

Now we bound the other two terms, 142 and 143. We need the following fact, which can be shown with calculus: the expression $\left|\sum_{i=0}^{N-1} a_i x_i\right|$ subject to the condition $\sum_{i=0}^{N-1} x_i^2 = 1$, has maximum value $\sqrt{\sum_{i=0}^{N-1} a_i^2}$. Then term 142 can be bounded using a similar technique as in the proof of lemma A.9. Again we take $\gamma = \frac{1}{2} - \frac{N}{M}$.

$$\sum_{s \text{ with } \alpha \in C_s} \left| \sum_{\substack{i=0 \\ s\neq i}}^{N-1} \frac{x_i}{|q_{i,s}^\alpha|_M} \right|^2 \le \sum_s \left| \sqrt{\sum_{\substack{i=0 \\ s\neq i}}^{N-1} \frac{1}{|q_{i,s}^\alpha|_M^2}} \right|^2 \tag{155}$$

$$\le N \frac{2N^2}{M^2} \left(\frac{1}{\gamma^2} + \sum_{a=1}^{\frac{N-1}{2}} \frac{1}{\left(\frac{1}{2} - \frac{N}{M} + a\right)^2} \right) \tag{156}$$

$$\le \frac{2N^3}{M^2} \left(\frac{1}{\gamma^2} + \frac{1}{\gamma} - \frac{1}{\frac{N-1}{2} + \gamma} \right) \tag{157}$$

$$\le \frac{16N^3}{M^2} \tag{158}$$

Term 143 is bound with the same method and result, and adding these three bounds gives the desired inequality 139. □

We now use these lemmata to bound the tails $\left\|\sum_i \hat{u}_i |T^i\rangle\right\|$.

Lemma A.11. *Given three integers: an odd integer $N \geq 13$, $L \geq 2$ a power of two, and $M \geq 16N$ a power of two, then*

$$\left\| \sum_{i=0}^{N-1} \hat{u}_i |T^i\rangle \right\| \leq \frac{2}{\pi} \sqrt{\frac{22 \ln^2 N}{L} + \frac{32N^2}{LM}} \tag{159}$$

Proof.

$$\left\| \sum_{i=0}^{N-1} \hat{u}_i |T^i\rangle \right\|^2 = \sum_{k=0}^{M-1} \left| \sum_{\substack{i=0 \\ k \notin (i)}}^{N-1} \hat{u}_i T_k^i \right|^2 \tag{160}$$

$$\leq \sum_k \frac{4M}{\pi^2 LN} \left(\sum_{\substack{i=0 \\ k \notin (i)}}^{N-1} \frac{|\hat{u}_i|}{\left|k - \frac{M}{N} i\right|_M} \right)^2 \tag{161}$$

$$\leq \frac{4M}{\pi^2 LN} \left(\frac{22N \ln^2 N}{M} + \frac{32N^3}{M^2} \right) \tag{162}$$

Taking square roots gives the result. Note that the requirements of lemma A.7 are satisfied when obtaining line 161, since we avoid the $k = i = 0$ case, and $(M, N) = 1$. □

Next we show that the shifted $|S^i\rangle$ are close to the $|B^i\rangle$, which will allow us to show the algorithm output is close to a tensor product.

Lemma A.12.

$$\left\| |S^i\rangle - |B^i\rangle \right\| \leq \frac{\pi LN}{M\sqrt{3}} \tag{163}$$

Proof. Recall $|S^i\rangle = \sum_{b \in (i)} A^0_{b - i' \bmod M} |b\rangle$ and $|B^i\rangle = \sum_{b \in (i)} A^i_b |b\rangle$. It is important these are supported on the same indices! Also recall that $|A^i\rangle = F_M F_{LN}^{-1} |Li\rangle$ and that F_M is unitary. Then (dropping mod M throughout for brevity)

$$\left\| |S^i\rangle - |B^i\rangle \right\|^2 = \left\| \sum_{b \in (i)} A^0_{b-i'} |b\rangle - \sum_{b \in (i)} A^i_b |b\rangle \right\|^2 \tag{164}$$

$$\leq \left\| \sum_{k=0}^{M-1} A^0_{k-i'} |k\rangle - \sum_{k=0}^{M-1} A^i_k |k\rangle \right\|^2 \tag{165}$$

$$= \left\| F_M^{-1} \left(\sum_{k=0}^{M-1} A^0_k |k + i'\rangle - |A^i\rangle \right) \right\|^2 \tag{166}$$

$$= \sum_{a=0}^{LN-1} \left| \frac{1}{\sqrt{LN}} \omega_M^{-ai'} - \frac{1}{\sqrt{LN}} \omega_N^{-ai} \right|^2 \tag{167}$$

$$= \frac{1}{LN} \sum_{a=0}^{LN-1} \left| \omega_M^{-ai'} \left(1 - \omega_M^{a\delta_i} \right) \right|^2 \tag{168}$$

and this can be bounded by

$$\frac{1}{LN}\sum_{a=0}^{LN-1}\left|\frac{2\pi a\delta_i}{M}\right|^2 \le \frac{\pi^2}{LNM^2}\sum_{a=0}^{LN-1}a^2 \le \frac{\pi^2}{LNM^2}\frac{(LN)^3}{3} \tag{169}$$

Taking square roots gives the bound. □

In the above proof, to obtain line 165 we needed that $|S^i\rangle$ and $|B^i\rangle$ have the same support, but $|S^i\rangle$ is a shifted version of $|B^0\rangle$, so we implicitly needed all the sets (i) to have the same cardinality. This is not satisfied in [57] (although it is needed) but is met in [55].

For the rest of the section we need a set which is (0) without mod M applied: let Λ be those integers in the open interval $(-\lfloor\frac{M}{2N}-\frac{1}{2}\rfloor, \lfloor\frac{M}{2N}-\frac{1}{2}\rfloor)$. Then

Lemma A.13.

$$\Delta|S^i\rangle \quad = \quad |i\rangle\sum_{t\in\Lambda}A_t^0|t+\alpha\rangle \tag{170}$$

Proof. By definition, $|S^i\rangle = \sum_{b\in(0)}A_b^0|b+\lfloor\frac{M}{N}i\rceil \bmod M\rangle$. $\Delta\left(b+\lfloor\frac{M}{N}i\rceil\right) = (i,b)$ (the proof uses $(M,N)=1$), and Δ a bijection implies $\Delta|b+\lfloor\frac{M}{N}i\rceil \bmod M\rangle = |i\rangle|b+\alpha\rangle$. The rest follows[43]. □

Main results. Now we are ready to use the above lemmata to prove the main theorem.

Theorem A.14. *Given three integers: an odd integer $N\ge 13$, $L\ge 16$ a power of two, and $M\ge LN$ a power of two. Then the output $|v\rangle$ of the algorithm in section A.2.3 satisfies*

$$\left\||v\rangle - F_N|u\rangle\otimes\sum_{t\in\Lambda}A_t^0|t+\alpha\rangle\right\| \le \frac{2}{\pi}\sqrt{\frac{22\ln^2 N}{L}+\frac{32N^2}{LM}}+\frac{\pi LN}{M\sqrt{3}} \tag{171}$$

Proof. Note

$$|\hat{u}\rangle := F_N|u\rangle = \sum_{i=0}^{N-1}\hat{u}_i|i\rangle \qquad\qquad F_M|u_L\rangle = \sum_{i=0}^{N-1}\hat{u}_i|A^i\rangle \tag{172}$$

Using lemma A.13 and that Δ is unitary allows us to rewrite the left hand side as

$$\left\||v\rangle - \sum_{\substack{s=0\\ t\in C_s}}^{N-1}\hat{u}_s A_t^0|s\rangle|t+\alpha\rangle\right\| \quad = \quad \left\|\Delta F_M|u_L\rangle - \sum_{s=0}^{N-1}\hat{u_s}\Delta|S^s\rangle\right\| \tag{173}$$

$$= \quad \left\|\sum_{s=0}^{N-1}\hat{u_s}|A^s\rangle - \sum_{s=0}^{N-1}\hat{u_s}|S^s\rangle\right\| \tag{174}$$

$$= \quad \left\|\sum_{s=0}^{N-1}\hat{u_s}(|B^s\rangle+|T^s\rangle) - \sum_{s=0}^{N-1}\hat{u_s}|S^s\rangle\right\| \tag{175}$$

[43] It is tempting to use C_0 instead of Λ, but this is not correct in all cases.

By the triangle inequality this is bounded by

$$\left\| \sum_{s=0}^{N-1} \hat{u_s} |T^s\rangle) \right\| + \left\| \sum_{s=0}^{N-1} \hat{u_s} |B^s\rangle) - \sum_{s=0}^{N-1} \hat{u_s} |S^s\rangle \right\| \tag{176}$$

which in turn by lemmata A.11 and A.12 is bounded by

$$\frac{2}{\pi} \sqrt{\frac{22 \ln^2 N}{L} + \frac{32N^2}{LM}} + \frac{\pi LN}{M\sqrt{3}} \sqrt{\sum_s |\hat{u_s}|^2} \tag{177}$$

The last expression has $\| |\hat{u}\rangle \| = 1$, which gives the result. Note that to obtain line 177 we needed the supports of the $|B^s\rangle$ disjoint, and that the $|S^i\rangle$ and $|B^i\rangle$ have the same support[44]. □

This shows that the output of the algorithm in section A.2.3 is close to a tensor product of the desired output $F_N|u\rangle$ and another vector (which is not in general a unit vector). Since a quantum state is a unit vector, we compare the output to a unit vector in the direction of our approximation via:

Lemma A.15. *Let $\vec{a}$ be a unit vector in a finite dimensional vector space, and $\vec{b}$ any vector in that space. For any $0 \leq \epsilon \leq 1$, if $\left\| \vec{a} - \vec{b} \right\| \leq \epsilon$ then the unit vector $\vec{b'}$ in the direction of $\vec{b}$ satisfies $\left\| \vec{a} - \vec{b'} \right\| \leq \epsilon\sqrt{2}$.*

Proof. Simple geometry shows the distance is bounded by $\sqrt{2(1 - \sqrt{1 - \epsilon^2})}$, and this expression divided by ϵ has maximum value $\sqrt{2}$ on $(0, 1]$. The $\epsilon = 0$ case is direct. □

So we only need a $\sqrt{2}$ factor to compare the algorithm output with a unit vector which is $F_N|u\rangle$ tensor another unit vector. We let $|\psi\rangle$ denote the unit length vector in the direction of $\sum_{t \in \Lambda} A_t^0 |t + \alpha\rangle$ for the rest of this paper.

For completeness, we repeat arguments from [57, 65] to obtain the operation complexity and probability distribution, and we show concrete choices for M and L achieving a desired error bound.

To show that measuring the first register gives measurement statistics which are very close to the desired distribution, we need some notation. Given two probability distributions $\mathcal{D}$ and $\mathcal{D}'$ over $\{0, 1, \ldots, M-1\}$, let $|\mathcal{D} - \mathcal{D}'| = \sum_{k=0}^{M-1} |\mathcal{D}(k) - \mathcal{D}'(k)|$ denote the total variation distance. Then a result[45] of Bernstein and Vazirani [18] states that if the distance between any two states is small, then so are the induced[46] probability distributions:

Lemma A.16 ([18], Lemma 3.6)**.** *Let $|\alpha\rangle$ and $|\beta\rangle$ be two normalized states, inducing probability distributions $\mathcal{D}_\alpha$ and $\mathcal{D}_\beta$. Then for any $\epsilon > 0$*

$$\| |\alpha\rangle - |\beta\rangle \| \leq \epsilon \Rightarrow |\mathcal{D}_\alpha - \mathcal{D}_\beta| \leq 2\epsilon + \epsilon^2 \tag{178}$$

independent of what basis is used for measurement.

[44] This is not satisfied in [55], and the overlapping portions make that proof invalid.

[45] Their statement is a bound of 4ϵ, but their proof gives the stronger result listed above. We choose the stronger form to help minimize the number of qubits needed for simulations.

[46] The induced distribution from a state $|\phi\rangle$ is $\mathcal{D}(k) = |\langle k|\phi\rangle|^2$.

Combining this with theorem A.14 and lemmata A.15 and A.16 gives the final result

Theorem A.17.

1) *Given an odd integer* $N \geq 13$, *and any* $\sqrt{2} \geq \epsilon > 0$. *Choose* $L \geq 16$ *and* $M \geq LN$ *both integral powers of* 2 *satisfying*

$$\frac{2}{\pi}\sqrt{\frac{22\ln^2 N}{L} + \frac{32N^2}{LM}} + \frac{\pi LN}{M\sqrt{3}} \leq \frac{\epsilon}{\sqrt{2}} \tag{179}$$

Then there is a unit vector $|\psi\rangle$ *such that the output* $|v\rangle$ *of the algorithm in section A.2.3 satisfies*

$$||v\rangle - F_N|u\rangle \otimes |\psi\rangle|| \leq \epsilon \tag{180}$$

2) *We can always find such an* L *and* M *by choosing*

$$L = c_1 \frac{\sqrt{N}}{\epsilon^2} \tag{181}$$

$$M = c_2 \frac{N^{\frac{3}{2}}}{\epsilon^3} \tag{182}$$

for some constants c_1, c_2 *satisfying*

$$65 \leq c_1 \leq 2 \times 65 \tag{183}$$

$$735 \leq c_2 \leq 2 \times 735 \tag{184}$$

3) *The algorithm requires* $\lceil \log M \rceil + 2$ *qubits. By claim* 2 *a sufficient number of qubits is then* $\left\lceil 12.53 + 3\log\frac{\sqrt{N}}{\epsilon} \right\rceil$. *The algorithm has operation complexity* $O(\log M(\log\log M + \log 1/\epsilon))$. *Again using claim* 2 *yields an operation complexity of*

$$O\left(\log\frac{\sqrt{N}}{\epsilon}\left(\log\log\frac{\sqrt{N}}{\epsilon} + \log 1/\epsilon\right)\right) \tag{185}$$

4) *The induced probability distributions* $\mathcal{D}_v$ *from the output and* $\mathcal{D}$ *from* $F_N|u\rangle \otimes |\psi\rangle$ *satisfy*

$$|\mathcal{D}_v - \mathcal{D}| \leq 2\epsilon + \epsilon^2 \tag{186}$$

Proof. Claim 1 follows directly from theorem A.14 and lemma A.15. Claim 1 and lemma A.16 give claim 4.

To get claim 2, note that for the bound to be met, we must have $\frac{\ln^2 N}{L} < \epsilon^2$, $\frac{N^2}{LM} < \epsilon^2$, and $\frac{LN}{M} < \epsilon$. Trying to keep M small as N and ϵ vary leads to the forms for L and M chosen. If we substitute lines 181 and 182 into 179 and simplify, we get

$$\frac{4}{\pi}\sqrt{\frac{11\ln^2 N}{c_1\sqrt{N}} + \frac{16\epsilon^3}{c_1 c_2}} + \frac{\pi\sqrt{2}}{\sqrt{3}}\frac{c_1}{c_2} \leq 1 \tag{187}$$

The left hand side is largest when $\epsilon = \sqrt{2}$ and $N = 55$, so it is enough to find constants c_1 and c_2 such that

$$\frac{4}{\pi}\sqrt{\frac{11\ln^2 55}{c_1\sqrt{55}} + \frac{32\sqrt{2}}{c_1 c_2}} + \frac{\pi\sqrt{2}}{\sqrt{3}}\frac{c_1}{c_2} \leq 1 \tag{188}$$

Ultimately we want L and M to be powers of two, so we find a range for each of c_1 and c_2 such that the upper bound is at least twice the lower bound, and such that all pairs of values (c_1, c_2) in these ranges satisfy inequality 188. To check that the claimed ranges work, note that for a fixed c_1, the expression increases as c_2 decreases, so it is enough to check the bound for $c_2 = 735$. After replacing c_2 in the expression with 735, the resulting expression has first and second derivatives with respect to c_1 over the claimed range, and the second derivative is positive, giving that the maximum value is assumed at an endpoint. So we only need to check inequality 188 at two points: $(c_1, c_2) = (65, 735)$ and $(2 \times 65, 735)$, both of which work. Thus the bound is met for all (c_1, c_2) in the ranges claimed. With these choices for M and L, note that $L \geq 16$ and $M \geq LN \Leftrightarrow c_2 \geq \epsilon c_1$, which is met over the claimed range, so all the hypothesis for claim 1 are satisfied.

Finally, to prove claim 3, algorithm A.2.3 and the proof of lemma A.5 give that we need $\lceil \log N \rceil$ qubits in the first register and $\max\{\lceil \log L \rceil, \lceil \log(2\alpha + 1) \rceil\}$ qubits in the second register. $L \leq \frac{M}{N} < 2\alpha + 1$ gives that it is enough to have $\lceil \log(2\alpha + 1) \rceil$ qubits in the second register. Then $2\alpha + 1 \leq \frac{M}{2N} + 2$ gives

$$\lceil \log(2\alpha + 1) \rceil \leq \lceil 1 + \log M - \log N \rceil = 2 + \lceil \log M \rceil - \lceil \log N \rceil \tag{189}$$

Thus $\lceil \log M \rceil + 2$ is enough qubits[47] for the algorithm. By claim 2, we can take $M \leq 2 \times 735 \frac{N^{3/2}}{\epsilon^3}$ giving $\lceil \log M \rceil + 2 \leq \left\lceil 12.53 + 3 \log \frac{\sqrt{N}}{\epsilon} \right\rceil$.

As noted in [55] and [57], the most time consuming step in algorithm A.2.3 is the F_M Fourier computation. Coppersmith [35] (reproduced in section A.1) shows how to ϵ approximate the quantum Fourier transform for order $M = 2^m$ with operation complexity of $O(\log M(\log\log M + \log 1/\epsilon))$. Using this to approximate our approximation within error ϵ gives the time complexities in claim 3, finishing the proof. □

Appendix B. Graph Reductions

B.1. **Basic Graph Algorithm Relations.** Note that G in this section is no longer a group as in the rest of the paper, but a *graph*.

Following Mathon [94], we show several graph isomorphism problems to be polynomially equivalent. If the ability to solve problem P_1 allows solving problem P_2 with polynomially many uses of P_1, we say P_2 is polynomially reducible to P_1, and write $P_2 \propto_p P_1$. If $P_2 \propto_p P_1$ and $P_1 \propto_p P_2$ then we say P_1 and P_2 are *polynomially equivalent*.

Given two undirected graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ with vertex sets V_i and edge sets E_i, $i = 1, 2$, we say G_1 is isomorphic to G_2, written $G_1 \cong G_2$, if there exists a bijection $\rho : V_1 \to V_2$ such that for all $x, y \in V_1$, $(x, y) \in E_1$ if and only if $(\rho x, \rho y) \in E_2$.

Denote the group of automorphisms of G by aut G. The automorphism partition $\mathcal{P}$ denotes the set of disjoint orbits of each vertex under aut G.

[47] An example requiring $\lceil \log M \rceil + 2$ qubits is $M = 1024$, $N = 65$, so the bound is tight.

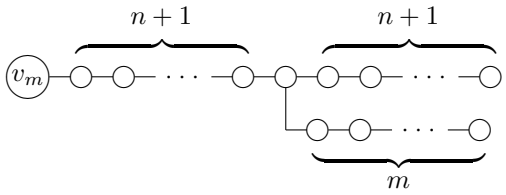
We consider the following six problems:

ISO(G_1, G_2)	isomorphism recognition for G_1 and G_2,
IMAP(G_1, G_2)	isomorphism map from G_1 onto G_2 if it exists,
ICOUNT(G_1, G_2)	number of isomorphisms from G_1 to G_2,
ACOUNT(G)	number of automorphisms of G,
AGEN(G)	generators of the automorphism group of G,
APART(G)	automorphism partition of G.

Surprisingly,

Theorem B.1. *The problems* **ISO**, **IMAP**, **ICOUNT**, **ACOUNT**, **AGEN**, *and* **APART** *are polynomially equivalent.*

Before proving this we define some notation. Suppose $G(V, E)$ is a graph with n vertices. We define graph labels: Let $G_{v_1,\ldots,v_k}$ denote a copy of G with unique distinct labels attached to the vertices $v_1, \ldots, v_k \in V$. This can be accomplished in the following manner. To vertex v_m, $1 \le m \le k$, attach label "m", which is a new graph using $2n + m + 3$ vertices as follows:



This modification has the property that vertices $v_1, \ldots, v_k$ are fixed by any $\rho \in \text{aut } G_{v_1,\ldots,v_k}$, and also there is a natural inclusion aut $G_{v_1,\ldots,v_k} \subseteq$ aut G, obtained by ignoring the labels in aut G. Finally, labelling all vertices adds $O(n^2)$ new vertices, retaining polynomial algorithm equivalence between problems on G and $G_{v_1,\ldots,v_k}$.

Proof. (Following Mathon [94])

IMAP $\propto_p$ **ISO**: Let $v_1, \ldots, v_n$ be the vertices of G_1. If G_2 does not have n vertices then there is no isomorphism. Otherwise use **ISO** at most n times to find a $u_1 \in V_2$ such that there is an isomorphism $G_{1v_1} \cong G_{2u_1}$, otherwise there is no isomorphism. If such a u_1 is found, there is an isomorphism ρ mapping $v_1 \to u_1$. Continue fixing $v_1, \ldots, v_j$, $u_1, \ldots, u_{j-1}$ and searching for $u_j \in V_2$. This constructs an isomorphism if it exists, calling **ISO** $O(n^2)$ times.

ACOUNT $\propto_p$ **ISO**: For a given labelling $G_{v_1,\ldots,v_k}$ of a graph G let aut $G_{v_1,\ldots,v_k}$ be the corresponding automorphism group, which is the subgroup of aut G that fixes the vertices $v_1, \ldots, v_k$. We will show that $|\text{aut } G_{v_1,\ldots,v_{k-1}}| = d_k |\text{aut } G_{v_1,\ldots,v_k}|$, where d_k is the size of the orbit π_k of v_k in aut $G_{v_1,\ldots,v_{k-1}}$. For $1 \le i \le d_k$ let $\phi_i \in \text{aut } G_{v_1,\ldots,v_{k-1}}$ be an automorphism which maps the i^{th} vertex of π_k onto v_k. Then every $\tau \in \text{aut } G_{v_1,\ldots,v_{k-1}}$ is a product of a unique $\phi \in \{\phi_1, \ldots, \phi_{d_k}\}$ and a unique $\psi \in \text{aut } G_{v_1,\ldots,v_k}$. Since $|\text{aut } G_{v_1,\ldots,v_n}| = 1$, $|\text{aut } G| = d_1 d_2 \ldots d_n$, and each d_k can be found by solving **ISO** at most $n - k$ times. Thus we compute $|\text{aut } G|$ by calling **ISO** at most $O(n^2)$ times.

ICOUNT $\propto_p$ **ISO**: Let N_I be the number of isomorphisms from G_1 onto G_2. If $G_1 \ncong G_2$ then $N_I = 0$ is determined with one call to **ISO**. Otherwise we claim $N_I = |\text{aut } G_1| = |\text{aut } G_2|$, in which case we use **ACOUNT** on G_1 and on G_2,

calling **ISO** $O(n^2)$ times as above. The claim is proved by the fact that if $\sigma : V_1 \to V_2$ is an isomorphism from G_1 onto G_2 and ρ is an automorphism of G_2 then $\rho \circ \sigma$ is also a graph isomorphism. Moreover any isomorphism σ' can be uniquely expressed as $\sigma' = \rho' \circ \sigma$ where $\rho' \in |\text{aut } G_2|$. This $1-1$ correspondence between $|\text{aut } G_2|$ and the number of isomorphisms $G_1 \to G_2$ proves the claim.

APART $\propto_p$ **ISO**: Two vertices $u, v \in V$ of a graph G belong to the same cell of the automorphism partition $\mathcal{P}$ of G if $G_u \cong G_v$ for identical labels of u and v. Hence at most $O(n^2)$ calls to **ISO** are needed to find $\mathcal{P}$, trying all combinations of u and v.

AGEN $\propto_p$ **ISO**: Applying **IMAP** to the graphs $G_{v_1,\ldots,v_k}$ and $G_{v_1,\ldots,v_{k-1},v_l}$ with identical labels for $k+1 \leq l \leq n$ we determine the sets of automorphisms $\Phi_k = \{\phi_1, \ldots, \phi_{d_k}\}$ at level k (using notation from above). From the proof of **IMAP** $\propto_p$ **ISO** it follows that the set $\Phi_1 \bigcup \cdots \bigcup \Phi_n$ of maps generates aut G. Since $d_k \leq n - k + 1$ implies

$$\sum_{k=1}^{n} d_k \leq n^2$$

we see that at most $O(n^4)$ calls to **ISO** solve **AGEN**. This order can be reduced to $O(n^3)$ using **APART** to find the partition of $G_{v_1,\ldots,v_k}$ and by generating only one ϕ_i for every feasible orbit in $V \setminus \{v_1, \ldots, v_k\}$ at each level k. It is easily shown at most n generators are produced in this case.

ISO $\propto_p$ **IMAP**, **ICOUNT**: A single call to either **IMAP** or **ICOUNT** gives **ISO**.

From now on assume G_1 and G_2 are each connected (otherwise we may use their complements).

ISO $\propto_p$ **ACOUNT**: Apply **ACOUNT** to G_1, G_2, and $G_3 = G_1 \bigcup G_2$. If $|\text{aut } G_1| = |\text{aut } G_2|$ and $|\text{aut } G_1| \cdot |\text{aut } G_2| \neq |\text{aut } G_3|$ then $G_1 \cong G_2$, else $G_1 \ncong G_2$.

ISO $\propto_p$ **AGEN**: Apply **AGEN** to $G_3 = G_1 \bigcup G_2$. If $\sigma(v) = u$ for some $v \in V_1$, $u \in V_2$, and $\sigma \in$ aut G_3 then $G_1 \cong G_2$, else $G_1 \ncong G_2$. From the proof of **AGEN** $\propto_p$ **ISO** we can assume we have at most n^2 generators of aut G to check, so this can be checked in at most $n^4 = |V_1||V_2||n^2|$ operations, assuming constant time to check one.

ISO $\propto_p$ **APART**: Apply **APART** to $G_3 = G_1 \bigcup G_2$. If v, u belong to the same cell of the partition $\mathcal{P}$ of G_3 for some $u \in V_1$, $u \in V_2$, then $G_1 \cong G_2$, otherwise $G_1 \ncong G_2$. This can be checked quickly by scanning the partition once.

This completes the proof of the theorem. □

Finally, following [81, Theorem 1.31], we can reduce this to efficient algorithms solving the following graph automorphism questions:

- **GA**(G) - Given a graph G, decide whether its automorphism group has a nontrivial automorphism.
- **GA1**(G) - Given that $|\text{aut } G| \in \{1, 2\}$, determine $|\text{aut } G|$

We note that **GA**(G) seems easier than **ISO**(G_1, G_2) [81].

As above, we are able to reduce the seemingly more complex **GA** to **GA1**:

Theorem B.2. GA $\propto_p$ **GA1**

For a proof, see [81].

So there are many ways to approach the graph isomorphism and graph automorphism problems, some of which at first glance seem easier than the original

question. For the purposes of quantum computation, and in particular reducing these questions to finding hidden subgroups of S_n, see the next section (B.2).

As a final note, there are far reaching proofs that show determining isomorphism between any finite algebraic structures (such as rings, groups, fields, etc.) is polynomial-time many-to-one reducible to **ISO**, making a fast **ISO** algorithm extremely useful across many disciplines [95]. These are a few of the reasons that an efficient **ISO** algorithm has seen such strong research interest.

B.2. **Quantum HSP for Graph Isomorphism.** We want to show how being able to find hidden subgroups H of S_n allows solving **ISO**, which then gives efficient algorithms for all the problems in the previous section. We define our hidden function $f : S_n \to \{\text{permutations of } G\}$ by $f(\pi) = \pi(G)$. So f applies a permutation π to the vertices of G. We need to show f separates cosets of $H = \text{aut } G$, and that f is efficiently computable. Then an algorithm giving generators of H, i.e., giving an algorithm for **AGEN**, gives the desired algorithm for **ISO**.

To make this precise, suppose G is represented on a computer by a list of pairs (v_i, v_j) of vertices where there is an edge from vertex i to vertex j. Assume this list is sorted and each pair is sorted. We define f at the programming level as taking a permutation (which can just be a list π of n pairs $i \to \pi(i)$) and doing the following two steps: apply the permutation to the integers v_i in time $O(\#\text{ edges})$, then sort the result efficiently by usual methods (Quicksort, etc.). Thus f can be computed efficiently, and leaves G in a state where comparisons can be done quickly (that is, $G \cong \pi(G)$ if and only if $G = f(\pi)$ using this encoding, which you should check).

Let S_n act on the n vertices of G, and let $H = \text{aut}(G) < S_n$. To show f separates cosets of H, we want $f(\pi_1) = f(\pi_2)$ if and only if $\pi_1 H = \pi_2 H$, which follows from

$$f(\pi_1) = f(\pi_2) \Leftrightarrow \pi_1(G) = \pi_2(G) \Leftrightarrow \pi_2^{-1}\pi_1 G = G \Leftrightarrow$$
$$\pi_2^{-1}\pi_1 \in \text{aut } G \Leftrightarrow \pi_2^{-1}\pi_1 H = H \Leftrightarrow \pi_1 H = \pi_2 H.$$

This shows f can be used in the standard quantum Fourier sampling algorithm to find generators for H. If this can be done efficiently is an open question.

Appendix C. Quantum Mechanics Details

C.1. The Rules and Math of Quantum Mechanics.

C.1.1. *Enter the Qubit.* First we start out with the basic block of quantum computing. Analogous to the bit in classical computing, there is a quantum bit in quantum computing. A classical bit is a 2 state system, with the states denoted 0 and 1. A classical bit is always in one of those states or the other, and measuring the state return a 0 or 1 with certainty. n bits can be in exactly one of 2^n different ordered states, usually denoted $000\ldots00$, $000\ldots01$,$\ldots$,$111\ldots11$.[48]

Quantum bits (which we shall call qubits) similarly can exist in two states, which we call $|0\rangle$ and $|1\rangle$. However, they behave as if existing in many "in between" states. A quantum bit can be physically represented by any two state (or more) system, such as electron spin up and down, photon energy states, atomic energy levels, molecular vibrational freedom, and many others. For our purposes we assume physical representations are available (they are).

To make the concept of a qubit precise, we define

[48] "There are only 10 kinds of people in the world. Those who understand binary and those who don't."

Definition C.1 (Qubit)**.** *A* **qubit** *(or quantum-bit) is a unit vector in* $\mathbb{C}^2$*.*

Definition C.2 (State vector)**.** *The* **state** *of a quantum system is a (column) vector in some vector space, written* $|\psi\rangle$*.*

With this definition, we fix an orthonormal basis of (column) vectors, labelled $|0\rangle = \binom{1}{0}$ and $|1\rangle = \binom{0}{1}$. It will turn out that physically, *we can only distinguish orthogonal quantum states*, thus the orthogonal requirement. And considerations of probability will make the normality convenient, thus we fix an orthonormal basis. Any such basis of $\mathbb{C}^2$ will work, but we choose the above representations since they are good to work with. Finally, we make a qubit a unit vector because, again, it makes calculations cleaner, and has some physical significance.

Now for the differences from classical bits. A qubit can be *any* unit vector, not just those corresponding to $|0\rangle$ and $|1\rangle$. A qubit can be in the state

$$\alpha|0\rangle + \beta|1\rangle \tag{190}$$

where α and β are complex numbers, with $|\alpha|^2 + |\beta|^2 = 1$. While it only takes one "bit" to fully describe the state of a classical bit, it takes two complex numbers to completely describe the state of one qubit, which intuitively is infinitely more information! However we will see there are practical limitations to the amount of "information" one can retrieve from a single qubit.

This gives us the first of four postulates of quantum mechanics:

Quantum Mechanics Postulate 1: State Space Associated to an isolated physical system is a complex vector space with inner product (a Hilbert space) known as the *state space* of the system. The system is completely described by its *state vector*, which is a unit vector in the system's state space. Thus an n-qubit system is a unit vector in $\mathbb{C}^{2^n}$.

We will explain the inner product below (we can use the Euclidean one).

C.1.2. *How to "Measure" a Qubit.* In principle you could store the knowledge in the Library of Congress on one qubit, but *you could never retrieve it.* When you read out the value in a qubit in the state in equation 190, it returns the state $|0\rangle$ with probability $|\alpha|^2$, or it returns the state $|1\rangle$ with probability $|\beta|^2$, and then the qubit assumes the state just returned. Thus we can only get one state back out from the qubit, which collapses (destroys) the rest of the information in the qubit.

For example, suppose we have a qubit in the state

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{191}$$

What are the odds that it returns a $|1\rangle$ when measured? A $|0\rangle$?

This generalizes to multiple qubits as we soon see.

One last point is worth mentioning - there is a useful way to visualize operations on a single qubit, using the **Bloch sphere**. It will turn out that under observation, states $|\psi\rangle$ and $e^{i\theta}|\psi\rangle$ have the same behavior, so we can modify a state up to the phase $i\theta$, where $i = \sqrt{-1}$. So given a single qubit state $\alpha|0\rangle + \beta|1\rangle$, we can remove a phase to write

$$\alpha|0\rangle + \beta|1\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right) \tag{192}$$

Since the phase out front has no effect on measurements, we can use θ and φ for spherical coordinates

$$x = \cos\varphi \sin\theta \tag{193}$$

$$y = \sin\varphi \sin\theta \tag{194}$$

$$z = \cos\theta \tag{195}$$

This allows us to picture a qubit as a point on a three dimensional sphere, and visualize operations upon a qubit.

Unfortunately, this has no known generalization to multiple qubits

C.1.3. *Qubits Galore.* Similar to concatenating n classical bits to get "bitstrings", we concatenate qubits to get larger systems. Two qubits form a space spanned by four vectors

$$|0\rangle \otimes |0\rangle, \;\; |0\rangle \otimes |1\rangle, \;\; |1\rangle \otimes |0\rangle, \;\; \text{and } |1\rangle \otimes |1\rangle \tag{196}$$

where we will define the "tensor product" $\otimes$ in a moment. Shorthand for the above expressions is

$$|00\rangle, \;\; |01\rangle, \;\; |10\rangle, \;\; \text{and } |11\rangle \tag{197}$$

Definition C.3. *The* **tensor product** *of two vectors $x = (x_1, x_2, \ldots, x_n)^T$ and $y = (y_1, y_2, \ldots, y_m)^T$ as the vector in nm dimensional space given by*

$$x \otimes y = \begin{pmatrix} x_1 y \\ x_2 y \\ \cdots \\ x_n y \end{pmatrix} = \begin{pmatrix} x_1 y_1 \\ x_1 y_2 \\ \cdots \\ x_1 y_m \\ x_2 y_1 \\ \cdots \\ x_n y_m \end{pmatrix} \tag{198}$$

Homework C.1. *Check this definition does not depend on a choice of basis.*

Now we can check the second basis element (dictionary ordering)

$$|01\rangle = |0\rangle \otimes |1\rangle \tag{199}$$

$$= \tbinom{1}{0} \otimes \tbinom{0}{1} \tag{200}$$

$$= \begin{pmatrix} 1\binom{0}{1} \\ 0\binom{0}{1} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \tag{201}$$

and we get the second usual basis element of $\mathbb{C}^4$. This works in general; that is, the vector corresponding to the state $|n\rangle$ where n is a binary number, is the $(n+1)^{\text{th}}$ standard basis element. We also use the decimal shorthand sometimes: $|32\rangle$ is the 33rd standard basis vector in some space which would be clear from context.

Back to the inner product from postulate 1: We write it using the "braket" notation, where the symbol $|k\rangle$ is called a ket, and the dual $\langle j|$ is a bra. Given a state (ket) $|\psi\rangle = \sum \alpha_j |j\rangle$, we define the dual (bra) as the conjugate transpose, that is,

$$\langle\psi| = |\psi\rangle^\dagger = \sum \alpha_j^* \langle j| \tag{202}$$

Together we write $\langle j|k\rangle$, which is the "braket" of states $|j\rangle$ and $|k\rangle$. Since the states are orthonormal, $\langle j|k\rangle$ is 1 if and only if $j = k$, otherwise it is zero. We extend this inner product $\langle -, -\rangle$ to general states via linearity. Thus states $|\psi_1\rangle = \sum \alpha_j|j\rangle$ and $|\psi_2\rangle = \sum \beta_k|k\rangle$ give

$$\begin{aligned}\langle\psi_1|\psi_2\rangle &= \textstyle\sum_j \alpha_j^* \langle j| \sum_k \beta_k|k\rangle \\ &= \textstyle\sum_{j,k} \alpha_j^*\beta_k\langle j|k\rangle \qquad = \displaystyle\sum_m \alpha_m^*\beta_m\end{aligned}$$

So we have the equivalent notations for a 5-qubit state:

$$\begin{aligned}|1\rangle \otimes |0\rangle \otimes |0\rangle \otimes |1\rangle \otimes |0\rangle = &\quad |10010\rangle \\ = &\quad |18\rangle\end{aligned}$$

It is worth noting that not all composite states are simple tensor products of single states. One of the simplest is one of the 2 qubit Bell states, $\beta_{00} = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$. This is an example of an *entangled state* which turns out to be a very useful computational resource later.

Homework C.2. *Prove β_{00} is not of the form $|\psi\rangle \otimes |\varphi\rangle$.*

When appropriate, we may drop the normalization factor to clean up calculations. Then we could write $\beta_{00} = |00\rangle + |11\rangle$, with the understanding this needs to be normalized.

C.1.4. *Measuring Revisited.* Now - how about measuring these states? An arbitrary 2-qubit state is

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

with complex valued α_{ij}. Requiring $\sum_{ij} |\alpha_{ij}|^2 = 1$ is called the "normalization requirement", and we assume all states are normalized. Sometimes to avoid clutter we will drop the coefficients.

Suppose we only measure the first qubit of $|\psi\rangle$. We will obtain $|0\rangle$ with probability $|\alpha_{00}|^2 + |\alpha_{01}|^2$, that is, we obtain a state with probability equal to the sum of the magnitudes of all states that contribute. After measuring, we know the first qubit is $|0\rangle$, so only those type of states are left, causing the new state to be

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

Notice the new normalization factor in the denominator. Again, this idea generalizes to arbitrary (finite) dimension.

Thus we have a way to denote arbitrary quantum states on n qubits:

$$|\psi\rangle = \sum_{j=0}^{2^n-1} \alpha_j|j\rangle \tag{203}$$

where the α_i are complex numbers satisfying the normalization requirement. Measuring $|\psi\rangle$ returns state $|j\rangle$ with probability $|\alpha_j|^2$, and then becomes state $|j\rangle$

C.1.5. *Qubit Evolution.* We would like our quantum computers to work similar to classical computers. Classically, a very basic operation at the bit level is the NOT gate, which flips bits, that is 0 becomes 1 and 1 becomes 0. So the quantum version would take the state $\alpha|0\rangle + \beta|1\rangle \xrightarrow{NOT} \beta|0\rangle + \alpha|1\rangle$. It is east to check the matrix

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{204}$$

performs the desired operation, by multiplying X on the left of the state. The name X is historical, and we will see the exponential of X rotates qubits around the x-axis on the Bloch sphere. Since X acts like a NOT gate on a qubit, it is often called the NOT operator.

For fun, we compute "the square root of NOT." We want an operator $\sqrt{\text{NOT}}$ that when applied twice to a qubit, has the effect of NOT. This procedure will be useful when we need to construct quantum circuits and when we explain exponentials.

In general, given a function $f(t)$ of one complex variable, we extend this definition to diagonalizable matrices $M = \text{diag}(m_1, m_2, \dots m_n)$ via:

$$f(M) = \text{diag}(f(m_1), f(m_2), \dots, f(m_n)) \tag{205}$$

Since we want $\sqrt{\text{X}}$, we need to diagonalize X. Note the eigenvectors of X are $\binom{1}{1}$ and $\binom{1}{-1}$. Setting a matrix P with these as column vectors, we have under this basis change the diagonal matrix

$$\begin{aligned} PXP^{-1} &= \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \end{aligned}$$

Applying $f(t) = \sqrt{t}$, and changing the basis back gives

$$\begin{aligned} P^{-1} f \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} P &= \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ &= \tfrac{1}{2} \begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix} \\ &= \sqrt{\text{NOT}} \end{aligned}$$

It is an easy check to see that $\sqrt{\text{NOT}}^2 = X$.

This process of diagonalizing an operator, applying a function, and restoring the basis will be invaluable later.

Homework C.3. *What is the effect of $e^{-i\theta X/2}$ on the Bloch sphere, where θ is a real number?*

C.1.6. *A Universal Quantum Gate?* It is a basic result in computer science that any circuit can be built with NAND gates, which performs the following operation on two bits a and b:

$a \setminus b$	0	1
0	1	1
1	1	0

Any function on n bits can be built up from NAND gates. However the general function requires exponentially many gates, so in practice we are restricted in the functions we utilize.

So is there a similar "gate" for quantum computing? Yes, and no. It will take a while to answer this precisely, but there are finite (and small) sets of gates sufficient to *approximate* any desired quantum operation to any degree of accuracy in an efficient manner.[49]

To understand what operations we can physically apply to a qubit (or set of qubits), we are led to study rules from quantum mechanics. It has become clear that abstract models of computation and information theory should be derived from physical law, rather than as standalone mathematical structures, since it is ultimately physical law that determines computability and information. Observation has led researchers to believe that at the quantum level, the following two facts hold:

- All quantum evolution is reversible. That is very unlike the classical case, where for example NAND is not reversible.[50] This is illustrated by the fact that an electron in orbit does not emit radiation and spiral into the nucleus.
- Quantum evolution is linear. That is, if an experiment is done on the state $|0\rangle$ and on the state $|1\rangle$, then when performed on mixed states the resulting state is the same state as if the initial two answers were added.

So we are left with "reversible" linear operators on the states, that is, matrices! Since the resulting state should satisfy the normalization requirement also, it turns out that any **unitary** operation is allowed. Recall U unitary means $UU^\dagger = I$. We now have :

Quantum Mechanics Postulate 2: State Evolution The evolution of a *closed* quantum system is described by a *unitary transformation.* That is, the state of a system $|\psi\rangle$ at time t_1 is related to the state $|\psi'\rangle$ at time t_2 by a unitary operator U which depends only on the times t_1 and t_2,

$$|\psi\rangle = U|\psi'\rangle \tag{206}$$

Now we know how to specify quantum states and what is legal for manipulating the state.

C.1.7. *Intermission - Linear Algebra Review.* We will need several facts, terms, and theorems from linear algebra. It will be easiest to just fire them off: (we also combine some previous facts here for the heck of it)

Definition C.4. *Let H, A, B, U be linear operators on a vector space V.*

(1) *$H^\dagger$ is the conjugate transpose of H.*
(2) *H is **Hermitian** or self-adjoint if $H = H^\dagger$.*
(3) *$|\psi\rangle$ is a column vector.*
(4) *$\langle\psi|$ is the dual to $|\psi\rangle$, defined $\langle v| \equiv |v\rangle^\dagger$.*
(5) *$|\psi\phi\rangle = |\psi\rangle|\phi\rangle = |\psi\rangle \otimes |\phi\rangle$.*
(6) *$[A, B] = AB - BA$.*
(7) *$\{A, B\} = AB + BA$.*
(8) *A is **normal** if $A^\dagger A = AA^\dagger$.*
(9) *U is **unitary** if $U^\dagger U = I$.*

[49] The Solovay-Kitaev theorem says that for any gate U on a single qubit, and given any $\epsilon > 0$, it is possible to approximate U to a precision ϵ using $\Theta(\log^c(1/\epsilon))$ gates from a fixed, finite set, where $1 \leq c \leq 2$. Determining c is an open problem.

[50] Charles Bennett of IBM research showed in the 1970's that energy is used in computations to *destroy* information. Lossless computation can theoretically be done with no energy usage whatsoever!

(10) *A is **positive** if $\langle\psi|A|\psi\rangle \geq 0$ for all ψ.*
(11) *$\langle\psi|A|\phi\rangle$ is the inner product of ψ and $A|\phi\rangle$.*
(12) *We define specific matrices (the first 4 are the Pauli matrices)*
$\sigma_0 = I$,
$\sigma_1 = \sigma_x = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$,
$\sigma_2 = \sigma_y = Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$,
$\sigma_3 = \sigma_z = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$,
$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, $S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$, $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix}$
(13) *For a unit vector $\vec{n} = (n_x, n_y, n_z) \in \mathbb{R}^3$, define $\vec{n}.\vec{\sigma} \equiv n_x\sigma_x + n_y\sigma_y + n_z\sigma_z$.*
(14) ***Bloch Sphere** Given a state $a|0\rangle + b|1\rangle$ we may assume a is real by phase rotation. Then define for $\phi \in [0, 2\pi]$ and $\theta \in [0, \pi]$*

$$\cos\left(\frac{\theta}{2}\right) = a \tag{207}$$

$$e^{i\phi}\sin\left(\frac{\theta}{2}\right) = b \tag{208}$$

Then the point on the Bloch Sphere is $(\cos\phi\sin\theta, \sin\phi\sin\theta, \cos\theta)$.
(15) *Define the three rotation matrices: $R_x(\theta) = e^{-\theta X i/2} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}X = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$*
$R_y(\theta) = e^{-\theta Y i/2} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Y = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$
$R_z(\theta) = e^{-\theta Z i/2} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Z = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$
(16) *For a composite quantum system AB, the **partial trace** is an operator from density operators on AB to density operators on A defined for $tr_B(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|) = \langle b_2|b_1\rangle|a_1\rangle\langle a_2|$, and extended by linearity. On matrices: let $\dim A = n$, $\dim B = m$, then it takes a mn by mn matrix, and replaces each m by m sub-block with its trace to give a n by n matrix.*
(17) *The **Bell States** are the 2-qubit basis states*

$$|\beta_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{209}$$

$$|\beta_{01}\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}} \tag{210}$$

$$|\beta_{10}\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}} \tag{211}$$

$$|\beta_{11}\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}} \tag{212}$$

Note: The four Pauli matrices (I, X, Y, and Z) have significance, since they form a basis of all linear operators on one qubit, and correspond to similarly named actions on the Bloch sphere.

We can write operators like X in an equivalent operator notation, which is often convenient to use in calculations. Noting that $\langle 0|$ is a row vector, then $|0\rangle\langle 0|$ is a 2×2 matrix. We can write X as:

$$X = |0\rangle\langle 1| + |1\rangle\langle 0| \tag{213}$$

$$= \tbinom{1}{0}(0\ 1) + \tbinom{0}{1}(1\ 0) \tag{214}$$

$$= \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{215}$$

This is interpreted quickly: X sends state 0 to 1, and vice versa.

Example: As an example calculation, we compute $\langle \beta_{00}|I_2 \otimes X|\beta_{10}\rangle$ two different ways. The first way is matrix multiplication: Noting that $|00\rangle = (1,0,0,0)^T$ and $|11\rangle = (0,0,0,1)^T$, we have

$$\langle \beta_{00}|I \otimes X|\beta_{10}\rangle = \left(\tfrac{|00\rangle+|11\rangle}{\sqrt{2}}\right)^\dagger \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \left(\tfrac{|00\rangle-|11\rangle}{\sqrt{2}}\right) \tag{216}$$

$$= \left(\tfrac{1}{\sqrt{2}}\right)^2 \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ -1 \end{pmatrix} \tag{217}$$

$$= 0 \tag{218}$$

For the other method, note as operators we can write $I = |0\rangle\langle 0| + |1\rangle\langle 1|$, and X swaps basis vectors, giving $X = |0\rangle\langle 1| + |1\rangle\langle 0|$. Then we have

$$I \otimes X = (|0\rangle\langle 0| + |1\rangle\langle 1|) \otimes (|0\rangle\langle 1| + |1\rangle\langle 0|) \tag{219}$$

$$= |00\rangle\langle 01| + |01\rangle\langle 00| + |10\rangle\langle 11| + |11\rangle\langle 10| \tag{220}$$

where we used the fact $|a\rangle\langle b| \otimes |c\rangle\langle d| = |ac\rangle\langle bd|$. Apply this and use orthonormality,

$$\langle \beta_{00}|I \otimes X|\beta_{10}\rangle = \left(\tfrac{\langle 00|+\langle 11|}{\sqrt{2}}\right)(|00\rangle\langle 01| + |01\rangle\langle 00| + |10\rangle\langle 11| + |11\rangle\langle 10|)\left(\tfrac{|00\rangle-|11\rangle}{\sqrt{2}}\right) \tag{221}$$

$$= \left(\tfrac{1}{\sqrt{2}}\right)^2 (0 + 0 + 0 + \cdots + 0) \tag{222}$$

$$= 0 \tag{223}$$

where we get terms like $\langle 00|00\rangle\langle 01|00\rangle = 1 \cdot 0 = 0$.

Homework C.4. *Write the matrices above in operator form for practice.*

Homework C.5. *Compute the eigen-values and eigen-vectors for the matrices defined above. They will be useful.*

Homework C.6. *Understand the behavior of each matrix above on the Bloch sphere representation of a qubit.*

C.1.8. *Useful Linear Algebra Theorems.*

Theorem C.5 (Cauchy Schwartz Inequality)**.** $|\langle v|w\rangle|^2 \leq \langle v|v\rangle\langle w|w\rangle$

Theorem C.6 (Spectral Decomposition)**.** *Any normal operator M on a vector space V is diagonal with respect to some orthonormal basis for V. Conversely, any diagonalizable operator is normal.*

Proof. Sketch: Induct on $d = \dim V$. $d = 1$ is trivial. Let λ be an eigenvalue of M, P the projector onto the λ eigenspace, and Q the projector onto the orthogonal complement. $M = PMP + QMQ$ is diagonal with respect to some basis (strip off an eigenvalue one at a time...) □

Check: There is a matrix P, with unit eigenvectors as columns, so that $PMP^\dagger$ is diagonal, with entries the eigenvalues.

Theorem C.7 (Simultaneous diagonalization)**.** *Suppose A and B are Hermitian operators on a vector space V. Then $[A, B] = 0 \Leftrightarrow$ there exists an orthonormal basis such that both A and B are diagonal with respect to that basis.*

Theorem C.8 (Polar decomposition)**.** *Let A be a linear operator on a vector space V. Then there exists a unitary U and positive operators J and K such that*

$$A = UJ = KU$$

where the unique J and K are given by $J \equiv \sqrt{A^\dagger A}$ and $K \equiv \sqrt{AA^\dagger}$. Moreover, A invertible implies U is unique.

Proof. $J \equiv \sqrt{A^\dagger A}$ is positive, so spectral gives $J = \sum_i \lambda_i |i\rangle\langle i|$, $(\lambda_i \geq 0)$. Let $|\phi_i\rangle = A|i\rangle$. For $\lambda_i \neq 0$, let $|e_i\rangle = |\phi_i\rangle/\lambda_i$. Extend to orthogonal basis $|e_i\rangle$, and define unitary $U \equiv \sum_i |e_i\rangle\langle i|$. This satisfies $A = UJ$. Multiply on left by adjoint $A^\dagger = JU^\dagger$ giving $J^2 = A^\dagger A$, so $J = \sqrt{A^\dagger A}$.

Then $A = UJ = UJU^\dagger U = KU$ with $K = UJU^\dagger$. This $K = \sqrt{AA^\dagger}$. □

Theorem C.9 (Singular value decomposition)**.** *Let A be a square matrix. Then there exists unitary U and V, and diagonal D, such that*

$$A = UDV$$

The diagonal elements of D are called singular values of A.

Proof. By polar decomposition, $A = SJ$ for S unitary and J positive. By spectral $J = TDT^\dagger$, T unitary, D diagonal with nonnegative entries. $U \equiv ST$ and $V \equiv T^\dagger$ completes the proof. □

Theorem C.10. *Every unitary 2×2 matrix can be expressed as*

$$\begin{pmatrix} e^{i\alpha} & 0 \\ 0 & e^{i\alpha} \end{pmatrix} \cdot \begin{pmatrix} e^{\frac{i\beta}{2}} & 0 \\ 0 & e^{-\frac{i\beta}{2}} \end{pmatrix} \cdot \begin{pmatrix} \cos\frac{\gamma}{2} & -\sin\frac{\gamma}{2} \\ \sin\frac{\gamma}{2} & \cos\frac{\gamma}{2} \end{pmatrix} \cdot \begin{pmatrix} e^{\frac{i\delta}{2}} & 0 \\ 0 & e^{-\frac{i\delta}{2}} \end{pmatrix} \tag{224}$$

Note: Notice the third matrix is a usual rotation in the plane. The 2nd and 4th matrices are Z-axis rotation on the Bloch sphere, and the first matrix is merely a phase shift of the entire state. This decomposition gives some intuition of how a single qubit operator acts.

Theorem C.11 (Z-Y decomposition for a single qubit)**.** *U is a unitary operation on a single qubit. Then there are real numbers $\alpha, \beta, \delta, \gamma$ such that*

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta)$$

Note: Similarly there are **X-Y**, **Z-X**, etc. decomposition theorems.

Theorem C.12 (ABC corollary)**.** *Suppose U is a unitary gate on a single qubit. Then there are unitary operators A, B, and C, such that $ABC = I$, and $U = e^{i\alpha} AXBXC$, where α is some overall phase factor.*

Proof. Apply theorem C.11 with $A \equiv R_z(\beta)R_y(\gamma/2)$, $B \equiv R_y(-\gamma/2)R_z(-(\delta+\beta)/2)$, and $C \equiv R_z((\delta-\beta)/2)$. □

This weird looking theorem becomes very useful when trying to construct quantum circuits. It allows one to use a Controlled NOT gate (a circuit that flips a qubit based on the state of another qubit) to contract arbitrary controlled U gates.

C.1.9. *Useful Linear Algebra Facts!* Here are some facts that help in computations and proofs when dealing with quantum computing.

(1) Any complex $n \times n$ matrix A can be written as a sum of 4 positive Hermitian matrices: $A = B + iC$ with B, C Hermitian $B = \frac{1}{2}(A^* + A)$, and C accordingly. Then any Hermitian B can be written as the sum of 2 positive Hermitian matrices $B = (B + \lambda I) - \lambda I$ where $-\lambda$ is the most negative eigenvalue of B.
(2) Every positive A is of the form BB^*.
(3) $|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2| = |a_1 b_1\rangle\langle a_2 b_2|$ (useful in partial trace operations).
(4) Trace of kets: $|\psi\rangle = \sum_{i,j} a_{ij}|ij\rangle$, when converted to a density matrix $\rho = |\psi\rangle\langle\psi|$, and then the trace is taken over the j, gives
$$tr_B(p) = \sum_i \left(\sum_j |a_{i,j}|^2 \right) |i\rangle\langle i|,$$
so it seems $tr_B(|\psi\rangle)$ should be something like $\sum_i \sqrt{\sum_j |a_{i,j}|^2}\ |i\rangle$. In particular, tracing out some columns in $|011010\rangle$ removes those columns, but the new kets are not a simple sum of the previous ones... It may be ok to sum probabilities, then sqrt when collapsing, but I am not clear.
(5) Unitary also satisfies $UU^\dagger = I$, so U is normal and has spectral decomposition (all QC ops unitary!).
(6) Unitary preserves inner products.
(7) **Positive ⇒ Hermitian ⇒ normal**.
(8) $A^\dagger A$ is positive for any linear operator A.
(9) Tensor of unitary (resp Hermitian, positive, projector) is unitary (resp,...).
(10) If $P = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is invertible, then $P^{-1} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$.
(11) Given eigenvectors v_1 and v_2 of B, with eigenvalues λ_1 and λ_2, create the change of basis matrix $P = \begin{pmatrix} v_1 & v_2 \end{pmatrix}$. Then the diagonal matrix D is
$$D = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} = P^{-1}BP$$
(12) W is a subspace of V with basis $|i\rangle$. Projection to W is $P = \sum_i |i\rangle\langle i|$. $Q = I - P$ is the orthogonal complement.
(13) Eigenvectors with distinct eigenvalues of a Hermitian operator are orthogonal.
(14) $\vec{n}.\vec{\sigma}$ has eigenvalues ± 1 with corresponding eigenvectors $\begin{pmatrix} n_z \pm 1 \\ n_x + in_y \end{pmatrix}$.
(15) U unitary ⇒ U has a spectral decomposition ⇒ U is diagonal in some orthonormal basis ⇒ $U = \mathrm{diag}(e^{i\alpha_1}, e^{i\alpha_2}, \ldots, e^{i\alpha_n})$ ⇒ U has a *unitary* n^{th} root V, $V^n = U$.
(16) tr $(|\psi\rangle\langle\phi|) = \langle\phi|\psi\rangle$.

(17) For unit vectors $\vec{r}$ and $\vec{s}$, $(\vec{r}.\vec{\sigma})\cdot(\vec{s}.\vec{\sigma}) = \vec{r}\cdot\vec{s}I + (\vec{r}\times\vec{s}).\vec{\sigma}$.

C.1.10. *Some Basic Identities.* There are lots of identities between the operators we have above which will be useful in reducing circuits later on. This is a good place to list some.

$$[X,Y] = 2iZ \quad [Y,Z] = 2iX \quad [Z,X] = 2iY$$
$$\{\sigma_i,\sigma_j\} = 2\delta ij \text{ if } i,j \neq 0 \quad \sigma_i^2 = I$$
$$R_z(\frac{\pi}{2})R_x(\frac{\pi}{2})R_z(\frac{\pi}{2}) = e^{-i\pi/2}H$$
$$XYX = -Y \Rightarrow XR_y(\theta)X = R_y(-\theta)$$
$$HXH = Z \quad HYH = -Y \quad HZH = X$$
$$HTH = phase * R_x(\frac{\pi}{4})$$

C is CNOT, X_j is X acting on qubit j, etc.

$$\begin{array}{rclrcl} CX_1X &=& X_1X_2 & CY_1C &=& Y_1X_2 \\ CZ_1C &=& Z_1 & CX_2C &=& X_2 \\ CY_2C &=& Z_1Y_2 & CZ_2C &=& Z_1Z_2 \\ R_{z,1}(\theta)C &=& CR_{z,1}(\theta) & R_{x,2}(\theta)C &=& CR_{x,2}(\theta) \end{array}$$

For $i,j = 1,2,3$, $\sigma_j\sigma_k = \delta_{jk}I + i\sum_{l=1}^{3}\epsilon_{jkl}\sigma_l$ where ϵ_{jkl} is the antisymmetric tensor on 3 indices.[51]

Homework C.7. *Check these identities using the matrix form and the operator form to gain mastery of these calculations.*

C.1.11. *Measuring the Qubits.* The final operation we need to understand about qubits is, how can we get information back out of them? The process is called measurement, and there are several equivalent ways to think about it. We will cover the easiest to understand, intuitively and mathematically. However, to gain the precise control over measurements, we will have to resort later to an equivalent, yet more complicated, measurement framework.

Quantum Mechanics Postulate 3: State Measurement Quantum measurements are described by a collection $\{M_m\}$ of *measurement operators*. These are operators acting on the state space of a system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the system is $|\psi\rangle$ immediately before the measurement, then the probability that result m occurs is given by

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle \tag{225}$$

and the state of the system after the measurement is

$$\frac{M_m|\psi\rangle}{\sqrt{p(m)}} \tag{226}$$

The measurement operators satisfy the *completeness equation*

$$\sum_m M_m^\dagger M_m = I \tag{227}$$

[51] Exercise 2.43 in Neilsen and Chuang. All of these identities appear in the book, as exercises or in the text.

Finally, note cascaded measurements are single measurements. Thus if your algorithm calls for a succession of measurements, this is equivalent to a single measurement.

C.1.12. *Combining States and Partial States.* **Quantum Mechanics Postulate 4: State Combining** The state space of a composite physical system is the tensor product of the state spaces of the component systems. Moreover, if we have systems numbered 1 through n, and system number j is prepared in the state $|\psi_j\rangle$, then the joint state of the total system is $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \ldots |\psi_n\rangle$.

And that is all there is to quantum mechanics (as far as we are concerned). These four postulates form the basis of all that is known about quantum mechanics, a physical theory that has stood for over seven decades, and is used to explain phenomena at many scales.

However, quantum mechanics does not mesh well with the other main intellectual achievement in theoretical physics in the 20th century, relativity. Combining these two theories into a unified framework has occupied the best minds for over 50 years, and currently superstring theory is the best candidate for this unification.

Using the above postulates gives us an important theorem from Wootters and Zurek [127]:

C.1.13. *The No Cloning Theorem.*

Theorem C.13. ***The No Cloning Theorem.*** *It is impossible to build a machine that can clone any given quantum state.*

This is in stark contrast to the classical case, where we copy information all the time.

Proof. Suppose we have a machine with two slots: A for the quantum state $|\psi\rangle$ to be cloned, and B in some fixed initial state $|s\rangle$, and the machine makes a copy of the quantum state A. By the rules of quantum mechanics, the evolution U is unitary, so we have

$$|\psi\rangle \otimes |s\rangle \xrightarrow{U} |\psi\rangle \otimes |\psi\rangle \tag{228}$$

Now suppose we have two states we wish to clone, $|\psi\rangle$ and $|\varphi\rangle$, giving

$$\begin{aligned} U\left(|\psi\rangle \otimes |s\rangle\right) &= |\psi\rangle \otimes |\psi\rangle \\ U\left(|\varphi\rangle \otimes |s\rangle\right) &= |\varphi\rangle \otimes |\varphi\rangle \end{aligned}$$

Taking the inner product of these two equations, and using $U^\dagger U + =$:

$$\begin{aligned} \left(\langle\varphi| \otimes \langle s|\right) U^\dagger U \left(|\psi\rangle \otimes |s\rangle\right) &= \left(\langle\varphi| \otimes \langle\varphi|\right)\left(|\psi\rangle \otimes |\psi\rangle\right) \\ \langle\varphi|\psi\rangle\langle s|s\rangle &= \langle\varphi|\psi\rangle\langle\varphi|\psi\rangle \\ \langle\varphi|\psi\rangle &= \left(\langle\varphi|\psi\rangle\right)^2 \end{aligned}$$

This has solutions if and only if $\langle\varphi|\psi\rangle$ is 0 or 1, so cloning cannot be done for general states.[52]

□

This ends the quantum mechanics for quantum computing primer.

[52]There is a lot of research on what can be cloned, how much information can be cloned, etc.

Appendix D. Random Group Generation

This section is derived from Igor Pak's online lecture notes [103]. The point of this section is to prove

Theorem D.1. *Let G be a finite group. For an integer $t \geq 0$, the probability that $t + \lceil \log |G| \rceil$ elements chosen uniformly at random from G will generate G is bounded by*

$$\text{prob}\{\langle g_1, g_2, \ldots, g_{t+\lceil \log |G| \rceil} \rangle = G\} \geq 1 - \frac{1}{2^t} \text{ for } t \geq 0 \tag{229}$$

We will need some preliminaries to prove this. The idea will be to bound the number of elements that should generate G by the number needed by the "hardest" to generate group, which can be shown to be $\mathbb{Z}_2^r$, and then estimate how many elements are needed to generate the latter group. First some notation:

Definition D.2.
Given a finite group G, and elements $g_1, g_2, \ldots, g_t$ chosen uniformly at random from G, denote the probability that the g_i generate G by

$$\psi_t(G) = \text{prob}\{\langle g_1, g_2, \ldots, g_t \rangle = G\}.$$

First a reduction to a simpler group:

Lemma D.3. *Let $|G| \leq 2^r$, $r \geq 1$. Then for all $t \geq 1$, $\psi_t(G) \geq \psi_t(\mathbb{Z}_2^r)$, where $\mathbb{Z}_2^r$ is the additive group of binary r-tuples.*

Proof. Fix t and a subgroup $H \subsetneq G$. For a given sequence $g_1, g_2, \ldots, g_t$ of G, define subgroups H_j of G as $H_1 = \langle g_1 \rangle$, $H_2 = \langle g_1, g_2 \rangle$, $H_3 = \langle g_1, g_2, g_3 \rangle$, etc. Let H'_j be the similarly defined subgroups of $\mathbb{Z}_2^r$. Let $\tau_1, \tau_2, \ldots, \tau_L$ be the indices j where $H_j \neq H_{j-1}$, and define similarly $\tau'_1, \tau'_2, \ldots, \tau'_R$ for the H'_j. We will induct on $|G|$. When $|G| = 1$, the theorem is true. Let $s = \tau_{L-1}$. We compute

$$\begin{aligned}
\text{prob}\left(\tau_L - \tau_{L-1} \leq t \;\middle|\; H_s = H\right) &= 1 - \left(\frac{|H|}{|G|}\right)^t \\
&\geq 1 - \frac{1}{2^t} \\
&= 1 - \text{prob}\left(\tau'_R - \tau'_{R-1} > t\right) \\
&= \text{prob}\left(\tau'_R - \tau'_{R-1} \leq t\right)
\end{aligned}$$

This, combined with the induction assumption $\text{prob}\left(\tau_{L-1} \leq t\right) \geq \text{prob}\left(\tau'_{R-1} \leq t\right)$, gives

$$\text{prob}\left(\tau_L \leq t | H_s = H\right) \geq \text{prob}\left(\tau'_R \leq t\right) = \psi_t\left(\mathbb{Z}_2^r\right) \tag{230}$$

This holds for any fixed t and H, so the theorem follows. □

Lemma D.4. [53]

$$\psi_{r+t}(\mathbb{Z}_2^r) \geq 1 - \frac{1}{2^t} \text{ for } t \geq 0$$

[53] The article [103] proved a stronger form, but this is sufficient for our purposes.

Proof. View $\mathbb{Z}_2^r$ as the r dimensional vector space over the 2 element field $\mathbb{Z}_2$. Then $\psi_{r+t}(\mathbb{Z}_2^r)$ is the probability that $r+t$ randomly chosen vectors spans the entire r dimensional space $\mathbb{Z}_2^r$. If we write the $r+t$ vectors as rows of a $(r+t)\times r$ matrix, then this is the probability that the matrix has column rank r. This happens if and only if all r columns are linearly independent.

The first column (which has $r+t$ entries) is nonzero with probability $\left(1-\frac{1}{2^{r+t}}\right)$. The probability that the second column is linearly independent of the first is $\left(1-\frac{1}{2^{r+t-1}}\right)$, and so on. Thus for $t\geq 0$ we get that

$$\begin{aligned}
\psi_{r+t}(\mathbb{Z}_2^r) &= \left(1-\frac{1}{2^{t+r}}\right)\left(1-\frac{1}{2^{t+r-1}}\right)\cdots\left(1-\frac{1}{2^{t+1}}\right)\\
&= 1-\frac{1}{2^t}\sum_{a=1}^{r}\frac{1}{2^a}+\frac{1}{4^t}\sum_{\substack{a,b=1\\a\neq b}}^{r}\frac{1}{2^a}\frac{1}{2^b}-\frac{1}{8^t}\sum_{\substack{a,b,c=1\\a\neq b\neq c}}^{r}\frac{1}{2^{a+b+c}}+\cdots\\
&= 1-\frac{1}{2^t}\left(1-\frac{1}{2^r}\right)+\frac{1}{4^t}\sum_{\substack{a,b=1\\a\neq b}}^{r}\left(\frac{1}{2^{a+b}}-\sum_{\substack{c=1\\c\neq a\neq b}}^{r}\frac{1}{2^{a+b+c}}\right)+\cdots\\
&\geq 1-\frac{1}{2^t}+\frac{1}{4^t}\sum_{a\neq b}\left(\frac{1}{2^{a+b}}-\sum_{c=1}^{r}\frac{1}{2^{a+b+c}}\right)+\cdots\\
&= 1-\frac{1}{2^t}+\frac{1}{4^t}\sum_{a\neq b}\left(\frac{1}{2^{a+b}}\left(1-\sum_{c=1}^{r}\frac{1}{2^c}\right)\right)+\cdots\\
&\geq 1-\frac{1}{2^t}
\end{aligned}$$

Note that in the lines above that the ellipses denotes a finite number of terms, which can be paired up similarly to the two terms shown, with at most one final positive term which can then be dropped in the inequality. □

Now we prove theorem D.1.

Proof. Set $r=\lceil\log|G|\rceil$, giving $|G|\leq 2^r$. Then for $t\geq 0$ we have $\psi_{t+r}(G)\geq\psi_{t+r}(\mathbb{Z}_2^r)$ by lemma D.3, and then this is $\geq 1-\frac{1}{2^t}$ by lemma D.4, which proves theorem D.1. □

Finally, note there are much better bounds, but this one gives the exponential performance we need for our purposes.

Appendix E. GCD Probabilities

This appendix shows the proof that the probability of the GCD of integers uniformly sampled from a fixed range becomes exponentially close to 1 in terms of the number of samples. The formal result is lemma E.3.

Unfortunately we need the next result without proof to start off the result.

Lemma E.1 ([121]). *Let $\varphi(n)$ be the Euler totient function*[54]*. Then for any positive integer n,*

$$\left|\sum_{c=1}^{n}\varphi(c)-\frac{3n^2}{\pi^2}\right|<n\ln n \tag{231}$$

where $\ln n$ is log base e.

Lemma E.2. *Fix an integer $n>0$. Choose two nonnegative integers $a,b\le n$ uniformly at random. Then the probability that $\gcd(a,b)=1$ is $\ge\frac{1}{2}$.*

Proof. Given the uniformly randomly chosen integers a,b, the probability that $\max\{a,b\}=c$ is $\frac{2c+1}{(n+1)^2}$. This can be seen by looking at a matrix with a_{ij} entry (i,j), and counting elements, for $i,j\in\{0,1,\ldots,n\}$. Assuming $c>0$, which happens with probability $p_0=\frac{(n+1)^2-1}{(n+1)^2}$, the probability that the second integer is relatively prime to the largest one c is precisely $\frac{\varphi(c)}{c}$. So the probability p_n that $\gcd(a,b)=1$ is exactly

$$p_n = p_0\sum_{c=1}^{n}\frac{2c+1}{(n+1)^2}\frac{\varphi(c)}{c} \tag{232}$$

$$= \frac{n^2+2n}{(n+1)^4}\sum_{c=1}^{n}\left(2+\frac{1}{c}\right)\varphi(c) \tag{233}$$

$$\ge \frac{2n^2+4n}{(n+1)^4}\sum_{c=1}^{n}\varphi(c) \tag{234}$$

By lemma E.1 $\sum\varphi(c)>\frac{3n^2}{\pi^2}-n\log n$, giving

$$p_n\ge\left(\frac{2n^2+4n}{(n+1)^4}\right)\left(\frac{3n^2-\pi^2 n\log n}{\pi^2}\right) \tag{235}$$

Denoting the right hand side by $f(n)$, it is easy to check f is increasing[55] for $n\ge4$ and that $f(94)>0.5$, proving the proposition for integers $n\ge94$. The remaining cases $n=1,2,\ldots,93$ can be easily (yet tediously) checked using equation 233. I recommend Mathematica or Maple. □

Lemma E.3. *Suppose we have $k\ge2$ uniformly random samples $t_1,t_2,\ldots,t_k$ from the integers $\{0,1,\ldots,d-1\}$ for an integer $d\ge2$. Then*

$$\text{prob}\left(\gcd(t_1,t_2,\ldots,t_k)=1\right)\ge1-\left(\frac{1}{2}\right)^{k/2}$$

Proof. Consider the samples taken as pairs. Certainly if any pair t_{2j-1} and t_{2j} are relatively prime, then $\gcd(t_1,t_2,\ldots,t_k)=1$. By lemma E.2 the probability that $\gcd(t_{2j-1},t_{2j})>1$ is $\le\frac{1}{2}$, so the probability that every such pair, $j=1,2,\ldots,\lfloor k/2\rfloor$, has gcd >1 is $\le\left(\frac{1}{2}\right)^{\lfloor k/2\rfloor}\le\left(\frac{1}{2}\right)^{k/2}$. Thus the probability that $\gcd(t_1,t_2,\ldots,t_k)=1$ is $\ge1-\left(\frac{1}{2}\right)^{k/2}$. □

[54] For a positive integer n, $\varphi(n)$ returns the number of positive integers less than n and relatively prime to n.

[55] $\lim_{n\to\infty}f(n)=6/\pi^2$, agreeing with Dirichlet's 1849 theorem to that effect.

Finally we note that the above estimates and probabilities are very conservative, yet yield the essential fact that the probability of success increases exponentially with the number of trials.

References

1. S. Aaronson, *Quantum lower bound for the collision problem*, STOC' 02 (Montreal, Quebec, Canada), 19-21 May 2002, quant-ph/0111102.
2. Daniel S. Abrams and Seth Lloyd, *Nonlinear quantum mechanics implies polynomial-time solution for NP-complete and #P problems*, Phys.Rev.Lett. **81** (1998), 3992–3995.
3. Dorit Aharonov, Win van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev, *Adiabatic quantum computation is equivalent to standard quantum computation*, 2004, quant-ph/0405098.
4. L. Babai and R. Beals, *Las Vegas algorithms for matrix groups*, Proc. 34th IEEE Foundations of Computer Science, 1993, pp. 427–436.
5. L. Babai and L. Rónyai, *Computing irreducible representations of finite groups*, Math. Comp. **55** (1990), 705–722.
6. L. Babai and E. Szemeréedi, *On the complexity of matrix group problems I*, Proceedings of the 25th Annual Symposium on Foundations of Computer Science, 1984, pp. 229–240.
7. Adriano Barenco, *Quantum physics and computers*, Contemporary Physics **38** (1996), 357–389.
8. Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John Smolin, and Harald Weinfurter, *Elementary gates for quantum computation*, Physical Review **52** (1995), no. 5, 3457–3467, also at quant-ph/9503016 v1.
9. U. Baum, *Existence and efficient construction of fast Fourier transforms for supersolvable groups*, Comput. Complexity **1** (1991), 235–256.
10. U. Baum and M. Clausen, *Some lower and upper complexity bounds for generalized Fourier transforms and their inverses*, SIAM J. of Comput. **20** (1991), no. 3, 451–459.
11. ———, *Fast Fourier transforms*, BI-Verlag, 1993.
12. ———, *Fast Fourier transforms for symmetric groups, theory and implementations*, Math. Comp. **61** (1993), no. 204, 833–847.
13. U. Baum, M. Clausen, and B. Tietz, *Improved upper complexity bounds for the discrete Fourier transform*, AAECC **2** (1991), 35–43.
14. R. Beals, *Quantum computation of Fourier transforms over symmetric groups*, Proc. 29th Ann. ACM Symp. Theory of Computation (El Paso, Texas), ACM Press, 4-6 May 1997, pp. 48–53.
15. P.W. Beame, S.A. Cook, and H.J. Hoover, *Log depth circuits for division and related problems*, Proceedings of the 25th Annual Symposium on Foundations of Computer Science, 1984, pp. 1–6.
16. Charles H. Bennett, *Logical reversibility of computation*, IBM J. of Research and Development **17** (1973), 525–532.
17. Charles H. Bennett, Ethan Bernstein, Giles Brassard, and Umesh Vazirani, *Strengths and weaknesses of quantum computing*, SIAM Journal on Computing **26** (1997), no. 5, 1510–1523.
18. Ethan Bernstein and Umesh Vazirani, *Quantum complexity theory*, SIAM Journal on Computing **26** (1997), no. 5, 1411–1473.
19. Thomas Beth, *On the computational complexity of the general discrete Fourier transform*, Theor. Comp. Sci. **51** (1987), no. 3, 331–339.
20. Thomas Beth, Markus Püschel, and Martin Rötteler, *Fast quantum Fourier transforms for a class of non-abelian groups*, Proc. of Applied Algebra Algebraic Algorithms, and Error-Correction Codes (AAECC-13), Springer-Verlag, 1999, volume 1719 in Lecture Notes in Computer Science, pp. 148–159.
21. M. Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp, *Tight bounds on quantum searching*, Proc. 4th Workshop on Physics and Computation-PhysComp, 1996, quant-ph/9605034, pp. 36–43.
22. Robert S. Boyer and J. Strother Moore, *A fast string-searching algorithm*, Communications of the ACM **20** (1977), no. 10, 762–772.

23. Gilles Brassard and Peter Høyer, *An exact polynomial-time algorithm for Simon's problem*, Proc. 5th Israeli Symposium on Theory of Computing and Systems, IEEE Computer Society Press, 1997, quant-ph/9704027, pp. 12–33.
24. Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp, *Quantum amplitude amplification*, 2000, quant-ph/0005055.
25. Gilles Brassard, Peter Høyer, and Alain Tapp, *Quantum counting*, Lecture Notes in Computer Science **1443** (1998), 820+, quant-ph/9805082.
26. A. Robert Calderbank, Eric M. Raines, Peter W. Shor, and Neil J. Sloane, *Quantum error correction and orthogonal geometry*, Physical Review Letters (1997), to appear.
27. A. Robert Calderbank and Peter W. Shor, *Good quantum error-correcting codes exist*, Physical Review A **54** (1996), 1098–1106.
28. Kevin K. H. Cheung and Michele Mosca, *Decomposing finite abelian groups*, J. Quantum Inf. Comp. **1** (2001), no. 3, 26–32, quant-ph/0101004.
29. I. L. Chuang and M. A. Nielsen, *Quantum computation and quantum information*, Cambridge University Press, Cambridge, 2000.
30. Michael Clausen, *Fast generalized Fourier transforms*, Thoeret. Comput. Sci. **67** (1989), no. 1, 55–63.
31. R. Cleve, *An introduction to quantum complexity theory*, 1999, http://www.cpsc.ucalgary.ca/∼cleve/papers.html.
32. R. Cleve, E. Ekert, C. Macchiavello, and M. Mosca, *Quantum algorithms revisited*, Proc. Roy. Soc. Lond. A **454** (1998), 339–354.
33. R. Cleve and J. Watrous, *Fast parallel circuits for the quantum Fourier transform*, Proceedings of the 41st Annual Symposium on Foundations of Computer Science, vol. 454, 2000, http://www.cpsc.ucalgary.edu/∼jwatrous/papers/qft.ps, pp. 526–536.
34. James W. Cooley and John W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comput. **19** (1965), 297–301.
35. D. Coppersmith, *An approximate Fourier transform useful in quantum computing*, IBM Technical Report RC 19642 (1994), quant-ph/0201067.
36. Ivan Damgård, *QIP Note: on the quantum Fourier transform and applications*, 2001, http://www.daimi.au.dk/∼ivan/fourier.ps.
37. David Deutsch, *Quantum theory, the Church-Turing principle and the universal quantum computer*, Proc. Roy. Soc. Lond. A **400** (1985), 97–117.
38. David Deutsch and Richard Jozsa, *Rapid solution of problems by quantum computation*, Proc. Roy. Soc. Lond. A **439** (1992), 553–558.
39. Persi Diaconis and Daniel Rockmore, *Efficient computation of the Fourier transform on finite groups*, J. Amer. Math. Soc. **3** (1990), no. 2, 297–332.
40. C. Dürr and P. Høyer, *A quantum algorithm for finding the minimum*, 1996, quant-ph/9607014.
41. S. Egner and M. Püschel, *AREP - a package for constructive representation theory*, 1998.
42. A. Ekert and R. Jozsa, *Quantum computation and Shor's factoring algorithm*, Rev. Modern Physics **68** (July 1996), no. 3, 733.
43. Mark Ettinger and Peter Høyer, *A quantum observable for the graph isomorphism problem*, 1999, quant-ph/9901029.
44. ———, *Quantum state detection via elimination*, 1999, quant-ph/9905099.
45. ———, *On quantum algorithms for noncommutative hidden subgroups*, Advances in Applied Mathematics **25** (2000), 239–251.
46. Mark Ettinger, Peter Høyer, and E. Knill, *Hidden subgroup states are almost orthogonal*, 1999, quant-ph/9901034.
47. ———, *The quantum query complexity of the hidden subgroup problem is polynomial*, Information Processing Letters **91** (2004), no. 1, 43–48, quant-ph/0401083.
48. Stephen Fenner and Yong Zhang, *Quantum algorithms for a set of group theoretic problems*, 2004, quant-ph/0408150.
49. Katalin Friedl, Gábor Ivanyos, Frédéroc Magniez, Miklos Santha, and Pranab Sen, *Hidden translation and orbit coset in quantum computing*, Proc. 35th ACM Symp. on Theory of Computing, 2003, pp. 1–9.
50. A. Galindo and M. A. Matrín-Delgado, *Information and computation: Classical and quantum apsects*, 2001, quant-ph/0112105.

51. Dmitry Gavinsky, *Quantum solution to the hidden subgroup problem for poly-near-hamiltonian groups*, Quantum Information and Computation **4** (2004).
52. Robert B. Griffiths and Chi-Sheng Niu, *Semiclassical Fourier transform for quantum computation*, 1995, quant-ph/9511007.
53. M. Grigni, L. J. Schulman, M. Vazirani, and U. V. Vazirani, *Quantum mechanical algorithms for the nonabelian hidden subgroup problem*, Proc. 33rd ACM Symp. on Theory of Computing, 2001, pp. 68–74.
54. L. K. Grover, *A fast quantum mechanical algorithm for database search*, Proc. 28th Ann. ACM Symp. on Theory of Comput., 1996, pp. 212–219.
55. Lisa Hales, *The quantum Fourier transform and extensions of the abelian subgroup problem*, Ph.D. thesis, University of California at Berkeley, Berkeley, CA, 2002, quant-ph/0212002.
56. Lisa Hales and Sean Hallgren, *Quantum Fourier sampling simplified*, Proc. 31st Ann. ACM Symp. on Theory of Comput., 1999, Atlanta, Georgia, 1-4 May, pp. 330–338.
57. ———, *An improved quantum Fourier transform algorithm and applications*, Proc. 41st Ann. Symp. on Foundations of Computer Science, 2000, Redonda Beach, California, 12-14 November, pp. 515–525.
58. Sean Hallgren, *Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem*, Proc. 34th Ann. Symp. on Foundations of Computer Science, 2002, Montreal, Quebec, Canada, 19-21 May, pp. 515–525.
59. Sean Hallgren, A. Russell, and A. Ta-Shma, *Normal subgroup reconstruction and quantum computing using group representations*, Proc. 32nd Ann. ACM Symp. Theory of Computing (New York, NY), ACM Press, 2000, Portland, Oregon, 21-23 May, pp. 627–635.
60. Sean Hallgren and Wim van Dam, *Efficient quantum algorithms for shifted quadratic character problems*, 2000, quant-ph/0011067.
61. Sean Hallgren, Wim van Dam, and Lawrence Ip, *Quantum algorithms for hidden coset problems*, unpublished.
62. ———, *Quantum algorithms for some hidden shift problems*, ACM-SIAM Symposium on Discrete Algorithms (SODA) (2003), to appear.
63. Joe Harris and William Fulton, *Representation theory*, no. 129 in Graduate Texts in Mathematics, Springer-Verlag, New York, NY, 1991.
64. Peter Høyer, *Efficient quantum transforms*, 1997, quant-ph/9702028.
65. ———, *Simplified proof of the Fourier sampling theorem*, Information Processing Letters **75** (2000), no. 4, 139–143.
66. Gábor Ivanyos, Frédéroc Magniez, and Miklos Santha, *Efficient quantum algorithms for some instances of the non-abelian hidden subgroup problem*, Proc. 13th Ann. ACM Symp. on Parallel Algorithms and Architectures (New York, NY), ACM Press, 2001, Heraklion, Crete Island, Greece, 4-6 July, pp. 263–270.
67. G. James and Adalbert Kerber, *The representation theory of the symmetric group*, Cambridge University Press, 1982.
68. R. Jozsa, *Quantum algorithms and the Fourier transform*, Proc. Royal Soc. London Series A **454** (1998), no. 1969, 323–337.
69. ———, *Quantum factoring, discrete logarithms and the hidden subgroup problem*, 2000, quant-ph/0012084.
70. ———, *Notes on Hallgren's efficient quantum algorithm for solving Pell's equation*, 2003, quant-ph/0302134.
71. H. W. Lenstra Jr. and C. Pomerance, *A rigorous time bound for factoring integers*, Journal of the AMS **5** (1992), no. 2, 483–516.
72. M. Karpovsky, *Fast Fourier transforms on finite abelian groups*, IEEE Trans. Comput. **26** (1977), no. 10, 1028–1030.
73. Julia Kempe, *Quantum random walks hit exponentially faster*, 2002, quant-ph/0205083.
74. Julia Kempe and Aner Shalev, *The hidden subgroup problem and permutation group theory*, 2004, quant-ph/0406046.
75. Adalbert Kerber, *Representations of permutation groups I*, vol. Lecture Notes in Mathematics 240, Springer-Verlag, Berlin, 1971.
76. ———, *Representations of permutation groups II*, vol. Lecture Notes in Mathematics 495, Springer-Verlag, Berlin, 1975.
77. Alexi Yu. Kitaev, *Quantum measurements and the Abelian stabilizer problem*, 1995, quant-ph/9511026.

78. _____, *Quantum computations: algorithms and error correction*, Russ. Math. Surv. **52** (1997), no. 6, 1191–1249.
79. Donald Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms (3rd Edition)*, Addison-Wesley, 1997, ISBN: 0201896842.
80. Donald Knuth, James Morris, and Vaughan Pratt, *Fast pattern matching in strings*, SIAM Journal on Computing **6** (1977), no. 2, 323–350.
81. Johannes Köbler, Uwe Schöning, and Jacobo Torán, *The graph isomorphism problem: Its structural complexity*, Birkhauser Boston Inc., Boston, MA, 1993.
82. Greg Kuperberg, *A subexponential-time algorithm for the dihedral hidden subgroup problem*, 2003, quant-ph/0302112.
83. Serge Lang, *Algebra*, Addison-Wesley Publishing, 1993, ISBN 0-201-55540-9.
84. S. J. Lomonaco and L.H. Kauffman, *Quantum hidden subgroup problems: A mathematical perspective*, 2002, quant-ph/0201095.
85. _____, *Continuous quantum hidden subgroup algorithms*, 2003, quant-ph/0304084.
86. Samuel J. Lomonaco (ed.), *Quantum computation: A grand mathematical challenge for the twenty-first century and the millenium*, AMS, Providence, RI., 2002, PSAPM 58.
87. Chris Lomont, *Quantum convolution and quantum correlation algorithms are physically impossible*, 2003, quant-ph/0309070.
88. _____, *A quantum Fourier transform algorithm*, 2004, quant-ph/0404060.
89. David K. Maslen and Daniel N. Rockmore, *Separation of variables and the efficient computation of Fourier transforms on finite groups, II*, in preparation.
90. _____, *Adapted diameters and the efficient computation of Fourier transforms on finite groups*, Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, California, 22-24 January), 1995, pp. 253–262.
91. _____, *Separation of variables and the computation of Fourier transforms on finite groups, I*, J. Amer. Math. Soc. **10** (1997), no. 1, 169–214.
92. _____, *Generalized FFT's: A survey of some recent results*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 28, ACM, 1997(1995), June 7-10, pp. 183–237.
93. _____, *The Cooley-Tukey FFT and group theory*, Notices Amer. Math. Soc. **48** (2001), no. 10, 1151–1160.
94. Rudolf Mathon, *A note on the graph isomorphism problem*, Information Processing Letters **8** (1979), 131–132.
95. G. L. Miller, *Graph Isomorphism, general remarks*, Journal of Computer and System Sciences **18** (1979), 128–142.
96. Christopher Moore, Daniel Rockmore, and Alexander Russell, *Generic quantum FFTs*, http://www.cs.dartmouth.edu/∼rockmore/qfftcamera.pdf, 2004, SODA 2004, to appear.
97. Christopher Moore, Daniel Rockmore, Alexander Russell, and Leonard Schulman, *The hidden subgroup problem in affine groups: Basis selection in Fourier sampling*, quant-ph/0211124, 2002, SODA 2004, to appear.
98. Michele Mosca, *Quantum computer algorithms*, Ph.D. thesis, Wolfson College, University of Oxford, Oxford, United Kingdom, 1999, www.cacr.math.uwaterloo.ca/∼mmosca/moscathesis.ps.
99. Michele Mosca and Artur Ekert, *The hidden subgroup problem and eigenvalue estimation on a quantum computer*, QCQS: NASA International Conference on Quantum Computing and Quantum Communications, LNCS, 1998.
100. Michele Mosca and Christof Zalka, *Exact quantum Fourier transforms and discrete logarithm algorithms*, 2003, quant-ph/0301093.
101. Román Orús, José I. Latorre, and Miguel A. Martín-Delgado, *Natural majorization of the quantum Fourier transformation in phase-estimation algorithms*, 2003, quant-ph/0206134.
102. _____, *Systematic analysis of majorization in quantum algorithms*, 2003, quant-ph/0212094.
103. Igor Pak, *18.317 combinatorics, probability and computations on groups (fall 2001)*, 2001, http://www-math.mit.edu/∼pak/courses/pg.html.
104. Arun K. Pati and Samuel L. Braustein, *Deutsch-Jozsa algorithm for continuous variables*, 2002, quant-ph/0207108.
105. Arun K. Pati, Samuel L. Braustein, and Seth Lloyd, *Quantum searching with continuous variables*, 2000, quant-ph/0002082.

106. H. Ramesh and V. Vinay, *String matching in $\tilde{O}(\sqrt{n}+\sqrt{m})$ quantum time*, Journal of Discrete Algorithms **2** (2001), no. 1, quant-ph/0011049.
107. O. Regev, *Quantum computation and lattice problems*, Proceedings of the 43rd Annual Symposium on Foundations of Computer Science (FOCS), 2002.
108. Daniel N. Rockmore, *Fast Fourier analysis for abelian group extensions*, Advances in Applied Mathematics **11** (1990), 164–204.
109. ———, *Efficient computation of Fourier inversion for finite groups*, J. of the ACM **41** (1994), no. 1, 31–66.
110. ———, *Fast Fourier transforms for wreath products*, J. Applied and Computational Harmonic Analysis **2** (1995), 279–292.
111. Joseph Rotman, *An introduction to the theory of groups*, vol. Number 148 in Graduate Texts in Mathematics, Springer-Verlag, 1995.
112. Martin Rötteler and Thomas Beth, *Polynomial time solution to the hidden subgroup problem for a class of non-abelian groups*, 1998, quant-ph/9812070.
113. Jean-Pierre Serre, *Linear representations on finite groups*, Springer-Verlag, 1977.
114. P. W. Shor, *Algorithms for quantum computation: discrete logarithms and factoring*, Proceedings, 35th Annual Symposium on Fundamentals of Comp. Science (FOCS), 1994, pp. 124–134.
115. ———, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Computing **26** (1997), no. 5, 1484–1509.
116. ———, *Introduction to quantum algorithms*, AMS PSAPM/58, 2002, pp. 124–134.
117. Barry Simon, *Representations of finite and compact groups*, vol. 10 in Graduate Studies in Mathematics, American Mathematical Society, 1996.
118. Daniel Simon, *On the power of quantum computation*, Proceedings of the 35th Annual Symposium on Foundations of Computer Science (Los Alamitos, CA), Institute of Electrical and Electronic Engineers Computer Society Press, 1994, citeseer.nj.nec.com/article/simon94power.html, pp. 116–123.
119. ———, *On the power of quantum computation*, SIAM J. Computing **26** (1997), no. 5, 1474–1483.
120. Arne Storjohann, *Near optimal algorithms for computing smith normal forms of integer matrices*, Proceedings of the 1996 international symposium on Symbolic and algebraic computation, ACM Press, 1996, pp. 267–274.
121. R. Tambs-Lyche, *Kongelige norske videnskabers skelskabs forhandlinger*, vol. 9, Trondhjem, Norway, 1936.
122. Audrey Terras, *Fourier analysis on finite goups and applications*, no. 43 in London Mathematical Society Student Texts, Cambridge University Press, 1999.
123. W. van Dam, Michele Mosca, and U. Vazirani, *How powerful is adiabatic quantum computation*, 2002, quant-ph/0206003.
124. Wim van Dam, *Quantum algorithms for weighting matrices and quadratic residues*, 2000, quant-ph/0008059.
125. J. Waltrous, *Succinct quantum proofs of properties of finite groups*, Proceedings of the 41st Annual Symposium on Foundations of Computer Science, 2000, pp. 537–546.
126. ———, *Quantum algorithms for solvable groups*, Proceedings of the 33rd ACM Symposium on Theory of Computing, 2001, pp. 60–67.
127. W. K. Wootters and W. H. Zurek, *A single quantum cannot be cloned*, Nature **299** (1982), 802–803.
128. Andrew Chi-Chih Yao, *Quantum circuit complexity*, Proc. 34th Ann. Symp. on Found. of Comp. Sci., 1996, pp. 352–361.
129. Christof Zalka, *On a particular non-abelian hidden subgroup problem*, 1999, http://qso.lanl.gov/∼zalka/QC/QC.html.